

# 資料庫

許新翎

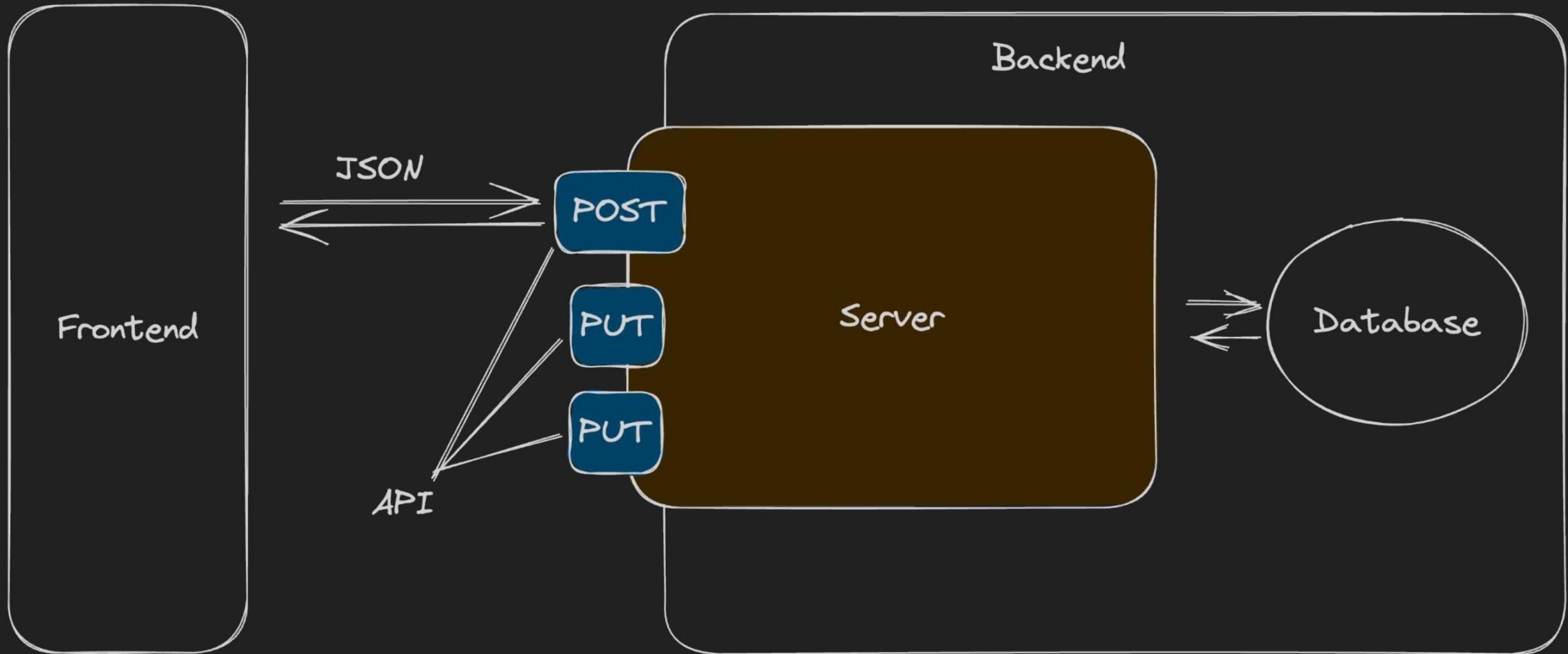
# 今日實作 CODE

- [HTTPS://GITHUB.COM/JUSTINHSU1019/GDSC-DATABASE-TUTORIAL](https://github.com/justinhsu1019/GDSC-DATABASE-TUTORIAL)

# 一段小故事

CTF? SQL注入?

什麼是資料庫？



# 資料庫的基本概念

- 什麼是資料庫，它的主要功能和用途
- 1. 資料庫就像是資訊的集中存儲地
- 2. 你可以在資料庫裡新增、查詢、修改或刪除資料
- 3. 資料庫可以多人同時使用，且避免資料重複
- 4. 資料庫與其他應用程式是分開的，這樣更靈活

在資料庫之前，電腦  
如何儲存數據？

# 資料庫的歷史背景

- 人工處理階段
  - 1. 數據不保存，不共享
  - 2. 數據需要由應用程式自己管理
  - 3. 數據不具有獨立性，變化後需修改應用程式

# 資料庫的歷史背景

- 平面文件系統

- 1. 數據可以長期保存，由文件系統進行管理
- 2. 程式與數據有一定的獨立性，但共享性差
- 3. 數據獨立性低，冗餘度較大

# 資料庫的歷史背景

- 資料庫管理系統
  - 1. 數據結構化，共享性好，冗餘度低
  - 2. 數據獨立性高，由**DBMS**統一管理
  - 3. 應用程式和數據具有高度的獨立性

什麼是資料庫管理系統？

# 資料庫管理系統 (DBMS)

- 資料庫管理系統 (DBMS) 的角色和功能
- DBMS是一套大型軟體，專門用於建立、使用及維護資料庫，確保資料的安全性和完整性
- 用戶和資料庫管理員可以透過DBMS來訪問及維護資料庫中的數據
- DBMS允許多個應用程式和用戶同時或在不同時間使用多種方法來操作資料庫
- DBMS使用者能夠方便地定義和操作數據，並在多用戶環境下進行數據併發控制和恢複

# 幾個常見的DBMS

- 1. MySQL：是一個開源的關聯式資料庫管理系統，常用於網頁應用



# 幾個常見的DBMS

- 2. MICROSOFT SQL SERVER：是MICROSOFT公司開發的關聯式資料庫管理系統



# 幾個常見的DBMS

- 3. ORACLE DATABASE：是一個物件-關聯式資料庫管理系統，廣泛用於大型企業



# 幾個常見的DBMS

- 4. POSTGRESQL：是一個強大的開源物件-關聯式資料庫系統



# 幾個常見的DBMS

- 5. MONGODB：是一個開源的NOSQL資料庫，使用JSON樣式的文件進行資料存儲



# SQL 語言深入

## 搭配 MySQL 實作進行

# MySQL 安裝

# 今日主題

# 學生資料管理系統

# 資料庫結構

schoolDB (資料庫/Database)

  └ subjects (資料表/Table)

- └ id (欄位/Column) - INT (自動遞增/AUTO\_INCREMENT), 主鍵/PRIMARY KEY
- └ subject\_name (欄位/Column) - VARCHAR(50) (不可以是空的/NOT NULL)

  └ students (資料表/Table)

- └ id (欄位/Column) - INT (自動遞增/AUTO\_INCREMENT), 主鍵/PRIMARY KEY
- └ first\_name (欄位/Column) - VARCHAR(50)
- └ last\_name (欄位/Column) - VARCHAR(50)
- └ subject\_id (欄位/Column) - INT (外來鍵/FOREIGN KEY, 參照 subjects 的 id)
- └ grade (欄位/Column) - DECIMAL(4,2)
- └ email (欄位/Column) - VARCHAR(100)

