Docker容器虛擬化資安最佳化實務與應用

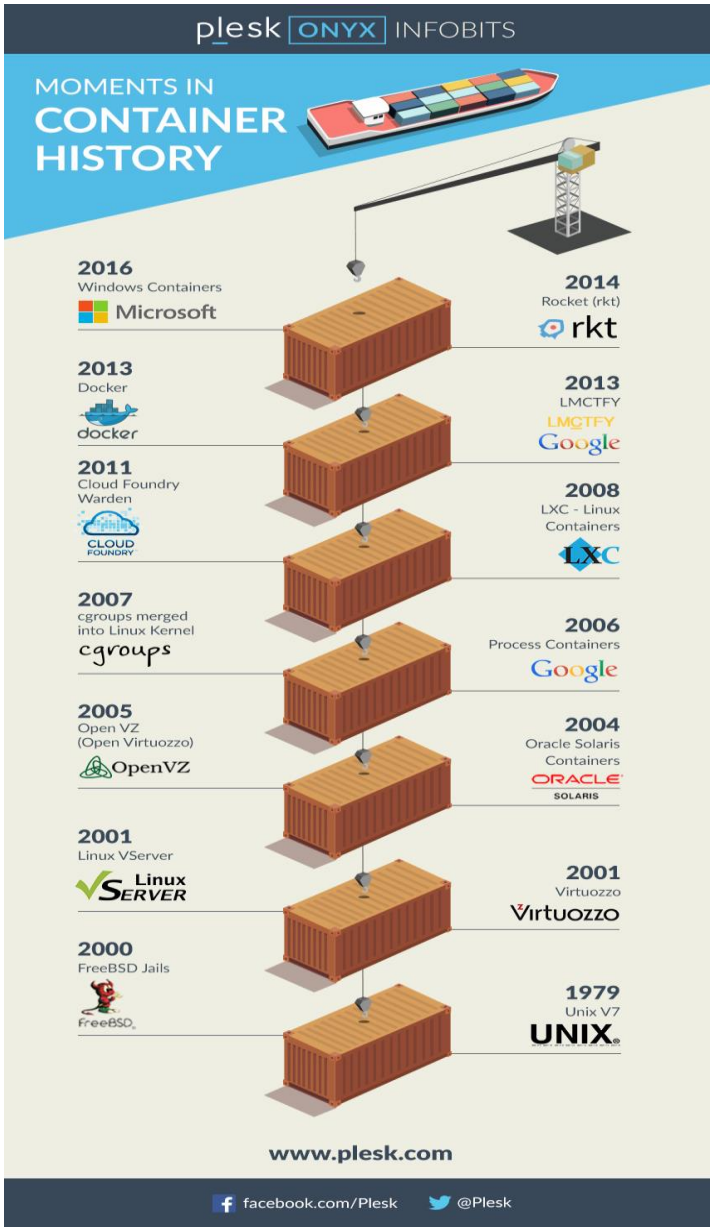
資安科技研究所/技術研發中心
財團法人資訊工業策進會

大綱

- 上午課程：
  - 容器虛擬化技術Docker 介紹
  - [Lab1] Github帳號申請、 AWS VM 申請
  - 容器虛擬化技術Docker 資安現況
  - 容器虛擬化技術Docker 資安因應實務建議
  - [Lab2] Docker 安裝與操作 part 1
  - DevOps 持續整合開發介紹
- 下午課程：
  - [Lab3] Docker 安裝與操作 part 2 、Hello World – Web程式開發、Jenkins安裝、設定、自動化deploy
  - [Lab4] Docker 安全設定實務、Clair 等Docker資安工具安裝與設定

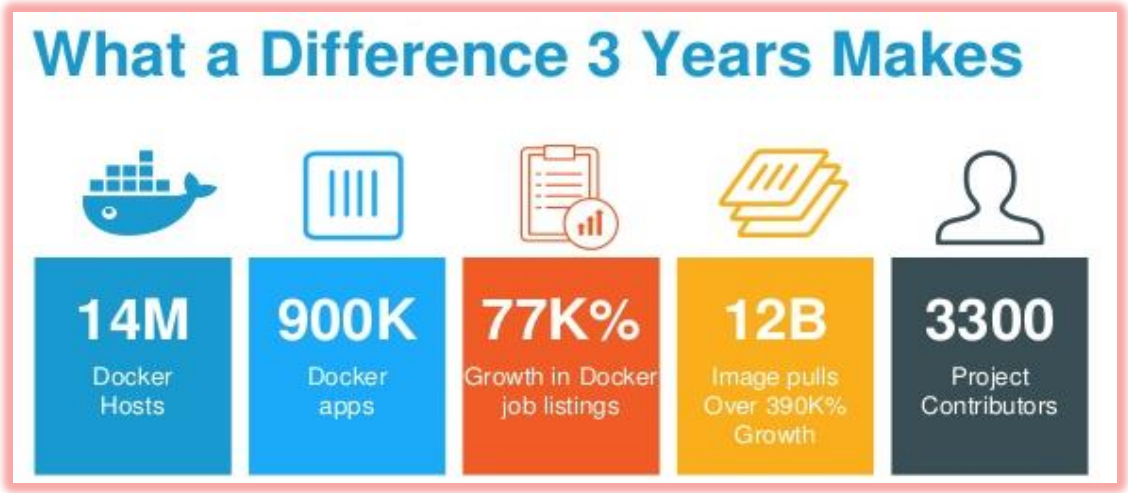
容器虛擬化技術Docker介紹

容器的歷史故事



資料來源：www.plesk.com

2017



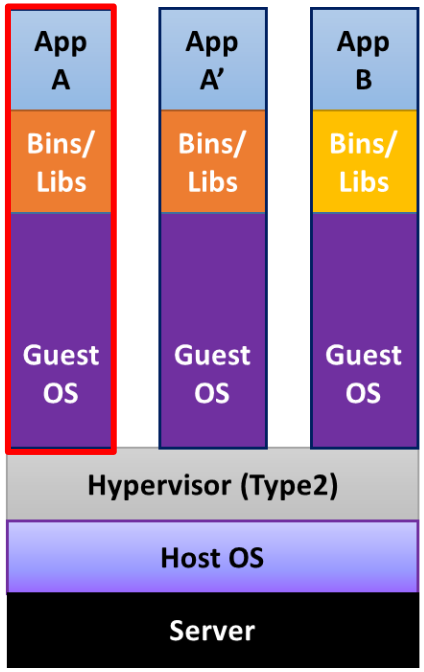
資料來源：Dockercon 2017

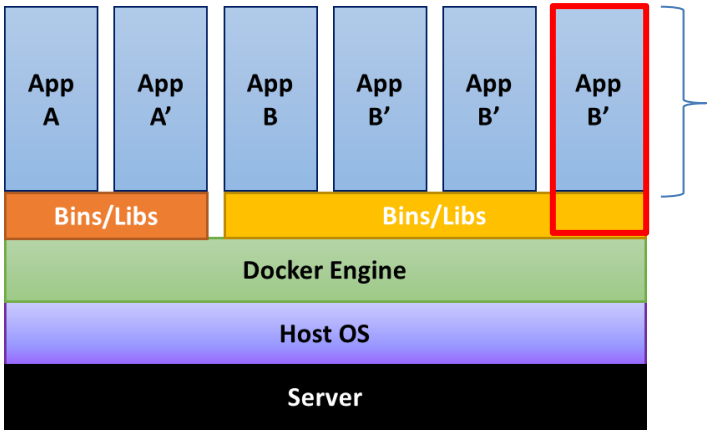
Docker簡介

- Docker是一個開源專案 (start from 2013)
  - 輕量級虛擬化技術
  - 基於Linux的LXC技術 (Cgroups / Namespace)
- 2017.04 Docker開源專案已經改名成Moby
  - 可以客製化自己的容器系統
  - Docker CE、Docker EE

Virtual Machine

包含應用程式、各自的Library及Guest OS

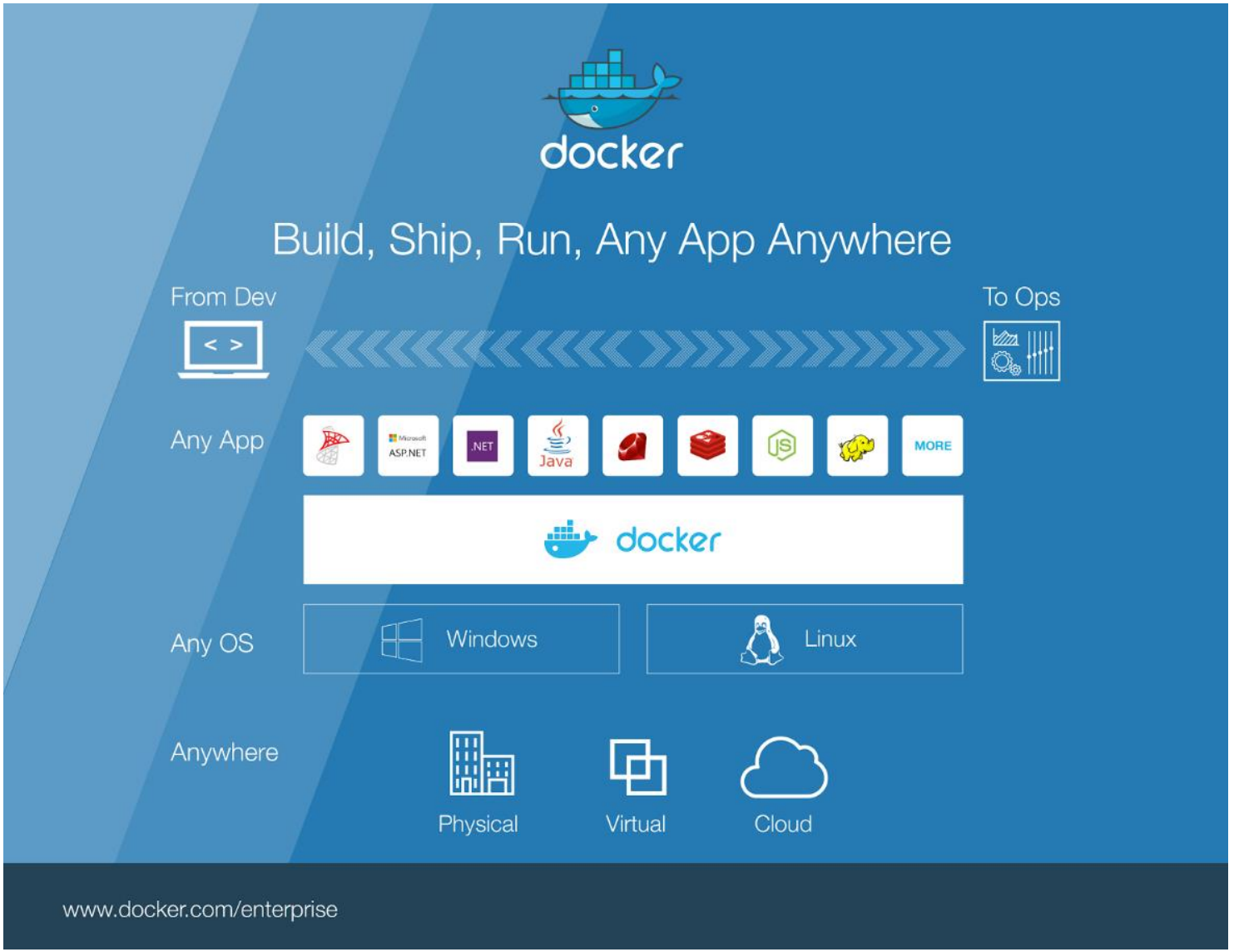




Docker

包含應用程式，及各自的Library、Guest OS

Docker 願景



資料來源：www.docker.com

Docker 優勢

更快速的交付和部署

- Developer快速建置開發環境
- DevOps可透過開發環境之容器快速部屬
- 節省開發、測試、部屬時間

更有效率的虛擬化

- 不需額外的虛擬化支援，它是核心層級(應用程式層)的虛擬化

更輕鬆的遷移和擴展

- 平台相容性佳，包含實體機器、虛擬機、個人電腦、私有雲等

更簡單的管理

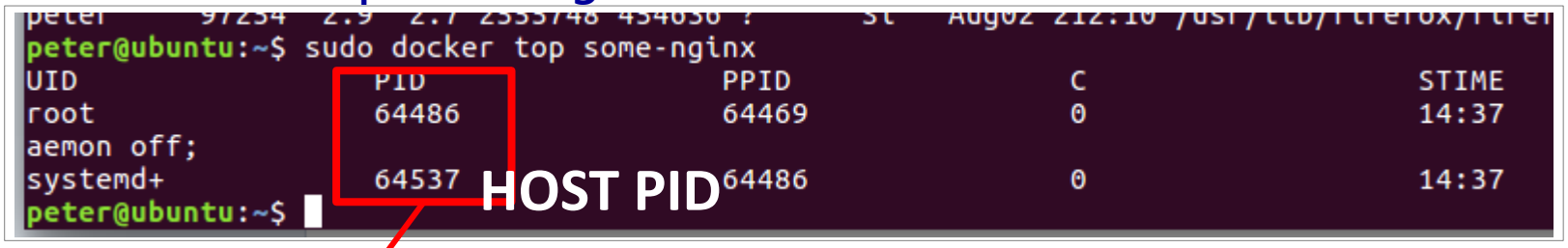
- 應用程式更新、環境建置與部屬管理、叢集(Cluster)管理

特性	容器	虛擬機
開機載入速度	秒級	分鐘級
硬碟容量	一般為MB	一般為GB
效能	接近原生	比較慢
系統支援量	單機支援上千個容器	一般十幾個

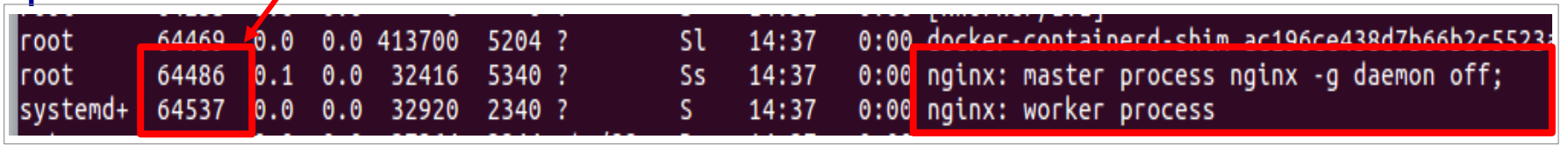
Container與VM之差異

- Container VS VM
  - VM 完全隔離的環境
    - Host 完全看不到VM內部的程序
  - Container 輕量化隔離環境
    - 算是Host OS 的一部分
      - 可以從Host上看到內部的程序

sudo docker top some-nginx指令

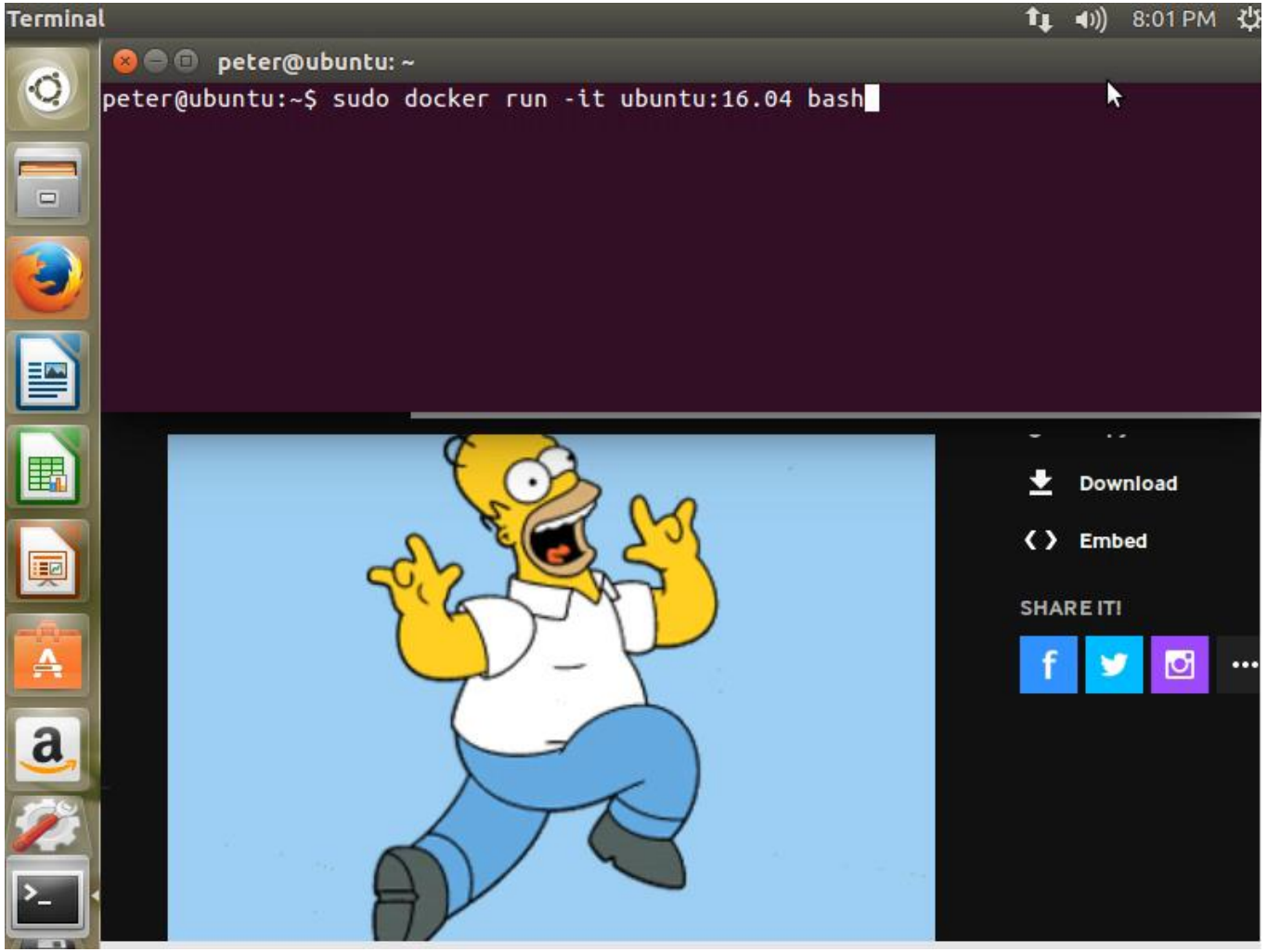


ps –aux 指令



為何要使用Docker (1)

