

Elasticsearch簡介與實戰應用

[What is Elasticsearch \[1\].?](#)



可以點進官網查看註解

- 一個開源的搜尋引擎，建築在 Lucene 上，Lucene 的缺點如下：
 - 使用起來較複雜，需要一些背景知識
 - 需要會使用 Java
- Elasticsearch 透過封裝的概念 → 變成大家都易於使用的搜尋引擎
- Lucene is very complex, 要懂 Java

- Search engine

特色：

- [Open Source \[2\]](#)
- Distributed
- Save & Search
 - 能夠提供儲存與檢索資料
- base on Apache Lucene
- RESTful API
- 今年已推出 8 版 → 但社課用 7 版，因為 8 版設定上較為複雜
- [DB-Engines Ranking \[3\]](#) ⇒ 唯一在前十名榜上的搜尋引擎

410 systems in ranking, March 2023									
Rank			DBMS	Database Model	Score				
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022		
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1261.29	+13.77	+9.97		
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1182.79	-12.66	-15.45		
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	922.01	-7.08	-11.77		
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	613.83	-2.67	-3.10		
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	458.78	+6.02	-26.88		
6.	6.	6.	Redis +	Key-value, Multi-model ⓘ	172.45	-1.39	-4.31		
7.	7.	7.	IBM Db2	Relational, Multi-model ⓘ	142.92	-0.04	-19.22		
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	139.07	+0.47	-20.88		
9.	9.	↑ 10.	SQLite +	Relational	133.82	+1.15	+1.64		
10.	10.	↓ 9.	Microsoft Access	Relational	132.06	+1.03	-3.37		

- 跟 Mysql、Database 的差異

SQL	NoSQL	Elasticsearch
table	collection	index
row	document	document
column	field	field
index	index	Inverted index

- 可以將數據用 json 儲存，而且不用事先定義格式
 - 不像 MySQL 需要先設好 Schema



ElasticSearch 沒有支援 ACID，所以狹義上來說不能算是資料庫（DBMS）

*ACID，是指資料庫管理系統（DBMS）在寫入或更新資料的過程中，為保證交易（transaction）是正確可靠的，所必須具備的四個特性。

Atomicity（原子性）

- 有些指令不可以交錯執行，需要一口氣執行完
- 例如當你要去銀行提款：
 - **ATM send command**
 - **Database balance -= 3000**
 - **ATM spit money**
 - **ATM return transaction success**

Consistency（一致性）

- [中租租車紅利金超用\[4\]](#)
 - 演算法中的比例：學長讓現金對比打折的比例改動
 - 他們的資料庫沒有一致性：若紅利金為負，應該要恢復到不合理的交易前的狀態（中租在資料庫的設定上有不合邏輯的地方）
- 欄位型態不符合，需要重用一遍

Time	A	B
1	book seat C87	
2		book seat C87
3		pay for seat C87
4	pay for seat C87	
5	order Success	
6		order Success

Isolation (隔離性)

- [高鐵售票系統設計出錯 超賣八百餘個座位 | 大紀元 \[5\]](#)
 - 有八個人跟你訂同一個位置
 - 很多人在同一時間訂位，該位置就同時賣出給多人

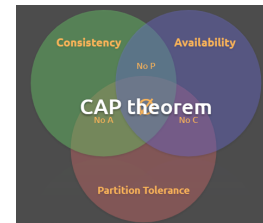
Durability (持續性)

- Elasticsearch 是有機率遺失資料的，各個 Index 有狀態燈號可以參考如下：
 - Green：主分片及副分片都已經有分配好。
 - Yellow：有部分的分片遺失，這可能會影響效能或甚至是遺漏資料（算 warning）。
 - Red：至少有一個主分片遺失，這代表可能會遺漏不少數據，需要及時處理。

CAP theorem

為分散式系統的定理：

- 三個特性無法同時滿足，只能選兩個
 - **Consistency 一致性**
各節點間資料必需同步
 - **Availability 可用性**
讓使用者一定可以得到節點回應的資料
 - **Partition Tolerance 分區容錯性**
在網路環境出問題時，也能夠維持系統的運作



