

容器架構維運探討 期末報告

計畫主持人：鮑興國

台灣科技大學資訊工程系

計畫共同主持人：查士朝

台灣科技大學資訊管理系

台灣科技大學資通安全研究與教學中心

大綱

- 計畫目標
- 容器架構的特色與適用性研究成果摘要
- 容器架構的導入策略期中成果摘要
- 容器架構的比較框架期中成果摘要
- 整理與建議

簡報大綱



計畫背景與目標

容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議



計畫背景與目標

容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議

團隊成員



Project Leader

Hsing-Kuo Pao
Professor and Chairman
Dept. of CSIE, NTUST
Co-Director



Co-Project Leader

Shi-Cha Cha
Professor and Chairman
Dept. of IM, NTUST
Director,
TWISC@NTUST



Yi-Hsuan Hung
(洪蕙璇)



Yi-Xin Li
(李奕欣)



Yi-Zhen Zhang
(張宜楨)



Yan-Sing Ji
(紀彥興)



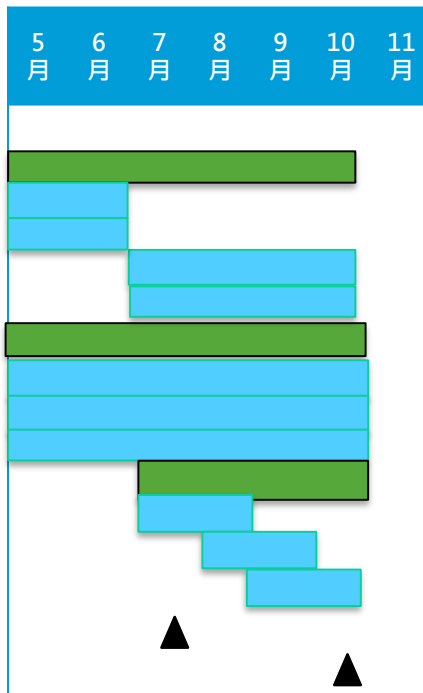
Yen-Chia Chen
(陳彥家)



Jie-Yao Tang
(湯傑堯)

重要工作時程

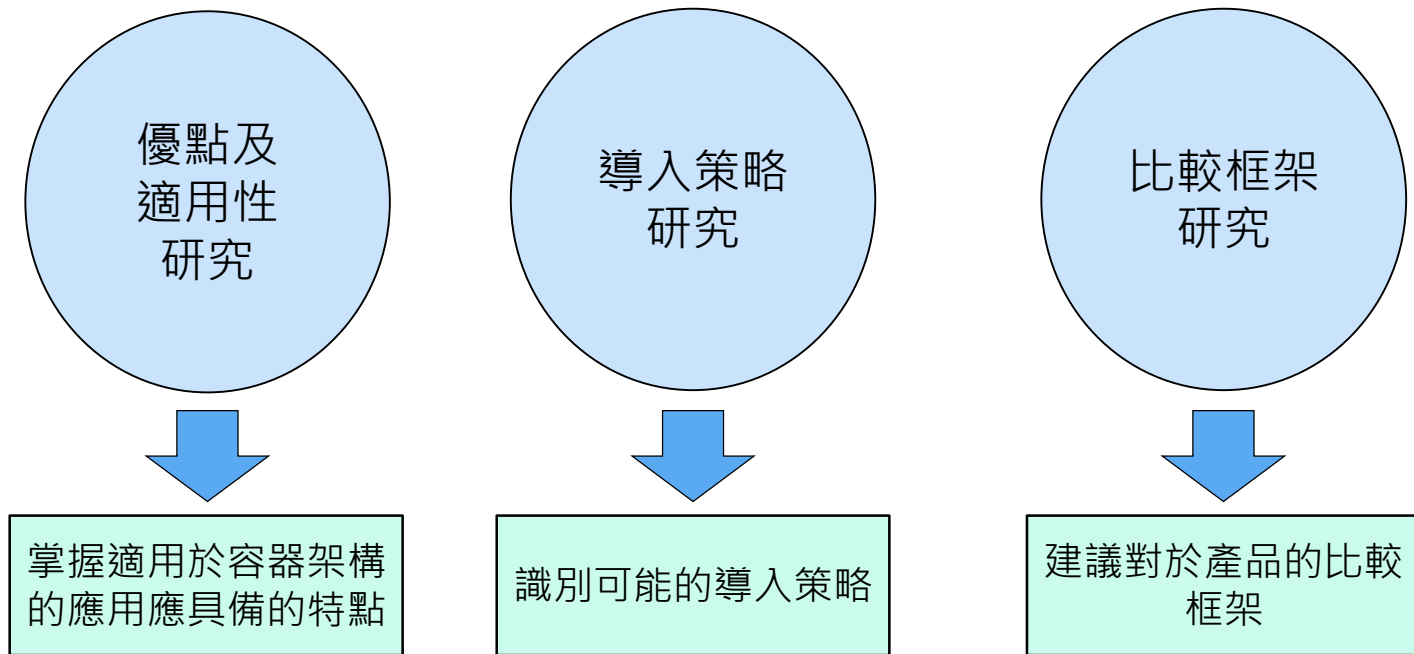
- 1.優點與適用性研究
 - 1.1. 分析基於容器架構之優點以及原因
 - 1.2 區隔容器架構與雲原生架構其他技術的關聯
 - 1.3 設計優點與缺點的驗證案例與實驗
 - 1.4 彙整容器架構的適用情境
2. 導入策略研究
 - 2.1 收集相關最佳實務
 - 2.2 了解 貴單位的現況
 - 2.3 產出對於 貴單位導入的建議
3. 比較框架研究
 - 3.1識別容器架構評選指標
 - 3.2將指標套用到現今主流產品
 - 3.3彙整評比結果
4. 期中報告
5. 期末報告



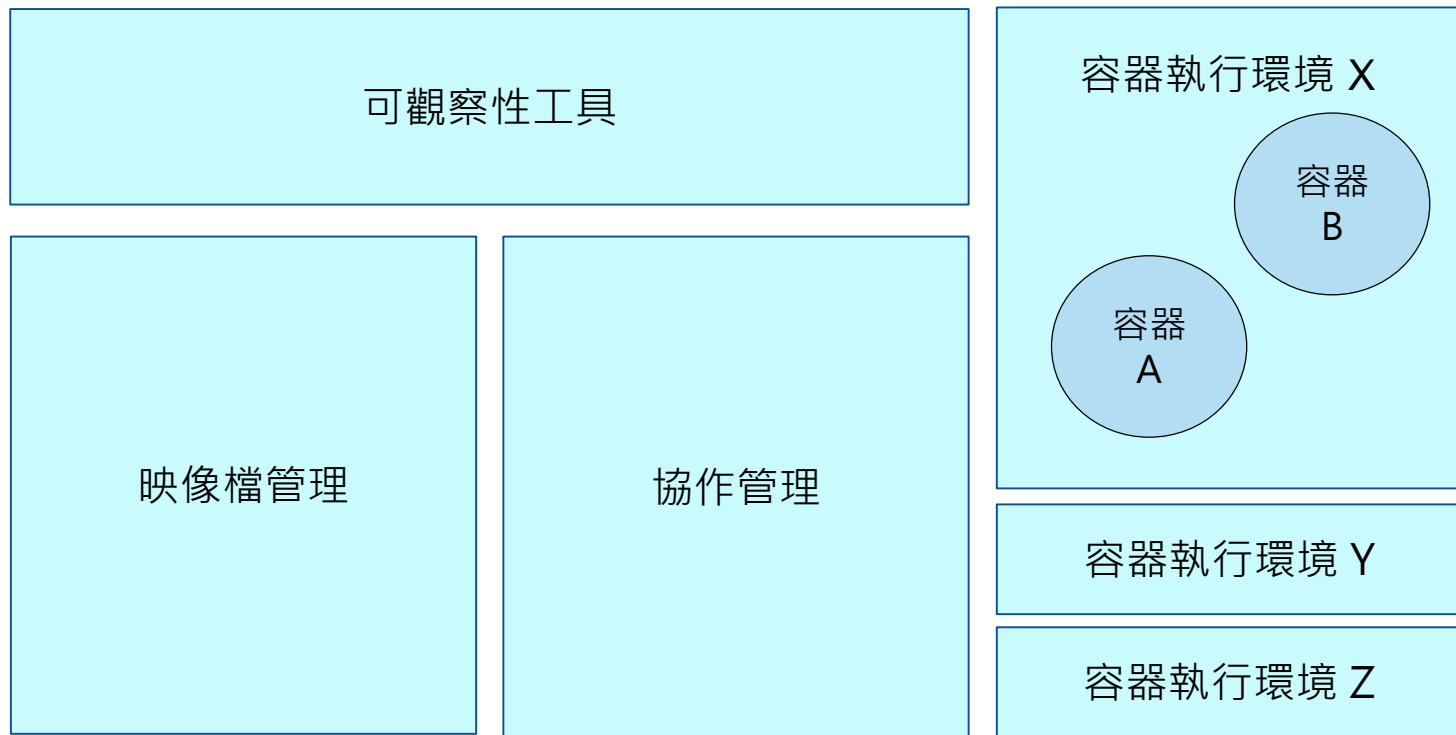
期中報告初稿	111 年7月 15日前
期末報告初稿	111年 9月 30日前
期末報告	111年11月 15日前

目標

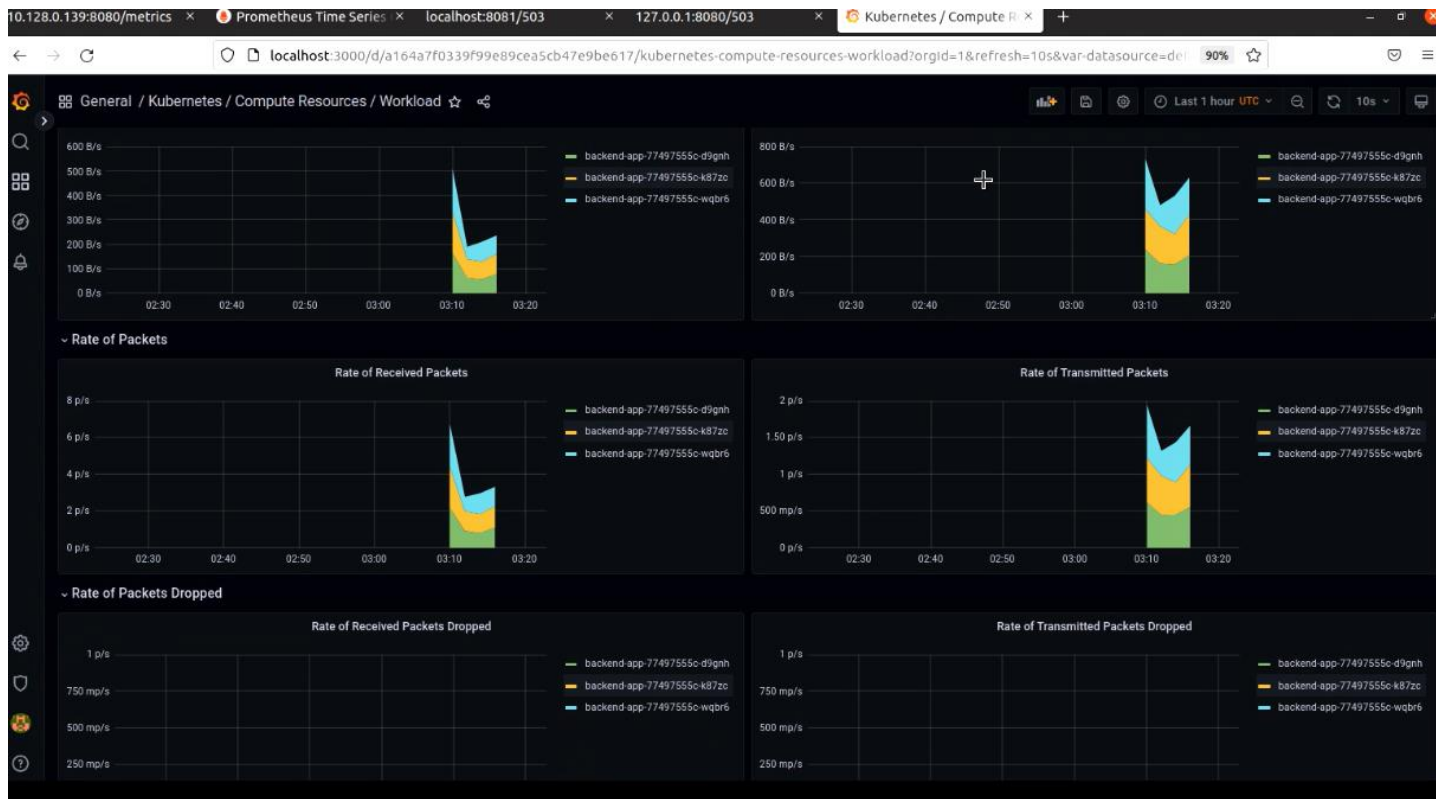
- 協助 貴公司掌握使用容器架構進行維運的最佳實務與導入策略



容器架構



容器架構的概念展示





計畫背景與目標

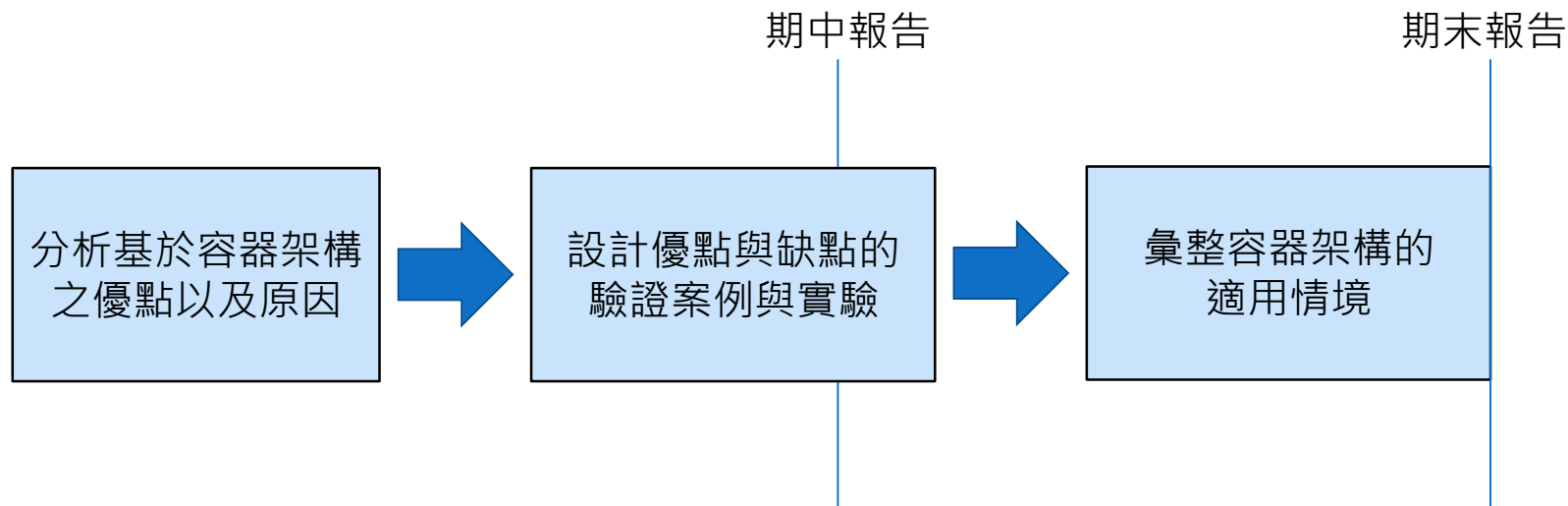
容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議