- Fork Bomb：大量調用fork，進行的服務阻斷攻擊

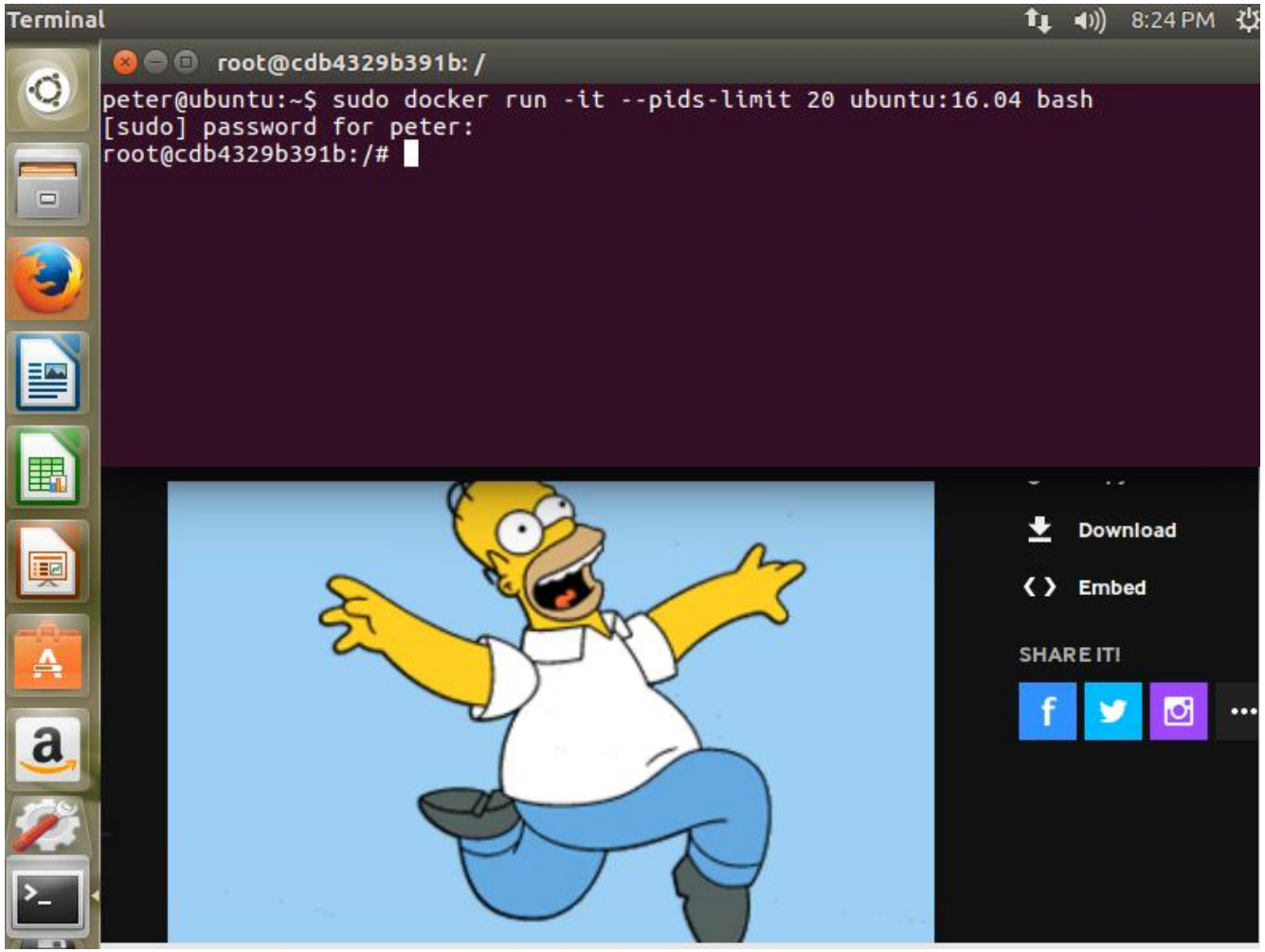


1. 啟動container (未設定任何參數)
2. 進入container，執行Fork Bomb攻擊
3. Host觀察圖片狀態

Host crush!!!!

為何要使用Docker (2)

- Container透過參數設定可提升安全性，抵擋Fork Bomb 攻擊



參數：
--pids-limit 20

Docker 基本概念

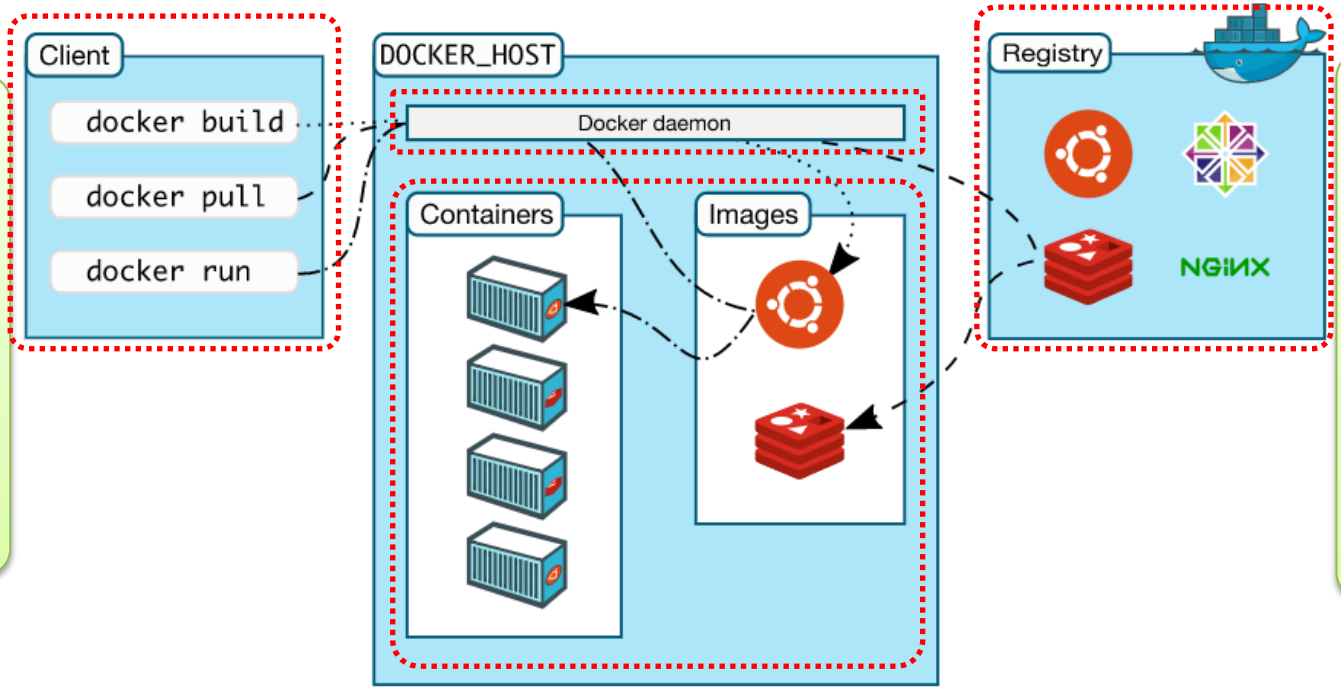
Docker 核心架構

Docker Deamon

等待Docker API之需求、管理Image/Container/Network/Volumes

Docker Client

利用Docker API與Docker Deamon溝通



Docker Registry

存放Docker Image，如：Docker Hub、Private Hub

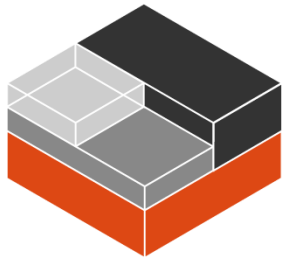
Docker Object

Image – 用來建裡Docker容器

Container- 執行的鏡像容器，可透過Docker API、 CLI

資料來源：Docker Doc

Docker核心原理：隔離性-基於LXC



- LXC (Linux Container)
  - **作業系統層虛擬化技術**（Operating system–level virtualization）：
    為Linux內核容器功能的一個用戶空間介面。
    將應用軟體系統打包成一軟體容器（Container），內含應用軟體的程式碼、所需的作業系統核心及函式庫
  - **硬體資源共享**：透過統一的命名空間和共用API來分配不同軟體容器的可用硬體資源，創造應用程式的獨立沙箱執行環境
  - **獨立環境**：利用cgroups與namespace功能，建立應用軟體一個獨立的作業系統環境
  - **不需要Hypervisor軟體層**：軟體容器（Container）本身極為輕量化，提升了建立虛擬機器的速度

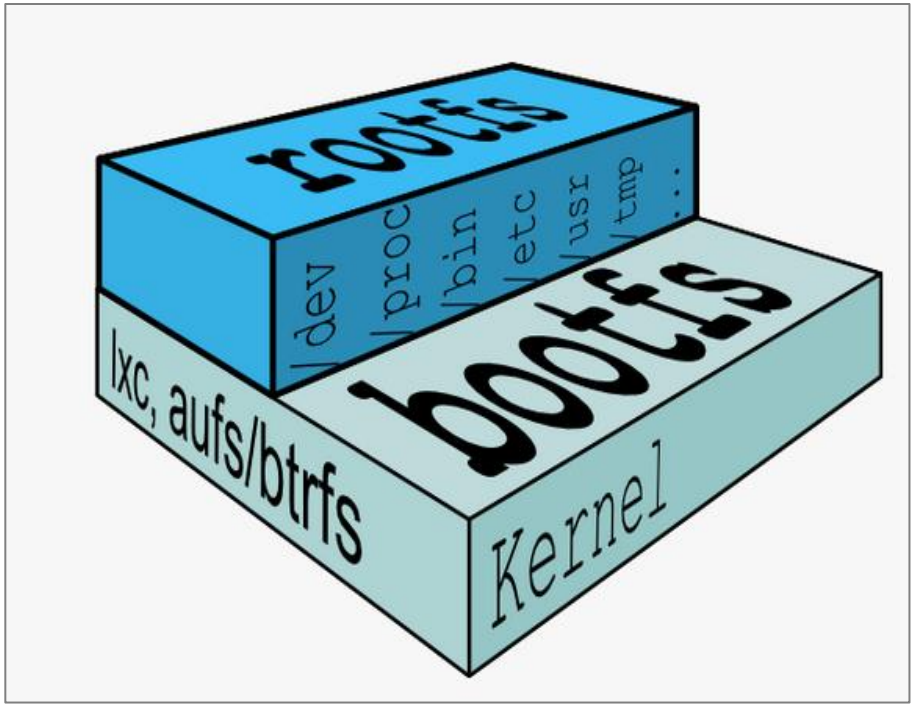
Docker核心原理：AUFS

- 運用AUFS技術
  - UnionFS ：
    - 支援將不同目錄掛載到同一個虛擬文件系統下的文件系統
    - 支持為每一個成員目錄（類似Git Branch）設定read-only、read-write 和whiteout-able 權限
  - 類似分層的概念：
    - 對read-only 權限的branch 可以邏輯上進行修改
      將一個read-only的branch 和一個writeable 的branch 聯合在一起，即可在不變的基礎上進行讀寫
  - Docker 在AUFS 上構建的container image 就是用類似方法

Docker核心原理

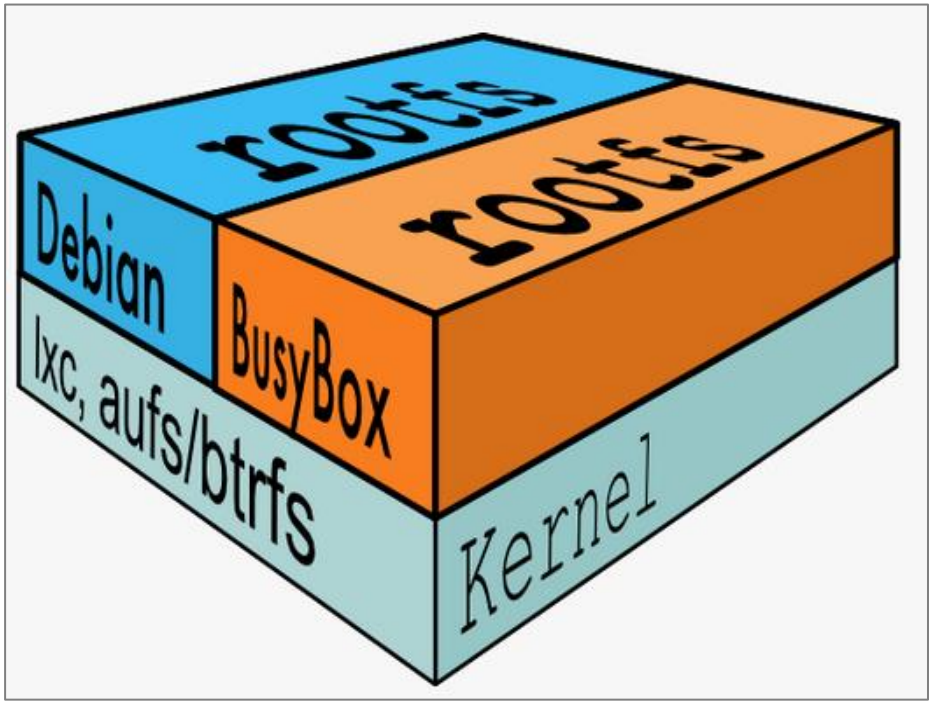
典型的Linux：

啟動到運行需兩個FS (bootfs + rootfs)



不同的Linux版本：

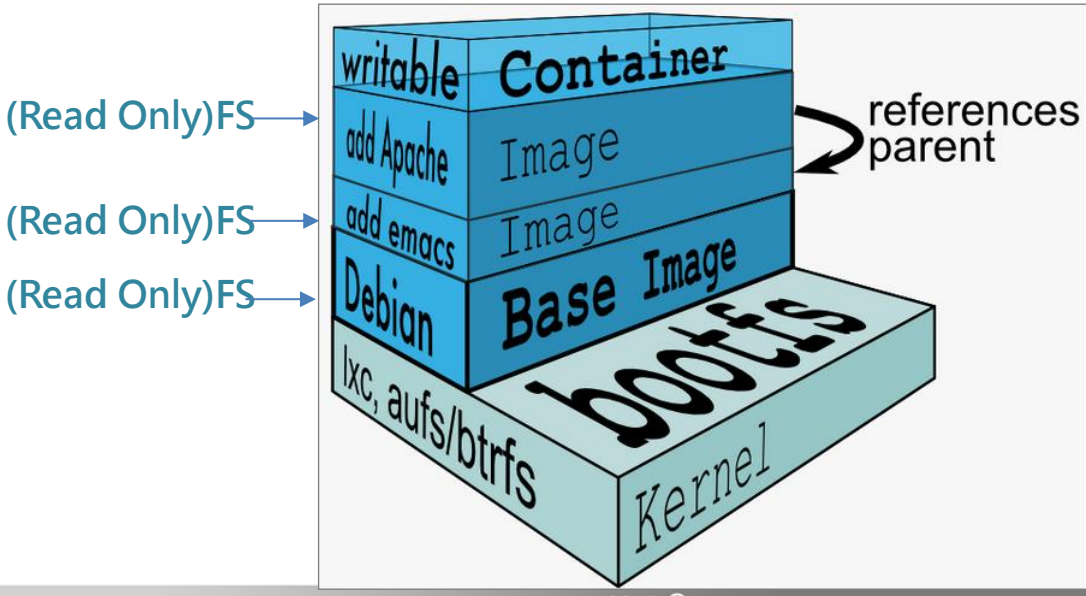
相同的bootfs，rootfs不同



Docker 核心原理

- **AUFS的特性：**
  每一個對read-only層文件/目錄的修改都只會存在於上層的writeable層中
- **共享read-only的FS層**：多個container可共享read-only的FS層
  - Docker將read-only的FS層稱作 "**image**"
  - 對於container，整個rootfs都是read-write的

事實上，所有的修改都寫入最上層的writeable層中，image不保存用戶狀態，只用於模板、新建和復制使用



Docker網路

- Docker網路模型之技術

Network namespace

- 網絡資源的隔離，包含網絡設備、IPv4和IPv6協議棧、IP路由表、防火牆、/proc/net目錄、/sys/class/net目錄、port（socket）等

Linux Bridge

- 功能相當於物理交換機，為連在其上的設備（容器）轉發數據封包。如docker0 Bridge

Iptables

- 主要為容器提供NAT及容器網絡安全

veth pair

- 兩個虛擬網卡組成的數據通道
- 用於連接Docker容器和Linux Bridge。一端在容器中作為eth0網卡，另一端在Linux Bridge中作為一端口

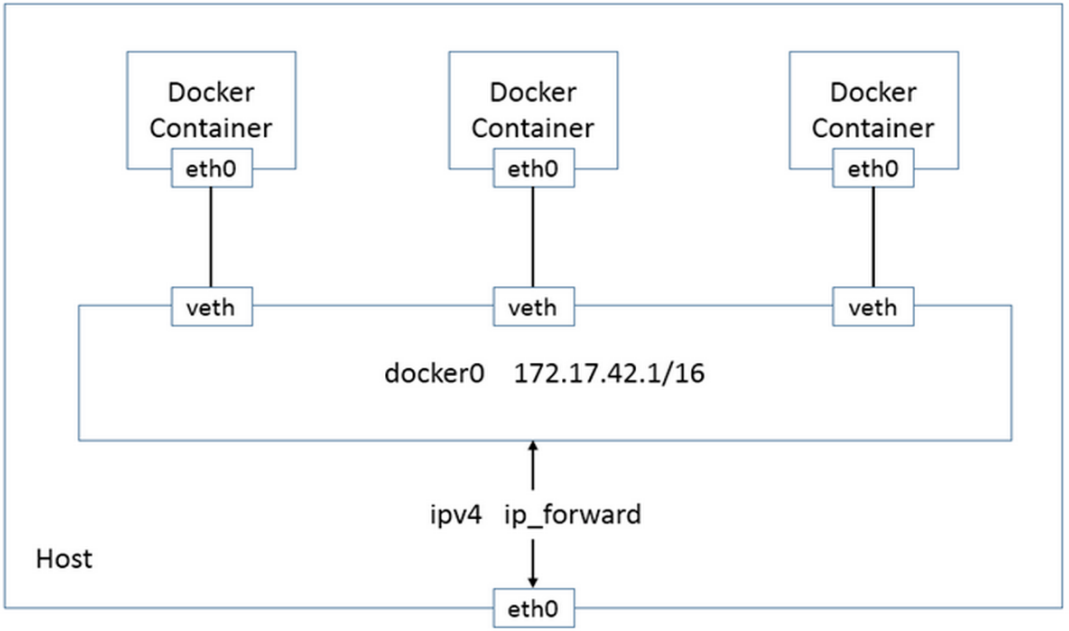
Container網路模式

host模式：
容器和Host共享Network namespace，直接暴露在公共網絡中，需藉由port mapping進行協調

container模式：
兩容器共享Network namespace。如：kubernetes中的pod，多個容器共享一個Network namespace，而非Host共享

none模式：
容器有獨立的Network namespace，但沒有對其進行任何網絡設置，如分配veth pair 和bridge連接，配置IP等

bridge模式：
Docker預設的網路設定，每個容器有各自Network Namespace、IP等，並將Host上的容器連接到一個虛擬bridge



Docker相關資源

- Docker 目前支援


DOCKER CE FOR AWS


DOCKER CE FOR AZURE


DOCKER CE FOR DEBIAN


DOCKER CE FOR CENTOS DISTRIBUTION


DOCKER CE FOR FEDORA


DOCKER CE FOR MAC

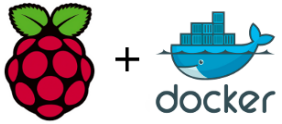

DOCKER CE FOR UBUNTU


DOCKER CE FOR WINDOWS

- Docker 相關資源

- Docker應用

Google
Cloud Platform



雲端服務平台：AWS

- Amazon web services：雲端運算整合，提供許多遠端Web服務
  - 提供運算能力、儲存選項、聯網與資料庫 (如：**Amazon EC2**、Amazon S3、EC2 Container Service)
  - 快速部屬新服務，提供50種以上的服務
    Web應用程式、大數據與HPC、資料備份儲存等
  - 安全性高於現場部署
  - 運作區域擴及全球
- 現有客戶：

