**P1**

**應實行(較為引人注目的**

1. **原文提到技術債(technical debt)**

由於以下原因，基礎設施和運營（I＆O）仍然存在大量技術債務：

複雜的舊版應用程序和過時的流程擴展了敏捷性，增強了業務

風險，通常會導致成本過高。

**---**

**3.那些平台軟體是合，著重在部署和擴充能力**

**減少開發完成到上線(deploye)的Lead Time**

**P3**

透過大量截取分析資料的數量質量 多樣性 These platforms enable the concurrent use of multiple data sources, data collection methods, and analytical and presentation technologies.

生產率的提高:通過分析IT和業務數據，取得用戶業務活動和提供IT系統的行為。

**P7 ac cele rator**

Deep neural network

Field-programmable gate array

計算加速器

GPU:GPU是一種特殊類型的處理器，具有數百或數千個核心，經過最佳化，可並列執行大量計算 ,對深度學習機器學習演算法執行上特別有效

HPC:high performance computing高效能運算,一般電腦無法達成的計算 像是基因分析比對、氣候模擬、天文科學、太空與各種物理與化學精算。

**P8**

2022 AI計算消耗的資源預計多4x gpu 常出現了 後面兩個1-5%

成熟度：GPU最成熟（市場採用率在5％到20％之間），其次是ASIC和FPGA，

分別是青少年和早期主流（市場採用率從1％到5％）

**P10**

在AWS MARKETPLACE搜尋就可找到非常多的平台看需要建置的環境，OS或其他硬軟體需求

**P11**

iNTEL在2015收購原本的fpga第二大廠

xILINX也有推出fpga embbed的DNN develop kit(DNNDK) <https://aws.amazon.com/marketplace/pp/B07QW4TC85?qid=1586422384802&sr=0-1&ref_=srh_res_product_title>

**P12**

CMS includes container runtimes, container orchestration, job scheduling and resource management

**P13**

CMS 的環境來說windows的整合跟整個環境都是落後於linux環境的

1.3應該較為符合

4.尖峰時段需額外效能(

**P14**

Kubernetes:

**1.管理功能特性**

(1) 多租戶管理介面（Multi-tenancy）  
Kubernetes的多租戶管理以RBAC系統為基礎，管理者可以定義多個系統角色（Cluster Role），各自擁有不同的叢集權限，操作特定的命名空間（namespace）及資源（Resources）。管理者在建立使用者的系統帳戶（context-user）後，透過ClusterRoleBinding資源，將帳戶與單一或複數角色綁定，讓使用者利用帳戶資訊呼叫API時，得以操作相應角色之權限。  
例如：管理者建立一命名空間「Dev」、系統角色「dev-user」、「dev-monitor」、系統帳戶「user」及「monitor」，綁定同名角色及帳戶，並生成相應定義檔（kube-config）。  
dev-user角色僅能在Dev命名空間部署、監控、管理容器應用，而非其他命名空間（如default、kube-system等）；dev-monitor角色則僅能在Dev命名空間監控容器應用。根據管理者發給使用者的kube-config定義檔，使用者在呼叫叢集API時，無論是透過CLI或kube-dashboard，操作權限將因應上述角色定義。

(2) 平台擴充性（Cluster Scalability）  
Kubernetes叢集中，以Node為主要運算節點。當使用者的資源需求增加（CPU、RAM、GPU⋯⋯），而叢集運算資源不足時，維運人員能夠輕易地擴充運算節點。擴充的節點只要能安裝相應版本之Kubernetes元件即可，故無論是硬體規格或作業系統皆相當彈性。維運人員可以因應當下不足的資源，自由配置新的節點設備。

(3) 高可用性管理介面（Master HA）  
無論是管理者或使用者，皆需要取得管理節點（Master）的核心服務（API Server、Scheduler⋯⋯）才能對叢集進行服務部署及管理。而管理節點的核心元件中，Etcd尤為重要。Etcd保存了叢集中所有的網路配置及節點狀態訊息，而部分訊息是由複數個Etcd服務進行投票獲得，故Etcd服務數量（即Master節點數量）須為單數。  
本專案規劃採用3台管理節點實現管理介面的高可用性。正常運作期間，Etcd將選舉出1台Master作為Leader，並由該節點提供用戶端API服務。當該節點崩壞，則由另1台Master接手Leader，持續提供用戶端呼叫API。