容器架構的特色與適用性研究方法



容器架構一般被認為有以下好處

資源利用率提高

規模可擴充性提高

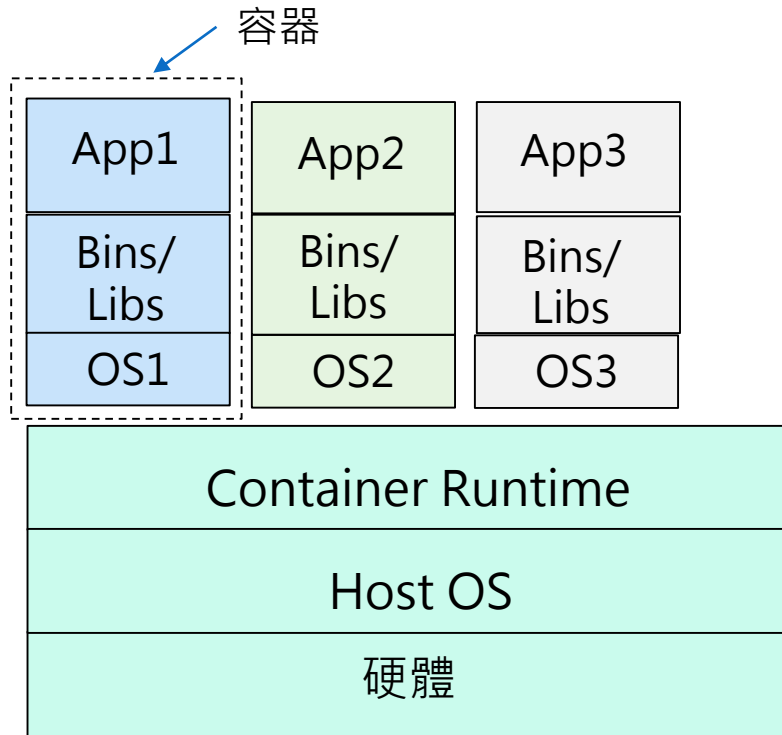
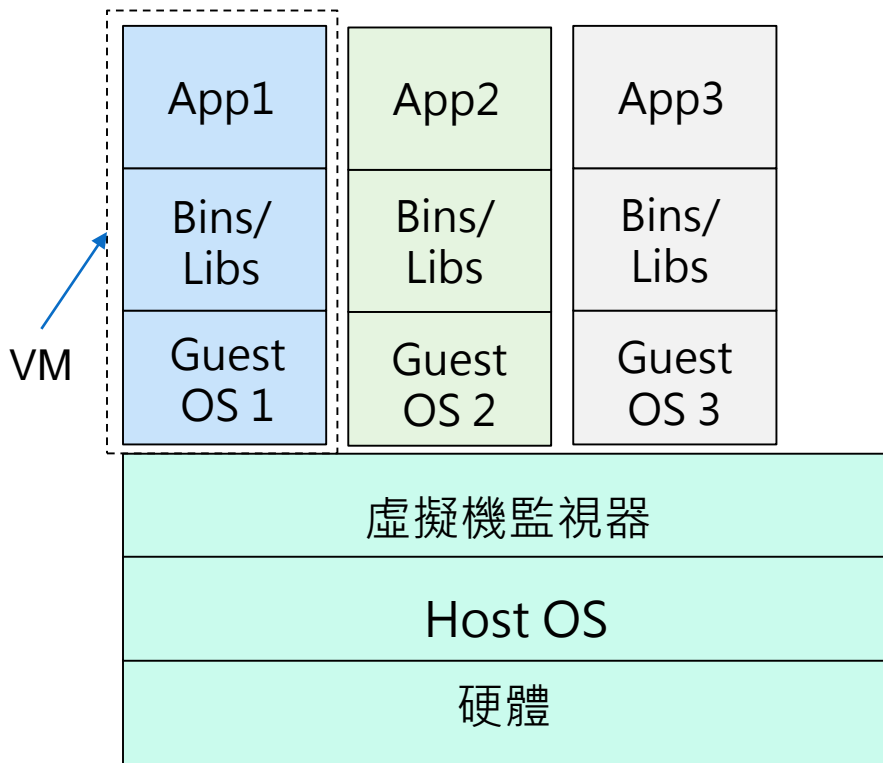
部署速度加快

容錯能力提高

可維護性提高

容器架構與資源利用率 (1/2)

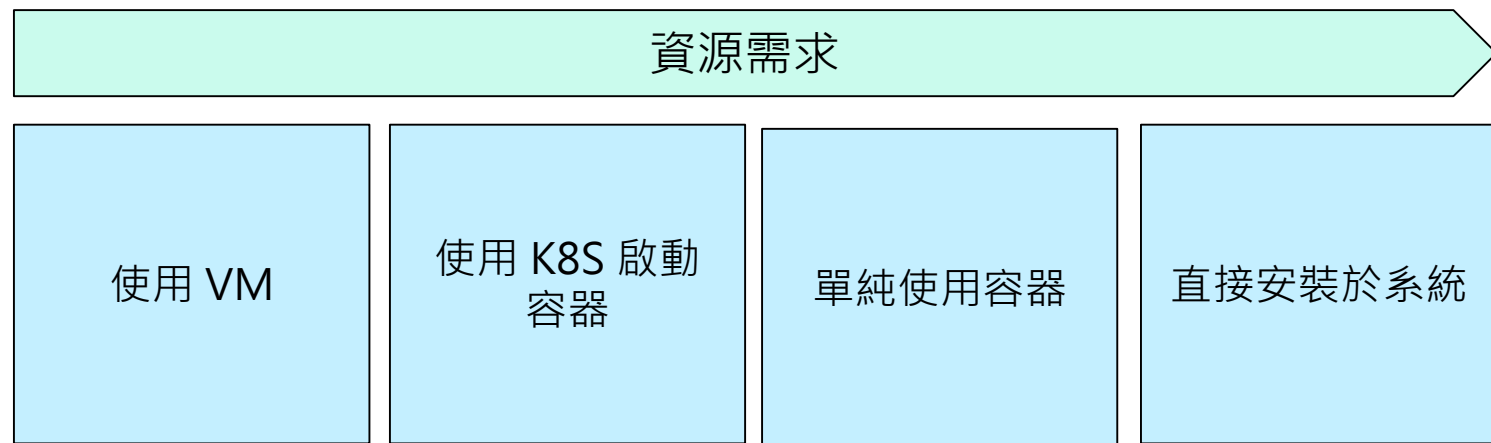
相較於使用虛擬機，容器所需要的作業系統與資源相對較小



容器架構與資源利用率 (2/2)




大

小



- 由此角度較能發揮容器架構效益的情形
 - 資源有限且需要利用有限資源按需求調配資源運用
 - 會需要限制單一服務的資源使用

- 不見得會發揮容器架構效益的情況
 - 資源充足不需動態調配
 - 程式不適用於容器架構

名稱	修改日期
classes	2022/8/4
docker	2022/8/4
generated-sources	2022/8/4
generated-test-sources	2022/8/4
maven-archiver	2022/8/4
maven-status	2022/8/4
surefire-reports	2022/8/4
test-classes	2022/8/4
 simpleshopmysql-0.0.1-SNAPSHOT.jar	2022/8/4 上午 12:19 Executable Jar File 37,985 KB
 simpleshopmysql-0.0.1-SNAPSHOT.jar.original	2022/8/4 上午 12:19 ORIGINAL 檔案 19 KB
 simpleshopmysql-0.0.1-SNAPSHOT-docker-inf...	2022/8/4 上午 12:20 Executable Jar File 4 KB

Images on disk

Last refresh: 3 minutes ago

3 Images

1.65 GB total size

0 Bytes / 1.65 GB in use

Clean up

Images

Give Feedback

LOCAL

REMOTE REPOSITORIES

Search

In use only

NAME ↑	TAG	IMAGE ID	CREATED	SIZE
mysql	8.0.26	9da615fced53	about 1 year ago	513.81 MB
shichoc/mysqlshop	latest	5dc116ea5d2d	3 months ago	447.3 MB
shichoc/simpleshopmy...	0.0.1-SNAPSHOT	d15e3585ac11	3 months ago	693.06 MB

shichoc/simpleshopmysql

RUN ▶

容器架構與規模可擴充性

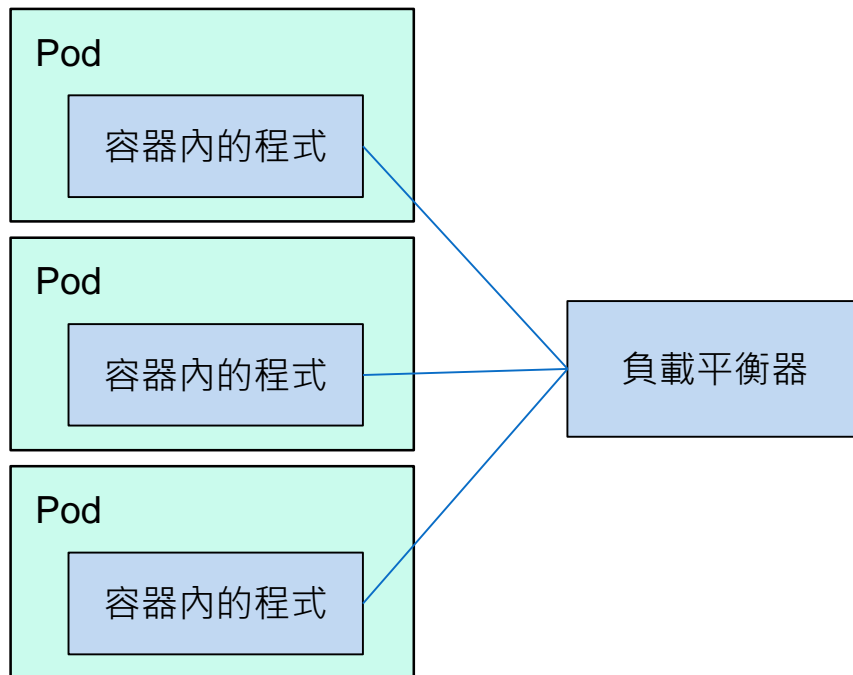
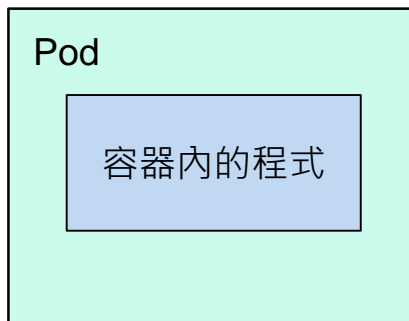
- 容器架構的規模可擴充性理論上來自：
 - 可以快速的配置資源給需要的服務
 - 可以將要求平行交給多個節點處理
- 在此角度上可能發揮效果的應用
 - 服務種類很多，可能突然有服務需要額外配置額外節點
- 不見得會有那麼大效果的應用
 - 服務需求可預測且量在既有資源下可負荷
 - 如果是有狀態服務，以致於啟動新的節點有很高的同步成本

實驗一：平行
處理效果實驗

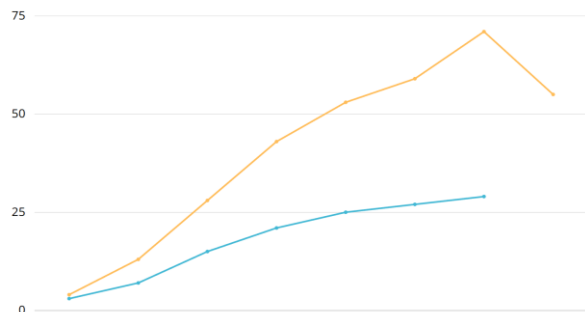
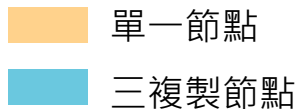
實驗二：規模可
擴充性實驗

容器架構與規模可擴充性：實驗一

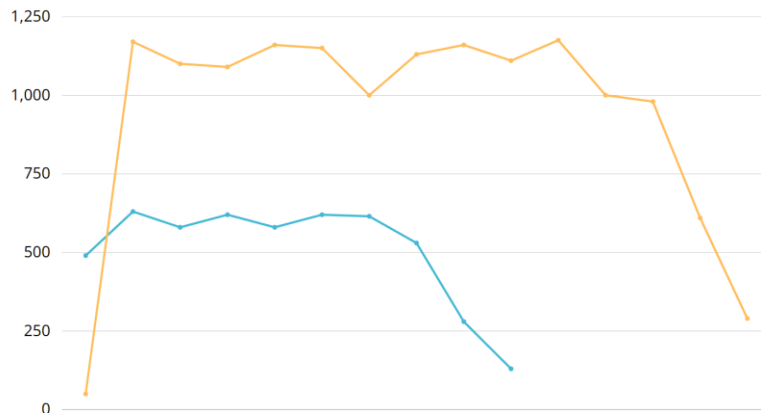
- 比較單一節點及三複製節點



容器架構與規模可擴充性：實驗一結果之回應時間



60 秒內一萬個要求



5 秒內一萬個要求

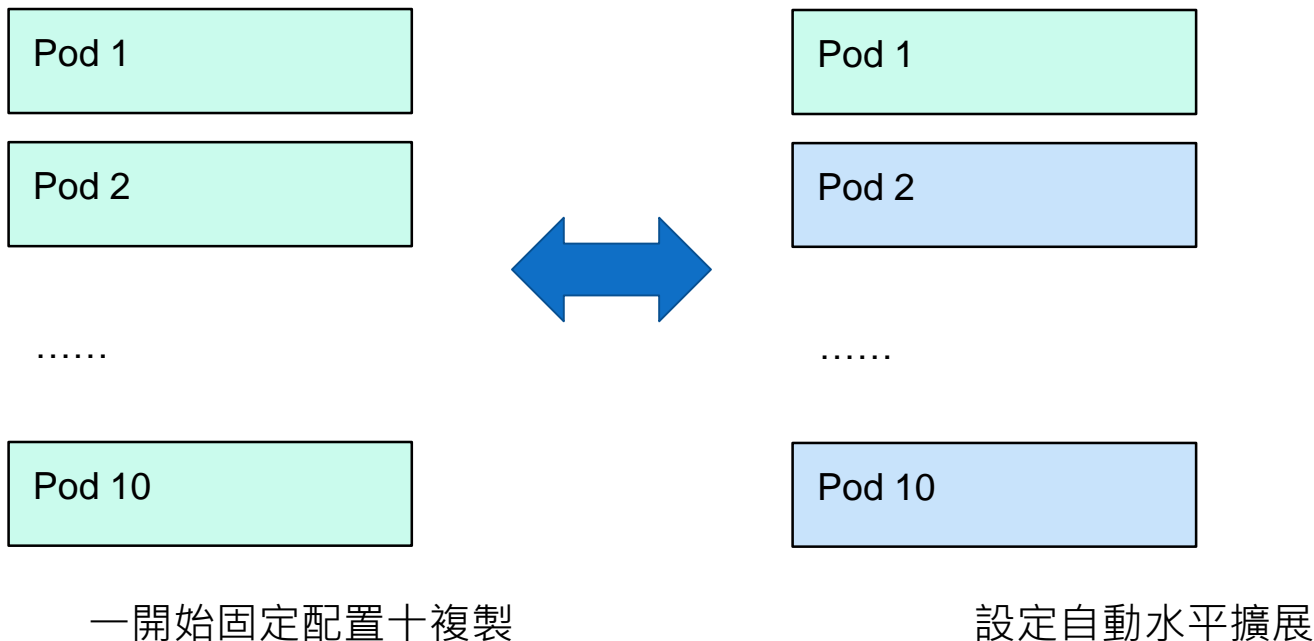
容器架構與規模可擴充性：實驗一結果摘要

- 使用三複製節點稍微提升記憶體使用量，約莫在5%以內。
- 使用三複製節點小幅度提升CPU使用量，會隨著及時Request的數量提升而提升幅度。
- 三複製節點在全部的測試環境下都減少了約100%的Response Time。
- 三複製節點在Response Throughput上大約都提升了50%以上。
- 單一容器運行時當Request達到50000/10s以上，便會出現0.3%的Error。
- 三複製節點穩定性遠高於使用單一容器提供服務。
- 三複製節點在Response Time的標準差在所有環境下都小於單一容器約100%~300%。

