

CYBERSEC 2024
臺灣資安大會

5/14^{Tue} – 5/16^{Thu}
臺北南港展覽二館

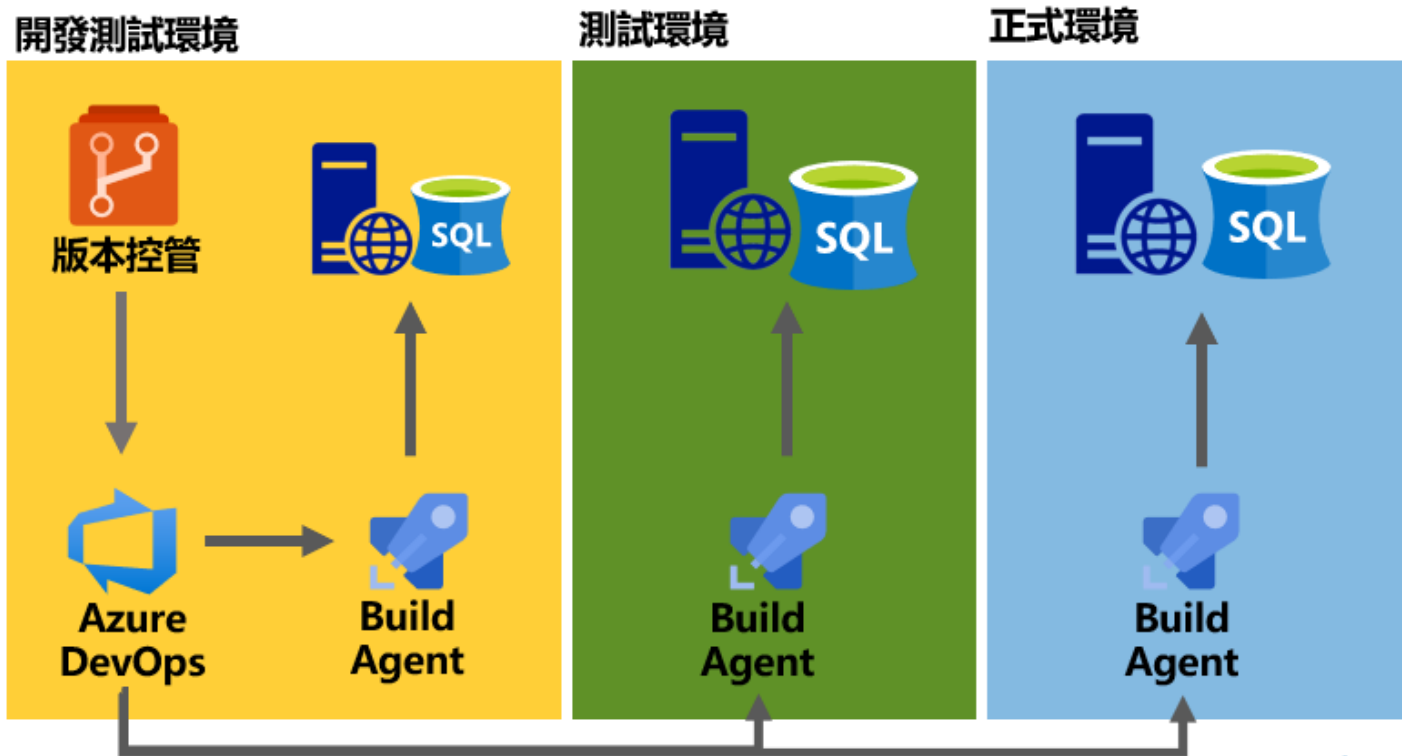
Generative
Future

DevSecOps Forum

公部門小型系統容器化後的 DevSecOps 轉型經驗分享

衛生福利部資訊處楊世鈺

- 公部門版DevSecOps 4個階段
- 容器化後加速自動化的DevSecOps
- 容器化後的維管挑戰



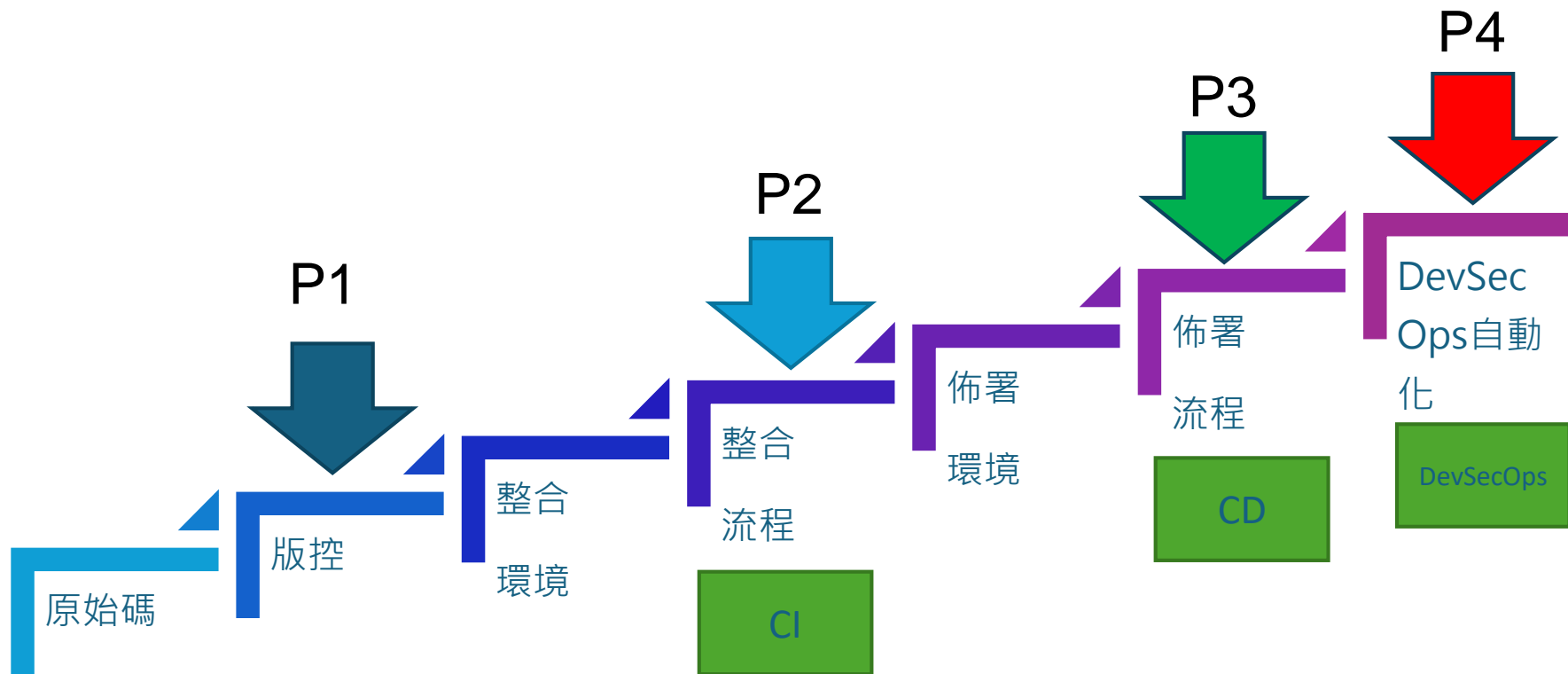
- 契約/廠商背景：
 - 契約要求原始碼提供
 - 契約要求配合開發佈署提供指令
 - 廠商認真配合與詳細說明
- 承辦人背景：
 - 技術熱忱
 - 隨時擔心專案廠商不想做/技術人員斷層/壟斷問題
 - 重建/移機/備份
- 防護基準：中
- RTO：4小時
 - 有紙本作業流程
- RPO：24小時
- 技術規格：
 - Tomcat
 - Java
 - Spring Framework
 - MS SQL Server

- 版控要求
 - 原始碼與版本異動說明文件應於平台功能異動上版前依機關指定方式交付或交付至機關指定之原始碼版控平台
- 容器要求
 - 容器(Container)所使用之映像檔(image)應優先採用簽章等可信任之來源版本，維持開發、測試、正式均採用相同映像檔，以利快速部署測試，映像檔原則應不得放入憑證、連線密碼、組態(configuration)等資料，應採用環境變數(environment)或覆寫檔案等方式於容器啟動時設定為開發、測試或正式環境，映像檔之打包指令亦應包含自動更新機制，並應參考CIS Docker Benchmark實踐容器相關安全性
- DevSecOps自動化流程要求
 - 提供容器化，與配合持續整合/持續部署(CI/CD)運作架構(以Azure DevOps Server Pipeline為主，如使用其他平台需負責建置移轉)，相關流程需整合資安檢測、套件安全與品質檢測機制並通過機關可接受之門檻等級，品質檢測應包含Code Smell可維護性、可靠性等檢測內容

