智慧製造專題與實作報告

Edge computing implementation with Docker
以Docker實作邊緣運算系統

指導老師: 周哲維

109005510 簡光正

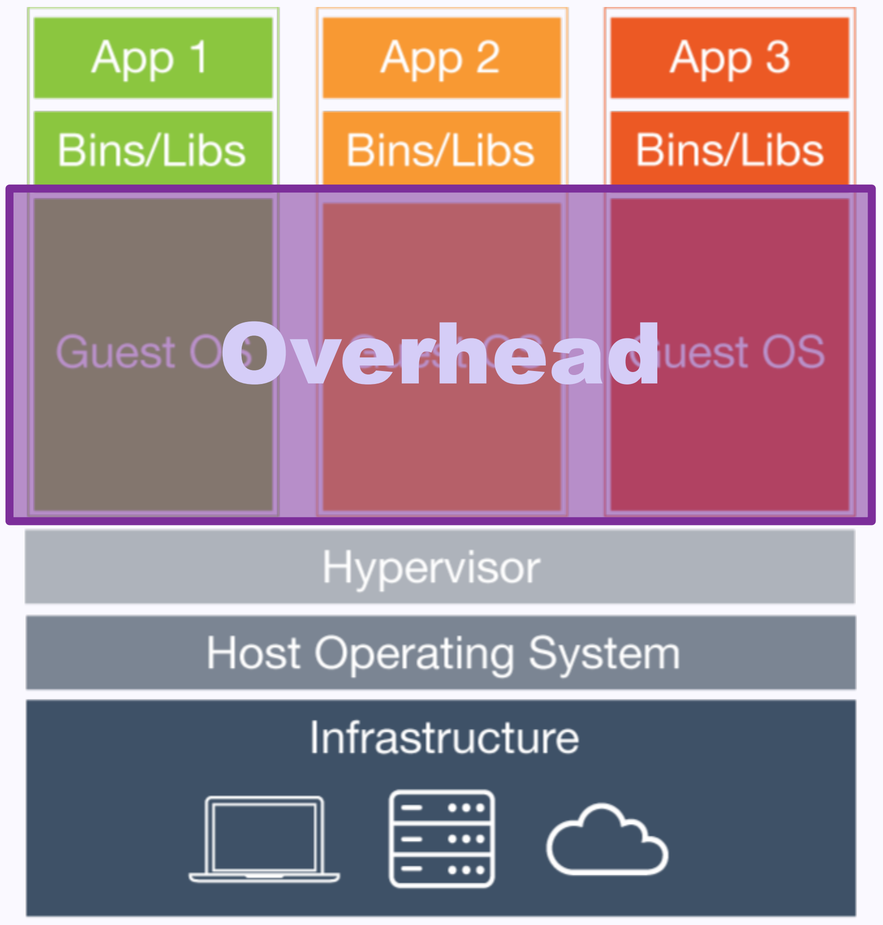
2022.05.07

大綱

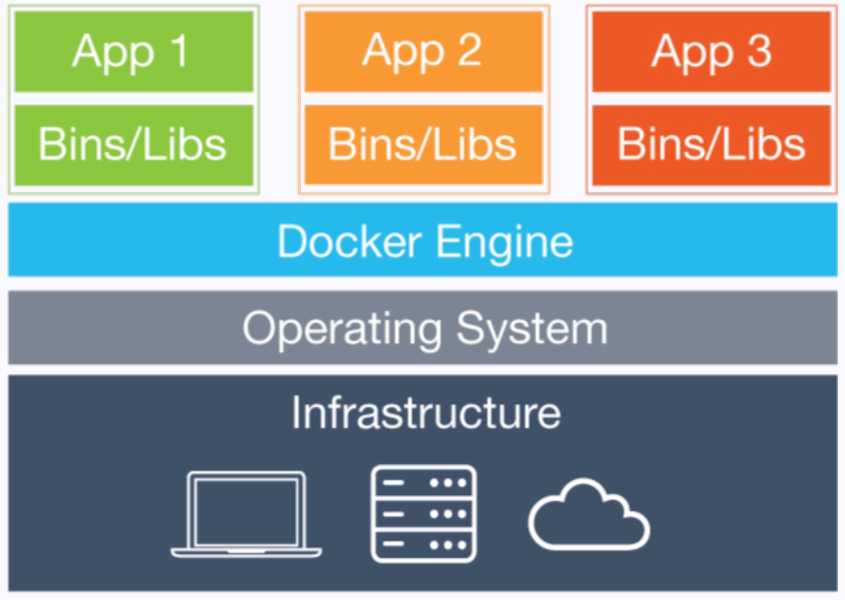
- What is Container / Docker? (5 mins)
- Why Docker? (5 mins)
- Docker Hands-On (20 mins)

