Introduction to MongoDB

2110322 Database Systems

February 2025

(ต้นฉบับจาก รศ.ดร.ดวงดาว วิชาดากุล)

ประเภทหลักๆ ของฐานข้อมูลกลุ่ม NoSQL (aka "not only SQL")

- Document Databases (e.g., MongoDB)
- Key-values stores (e.g., Redis)
- Column-oriented databases (e.g., Apache Hbase, Apache Parquet)
- Graph databases (e.g., Neo4j)
- Time series databases (e.g., InfluxDB)

NoSQL คืออะไรก็ตามที่
จัดการฐานข้อมูลได้
นอกจาก SQL

แนะนำ MongoDB เล็ก ๆน้อย ๆ

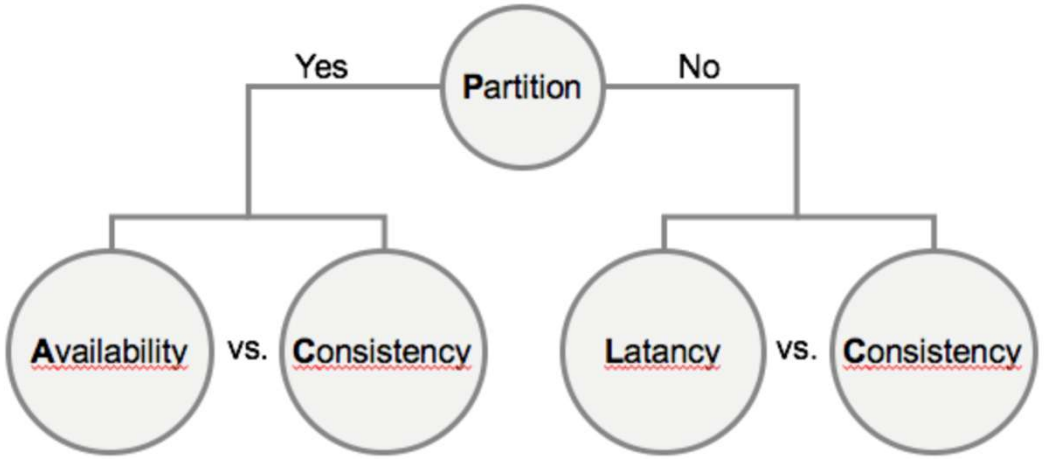
- MongoDB เป็น schema-less NoSQL document database
- เราสามารถเก็บข้อมูลในรูปแบบ JSON/BSON ได้โดยง่าย
- เหมาะกับข้อมูล unstructured ข้อมูลที่มีความซับซ้อน หรือที่อาจจะทำให้การกำหนดสกีมาตั้งแต่ต้นทำได้ยาก
- Scale ได้ดีทั้งเรื่อง data volume และ network traffic

Consistency and Availability of Databases

https://medium.com/@sumitsethia94/consistency-or-availability-of-databases-do-you-really-understand-cap-8ecf2b3bb099

ACID, CAP, PACELC

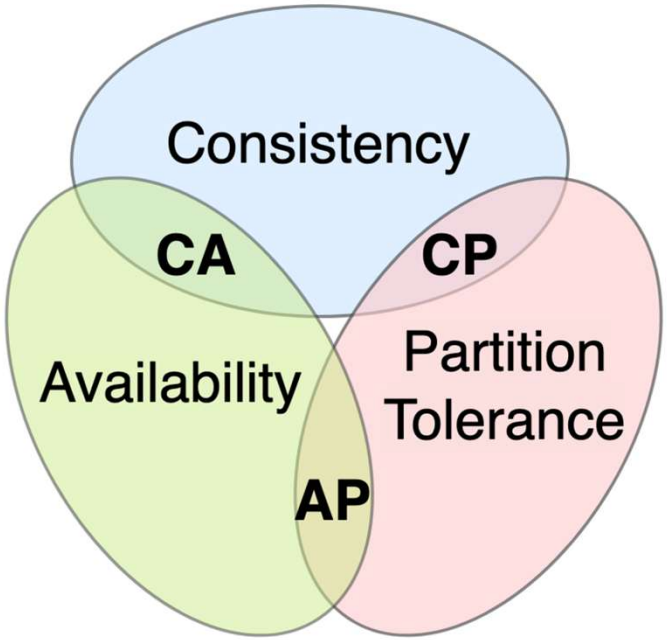
ทฤษฎี CAP คือเราได้แค่ 2 ใน 3 ของ Consistency, Availability, Partition เท่านั้น (ไม่ได้ทั้ง 3 พร้อมกัน)



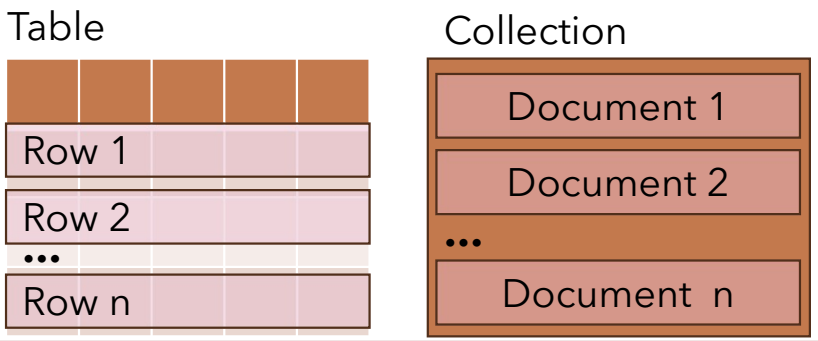
รูปจาก Medium ข้างต้น online access on Feb 1, 2024