合於預期

容器架構與規模可擴充性：實驗二

- 水平自動擴展 (Horizontal Automatically Scaling)

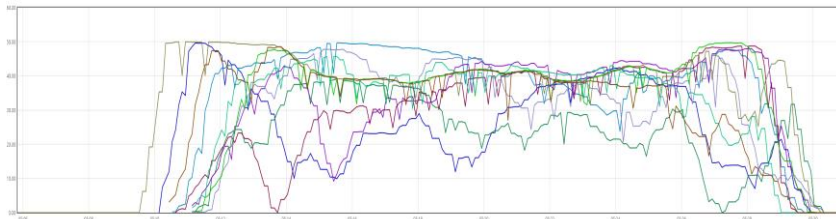
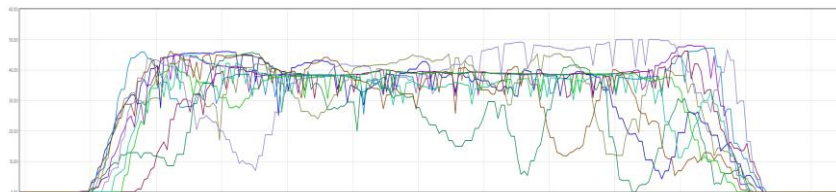


容器架構與規模可擴充性：實驗二

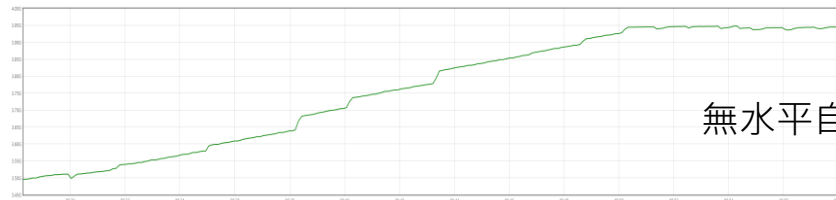
還是要先預估可能的
容量需求

- 由於HPA是定期監控服務的負載量，而非即時監控，導致在遇到流量時仍然會受到監控週期的限制而降低整體效能，當遇到越高且越即時的流量時，定期監控機制的缺點將會更為放大

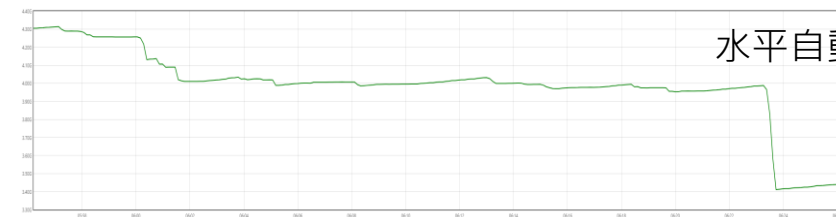
CPU 使用率



快取記憶體使用狀況



無水平自動擴展



水平自動擴展

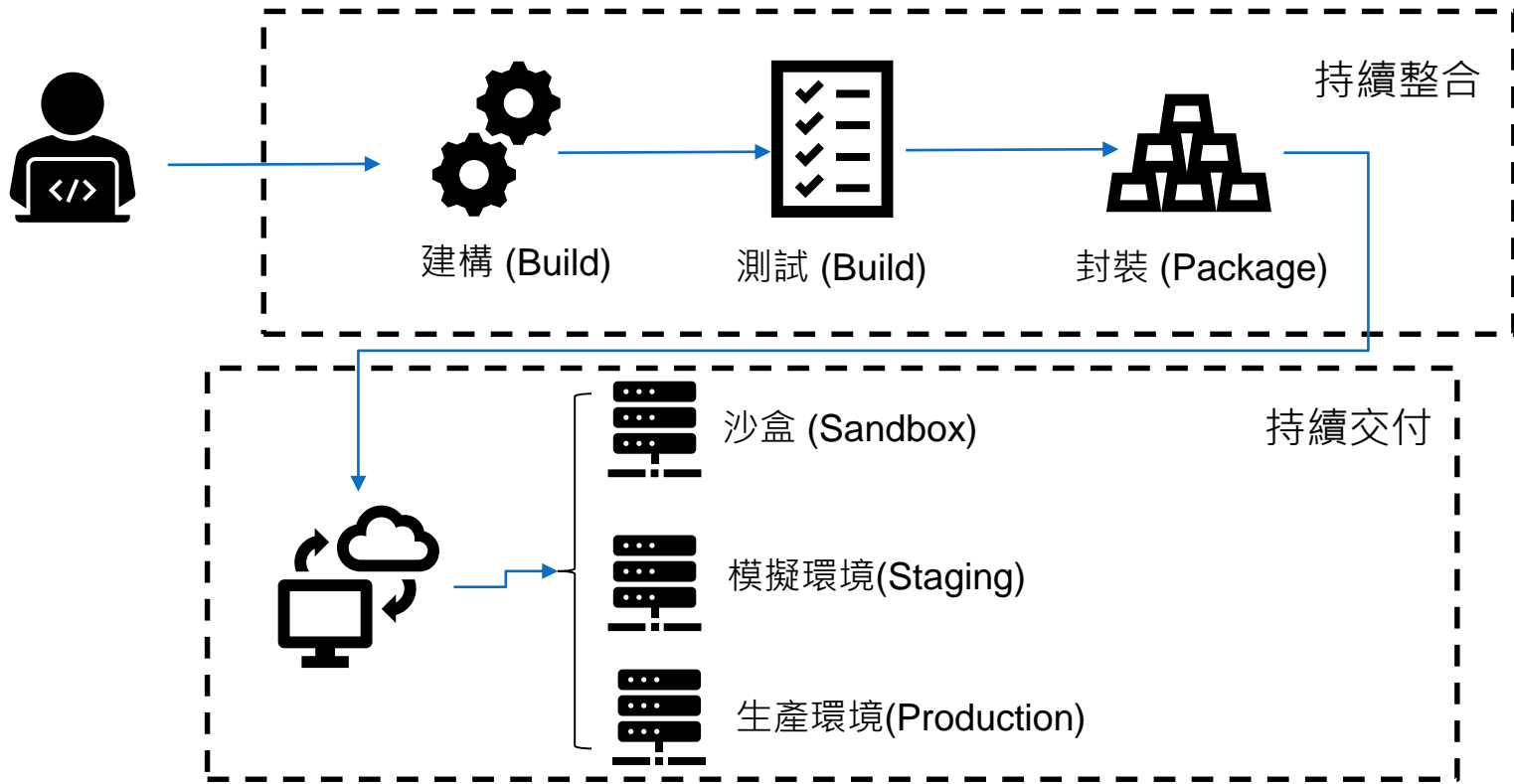
容器與部署速度

- 在要部署應用程式時，容器架構比虛擬機架構要來的輕量
- 在搭配 CD 工具時，安裝系統之速度會比直接在系統中安裝程式要快
- 與 CI/CD 工具整合，可以提升系統開發效率

開發與設計階段中容器架構的使用

系統上線後容器架構的使用

容器與部署速度 — 對於 CI/CD 的支援



容器與容錯能力

- 在 Kubernetes 環境中，可以直接建立服務的複製 (Replication)，並且在系統程序崩潰時，可以另外自動部署程序。
- 但是不見得所有程式都單純在骨幹層級進行複製就好

呈現當單一容器當機後
會部署新的容器的情境

呈現當整個節點當機後
會將節點部署於新容器
的情境

呈現骨幹層級的容錯不
見得適用於應用程式



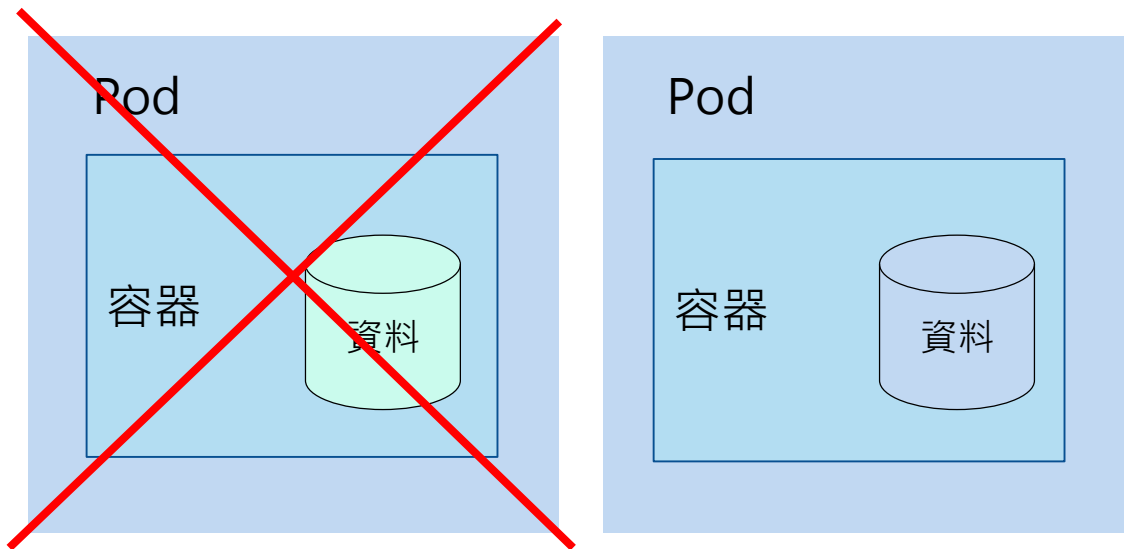
最主要還是看一致性的維護成本

容器與容錯能力—問題概述

StatefulSet 只是確保更新或還原的順序

Pod 的重啟政策
(restartPolicy)

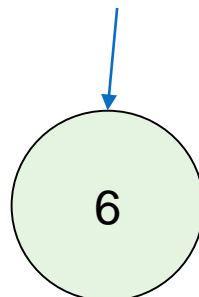
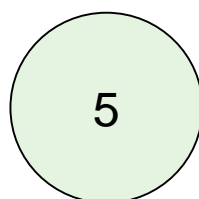
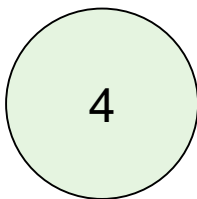
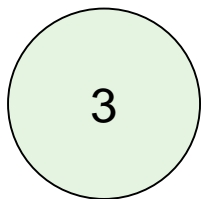
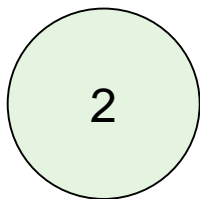
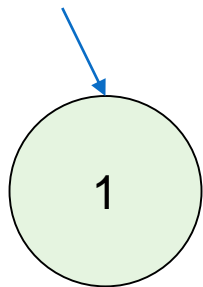
- Always (預設)
- OnFailure
- Never



一個 Pod 內的容器重啟時，會是直接按照映像檔與設定檔建立一個新的容器

如果不是 StatefulSet 的可能問題

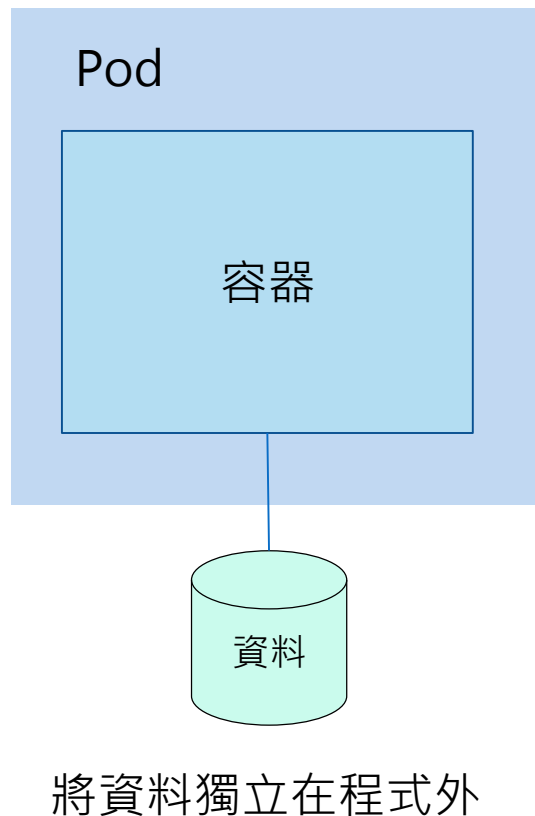
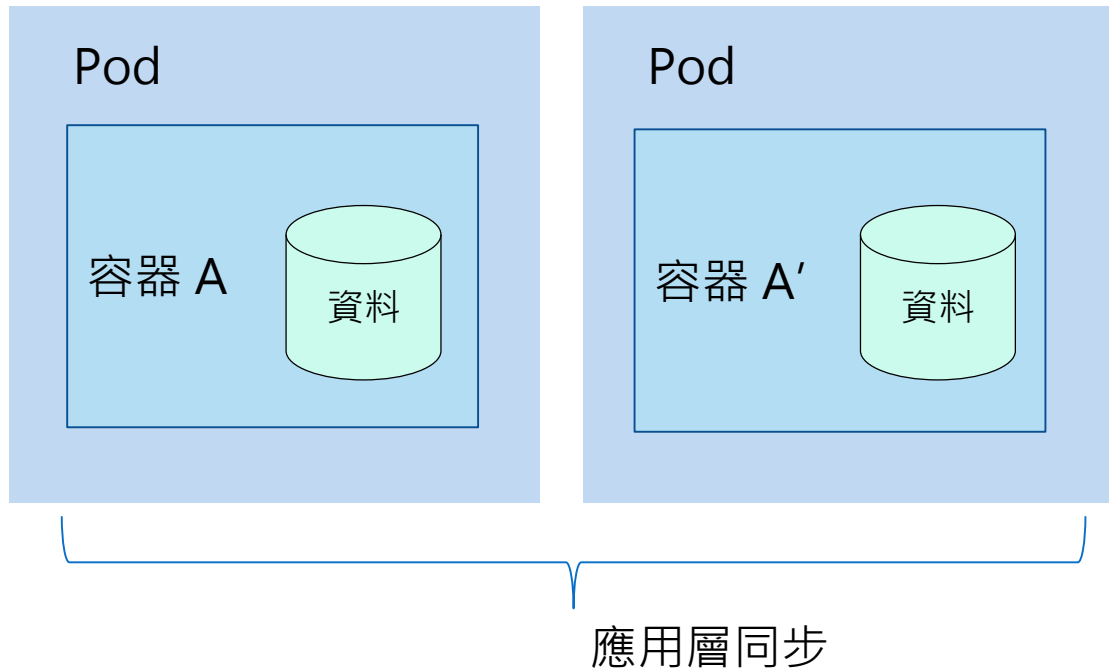
版本 x 的更新從 1 到 6



