

基於Kubernetes與Docker之 容器化智慧預測保養系統之全廠導入

(Factory-wide Deployment of IPM_C based on Kubernetes and Docker)

講師：洪敏雄

中國文化大學資訊工程學系 教授

洪敏雄、謝昱銘、鄭芳田

國立成功大學 智慧製造研究中心

Intelligent Manufacturing Research Center (iMRC)

National Cheng Kung University

September 27, 2020



演講大綱

2

- 智慧製造平台-基於容器技術之先進製造物聯雲(AMCoT_C)
- 容器化智慧預測保養系統(IPM_C)導入面板製造廠之成果
- 基於Kubernetes與Docker之IPM_C全廠導入實踐作法
- 結語與展望



智慧製造平台

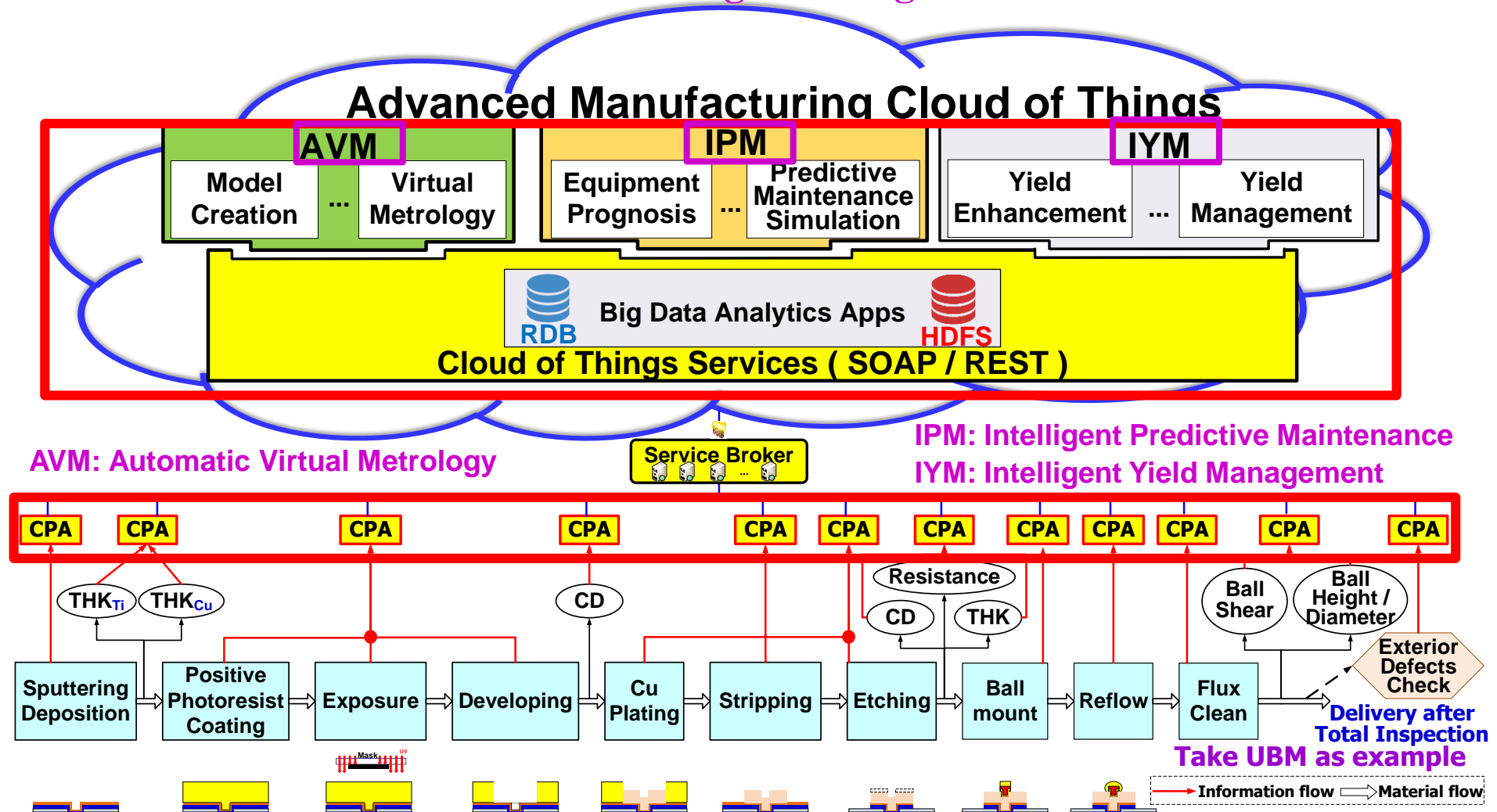
基於容器技術之先進製造物聯雲(AMCoT_C)



智慧製造平台-先進製造物聯雲(AMCoT)

4

- Cloud Side contains various Intelligent CMfg Services



- Factory Side contains various IoT edge devices, i.e., CPA (Cyber-Physical Agent)



工廠端物聯網裝置CPA

(Cyber-Physical Agent)

5

- **Communication Service:**

Facilitate communications with different cyber systems (e.g., CPAs and cloud services) over network, particularly the Internet.

- **CPA Control Kernel:**

The core for communications among all modules of CPA so as to serve as the bridge between cyber and physical worlds.

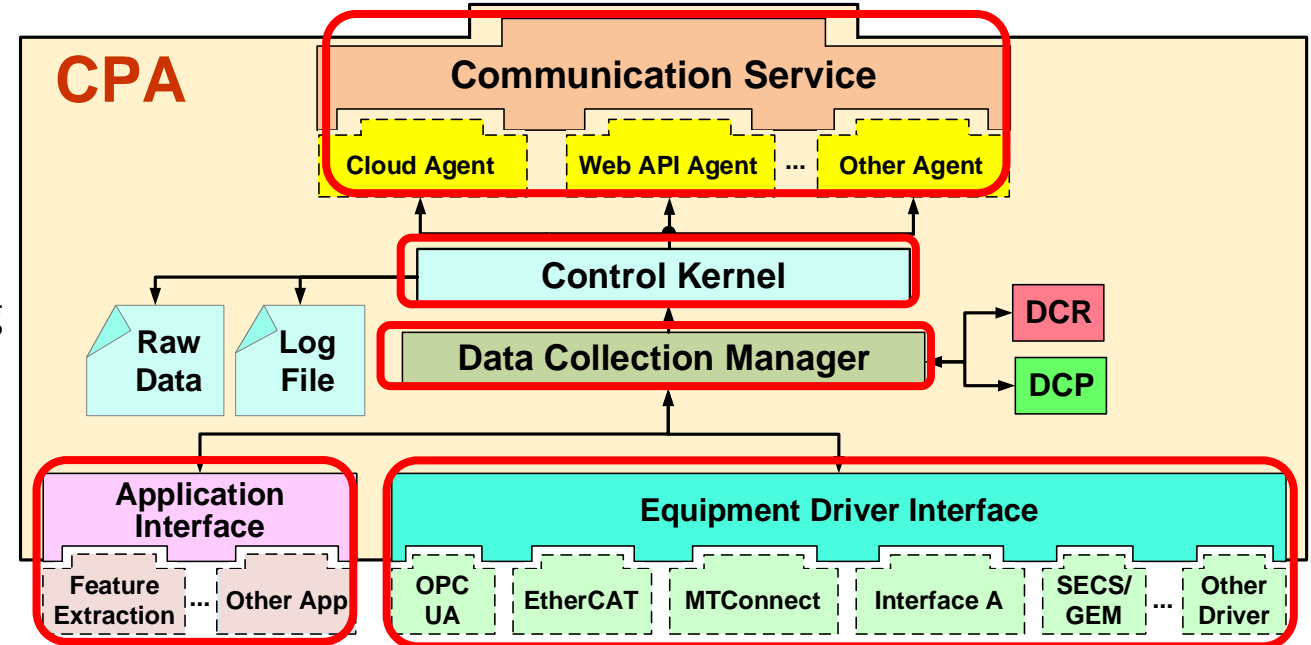
- **Data Collection Manager:**

Handle and manage data collection through Equipment Driver.