CAP Theory

เลือกแค่ 2 ใน 3



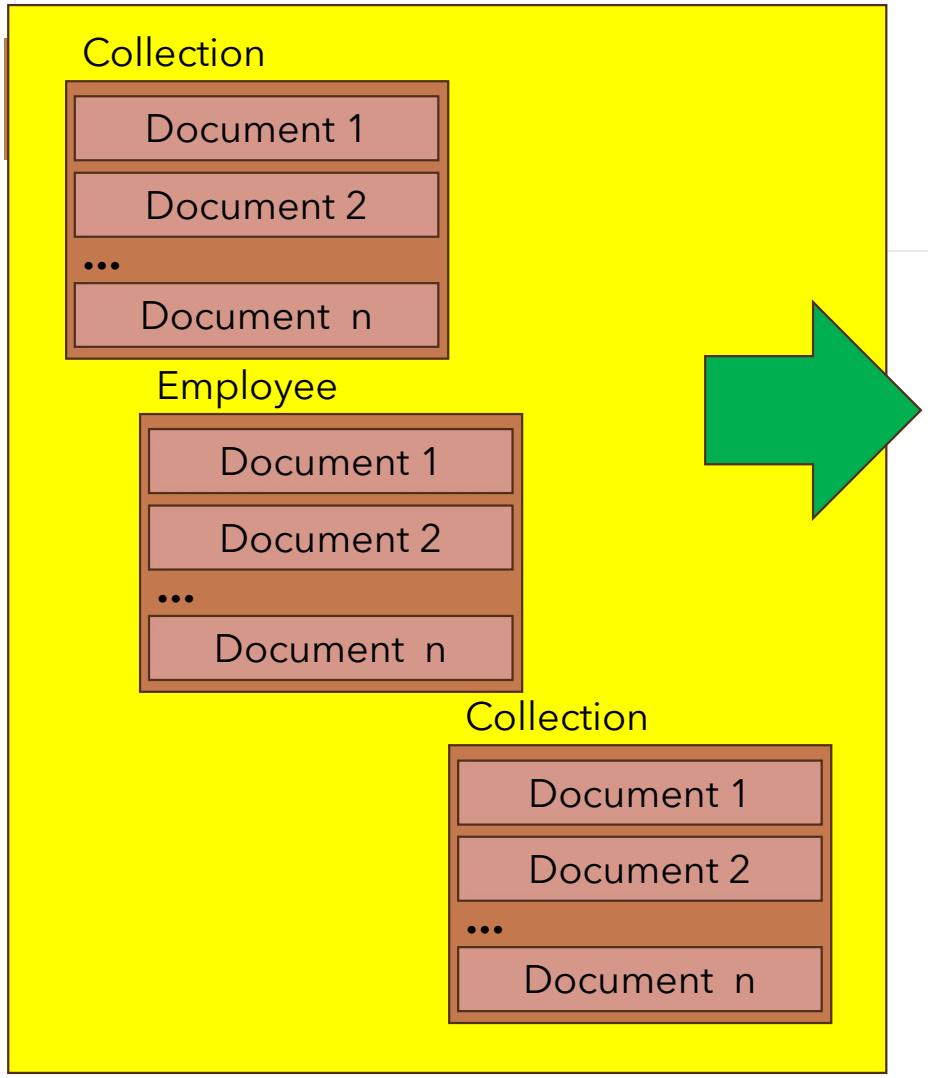
เปรียบเทียบคำศัพท์ที่ใช้



MySQL / PostgreSQL	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field
Index	Index
Table joins	$lookup
…	…

https://blog.sqlauthority.com/2020/05/23/mongodb-fundamentals-mapping-relational-sql-day-6-of-6/

NoSQL MongoDB Database



Employee

```
{
  "name": "Jane Doe",
  "age": 42,
}
```

```
{
  "name": "John Smith",
  "age": 45,
}
```

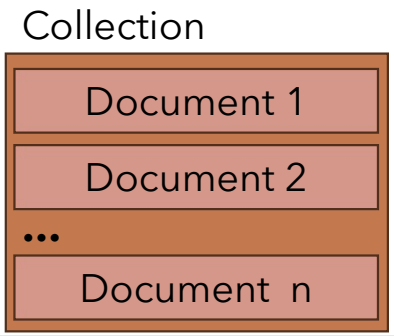
•••

```
{
  "name": “Susan White",
  "age": 45,
}
```

หนึ่ง Collection (table) Employee มี

หลายDocument (Row)

Document Schema



- เป็น JSON object ที่ใช้ในการอธิบายองค์ประกอบของเนื้อหาภายใน document และ embedded document ในแต่ละ collection
- เราสามารถใช้ **สกีมา** ในการระบุ required fields ระบุ content ของ fields และ validate schema เมื่อมีการ update / insert

https://docs.mongodb.com/realm/mongodb/document-schemas/

Filed level schema definition

ตัวอย่างข้อมูล

```
{
  "name": "Jane Doe",
  "age": 42,
  "favoriteColors": [
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" },
  ]
}
```

ตัวอยางสกีมาของข้อมูล

```
{
  "bsonType": "object",
  "required": ["name", "age", "favoriteColors"],
  "properties": {
    "name": {
      "bsonType": "string"
    },
    "age": {
      "bsonType": "int",
      "minimum": 13,
      "exclusiveMinimum": false
    },
    "favoriteColors": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "rank": { "bsonType": "int" },
          "name": { "bsonType": "string" },
          "hexCode": {
            "bsonType": "string",
            "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
          }
        }
      }
    }
  }
}
```

https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/

Filed level schema definition

ตัวอย่างข้อมูล

```
{
  "name": "Jane Doe",
  "age": 42,
  "favoriteColors": [
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" },
  ]
}
```

ตัวอยางสกีมาของข้อมูล

```
{
  "bsonType": "object",
  "required": ["name", "age", "favoriteColors"],
  "properties": {
    "name": {
      "bsonType": "string"
    },
    "age": {
      "bsonType": "int",
      "minimum": 13,
      "exclusiveMinimum": false
    },
    "favoriteColors": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "rank": { "bsonType": "int" },
          "name": { "bsonType": "string" },
          "hexCode": {
            "bsonType": "string",
            "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
          }
        }
      }
    }
  }
}
```

https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/

Filed level schema definition

ตัวอย่างข้อมูล

```
{
  "name": "Jane Doe",
  "age": 42,
  "favoriteColors": [
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" },
  ]
}
```

ตัวอยางสกีมาของข้อมูล

```
{
  "bsonType": "object",
  "required": ["name", "age", "favoriteColors"],
  "properties": {
    "name": {
      "bsonType": "string"
    },
    "age": {
      "bsonType": "int",
      "minimum": 13,
      "exclusiveMinimum": false
    },
    "favoriteColors": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "rank": { "bsonType": "int" },
          "name": { "bsonType": "string" },
          "hexCode": {
            "bsonType": "string",
            "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
          }
        }
      }
    }
  }
}
```

https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/

Filed level schema definition

ตัวอย่างข้อมูล

```
{
  "name": "Jane Doe",
  "age": 42,
  "favoriteColors": [
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" },
  ]
}
```

ตัวอย่างสกีมาของข้อมูล

```
{
  "bsonType": "object",
  "required": ["name", "age", "favoriteColors"],
  "properties": {
    "name": {
      "bsonType": "string"
    },
    "age": {
      "bsonType": "int",
      "minimum": 13,
      "exclusiveMinimum": false
    },
    "favoriteColors": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "rank": { "bsonType": "int" },
          "name": { "bsonType": "string" },
          "hexCode": {
            "bsonType": "string",
            "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
          }
        }
      }
    }
  }
}
```

https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/

Filed level schema definition

ตัวอย่างข้อมูล

```
{
  "name": "Jane Doe",
  "age": 42,
  "favoriteColors": [
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" },
  ]
}
```

ตัวอย่างสกีมาของข้อมูล

```
{
  "bsonType": "object",
  "required": ["name", "age", "favoriteColors"],
  "properties": {
    "name": {
      "bsonType": "string"
    },
    "age": {
      "bsonType": "int",
      "minimum": 13,
      "exclusiveMinimum": false
    },
    "favoriteColors": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "rank": { "bsonType": "int" },
          "name": { "bsonType": "string" },
          "hexCode": {
            "bsonType": "string",
            "pattern": "^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$"
          }
        }
      }
    }
  }
}
```

https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/

Common types in MongoDB

- Object
- Array
- String
- Number
- Boolean
- UUID (16 bytes)
- ObjectId
- Binary Data
- Mixed
- Sets
- Dictionaries

https://docs.mongodb.com/realm/mongodb/document-schemas/

Validation based-on defined schema

```
{
  "properties": {
    "_id": { "bsonType": "objectId" },
    "name": { "bsonType": "string" }
  }
}
```

จะมีการ Update JSON Object ด้วย
การ set name ให้เป็นค่าใหม่คือ 42 ซึ่ง
name ควรเป็น string

```
collection.updateOne(
{ "_id": BSON.ObjectId("5ae782e48f25b9dc5c51c4d0") },
{ "$set": { "name": 42 } }
)
```

WHY not correct?