Jet Propulsion Laboratory
California Institute of Technology

QNAP

dropcam

NETFLIX

Expedia

D-Link

MEDIATEK

資料來：https://aws.amazon.com/tw/

LAB 1

容器虛擬化技術Docker 資安現況

Docker風險層面

Host
主機風險

Host OS Risks

- Improper user access rights
- Host component vulnerabilities

Image Risks

- Image vulnerabilities
- Image configuration
- Embedded malware
- Embedded secrets
- Image trust

Image / Container
風險

Orchestrator Risks

- Unbounded administrative access
- Weak or unmanaged credentials
- Unmanaged inter-container network traffic
- Mixing of workload sensitivity levels

Registry Risks

- Insecure connections to registries
- Stale images in registries

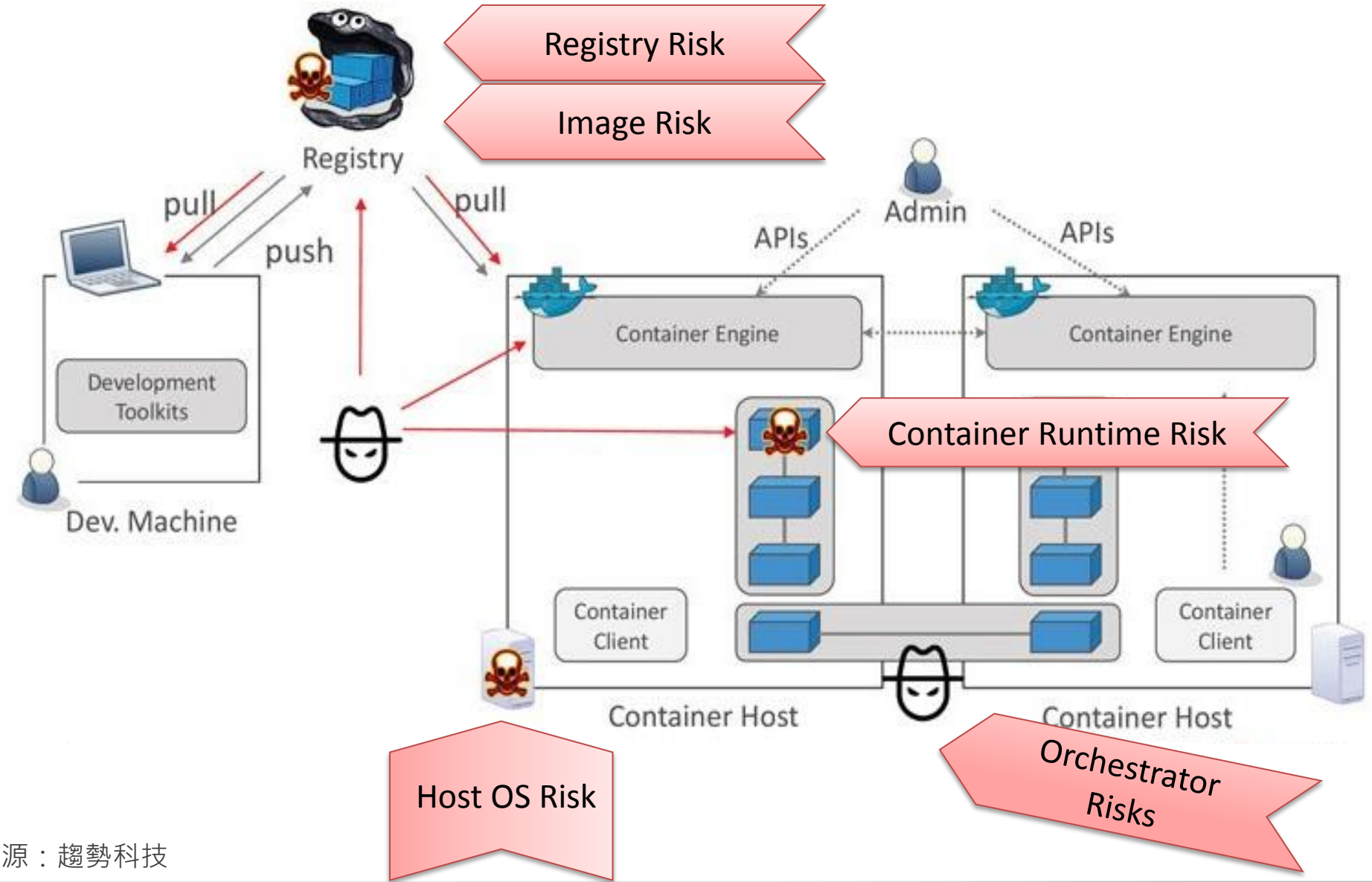
Container Runtime Risks

- Vulnerabilities within the runtime software
- Unbounded network access from containers
- Insecure container runtime configurations
- Shared kernel

系統管理風險

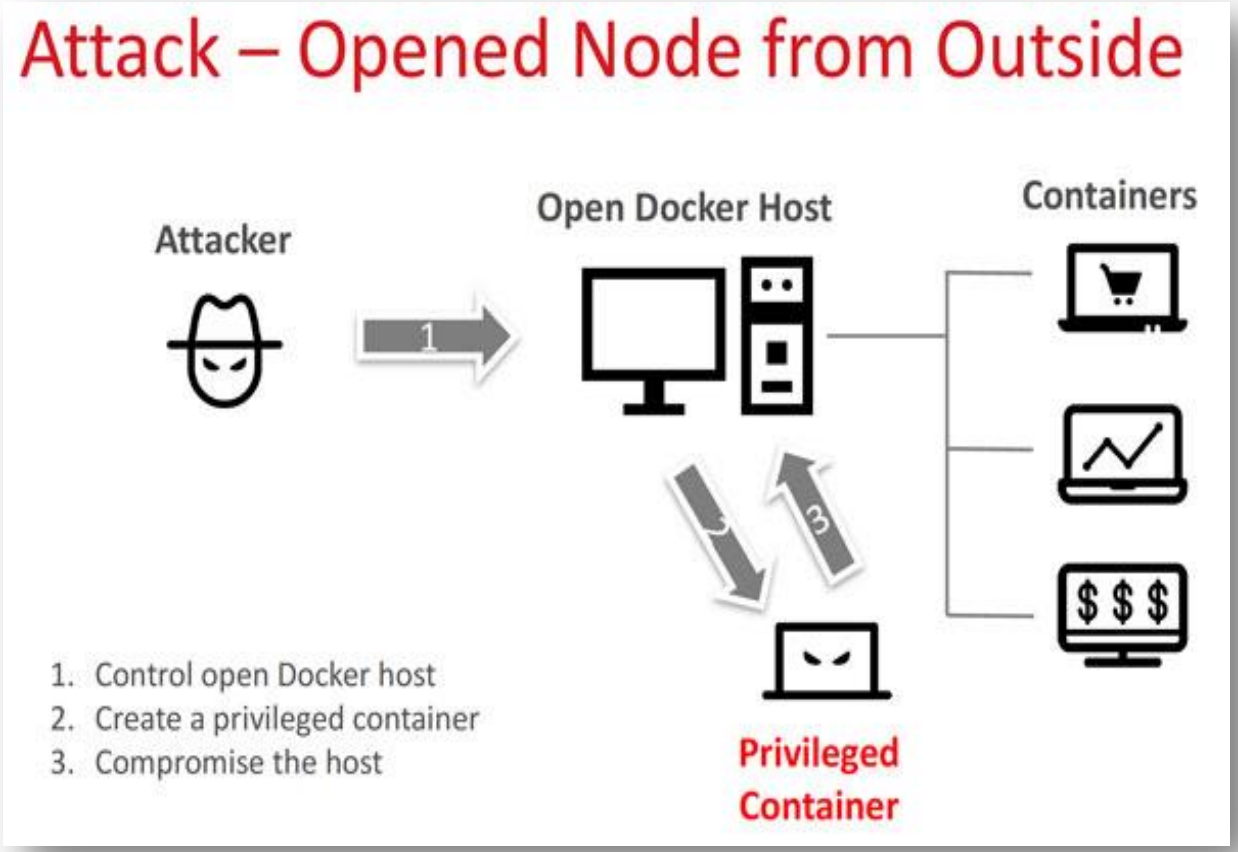
資料來源：NIST SP800-190

Docker 環境安全風險



資料來源：趨勢科技

Docker 攻擊情境 I



Host OS Risks

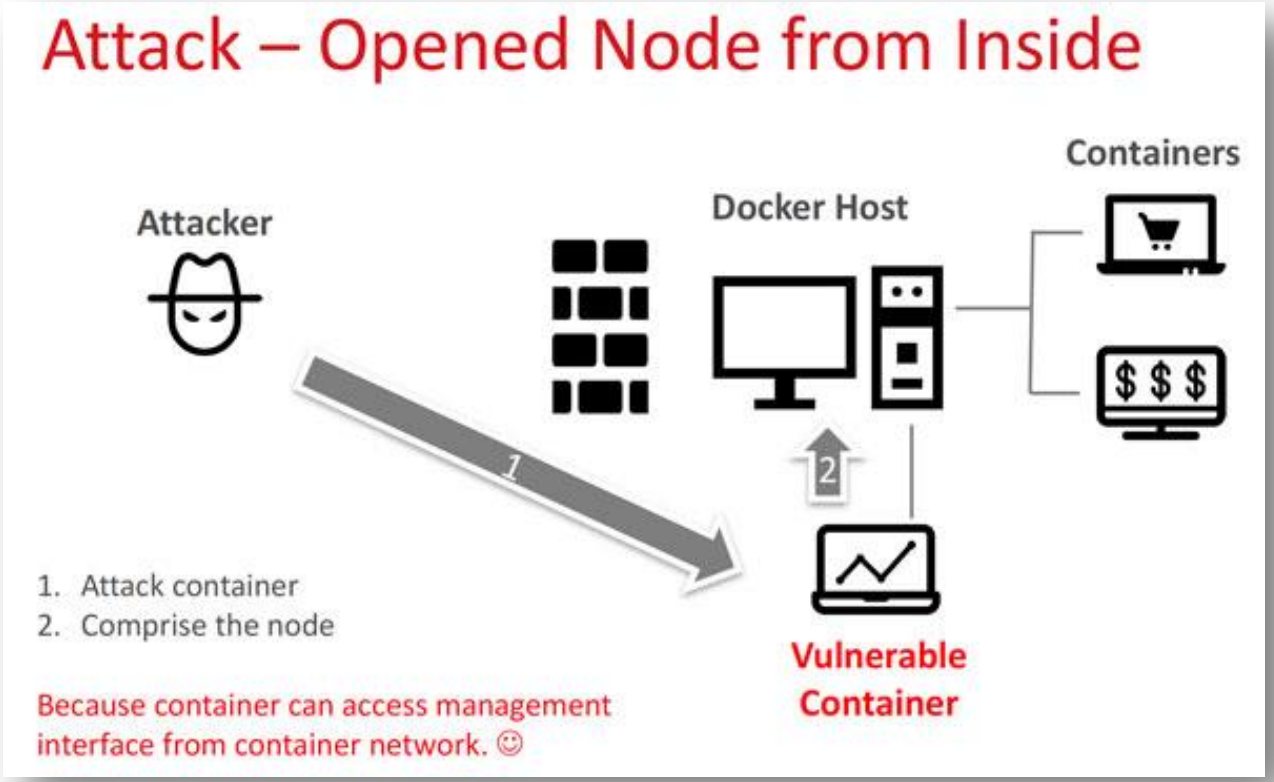
1. 駭客進入公開Docker主機

2. 建立一個Root權限的Container

3. 藉由駭客創建的特權Container，攻擊Host 及 同一機器上的Containers

資料來源：http://www.ithome.com.tw/news/113303

Docker 攻擊情境 II



Container Runtime Risks

Orchestrator Risks

1. 駭客進入有漏洞的Container，因Container網路可連線至管理介面

2. 駭客透過Container網路操作管理介面，取得Host的權限

資料來源：http://www.ithome.com.tw/news/113303

Docker 攻擊情境 III

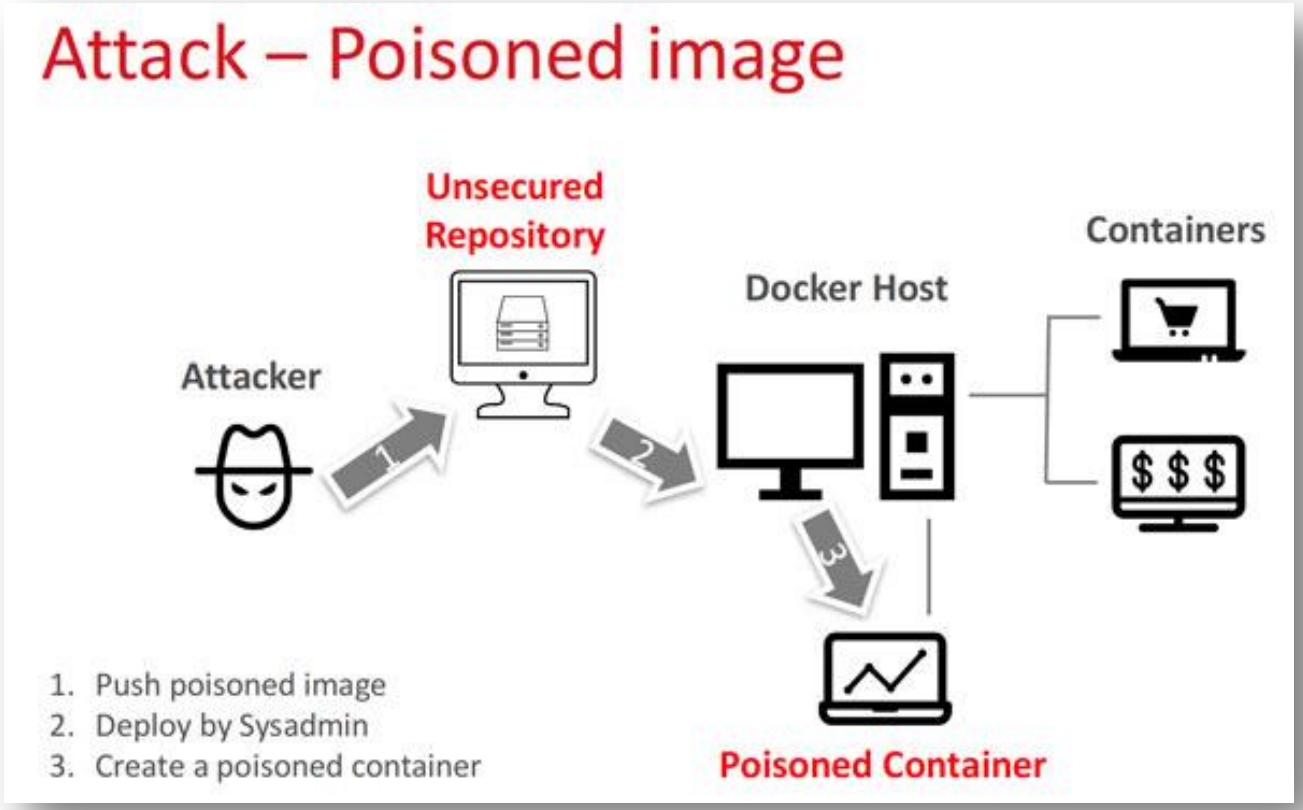


Image Risk

1. 駭客上傳有漏洞或遭感染的Image至Repository

⬇

2. 等待該Image被載入執行至Docker Host

⬇

3. 駭客即可進入有漏洞之Container，進而去攻擊該網域或Host

資料來源：http://www.ithome.com.tw/news/113303

Containers 安全??

CONTAINERS

For containers, security is problem #1

It may take a disaster or two for the lessons of needing to do security right sink in. Only then will containers be ready for prime time.

By Steven J. Vaughan-Nichols

ITworld | MAY 11, 2015

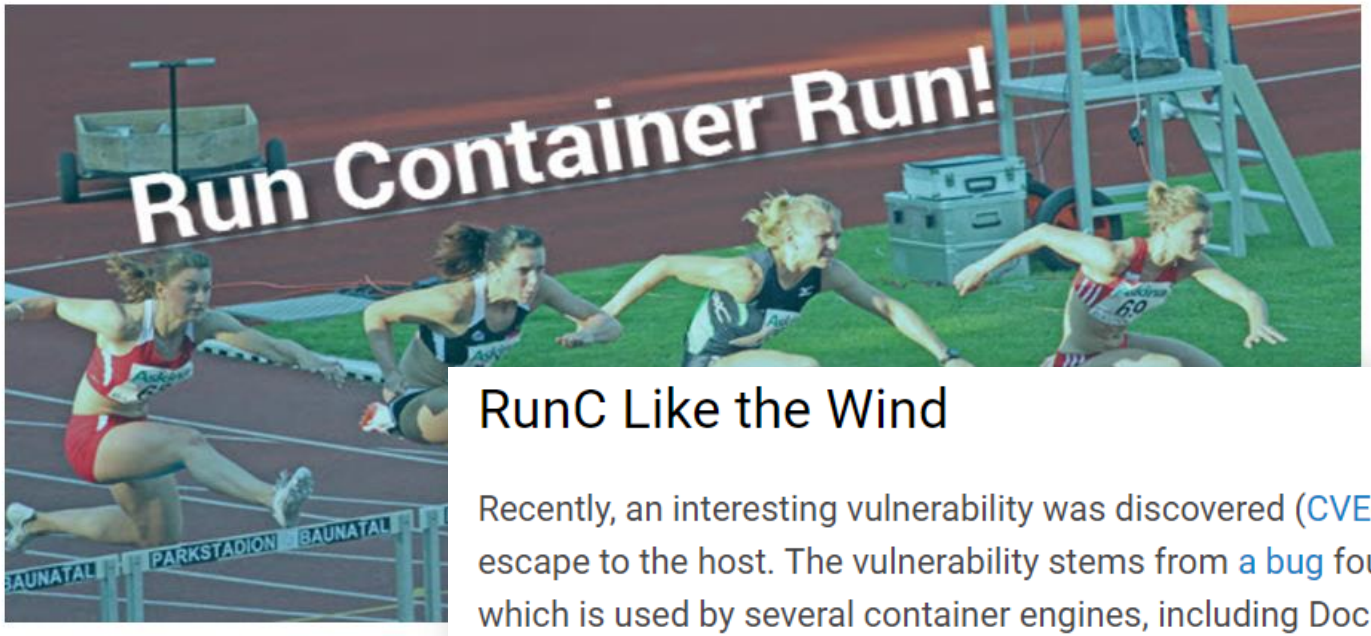
Docker and other container technologies run as root

I quote directly from the security documentation from the most popular of all container technologies, Docker: "Running containers (and applications) with Docker implies running the Docker daemon. This daemon currently requires root privileges.