# CREATE 和 ALTER 語句

- 1. CREATE DATABASE 語句用於建立新的資料庫
- 2. CREATE 語句用於在資料庫中建立新的表格
- 3. ALTER 語句用於修改已存在的表格結構，例如添加、刪除或修改欄位

# CREATE 和 ALTER 語句

-- 建立新的資料庫

```
CREATE DATABASE schoolDB;
```

-- 切換到該資料庫

```
USE schoolDB;
```

-- 建立科目資料表

```
CREATE TABLE subjects (
    id INT AUTO_INCREMENT PRIMARY KEY,
    subject_name VARCHAR(50) NOT NULL
);
```

# CREATE 和 ALTER 語句

-- 建立學生資料表

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    subject_id INT,
    grade DECIMAL(4,2),
    FOREIGN KEY (subject_id) REFERENCES subjects(id)
);
```

-- 修改學生資料表

```
ALTER TABLE students ADD COLUMN email VARCHAR(100);
```

# 資料操作語句

- 1. **INSERT** 語句用於向表格中插入新的資料。
- 2. **UPDATE** 語句用於修改表格中的資料。

# 資料操作語句

-- Section 2: Basic Query Statements

-- 插入科目資料

```
INSERT INTO subjects (subject_name) VALUES ('Mathematics'), ('History'), ('Physics');
```

-- 插入學生資料

```
INSERT INTO students (first_name, last_name, subject_id, grade, email)
VALUES ('John', 'Doe', 1, 88.5, 'johndoe@gmail.com'),
       ('Jane', 'Smith', 2, 90.0, 'janeshmith@gmail.com'),
       ('Robert', 'Johnson', 3, 85.5, 'robertjohnson@gmail.com');
```

-- 更新學生成績

```
UPDATE students SET grade = 89.5 WHERE id = 3;
```

# 基本查詢語句

## 搭配 MySQL 實作進行

# SELECT 語句的基本結構

- 1. SELECT 語句用於從資料庫表格中選擇資料
- 2. 可以指定要選擇的欄位名稱，或使用 \* 來選擇所有欄位

# SELECT 語句的基本結構

-- 基本查詢語句

```
SELECT * FROM students;
```

```
SELECT first_name, last_name FROM students;
```

# WHERE 子句

- 如何過濾資料
- 1. WHERE 子句用於過濾查詢結果
- 2. 可以使用比較運算符，如 =、<>、>、< 等來指定過濾條件

# WHERE 子句

- 如何過濾資料

-- 使用 WHERE 子句來過濾資料

```
SELECT first_name, last_name FROM students WHERE grade > 85;  
SELECT first_name, last_name FROM students WHERE subject_id = 2;
```

# 排序和分組

- ORDER BY 和 GROUP BY 的使用
  - 1. ORDER BY 子句用於對查詢結果進行排序。
  - 2. GROUP BY 子句用於對查詢結果進行分組。

# 排序和分組

- ORDER BY 的使用

假設你有以下學生成績表：

diff

Save to grepper    Copy code

name		score
-----		-----
小明		85
小華		90
小王		78
小李		92

# 排序和分組

- ORDER BY 的使用

你想按照分數從高到低排序，可以這麼做：

sql



Save to grepper



Copy code

```
SELECT name, score  
FROM students  
ORDER BY score DESC;
```

# 排序和分組

- ORDER BY 的使用

結果會是：

```
diff
name | score
-----|-----
小李 | 92
小華 | 90
小明 | 85
小王 | 78
```

# 排序和分組

- GROUP BY 的使用

假設你有以下的果汁銷售紀錄：

diff



Save to grepper



Copy code

fruit		quantity
蘋果		5
香蕉		3
蘋果		7
橙子		4
香蕉		2

# 排序和分組

- GROUP BY 的使用

你想知道每種果汁賣了多少瓶，可以這麼做：

sql



Save to grepper



Copy code

```
SELECT fruit, SUM(quantity) as total_quantity  
FROM juice_sales  
GROUP BY fruit;
```

# 排序和分組

- GROUP BY 的使用

結果會是：

diff



Save to grepper



Copy code

fruit	total_quantity
蘋果	12
香蕉	5
橙子	4

# 排序和分組

- ORDER BY 和 GROUP BY 的使用

-- 資料的排序和分組

```
SELECT subject_id, COUNT(*) as num_students FROM students GROUP BY subject_id;  
SELECT first_name, last_name FROM students ORDER BY grade DESC;
```

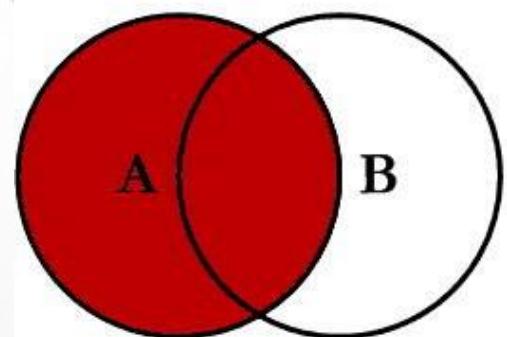
# 進階查詢技巧

## 搭配 MySQL 實作進行

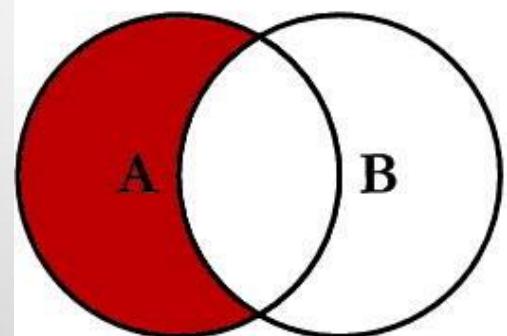
# 聯接 (JOIN) 的概念

- 如何結合多個表格
- 1. JOIN 用於結合兩個或多個表格的資料。
- 2. 可以使用 INNER JOIN、LEFT JOIN、RIGHT JOIN 和 FULL JOIN 等不同類型的聯接。

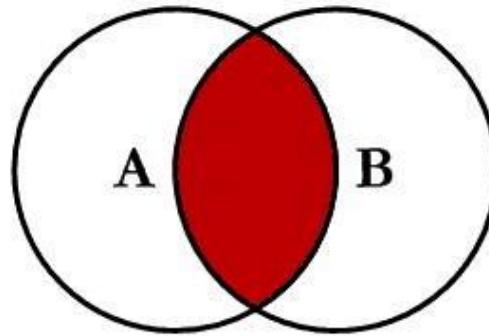
# SQL JOINS



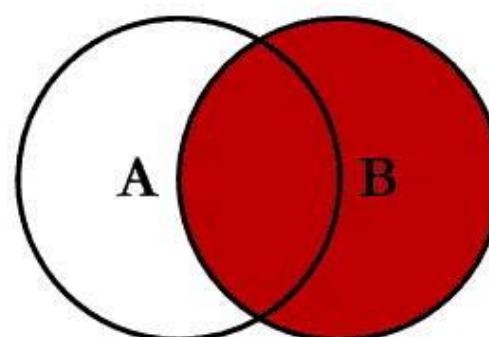
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



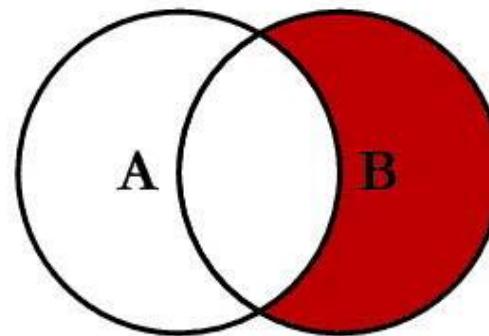
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



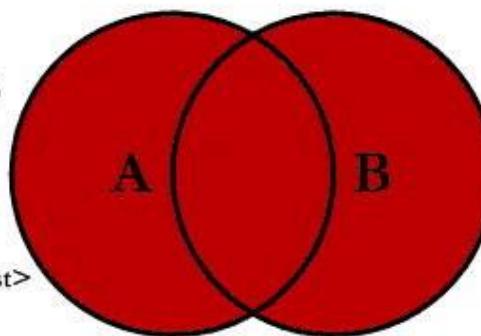
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



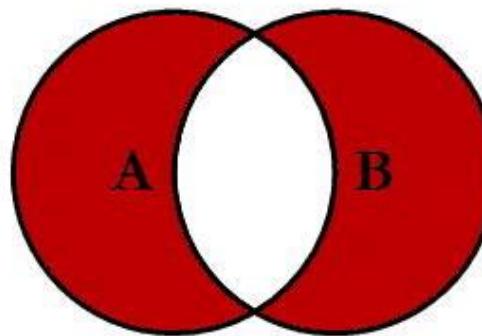
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

# 聯接 (JOIN) 的概念

- 如何結合多個表格

-- JOINs

-- 透過學生和科目資料表的JOIN，查詢學生的名字和他們所修的科目