https://docs.mongodb.com/realm/mongodb/document-schemas/

ลูกค้าต้องการทำเว็บไซต์ (HiVE) ให้ผู้ใช้มาสร้างคอร์สออนไลน์ผ่านโพสของตัวเองโดยต้องการเก็บข้อมูลต่อไปนี้

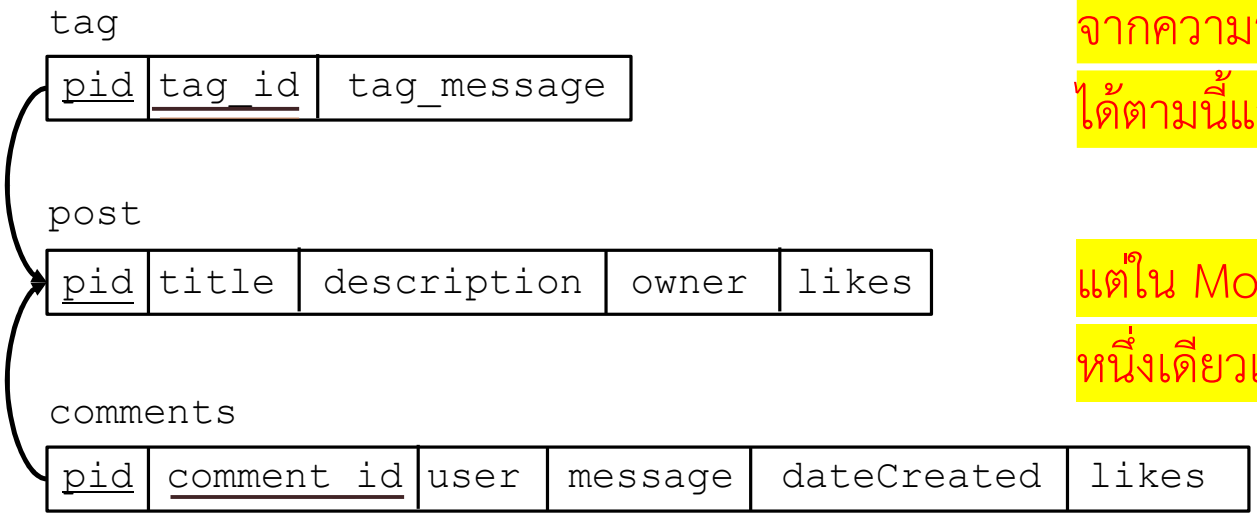
Case study ทำ website ชื่อ HIVE

- ทุกโพส (post) ต้องมี (1) ชื่อคอร์ส (2) คำอธิบาย (3) ชื่อเจ้าของคอร์ส (4) URL ของคอร์ส และ (5) จำนวนไลค์ (likes)
- ทุกโพสสามารถใส่แท็ก (tag) ได้หลายๆ แท็ก
- ทุกโพสอาจมีคอมเมนต์ (comments) จากผู้ใช้คนอื่นๆ โดยแต่ละ comment มี (1) ชื่อคนคอมเมนต์ (2) วัน-เวลาที่คอมเมนต์ (3) ตัวเนื้อความคอมเมนต์ และ (4) จำนวนไลค์ของคอมเมนต์

ถ้าต้องออกแบบสกีมาของ RDBMS จะมีกี่ตาราง?

HiVE's ER Diagram

HiVE's RDBMS schema design w/ referential integrity



จากความรู้ที่ผ่านมา เราสามารถออกแบบ ERD ได้ตามนี้และมี Schema ตามนี้

แต่ใน MongoDB เราอาจจะรวมตารางเป็นหนึ่งเดียวเลย ไม่ใช้กฎที่เรียนมาใน SQL

HiVE's MongoDB schema design (Only one collection "post")

แทนการแตกตารางในตัวอย่างนี้เราออกแบบสกีมา
ยการ embed ข้อมูลที่เกี่ยวข้องเข้ามาอยู่รวมกัน

Post collection

```
{
    title:'MongoDB Overview',
    description:'An intro to MongoDB/document',
    owner: 'lionking',
    tags: ['mongodb','database','NoSQL'],
    likes: 340
    comments: [
        {
            user: 'hikaru',
            message: 'nice intro',
            dateCreated:'2021-10-10',
            likes: 5
        },
        {
            user: 'miko',
            message: 'thanks',
            dateCreated:'2020-11-2',
            likes: 4
        },…
    ]
}
```

```
{
 "bsonType": "object",
 "required": ["title", "owner"],
 "properties": {
  "title": {
   "bsonType": "string"
  },
  "owner": {
   "bsonType": "string",
  },
  ....
  "comments": {
   "bsonType": "array",
   "uniqueItems": true,
   "items": {
    "bsonType": "object",
    "properties": {
     "user": { "bsonType": "objectId"},
     "message": { "bsonType": "string" },
     "dateCreated": {"bsonType": "date"},
      "like": {"bsonType": "int"}
    }
   }
  }
 }
```

จากความรู้ที่ผ่านมา เราไม่สามารถใช้กฎที่เรียนมาจาก RDB เดิมได้

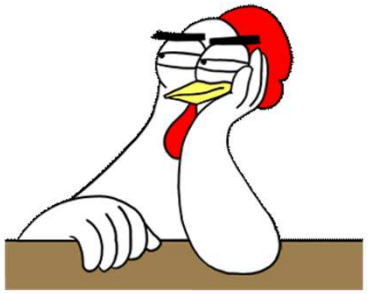
เราจะทำอย่างไรดี

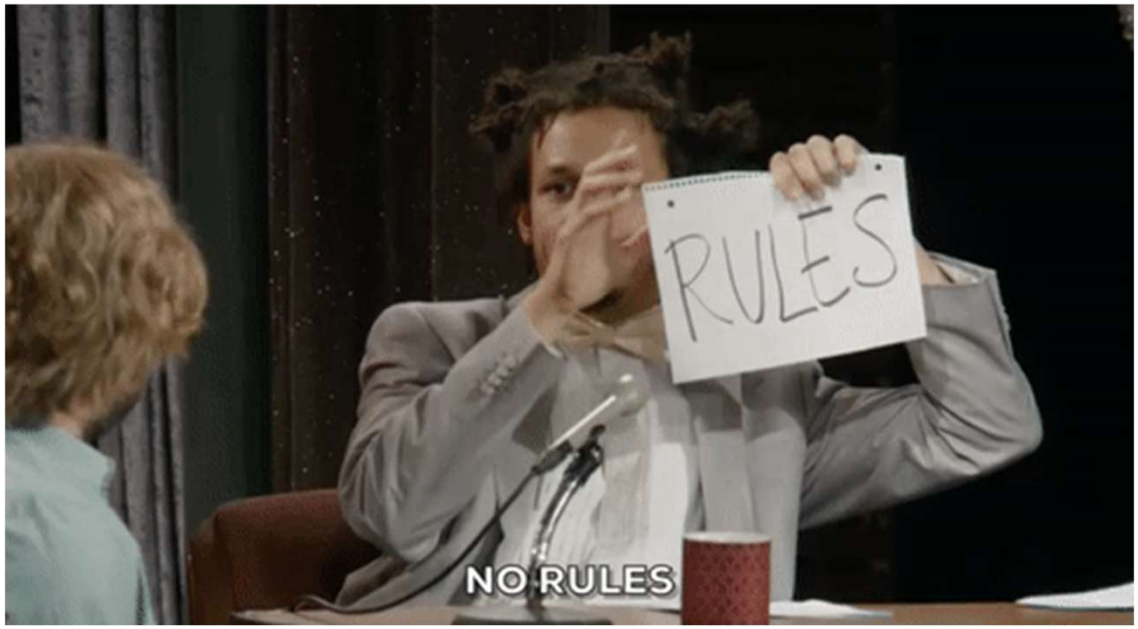
ถึงจุดนี้อาจจะมีคำถามว่า แล้วเมื่อไหร่จะรวมตารางกลายเป็น single collection ดี