選擇方式

1. 選擇 C 和 A

➡ 沒有分區容錯性，表示所有節點必須達成完美的資料同步，不能出現資料分區，這是不切實際的，若退回單一資料節點可以解決這個問題，但這就不能夠稱為分散式系統了。

- 因此：Partition Tolerance 一定要選！

2. 選擇 C 和 P

➡ 在網路出問題時，也可以維持資料的一致性，將使用戶有收到 Error 的可能。

e.g. A 收到資料更新後，因網路出現問題，無法將資料同步更新給 B 節點，使用者若此時對 B 發出 Request，B 節點若回覆舊資料則違反了 C，所以只能回覆 Error（喪失了可用性）。

3. 選擇 A 和 P

➡ 在網路出問題時，將導致各節點資料不同步。

e.g. 同上例，使用者若向 B 節點發出 Request，就會拿到未更新的資料（喪失了一致性）。

- 有可能一個節點是新資料，使用者拿到另一個節點的舊資料
- e.g. 低卡剛留言馬上去上一頁，再回下一頁，不會馬上看到新留言
 - 捨棄一致性，換來可用性



e.g. 不可能找到同時很快、很便宜又很好的工程師，只能三選二

ElasticSearch 小補充

- 可以像 MySQL 一樣存資料，也可以當作搜尋引擎
- 非常快，但很吃 RAM
- e.g. 去搜尋一千七百萬篇判決書中的一篇，可以在 100 毫秒內完成
 - 可以使用 Invert Table 反查表做到（在上傳資料時，Elasticsearch 就自動幫你做完了，完全不需要額外的人工成本）
 - MySQL：B+ Tree, HashTable（底層），需要調教效能則需要額外去下 Index 設定。

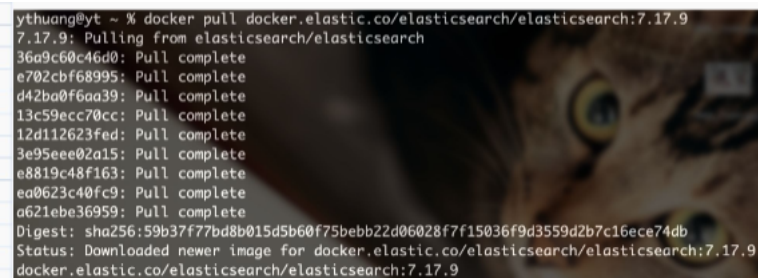
Install Docker

- [Install Docker\[6\]](#)
 - on Windows
 - <https://reurl.cc/0EA4RI>
 - on MacOS
 - <https://reurl.cc/Q4r5gZ>
 - on Linux
 - <https://reurl.cc/pLI9Rb>
- Virtual Machine v.s. Docker Container
 - [docker documentation \[7\]](#)

	virtual machine	docker container
virtualization	operating system	application
boot time	in minutes	in seconds
space	GBs	KBs, MBs
resource usage	More	Less
copy environment	slow	fast
services content	Projects	Microservices

- 先 pull 下來

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:7.17.9
```



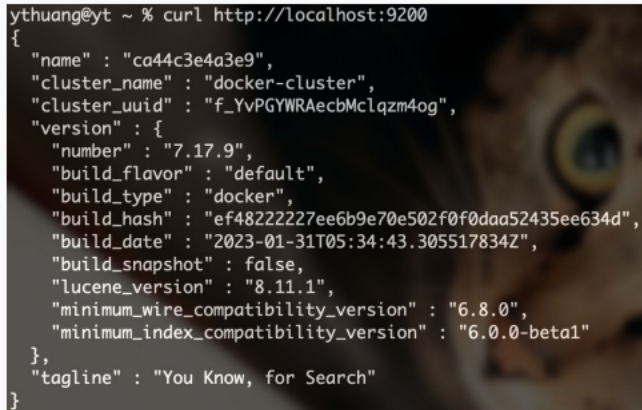
```
ythuang@yt ~ % docker pull docker.elastic.co/elasticsearch/elasticsearch:7.17.9
7.17.9: Pulling from elasticsearch/elasticsearch
36a9c60c46d0: Pull complete
e702cbf68995: Pull complete
d42ba0f6aa39: Pull complete
13c59ecc70cc: Pull complete
12d112623fed: Pull complete
3e95eee02a15: Pull complete
e8819c48f163: Pull complete
ea0623c40fc9: Pull complete
a621ebe36959: Pull complete
Digest: sha256:59b37f77bd8b015d5b60f75bebb22d06028f7f15036f9d3559d2b7c16ece74db
Status: Downloaded newer image for docker.elastic.co/elasticsearch/elasticsearch:7.17.9
docker.elastic.co/elasticsearch/elasticsearch:7.17.9
```

- 再跑起來