(4) 多元儲存介面（Storage Classes）  
根據使用者的容器應用，不同應用所搭配的資料或檔案儲存系統，可能有各式各樣的偏好。Kubernetes支援多種儲存介面，供容器應用介接。其中包括公有雲儲存服務GCEPersistentDisk、AzureDisk、AWSElasticBlockStore、NFS檔案共享系統，以及軟體定義儲存CephFS、RBD等等⋯⋯。  
在支援多種儲存介面之餘，根據所選的儲存介面，使用者也可以定義該儲存區的讀寫模式，供單一或複數個服務讀寫檔案資料。

更多儲存類型請參考：  
https://kubernetes.io/docs/concepts/storage/storage-classes/

(5) 自定義叢集資源（Custom Resource Definitions）  
Kubernetes原生的叢集資源包括Pod、Deployment、DaemonSet、StatefulSet等等⋯⋯。藉由Kubernetes的apiserver-builder，用戶可以自行定義簡單的叢集資源，用自定義的Controller模式部署容器應用。

**2.應用功能特性**

(1) 滾動式升級（Rolling Update）  
Kubernetes自動以纍進方式更新容器，並且同時監控更新程序，避免在更新工作中一次性停擺所有舊版本容器，而造成服務中斷。若更新過程中發現錯誤訊息，Kubernetes也能夠自動回朔容器至更新前的穩定版本。

(2) 應用服務擴展性（Service Scalability）  
根據服務需求，管理人員能夠輕鬆設置該服務的自動擴展（Auto Scaling）。在服務負載量超出正常預期時，Kubernetes將即時增加相應的容器應用，大幅減少維運人員面對臨時超載的工作負擔。

(3) 自動化負載平衡（Service Discovery and Load Balancing）  
在部署容器應用時，Kubernetes會發配給容器相應的IP位址，並藉Service叢集資源實現負載平衡的工作。使用者將無需自行設計負載平衡的功能架構與機制。

(4) 自動重啟失能服務（Self-healing）  
當容器應用出現錯誤訊息並崩潰，Kubernetes能自主利用相同的容器映像檔及部署定義（YAML檔），排程該容器的重啟，以修復服務。若計算節點出錯當機，Kubernetes也能即時在其他計算節點重新部署容器應用，以恢復服務的提供。

(5) 容器網路管理（Network Policy）  
Kubernetes支援多種容器網路介面（CNI），並根據選擇的CNI方案，管理人員能夠自行設置網路管理規則，一方面進一步實現容器間相互溝通的管理，也提供網路安全的解決方案。

**1.管理功能特性**

(1) 多租戶管理介面（Multi-tenancy）  
Kubernetes的多租戶管理以RBAC系統為基礎，管理者可以定義多個系統角色（Cluster Role），各自擁有不同的叢集權限，操作特定的命名空間（namespace）及資源（Resources）。管理者在建立使用者的系統帳戶（context-user）後，透過ClusterRoleBinding資源，將帳戶與單一或複數角色綁定，讓使用者利用帳戶資訊呼叫API時，得以操作相應角色之權限。  
例如：管理者建立一命名空間「Dev」、系統角色「dev-user」、「dev-monitor」、系統帳戶「user」及「monitor」，綁定同名角色及帳戶，並生成相應定義檔（kube-config）。  
dev-user角色僅能在Dev命名空間部署、監控、管理容器應用，而非其他命名空間（如default、kube-system等）；dev-monitor角色則僅能在Dev命名空間監控容器應用。根據管理者發給使用者的kube-config定義檔，使用者在呼叫叢集API時，無論是透過CLI或kube-dashboard，操作權限將因應上述角色定義。