版本 y 的更新從 6 到 1

如果同時進行這兩個更新，最終節點 3 會是版本 x 或版本 y？

容器與容錯能力－同步策略



容器與可維護性

- 容器架構有提供工具，方便管理系統部署
- 監控容器時，可以不需要額外安裝監控代理人程式
- 有相關的工具，可以方便管理與維護

Deploy 時可以設定
revisionHistoryLimit
紀錄保存版本

可以使用 rollout
history 指令去檢視部
署歷史

可以使用 rollout undo
去還原先前版本

Kubernetes 的發布管理

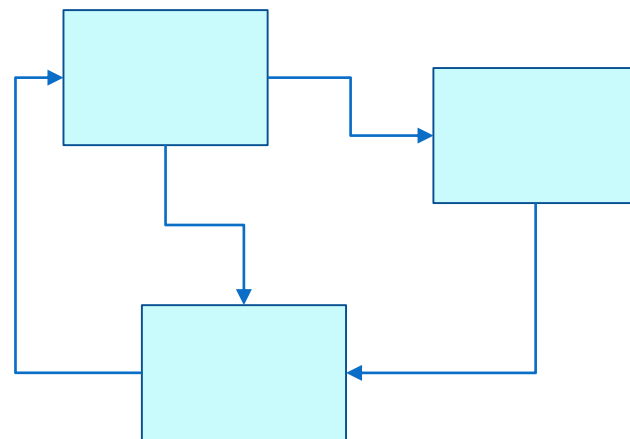
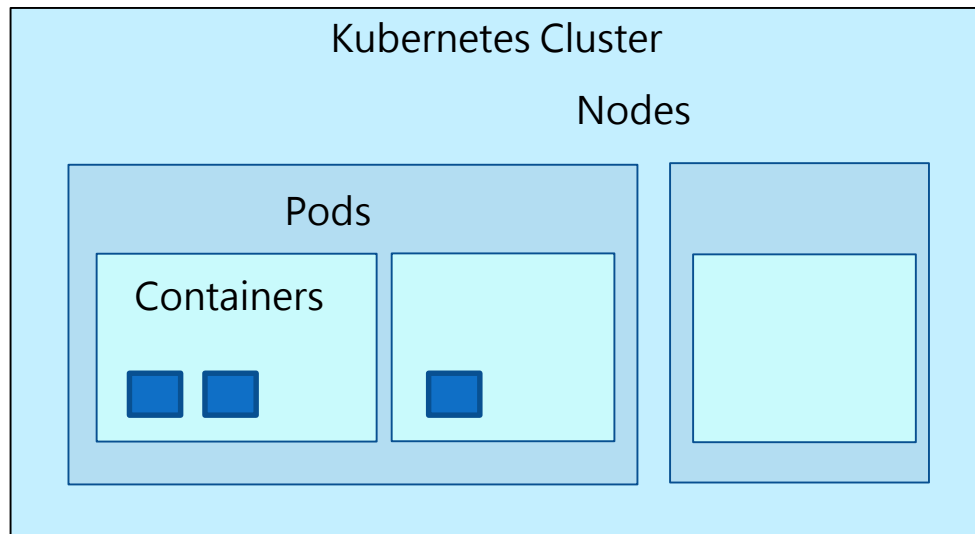
```
prlab@prlab:~/josh/deployment_test$ kubectl rollout history deployment test-deploy
deployment.apps/test-deploy
REVISION  CHANGE-CAUSE
4         kubectl set image deployment test-deploy nginx=nginx:1.16.1 --record=true
5         kubectl edit deployment test-deploy --record=true
6         kubectl edit deployment test-deploy --record=true

prlab@prlab:~/josh/deployment_test$ kubectl rollout undo deployment/test-deploy --to-revision=4
deployment.apps/test-deploy rolled back
prlab@prlab:~/josh/deployment_test$ kubectl rollout history deployment test-deploy
deployment.apps/test-deploy
REVISION  CHANGE-CAUSE
5         kubectl edit deployment test-deploy --record=true
6         kubectl edit deployment test-deploy --record=true
7         kubectl set image deployment test-deploy nginx=nginx:1.16.1 --record=true

prlab@prlab:~/josh/deployment_test$
```

監控是透過遙測的數據，掌握系統或骨幹的運作狀態

觀察是針對應用，直接取得其函式或功能的執行數據





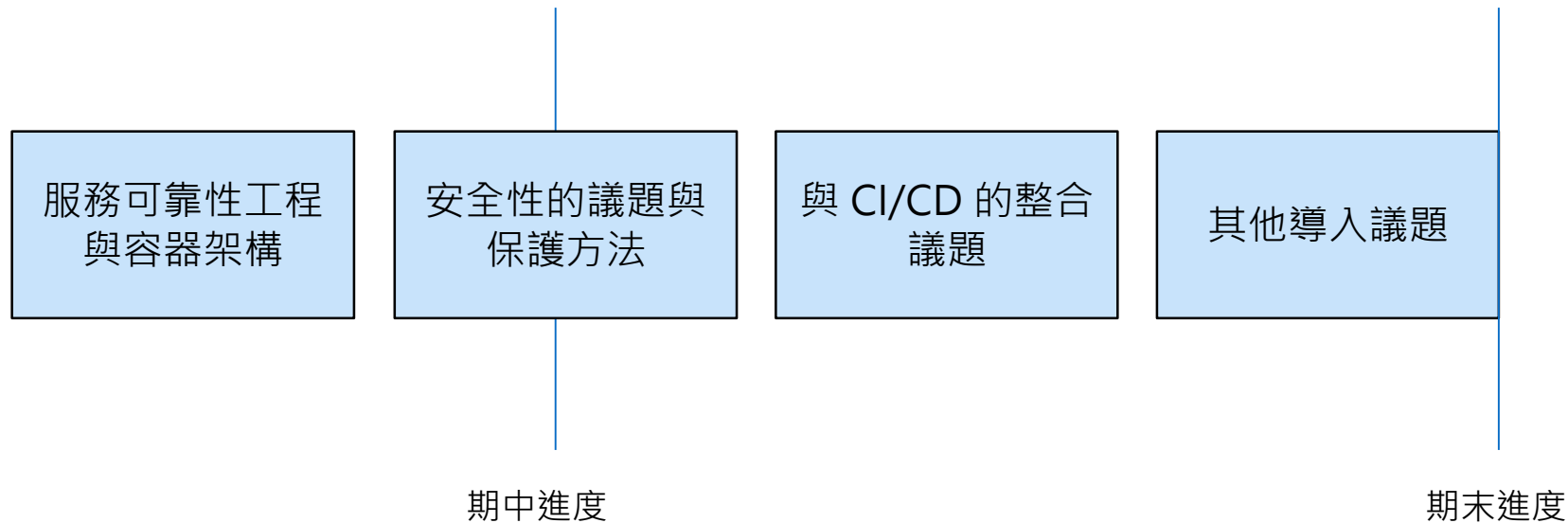
計畫背景與目標

容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議



服務可靠性工程
與容器架構

安全性的議題與
保護方法

與 CI/CD 的整合
議題

其他導入議題

系統管理 (System Administration)

- 提供硬體
- 管理軟體
- 管理網路

DevOps

- 持續整合
- 持續交付
- 對程式有更多的了解

網站可靠性

- 聚焦於效能
- 客戶導向

效能

程式

骨幹

服務可靠性工程

- **Wikipedia:** 網站可靠性工程 (Site Reliable Engineering, SRE) 是引用軟體工程技術來解決骨幹與作業問題的原則與實務。主要的目的是要建立規模可擴充性與高穩定度的軟體系統。
- 網站可靠性工程師 (Site Reliable Engineer)，則是在企業內運用上述實務，來協助企業建立與維護規模可擴充性與高穩定度軟體系統的工作角色。

用軟體工程的方法來
解決維運問題

透過服務水準目標來
進行管理

儘量使用機器來最小
化人力工作

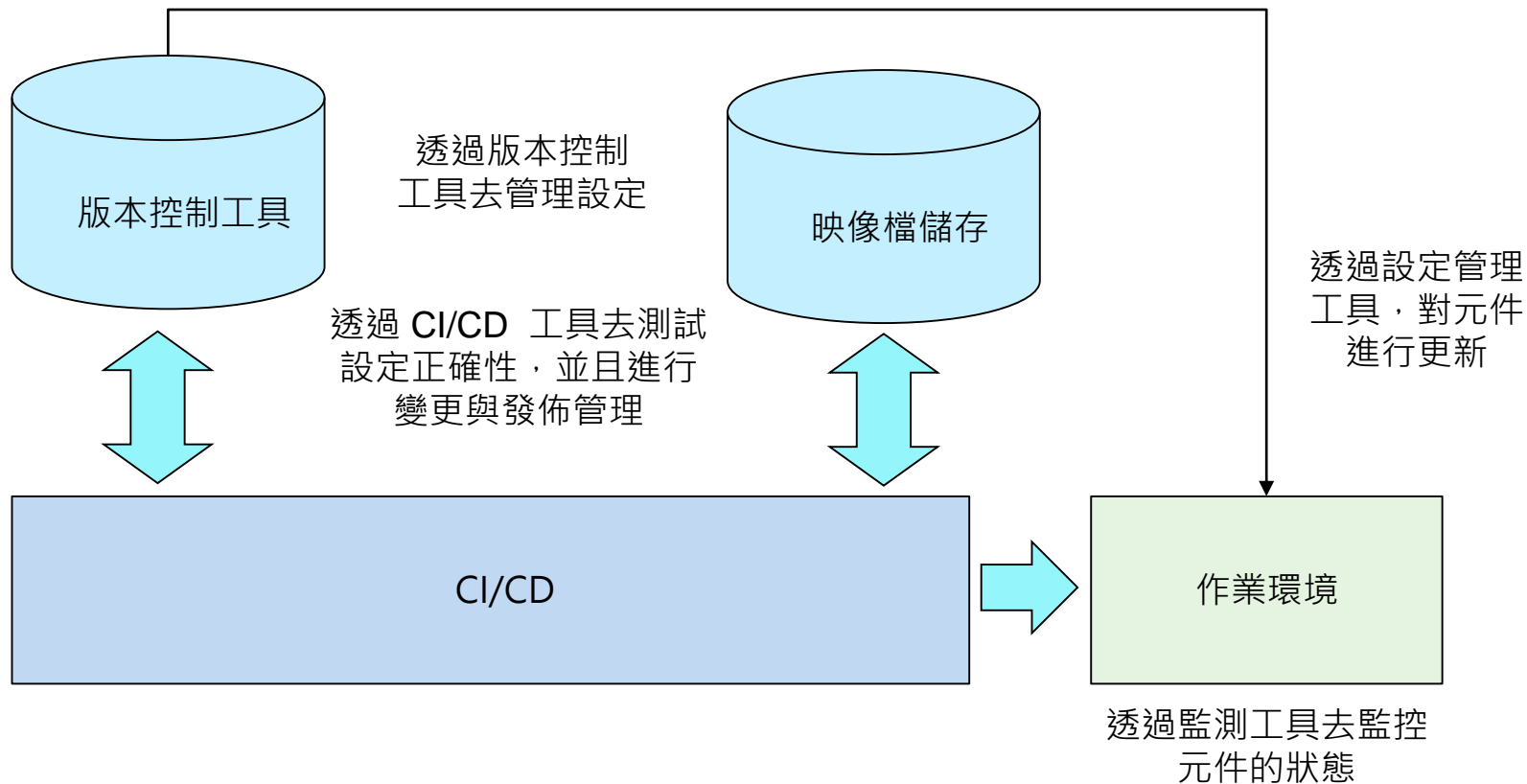
確定要自動化什麼？
甚麼條件可以自動化？
如何進行自動化？

降低故障以加快步調

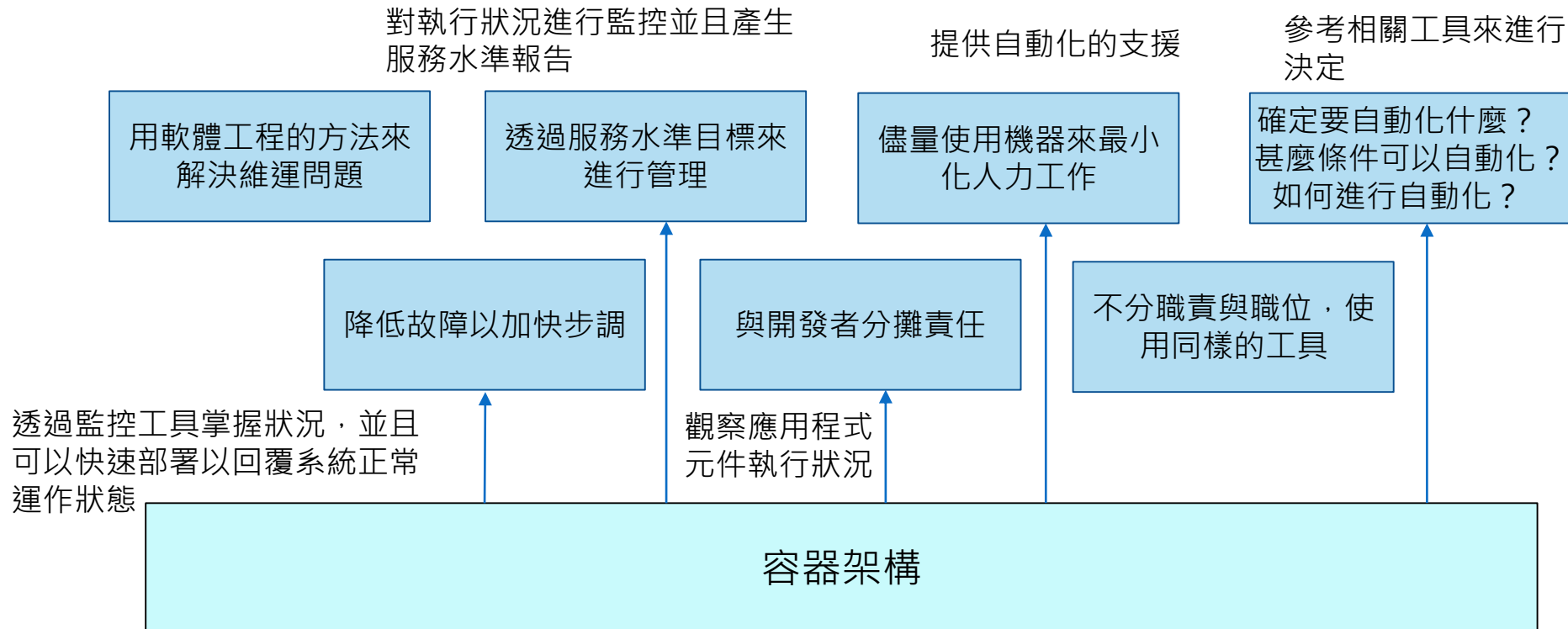
與開發者分攤責任

不分職責與職位，使
用同樣的工具

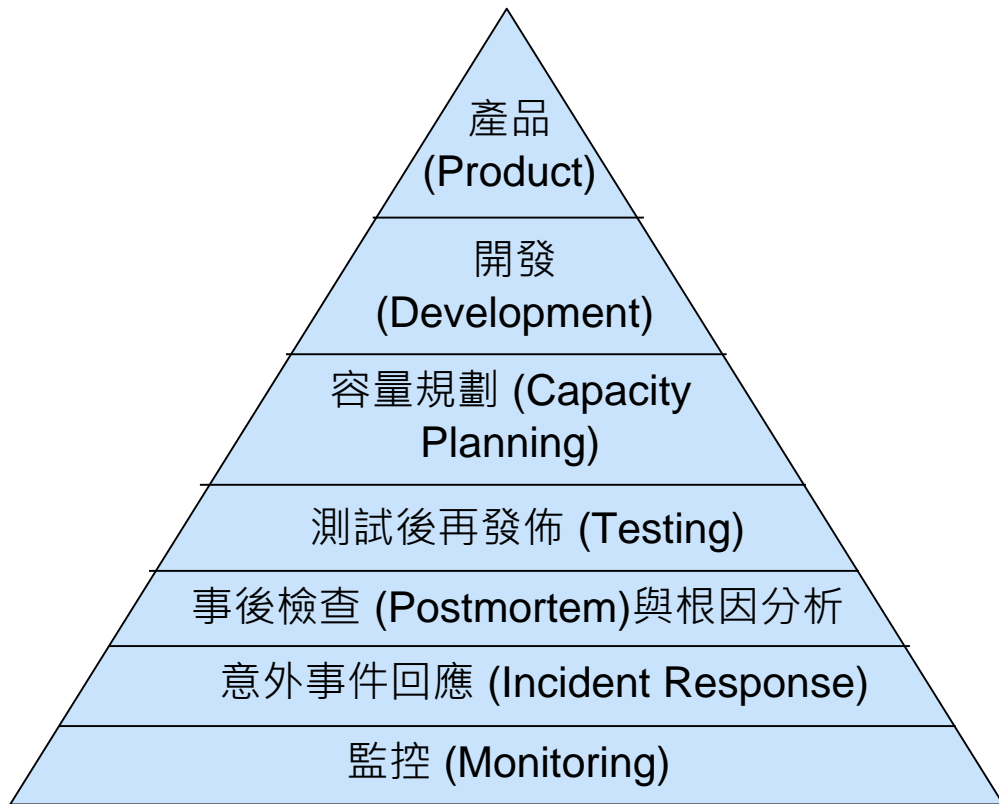
從程式開發來學習骨幹管理 (Infrastructure as Code)