```
docker run -d -p 127.0.0.1:9200:9200 -p 127.0.0.1:9300:9300 -e"discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:7.
```

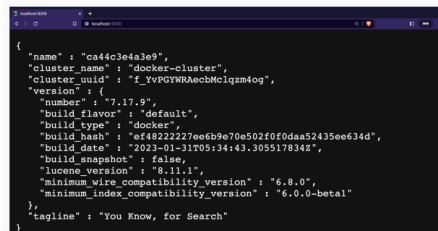
- 在終端機中（CMD/Terminal）輸入此指令，看到結果畫面才算成功

```
curl http://localhost:9200
```



```
ythuang@yt ~ % curl http://localhost:9200
{
  "name" : "ca44c3e4a3e9",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "f_YvPGYWRaEcbMclqzm4og",
  "version" : {
    "number" : "7.17.9",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "ef4822227ee6b9e70e502f0f0daa52435ee634d",
    "build_date" : "2023-01-31T05:34:43.305517834Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

網站：<http://localhost:9200>



```
{
  "name" : "ca44c3e4a3e9",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "f_YvPGYWRaEcbMclqzm4og",
  "version" : {
    "number" : "7.17.9",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "ef4822227ee6b9e70e502f0f0daa52435ee634d",
    "build_date" : "2023-01-31T05:34:43.305517834Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Access data by RESTful API

- 有兩種版本：Node.js 和 Python，其他程式語言也可以透過 HTTP Request 的方式存取資料
 - 分別有 GitHub 網址，看要用哪一種並抓下來

JavaScript

- <https://gist.github.com/s1031432/5ae8f47d81ebb0b59d393514ae95fccb>

```
const createData = (indexName, data) => {
  return new Promise((resolve, reject) => {
    axios({
      method: "POST",
      url: `http://localhost:9200/${indexName}/_doc`,
      data: data
    })
    .then(function (response) {
      resolve(response.data);
    })
    .catch(function (error) {
      reject(error);
    });
  });
};
```

- <https://gist.github.com/s1031432/58e7e48d77132079ed7871f8dab7c6b4>

```
const readData = (indexName, id="/_search") => {
  return new Promise((resolve, reject) => {
    axios({
      method: "GET",
      url: `http://localhost:9200/${indexName}/_doc/${id}`
    })
    .then(function (response) {
      resolve(response);
    })
    .catch(function (error) {
      reject(error);
    });
  });
}
```

- <https://gist.github.com/s1031432/abdc719eaa449b2df7691b3e77341de8>

```
const searchData = (indexName, query) => {
  return new Promise((resolve, reject) => {
    axios({
      method: "POST",
      url: `http://localhost:9200/${indexName}/_doc/_search`,
      data: { "query": { "match" : query } }
    })
    .then(function (response) {
      resolve(response.data);
    })
    .catch(function (error) {
      reject(error);
    });
  });
}
```

- <https://gist.github.com/s1031432/18620b608713d8aef490ebce85815087>

```
const updateData = (indexName, id, newData) => {
  return new Promise((resolve, reject) => {
    axios({
      method: "POST",
      url: `http://localhost:9200/${indexName}/_doc/${id}`,
      data: newData
    })
    .then(function (response) {
      resolve(response.data);
    })
    .catch(function (error) {
      reject(error);
    });
  });
}
```

- <https://gist.github.com/s1031432/ac852b599bea69abf7f33f04d54f81d2>

```
const deleteIndex = (indexName) => {
  return new Promise((resolve, reject) => {
    axios({
      method: "DELETE",
      url: `http://localhost:9200/${indexName}`
    })
    .then(function (response) {
      resolve(response.data);
    })
    .catch(function (error) {
      reject(error);
    });
  });
}
```

Python

- create data

- <https://gist.github.com/s1031432/8b8042811380f8de9b6ef3e50a524d70>