(2) 平台擴充性（Cluster Scalability）  
Kubernetes叢集中，以Node為主要運算節點。當使用者的資源需求增加（CPU、RAM、GPU⋯⋯），而叢集運算資源不足時，維運人員能夠輕易地擴充運算節點。擴充的節點只要能安裝相應版本之Kubernetes元件即可，故無論是硬體規格或作業系統皆相當彈性。維運人員可以因應當下不足的資源，自由配置新的節點設備。

(3) 高可用性管理介面（Master HA）  
無論是管理者或使用者，皆需要取得管理節點（Master）的核心服務（API Server、Scheduler⋯⋯）才能對叢集進行服務部署及管理。而管理節點的核心元件中，Etcd尤為重要。Etcd保存了叢集中所有的網路配置及節點狀態訊息，而部分訊息是由複數個Etcd服務進行投票獲得，故Etcd服務數量（即Master節點數量）須為單數。  
本專案規劃採用3台管理節點實現管理介面的高可用性。正常運作期間，Etcd將選舉出1台Master作為Leader，並由該節點提供用戶端API服務。當該節點崩壞，則由另1台Master接手Leader，持續提供用戶端呼叫API。

(4) 多元儲存介面（Storage Classes）  
根據使用者的容器應用，不同應用所搭配的資料或檔案儲存系統，可能有各式各樣的偏好。Kubernetes支援多種儲存介面，供容器應用介接。其中包括公有雲儲存服務GCEPersistentDisk、AzureDisk、AWSElasticBlockStore、NFS檔案共享系統，以及軟體定義儲存CephFS、RBD等等⋯⋯。  
在支援多種儲存介面之餘，根據所選的儲存介面，使用者也可以定義該儲存區的讀寫模式，供單一或複數個服務讀寫檔案資料。

更多儲存類型請參考：  
https://kubernetes.io/docs/concepts/storage/storage-classes/

(5) 自定義叢集資源（Custom Resource Definitions）  
Kubernetes原生的叢集資源包括Pod、Deployment、DaemonSet、StatefulSet等等⋯⋯。藉由Kubernetes的apiserver-builder，用戶可以自行定義簡單的叢集資源，用自定義的Controller模式部署容器應用。

**2.應用功能特性**

(1) 滾動式升級（Rolling Update）  
Kubernetes自動以纍進方式更新容器，並且同時監控更新程序，避免在更新工作中一次性停擺所有舊版本容器，而造成服務中斷。若更新過程中發現錯誤訊息，Kubernetes也能夠自動回朔容器至更新前的穩定版本。

(2) 應用服務擴展性（Service Scalability）  
根據服務需求，管理人員能夠輕鬆設置該服務的自動擴展（Auto Scaling）。在服務負載量超出正常預期時，Kubernetes將即時增加相應的容器應用，大幅減少維運人員面對臨時超載的工作負擔。

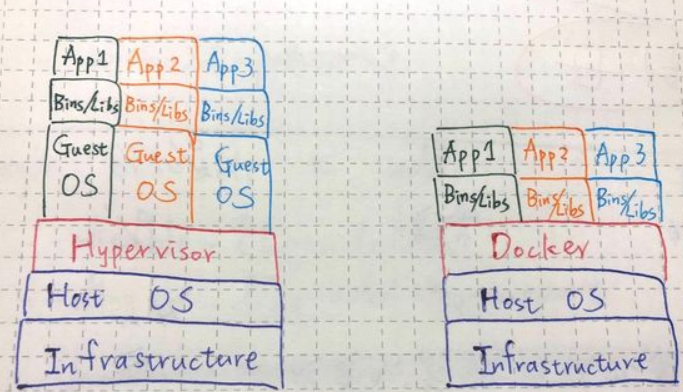
(3) 自動化負載平衡（Service Discovery and Load Balancing）  
在部署容器應用時，Kubernetes會發配給容器相應的IP位址，並藉Service叢集資源實現負載平衡的工作。使用者將無需自行設計負載平衡的功能架構與機制。

(4) 自動重啟失能服務（Self-healing）  
當容器應用出現錯誤訊息並崩潰，Kubernetes能自主利用相同的容器映像檔及部署定義（YAML檔），排程該容器的重啟，以修復服務。若計算節點出錯當機，Kubernetes也能即時在其他計算節點重新部署容器應用，以恢復服務的提供。