Docker Explained

Docker Explained

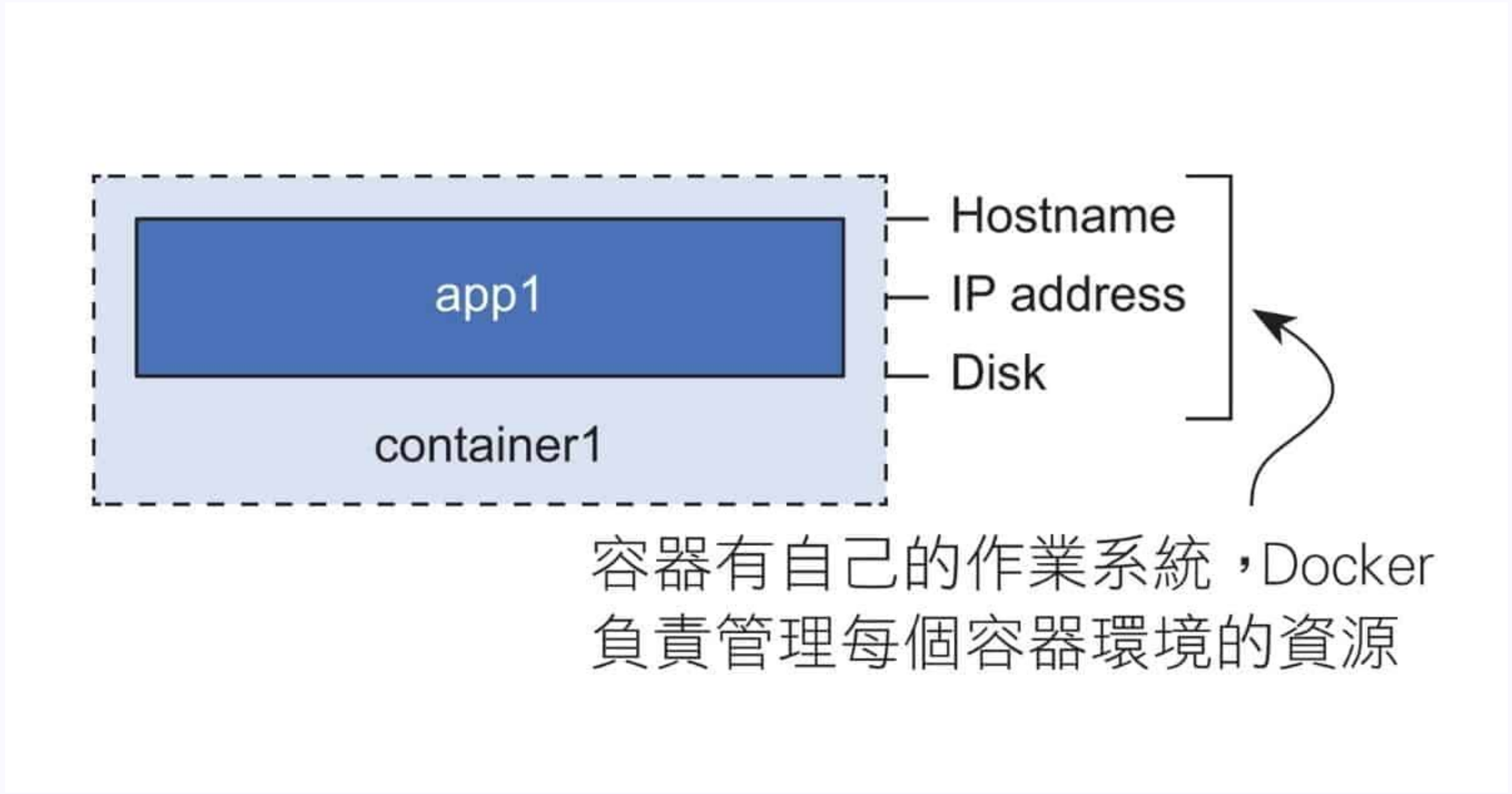


Virtual machines

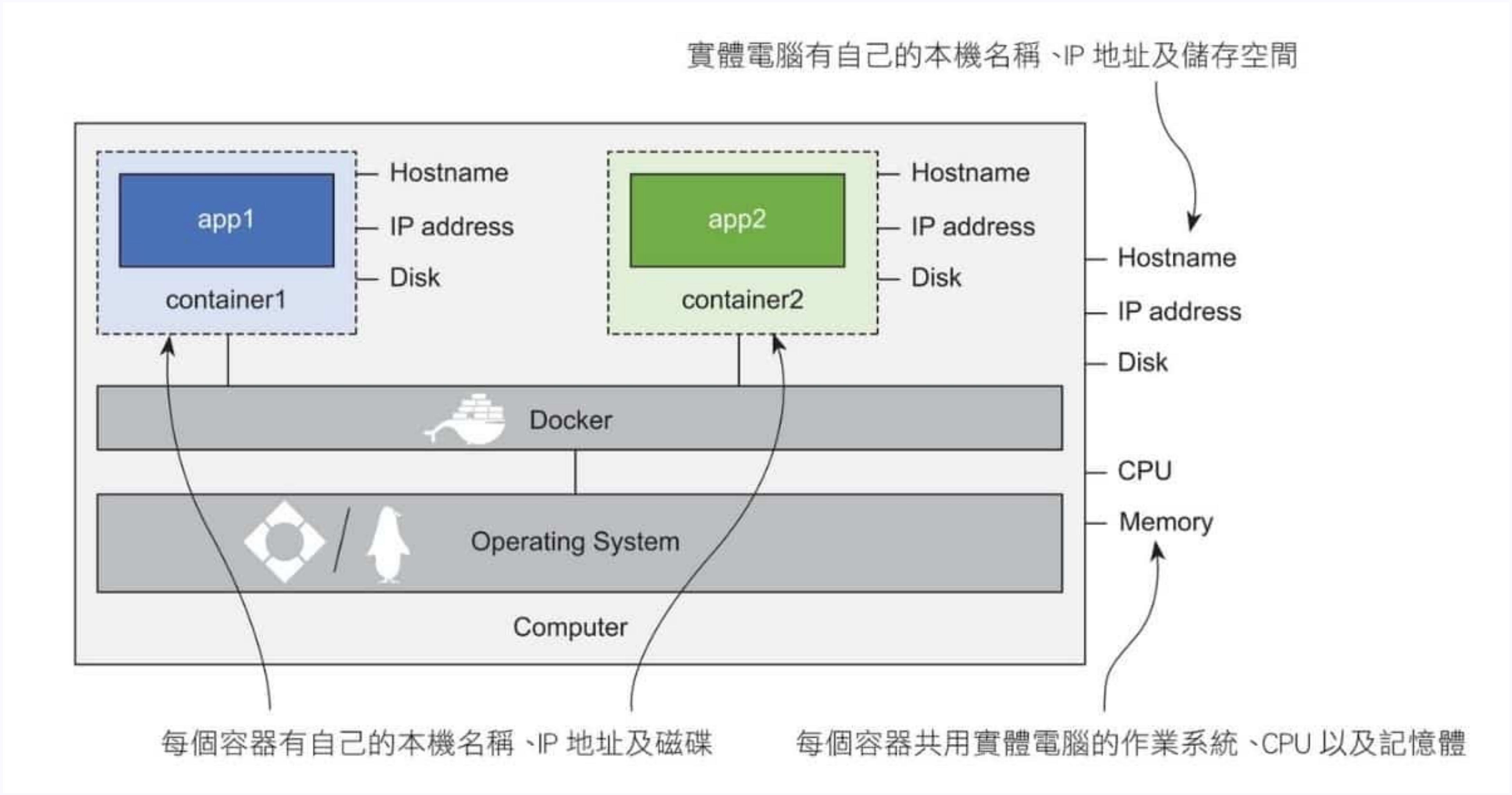


Containers

Docker Explained



Docker Explained



Docker key use cases

#1 部署應用程式到雲端服務

#2 微服務架構 Microservices

#3 原生雲端應用程式 Cloud Native

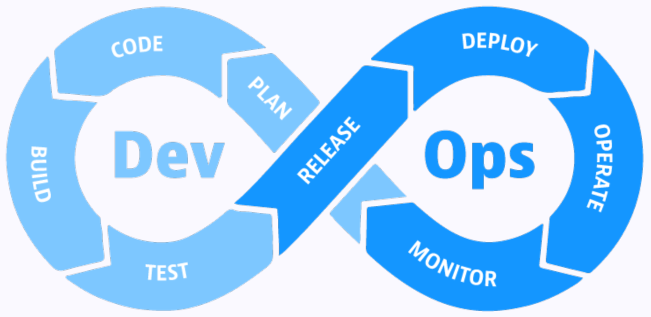
#4 Serverless架構

#5 DevOps數位轉型

Why Docker?