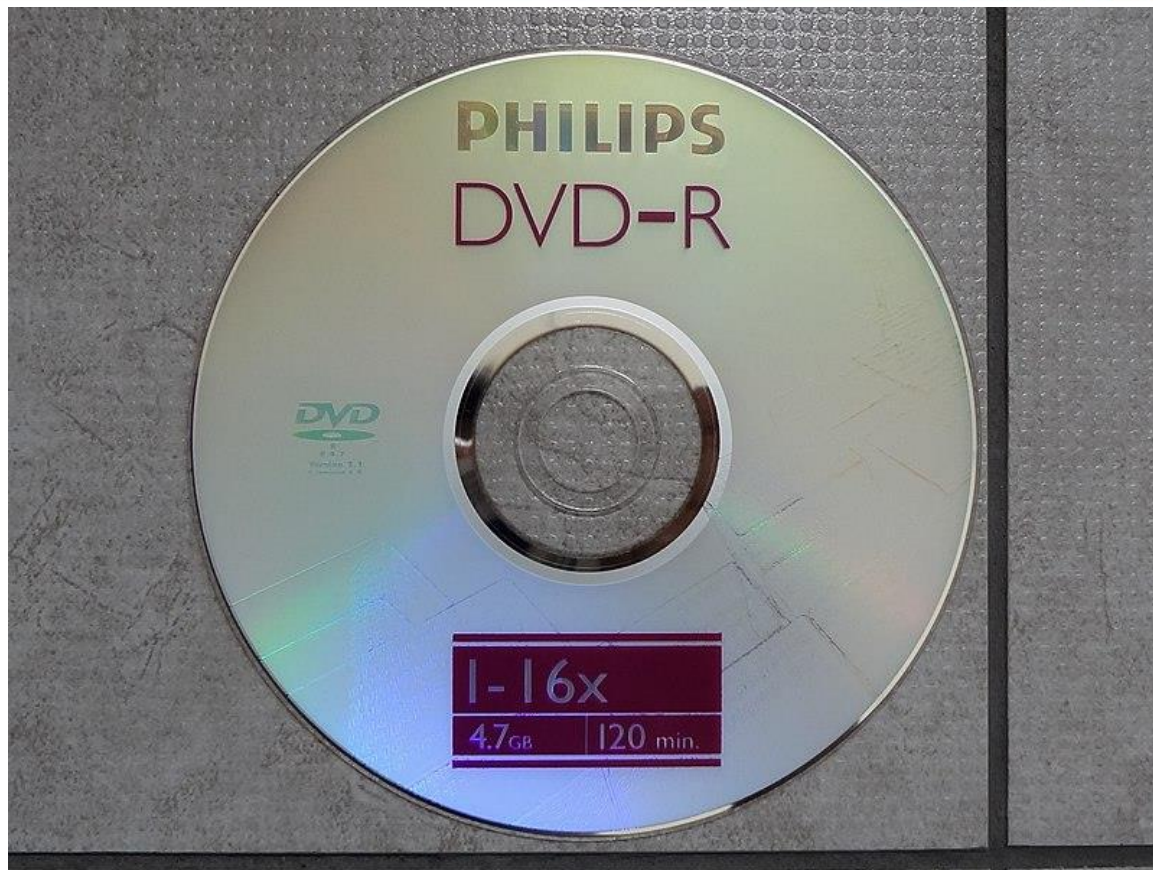
- 指引5.1 網站軟體開發過程宜採用版本控制系統。
- 網站軟體開發工作專業與複雜建議開發工作使用版本控制軟體，進行版本控制管理。
- 參考指南-1軟體開發的過程中，可運用版本控制來追蹤與維護原始碼、檔案，以及設定檔等改動歷程，以確保由不同人所編輯的同一程式檔案都得到同步。以下提供建議作業事項：
 - 擇定並安裝版本控制軟體，例如Git、SVN等版本控制軟體。
 - 訂定版控編號之命名原則，以利專案成員使用正確的原始碼、檔案設定之版本。