```
SELECT s.first_name, s.last_name, sub.subject_name  
FROM students s  
INNER JOIN subjects sub ON s.subject_id = sub.id;
```

# 子查詢的使用

- 1. 子查詢是嵌套在另一個查詢中的查詢。
- 2. 可以在 SELECT、FROM 和 WHERE 子句中使用子查詢。

# 子查詢的使用

```
-- 子查詢  
SELECT  
    s.first_name,  
    s.last_name,  
    sub.subject_name  
FROM students s  
INNER JOIN subjects sub ON s.subject_id = sub.id  
WHERE s.grade > (  
    SELECT AVG(grade) FROM students  
);
```

# 視圖 (VIEW) 的概念和用途

- 1. 視圖是一個虛擬的表格，它可以是基於一個或多個表格的查詢結果
- 2. 視圖可以用於簡化複雜的查詢，隱藏資料的某些部分或提供額外的資料安全性

# 視圖 (VIEW) 的概念和用途

## 什麼是 View (視圖)？

想像一下，你有一本厚厚的食譜書，但有一些食譜你特別常用。每次找這些食譜時，都要翻那麼厚的書很麻煩。所以，你決定把那些常用食譜拍照，然後放在一個資料夾裡，這樣每次想找時就很方便。

在 SQL 中的 `VIEW` 就像是那個資料夾，它儲存了一個查詢的結果，這樣每次你需要那個結果時，只要查詢這個 `VIEW` 就可以，不需要再輸入整個查詢語句。

但要記住，`VIEW` 並不真的儲存資料，它只是保存查詢語句。所以，當基礎的資料有變化時，`VIEW` 會自動更新。

# 視圖 (VIEW) 的概念和用途

-- 建立View

```
CREATE VIEW view_above_avg_students AS
SELECT
    s.first_name,
    s.last_name,
    sub.subject_name
FROM students s
INNER JOIN subjects sub ON s.subject_id = sub.id
WHERE s.grade > (
    SELECT AVG(grade) FROM students
);
```

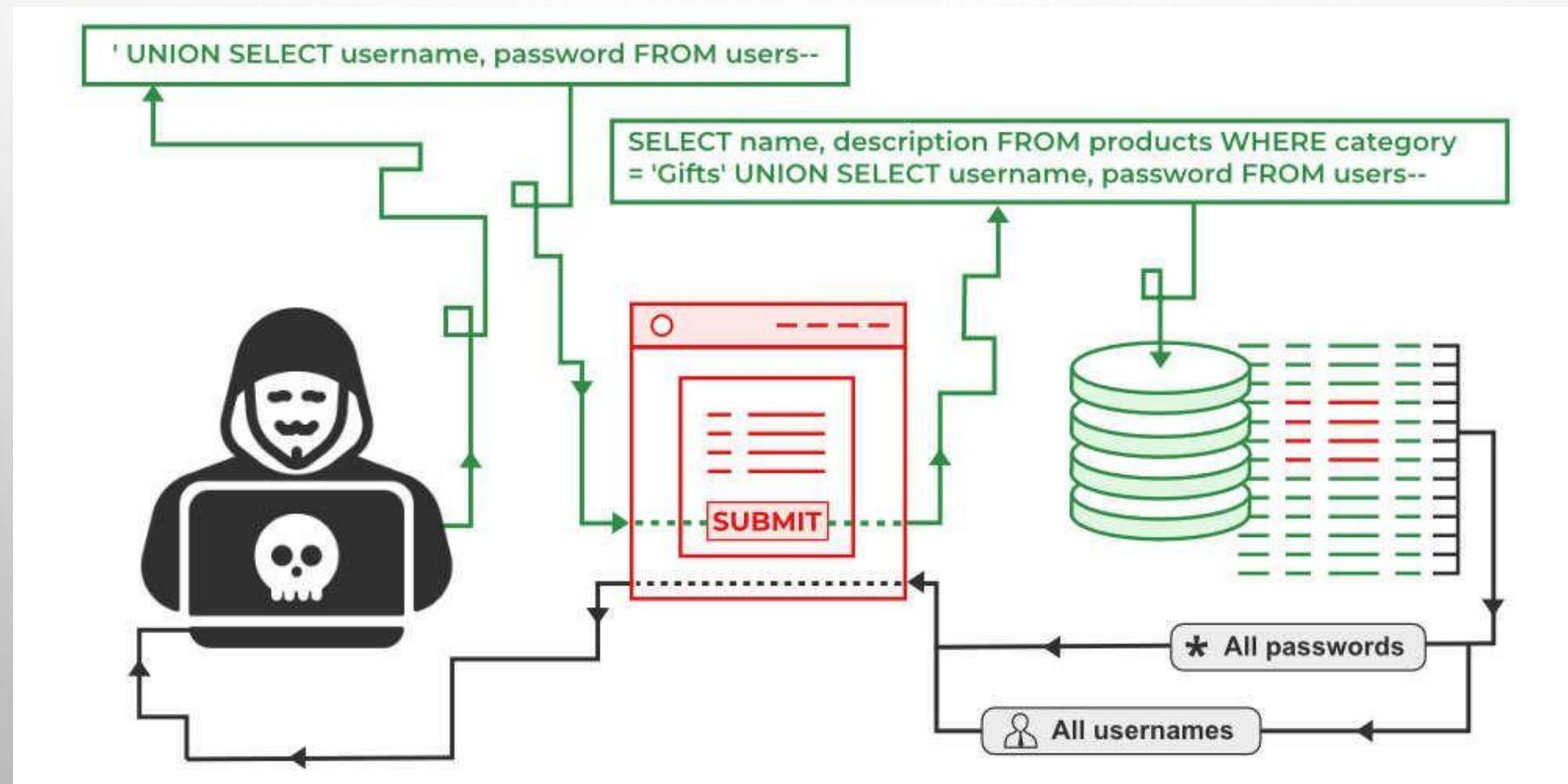
-- 查看View