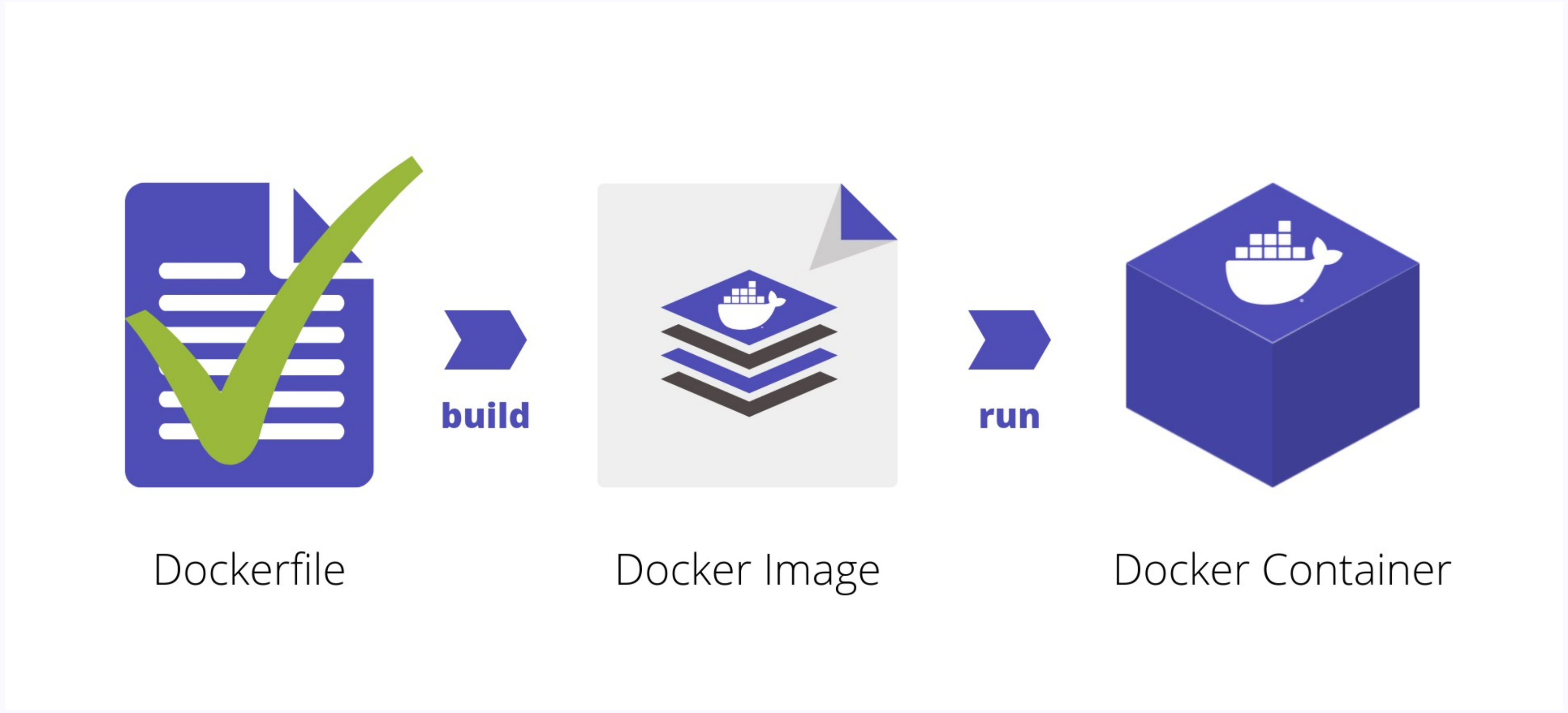
Advantages of Docker

- Portability
- Performance
- Agility
- Isolation
- Scalability
- Security



Docker Hands-On in 20mins

Docker Flow



Usual docker commands (Part I)

docker run 執行容器

```
> docker run --name webserver -p 5000:80 nginx
```

執行一個Nginx的Web伺服器，並將電腦上的 5000 port 導到容器的 80 port

docker stop 停止容器運行

```
> docker container stop webserver
```

docker start 運行已停止的容器

```
> docker container start webserver
```

docker rm 刪除容器

```
> docker rm webserver
```

Usual docker commands (Part II)

docker container ls 列出容器清單 (等同 docker ps -a)

```
> docker container ls -a
```

docker image ls 列出映像清單

```
> docker image ls
```

docker kill 強制停止容器

```
> docker container kill webserver
```

docker image rm 刪除映像

```
> docker image rm nginx
```

#1 Hello World!