การออกแบบ MongoDB schema มีกฎเกณฑ์อย่างไร

การออกแบบสกีมาของ MongoDB

- ไม่มีขั้นตอนที่เป็นมาตรฐาน
- ไม่มีอัลกอริทึมจำเพาะ
- ไม่มีกฎเกณฑ์ตายตัว





https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/

MongoDB Best Practices

- https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/

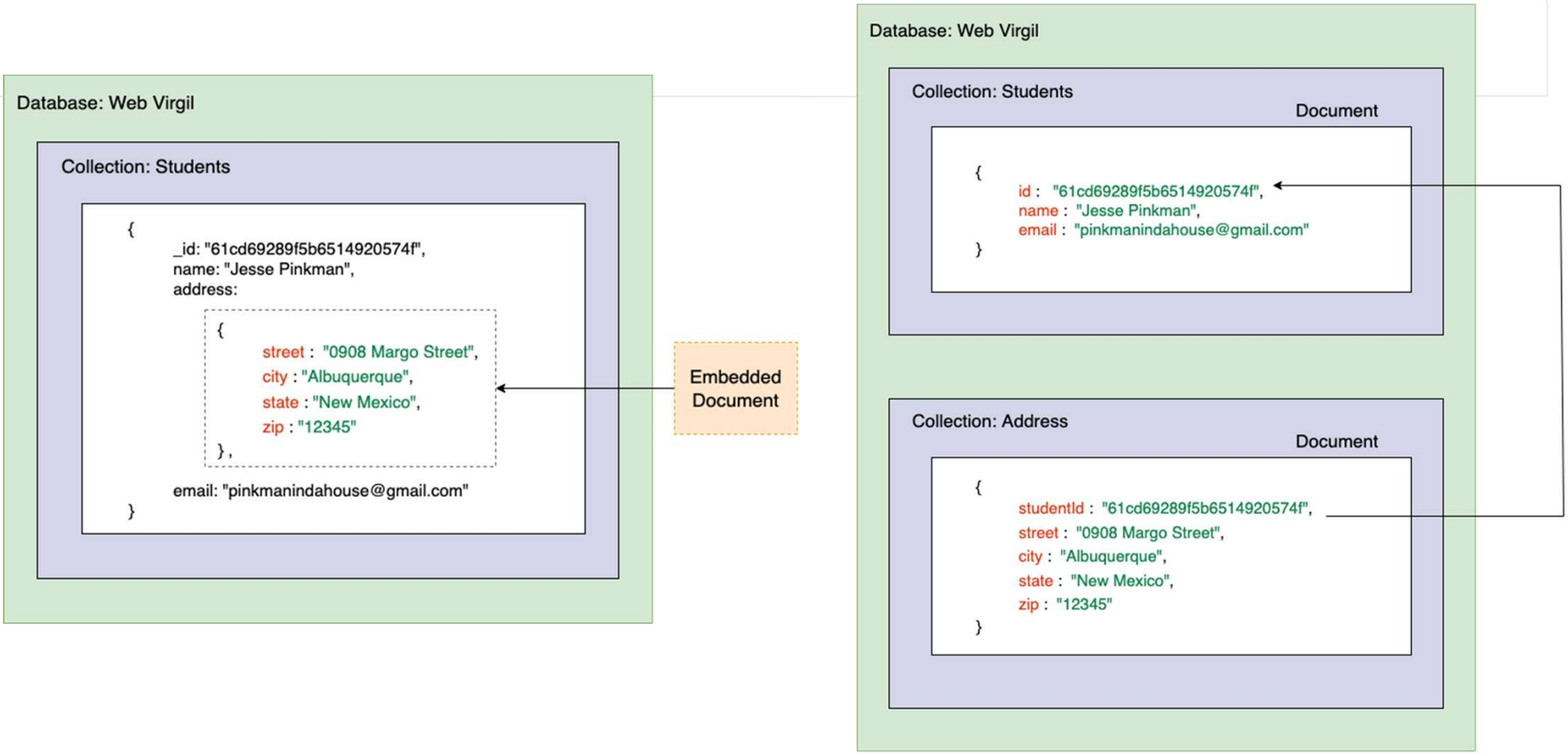
Best Practices vs. Standards?

แล้วเราจะใช้ Referencing หรือ Embedding ดี

- การออกแบบสกีมาของ MongoDB ต้องตัดสินใจว่าจะเก็บข้อมูลในลักษณะใดต่อไปนี้
  - Embedding คือ เก็บใน document เดียวกันเป็น Key-value ใหม่
  - Referencing คือ เก็บเป็น collection ใหม่ และลิงค์กันผ่าน operator ชื่อ $lookup

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

Embedding vs. Referencing



ข้อดี ข้อเสีย ของ Embedding

Pro

- ดึงข้อมูลทั้งหมดได้ใน query เดียว
- เลี่ยงการต้องลิงค์ข้อมูลระหว่าง collection ผ่าน $lookup เพื่อลด overhead
- การแก้ไขข้อมูลเป็น atomic มีการประกัน ACID property
- ถ้าใช้ multiple operations ต้องใช้ transaction และก็มีรองรับเรื่อง transaction ตั้งแต่ MongoDB 4

Con

- Document อาจมีขนาดใหญ่เกินทำให้มี overhead ในการดึงข้อมูล โดยข้อมูลที่ไม่จำเป็นในการ query ก็ต้องโหลดลงหน่วยความจำด้วย
- MongoDB จำกัดขนาด document ในรูปแบบ BSON ที่ 16 MB

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ข้อดี ข้อเสีย ของ Referencing

Pro	Con
• ขนาดของ document ไม่ใหญ่เกินและไม่เกินข้อจำกัดของ MongoDB • ลดความซ้ำซ้อนของข้อมูล แต่ไม่ใช่ประเด็นมากนัก เพราะถ้าซ้ำแต่ performance ดีตาม use cases ก็ไม่เป็นไร	• ต้องใช้อย่างน้อยสอง queries หรือใช้ operator $lookup

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

กฎข้อที่ 1 ของการออกแบบสกีมา 1

1. เลือกใช้ embedding ก่อนถ้าทำได้ รวมทั้งกรณีของ one-to-few ด้วยให้ใช้ embedding

Embedding First

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ประเภทความสัมพันธ์ของข้อมูล one-to-one

employee document

```
{
    "_id": "ObjectId('AAA')",
    "name": "Joe Karlsson",
    "company": "MongoDB",
    "twitter": "@JoeKarlsson1",
    "twitch": "joe_karlsson",
    "tiktok": "joekarlsson",
    "website": "joekarlsson.com",
}
```

รวมทุกอย่างใน Document เดียวเท่านั้น ไปไหน ไปด้วยกัน เพราะไม่มีภาระมาก

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

กฎข้อที่ 2 One-to-Few รวมอยู่ด้วยกันแบบ Embedding ได้ ยกเว้นต้องทำ Direct access จึงพิจารณาแยกแบบ Referencing