```
SELECT * FROM view_above_avg_students;
```

# 資料庫的安全威脅

# 資料庫的安全威脅

- SQL注入：攻擊者利用應用程式的安全漏洞，向資料庫查詢中注入惡意SQL代碼，從而獲取、修改或刪除資料



# 資料庫的安全威脅

- 權限管理：不正確的權限設定可能允許未經授權的用戶訪問、修改或刪除資料

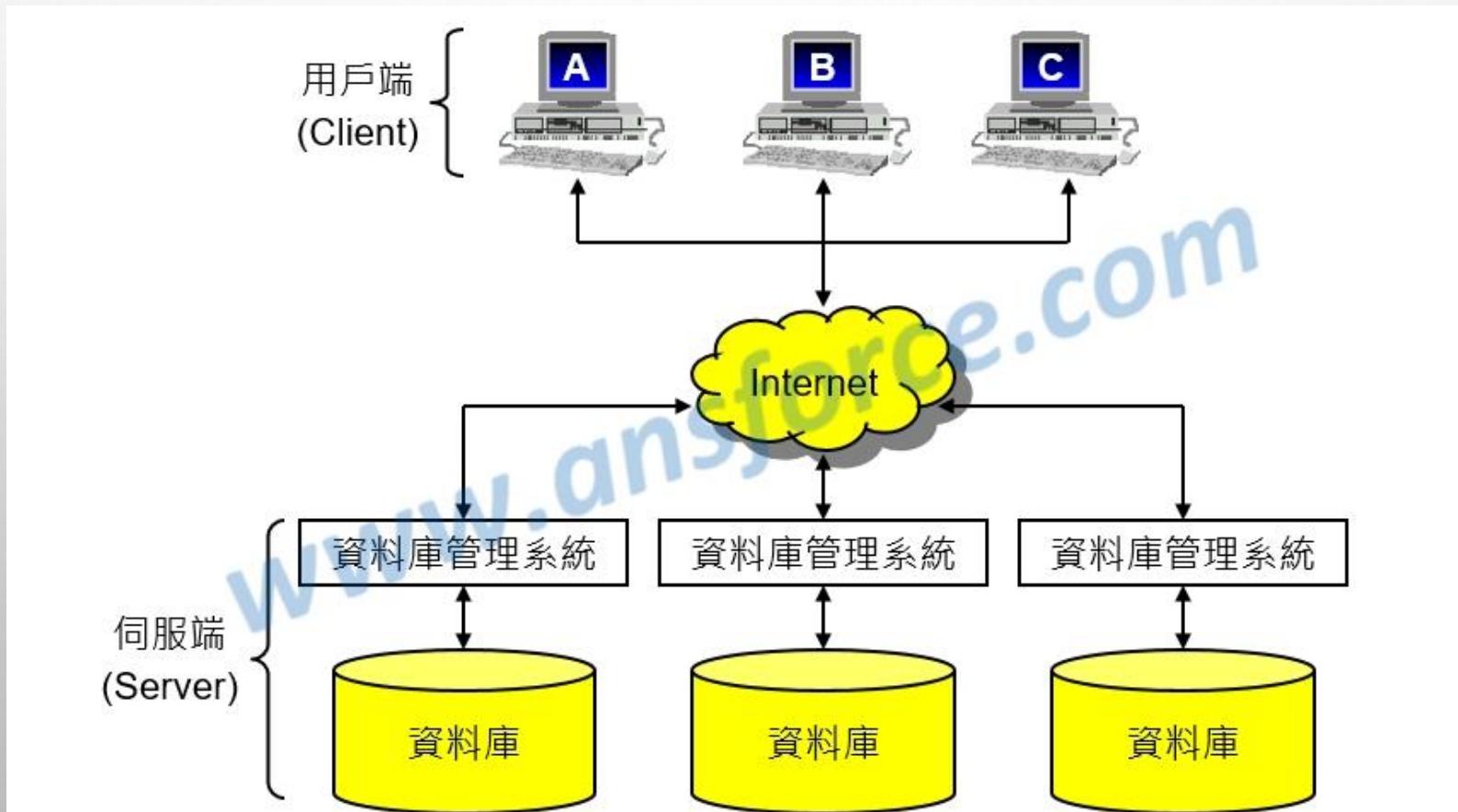


還有哪些資料庫？

# 分散式資料庫

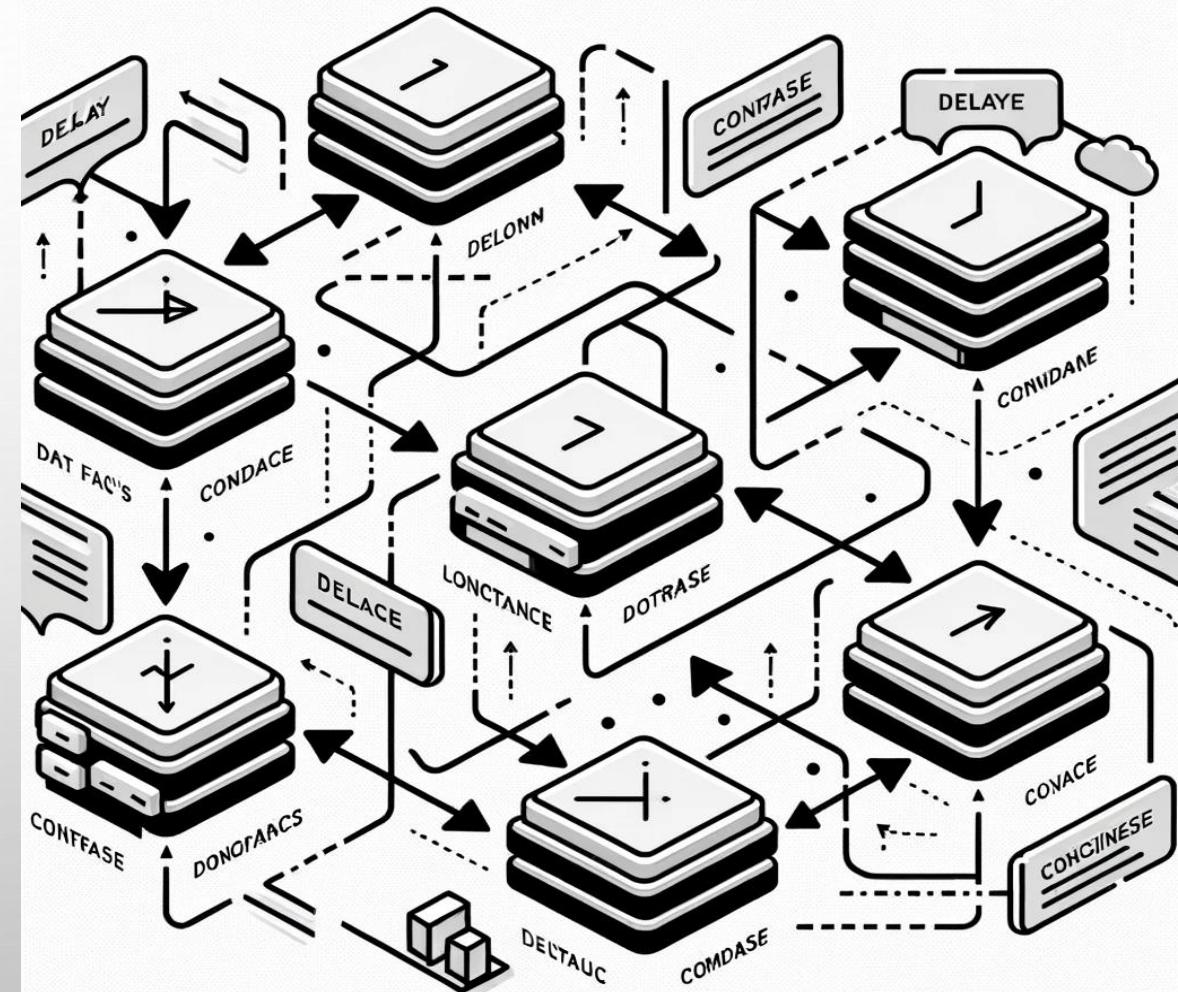
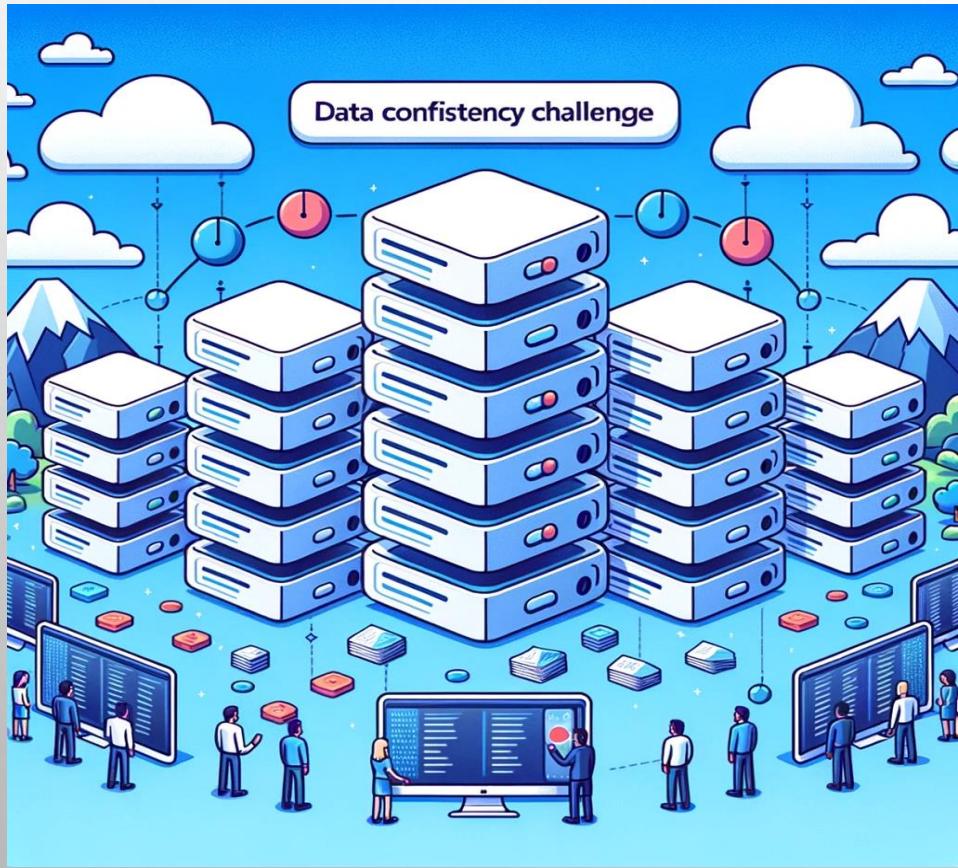
# 分散式資料庫的概念和優勢

- 1. 分散式資料庫是將資料存儲在多台機器上，而不是單一機器上