Before we start

Quick Notes

https://hackmd.io/@kcchien/HyZXzXzIc

Before we start

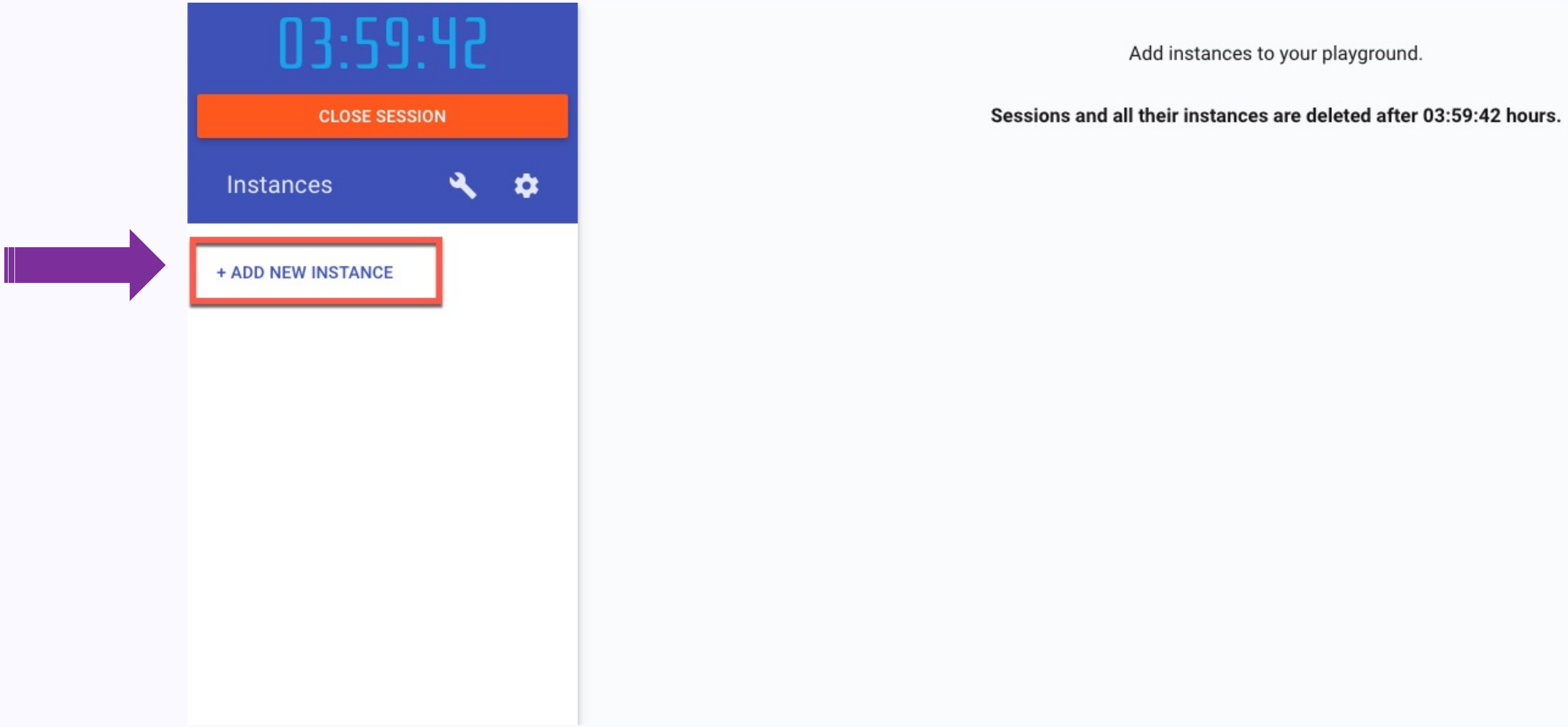
> **Play with Docker**

`https://labs.play-with-docker.com/`

aims001

aims12345

Before we start



Add instances to your playground.

Sessions and all their instances are deleted after 03:59:42 hours.