服務可靠性工程與容器架構關係緊密



從服務可靠性階層來看導入策略



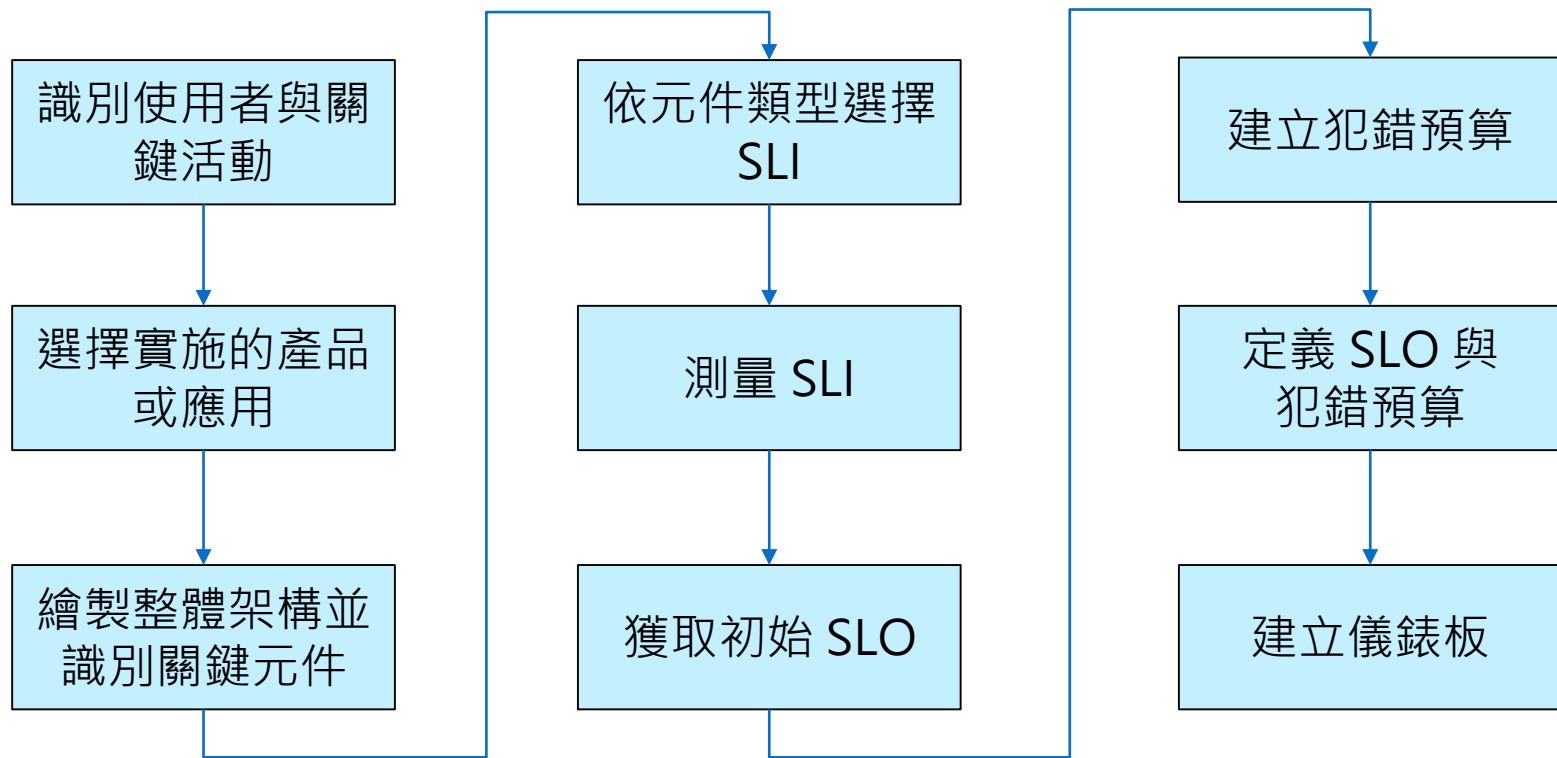
訂定服務水準指標、
目標，與協定

連接到產品

逐步賦予意義

先從骨幹監控開始

建立 SLO 的步驟



SRE 的四大黃金訊號

延遲 (Latency)

處理要求所需的時間

流量 (Traffic)

單位時間內可處理的要求

錯誤 (Error)

應用程式的錯誤率

飽和度
(Saturation)

記憶體或運算資源被使用的程度

服務可靠性工程與容器架構的研究整理

- 雖然要導入 SRE 不見得要採用容器架構，但目前的容器架構有搭配許多可觀察性與追蹤工具，這些工具可以用來推動 SRE。
- 可以從服務可靠性階層的角度逐步推動 SRE
 - 可以從監控開始，之後將可收集到的數據對應到產品的關鍵元件
 - 健全監控骨幹後，之後可以從由上而下的角度，去選擇 SLI，並且制定 SLO 與 SLA。