- 2. 主要優勢包括高可用性、擴展性和容錯性



# 分散式查詢的挑戰

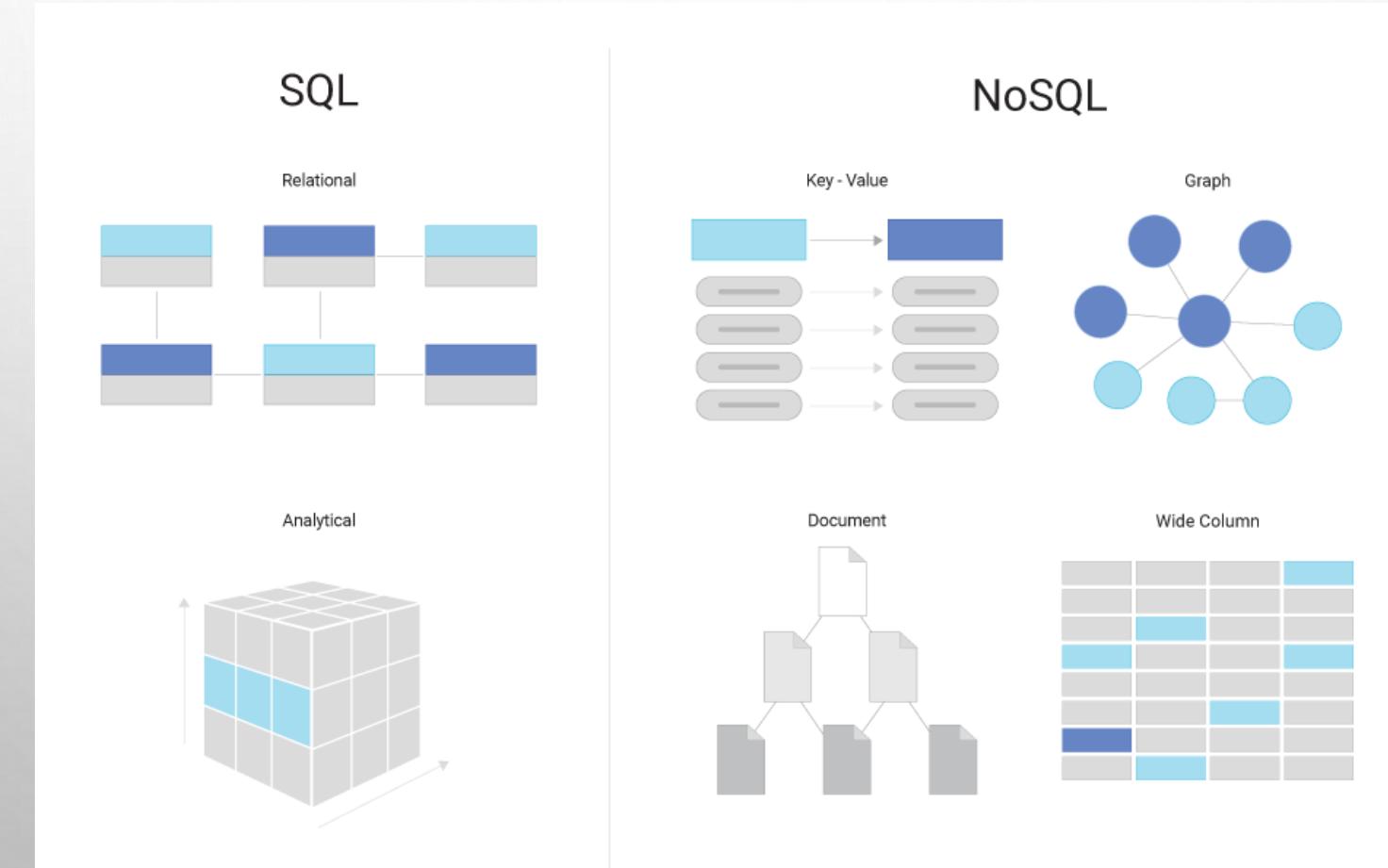
- 1. 分散式查詢需要從多台機器上獲取資料，這可能會增加查詢的複雜性和延遲
- 2. 需要考慮資料的一致性和完整性問題



# NOSQL 資料庫介紹

# NOSQL 與關聯式資料庫的區別

- 1. NOSQL 資料庫通常不使用固定的表格結構，而是使用更靈活的資料模型。
- 2. NOSQL 資料庫設計用於滿足大規模資料和高性能的需求。



# 常見的 NOSQL 資料庫

## MONGODB、REDIS

- 1. MONGODB 是一個文檔導向的資料庫，適用於存儲大量的非結構化資料
- 2. REDIS 是一個內存中的鍵-值存儲系統，常用於快速緩存