```
def createData(indexName, data):
    headers = {"content-type": "application/json"}
    url = "http://localhost:9200/{}/_doc".format(indexName)
    result = requests.post(url, headers = headers, data=json.dumps(data))
    return result
```

- 可以定義 index name , 要傳什麼 data
- 在 `localhost:9200/gdsctest/_doc/_search?pretty` 可以看到結果
 - gdsctest : index name
 - `/_doc/_search?pretty`, `?pretty` 也可不加
 - pretty 可幫你排版, 若未加 pretty, 呈現結果會較醜 (都放在同一行)
- index 的角色相當於關聯式資料庫中的 table
- id 由 elasticsearch 自動產生 (也可以自己指定, 但會影響效能)
 - 從 `_search` 開始汰換成 id, 並按下 enter, 就會變成該 id 的資料

範例：新增一筆id=2, data=9000的資料

```
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "gdsctest",
        "_type" : "_doc",
        "_id" : "twqiBYg8n_DoG8mHk9XY",
        "_score" : 1.0,
        "_source" : {
          "id" : 2,
          "data" : 9000
        }
      }
    ]
  }
}
```

- read data

- <https://gist.github.com/s1031432/6ad75ed0ce9ccbe68535c57858268dc3>

```
def readData(indexName, id="/_search"):
    url = "http://localhost:9200/{}/_doc{}".format(indexName, id)
    result = requests.get(url)
    return result
to join thi
```

- search data

- 會回傳找到符合的資料, 以及花費多久
- <https://gist.github.com/s1031432/0970acf5863b8319dbf29f072288bf62>

```
def searchData(indexName, query):
    headers = {"content-type": "application/json"}
```

```
url = "http://localhost:9200/{}/{}/_doc/_search".format(indexName)
result = requests.post(url, headers = headers, data=json.dumps({ "query": { "match" : query } }))
return result
```

- update data

- <https://gist.github.com/s1031432/86fd635bdd7360d7d87d5d7a51d53f6a>

```
def updateData(indexName, id, newData):
    headers = {"content-type": "application/json"}
    url = "http://localhost:9200/{}/{}/_doc".format(indexName, id)
    result = requests.post(url, headers = headers, data = json.dumps(newData))
    return result
```

- delete index

- <https://gist.github.com/s1031432/1696d316499cd5b55fcb8b176a5e9e7>

```
def deleteIndex(indexName):
    url = "http://localhost:9200/{}/".format(indexName)
    result = requests.delete(url)
    return result
```

- 新增 → 讀取 → 查詢 → 更新 → 刪除

- 形成一個 cycle：最後資料刪除後會消失，所以不能五個 function 同時 call，不會看到結果

Sample code of (JavaScript) & (Python)

- Javascript：

- <https://gist.github.com/s1031432/33743cbfa2ef140642dd775da1e753c0>

```
npm i axios
node main.js
```

- Python：

- <https://gist.github.com/s1031432/7c964df9a44b1bf8f95b21578f967188>

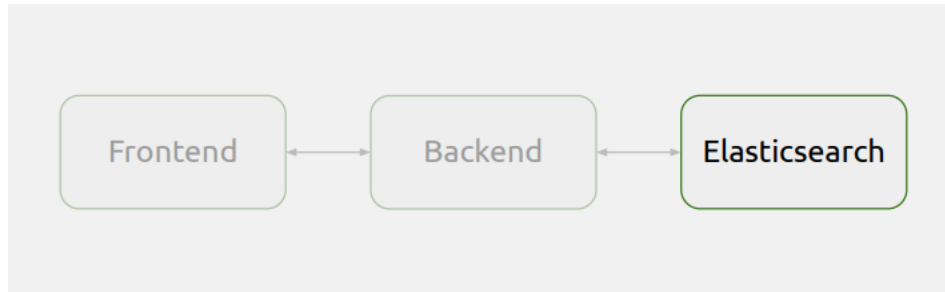
```
pip install requests
python main.py
```

- Pull sample data