- **Application Interface (AI):**

Allow CPA to add various PAMs (Pluggable Application Modules)(e.g., data preprocessing, **feature extractions, predictive maintenance applications, etc.**)) in a plug-and-play manner.

Framework of CPA



Taiwan Patent : I225606
US Patent : 7,162,394
Japan Patent : 4303640

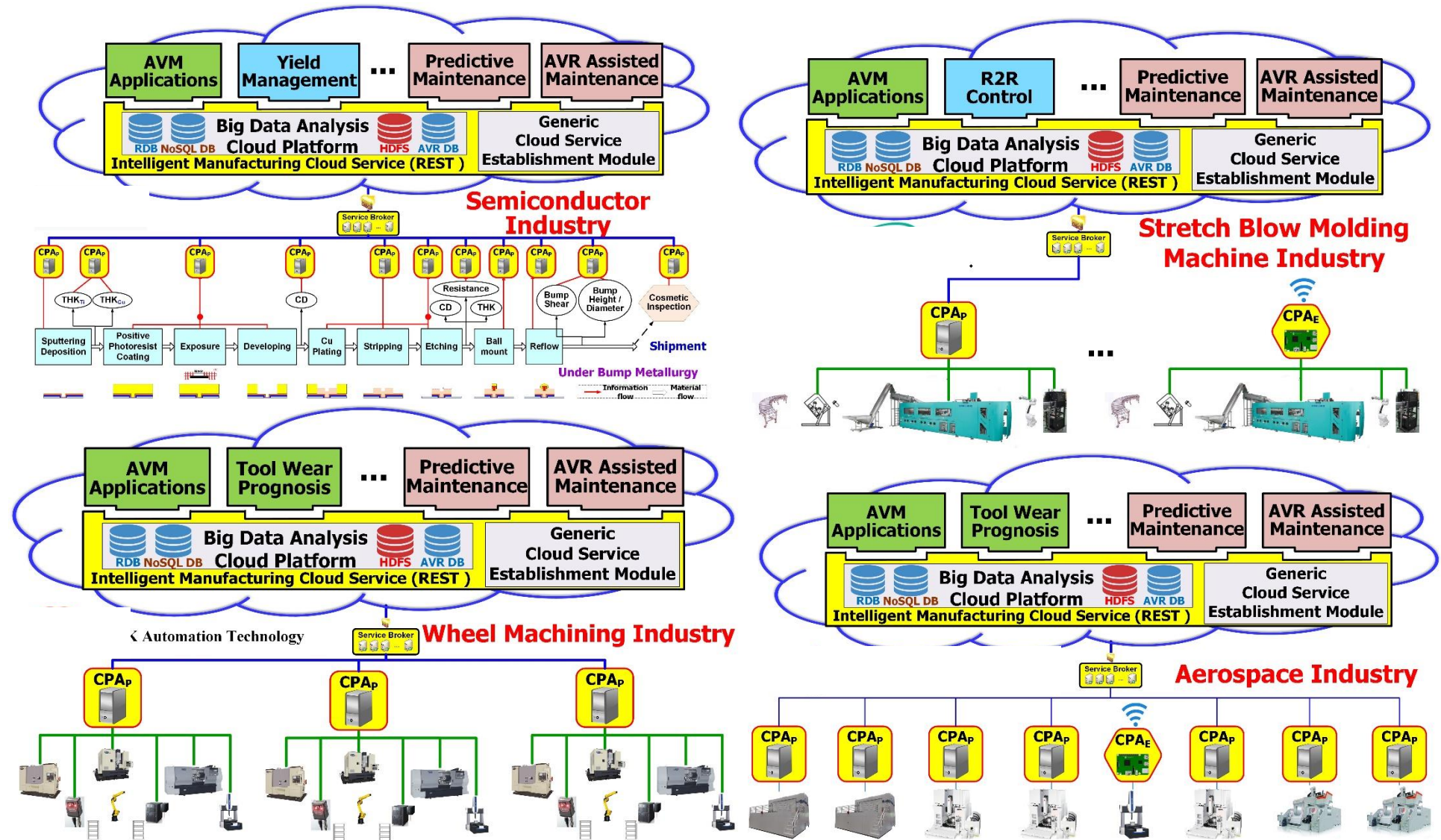
- **Equipment Driver:**

Enable CPA to communicate with various kinds of sensors, devices, machines, and equipment for the purpose of data collection.



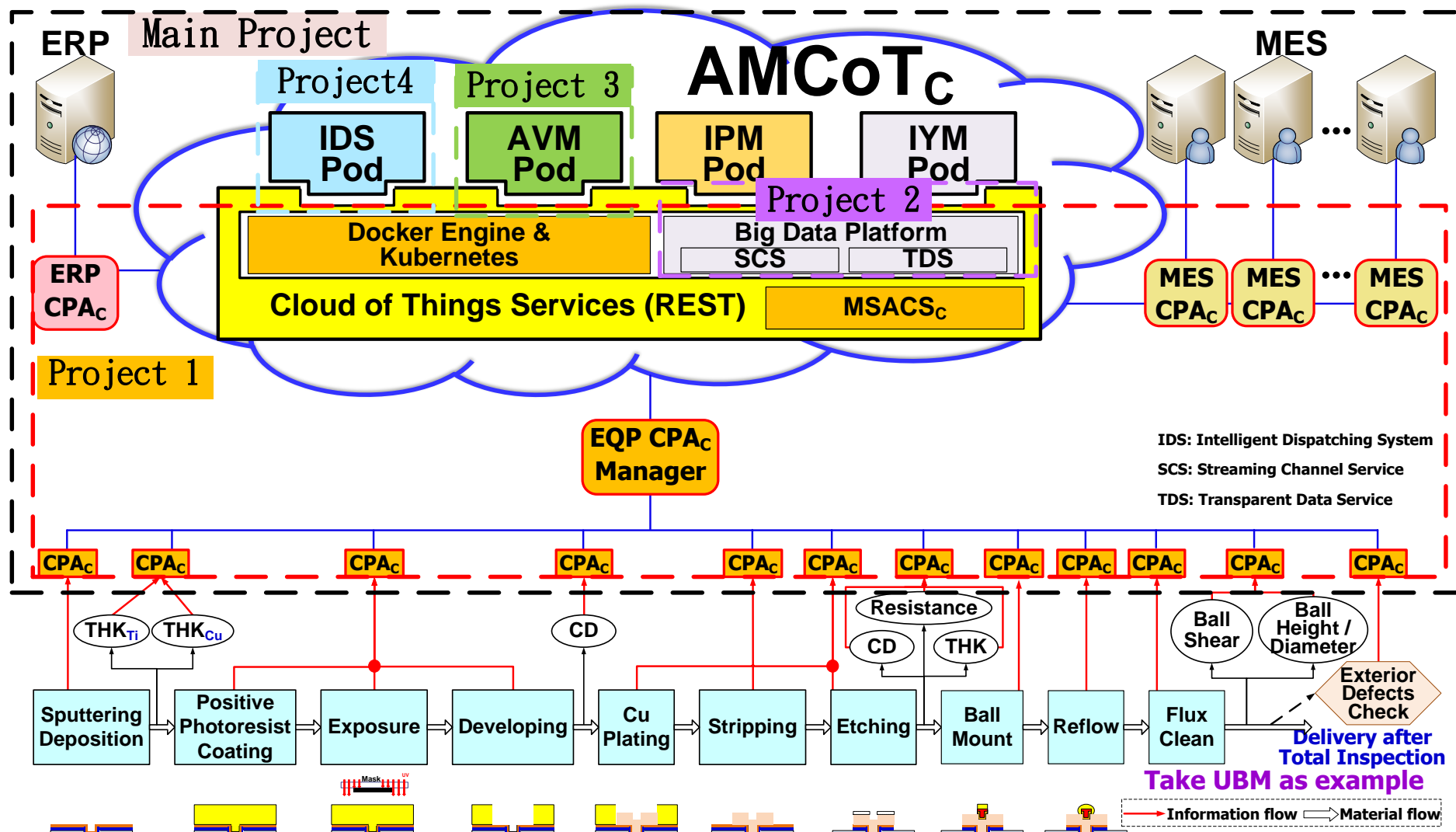
AMCoT之製造產業應用

6



基於容器技術之智慧製造平台AMCoT_C

7



容器化智慧預測保養系統(IPM_C) 導入面板製造廠之介紹



薄膜電晶體製程

(TFT Manufacturing Process)

9

- The TFT process consists of five layers: gate, semiconductor, data, protection, and indium tin oxide (ITO) layers. Each layer includes the so-called photo engraving processes (PEP) as shown in Fig. 2.