2

2.1 ปกติกรณี One-to-Few คือมีจำนวนไม่มากก็ทำแบบ Embedding ได้เลย

2.2 แต่ถ้ามี use cases ที่ต้องการเข้าถึงข้อมูลใดโดยตรง ก็ให้ทำข้อมูลนั้นๆเป็น document ของ collection ใหม่ เช่น ข้อมูล parts ในสไลด์หน้าที่แล้ว (Referencing) ให้ทำแบบกฎข้อ 3 ได้เลย

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ประเภทความสัมพันธ์ของข้อมูล One-to-Few

Access ตรง

Product document

```
{   "name": "left-handed smoke shifter",
    "manufacturer": "Acme Corp",
    "catalog_number": "1234",
    "parts": [
        {   "_id" : "ObjectID('AAAA')",
            "partno" : "123-aff-456",
            "name" : "#4 grommet",
            "qty": "94",
            "price":" 3.99"
        },
        {   "_id" : "ObjectID('AAAA')",
            "partno" : "123-aff-456",
            "name" : "#4 grommet",
            "qty": "94",
            "price":" 3.99"
        }
        .....
    ]
}
```

Few Embedding Documents (ตาราง Parts ซ้อนในตาราง Product)

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

กฎข้อที่ 3 One-to-Many แยกจากกันแบบ Referencing

3

3. สร้าง Collection ใหม่สำหรับส่วน Many (Referencing)

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ประเภทความสัมพันธ์ของข้อมูล One-to-Many

แยก Document ออกจากกัน
แบบ Referencing

Access ตรง

Product document

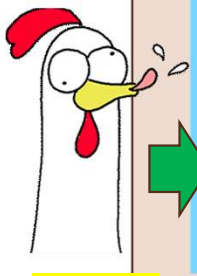
```
{    "name": "left-handed smoke shifter",
     "manufacturer": "Acme Corp",
     "catalog_number": "1234",
     "parts": ["ObjectID('AAAA')", "ObjectID('BBBB')", "ObjectID('CCCC')"]
}
```

Part document

```
{    "_id" : "ObjectID('AAAA')",
     "partno" : "123-aff-456",
     "name" : "#4 grommet",
     "qty": "94",
     "price":" 3.99"
}
```

```
{"_id" : "ObjectID('BBBB')",
.....
}
```

```
{"_id" : "ObjectID(CCCC')",
.....
}
```



Access ตรง

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

กฎข้อที่ 4 One-to-Squillions (อ้างถึงข้อมูลจำนวนมากมาย) 4

4. กรณีกฎข้อ 1 ข้อมูลที่ Embed อยู่ในรูปแบบ array ต้องมีการกำหนดขอบเขตไว้ด้วย เพราะจะบวมจนระเบิดได้ เช่น

- เดิมมี items เป็นหลักร้อย ก็ใช้แบบ 1-many reference ids ไปก่อน
- แต่ถ้ามี items หลักพัน ก็ชี้กลับแบบ one-to-squillions

สรุปค่า cardinality สูงๆ เป็นเหตุผลหลักในการ แยกออกไปเป็น document ใน collection ใหม่

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ประเภทความสัมพันธ์ของข้อมูล

one-to-squillions

ในกรณีนี้เราควรแยกข้อมูล log ของ host ออกมาจาก host

ตัวอย่างของข้อมูลในระดับ squillions เช่น ข้อมูล log ของระบบ

log document

```
{
    "time": ISODate("2014-03-28T09:42:41.382Z"),
    "message": "cpu is on fire!",
    "host": ObjectID("AAAB")
}
```

ชี้กลับไปที่ host แทน

```
{...
}
```

```
{ ...
}
```

host document

```
{
    "_id": ObjectID("AAAB"),
    "name": "goofy.example.com",
    "ipaddr": "127.66.66.66"
}
```

ประเภทความสัมพันธ์ของข้อมูล many-to-many

-

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

กฎข้อที่ 5 Many-to-Many

5

- ออกแบบให้มีการ Referencing ไปกลับระหว่าง 2 Collections
- ทั้งนี้เราจะออกแบบสกีมาอย่างไร ขึ้นอยู่กับ use cases และรูปแบบการเข้าถึงข้อมูลเป็นหลัก ว่ามีการ queries / updates ข้อมูลในลักษณะใดบ้าง โดยออกแบบสกีมาเพื่อให้สอดคล้องกับการใช้งาน
- ตัวอย่างแอปพลิเคชัน TODO ใน Sprint tasks ที่สมาชิกในทีม dev สามารถอาสา (volunteer) ในการทำได้หลาย tasks และ แต่ละ tasks ก็มีสมาชิกช่วยทำได้หลายคน

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

ประเภทความสัมพันธ์ของข้อมูล many-to-many

ในกรณีนี้เรามารถทำ reference ทั้งไปและกลับ

developer document

```
{    "_id": ObjectID("AAF1"),
     "name": "Kate Monster",
     "tasks": [ObjectID("ADF9"), ObjectID("AE02"), ObjectID("AE73")]
}
```

task document

```
{    "_id": ObjectID("ADF9"), "description":
     "Write blog post about MongoDB schema design",
     "due_date": ISODate("2014-04-01"),
     ”volunteers": [ObjectID("AAF1"), ObjectID("BB3G")]
}
```

https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

สรุป กฎของการออกแบบสกีมา (ถือเป็นแนวทางเท่านั้น)

1. One-to-One: embed ข้อมูลใน document เลย
2. One-to-Few: embed ข้อมูลใน document เลย แต่ก็อย่างว่า ถ้า few นั้นมีการเข้าถึงโดยตรงด้วย บ่อยๆ ก็แยกไปได้
3. One-to-Many: ถ้า embedding ไหวก็ทำ
4. One-to-Squilliion: ทำ referencing ชี้กลับ
5. Many-to-Many: ทำ referencing ไป-กลับ

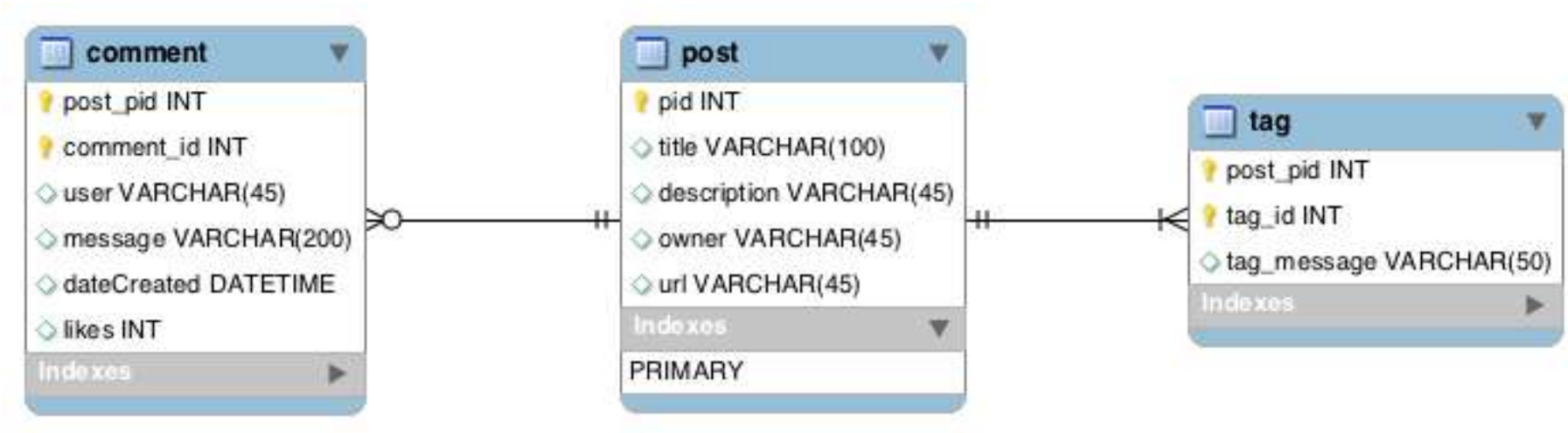
https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/