資料來源：IT World（2015/05/11)

Containers 安全??

17 JAN 2017 | **CVE-2016-9962: Run Container Run**



By Sagie Dulce

RunC Like the Wind

Recently, an interesting vulnerabili
the host. The vulnerability stems from a bug found in *opencontainers' runc code*, which is used by several container engines, including Docker.

RunC Like the Wind

Recently, an interesting vulnerability was discovered (CVE-2016-9962) that enables container escape to the host. The vulnerability stems from a bug found in *opencontainers' runc code*, which is used by several container engines, including Docker.

The vulnerability is exploited when *exec*-ing a command inside an already running container. When that happens, a malicious process inside the container can access a "forgotten" file descriptor of a directory that resides on the host. This in turn can be used to perform directory traversal to the host's file system, thus facilitating a nasty and easy escape.

資料來源：aqua security

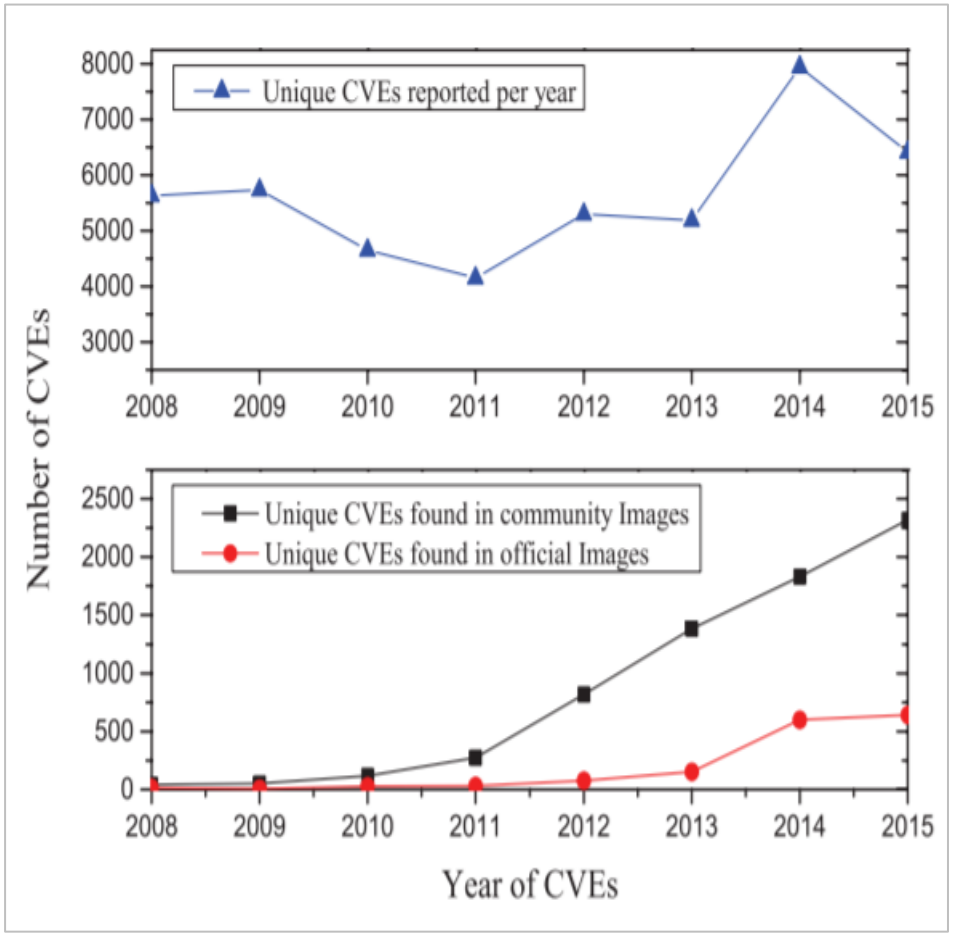
Containers 安全??

攻擊Docker開發者攻擊：

1. 將Docker開發人員誘導至駭客所控管的網頁
2. 利用Docker API執行**非特權的程式**，展開**主機重新綁定（Host Rebinding）攻擊**
3. 取得受害者機器上的Docker守護進程控制權
4. 駭客已可呼叫任何Docker API，最後再於Docker中**植入影子容器**，以長駐於Hypervisor中。

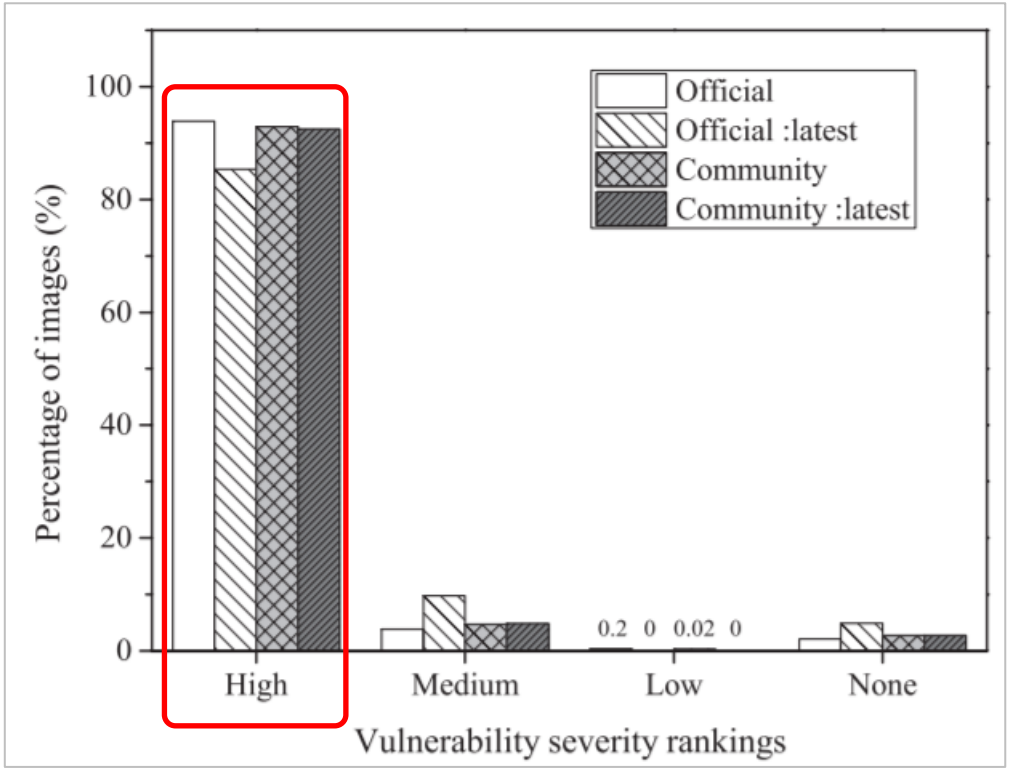
資料來源：iThome（2017/07/31）

Images 安全嗎??



Comparison between CVEs discovered in CVE database and CVEs found in community images and official images from 2008 to 2015

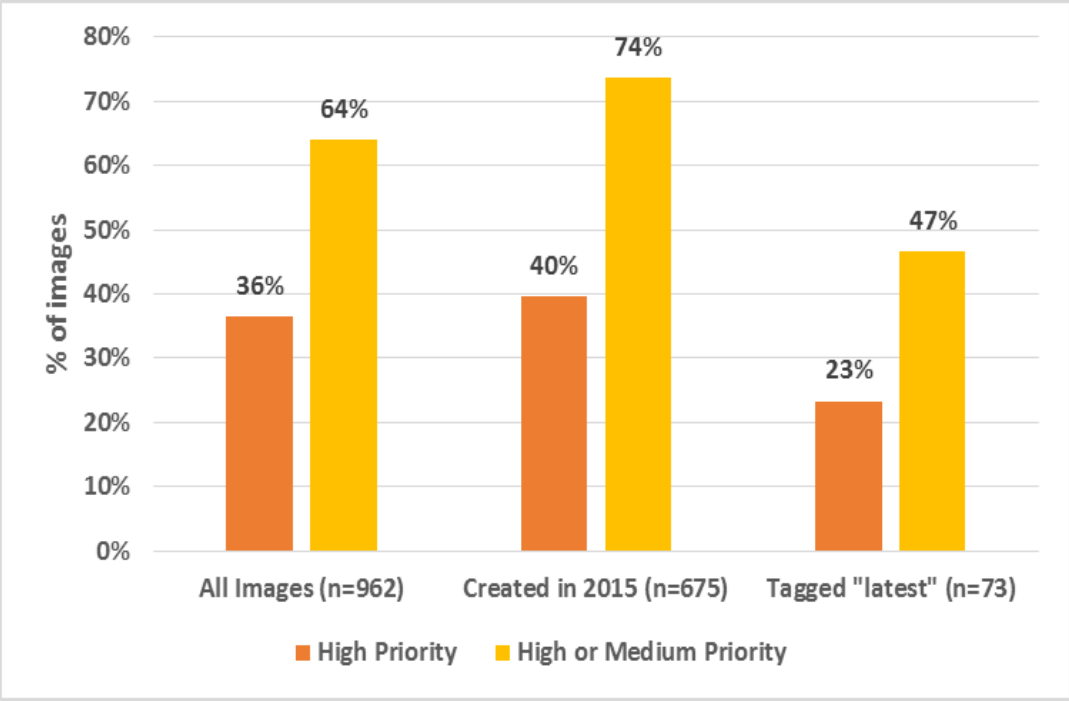
含有高風險的漏洞之images，高達90%



資料來源：http://dance.csc.ncsu.edu/papers/codaspy17.pdf (2017/03/24)

2015年 : Official Images 安全?? (1)

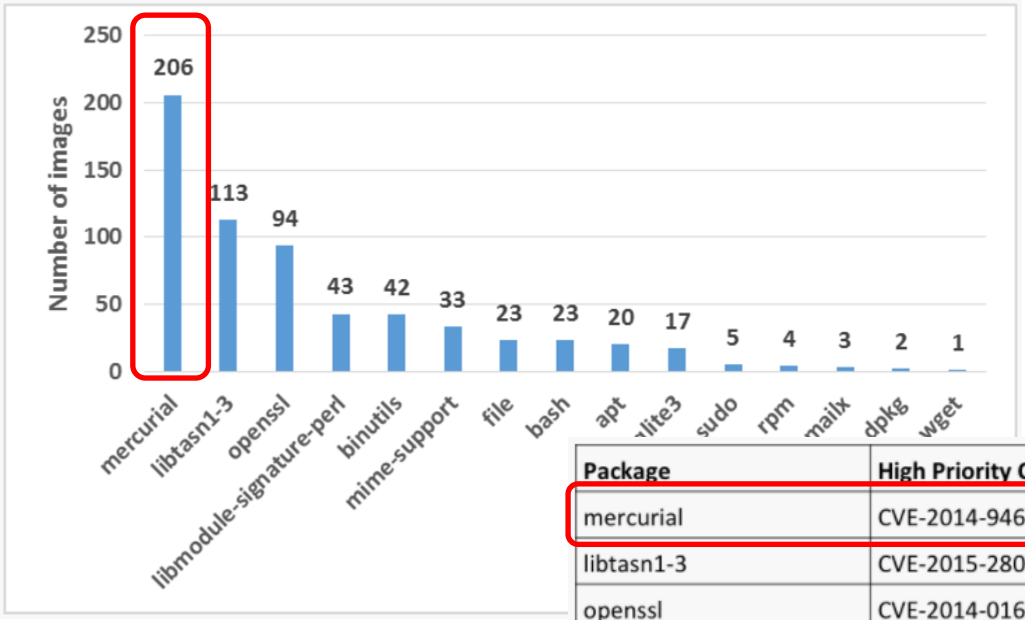
BanyanOps have published a report stating that 'Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities', which include some of the sensational 2014 issues such as ShellShock and Heartbleed. The analysis also looks at user generated 'general'



- Docker Hub上有約75個官方repository (約960個images)
  - **超過1/3 images有高風險**的漏洞
  - 在2015年創建的images中，有**四成images有高風險漏洞**
  - 標記為 `latest tag`的image中，仍有**23% and 47%的images**有中、高級漏洞

資料來源： https://banyanops.com/blog/analyzing-docker-hub/ (2015/05/26)

2015年 : Official Images 安全?? (2)

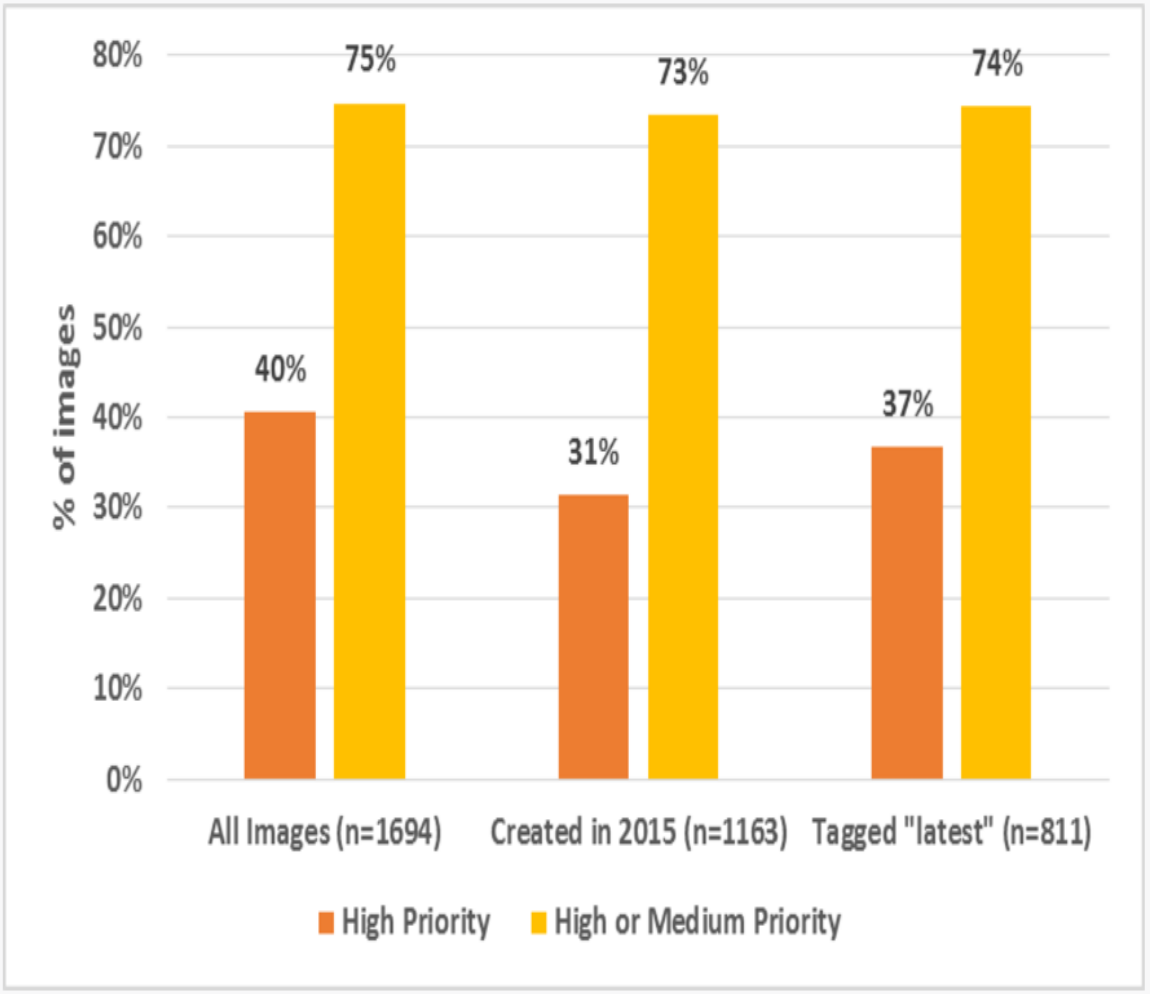


- 官方Image所檢測出的高風險CVE統計

Package	High Priority CVEs
mercurial	CVE-2014-9462
libtasn1-3	CVE-2015-2806
openssl	CVE-2014-0160,CVE-2014-3513,CVE-2014-3567,CVE-2015-0292
libmodule-signature-perl	CVE-2015-3408,CVE-2015-3409
binutils	CVE-2014-8485,CVE-2014-8501,CVE-2014-8502,CVE-2014-8503,CVE-2014-8504
mime-support	CVE-2014-7209
file	CVE-2014-9653
bash	CVE-2014-6271,CVE-2014-7169,CVE-2014-7186,CVE-2014-7187
apt	CVE-2012-0954,CVE-2012-3587,CVE-2013-1051,CVE-2014-0487,CVE-2014-0488,CVE-2014-0489,CVE-2014-0490
sqlite3	CVE-2015-3414,CVE-2015-3415,CVE-2015-3416
sudo	CVE-2013-1775
rpm	CVE-2013-6435,CVE-2014-8118
heirloom-mailx	CVE-2004-2771
dpkg	CVE-2014-3127
wget	CVE-2014-4877

資料來源：
https://banyanops.com/blog/analyzing-docker-hub/ (2015/05/26)