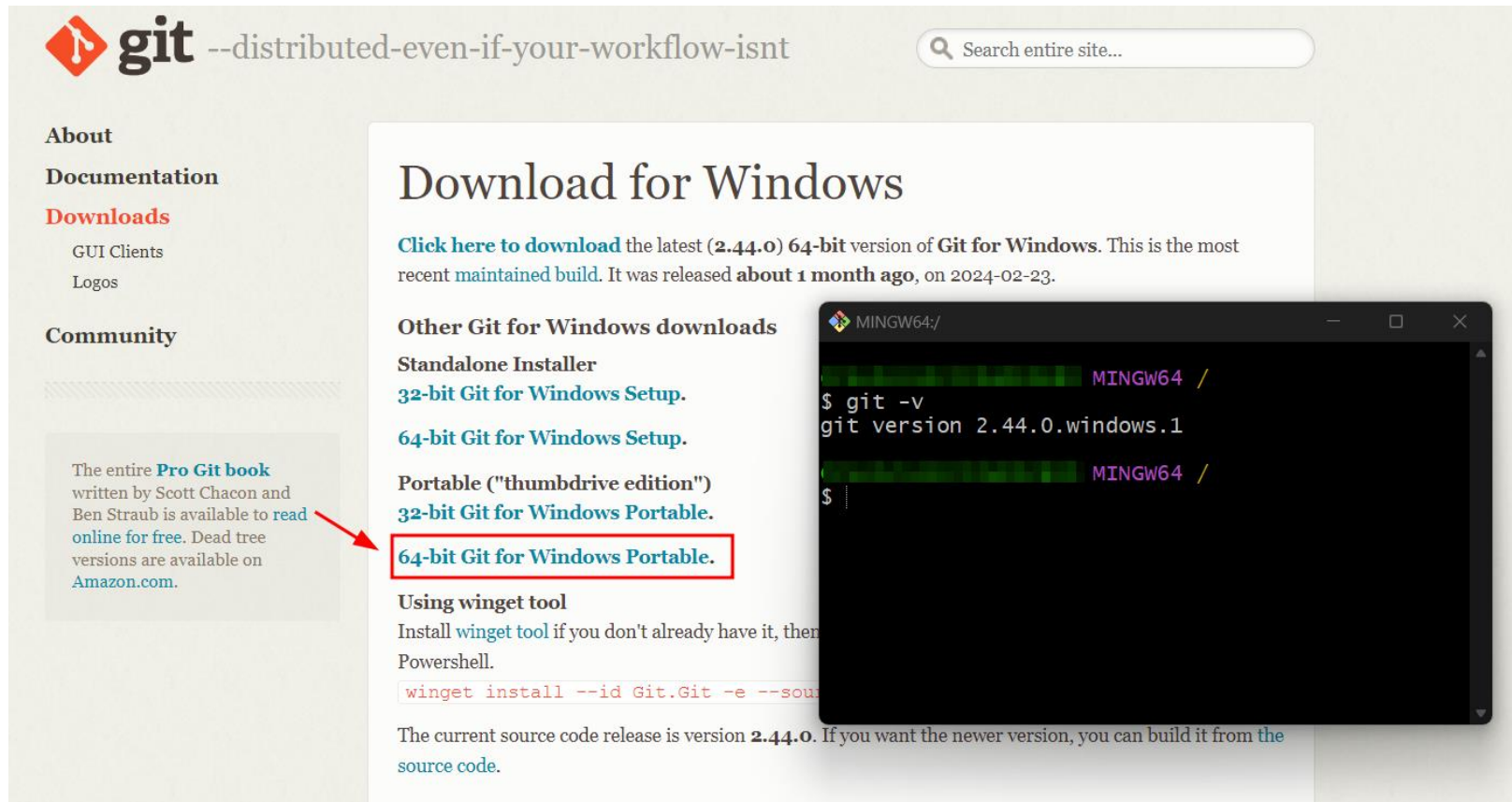
- **指引5.2 提供程式化、自動化的測試，在版本更迭時確保服務品質及一致性。**
- 建議導入程式化、自動化的測試，以利版本更迭時能確保服務品質及一致性。
- 參考指南-1軟體開發的過程中，開發工作可能由不同的開發者或團隊合作進行，可導入自動化測試。以下提供建議作業事項：
 - 使用自動程式碼風格檢查工具，以達到軟體程式寫法風格的一致性。
 - 常見案例製作成測試程式，經由系統化、自動化流程於每次版本更新時進行檢查，減少因人工檢查的疏漏，造成服務缺陷或中斷發生的情形。