การใช้ MongoDB

หัวข้อต่อจากนี้ไป

- ใช้ Case study เป็นระบบ Hive
- ติดตั้ง MongoDB Community Server แบบ local
- ติดตั้งและใช้งานผ่าน Mongosh
- เรียนรู้ method ของ MongoDB เพื่อทำ CRUD (คล้าย DML ของ SQL)
- ติดตั้งและใช้งานผ่าน Compass
- ใช้งาน MongoDB ผ่าน Atlas (online MongoDB)

Case Study ที่ใช้คือ HiVE's ER Diagram



HiVE's MongoDB schema design

(Only one collection "post")

แทนการแตกตารางในตัวอย่างนี้เราออกแบบสกีมา โดยการembed ข้อมูลที่เกี่ยวข้องเข้ามาอยู่รวมกัน

Post collection

```
{
    title:'MongoDB Overview',
    description:'An intro to MongoDB/document',
    owner: 'lionking',
    tags: ['mongodb','database','NoSQL'],
    likes: 340
    comments: [
        {
            user: 'hikaru',
            message: 'nice intro',
            dateCreated:'2021-10-10',
            likes: 5
        },
        {
            user: 'miko',
            message: 'thanks',
            dateCreated:'2020-11-2',
            likes: 4
        },…
    ]
}
```

ออกแบบให้เป็น Embedding

```
{
  "bsonType": "object",
  "required": ["title", "owner"],
  "properties": {
    "title": {
      "bsonType": "string"
    },
    "owner": {
      "bsonType": "string",
    },
    ….
    "comments": {
      "bsonType": "array",
      "uniqueItems": true,
      "items": {
        "bsonType": "object",
        "properties": {
          "user": { "bsonType": "objectId"},
          "message": { "bsonType": "string" },
          "dateCreated": {"bsonType": "date"},
          "like": {"bsonType": "int"
        }
      }
    }
  }
}
```

Start to use mongodb

ติดตั้ง MongoDB ไว้แล้ว (ใครยังไม่มียกมือขึ้น)

1. Start the MongoDB server using

   System shell > mongod

2. Login to the MongoDB server via the MongoDB shell

   System shell > mongosh

MongoDB - create database

พิมพ์หน้าจอตามด้วย

- `show dbs` ใช้แสดงฐานข้อมูลที่อยู่ในระบบ

- `use DATABASE_NAME` ใช้ในการ `create or use` ฐานข้อมูลหนึ่งๆ

```
test> use hive
switched to db hive
hive>
```

- `db` ใช้เช็คว่าตอนนี้อยู่ `db` ไหน

```
hive> db
hive
```

MongoDB - drop database

พิมพ์หน้าจอตามด้วย

- `db.dropDatabase()` ใช้ในการ ลบ ฐานข้อมูลที่ `use` อยู่ ถ้ายังไม่ได้ `use` เลย ฐานชื่อ `test` จะถูกลบ

```
> db.dropDatabase()
{ "dropped" : "hive", "ok" : 1 }
```

Case sensitive นะครับ
ต้อง db.dropDatabase()

MongoDB - create collection

- `db.createCollection(name, option)`
  - `name` : ชื่อของคอลเลคชันที่จะสร้าง
  - `option` : ออปชันเกี่ยวกับหน่วยความจำที่จะใช้ และการทำ `index`

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a collection fixed size collecction that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexID	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. If **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

Deprecated since version 3.2

MongoDB - create collection
3 วิธีในการสร้าง collection ใหม่

พิมพ์หน้าจอตามด้วย

1
2
3

```
> use hive
switched to db hive
> db.createCollection("post")
{ "ok" : 1 }
> show collections
post
> db.createCollection("post_w_options",{capped : true, size : 6142800, max : 10000 })
{ "ok" : 1 }
> show collections
post
post_w_options
> db.post1.insert({"owner_name" : "lionking"})
WriteResult({ "nInserted" : 1 })
> show collections
post
post1
post_w_options
```