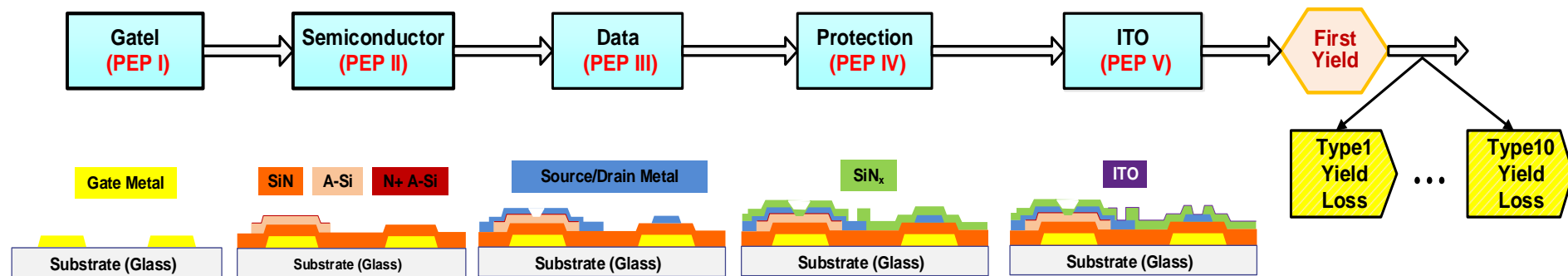


Fig. 1. TFT manufacturing process.

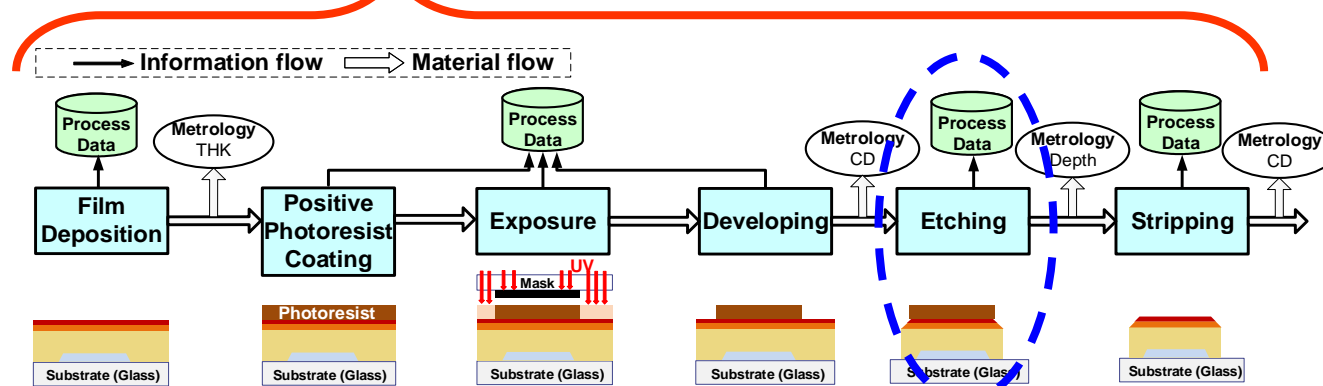


Fig. 2. PEP flow of the semiconductor layer.



乾蝕刻製程之渦輪真空幫浦(Turbo Pump) (1/2)

10

- 乾蝕刻：乾蝕刻是以電漿(因此又稱電漿蝕刻)來進行薄膜蝕刻的一種技術。又分為：**濺鍍(Sputter)**及**純化學蝕刻(Pure Chemical Etching)**。
- 電漿刻蝕是現今技術中唯一能極有效率地將此工作在高良率下完成的技術，**因此電漿刻蝕便成為半導體製造以及TFT LCD Array製造中的主要技術之一**。
- Fig.3為乾蝕刻機台之CEM圖，其中Turbo pump協助在製程中穩定壓力、保持生產品質。

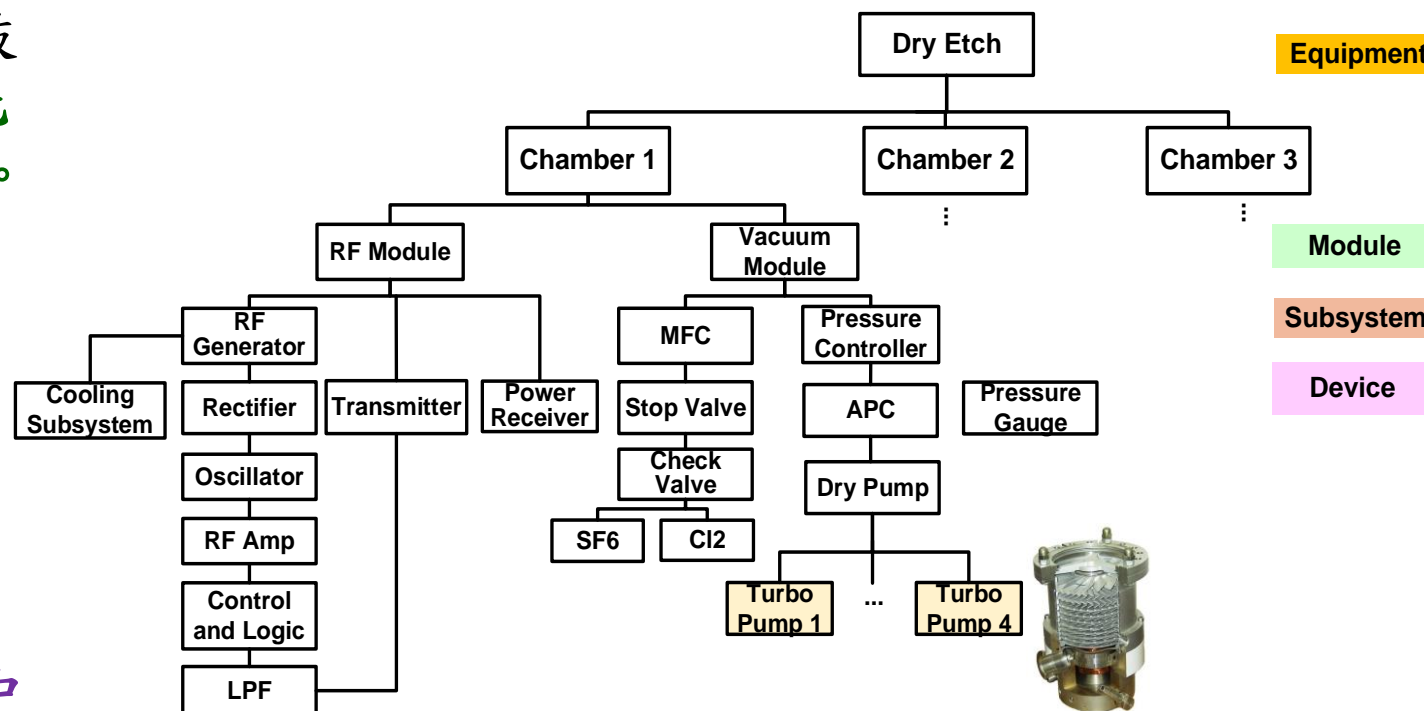


Fig.3. Common Equipment Model(CEM) of Dry Etch



乾蝕刻製程之渦輪真空幫浦(Turbo Pump) (2/2)

11

- 由於乾蝕刻過程會產生揮發物使得**Turbo pump**在抽**Chamber**內的氣體進行維持壓力作業時，一併將揮發物吸出，如**Fig.4**紅色方框所示。
- 如此一來，會使得Turbo pump葉片上沉積汙染分子，致軸偏度歪斜進而pump腔體炸裂**損壞到Dry Etch機台本身，造成極大的損失。**
- 因此可透過**IPM**系統針對**Turbo pump**之健康狀態進行嚴密的監控，避免腔體炸裂所造成的損失。