```
2
3  ## MongoDB 插入操作
4  `db.collection.insert({name: "John", age: 30})`
5
6  ## Redis 設置鍵值
7  `SET user:1 "John"`
8
```

# 向量資料庫介紹

# AI 浪潮

# 向量資料庫的基本概念

- 1. 向量資料庫是一種特殊的資料庫，專為高效處理向量數據而設計
- 2. 它可以快速進行向量間的相似性比較和搜索
- 3. 範例：使用向量資料庫進行圖像或文本的相似性搜索
- 4. 範例：計算兩個向量之間的餘弦相似性

# 常見的向量資料庫工具

- 1. WEAVIATE、QDRANT 和 CHROMA 是一些流行的向量資料庫工具。
- 2. 這些工具提供了高效的語意搜索，混合搜索等功能。



# 向量資料庫的適用場景

- 1. 當需要進行大量的向量相似性比較和搜索時
- 2. 適用於機器學習、圖像識別和推薦系統等應用
- 3. 範例：基於用戶的購物歷史，使用向量資料庫推薦相似的商品

# 演示 - 向量資料庫

- 安裝 WEAVIATE

# Weaviate建置

設定Weaviate:

## 下載Docker

- 透過Docker啟用 Weaviate

## 啟動Weaviate

- 到達指定路徑後在 powershell端輸入 docker compose up

成功啟動

# Weaviate建置

設定Weaviate:

## 下載Docker

- 透過Docker啟用 Weaviate

## 啟動Weaviate

- 到達指定路徑後在 powershell端輸入 docker compse up

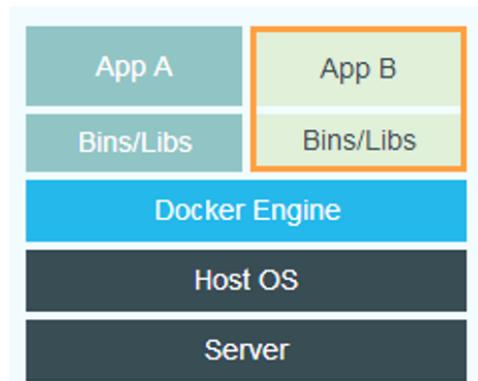
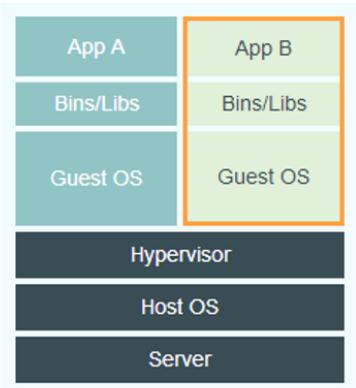
成功啟動

# Docker簡介

Docker 是一個開源平台，最初是 dotCloud 公司內部於 2013 年的專案。Docker 主要用於在容器中構建、部署和運行應用程序。它藉由容器化技術，使應用程序及其相關的應用能夠在隔離的環境中運行，且可以在不同的電腦設備環境中保持一致性和可移植性。

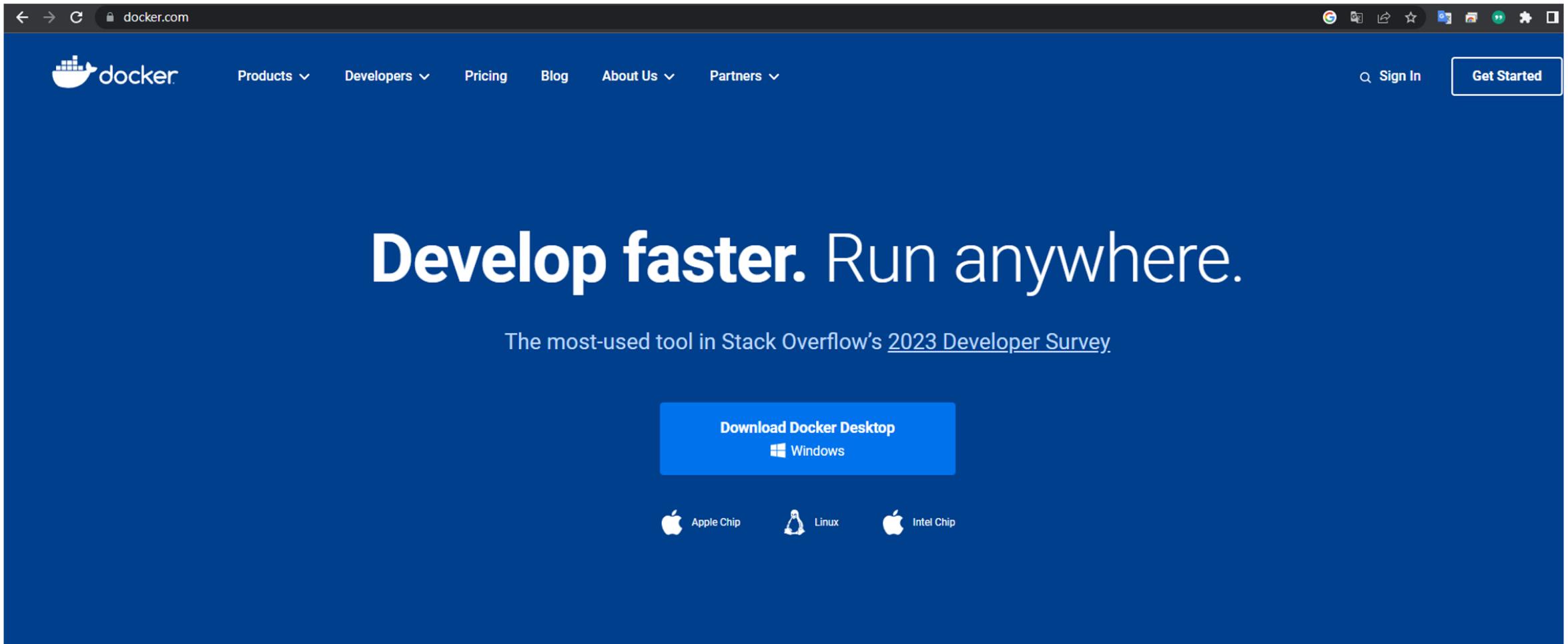


類似於Virtual Machines 但其泛用性及兼容性更好



# 安裝 docker

安裝 docker:



The screenshot shows the official Docker website at docker.com. The page has a dark blue header with the Docker logo and navigation links for Products, Developers, Pricing, Blog, About Us, and Partners. On the right side of the header are