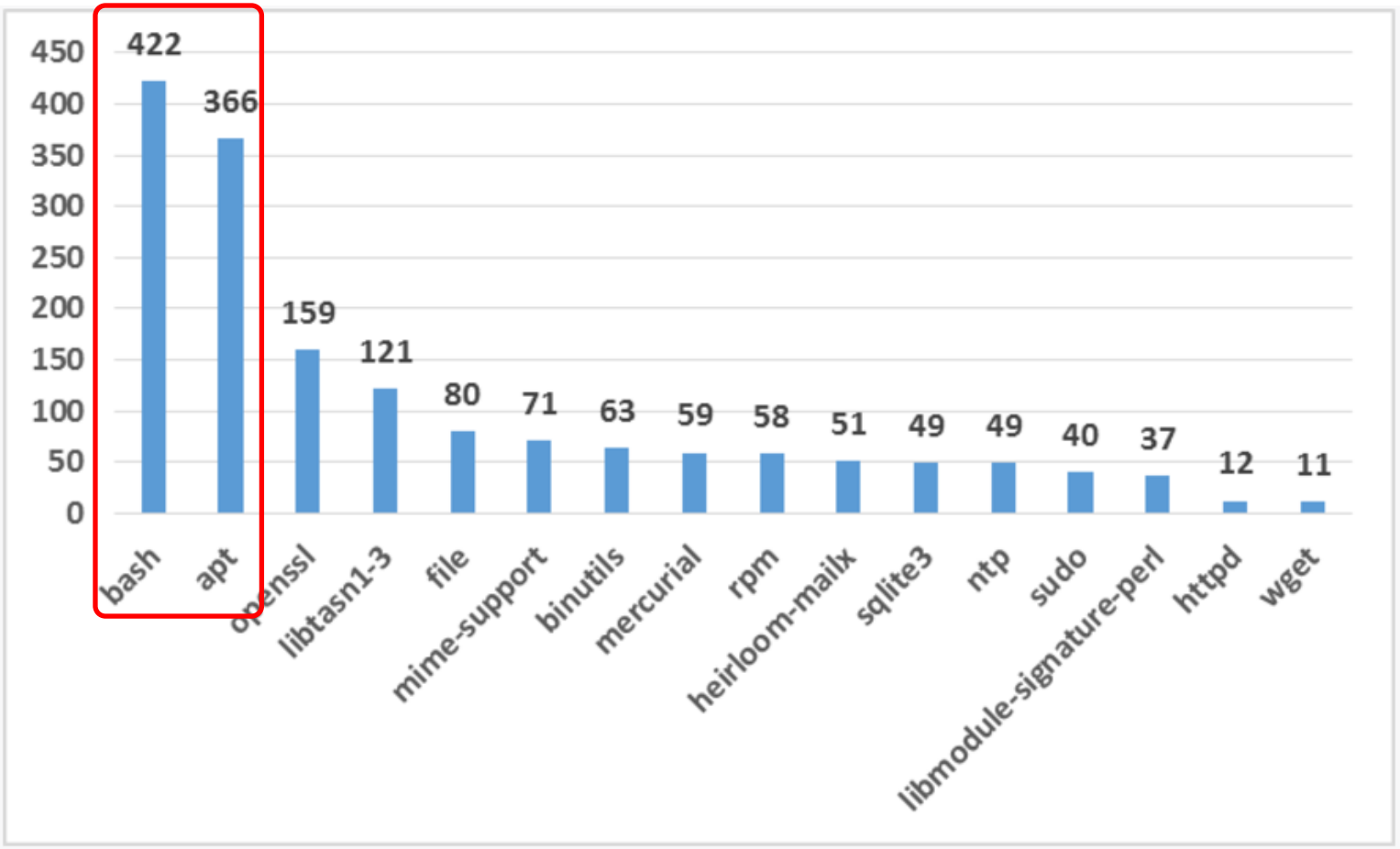
2015年：非官方 Images 安全?? (1)



- Docker Hub上有約95,000個一般repository (約10萬多個images)
- 隨機取樣約1700個images作分析
  - **40% images有高風險**的漏洞
  - 在2015年創建的images中，約1/3 **images有高風險漏洞**
  - 佔有**70%**以上的images有中、高級漏洞

資料來源： https://banyanops.com/blog/analyzing-docker-hub/ (2015/05/26)

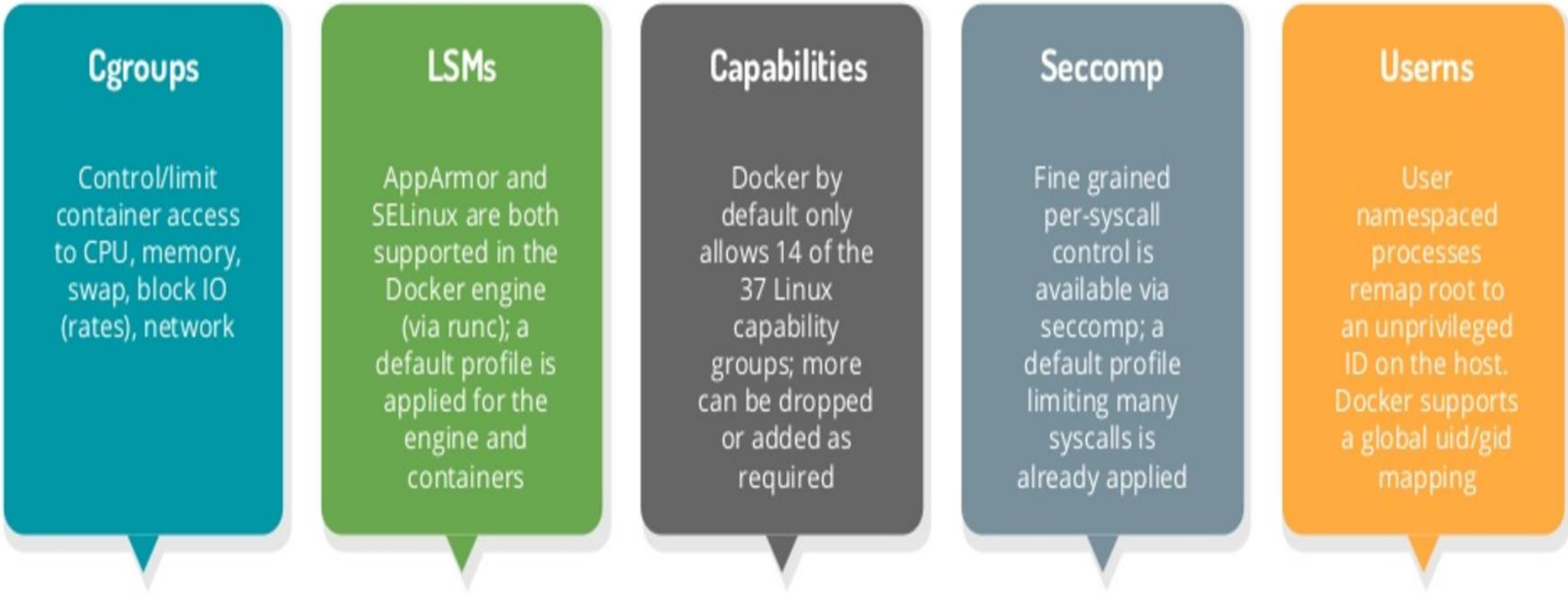
2015年：非官方 Images 安全?? (2)



資料來源： https://banyanops.com/blog/analyzing-docker-hub/ (2015/05/26)

容器虛擬化技術Docker 資安因應實務建議

Docker 安全概念



資料來源：https://www.slideshare.net/PhilEstes/docker-london-container-security

安全概念 1 : Cgroup (1)

限制系統資源取用

- Memory
  - -m 或 –memory：設定記憶體使用量，e.g. 100M (最小單位4M)
  - --memory-swap： 設定記憶體與swap的用量
    docker run -m 300M --memory-swap=400M Ubuntu
    // 設定記憶體 300M、swap 100M
- CPU
  - Docker v1.13版本以上：限制該Container 最多只會使用一顆CPU來執行
    docker run -it --cpus="1" ubuntu:16.04 bash
  - Docker 1.13版本以下：使用 -cpu-period 和--cpu-quota 來代替
    docker run -it --cpu-period="100000" --cpu-quota="200000 ubuntu:16.04 bash
    // 最多使用2顆CPU
- PID
  - --pids-limit 20：限制該Contianer容器最大的Process數量為20
    docker run –it --pids-limit 20 ubuntu:16.04 bash

安全概念 1 : Cgroup (2)

限制系統資源取用

- 其他參數設定
  - *--kernel–memory*
  - *--cpu-shares*
  - *--cpuset-cpus*
  - *--cpuset-mems*
  - *--device-read-bps*
  - *--device-read-ios*
  - *--device-write-bps*
  - *--device-write-ios*
  - *--blkio-weight*
  - *--blkio-weight-device*

安全概念 2 : LSMs

AppArmor / SELinux

- **AppArmor ：** Linux 內建Mandatory Access Control (MAC)系統
  - 設置執行程序的操作控制權限，可以限制程序的讀寫目錄或文件，開啟、讀寫port等等
  - Docker 1.13版之後，自動產生一份default的版本

    docker run --rm -it --security-opt apparmor=docker-default ubuntu:16.04 bash
  - 使用者也可以寫一份自己的版本，在Host上載入

    apparmor_parser -r -W /path/to/your_profile

    docker run -it -security-opt apparmor=[profilename] ubuntu:16.04
  - 停止 apparmor

    /etc/init.d/apparmor stop
  - 卸載 apparmor 規則

    apparmor_parser -R /path/to/profile
  - 啟動 apparmor

    /etc/init.d/apparmor start

安全概念 3 : Capabilities

新增/刪除 Linux Kernel Capabilities

- 範例：新增SYS_ADMIN、刪除NET_RAW功能

```
$ docker run  --rm  -ti  busybox  sh
/  #  hostname foo
hostname:  sethostname:  Operation not permitted

$ docker run  --rm  -ti  --cap-add=SYS_ADMIN busybox  sh
/  #  hostname foo
<hostname  changed>

$ docker run  --rm  -ti  --cap-drop=NET_RAW  busybox  sh
/  #  ping 8.8.8.8
Ping  permission  denied  (are  you  root?)
/  #
```

安全概念 4 : Seccomp

Linux Security Computing

- 限制一個Container可以執行的系统调用

```
$ car  policy.json
{
        "defaultAction" : "SCMP_ACT_ALLOW",
        "syscalls" : [
                {
                        "name" : "chmod",
                        "action" : "SCMP_ACT_ERRNO"
                }
        ]
}
$ docker run  --rm  -it  --security-opt  seccomp:policy.json
busybox  chmod 640
/etc/resolv.conf
Chmod:  /etc/resolve.conf: Opration not permitted
```

安全概念 5 : User Namespaces

支援Linux User Namespaces

- 設定Container內外皆為Root
  - 使用參數 --priviledge

docker run –privileged Ubuntu

// Container有權限操作Host

- 預防Container提權： 設定docker deamon的餐數 --userns-remap

```
$  docker  deamon –userns-remap=default(|someuser:somegrp)
<deamon starts with uid and gid mappings from /etc/sub{u,g}id>

$  docker  run  –rm  –ti  –v  /bin:/host/bin  busybox  sh
/  #  cp  mybadshell  /host/bin/sh
cp:  can't create '/host/bin/sh' : File exists

/  #  cp  /host/bin  &&  mv  sh  sh.bak
mv:  can't  rename  'sh' : Permission denied

/  #
```

Docker Image 實務

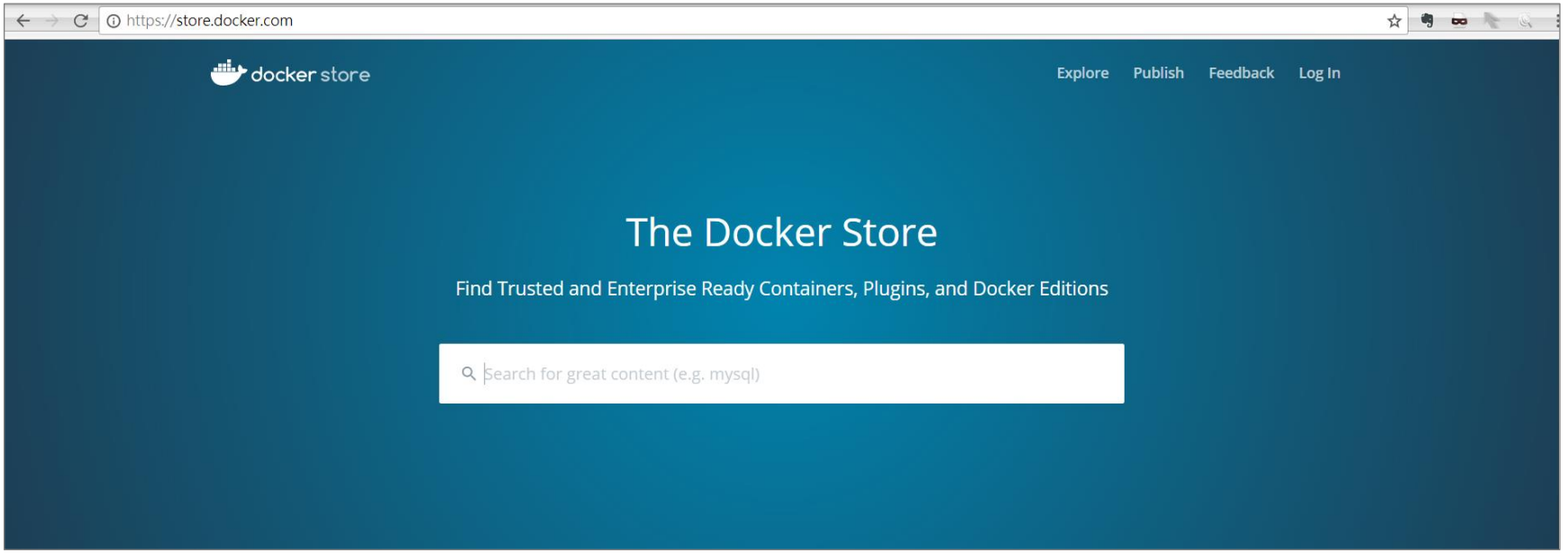
- Docker可透過Dockerfile建置image
- 官方提供Dockerfile 優化實務

Container 生命週期短	• 可被停止、消毀 • 以最小配置設定，建立新的一個container
善用.dockerignore file	• 排除不須用到的文件及目錄 • 可提升建構效率
避免安裝不必要的Package	• 減少相依姓、複雜性及檔案大小 • 多餘的package可能利於駭客使用
Container功能單一化	• 一個Container只有一種應用程式 • 有效水平擴增、重複使用，減少相依性
Layer數最小化	• Layer數小可提升Container啟動速度 • 考慮：Dockerfile的可讀性與維護性之間的平衡
排序參數	• 以字母、數字及換行排序參數 • 可避免重複安裝、易管理
建構時使用Cache	• –no-cache=true • 建構時，以Cache內優先搜尋相同的image

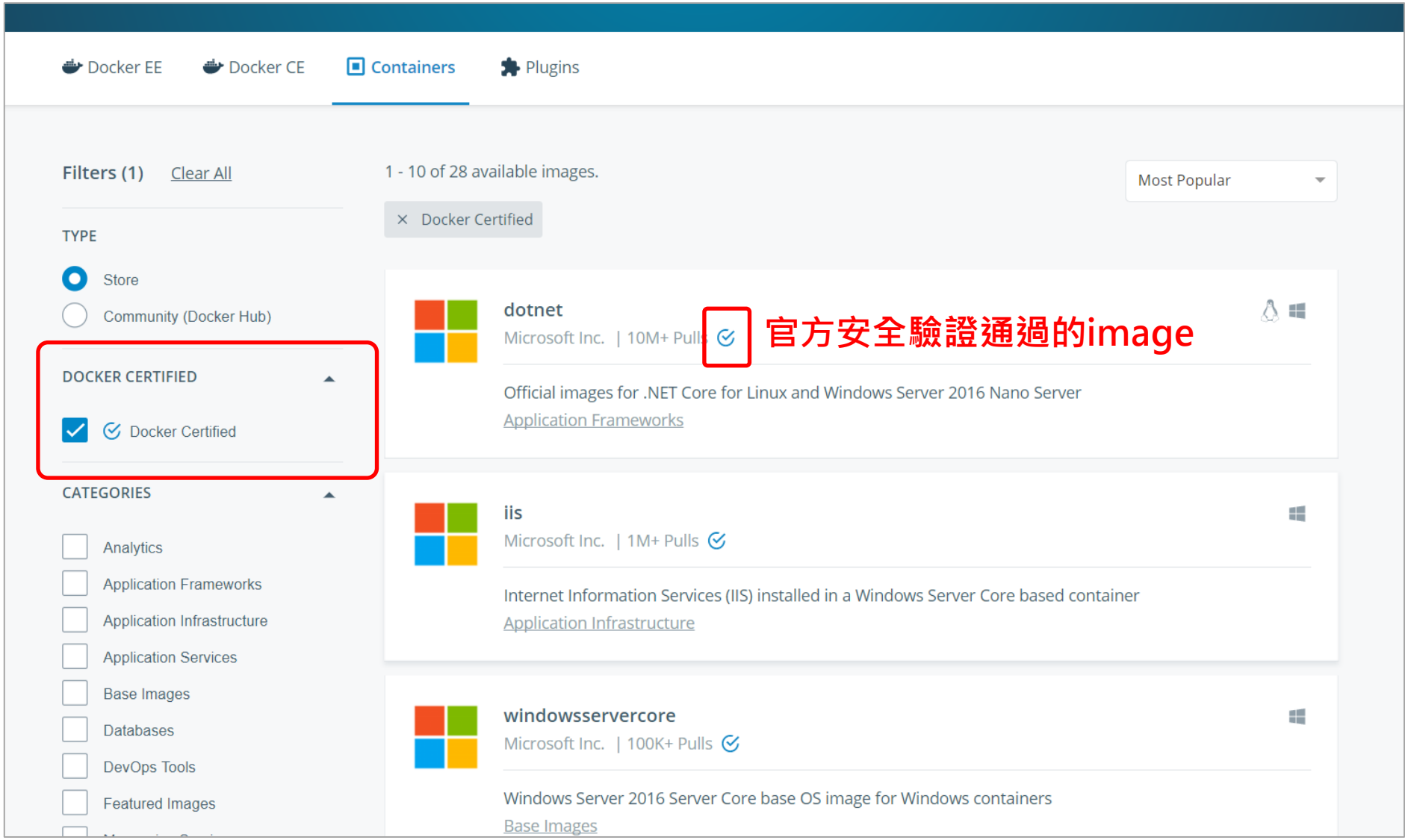
參考資料：Docker Document

使用公開Hub之Image安全 (1)

- docker store
  - 提供**開源**及**商用**工具
  - 支援Docker Store **封閉測試**
    - 鼓勵獨體開發商出版可靠的內容
    - 必須通過安全掃描、元件庫存、開源碼授權的使用等驗證
    - 具備完整的分類與搜尋功能

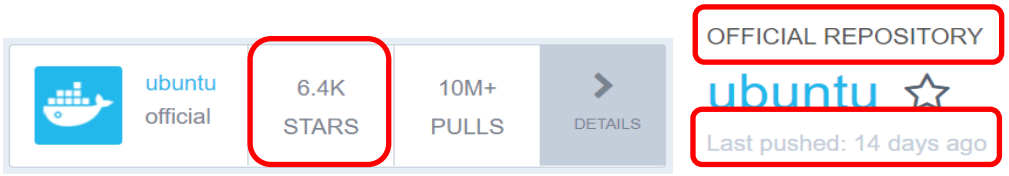


Docker Store



使用公開Hub之Image安全

- docker hub ： Docker 公開Hub，有很多不安全的image
- 如何選用??
  - 官方提供之image、 評價數高、定期更新



  - 透過工具檢測image安全性
    - 可以掃描哪些第三方函式庫含有弱點
    - 可以支援一些已知binary檔掃描其中引用的第三方函式庫弱點
      - Python
      - Ruby
      - Tomcat
      - Heartbleed / Shellshock
    - 可以掃出第三方函式庫的License及版本
    - 相關產品：