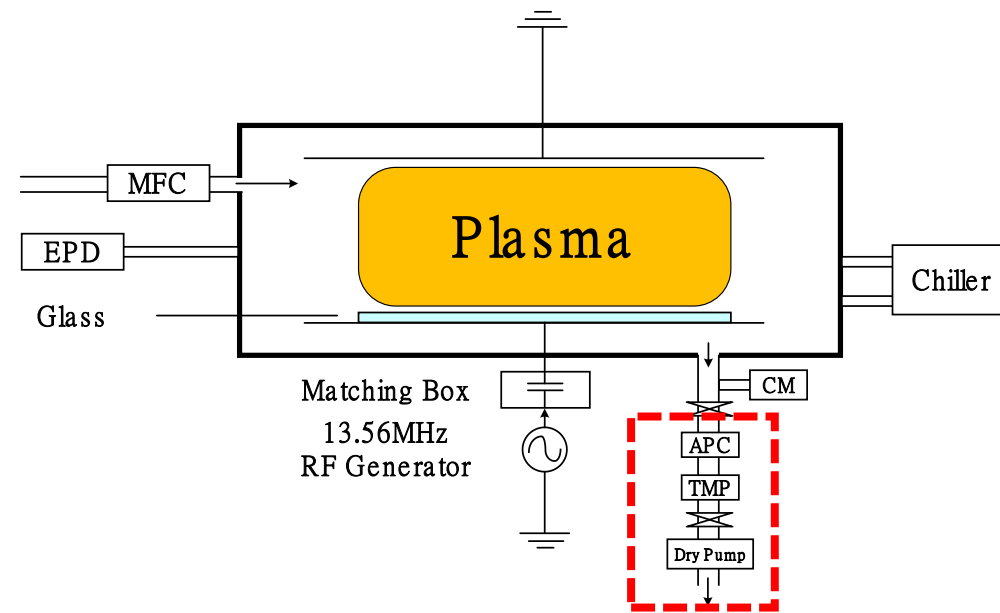
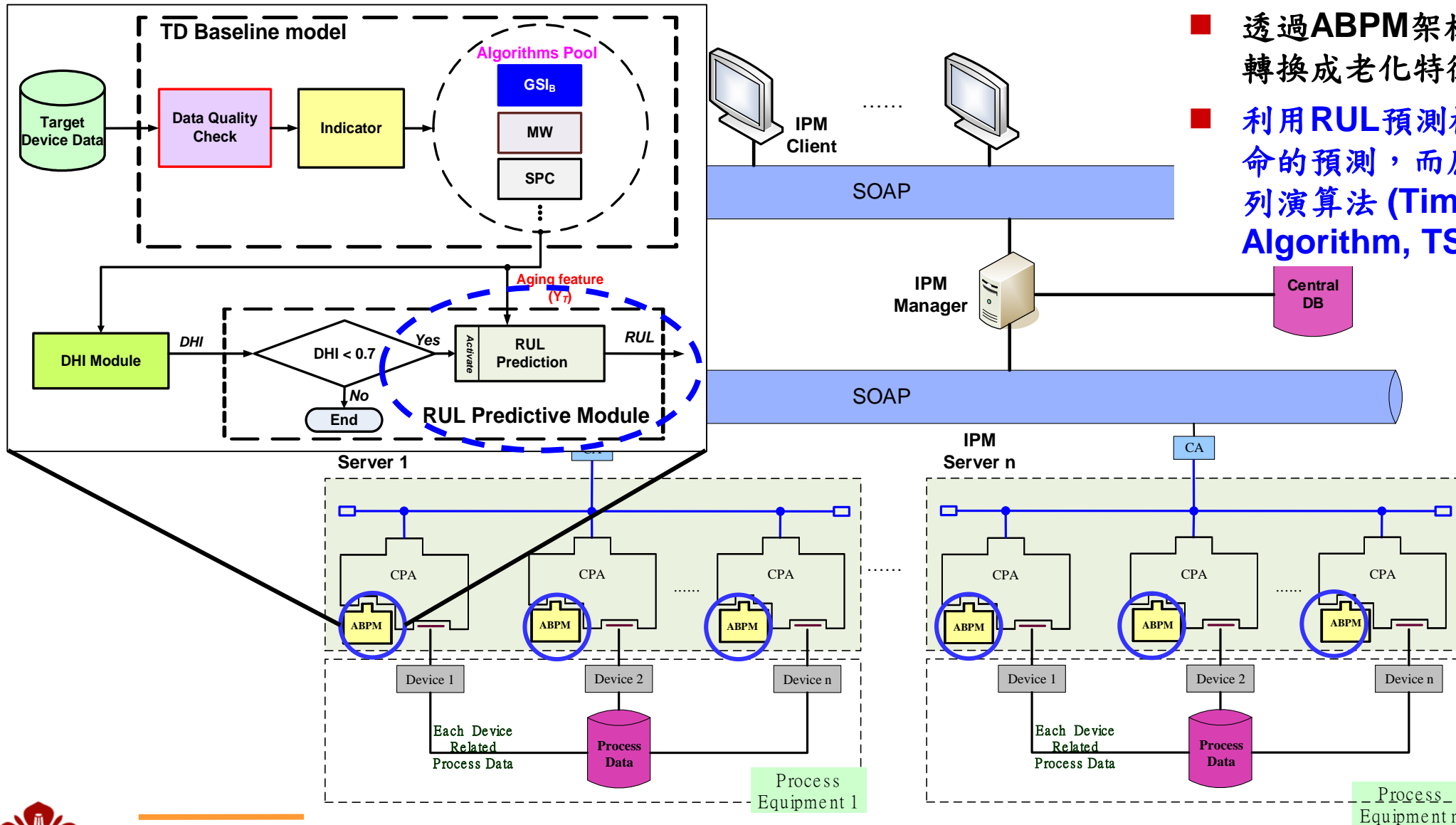


Fig.4. Dry Etch Chamber 機構圖



智慧型預測保養系統ABPM

12



- 透過ABPM架構，將Pump相關參數轉換成老化特徵。
- 利用RUL預測模組進行Pump剩餘壽命的預測，而應用之演算法為時間序列演算法 (Time Series Prediction Algorithm, TSP)。

Fig.5. IPM Framework



IPM_C系統全廠導入之利基

13

- 由於整廠共有100多個Turbo pump需被監控，若以原IPM架構，監控一個Turbo pump需要一台CPA，將需要大量的硬體及維護成本，手動維運這些IPM程式也是複雜挑戰的工作。
- 改以基於Kubernetes與Docker之IPM_C進行全廠導入，可帶來許多好處：
 - 易於採用Microservice架構實現IPM，可個別對IPM之組成元件進行更新與擴展。
 - 利用Images打包IPM各個組成元件與所需執行環境，提升移植性。
 - 利用Container提升IPM程式之隔絕性。
 - 利用自動化腳本減少部署IPM之複雜度，達到快速部署。
 - 一個節點(CPA)可同時部署許多個IPM_C，大幅節省硬體成本。
 - 利用Kubernetes自動化維運IPM_C: Self-healing, Failover, Resource Limitation, etc.