search, sign-in, and 'Get Started' buttons. The main content area features a large white text 'Develop faster. Run anywhere.' over a blue background. Below it, a subtext states 'The most-used tool in Stack Overflow's [2023 Developer Survey](#)'. A prominent blue button in the center says 'Download Docker Desktop' with a 'Windows' option below it. At the bottom, there are icons for Apple Chip, Linux, and Intel Chip.

Products ▾ Developers ▾ Pricing Blog About Us ▾ Partners ▾

Get Started

# Develop faster. Run anywhere.

The most-used tool in Stack Overflow's [2023 Developer Survey](#).

Download Docker Desktop

Windows

Apple Chip Linux Intel Chip

# Weaviate建置

設定Weaviate:

## 下載Docker

- 透過Docker啟用 Weaviate

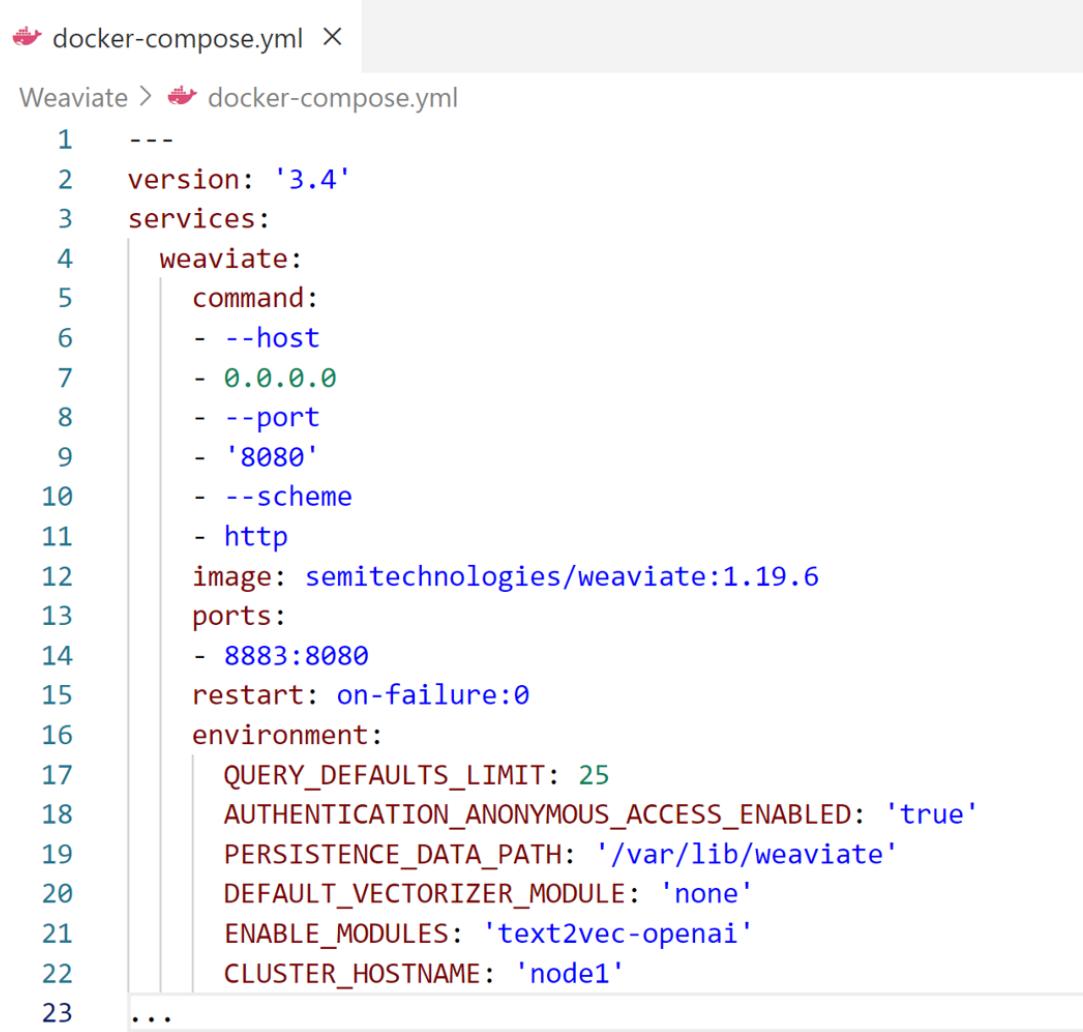
## 啟動Weaviate

- 到達指定路徑後在 powershell端輸入 docker compose up

成功啟動

# Weaviate建置

開啟docker-compose.yml:

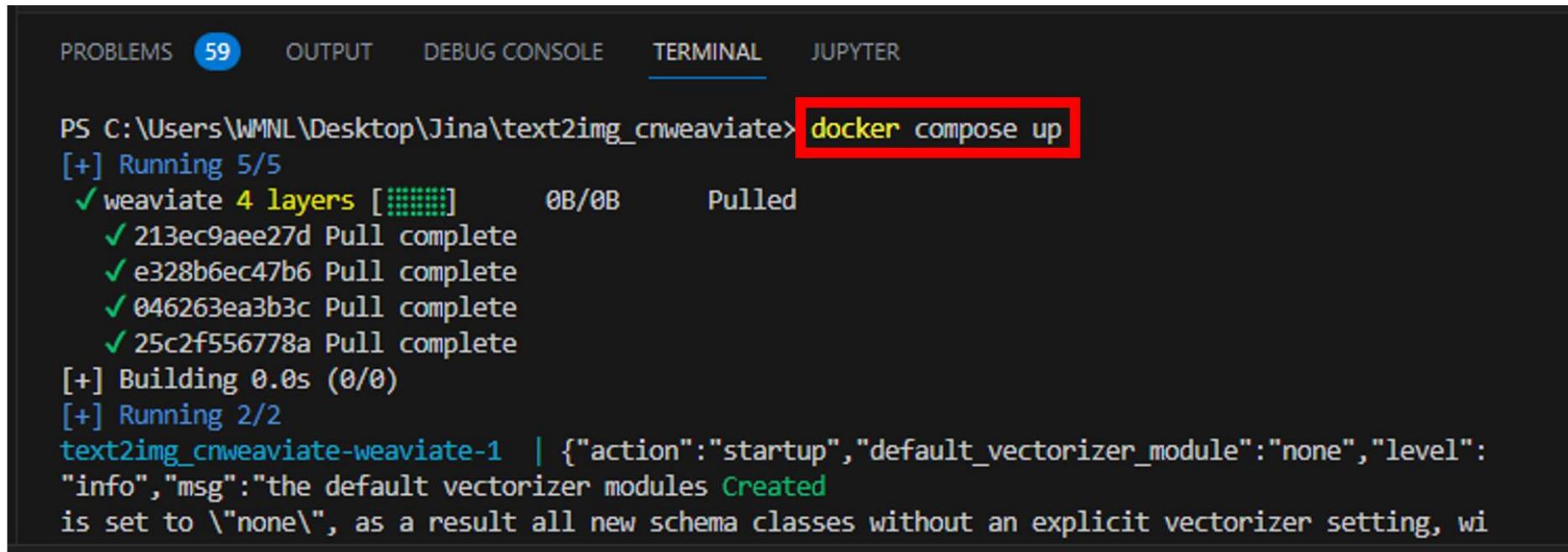


The screenshot shows a code editor window with a tab labeled "docker-compose.yml". The file contains YAML configuration for a Weaviate service. The configuration includes setting the version to 3.4, defining a service named "weaviate" with specific command-line arguments for host, port, and scheme, specifying the image to use, mapping ports, setting restart policies, and defining environment variables.