JFrog Xray　clair

Docker Image檢測工具：clair

- 靜態分析Docker Image之開源工具
- 支援的漏洞類型：

Data Source	Data Collected	Format	License
Debian Security Bug Tracker	Debian 6, 7, 8, unstable namespaces	dpkg	Debian
Ubuntu CVE Tracker	Ubuntu 12.04, 12.10, 13.04, 14.04, 14.10, 15.04, 15.10, 16.04 namespaces	dpkg	GPLv2
Red Hat Security Data	CentOS 5, 6, 7 namespaces	rpm	CVRF
Oracle Linux Security Data	Oracle Linux 5, 6, 7 namespaces	rpm	CVRF
Alpine SecDB	Alpine 3.3, Alpine 3.4, Alpine 3.5 namespaces	apk	MIT
NIST NVD	Generic Vulnerability Metadata	N/A	Public Domain

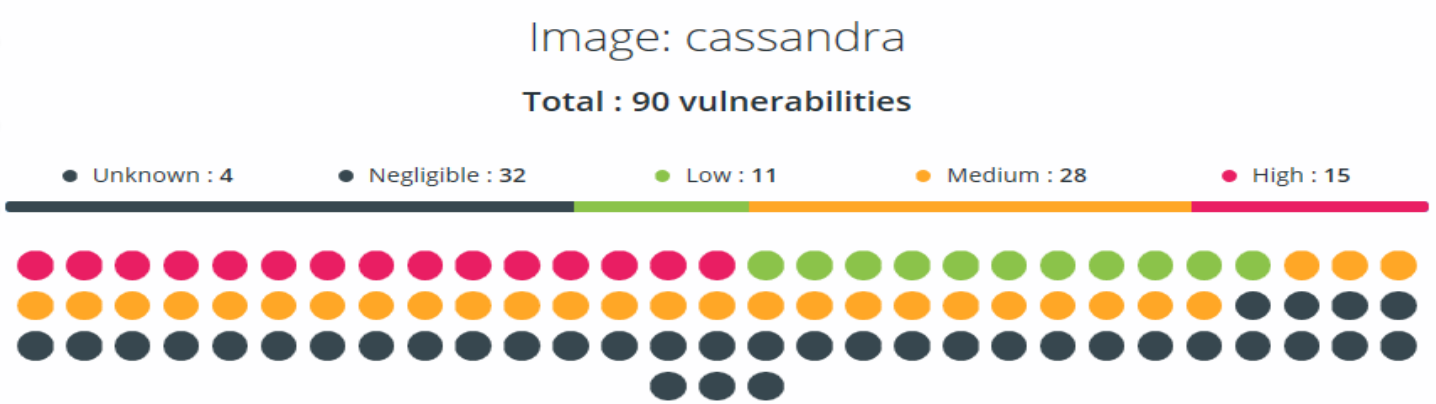
clair 分析結果

CLAIR CONTROL REPORT

Image name

漏洞總數

漏洞的風險數



說明第三方函式庫的版本、漏洞內容

4a9aa6aef7f39a5d69ec48f489f4bfdf32dbb8eed4bdd1914d88fadb669b6a2d

libtasn1-6 4.2-3+deb8u3 - ⚠

- CVE-2017-10790
  The _asn1_check_identifier function in GNU Libtasn1 through 4.12 causes a NULL pointer dereference and crash when reading crafted input that triggers assignment of a NULL value within an asn1_node structure. It may lead to a remote denial of service attack.
  Link

libxtst 2:1.2.2-1 - ⚠

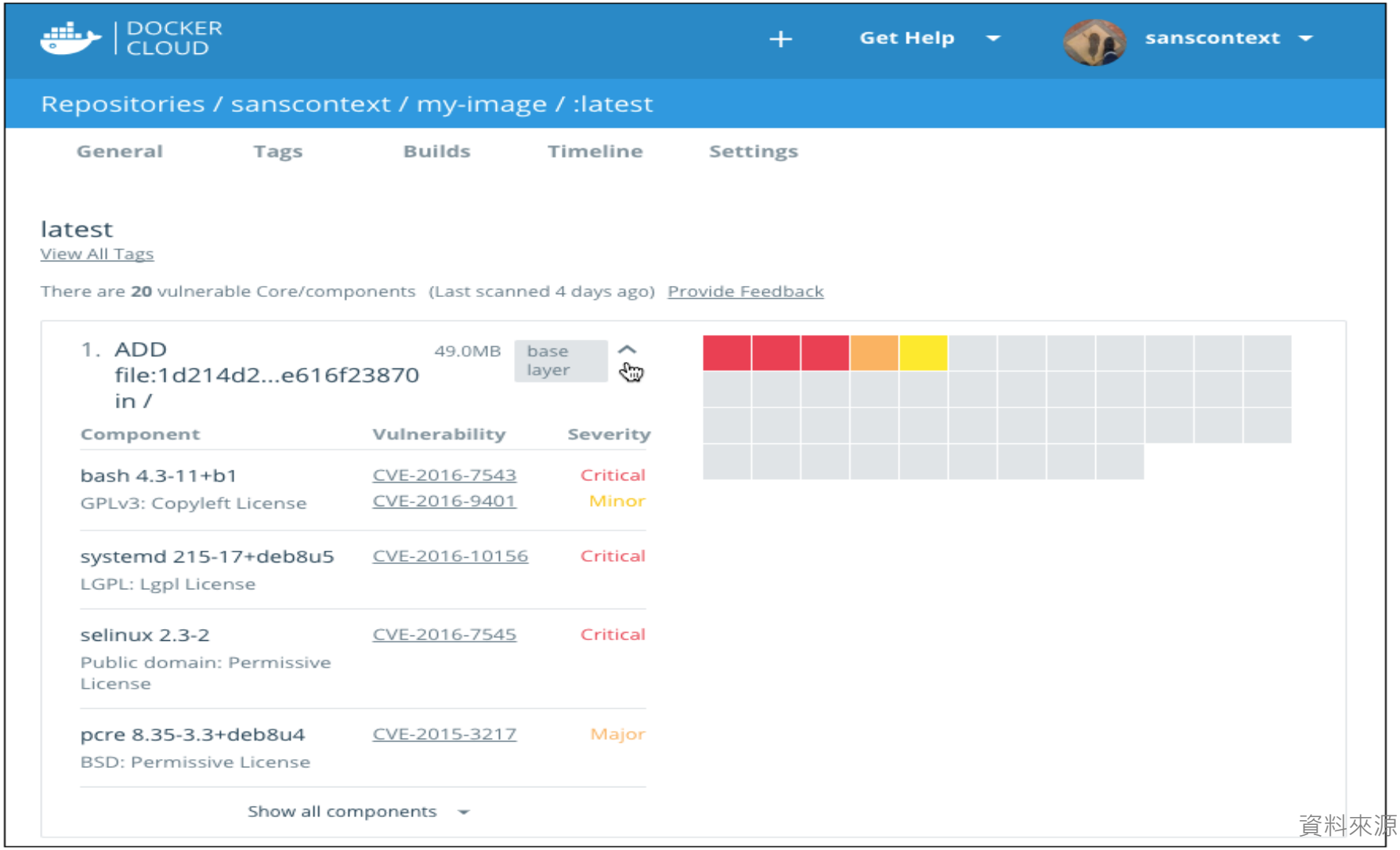
- CVE-2016-7951
  Multiple integer overflows in X.org libXtst before 1.2.3 allow remote X servers to trigger out-of-bounds memory access operations by leveraging the lack of range checks.
  Link

- CVE-2016-7952
  X.org libXtst before 1.2.3 allows remote X servers to cause a denial of service (infinite loop) via a reply in the (1) XRecordStartOfData, (2) XRecordEndOfData, or (3) XRecordClientDied category without a client sequence and with attached data.
  Link

資料來源：clair

Docker安全檢測

- Docker image Scan



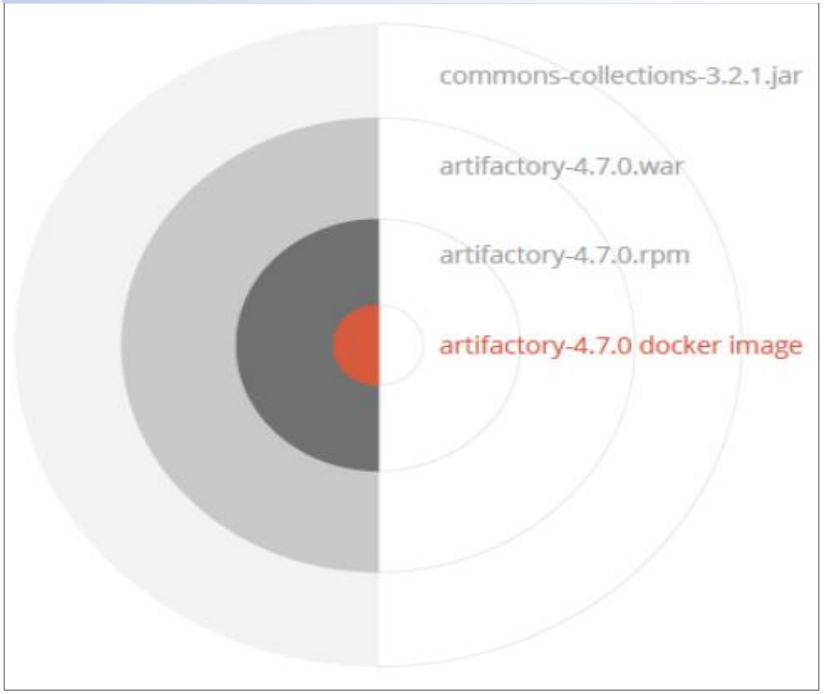
資料來源：Docker Doc

Docker安全檢測

- JFrog Xray
  - 針對已知的Binary檔案做掃描，Ex: Java, python …

- Report License

- Deep Recursive Scanning



- Top Vulnerabilities

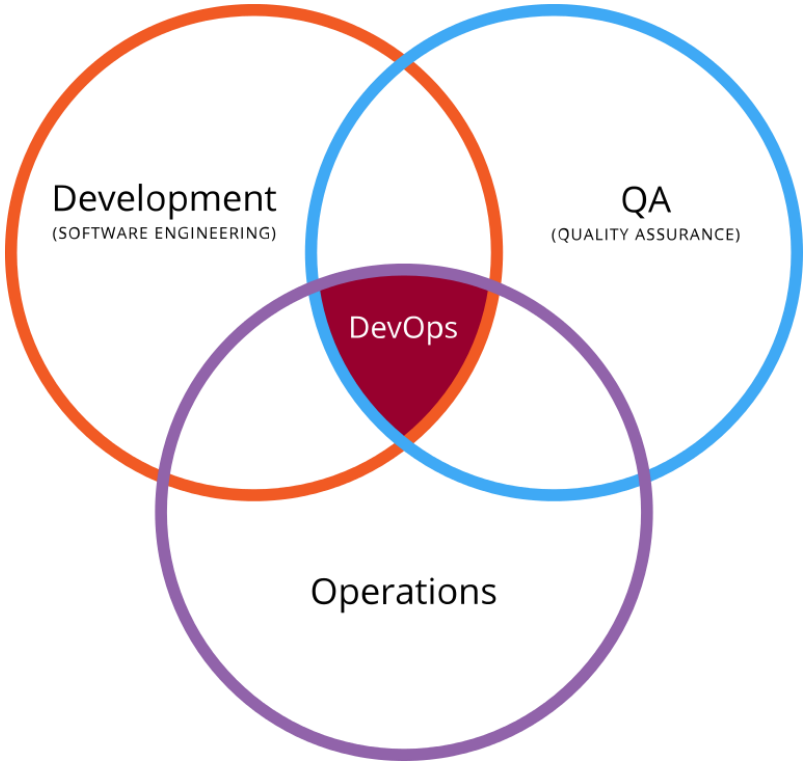
資料來源：JFrog Xray

LAB 2

DevOps 持續整合開發介紹

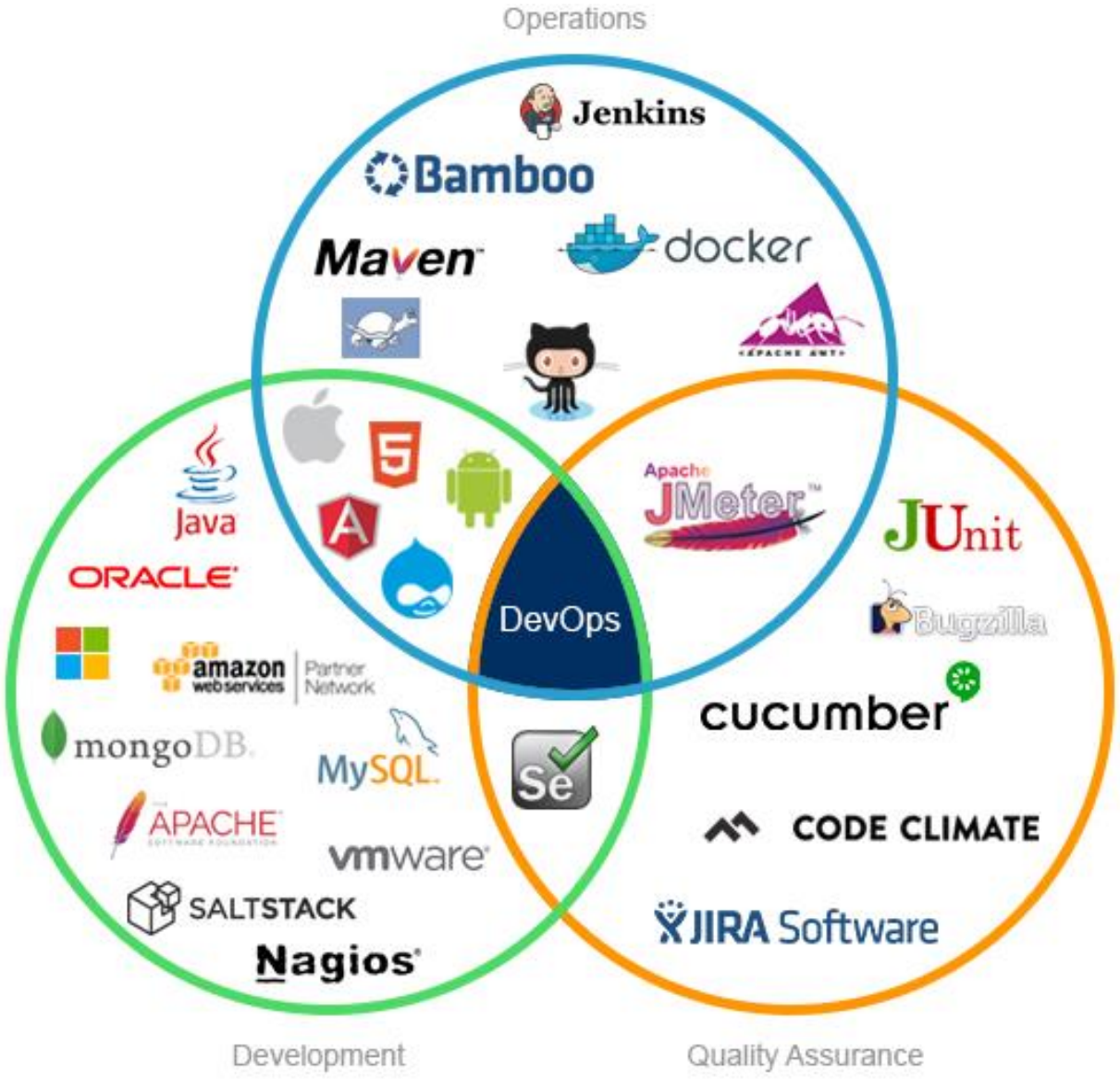
DevOps是什麼?

- DevOps = Development + Operations
- 透過自動化「**軟體交付**」和「**架構變更**」的流程，使構建、測試、發布軟體能夠更加地快捷、頻繁和可靠



資料來源：Wikipedia DevOps

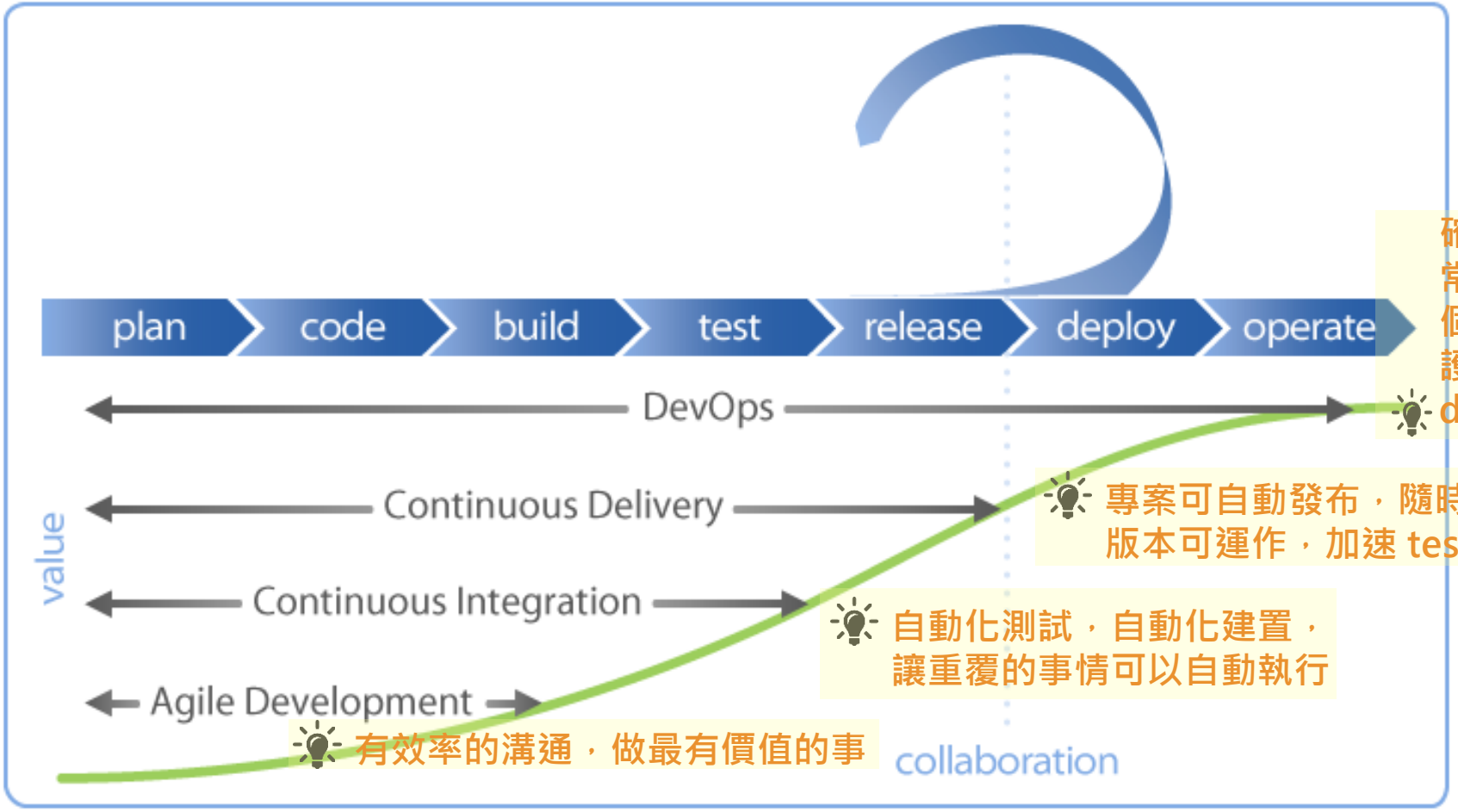
DevOps軟體套件



資料來源：Kraftic

持續整合(continuous integration, CI)