IPM_C系統之全廠導入架構

14

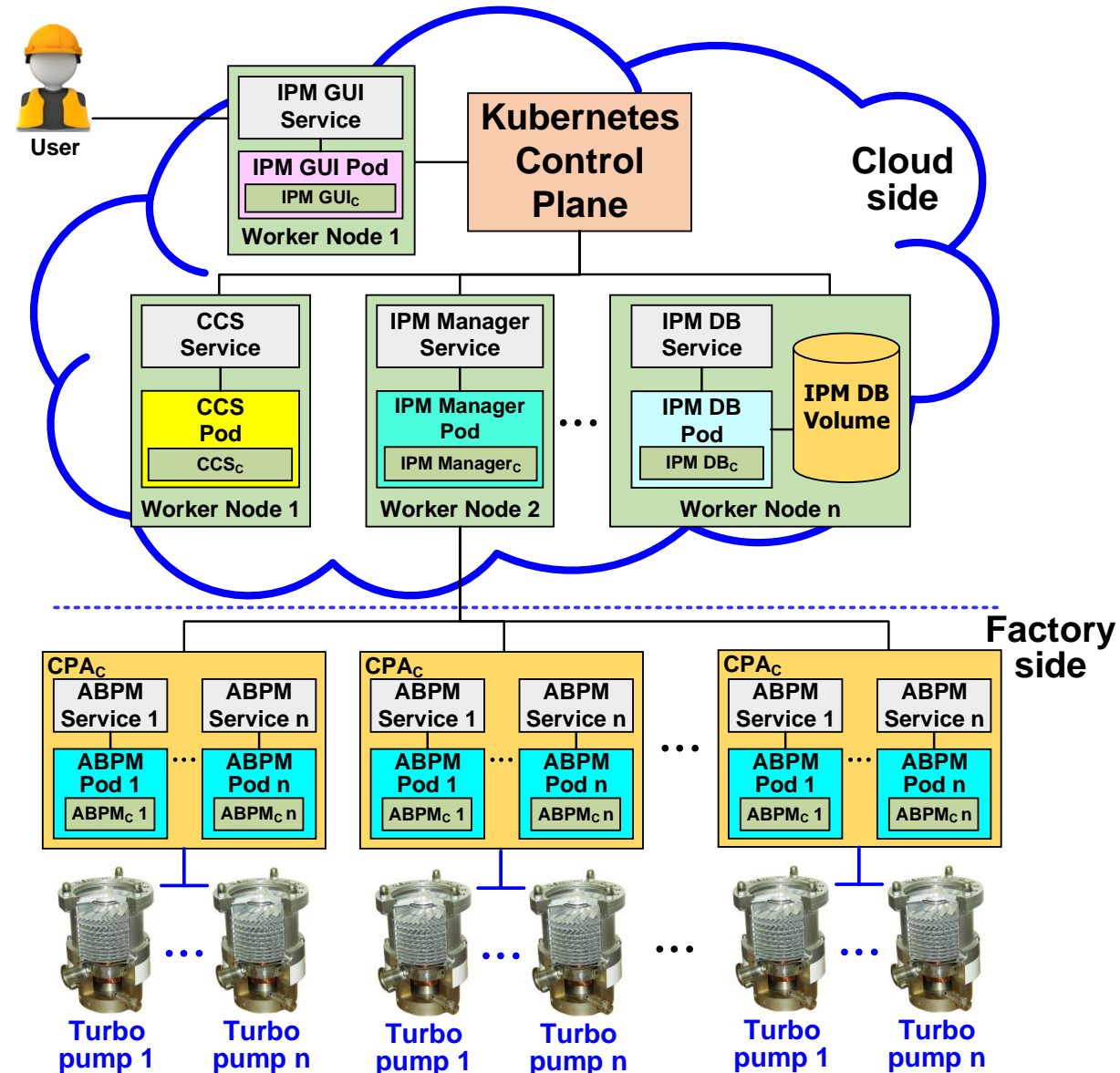


Fig.8. IPM_C Framework

基於Kubernetes與Docker之 IPM_C全廠導入實踐作法

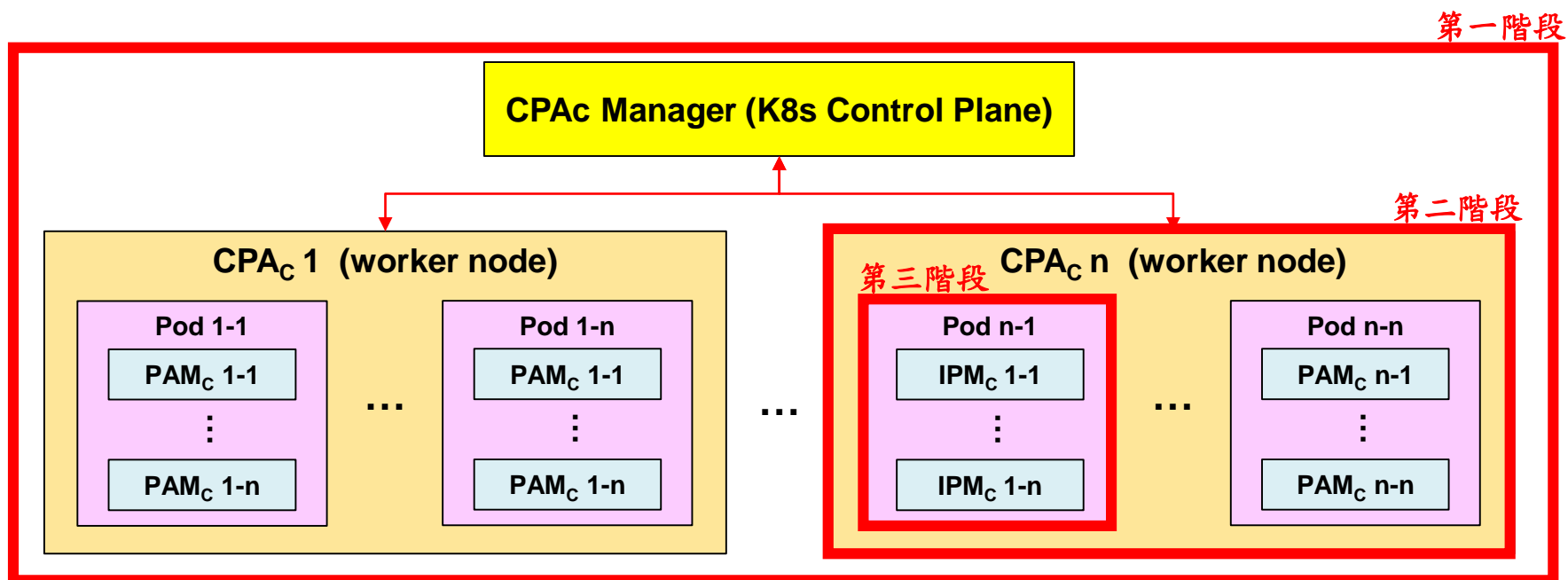


IPM_C之三階段部署策略

16

■ 部署IPM_C之三個階段策略：

- 第一階段：離線安裝Kubernetes與Docker環境
- 第二階段：離線部署Cluster
- 第三階段：離線部署IPM_C



第一階段： 離線安裝Kubernetes與Docker環境



Windows Node安裝方法 (無Internet)

18

1. 在主機上安裝 Windows Server 2019 Standard :
2. 將安裝包下載至儲存裝置中 :
3. 將安裝包複製到離線主機中，在安裝包目錄下執行以下指令 :

```
> #以Administrator執行
```

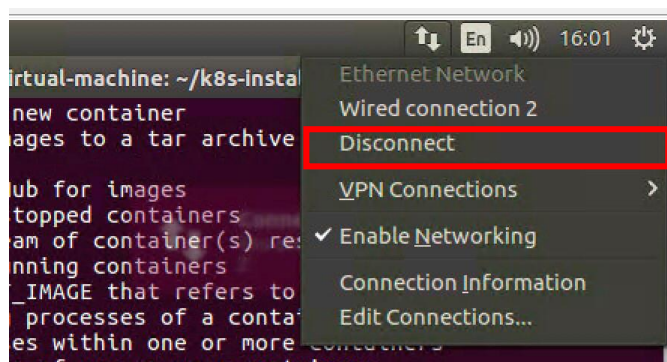
```
install-k8s-for-windows-server-withtout-internet.bat
```



Ubuntu Node安裝方法 (無Internet)

19

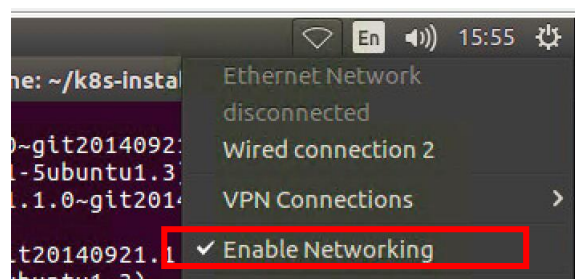
1. 在主機上安裝Ubuntu OS LTS：
2. 將安裝包下載至儲存裝置中：
3. 將安裝包複製到離線主機中，並暫時關閉主機的網路，直到安裝結束：