(5) 容器網路管理（Network Policy）  
Kubernetes支援多種容器網路介面（CNI），並根據選擇的CNI方案，管理人員能夠自行設置網路管理規則，一方面進一步實現容器間相互溝通的管理，也提供網路安全的解決方案。

**P15**

**Docker Container(DC)與VM間的差異**:DC是將作業系統層虛擬化，VM則是虛擬化硬體。

****

**P16**

is an enterprise-ready Kubernetes container platform with full-stack automated operations to manage hybrid cloud and multicloud deployments. Red Hat OpenShift is optimized to improve developer productivity and promote innovation.

<https://www.ibm.com/downloads/cas/7019JAXP>

**P22**

CDN content delivery network >如存取網站離用戶太遠 使用CDN將資料拉到使用者所在地區(cache)減少存取時間

1.用原有或較有名的 name server

2.確保隨時都有 SLA service leve agreement

3.Edge DNS 特別提到他們不識使用bind(Berkeley internet name domain)

4.偵測異常行為的IP 對其進行封鎖或終止服務

**P24**

Mission ctrl=集中管理雲端的K8s (PKS)

App service=提供部署.net spring 應用程式，用 Spring 微服務模式與 Spring Boot 的 jar 執行檔，

加速雲原生 Java 應用程式的開發。(運用 REST、WebSocket、Reactive、訊息、資料、整合與批次功能來建置、執行與擴充微服務)，並在 Tanzu Application Service 上按現況執行。

Catalog=維護原始碼元件和應用程式目錄

K8s grid=簡化 Kubernetes 安裝與多雲環境內的次要作業

Observability=提供指標導向的大規模分析，可以自訂指標，主動監控程式問題提早鎖定問題。

Service mesh=在 API 層級以更高的能見度掌控服務、資料及使用者

**P25**

Translation and validation:將一些比較高層級的商業策略輸入到預先設定好的網路環境: It can take a higher-level business policy as input from end users and convert it to the required network configuration.

Automation: 配置調整現有的網路It can configure appropriate network changes across existing network

State awareness: 擷取當前的網路狀態(監控)The system ingests real-time network status for systems under its control.

Assurance and dynamic optimization: (持續驗證、確保網路有跟企業意象相同)The system continuously validates that business intent is being met and can take corrective action when it isn’t.

網路整體朝同方向進行，就比較少需要特別處理的事情或使用特別工具

**P28**

Apstra 是一家提供網路方面解決方案，設計 架構 部署運行自動化的網路架構

所謂的意圖可能是業務單位希望加速回應客戶的要求，當然現在還沒辦法直接轉換人類的意圖為控制電腦系統的指令，但網管人員的確可以將業務需求轉譯成IT語言，更為技術性的描述，如：「我們必須部署一個EVPN leaf/spine網路來支援我們的企業應用程式，以及各個應用程式層所需的策略，以及該應用程式的特定 SLA，其中包括延遲和丟包要求。」

**P29**

特別提到只有這三個的cost, reliability and performance attributes of DRAM and flash memory比較可行

其中3D Xpoint在2020最有競爭力的

**P32**

**Non vo la tal**

優勢是平行處理跟低延遲

1.     SSA屬於具備可擴展性的專有解決方案，完全基於固態半導體技術以達成資料儲存功能，而且任何時候都無法利用HDD技術進行配置。不同於ECB（即基於控制器的外部儲存）陣列中的純SSD硬碟櫃；

2.     SSA必須是具備特定名稱與型號的獨立產品，其通常（但並非總是）包括針對固態儲存技術的作業系統與資料管理軟體。

**P36**

1.新技術 不容易找到對它有經驗的員工和工具

2.以前是擔心一個大的應用程式或微服務 現在是數十數百個獨立功能 生態系統改變，需要新的管理方法

3.如果組織習慣對雲使用是預付的話 無伺服計算會對組織帶來一些不確定性跟挑戰

<https://kknews.cc/zh-tw/tech/nmx58v8.html>