Command list

docker run 執行容器

```
> docker run hello-world
```

docker stop 停止容器運行

```
> docker container stop hello-world
```

docker start 運行已停止的容器

```
> docker container start hello-world
```

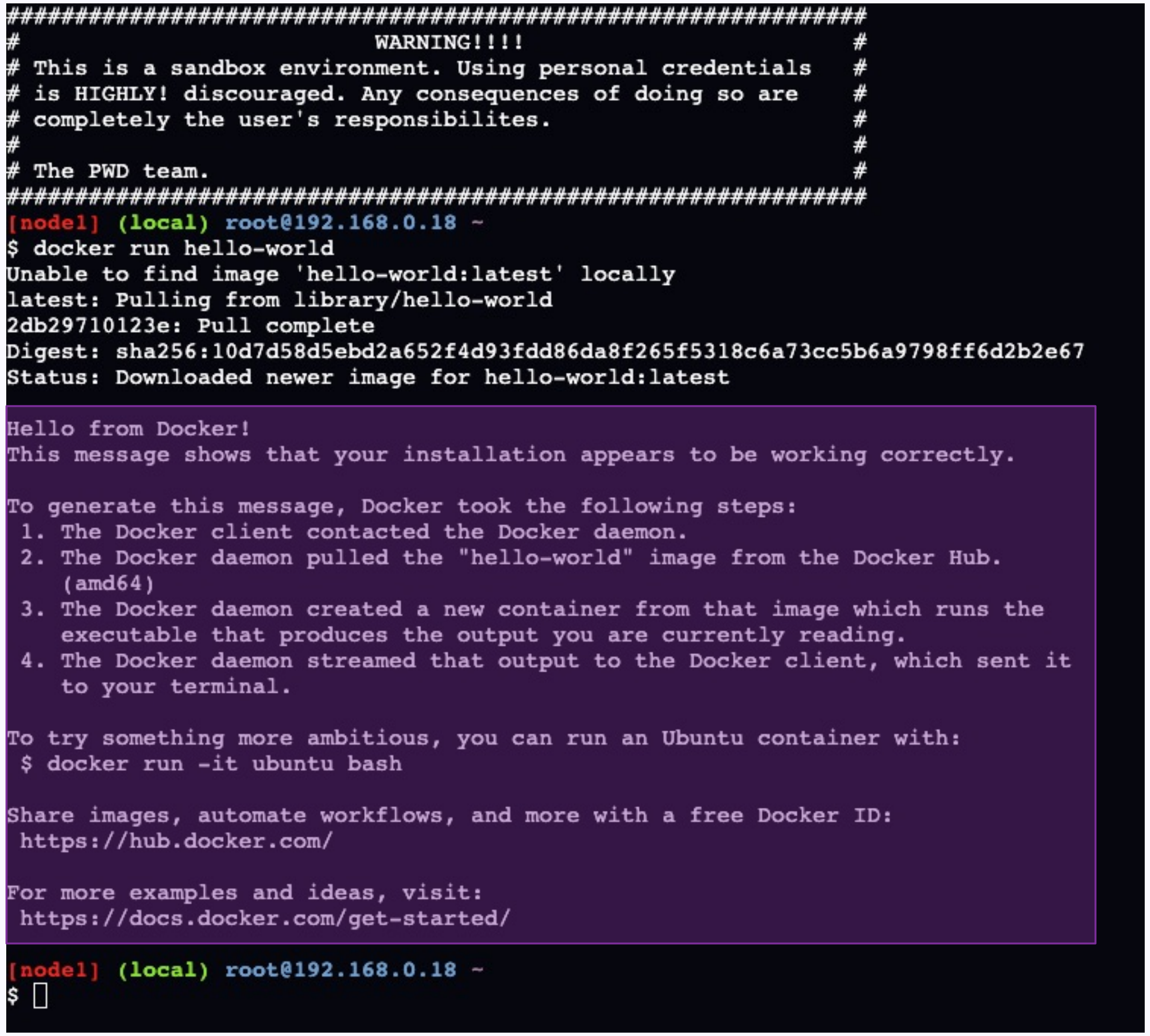
docker rm 刪除容器

```
> docker rm hello-world
```

Hello World!