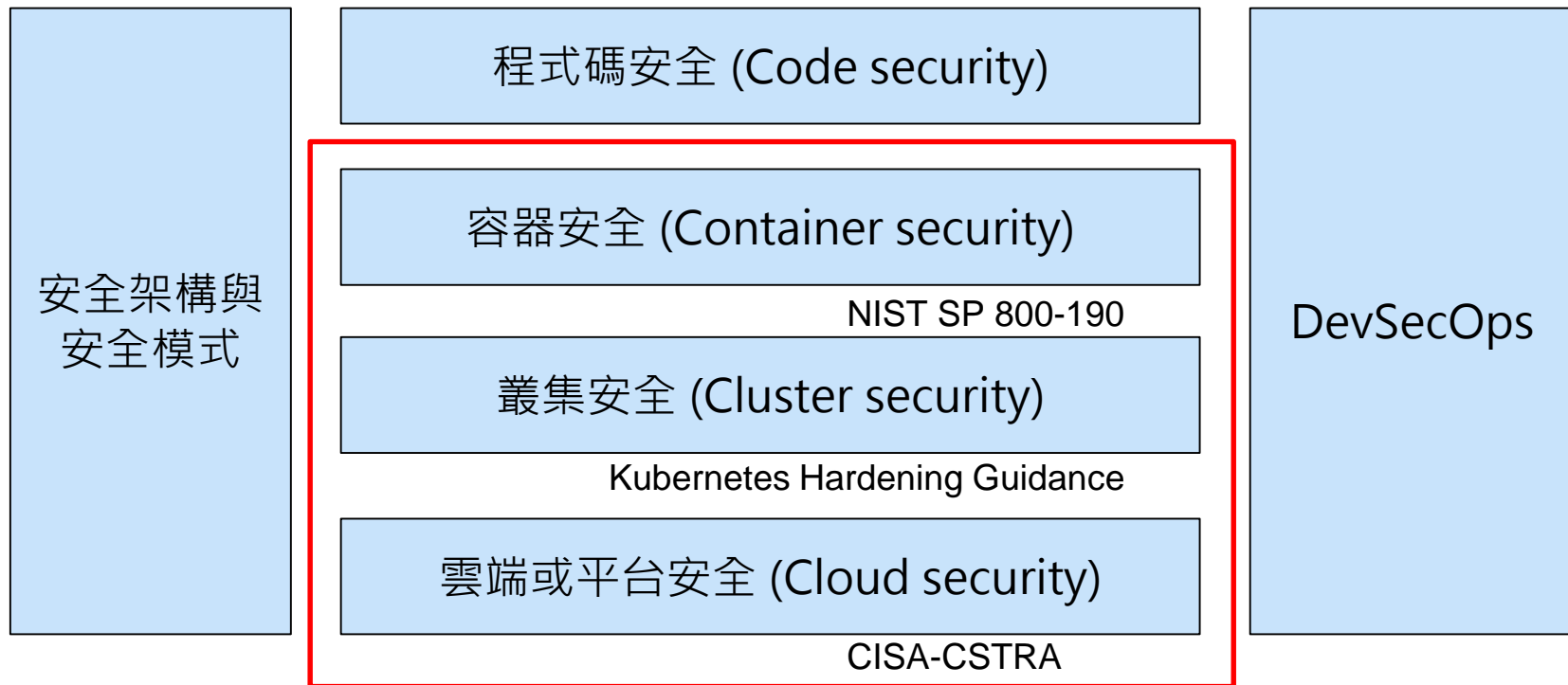
服務可靠性工程
與容器架構

安全性的議題與
保護方法

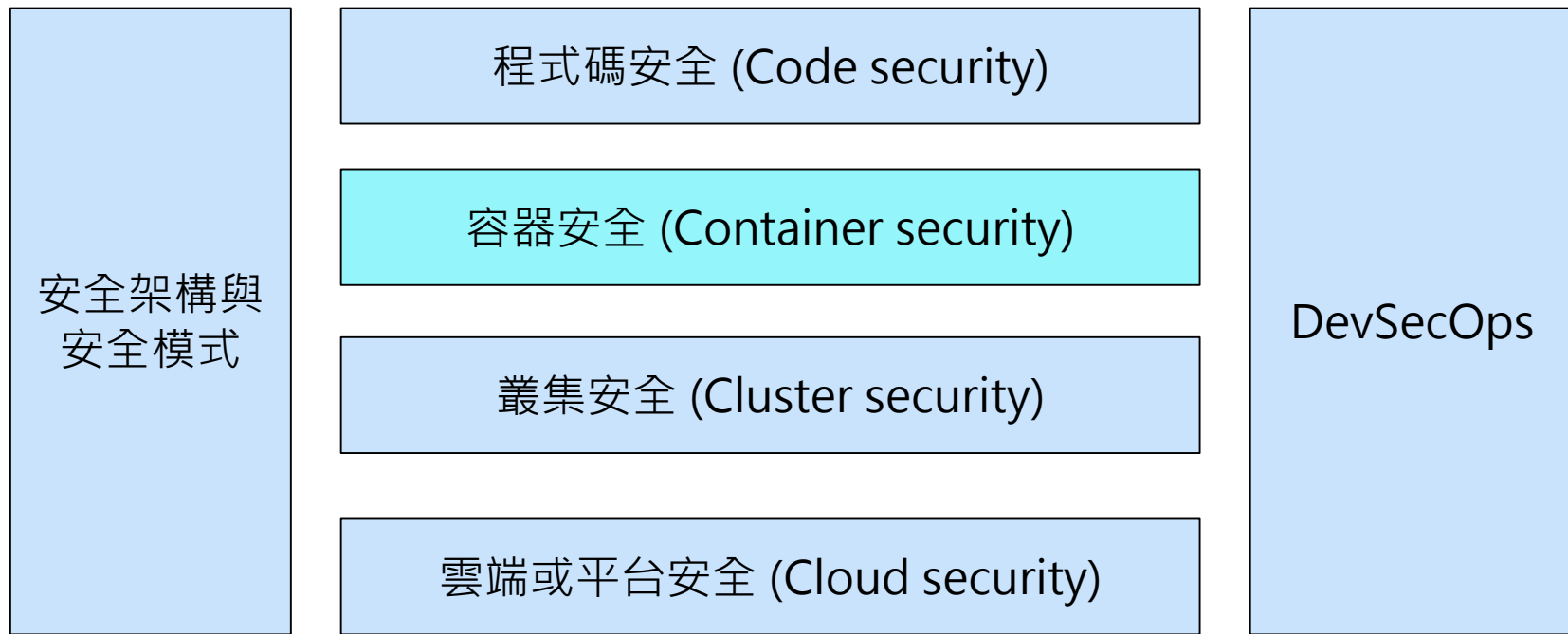
與 CI/CD 的整合
議題

其他導入議題

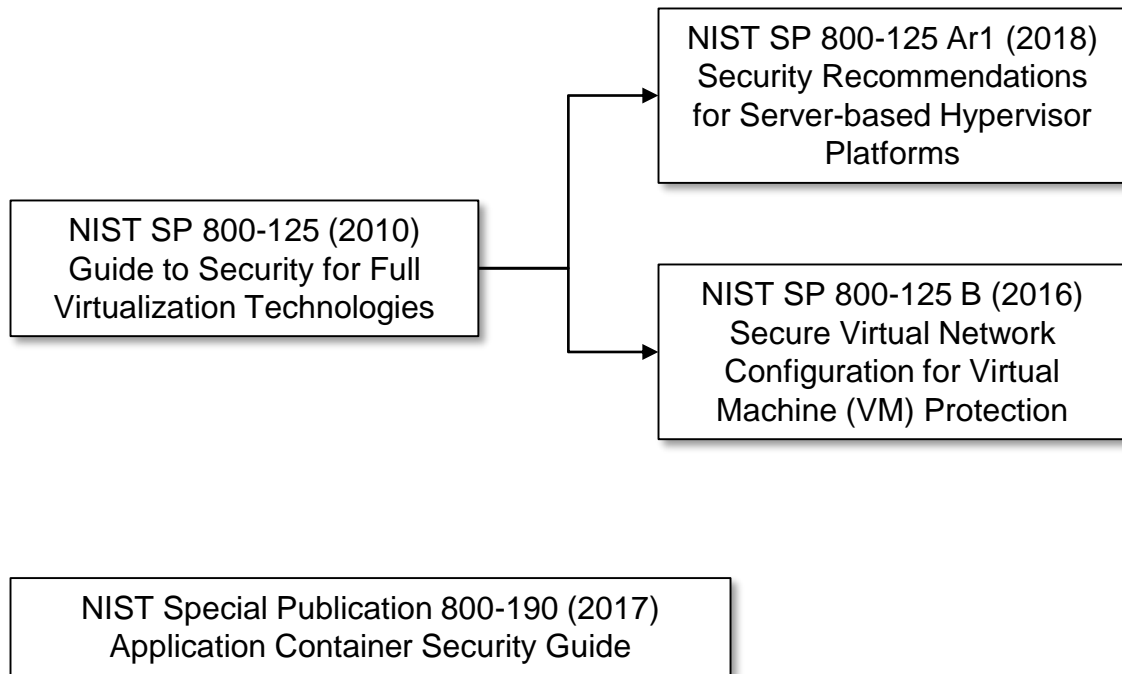
容器安全性的議題與保護方法



容器安全性的議題與保護方法

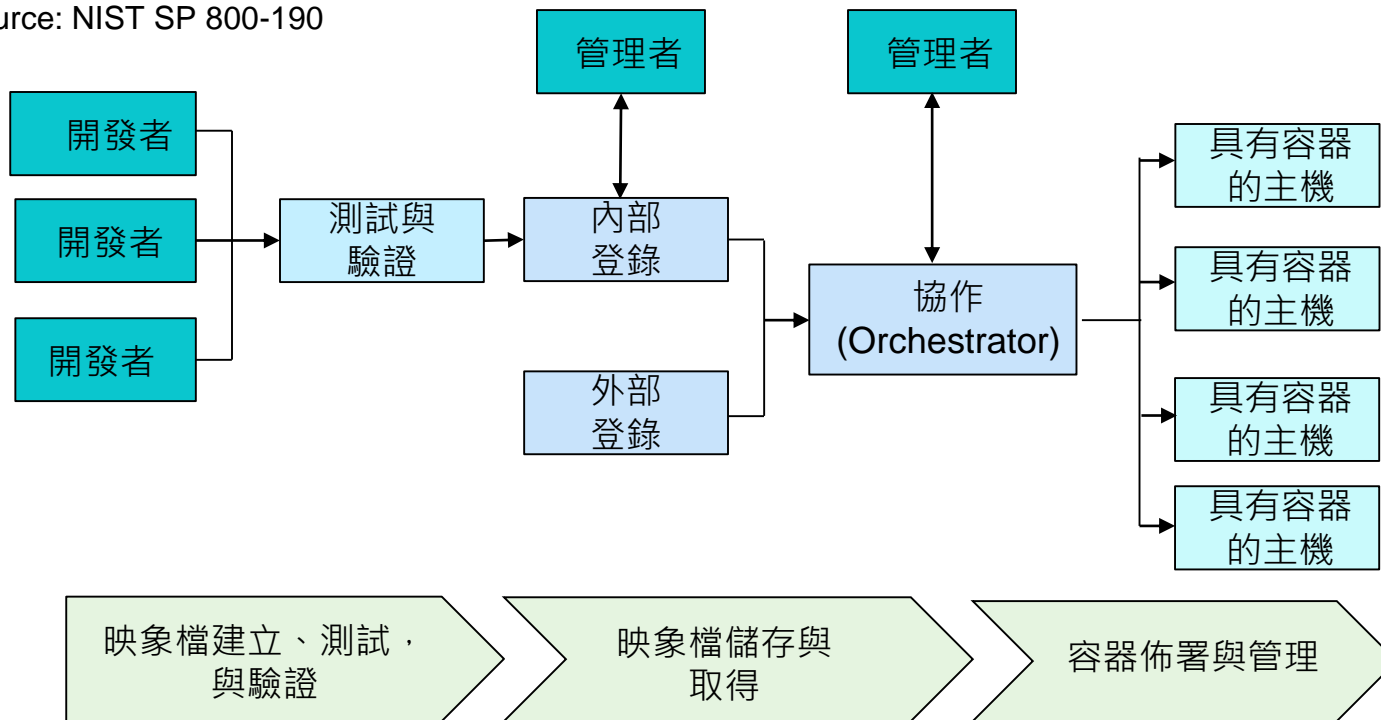


美國 NIST 對於虛擬化技術的安全建議

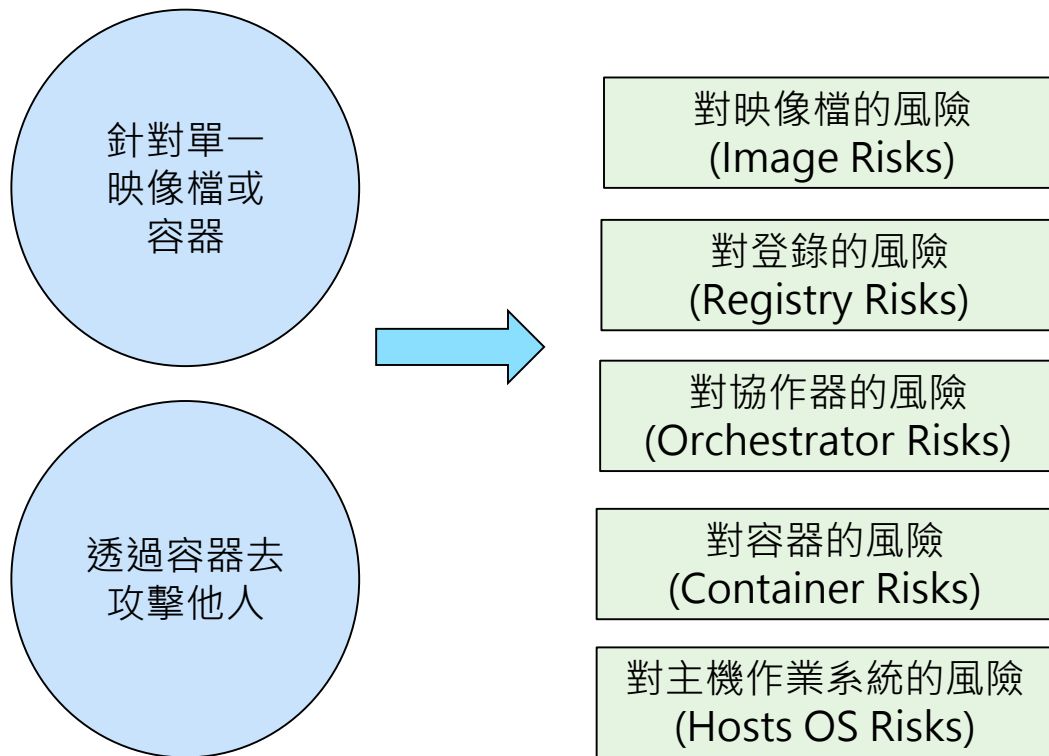


容器技術的架構層次、元件，與生命週期階段

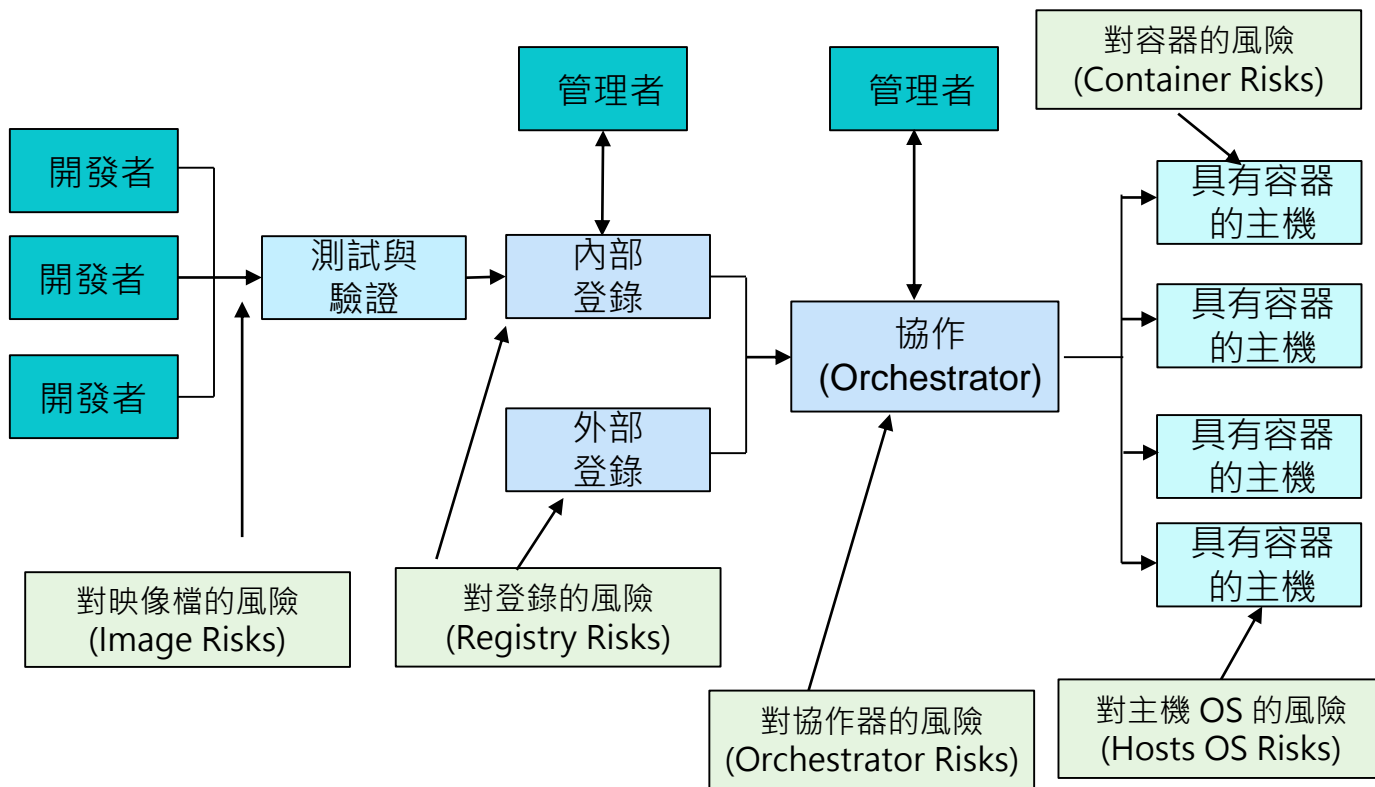
Source: NIST SP 800-190



主要風險種類



風險與生命週期



對映像檔的風險與對策

- 映像檔因為未更新而沒修補弱點 (Image Vulnerabilities) :
 - 考量映像檔中各元件的更新狀態
- 映像檔設定缺陷 (Image configuration defects)
 - 檢驗映像檔設定的安全性
- 內有惡意軟體 (Embedded Malware)
 - 對映像檔進行惡意軟體掃描
- 內有明文秘密 (Embedded clear text secrets)
 - 應該將機敏資訊存於映像檔之外
- 使用不可信的映像檔 (Use of Untrusted Image)
 - 建立信賴的映像檔或外部登錄清單

對登錄的風險與對策

- 對登錄的不安全連線 (Insecure connections to registries)
 - 確保映像檔的上傳或下載連線有加密
- 登錄當中的過期映像檔 (Stale images in registries)
 - 移除掉登錄當中不安全的映像檔，或是在映像檔名稱當中，加入可以識別版本或最新版本的資訊
- 不充分的身分鑑別與授權限制 (Insufficient authentication and authorization restrictions)
 - 對於存取任何包含敏感或專屬資訊的映像檔的下載應該要經過身分鑑別，並且確保有獲得授權
 - 限制可以將映像檔上傳至登錄的使用者之身分

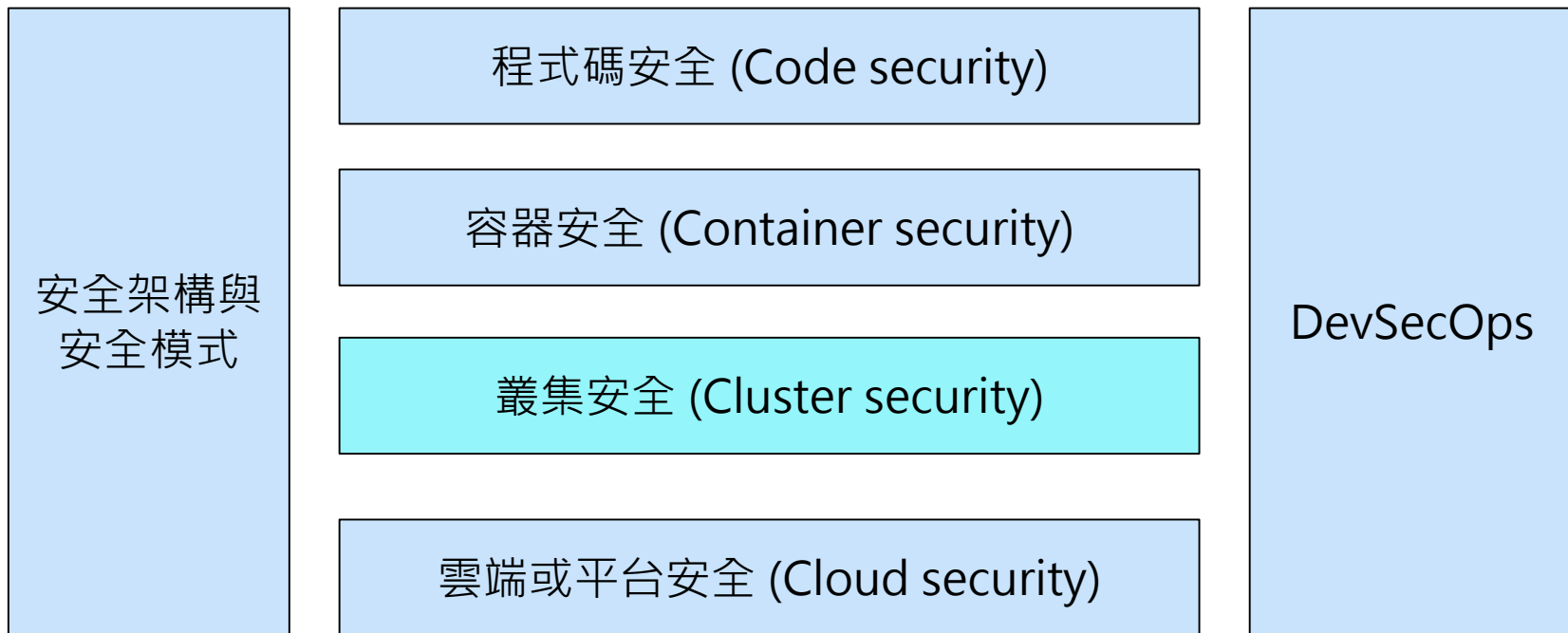
對協作器的風險與對策

- 無限制的管理存取 (Unbounded administrative access)
 - 依照最小權限原則給予每個映像檔的執行權限
- 未經授權的存取 (Unauthorized access)
 - 提升使用協作器的身分鑑別機制之等級
- 容器間網路隔離不良 (Poorly separated inter-container network traffic)
 - 按照敏感度等級去分割網路
- 將不同敏感度等級的工作放在一起跑 (Mixing of workload sensitivity levels)
 - 按照敏感度等級去佈署容器
- 未妥善管理節點間的信賴機制 (Orchestrator node trust)
 - 確保叢集內各節點的安全性

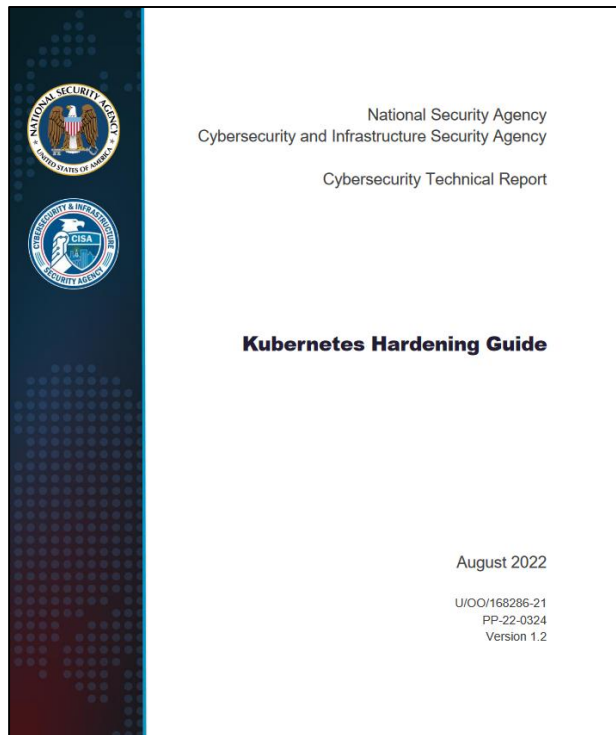
對容器的風險與對策

- 執行中軟體的弱點 (Vulnerabilities within the runtime software)
 - 偵測執行中容器的弱點
- 無限制的容器網路存取 (Unbounded network access from containers)
 - 控制容器送出的網路流量
- 不安全的容器執行環境設定 (Insecure container runtime configurations)
 - 自動符合容器設定標準
- 具有有弱點的應用程式 (App vulnerabilities)
 - 偵測容器中應用程式的異常行為
- 異常的容器 (Rogue containers)
 - 區隔開發、測試、線上環境
 - 紀錄容器的操作行為並留下稽核軌跡