公部門版DevSecOps 4個階段



- P1
 - 掌握原始碼版本與差異
- P2
 - 確認運行的原始碼版本與掌握的原始碼版本一致
- P3
 - 掌握運行環境所有參數、憑證與相依軟體版本，可一條龍式掌握原始碼到上版
- P4
 - 自動化腳本，增加維護易用性與減少人工錯誤





git --distributed-even-if-your-workflow-isnt

Search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (**2.44.0**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 1 month ago**, on 2024-02-23.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

Using winget tool

Install [winget tool](#) if you don't already have it, then Powershell.

```
winget install --id Git.Git -e --source
```

The current source code release is version **2.44.0**. If you want the newer version, you can build it from [the source code](#).

```
MINGW64:/
$ git -v
git version 2.44.0.windows.1
$
```

初始化

- 開啟Git Bash
- cd project
- mkdir PROJECT_NAME
- git init
- git add .
- git commit -m "version 1"

後續作業

- 請廠商完整交付
- 到project\PROJECT_NAME
- 刪除資料夾(.git資料夾不要刪除)
- 複製到這個資料夾
- 開啟Git Bash
- Cd project\PROJECT_NAME
- git add .
- git commit -m "change something"

The screenshot displays the Git GUI application. The top menu bar includes File, Edit, View, Repository, Actions, Tools, and Help. Below the menu is a toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, Stash, Discard, and Tag. On the right side of the toolbar are icons for Git-flow, Remote, Terminal, Explorer, and Settings.

The main window is divided into three panes:

- Left Pane:** Contains 'WORKSPACE' (File Status, History, Search), 'BRANCHES' (master), 'TAGS', 'REMOTES', and 'STASHES'.
- Center Pane:** Shows a list of commits under 'All Branches'. The commit 'Set up CI with Azure Pipelines' (7196771) is selected. The table below lists the commit details:

Graph	Description	Date	Author	Commit
o master	push new version	29 一月 2024 11:41	INTRA\CCSHIHUYU <	e642bdb
	create a test too	26 一月 2024 17:10	INTRA\CCSHIHUYU <	5d53f36
	create a test	26 一月 2024 17:09	INTRA\CCSHIHUYU <	37d4f63
	Deleted azure-pipelines.yml	26 一月 2024 16:14	devops_mohw <	52a91ae
	Deleted dockercompose.yml	26 一月 2024 16:13	devops_mohw <	462d322
	Deleted azure-pipelines-3.yml	26 一月 2024 16:13	devops_mohw <	18b36ad
	Deleted azure-pipelines-2.yml	26 一月 2024 16:13	devops_mohw <	d8f2648
	Deleted azure-pipelines-1.yml	26 一月 2024 16:13	devops_mohw <	5aa4dad
	Updated pom.xml	26 一月 2024 15:43	devops_mohw <	5a8bf89
	Set up CI with Azure Pipelines	26 一月 2024 15:25	devops_mohw <	7196771
	Updated pom.xml	26 一月 2024 15:04	devops_mohw <	55c1228
	Updated dockercompose.yml	26 一月 2024 14:32	devops_mohw <	aa21cea
	Updated dockercompose.yml	26 一月 2024 14:28	devops_mohw <	b31cd6e
	Updated dockercompose.yml	26 一月 2024 14:23	devops_mohw <	c2319b5
	Updated dockercompose.yml	26 一月 2024 14:22	devops_mohw <	c953a75

The bottom pane shows the diff for 'azure-pipelines.yml'. The commit information is: Commit: 71967718a5dead0331cf69d1a2696ddc0b1c9f93 [7196771], Parents: 55c122896d, Author: devops_mohw <>, Date: 2024年1月26日 上午 07:25:23, Committer: devops_mohw. The diff shows changes to the 'steps' section, with a new script added (highlighted in red):

```
10 10 ..... name: default
11 11 .....
12 12 ..... steps:
13 13 ..... - script: docker-compose -f dockercompose.yml up -d
14 14 ..... - script: mvn build deploy
15 15 ..... displayName: 'Run a one-line script'
```

1. 安裝OPENJDK
2. 下載MAVEN
3. 設定環境變數
4. 下載程式碼與IIBS，放置於.m2路徑
5. 到pom.xml路徑下執行mvn package
6. 產出target/xxx.war，即可進行tomcat佈署

- 下載Dockerfile

<https://github.com/carlossg/docker-maven/blob/main/eclipse-temurin-8-focal/Dockerfile>