4. 在安裝包目錄下執行以下指令：

```
$ su root
$ ./install-k8s-for-ubuntu-server-without-internet.sh
```

5. 恢復網路。



第二階段： 離線部署Kubernetes Cluster



Kubernetes Cluster部署 (1/2)

21

1. 修改環境設定檔(檔名為.env)：

```
metadata
  description: production environment
NetworkSpec:
  BackendType: host-gw
NodeSpec:
```

```
-
  OperatingSystemType: ubuntu
  Role: Master
  UserName: autolab
  Password: auto123456
  IPAddress: 192.168.50.199
  WorkPath: /home/autolab/
```

```
-
  OperatingSystemType: windows
  Role: Worker
  UserName: Administrator
  Password: @ut123456
  IPAddress: 192.168.50.100
  WorkPath: C:\\k\\
```

(省略)

值: host-gw or overlay
說明: 設定容器網路類型

值: [array]
說明: 表列所有Node

每個Node皆由指定的Key描述:
OperatingSystemType: 作業系統類型, 目前支援 windows,ubuntu
Role: Node的類型, 指定Worker或Master
UserName: ssh的使用者名稱
Password: ssh的使用者密碼
IPAddress: Node的IP
WorkPath: 使用者的工作目錄

每個Node的描述格式皆相同, 此例為windows node, 根據Node數量依序描述即可。



Kubernetes Cluster部署 (2/2)

22

- ## 2. 執行自動佈署腳本：在Master Node上執行以下指令

```
$ ./init-without-internet.sh
```

3. 等待佈署完成，可看到以下畫面，取得登入kubernetes dashboard登入所需的token：

[illegible]

此為kubernetes dashboard登入所需的token。

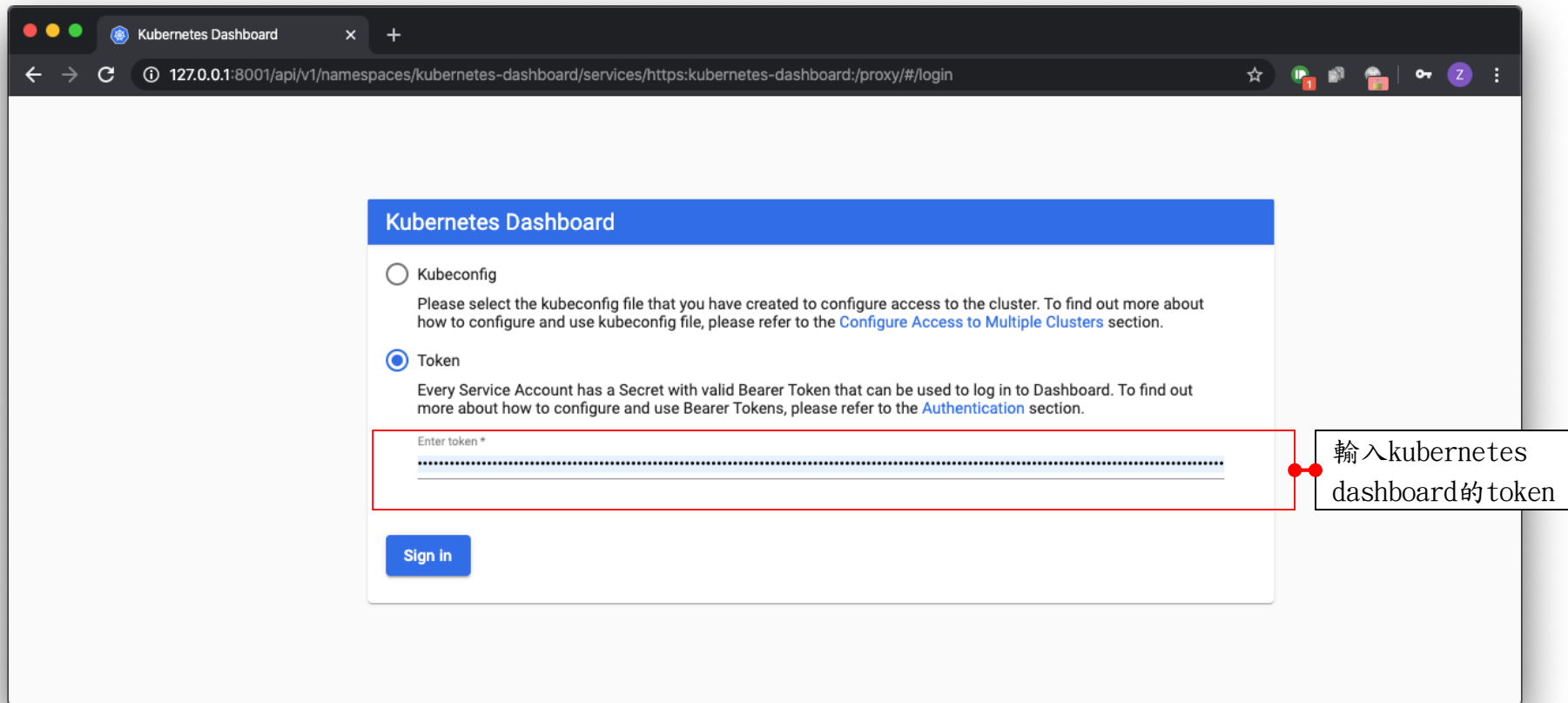


檢視部署成果(1/2)

23

1. 打開Kubernetes Dashboard:

<http://127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/login>



Kubernetes Dashboard

☐ Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.

☒ Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

Enter token *

Sign in

輸入kubernetes dashboard的token



檢視部署成果(2/2)

24

2. 可在Node分頁看到剛剛部署的所有Node，並且狀態都是顯示Ready：

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
upb-01	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	750.00m (18.75%)	100.00m (2.50%)	120.00Mi (1.53%)	220.00Mi (2.81%)	4 months
upb-02	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	100.00m (2.50%)	100.00m (2.50%)	50.00Mi (0.64%)	50.00Mi (0.64%)	4 months
upb-03	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	100.00m (2.50%)	100.00m (2.50%)	50.00Mi (0.64%)	50.00Mi (0.64%)	7 days
upb-04	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	100.00m (2.50%)	100.00m (2.50%)	50.00Mi (0.64%)	50.00Mi (0.64%)	4 months
upb-05	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	100.00m (2.50%)	100.00m (2.50%)	50.00Mi (0.64%)	50.00Mi (0.64%)	4 months
upb-06	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux	True	100.00m (2.50%)	100.00m (2.50%)	120.00Mi (1.53%)	220.00Mi (2.81%)	4 months
win-jtrke7tdm3t	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: windows	True	0.00m (0.00%)	0.00m (0.00%)	0.00 (0.00%)	0.00 (0.00%)	a month
win-m71hkpoqaja	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: windows	True	0.00m (0.00%)	0.00m (0.00%)	0.00 (0.00%)	0.00 (0.00%)	4 months
win-qa8ssoqgupt	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: windows	True	0.00m (0.00%)	0.00m (0.00%)	0.00 (0.00%)	0.00 (0.00%)	5 days