collection size in bytes

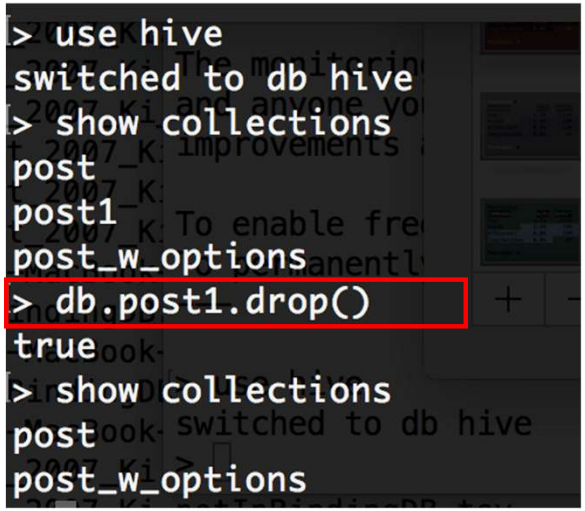
Max number of documents

Collection post1 ถูกสร้างโดยอัตโนมัติ โดยการ insert ข้อมูลเข้าไปเลย

MongoDB drop collection

พิมพ์หน้าจอตามด้วย

- `db.COLLECTION_NAME.drop()`



MongoDB datatypes

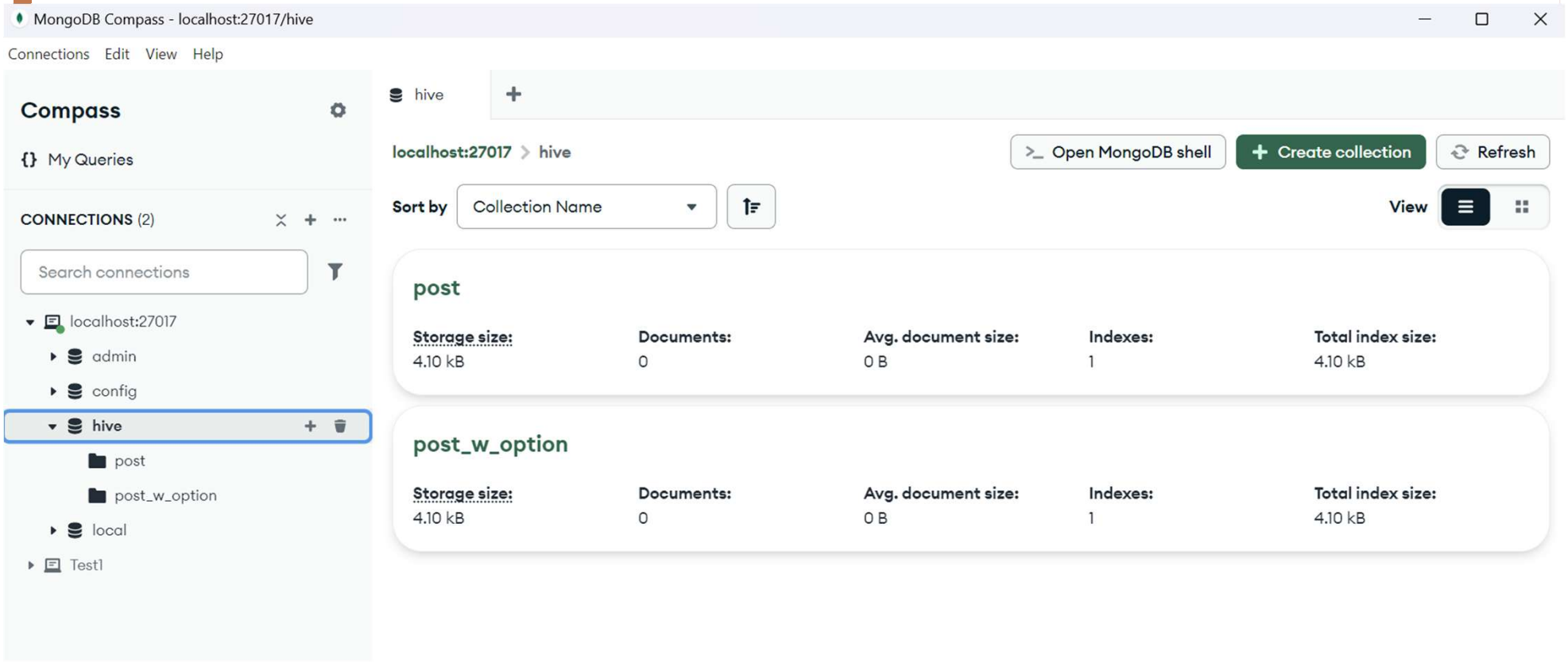
- **String** : This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.
- **Integer** : This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** : This type is used to store a boolean (true/ false) value.
- **Double** : This type is used to store floating point values.
- **Min/ Max keys** : This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** : This type is used to store arrays or list or multiple values into one key.
- **Timestamp** : ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** : This datatype is used for embedded documents.
- **Null** : This type is used to store a Null value.
- **Symbol** : This datatype is used identically to a string however, it's generally reserved for languages that use a specific symbol type.
- **Date** : This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** : This datatype is used to store the document's ID.
- **Binary data** : This datatype is used to store binay data.
- **Code** : This datatype is used to store javascript code into document.
- **Regular expression** : This datatype is used to store regular expression

http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf

ทดลองใช้ MongoDB Compass

- เปิด app ชื่อ MongoDBCompass
- แล้วทดลองสำรวจ Database ชื่อ hive
- แสดง collections ที่มีและ Document

เปิด app MongoDBCompass



MongoDB - Insert document

- `db.COLLECTION_NAME.insert(document)`

จะเห็นว่าเราไม่ได้กำหนด schema มาก่อนเลย

```
hive> db.post.insertOne({
...         title: 'MongoDB Overview',
...         'description': 'An introduction to MongoDB / document database',
...         'owner': 'lionking',
...         'url': 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
...         'tags': ['mongodb','database','NoSQL'],
...         'likes': 250})
{
  acknowledged: true,
  insertedId: ObjectId("652d05c44ce35c5655398092")
}
```

Response มาจากตัว MongoDB

MongoDB - Insert document (w/ multiple documents)

```
hive> db.post.insertMany([
...         {
...         title: 'MongoDB Overview',
...         description: 'An introduction to MongoDB / document database',
...         owner: 'lionking',
...         url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
...         tags: ['mongodb','database','NoSQL'],
...         likes: 250
...         },
...         {
...         title: 'MongoDB 101',
...         description: 'An introduction to MongoDB / document database via compass',
...         owner: 'lionking',
...         url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',
...         tags: ['mongodb','database','NoSQL', 'compass'],
...         likes: 500,
...         comments:[
...         {
...                 user: 'fiola',
...                 message: 'thanks! lionking',
...                 dateCreated: new Date(2020,7,26,1,30),
...                 likes: 0
...         }
...         ]
... }])
```

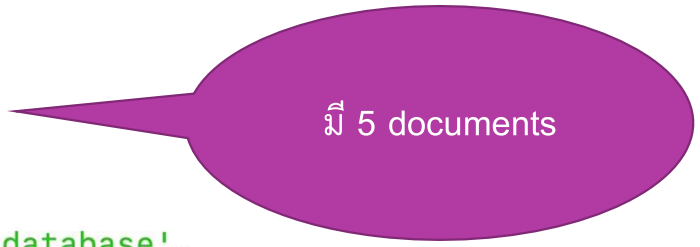
MongoDB – Insert more document

```
hive> db.post.insert({
...         title: 'MySQL Overview',
...         description: 'An introduction to MySQL / a RDBMS database',
...         owner: 'Hikaru',
...         url: 'http://mysql.com/mongodb_tutorial.pdf',
...         tags: ['mysql','database','SQL'],
...         likes: 180
...         })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652d09064ce35c5655398095") }
}
hive> db.post.insert({
...         title: 'Compass tutorial',
...         description: 'An introduction to MongoDB and Compass',
...         owner: 'Hikaru',
...         url: 'http://compass.com/tutorial',
...         tags: ['NoSQL','mongodb','database','compass'],
...         likes: 70
...         })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("652d09064ce35c5655398096") }
}
```

MongoDB - query document

- `db.COLLECTION_NAME.find()` SELECT * FROM post;

```
hive> db.post.find()
[
  {
    _id: ObjectId("652d05c44ce35c5655398092"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398093"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
```



MongoDB query document : RDBMS Vs MongoDB

Operation	Syntax	Example	RDBMS equivalent
Equality	{key:value}	db.post.find({owner: "lionking"})	where owner = "lionking"
Less than	{key:{$lt:value}}	db.post.find({likes: {**$lt**:100}})	where likes < 100
Less than or equal	{key:{$lte:value}}	db.post.find({likes:{ **$lte**:100}})	where likes <= 100
Greater than	{key:{$gt:value}}	db.post.find({likes:{ **$gt**:100}})	where likes > 100
Greater than or eqaul	{key:{$gte:value}}	db.post.find({likes:{ **$gte**:100}})	where likes >= 100
Not equal	{key:{$ne:value}}	db.post.find({likes:{ **$ne**:100}})	where likes != 100

```
hive> db.post.find({likes:{$gte:50}})
[
  {
    _id: ObjectId("652d05c44ce35c5655398092"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398093"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398094"),
    title: 'MongoDB 101',
    description: 'An introduction to MongoDB / document database via compass',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ],
    likes: 500,
    comments: [
```

MongoDB query document find() w/ conditions

```
db.post.find({likes:{$
gte:50}})
```

SELECT * FROM post WHERE likes >= 50;

Find() w/ AND in MongoDB

```
hive> db.post.find({likes:{$gte:50},owner:"Hikaru"})
[
  {
    _id: ObjectId("652d09064ce35c5655398095"),
    title: 'MySQL Overview',
    description: 'An introduction to MySQL / a RDBMS database',
    owner: 'Hikaru',
    url: 'http://mysql.com/mongodb_tutorial.pdf',
    tags: [ 'mysql', 'database', 'SQL' ],
    likes: 180
  },
  {
    _id: ObjectId("652d09064ce35c5655398096"),
    title: 'Compass tutorial',
    description: 'An introduction to MongoDB and Compass',
    owner: 'Hikaru',
    url: 'http://compass.com/tutorial',
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ],
    likes: 70
  }
]
```

```
{
attr:{$op:value},
attr:{$op:value},
...
}
```