```
1  ---
2  version: '3.4'
3  services:
4    weaviate:
5      command:
6        - --host
7        - 0.0.0.0
8        - --port
9        - '8080'
10       - --scheme
11       - http
12       image: semitechnologies/weaviate:1.19.6
13       ports:
14         - 8883:8080
15       restart: on-failure:0
16       environment:
17         QUERY_DEFAULTS_LIMIT: 25
18         AUTHENTICATION_ANONYMOUS_ACCESS_ENABLED: 'true'
19         PERSISTENCE_DATA_PATH: '/var/lib/weaviate'
20         DEFAULT_VECTORIZER_MODULE: 'none'
21         ENABLE_MODULES: 'text2vec-openai'
22         CLUSTER_HOSTNAME: 'node1'
23         ...
```

# Weaviate建置

抵達指定路徑後在終端機輸入docker compose up



The screenshot shows a terminal window with the following output:

```
PS C:\Users\WMNL\Desktop\Jina\text2img_cnweaviate> docker compose up
[+] Running 5/5
  ✓ weaviate 4 layers [██████]    0B/0B      Pulled
    ✓ 213ec9aee27d Pull complete
    ✓ e328b6ec47b6 Pull complete
    ✓ 046263ea3b3c Pull complete
    ✓ 25c2f556778a Pull complete
[+] Building 0.0s (0/0)
[+] Running 2/2
text2img_cnweaviate-weaviate-1 | {"action":"startup","default_vectorizer_module":"none","level":"info","msg":"the default vectorizer modules Created
is set to \"none\", as a result all new schema classes without an explicit vectorizer setting, wi
```

The command `docker compose up` is highlighted with a red box.

# Weaviate建置

設定Weaviate:

## 下載Docker

- 透過Docker啟用 Weaviate

## 啟動Weaviate

- 到達指定路徑後在 powershell端輸入 docker compose up

成功啟動

# Weaviate建置

確認Weaviate有成功啟動

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. The sidebar on the left includes links for 'Containers', 'Images', 'Volumes', 'Dev Environments (BETA)', and 'Learning Center'. An 'Extensions' section with a 'Add Extensions' button is also present. The main area displays container statistics and a list of running containers.

**Container CPU usage:** 1.57% / 400% (4 cores allocated)

**Container memory usage:** 40.75MB / 5.95GB

**Containers List:**

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
text2img		Exited	0%		12 days ago	[Actions]
hm-fashion-in		Exited	0%		2 days ago	[Actions]
text2img_cnw		Running (1/1)	1.57%		2 days ago	[Actions]

Showing 3 items

RAM 2.81 GB Not connected to Hub v4.20.1

# 演示 - 向量資料庫

- 使用 WEAVIATE 做TOP1 向量查詢

# CONTACT ME

INSTAGRAM: JUSTIN.HSU.99  
JUSTIN.HSU.1019@GMAIL.COM