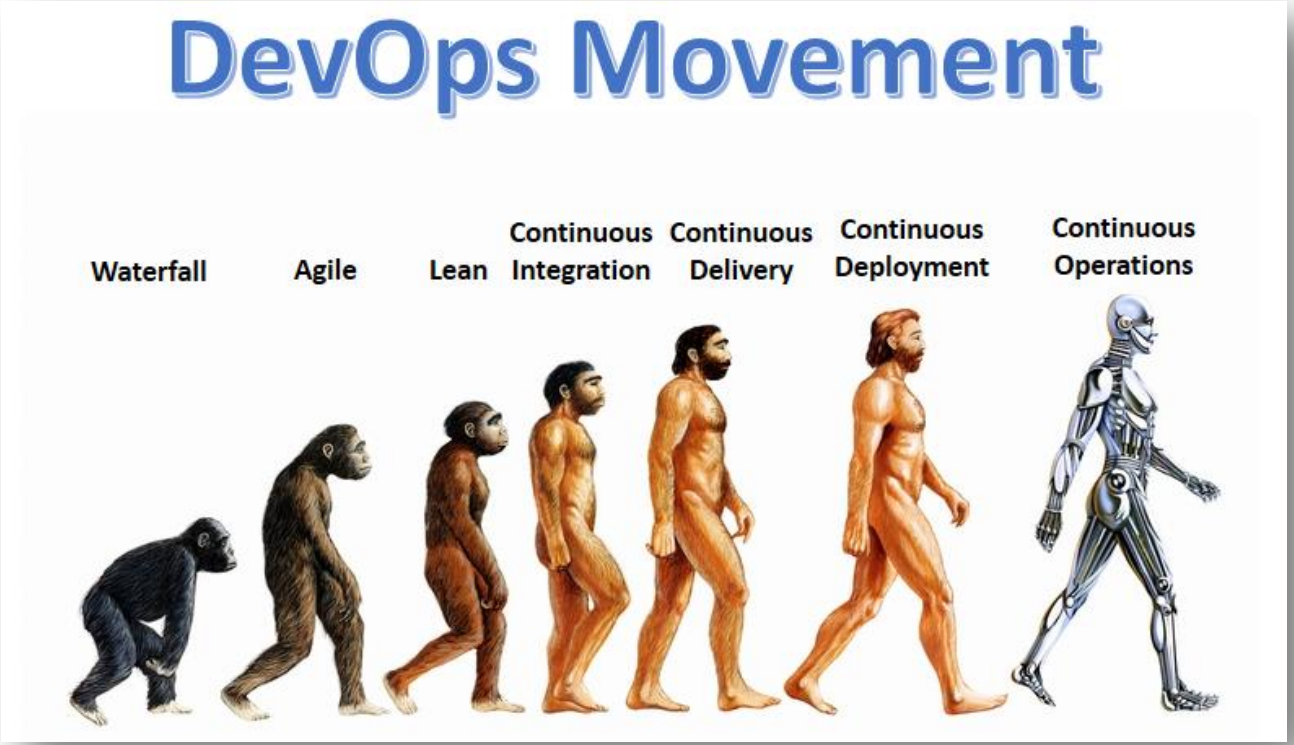
- 專案流程：



資料來源：iThome

持續整合(continuous integration, CI)

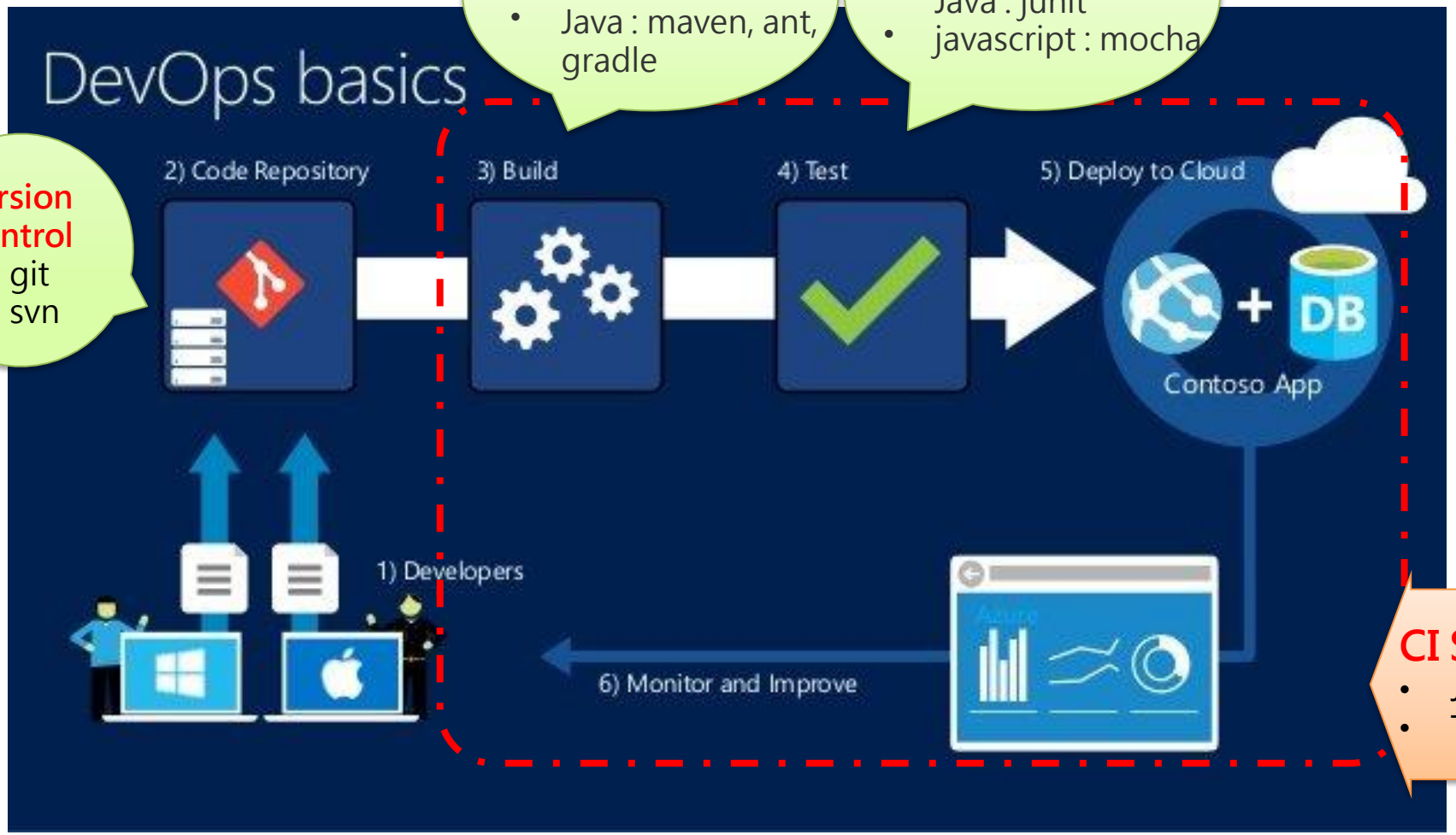
- 目標：
  - 更快尋找和解決錯誤
  - 改善軟體品質
  - 減少驗證和發行新軟體更新所需的時間



資料來源：iThome

DevOps持續整合開發

- 如何開始呢?



資料來源：https://www.linkedin.com/pulse/10-tools-devops-success-article-ericka-chickowski-parham-ashoori

Version Control : 

- Subversion(SVN)：開放原始碼的版本控制系統
  - 提供多人共有資料的一致性
  - 版本控管
  - 適用於文件檔案或是程式碼檔案
  - 接近一般的檔案管理操作模式
  - 可劃分不同的權限

- 目錄結構
  - **Trunk**：主分支，是日常開發進行的地方
  - **Branches**：階段性的release版本，可繼續進行開發和維護的 如：不同客制化的版本，也可以放在分支中進行開發
  - **Tags**：發布版本 (read-only)

Version Control :

基本指令

- checkout (co) : 將SVN上的資料checkout出來
  svn checkout svn+ssh://trac.net/home/svn/repos/branches/life
- update (up) : 更新為SVN上的最新版本
  svn up
- commit (ci) : 將目前有修改的 commit 到 SVN
  svn ci (全部有修改的都會commit)
  svn ci file1 file2 file3 (只將file1 2 3 commit)
- add : 將檔案或目錄加進 SVN ，之後還要再 commit 才會真的加進 SVN
  svn add file/folder
- mv : 改檔名，但之前的紀錄還是會繼續保留
  svn mv old_file new_file
- revert : 還原成前一個版本的狀態
  svn revert file/folder
  svn revert *
- resolved : 如果檔案有conflict，處理完後，要resolved，才可以ci
  svn resolved file
- diff : 將現在的檔案跟SVN做比較
  svn diff (比對全部)
  svn diff file/folder

Version Control :

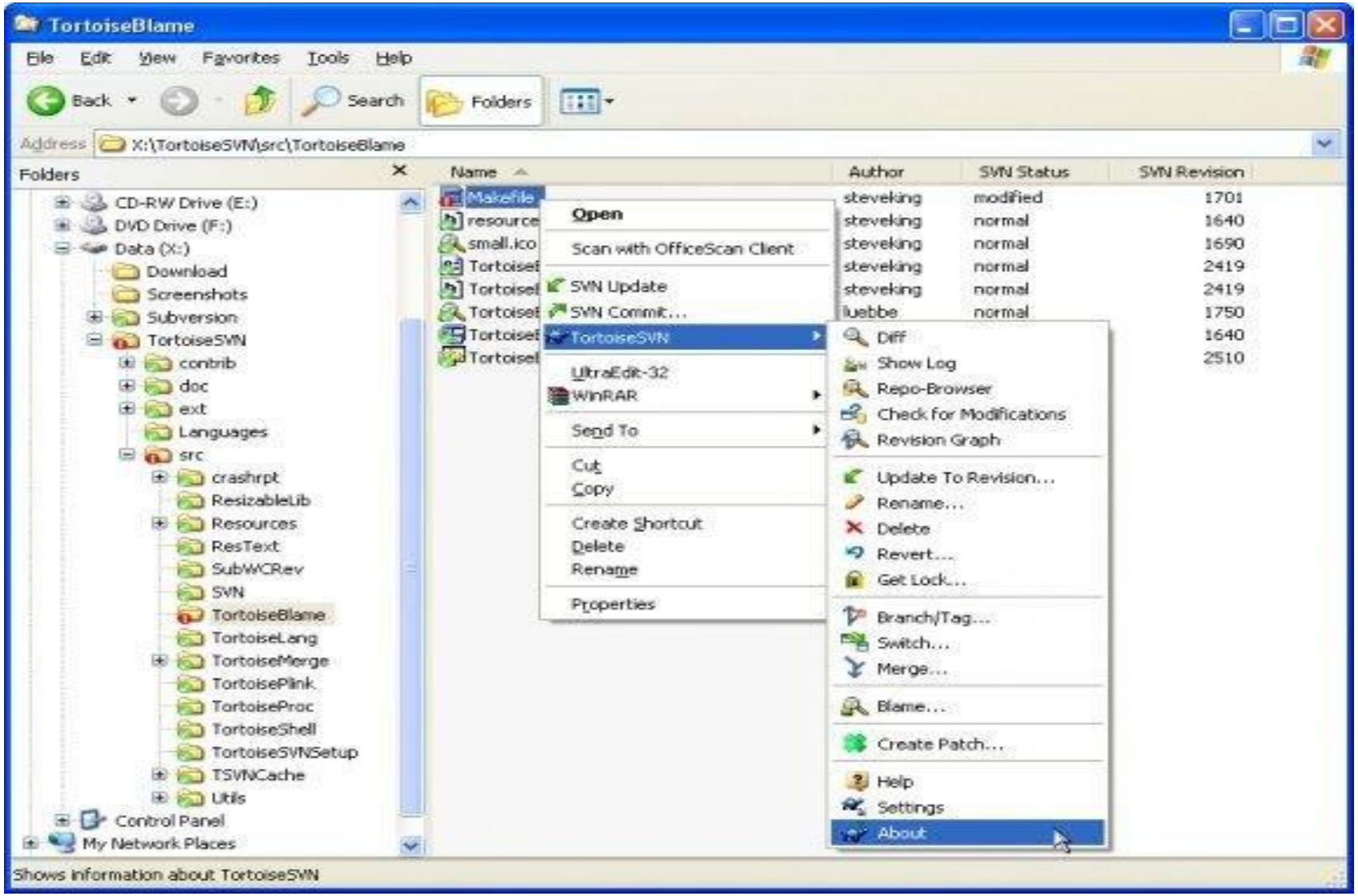
SVN 狀態說明

- ? - 是新的檔案，不在SVN裡
- A - 新增的檔案
- C - 檔案跟SVN的不同，合併失敗，要手動處理
- D - 移除的檔案
- M - 有修改過
- U - 有更新
- G - 跟SVN上的檔案不同，但合併成功

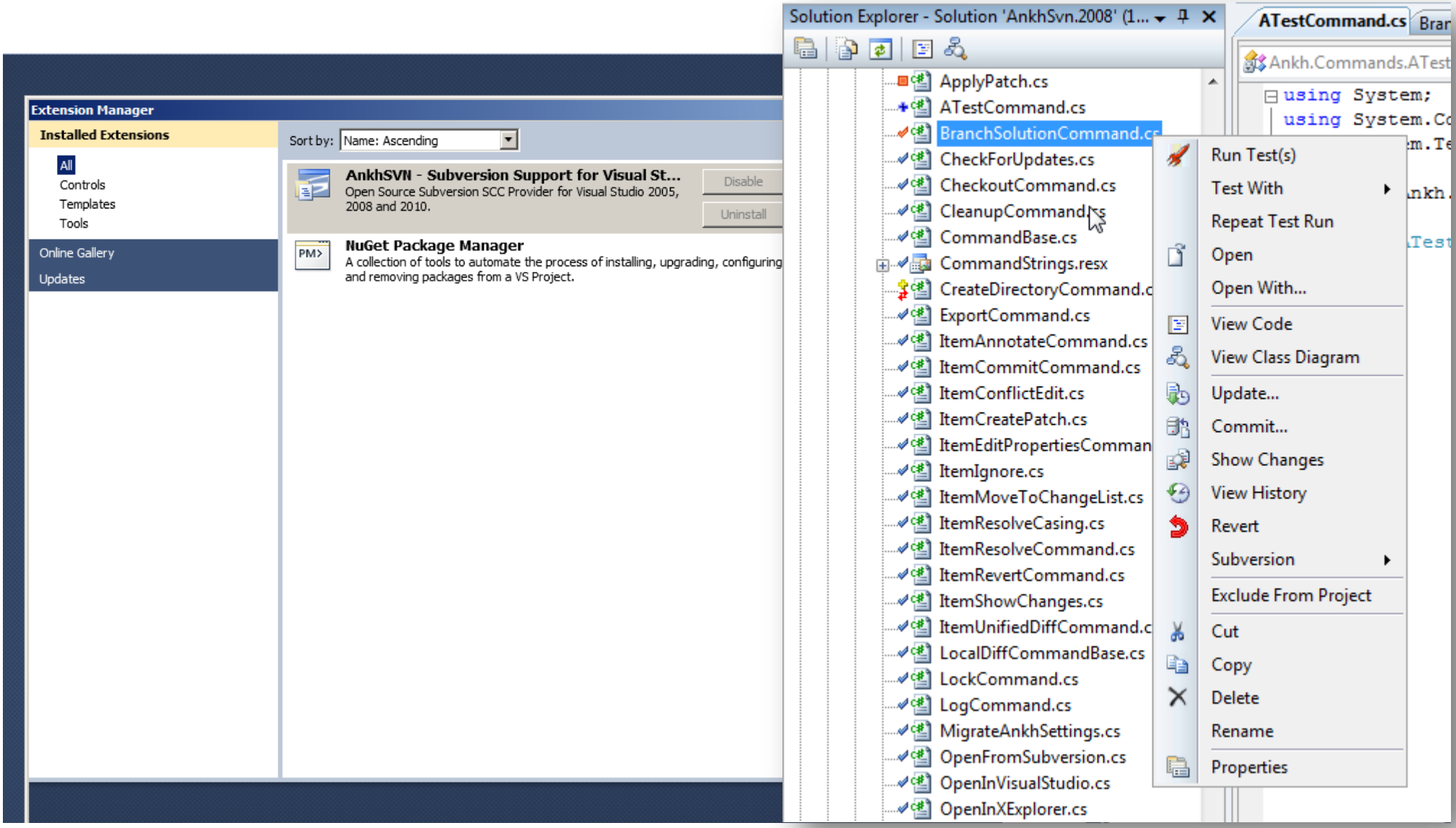
Version Control :

- 圖形化工具：
  - TortoiseSVN 
  - Ankhsvn：Visual Studio的SVN Client軟體
  - Subclipse：Eclipse的SVN Client軟體
  - SVNx：在Mac OS X下的一款Client軟體
  - eSVN：Unix下類似Tortoise SVN的軟體
  - kdesvn：在Linux下使用KDE桌面管理下類似TortoiseSVN的軟體
  - RabbitVCS：在Linux下使用Gnome桌面管理下類似TortoiseSVN的軟體