- `docker build -t maven:v1 .` //重新打包image，更新套件
- `docker volume create m2` //建立m2 volume，後續可以加速打包流程
- `cd SOURCE_CODE&POM_PATH` //切換到原始碼與pom.xml路徑
- `docker run --rm -v m2:/root/.m2 -v ./:/root -w /root maven:v1 mvn package` //執行maven 打包指令
- 即可產出target/PROJECT.war



- `docker run --rm -v ./root -w /root --network none anchore/syft PROJECT.war`
//直接使用docker hub image，採無網路狀態進行檢測

```
ted@ted-ThinkPad-T470p:~/下載$ docker run -v ./root -w /root --network none --rm anchore/syft Co
unterWebApp.war
NAME                VERSION          TYPE
CounterWebApp       1.0-SNAPSHOT     java-archive
aopalliance          1.0              java-archive
hamcrest-core        1.3              java-archive
jcl-over-slf4j        1.7.5            java-archive
jstl                 1.2              java-archive
junit                4.11             java-archive
logback-classic      1.0.13           java-archive
logback-core         1.0.13           java-archive
slf4j-api            1.7.5            java-archive
spring-aop            4.1.1.RELEASE    java-archive
spring-beans          4.1.1.RELEASE    java-archive
spring-context        4.1.1.RELEASE    java-archive
spring-core           4.1.1.RELEASE    java-archive
spring-expression     4.1.1.RELEASE    java-archive
spring-web            4.1.1.RELEASE    java-archive
spring-webmvc         4.1.1.RELEASE    java-archive
```



- `docker volume create grypedb`
- `docker run -it --rm -v grypedb:/.cache/grype/db anchore/grype db update`
- `cd PROJECT_WAR_PATH`
- `docker run -it --rm -v grypedb:/.cache/grype/db -v ./:/root -w /root --network none -e "GRYPE_DB_AUTO_UPDATE=false" anchore/grype PROJECT.war --only-fixed`



grype

- `docker volume create --name sonarqube_data`
- `docker volume create --name sonarqube_logs`
- `docker volume create --name sonarqube_extensions`

- `docker network create my-net`

//讓sonarqube server跟scanner cli同一個網路，要先建一個網路物件

- `docker run -d --name sonarqube \`
 `-p 9000:9000 \`
 `-v sonarqube_data:/opt/sonarqube/data \`
 `-v sonarqube_extensions:/opt/sonarqube/extensions \`
 `-v sonarqube_logs:/opt/sonarqube/logs \`
 `--network=my-net \`