> docker run hello-world

docker hub即時下載名稱為 “hello-world” 的映像檔，並立即生成一個container執行，執行內容很簡單，就是印出一篇文字訊息



```
###############################################################
#                          WARNING!!!!                        #
# This is a sandbox environment. Using personal credentials   #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                             #
# The PWD team.                                               #
###############################################################
[node1] (local) root@192.168.0.18 ~
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:10d7d58d5ebd2a652f4d93fdd86da8f265f5318c6a73cc5b6a9798ff6d2b2e67
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[node1] (local) root@192.168.0.18 ~
$ 
```

#2 Web Server

Command list

docker run 執行容器

```
> docker run --name webserver -p 5000:80 nginx
```

執行一個Nginx的Web伺服器，並將電腦上的 5000 port 導到容器的 80 port

docker stop 停止容器運行

```
> docker container stop webserver
```

docker start 運行已停止的容器

```
> docker container start webserver
```

docker rm 刪除容器

```
> docker rm webserver
```

#3 Node-RED

Deploy Node-RED

從 Node-RED 官方 docker hub 下載映像部署

```
> docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red
```

進階運用 docker-compose

#3 Node-RED
+
Grafana
+
InfluxDB

Deploy Node-RED

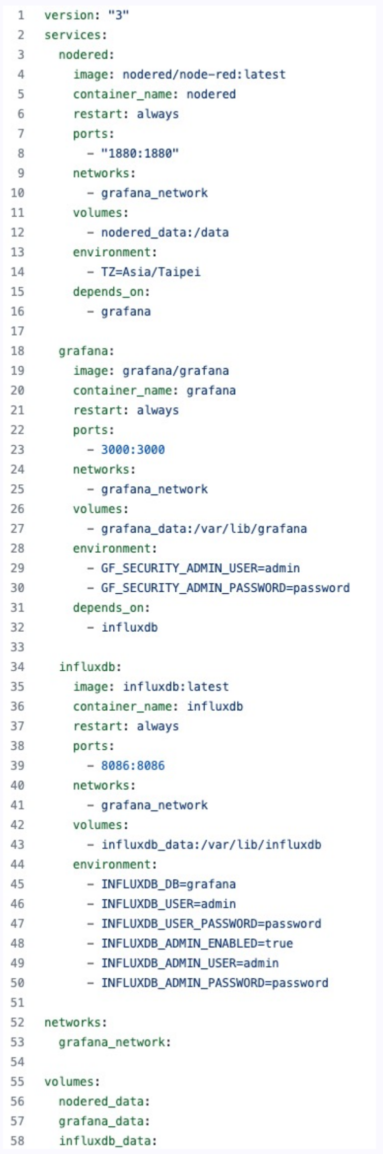
下載docker-compose.yaml

```
> wget https://github.com/kcchien/aims-docker-lab/raw/main/docker-compose.yaml
```

nodered
port 1880

grafana
port 3000

influxDB
port 8086



```yaml
version: "3"
services:
  nodered:
    image: nodered/node-red:latest
    container_name: nodered
    restart: always
    ports:
      - "1880:1880"
    networks:
      - grafana_network
    volumes:
      - nodered_data:/data
    environment:
      - TZ=Asia/Taipei
    depends_on:
      - grafana

  grafana:
    image: grafana/grafana
    container_name: grafana
    restart: always
    ports:
      - 3000:3000
    networks:
      - grafana_network
    volumes:
      - grafana_data:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=password
    depends_on:
      - influxdb

  influxdb:
    image: influxdb:latest
    container_name: influxdb
    restart: always
    ports:
      - 8086:8086
    networks:
      - grafana_network
    volumes:
      - influxdb_data:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=grafana
      - INFLUXDB_USER=admin
      - INFLUXDB_USER_PASSWORD=password
      - INFLUXDB_ADMIN_ENABLED=true
      - INFLUXDB_ADMIN_USER=admin
      - INFLUXDB_ADMIN_PASSWORD=password

networks:
  grafana_network:

volumes:
  nodered_data:
  grafana_data:
  influxdb_data:
```