```
docker pull ty80517/gdsc_elasticsearch_sample_data:latest
docker run --network="host" ty80517/gdsc_elasticsearch_sample_data
```

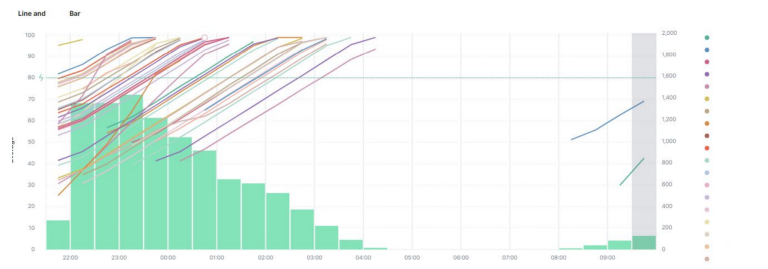
- 通常要把 Elasticsearch 放在另一台電腦後面，放在後端之後，或是透過一些安全機制讓資料庫存取的人是受掌控的。

- 避免我們的資料在網路上裸奔，造成資安危機，如和泰汽車

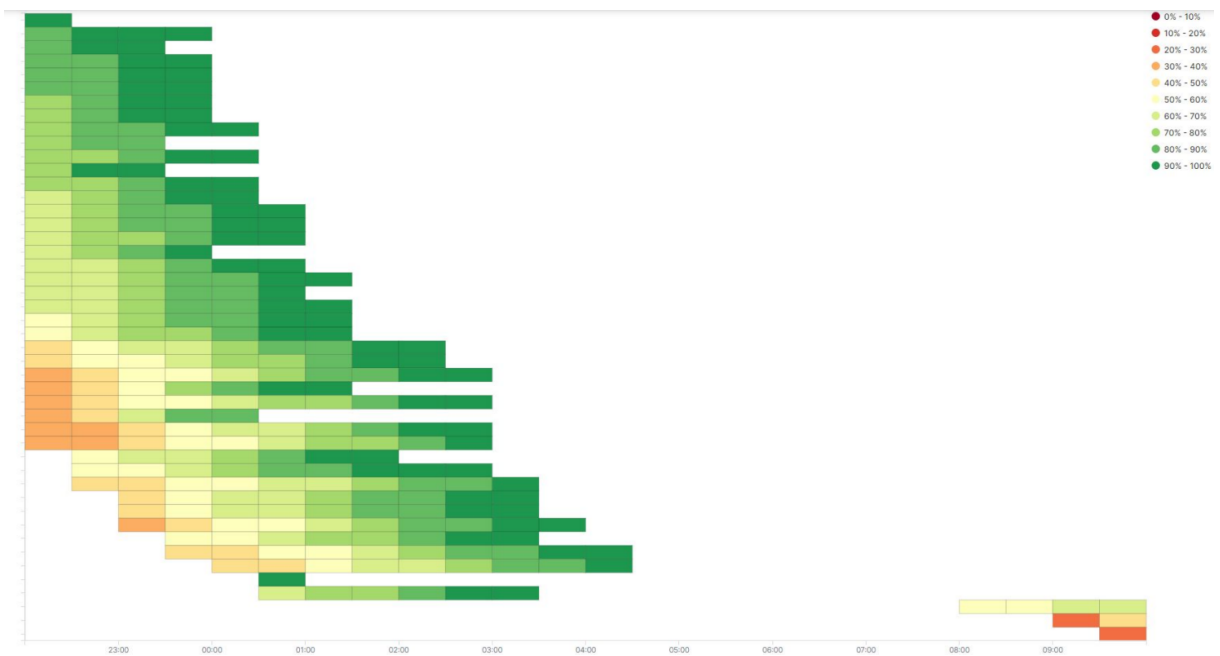


Extension

- Kibana 第 7 版



- 還可以做出這種酷酷的圖



Reference

- [1] You Know, for Search... | Elasticsearch: The Definitive Guide [master] | Elastic
 - Retrieved from <https://www.elastic.co/guide/en/elasticsearch/guide/master/intro.html> (March 23, 2023)
- [2] GitHub - elastic/elasticsearch: Free and Open, Distributed, RESTful Search Engine
 - Retrieved from <https://github.com/elastic/elasticsearch> (March 23, 2023)
- [3] DB-Engines Ranking - popularity ranking of database management systems

- Retrieved from <https://db-engines.com/en/ranking> (March 24, 2023)
- [4] 高鐵售票系統設計出錯 超賣八百餘個座位 | 大紀元
 - Retrieved from <https://www.epochtimes.com/b5/7/1/4/n1579307.htm> (March24, 2023)
- [5] 中租租車 用心服務
 - Retrieved from <https://www.rentalcar.com.tw/> (March24, 2023)
- [6] Install Elasticsearch with Docker | Elasticsearch Guide [8.6] | Elastic
 - Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/8.6/docker.html> (March 24, 2023)
- [7] Docker Docs: How to build, share, and run applications | Docker Documentation
 - Retrieved from <https://docs.docker.com/>