sonarqube:10.5-community

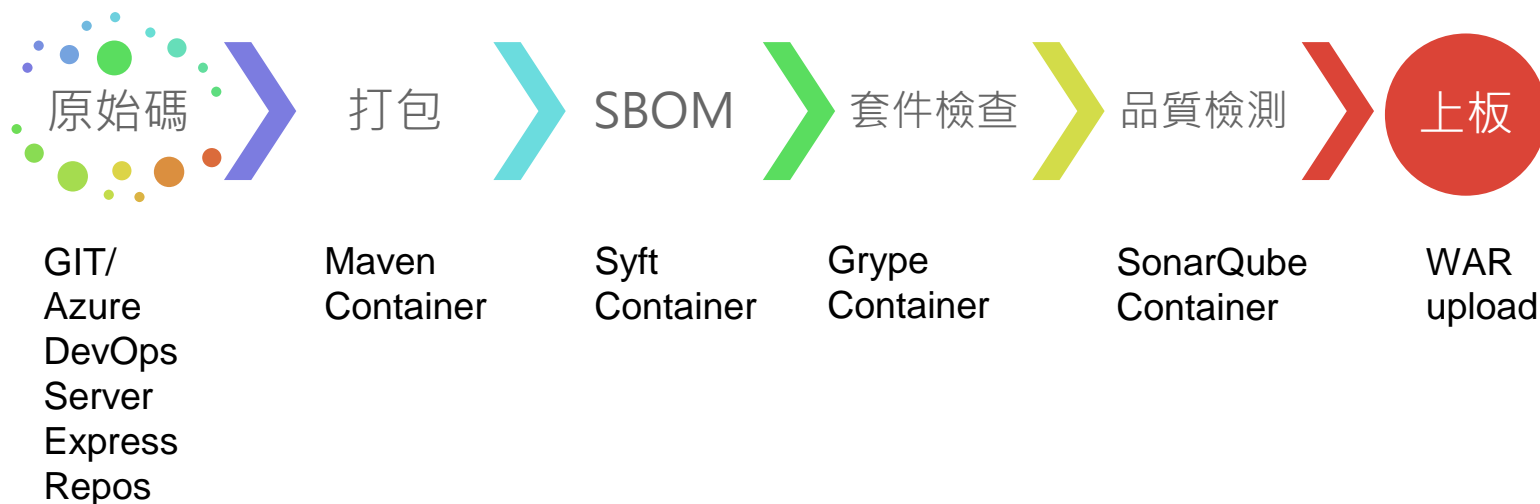
//sonarqube使用的版本請先在官網研究



//接下來請登入0.0.0.0:9000，建立project與產製token，project與token加入Pipeline Library
Variable groups，token即可加密

- `cd PROJECT_TARGET` //進入完成打包的位置
- `docker run \`
 `--rm \`
 `-e SONAR_HOST_URL="http://sonarqube:9000/" \`
 `-e SONAR_SCANNER_OPTS="-Dsonar.projectKey=$(project)" \`
 `-e SONAR_TOKEN=$(token) \`
 `-v ".:usr/src" \`
 `--network=my-net \`
 `sonarsource/sonar-scanner-cli`

//上述的projectKEY跟token請自己進入網站產製
//後續可以進sonarqube網站檢視報告



Azure DevOps DefaultCollection / [redacted] / Pipelines

搜尋

H [redacted] +

概觀
Boards
Repos
Pipelines
管線
環境
版本
程式庫
工作群組
部署群組
Test Plans
Artifacts
專案設定

工作 變數 觸發程序 選項 歷程記錄 | 儲存並排入佇列 捨棄 摘要 排入佇列 ...

下載web_build.xml安全檔案
下載安全檔案

下載[redacted].properties安全檔案
下載安全檔案

下載spring-web-config.xml安全檔案
下載安全檔案

下載WebProperty.java安全檔案
下載安全檔案

maven
命令列

SBOM
命令列

Grype
命令列

sonarqube
命令列

發行成品: [redacted]-web
發佈組建成品

發行成品: msjh.ttc
發佈組建成品

名稱 *

[redacted]-CI

代理程式集區 ⓘ | 集區資訊 | 管理

Default

參數 ⓘ

這個管線沒有任何管線參數。請加以建立，以在工作之間共用最重要的設定，而且在一處即可加以變更。

深入了解

1. 準備WAR與SQL語法
2. 備份資料庫/應用系統
3. 停用tomcat
4. 備份webapps
5. War檔放置到webapps
6. 在tomcat/conf置換server.xml(deploy使用)
7. 執行tomcat
8. 停用tomcat
9. 在tomcat/conf置換server.xml(run使用)
10. 覆蓋WEB-INF/classes全部
11. 覆蓋WEB-INF/相關properties與web.xml
12. 執行tomcat，進行功能測試

1. 設定運行環境(Linux/Docker)符合GCB/Security Harden
2. 分離組態檔案、pfx憑證檔、帳號密碼、token等安全敏感資料，不版控，進pipeline安全檔案管理
3. 使用volume或網路掛載、傳送方式進行persistent file存放
4. 撰寫Dockerfile，將WAR打包進tomcat image
5. 撰寫Docker Compose YAML，將image與env分離撰寫
6. 安全檢測
7. 使用docker save/load進行image交換
8. 使用Docker Compose deploy