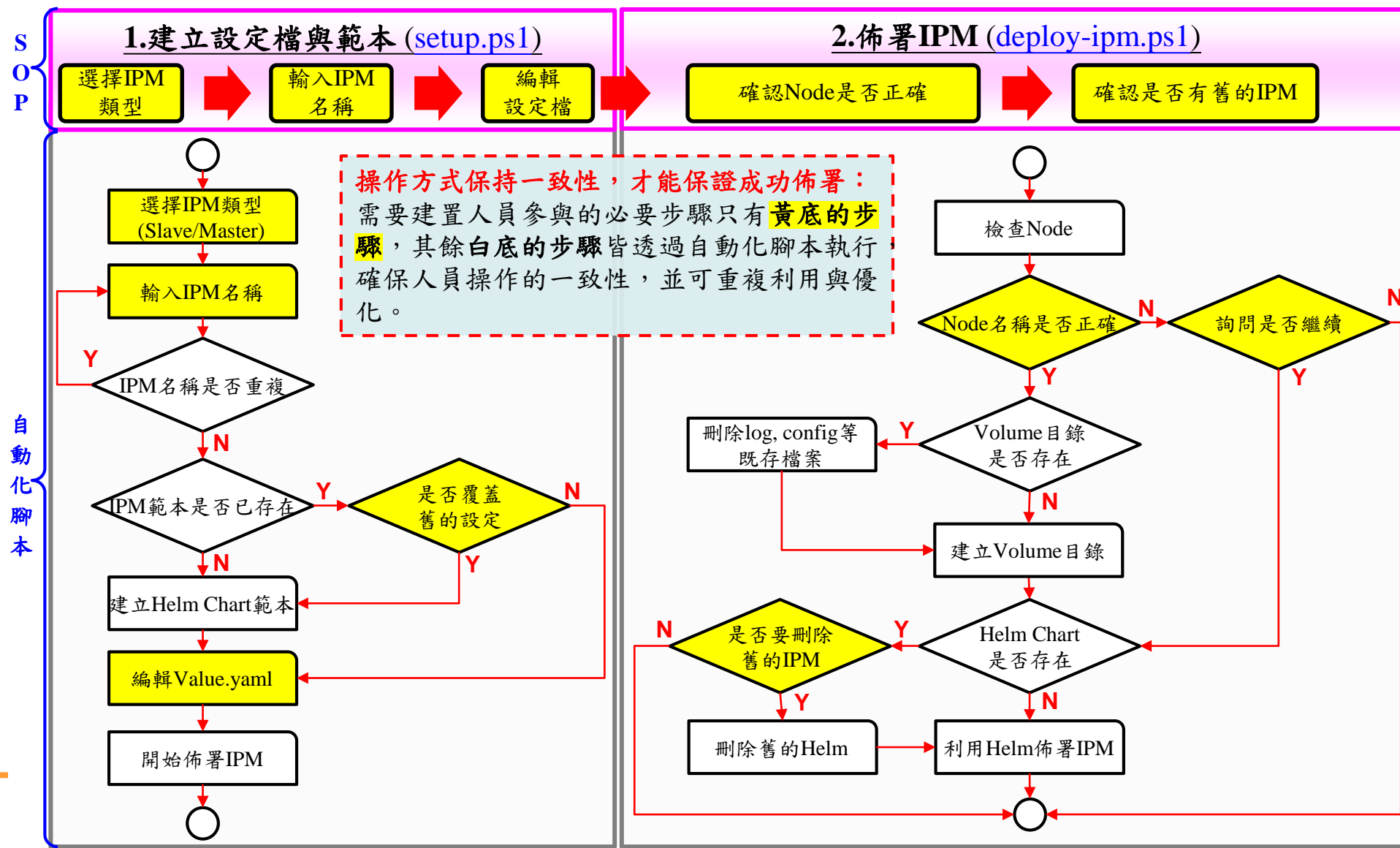
第三階段： 離線部署IPM_C



IPM_C離線安裝流程 (1/2)-標準化安裝流程之設計

26

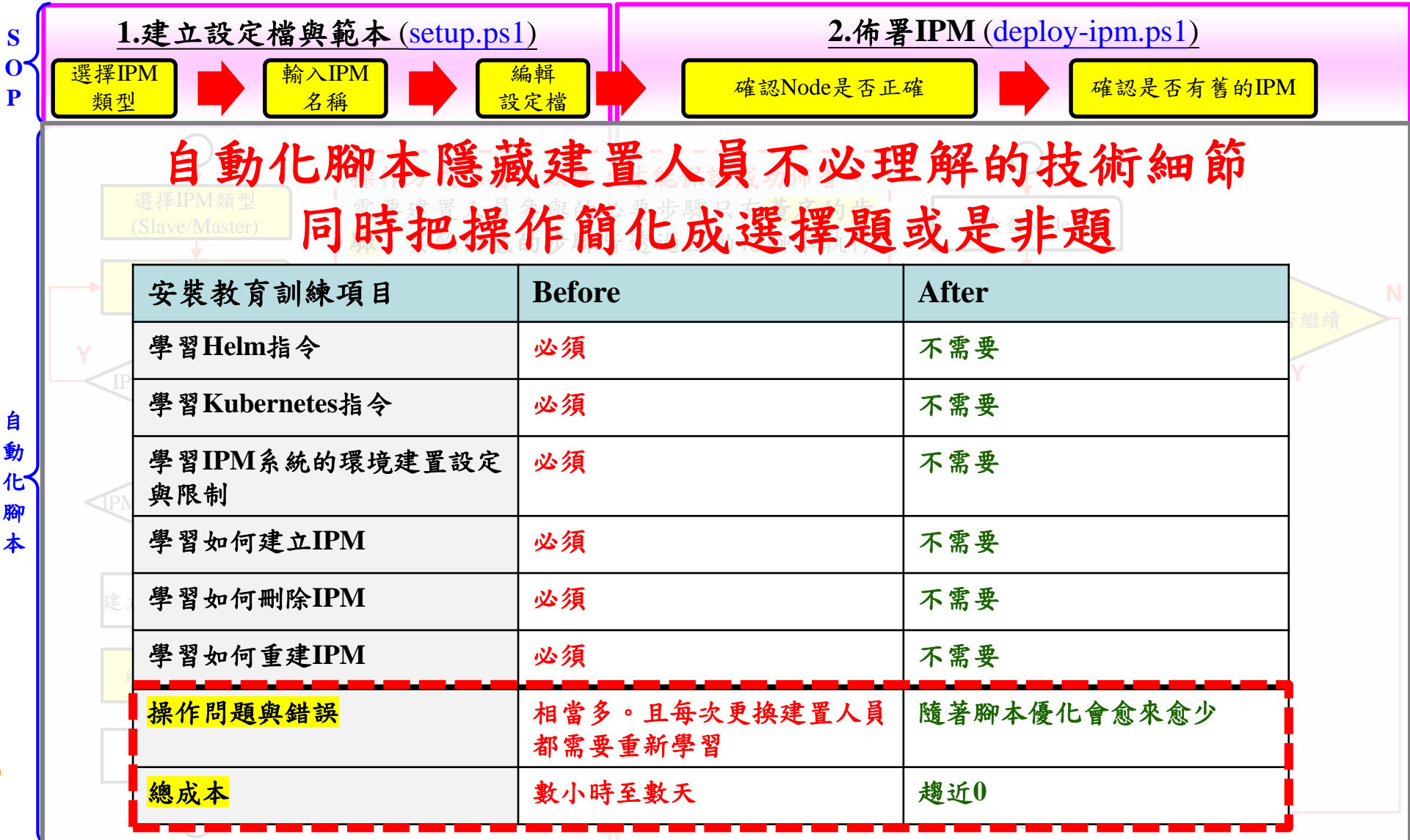
為了減少建置人員的安裝問題與人為疏失，並利於持續優化安裝流程，必須將安裝流程SOP化，並透過自動化腳本執行。IPM的安裝流程分成兩個步驟：1.建立設定檔與範本、2.佈署IPMc：



IPM_C離線安裝流程 (2/2)-自動化腳本隱藏技術細節之效益

27

為了減少建置人員的安裝問題與人為疏失，並利於持續優化安裝流程，必須將安裝流程SOP化，並透過自動化腳本執行。IPM的安裝流程分成兩個步驟：1.建立設定檔與範本、2.佈署IPMc：



部署成果：K8s Dashboard

28

kubernetes Search

Workloads > Pods

default

Overview

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods**
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

- Ingresses
- Services

Config and Storage

- Config Maps
- Persistent Volume Claims
- Secrets

Custom Resource Definitions

Settings

Pods

Name	Names/Labels	Node	Status	Restart	CPU Usage (cores)	Memory Usage (bytes)	Age
ipm-slave-gui-58f54647c7-6mbks	app.kubernetes.io/instance: ipm-slave default	win-2u4kqc	Running	0	-	-	14 hours
Show all							
ipm-slave-parserservice-56f7969b6f-qvkn2	app.kubernetes.io/instance: ipm-slave default	win-2u4kqc	Running	0	-	-	14 hours
Show all							
ipm-slave-service-75869bc664-ll59s	app.kubernetes.io/instance: ipm-slave default	win-2u4kqc	Running	0	-	-	14 hours
Show all							
ipm-slave-tokenserver-568998bf96-lqwzk	app.kubernetes.io/instance: ipm-slave default	win-2u4kqc	Running	0	-	-	14 hours