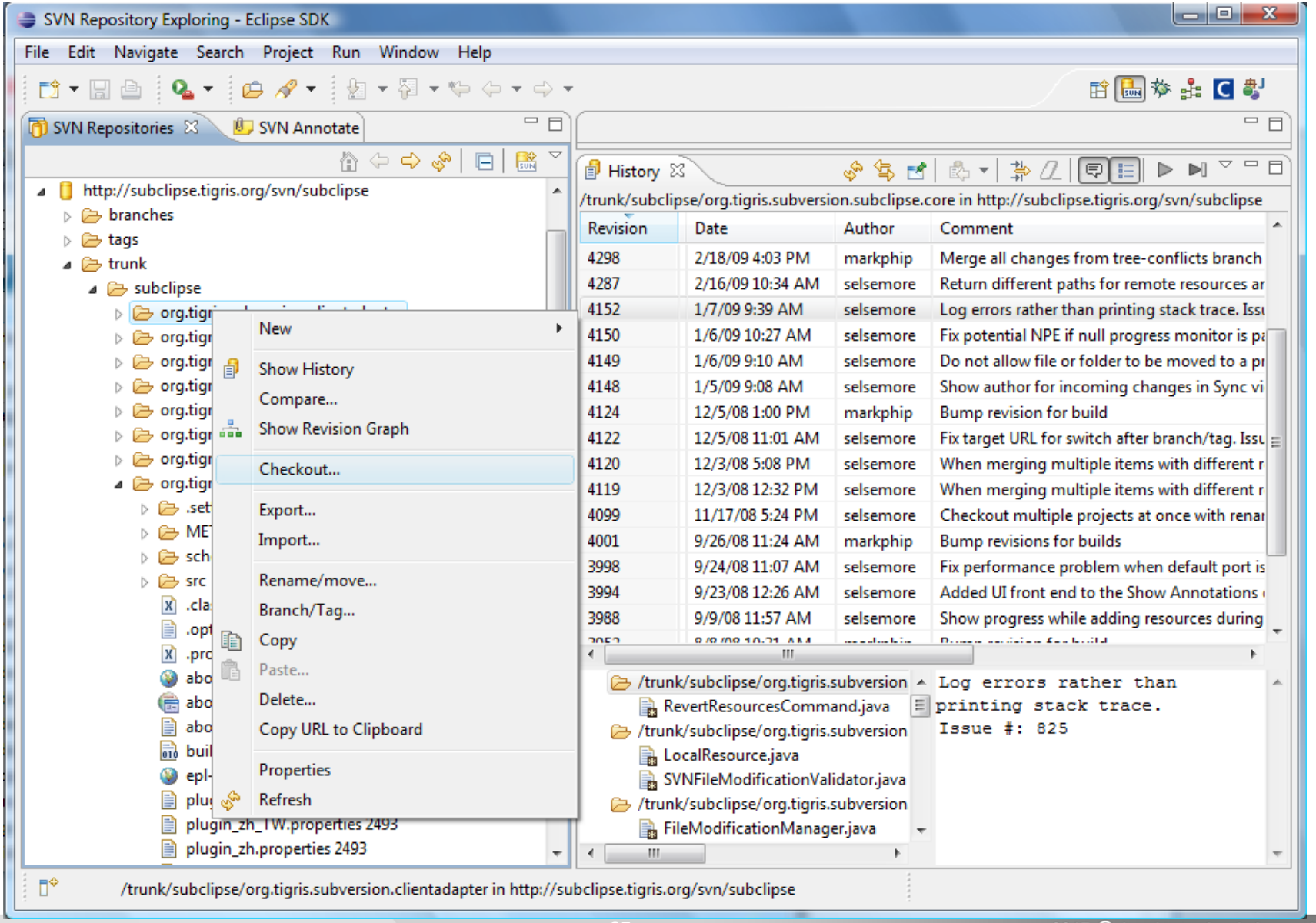
TortoriseSvn Tool



Ankhsvn in Visual Studio



Subclipse in Eclipse



Version Control : git

- Git：分散式版本控制系統
  - 保存更新歷史紀錄，不用另備份
  - 顯示檔案異動內容之差異
  - 舊檔案上傳到伺服器，覆蓋其他人的最新檔案時，系統會發出警告
- 業界狀況：



Version Control : git

基本指令

- 建立 Repository

```
mkdir sandbox
cd sandbox
git init
```

- 第一次commit

```
touch README
git add README
git status
git commit -m "First Commit"
```

- 修改

```
git status
git diff
git add .
#一次加入所有變更跟新增檔案，但不包括刪除的檔案!
git status
git diff --cached
git commit -m "Update README"
```

Version Control : git

基本指令

- 還原

```
# 新增一筆 commit 來做還原
# 例如本來的 commit 是新增一行，那麼 revert commit 就會移除那一行
git revert e37c75787
git revert HEAD^
```

- 比較差異 Diff

```
git diff <SHA1>
git diff <SHA1> <SHA1>
git diff --stat <SHA1>
```

- 忽略不需要追蹤的檔案

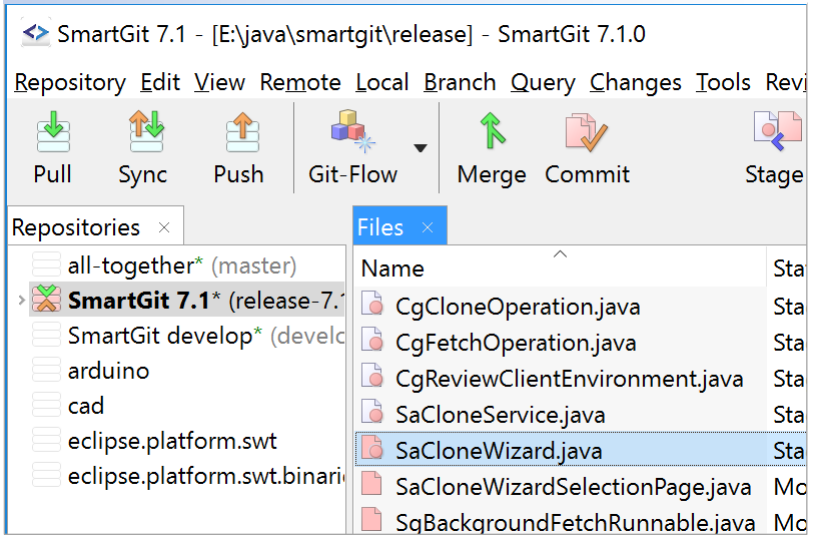
```
# 編輯 .gitignore (屬於專案的設定，會 commit 出去)
# 編輯 ~/.gitignore (你個人的 Global 設定)
空目錄不會 commit 出去，必要時需要放 .gitkeep 檔案
```

Commit 基本原則

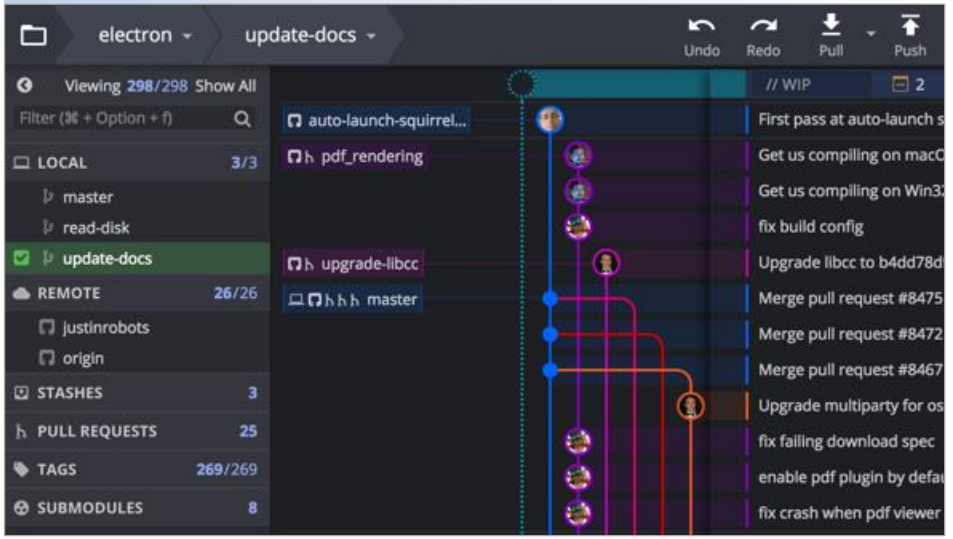
- 適當的粒度/相關性/獨立性
  - 以一個小功能、小改進或一個bug fixed為單位
  - 對應的unit test程式在同一個commit
  - 無相關的修改不在同一個commit
  - 語法錯誤的半成品程式不能commit
- *git diff -check* 可以檢查多餘的空白
- commit 訊息很重要：第一行寫摘要，有需要寫實作細節的話，放第二行之後

Git GUI Tool

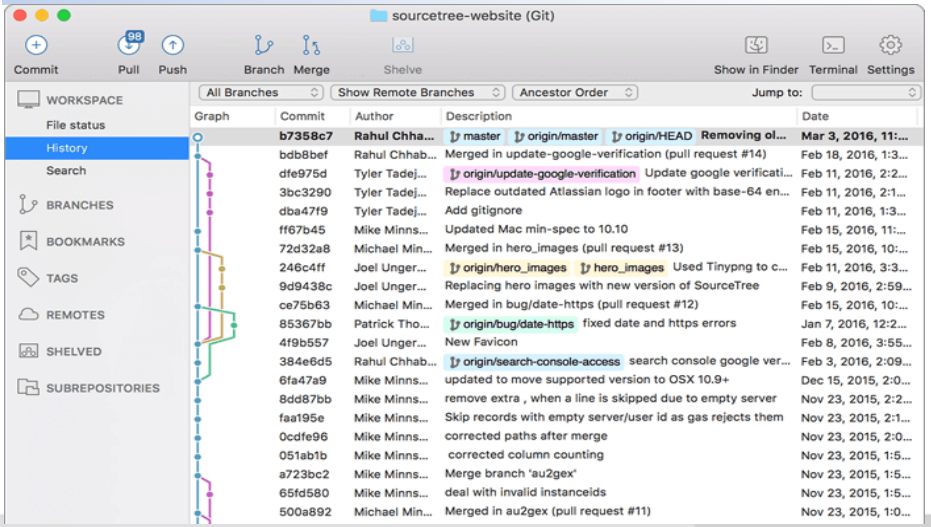
SmartGit (Windows, Mac, Linux)



GitKraken (Windows, Mac, Linux)



SourceTree (Windows, Mac)



GUI Client 軟體參考：
https://git-scm.com/downloads/guis

V.S.

	git	svn
版本控管	分散式 (DVCS) : 減少版本衝突	集中式 (VCS)
權限管理	帳號 (較不嚴謹)	可依個人 / 目錄設定權限
更新還原	可局部更新/還原	通過歷史版本還原
分支(branch)	針對repository作分支，一旦刪除無法復原	可針對子目錄(拷貝)
提交 (Commit)	屬於本地repository的活動 (**`git push`**到主要repositor即可，Git會自動同步(Sync))	直接記錄到中央repository (當更新物件有問題時，無法救回，網路中斷無法提交)
衝突	commit不中斷，自動merge，當發生衝突時，會提示由人工處理	先commit先贏 (建議commit前先update，減少commi錯誤)
內容完整性	較佳 (內容儲存使用SHA-1，減少硬碟故障或網路問題對repository的破壞)	-

Build工具

- Java:

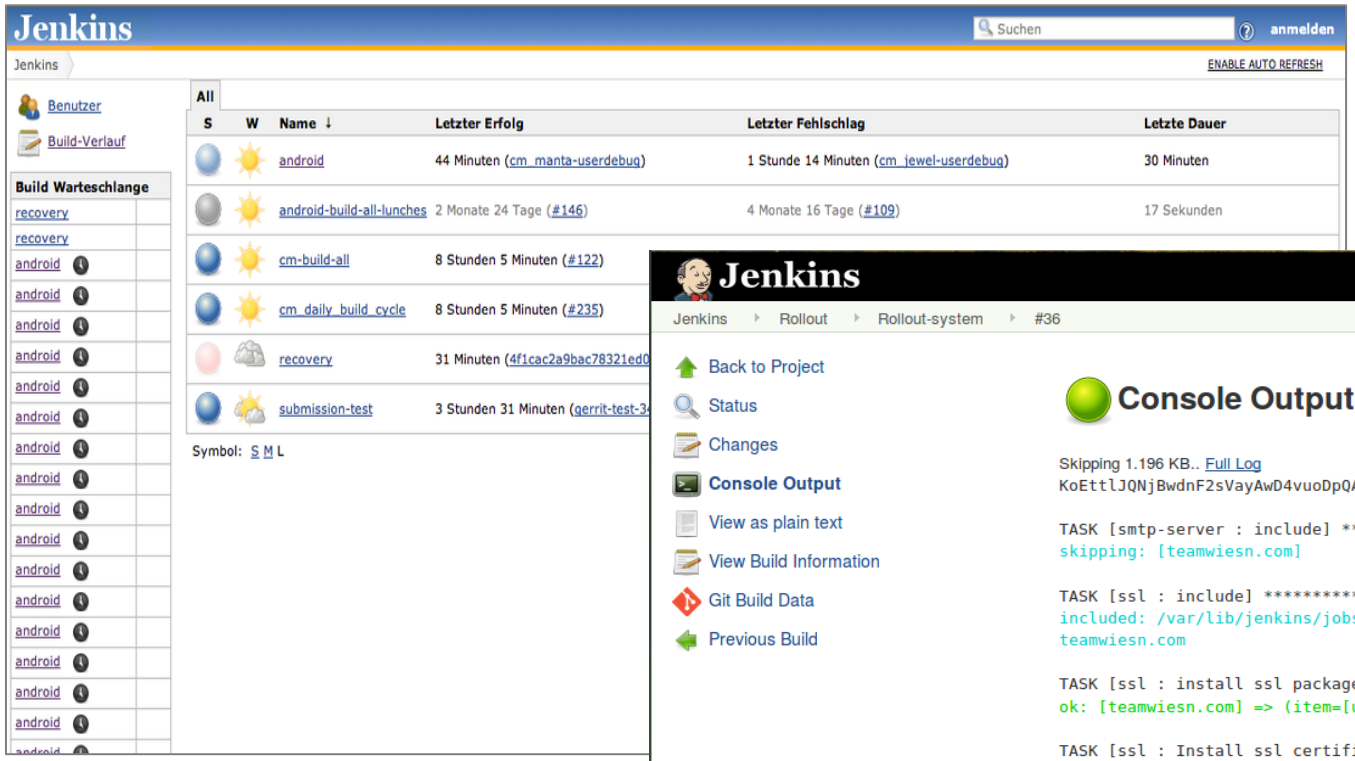
- Javascript :

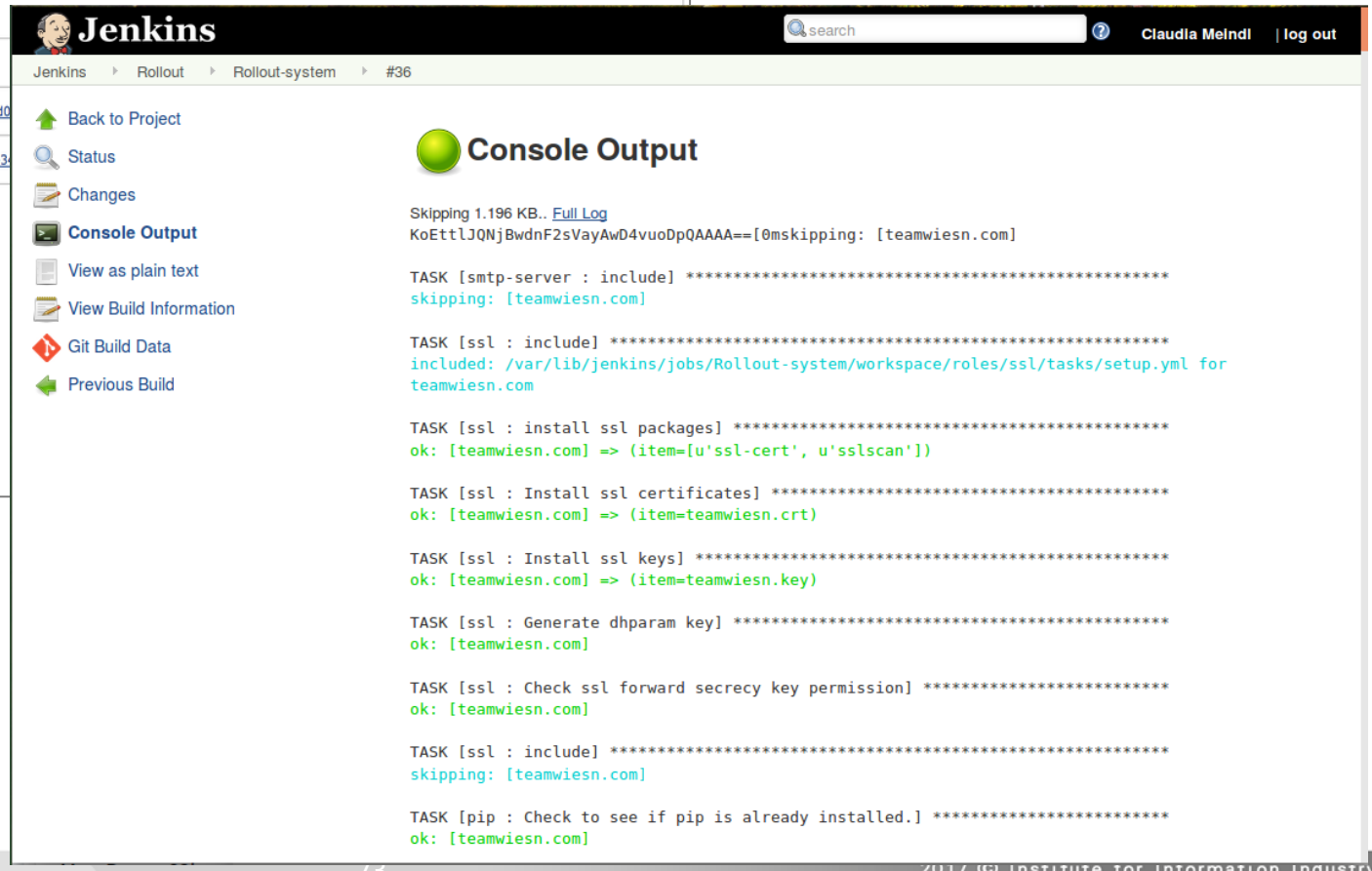
其他工具

- Test Tool & Framework

- CI Server

Jenkins





Travis CI

Travis CI – Free Hosted Co

https://travis-ci.org/alexgorbatchev/node-crc/pull_requests

Travis CI | Home | Blog | Status | Help | We're Hiring! | Travis CI for Private Repositories | Alex Gorbatchev

Search all repositories

My Repositories | Recent | +

- npmawesome/generator-npmawes... 12
  - 27 sec
  - a day ago
- alexgorbatchev/generator-coffee-m... 26
  - 20 sec
  - 9 days ago
- alexgorbatchev/coffee-errors 5
  - 7 sec
  - 16 days ago
- alexgorbatchev/node-crc 37
  - 18 sec
  - about a month ago
- ben-eb/gulp-symlink 72
  - 14 sec
  - about a month ago
- AriaMinaei/pretty-error 19
  - 8 sec
  - about a month ago
- alexgorbatchev/ackmate-parser 1
  - 17 sec
  - 2 months ago

alexgorbatchev/node-crc

build passing

Settings

Verious CRC for node.js and browser

Current | Build History | Pull Requests | Branch Summary

Build	Message	Commit	PR	Duration	Finished
30	updating generated library files to version that support 0x0 as CRC initial value	a348984 (master)	#28	9 sec	about a month ago
18	fix README.md: a synchronous reading of a file is implied	ddfd31b (master)	#21	7 sec	6 months ago
10	Add a test for buffer as parameter	ee37384 (master)	#17	43 sec	8 months ago
9	drop redundant badge	ea3c094 (master)	#19	36 sec	8 months ago
8	Improve performance of CRC#each_byte	ed7ebe4 (master)	#17	48 sec	8 months ago
6	use SVG badges instead of PNG badges	b0e71b3 (master)	#18	1 min 1 sec	8 months ago
5	Support Buffer object as input	a4268e8 (master)	#17	45 sec	8 months ago
3	the repository does not contain a makefile → `npm test` should run	bf20bd1 (master)	#14	25 sec	9 months ago

Show more

LAB 3

LAB 4