作業系統

- OpenSCAP
- DISA SCC
- CISOfy/lynis

容器

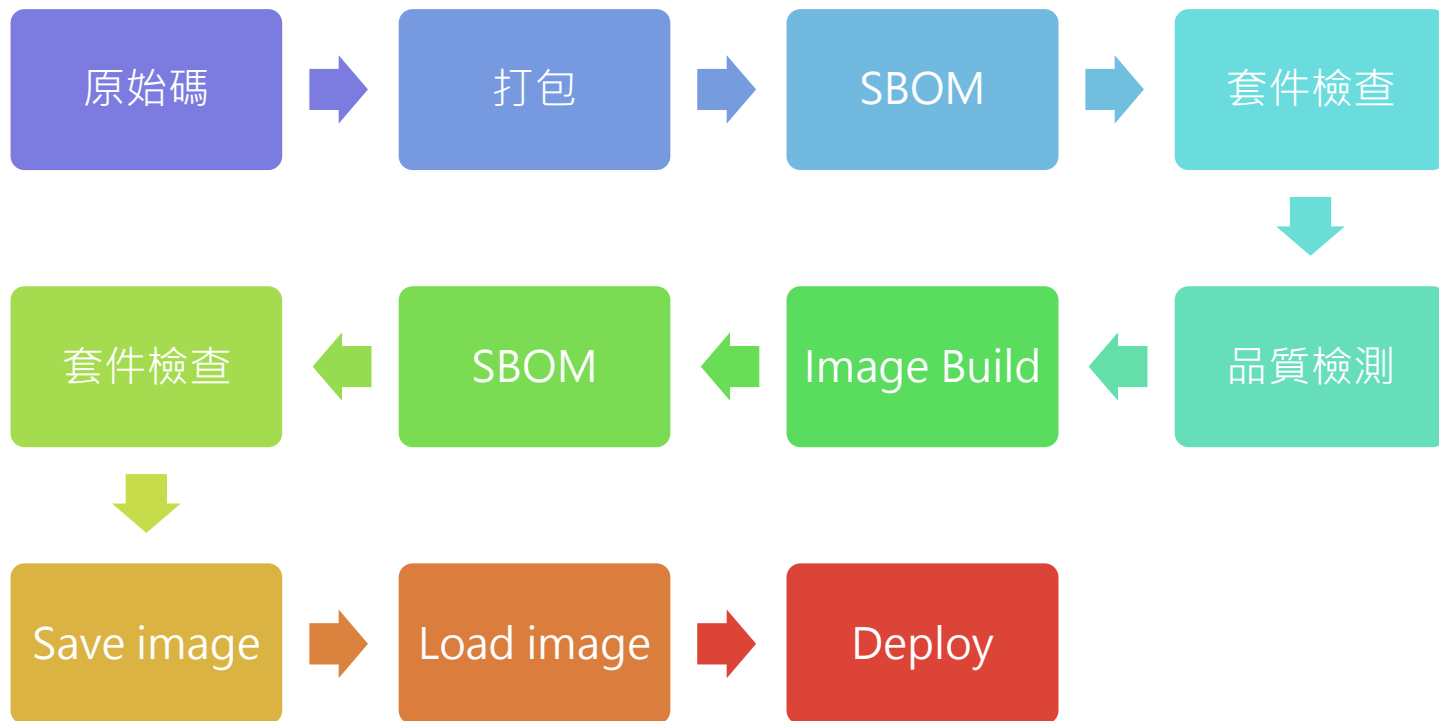
- docker-bench-security
- podman-security-bench

映像檔

- anchore/Syft
- anchore/Grype
- Trivy on Harbor
- SUSE NeuVector

Dockerfile指令

- SonarQube 10.1之後版本
- OpenText Fortify



Azure DevOps DefaultCollection / Pipelines / 版本 / imagebuilder

搜尋

儲存 + 版本 檢視發行 ...

所有管線 > imagebuilder

管線 工作 變數 保留 選項 歷程記錄

ImageBuilder
部署處理程序

Agent job
對代理程式執行

下載安全檔案server.xml
下載安全檔案

下載安全檔案web_deploy.xml
下載安全檔案

image build
命令列

SBOM
命令列

Grype
命令列

命令列 ⓘ

檢視 YAML 移除

工作版本 2.*

顯示名稱 *

image build

指令碼 *

```
copy $(Dockerfile.secureFilePath) .  
copy $(serverxml.secureFilePath) .  
copy $(webdeployxml.secureFilePath) .  
docker build -t :v$(Release.DeploymentID) .  
docker save :v$(Release.DeploymentID) > :tar
```

進階

控制選項

環境變數

輸出變數

專案設定



成品 | + 加入



CI



未設定排程

階段 | + 加入



ImageBuilder

1 個作業, 5 個工作



Stage Test

1 個作業, 6 個工作



Stage Pro

1 個作業, 6 個工作



容器化優勢

- 程式改版、應用系統更新、映像檔更新
- 憑證、帳號密碼更新
- 備份
 - IaC相關組建檔案備份取代完整備份
- 備援
 - 快速部署取代熱備援機制

容器化仍需面對議題

- 作業系統更新
 - Container專用OS
- 容器系統更新
- 日誌
- 合規
 - GCB?防毒?EDR?
- 資安鑑識
- 監控

- 容器操作甚至Linux均非政府資訊廠商主流
- 推廣?至少把原始碼版控做好
- 先從整合部署腳本化開始

Q&A