127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/pod?namespace=default



部署IPM與部署IPM_C之時間成本比較



手動部署IPM的時間成本

30

- 下表為部署一套IPM系統所需的**工作項目**與對應的手動部署時間成本，我們通常會根據效能需求以決定所需要的**主機數量N**；根據業務需求以決定所需要的**IPM數量M**。合計手動部署IPM的時間成本約為 **$30N+50M+50$** 分鐘：

工作項目	子項目	手動部署成本(分鐘)	執行次數	總計
建置環境	1. 安裝.Net Core 2. 安裝SQL Server Native Client 3. 安裝MATLAB Compiler Runtime 4. 啟動IIS	30	N	$30N$
部署程式	1. 下載主程式 1. Data Processing 2. Alg Service 3. GUI 4. Parser Service 5. Token Service 2. 設定參數 1. 編輯Config檔 2. 設定ODBC 3. 設定IIS 1. 建立Website 2. 選擇AppPool 3. 指定應用程式根目錄 4. 調整目錄權限	40 (5種類型的主程式， 部署一個約8分鐘)	M+1 (1個Master與M個Slave)	$40M+40$
啟動程式	1. 檢查相依性服務是否已啟動 2. 啟動IIS Website或IIS App	10	M+1 (1個Master與M個Slave)	$10M+10$



採用前述優化部署流程部署IPM_C之時間成本

31

- 優化部署流程後，僅需編輯Helm Chart設定檔Values.yaml即可完成所有的部署工作。合計手動部署IPMc的時間成本約為**5M+5**分鐘：

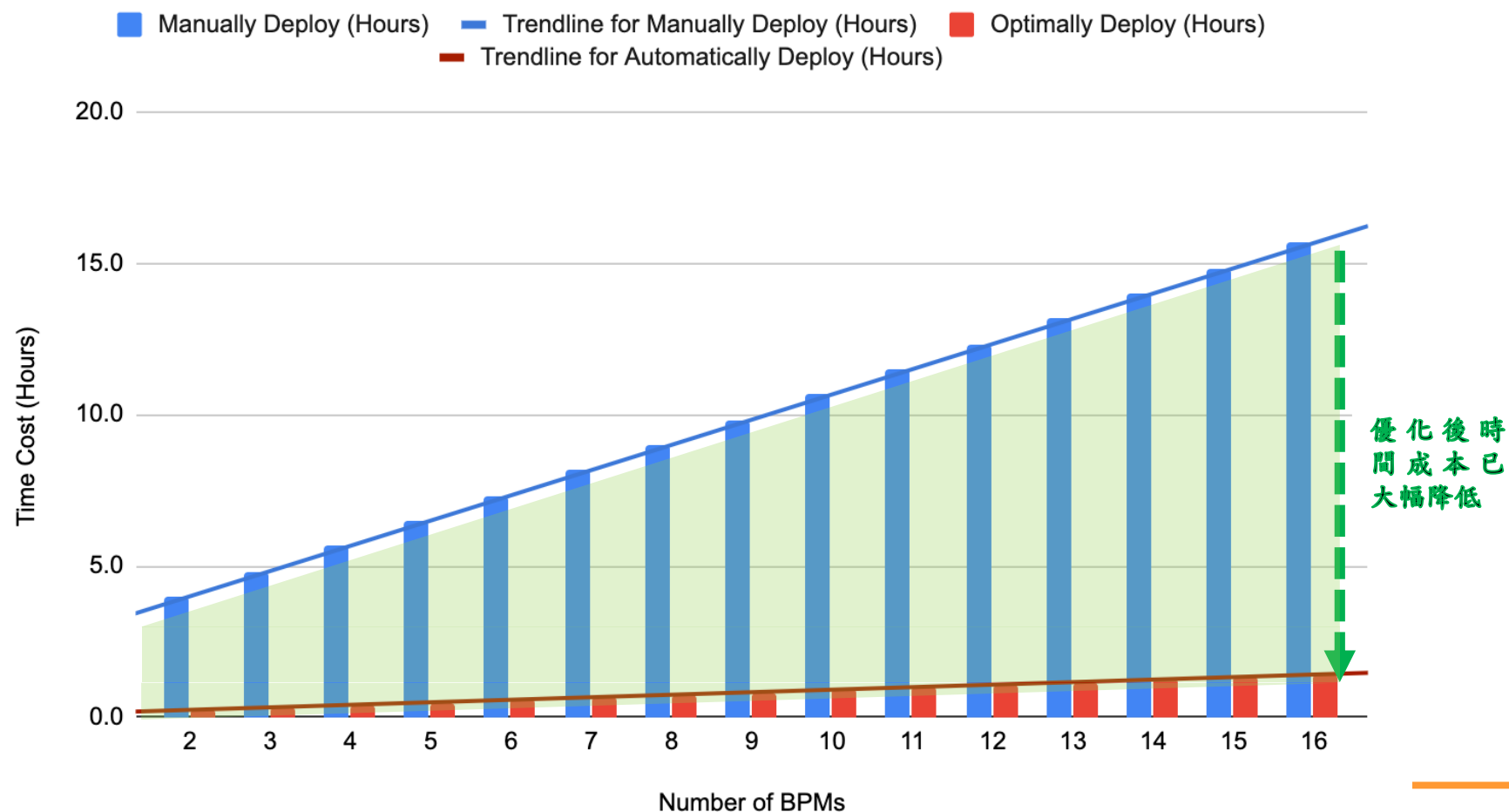
工作項目	子項目	優化部署流程後的手動佈署時間成本(分鐘)	執行次數	總計
建置環境	1. 安裝.Net Core 2. 安裝SQL Server Native Client (x64) 3. 安裝MATLAB Compiler Runtime 4. 啟動IIS	0 (已包裝成Docker Image)	0	0
部署程式	1. 下載主程式 1. Data Processing 2. Alg Service 3. GUI 4. Parser Service 5. Token Service 2. 設定參數 1. 編輯Config檔 2. 設定ODBC 3. 設定IIS 1. 建立Website 2. 選擇AppPool 3. 指定應用程式根目錄 4. 調整目錄權限	5 (僅需編輯Helm Chart設定檔Values.yaml，其餘工作已透過Powershell腳本與shell scripts自動化處理)	M+1 (1個Master與M個Slave)	5M+5
啟動程式	1. 檢查相依性服務是否已啟動 2. 啟動IIS Website或IIS App	0 (透過Kubernetes自動檢查與啟動)	0	0



部署IPM與IPM_C之部署時間成本比較

32

- 假如固定使用3台主機(M=3)，尚未容器化前，部署2~16個BPM，粗估花費時間約4~16小時(實際時間視操作人員而定)——部署的數量越多，時間成本將線性提高。
- 而優化後，手動部署IPM_C只剩下修改設定檔(Values.yaml)，其他步驟皆自動化處理，可大幅降低時間成本。



結語與展望



■ 今日分享之內容包含3部分：

- 第1部分：介紹我們團隊所研發之智慧製造平台AMCoT，已經成功應用於一些製造產業(如半導體、輪圈加工、吹瓶機、航太等)；並介紹基於容器技術之新一代智慧製造平台AMCoT_C。
- 第2部分：描述我們將容器化智慧預測保養系統(IPM_C)導入面板製造廠之成果。
- 第3部分：說明我們基於Kubernetes與Docker之IPM_C全廠導入實踐作法與效益。

■ 未來展望：

- 我們團隊所研發之智慧製造服務(AVM、IPM、IYM、IDS等)可協助生產線智慧化，而基於Kubernetes與Docker，更易於將這些智慧製造服務導入製造業。
- 我們已完成IPM_C、AVM_C、IYM_C等之開發，未來將持續擴充AMCoT_C之功能。