容器安全性的議題與保護方法



美國 NSA 的 Kubernetes 強化指引



Pod 安全

網路分隔與安全強化

身分鑑別與授權

稽核紀錄與威脅偵測

Pod 安全

- 包括容器及容器引擎，皆不使用root權限
- 檔案系統設定為唯讀
- 建置安全的容器映像檔
- 套用Pod安全政策
- 保護Pod 服務帳號識別符 (Token)
- 強化容器執行環境之安全

網路分隔與安全強化

- 命名空間隔離
- 依需求訂定網路安全政策
- 透過資源政策限制資源的使用
- 控制節點的安全強化
 - 包含 Etcd 與 kubeconfig 設定
- 工作節點的隔離
- 使用加密連線
- 使用 **Secret** 變數保護設定
- 保護相關設施

網路存取政策例 (只有有 access 標籤的才可存取 nginx)

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: example-access-nginx
  namespace: prod
spec:
  podSelector:
    matchLabels:
      app: nginx
  ingress:
    -from:
      -podSelector:
        matchLabels:
          access: "true"
```


網路存取政策例 (預設禁止外部的連線)

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-ingress
spec:
  podSelector: {}
  policyType:
    - Ingress
```

網路存取政策例 (預設禁止連線至外部)

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-egress
spec:
  podSelector: {}
  policyType:
    - Egress
```

資源限制政策例

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: example-cpu-mem-resourcequota
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

身分鑑別與授權

- 身分鑑別
- 使用 RBAC 設定存取權限

角色定義例

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: your-namespace-name
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

角色綁訂例

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: your-namespace-name
subjects:
- kind: User
  name: jane
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

稽核紀錄與威脅偵測

- 紀錄
- 威脅偵測與警報

惡意行為	偵測措施
部署惡意的Pod或容器，以相同的名字混淆。	檢視非典型Pod與容器的部署，檢查與映像檔的ID與雜湊值，監視容器應用是否以root權限執行。
將惡意映像檔送入映像庫	監控容器引擎與容器映像庫的紀錄及重新部署的行為。
透過容器應用程式的弱點取得執行任意程式的權限，並對API-server進行攻擊。	監控來自容器非常規的API請求及系統呼叫。
在入侵叢集後針對API Sever進行攻擊取得更進一步的控制。	監控傳送至API Server失敗的請求
在入侵叢集後植入挖礦軟體	監控是否有非常規的資源占用。
使用匿名帳號進行惡意活動。	監控匿名帳號在叢集間的活動。
在控制容器後嘗試掛載主機的檔案目錄，取得主機的執行權限。	必須緊密監控非正常的儲存空間掛載活動。
在Kubernetes叢集部署工作排程，執行重複性的惡意行為。	必須緊密監控非正常的工作排成部署。

使用 kube-bench 檢查是否符合 CIS Benchmark

```
root@ubuntu:/home/shichoc# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
kube-bench-5lvkl 0/1     Completed 0           3m39s
root@ubuntu:/home/shichoc# kubectl logs kube-bench-5lvkl
[INFO] 4 Worker Node Security Configuration
[INFO] 4.1 Worker Node Configuration Files
[FAIL] 4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Automated)
[FAIL] 4.1.2 Ensure that the kubelet service file ownership is set to root:root (Automated)
[PASS] 4.1.3 If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Manual)
[PASS] 4.1.4 If proxy kubeconfig file exists ensure ownership is set to root:root (Manual)
[FAIL] 4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)
[FAIL] 4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Automated)
[WARN] 4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Manual)
[WARN] 4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Manual)
[FAIL] 4.1.9 Ensure that the kubelet --config configuration file has permissions set to 644 or more restrictive (Automated)
[FAIL] 4.1.10 Ensure that the kubelet --config configuration file ownership is set to root:root (Automated)
[INFO] 4.2 Kubelet
[FAIL] 4.2.1 Ensure that the --anonymous-auth argument is set to false (Automated)
[FAIL] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[FAIL] 4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)
[WARN] 4.2.4 Ensure that the --read-only-port argument is set to 0 (Manual)
[WARN] 4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Manual)
[FAIL] 4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Automated)
[FAIL] 4.2.7 Ensure that the --make-iptables-util-chains argument is set to true (Automated)
[WARN] 4.2.8 Ensure that the --hostname-override argument is not set (Manual)
[WARN] 4.2.9 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Manual)
[WARN] 4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)
[FAIL] 4.2.11 Ensure that the --rotate-certificates argument is not set to false (Automated)
[WARN] 4.2.12 Verify that the RotateKubeletServerCertificate argument is set to true (Manual)
[WARN] 4.2.13 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)

== Remediations node ==
4.1.1 Run the below command (based on the file location on your system) on the each worker node.
For example, chmod 644 /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
4.1.2 Run the below command (based on the file location on your system) on the each worker node.
```



```
root@ubuntu: /home/shichoc

5.4.2 Refer to the Secrets management options offered by your cloud provider or a third-party secrets management solution.

5.5.1 Follow the Kubernetes documentation and setup image provenance.

5.7.1 Follow the documentation and create namespaces for objects in your deployment as you need them.

5.7.2 Use `securityContext` to enable the docker/default seccomp profile in your pod definitions.
An example is as below:
  securityContext:
    seccompProfile:
      type: RuntimeDefault

5.7.3 Follow the Kubernetes documentation and apply SecurityContexts to your Pods. For a suggested list of SecurityContexts, you may refer to the CIS Security Benchmark for Docker Containers.

5.7.4 Ensure that namespaces are created to allow for appropriate segregation of Kubernetes resources and that all new resources are created in a specific namespace.

== Summary policies ==
0 checks PASS
0 checks FAIL
30 checks WARN
0 checks INFO

== Summary total ==
2 checks PASS
12 checks FAIL
39 checks WARN
0 checks INFO
```

使用 Trivy 掃描映像檔弱點



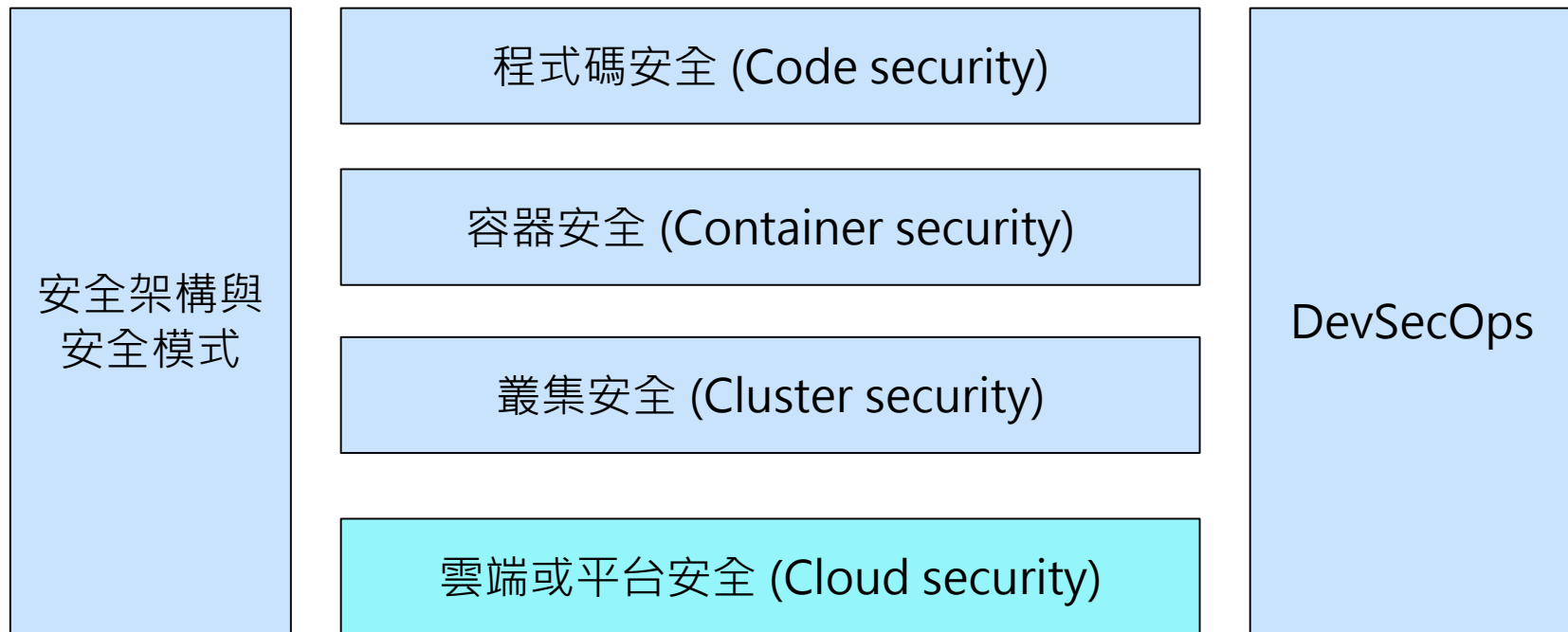
```
root@ubuntu: /home/shichoc
kindest/node <none> f5aa68ba122a 5 months ago 911MB
python 3.4-alpine c06adcf62f6e 3 years ago 72.9MB
root@ubuntu: /home/shichoc# trivy image python:3.4-alpine
2022-10-26T22:42:32.400-0700 INFO Vulnerability scanning is enabled
2022-10-26T22:42:32.400-0700 INFO Secret scanning is enabled
2022-10-26T22:42:32.400-0700 INFO If your scanning is slow, please try '--security-checks vuln' to disable secret scanning
2022-10-26T22:42:32.400-0700 INFO Please see also https://aquasecurity.github.io/trivy/v0.33/docs/secret/scanning/#recommendation for faster secret detection
2022-10-26T22:42:35.828-0700 INFO Detected OS: alpine
2022-10-26T22:42:35.828-0700 INFO Detecting Alpine vulnerabilities...
2022-10-26T22:42:35.829-0700 INFO Number of language-specific files: 1
2022-10-26T22:42:35.829-0700 INFO Detecting python-pkg vulnerabilities...
2022-10-26T22:42:35.834-0700 WARN This OS version is no longer supported by the distribution: alpine 3.9.2
2022-10-26T22:42:35.834-0700 WARN The vulnerability detection may be insufficient because security updates are not provided

python:3.4-alpine (alpine 3.9.2)