Command list

啟動所有容器

```
> docker-compose up -d
```

-d 參數代表要執行在背景的方式

停止所有容器運行

```
> docker-compose stop
```

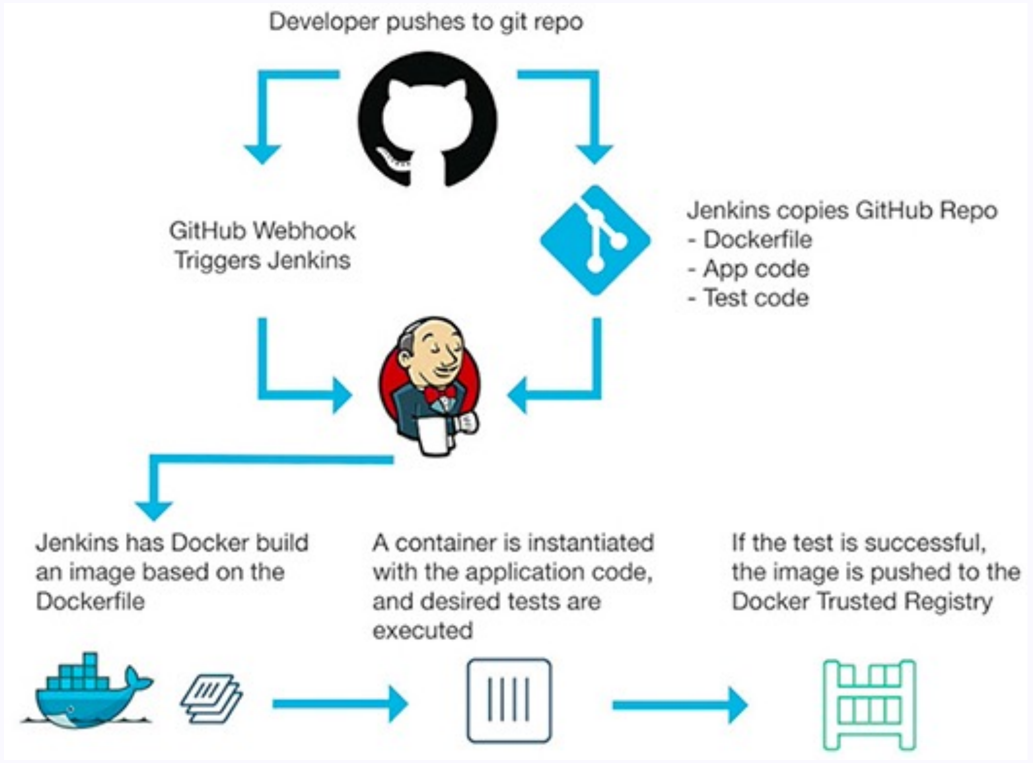
刪除所有已停止的容器

```
> docker-compose rm
```

Summary

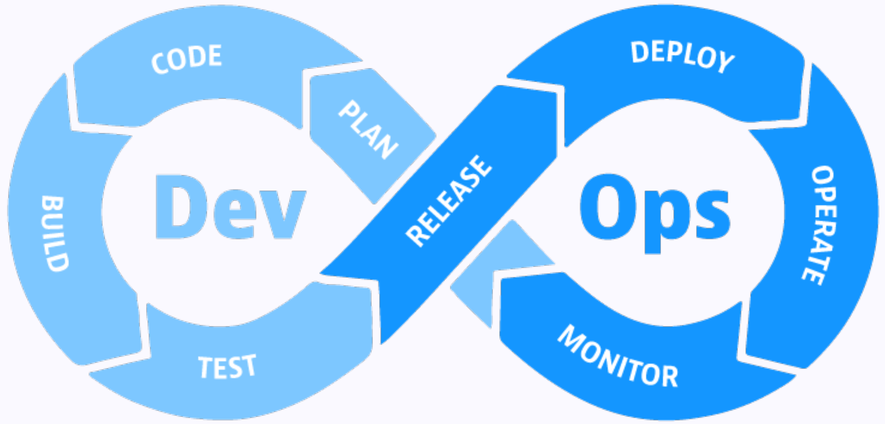
Benefit from using docker in IoT deployment

- Portability
- Lightweight
- Easy application upgrades
- Simplicity and faster configuration
- Reuse
- Security


Developer pushes to git repo
GitHub Webhook Triggers Jenkins
Jenkins copies GitHub Repo
- Dockerfile
- App code
- Test code
Jenkins has Docker build an image based on the Dockerfile
A container is instantiated with the application code, and desired tests are executed
If the test is successful, the image is pushed to the Docker Trusted Registry

Summary

透過這些現代化開發方法與技術，可以讓工廠的機聯網系統，可以讓工廠擁有更具彈性、可擴充彈性，在影響最小的情況下，進行新的功能更新與部署，也同時可以提高機聯網網路的安全性，讓工廠可以隨著科技技術的更新，同步成長。



Resources

- docker official site

  https://www.docker.com/

- docker hub

  https://hub.docker.com/

- Play with Docker
- https://labs.play-with-docker.com/
- Tensorflow with Docker
- https://www.tensorflow.org/install/docker/

Thank You