Total: 37 (UNKNOWN: 0, LOW: 4, MEDIUM: 16, HIGH: 13, CRITICAL: 4)
```

Library	Vulnerability	Severity	Installed Version	Fixed Version	Title
expat	CVE-2018-20843	HIGH	2.2.6-r0	2.2.7-r0	expat: large number of colons in input makes parser consume high amount... https://avd.aquasec.com/nvd/cve-2018-20843
	CVE-2019-15903			2.2.7-r1	expat: heap-based buffer over-read via crafted XML input https://avd.aquasec.com/nvd/cve-2019-15903
libbz2	CVE-2019-12900	CRITICAL	1.0.6-r6	1.0.6-r7	bzip2: out-of-bounds write in function BZ2_decompress https://avd.aquasec.com/nvd/cve-2019-12900
libcrypto1.1	CVE-2019-1543	HIGH	1.1.1a-r1	1.1.1b-r1	openssl: ChaCha20-Poly1305 with long nonces https://avd.aquasec.com/nvd/cve-2019-1543
	CVE-2020-1967			1.1.1g-r0	openssl: Segmentation fault in SSL_check_chain causes denial of service https://avd.aquasec.com/nvd/cve-2020-1967
	CVE-2021-23840			1.1.1j-r0	openssl: integer overflow in CipherUpdate https://avd.aquasec.com/nvd/cve-2021-23840

容器安全性的議題與保護方法



雲端安全狀態管理 (Cloud Security Posture Management)

治理與符合性 (Governance and Compliance)

骨幹與應用程式保護
(Infrastructure and Application Protections)

政策與標準 (Policies and Standards)

系統健康狀態與資源監控
(System Health and Resource Monitoring)

權限與身分存取管理 (Privilege and Identity Access Management)

意外事件回應與回復 (Incident Response and Recovery)

資料保護 (Data Protections)



Cloud Security Technical Reference Architecture

Coauthored by:

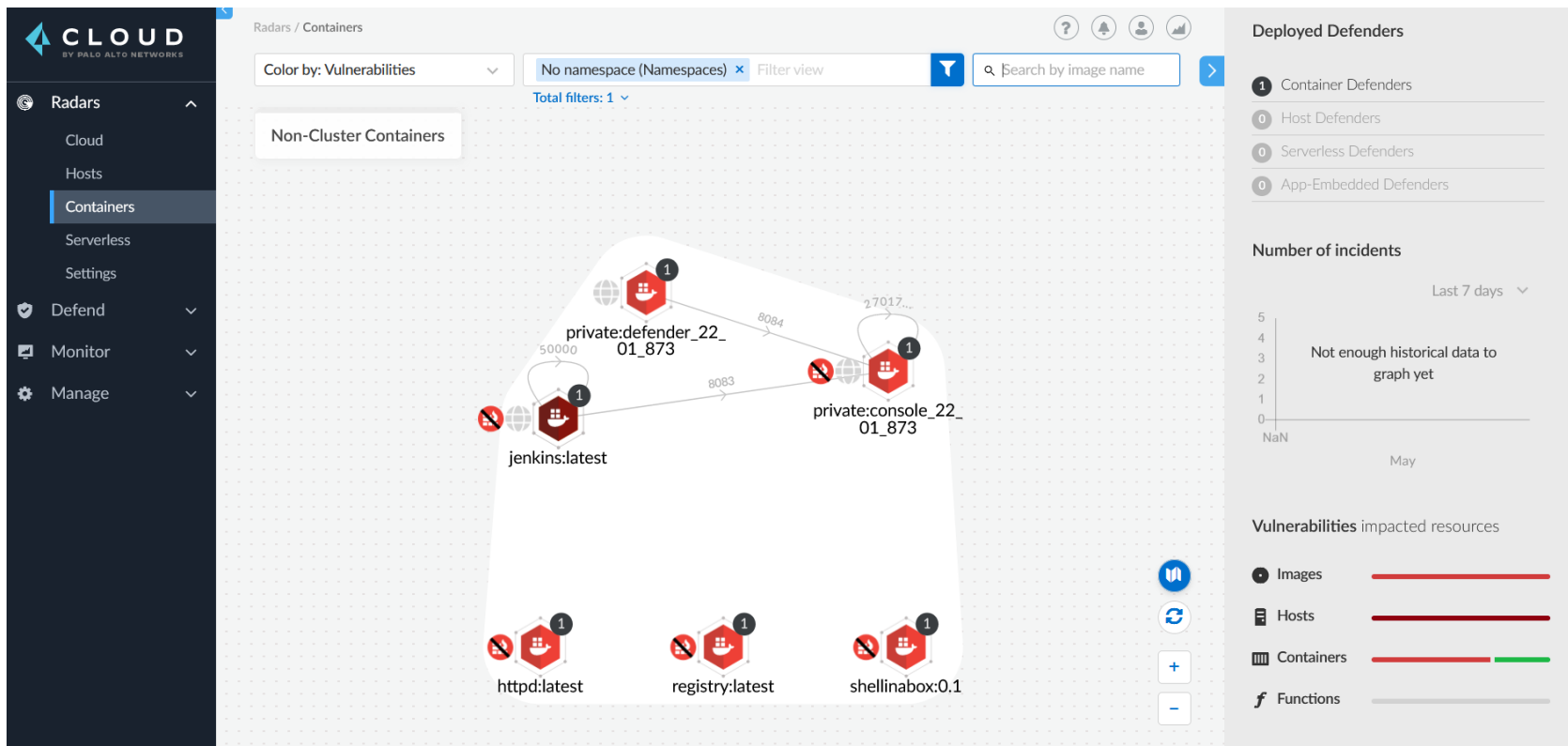
Cybersecurity and Infrastructure Security Agency,
United States Digital Service, and
Federal Risk and Authorization Management Program

June 2022

Version 2.0

	安全與風險管理	持續監控與警報	身分與存取管理	與 DevSecOps 的整合	AI 與 ML 為基礎的安全能力
治理與符合性	○	○	○		
政策與標準	○	○	○	○	
權限與身分存取管理	○	○	○	○	○
資料保護	○		○		○
骨幹與應用程式保護	○	○	○		○
系統健康狀態與資源監控	○	○		○	
意外事件回應與回復	○	○	○	○	○

目前也有針對容器架構的雲端安全監控工具



安全性的議題與保護方法的後續工作

- 目前已按照 NIST SP 800-190 整理出容器架構的主要安全風險與因應策略
- 從美國網路與骨幹安全局當中的雲端安全參考框架，可以釐清採用容器架構等雲端技術時，可以採用的資訊安全技術。
- 後續將持續整理相關資料，並且更進一步研究相關工具

服務可靠性工程
與容器架構

安全性的議題與
保護方法

與 CI/CD 的整合
議題

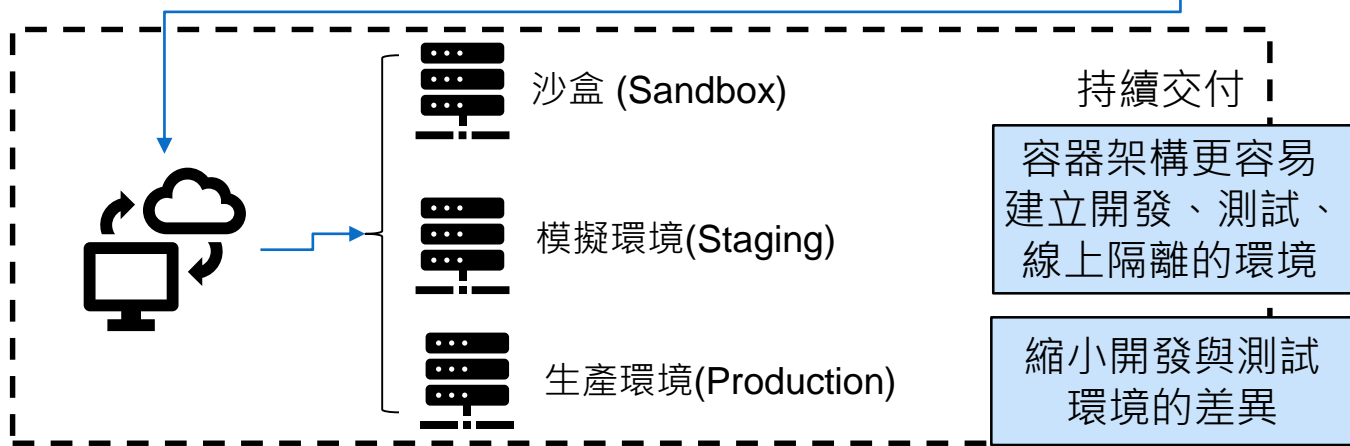
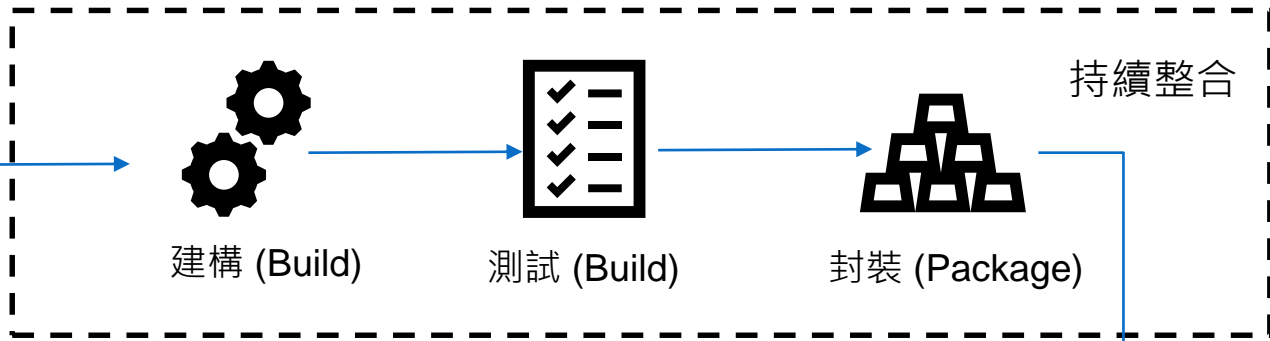
其他導入議題

容器架構可以提升 CI/CD 的效率



容器架構可以快速部署到多個平台進行測試，加快回饋週期

可加速發布與退回的速度



容器架構與 CI/CD 整合時的安全考量

- CI管道
 - 原始碼與設定的審核、發布流程所需的權限應滿足最小權限原則。
 - 確保變更皆保有紀錄，能夠追溯到變更的作者與審查對象。
 - 建構 Image 的方式是否安全符合安全原則
 - 避免使用到不該給的資源
 - 未妥善設定權限
- CD管道
 - 拉取映像檔時的來源應考慮安全，並且確保權限設置滿足最小權限原則。
 - 應針對Image Registry做好安全措施與隔離。
 - 應紀錄基礎架構的變更歷史，以便發生意外時是否能夠執行回滾措施。

服務可靠性工程
與容器架構

安全性的議題與
保護方法

與 CI/CD 的整合
議題

其他導入議題

發布策略

- 滾動式發布 (Rolling Deployment)：為預設的部署策略，逐步地去對 Pod 去進行部署或更新。而可以設定以下參數：
 - MaxSurge：最多一次部署的 Pod 數。
 - MaxUnavailable：在部署過程中，最多有多少個 Pod 不可被獲得。
- 重建 (Recreate)：將所有的 Pod 刪除後重建。此時會因此造成暫時性的服務中斷。
- 漸進式發布 (Ramped Deployment)：為滾動式發布的特例，透過限制 MaxSurge 為一很小的數值，而緩步的去進行發布或更新。
- 最大努力發布 (Best-Effort Controlled Rollout)：為滾動式發布的特例，將 MaxSurge 設為 0，而以最大的可能性去進行發布或更新。
- 金絲雀發布 (Canary Deployment)：將不同的版本分成兩個發布，而可以讓少部分人去使用新的測試版本。

使用金絲雀發布通常可有最大的穩定性與彈性，但是最好要有平台或工具支援



計畫背景與目標

容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議

主要針對容器架構的核心—協作工具

- 從以下幾個構面，來觀察商用工具如何可以彌補原生 K8S 平台的不足

資源要求與
相容性

功能性

可維護性

安全性

資源要求與相容性

- 最小資源需求
- 支援作業系統
- 支援的容器執行環境
- 對於原生K8S物件的支援

功能性

- 提供API可供操作
- 部署元件相依性考量
- 自動部署的最佳化
- 自動調整資源配置
- 內建私有映像檔儲存機制
- 內建容器與資源使用的監控工具
- 不同雲平台上的叢集管理
- 使用者資源使用的計價功能

可維護性

- 圖形化介面管理與升級
- 不停機升級
- 備份與還原
- 圖形化的組態設定管理介面
- 跨叢集資料自動同步

安全性

- 支援圖形化的使用者與權限管理
- 帳號管理與單一登入機制整合
- 內建容器映像檔的安全性
- 安全的容器映像檔儲存市集
- 安全設定檢查功能

資源要求與相容性

項目	K8S	Openshift	Tanzu	Rancher
最小資源需求	Single node node : 2 GB RAM 2 CPU	以One Cluster為例 Master node : 12 vCPU / 48 GB RAM Worker node : 4 vCPU / 8 GB RAM Single node : 8 vCPU /32GB RAM	以K8s為例 Master node : 至少3個，每個 4 core /16 GB RAM Worker node : 至少3個，每個 2 core /4 GB RAM	以RKE2為例： 2 CPU cores / 4 GB RAM 以K3S為例： 1 CPU core / 512MB
支援作業系統	Linux	CoreOS	Linux	Linux
支援的容器執行環境 (Container Runtime)	containerd、CRI-O、 使用 cri-dockerd 的 Docker Engine等支援 CRI(Container Runtime Interface) 的 容器執行環境	CRI-O	Containerd	Docker、Containerd， 並支援匯入已建立之K8S 叢集。
對於原生K8S物件的支援	是	是	是	是

功能性

項目	K8S	Openshift	Tanzu	Rancher
有提供API可供操作	是 可使用 Kubernetes API 進行操作	是 Kubernetes API 或 Openshift API	是 可使用 Kubernetes API進行操作	是 額外提供 Rancher 管理API
部署元件相依性考量	是 可透過 Init containers 去設定容器的相依性	是	是 可透過Init containers去設定容器的相依性	是
自動部署的最佳化	否 僅會挑選最小資源需求或標籤符合的節點進行部署	是 預設為部署最佳化	是 會平均部署在可利用的資源上	否 僅會挑選最小資源需求或標籤符合的節點進行部署
自動調整資源配置	否 需自行觀測資源使用狀況進行調整 (或透過程式)	否，運行中不會動態調整	否	否
內建私有映像檔儲存機制	否 (可安裝第三方 image registry)	是	有 Harbor(由VMware開發與商業技術支援的開源產品)	否 (可安裝第三方image registry)
內建容器與資源使用的監控工具	需另外安裝第三方套件 (Prometheus 、 Grafana 等)	是 (內建Prometheus)	有 (內含Prometheus與Grafana等並提供VMware商業技術支援)	是 (內建 Prometheus 與 Grafana工具)

功能性 (續)

項目	K8S	Openshift	Tanzu	Rancher
支援對於應用程式的追蹤	需另外安裝第三方套件(如jaeger與zipkin等)	支援 平台提供安裝套件需安裝(service mesh)	是 (透過Spring Cloud Sleuth，並提供VMware商業技術支援)	是 (內建 Istio / Jaeger / Kiali)
不同雲平台上的叢集管理	否	是 Red Hat Advanced Cluster Management (ACM)提供管理 K8s / AWS EKS / GCP GKE / Azure AKS..等雲端平台	是 透過Tanzu Mission Control管理不同雲平台的叢集	是 可以管理AWS EKS / Azure AKS / GCP GKE / Aliyun ACK / Tencent TKE / Huawei CCE等雲端平台
使用者資源使用的計價功能	否	Yes Cost management / CPU / Mem / Storage等 (SaaS provide by Red Hat)	是	否 (需安裝 Kubecost)

可維護性

項目	K8S	Openshift	Tanzu	Rancher
圖形化介面管理與升級	否 需人工下達指令以後重啟服務	是 含版本升級	是	是
不停機升級	否 需人工重啟kublet等服務	是 (cluster)	是 利用多叢集能力，可透過先建後拆或節點rolling update達成	是
備份與還原	否 需人工識別，如etcd與持久化儲存(PV)等需備份資料並進行備份與還原	是 (OADP)	是 可透過Velero備份/還原YAML與持久化儲存(PV)，VMware提供商業技術支持	是 有提供介面
圖形化yaml檔管理	否 需人工撰寫yaml檔	是	是 可透過Tanzu Application Platform的supply chain生成YAML	是
跨叢集資料自動同步	否 不支援自動同步	是 (option ODF)	是 但可透過Storage or vSAN同步功能來同步PV	否

安全性

項目	K8S	Openshift	Tanzu	Rancher
支援圖形化的使用者與權限管理	否 (需自行下達指令)	是	否 (需自行下達指令)	是
帳號管理與單一登入機制整合	否 (需自行進行整合)	是	否 (需自行進行整合)	是
內建容器映像檔的安全性檢查工具	否 (需使用第三方工具掃描)	是 (Clair)	是 (Trivy)	否 (可另外安裝 SUSE NeuVector)
安全的容器映像檔儲存市集	否 (Docker Hub 等未提供檢查功能)	是 Quay.io	是 (透過 Tanzu Build Service設定)	是 (SUSE BCI Registry)
安全設定檢查功能	否 (需另外安裝或執行第三方套件)	是 平台有提供、需安裝 (Compliance Operator)	是 (TMC / Sonobuoy)	是 (環境安裝可勾選)
對容器弱點掃描功能	否 (需另外安裝或執行第三方套件)	是 (Clair)	否 (需另外安裝或執行第三方套件)	否 (可另外安裝 SUSE NeuVector)

比較的限制

- 未反映價格：一個工具即便好用，但是如果價格與功能之間的比較並不划算，以目前協作工具平台多半可以使用人工做到類似的功能，因此價格還是很重要的考量。
- 未考慮 K8S 主要功能外的功能：本研究的指標是基於 K8S 的功能進行展開，但是或許有些商用工具有額外有特色的功能而超過協作工具的範圍，這就無法在本次研究當中被評估到，這點可能須要各廠商就自身的產品做介紹。
- 未考慮各廠商的支援能力：這和各廠商的人力有關，本研究並未針對各廠商的人力現況進行評估。

整理與建議

- 原生 K8S 有許多不足之處
- 如果經費許可，建議可以設定一個初期的應用環境，並邀請主流廠商就該情境進行提案，以比較其價格功能比。
- 或是先用原生 K8S 架設一個測試環境，先找出適用的應用情境後，再採用商業系統。



計畫背景與目標

容器架構的特色與適用性研究成果摘要

容器架構的導入策略期中研究成果摘要

容器架構的比較框架期中研究成果摘要

整理與建議

感謝 貴單位長官的協助，本計畫工作項目已完成

- 容器架構具有提高資源利用率、提升規模可擴充性、加快應用程式部署速度、提高容錯能力，與易於維護等優點，因此目前國內外許多企業紛紛朝向該架構發展
- 除了建立容器架構外，未來更需要考量到與應用程式的整合。本計畫進行優點與適用性研究，發現容架構對於需求變化大的無狀態應用特別能發揮其效果。但整體上，只要是可以轉到容器架構執行的程式，在方便部署與維護的角度上，都可獲得益處。
- 本計畫也探討了在容器架構導入的過程中，可能會面臨的議題並做出建議
 - 可以從 **Google** 提出的服務可靠性階層來由下而上推動，並且建立 **SRE** 團隊負責相關工作
 - 目前有容器架構安全與管理的最佳實務，可以從容器、叢集或協作平台，作業環境等角度來進行相關工作
- 本計畫基於功能、可維護性、安全性，對市面上的主流產品，和原生 **K8S** 平台進行比較，可考量自身需求，逐步引入容器架構。

謝謝您的聆聽