AND

Find() w/ OR in MongoDB

```
hive> db.post.find({$or:[{likes:{$gte:50}},{owner:"hikaru"}]})
[
  {
    _id: ObjectId("652d05c44ce35c5655398092"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398093"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },

                    ...
```

```
{
$or:
[
{attr:{$op:value}},
{attr:{$op:value}},
...
]
}
```

OR

SELECT * FROM post WHERE likes >= 50 OR owner = “hikaru”;

```
hive> db.post.find({title:{$eq:"MongoDB 101"},$or:[{likes:{$gte:200}},{owner:"hikaru"}]})
[
  {
    _id: ObjectId("652d07564ce35c5655398094"),
    title: 'MongoDB 101',
    description: 'An introduction to MongoDB / document database via compass',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ],
    likes: 500,
    comments: [
      {
        user: 'fiola',
        message: 'thanks! lionking',
        dateCreated: ISODate("2020-08-25T18:30:00.000Z"),
        likes: 0
      }
    ]
  }
]
```

Find() w/ AND plus OR in MongoDB

```
{
arrt: {$op:value},
arrt: {$op:value},
$or:
[
{attr:{$op:value}},
{attr:{$op:value}},
...
]
}
```

AND plus OR

MongoDB - update document

- `db.COLLECTION_NAME.updateOne(SELECTION_CRITERIA, UPDATED_DATA)`

```
hive> db.post.updateOne({title:"MongoDB 101"},{$set:{title:"MongoDB HelloWorld"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

UPDATE post SET title = “MongoDB Hello World” WHERE title = “MongoDB 101”;

MongoDB - delete document

```
db.COLLECTION_NAME.remove(DELETION_CRITERIA)
```
เท่ากับ
```
deleteMany()
```

หรือ

```
db.COLLECTION_NAME.deleteOne(DELETION_CRITERIA)
db.COLLECTION_NAME.deleteMany(DELETION_CRITERIA)
```

```
hive> db.post.find()
[
  {
    _id: ObjectId("652d05c44ce35c5655398092"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398093"),
    title: 'MongoDB Overview',
    description: 'An introduction to MongoDB / document database',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL' ],
    likes: 250
  },
  {
    _id: ObjectId("652d07564ce35c5655398094"),
    title: 'MongoDB HelloWorld',
    description: 'An introduction to MongoDB / document database via compass',
    owner: 'lionking',
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ],
    likes: 500,
    comments: [
      {
```

```
hive> db.post.remove({title:"MongoDB Overview"})
{ acknowledged: true, deletedCount: 2 }
```

```
hive> db.post.deleteMany({title:"MongoDB Overview"})
{ acknowledged: true, deletedCount: 0 }
```

DELETE FROM post WHERE title = “MongoDB Overview”;

```
hive> db.post.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
```

```
hive> db.post.find()
```

DELETE * FROM post;

MongoDB projection

Select only some columns

ON/OFF

- `db.COLLECTION_NAME.find({},{KEY:1})`

condition

อย่าลิม insert ข้อมูลกลับเข้าไปก่อน มี 5 documents จากตัวอย่างตอนต้น

MongoDB projection

Select only some columns

```
hive> db.post.find({},{title:1})
[
  {
    _id: ObjectId("652d13974ce35c5655398097"),
    title: 'MongoDB Overview'
  },
  {
    _id: ObjectId("652d139e4ce35c5655398098"),
    title: 'MongoDB Overview'
  },
  { _id: ObjectId("652d139e4ce35c5655398099"), title: 'MongoDB 101' },
  {
    _id: ObjectId("652d13a84ce35c565539809a"),
    title: 'MySQL Overview'
  },
  {
    _id: ObjectId("652d13a84ce35c565539809b"),
    title: 'Compass tutorial'
  }
]
```

SELECT title FROM post;

MongoDB projection

Select only some columns

```
hive> db.post.find({},{title:1, _id:0})
[
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' },
  { title: 'MySQL Overview' },
  { title: 'Compass tutorial' }
]
```

SELECT title FROM post;

MongoDB projection

Select only some columns (array ออกมาได้เลย)

```
hive> db.post.find({},{title:1, _id:0, tags:1})
[
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB 101',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]
  },
  { title: 'MySQL Overview', tags: [ 'mysql', 'database', 'SQL' ] },
  {
    title: 'Compass tutorial',
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]
  }
]
```

MongoDB projection

Select only some columns $all (all items in array)

```
hive> db.post.find({tags: {$all:["mongodb", "database"]}},{title:1, _id:0, tags:1})
[
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB 101',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]
  },
  {
    title: 'Compass tutorial',
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]
  }
]
```

MongoDB projection

Select only some columns $in (some items in array)

```
hive> db.post.find({tags: {$in:["compass", "database"]}},{title:1, _id:0, tags:1})
[
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB Overview',
    tags: [ 'mongodb', 'database', 'NoSQL' ]
  },
  {
    title: 'MongoDB 101',
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]
  },
  { title: 'MySQL Overview', tags: [ 'mysql', 'database', 'SQL' ] },
  {
    title: 'Compass tutorial',
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]
  }
]
```

MongoDB - Limit / Skip records

- `db.COLLECTION_NAME.find().limit(NUMBER)`

```
hive> db.post.find({},{title:1, _id:0}).limit(3);
[
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' }
]
```

SELECT title FROM post LIMIT 3;

- `db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)`

```
hive> db.post.find({},{title:1, _id:0}).limit(3).skip(2)
[
  { title: 'MongoDB 101' },
  { title: 'MySQL Overview' },
  { title: 'Compass tutorial' }
]
```

SELECT title FROM post OFFSET 2 FETCH FIRST 3 ROW ONLY;

MongoDB - sort document

ON/OFF, ASC/DESC

- `db.COLLECTION_NAME.find().sort({KEY:1})`

```
hive> db.post.find({},{title:1, _id:0})
[
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' },
  { title: 'MySQL Overview' },
  { title: 'Compass tutorial' }
]
hive> db.post.find({},{title:1, _id:0}).sort({title:1})
[
  { title: 'Compass tutorial' },
  { title: 'MongoDB 101' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MySQL Overview' }
]
```

ACS เรียงน้อยไปมาก by default

SELECT title FROM post ORDER BY title;

MongoDB - sort document

ON/OFF, ASC/DESC

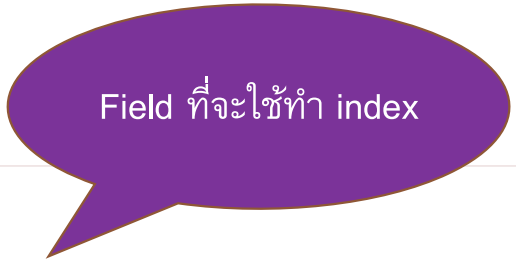
- `db.COLLECTION_NAME.find().sort({KEY:1})`

```
hive> db.post.find({},{title:1, _id:0})
[
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' },
  { title: 'MySQL Overview' },
  { title: 'Compass tutorial' }
]
hive> db.post.find({},{title:1, _id:0}).sort({title:-1})
[
  { title: 'MySQL Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' },
  { title: 'Compass tutorial' }
]
```

-1 เรียงจากมากไปน้อย

SELECT title FROM post ORDER BY title DESC;

MongoDB - indexing



- `db.COLLECTION_NAME.ensureIndex({KEY:1})`

```
hive> db.post.ensureIndex({title:1})
[ 'title_1' ]
```

MongoDB - aggregation

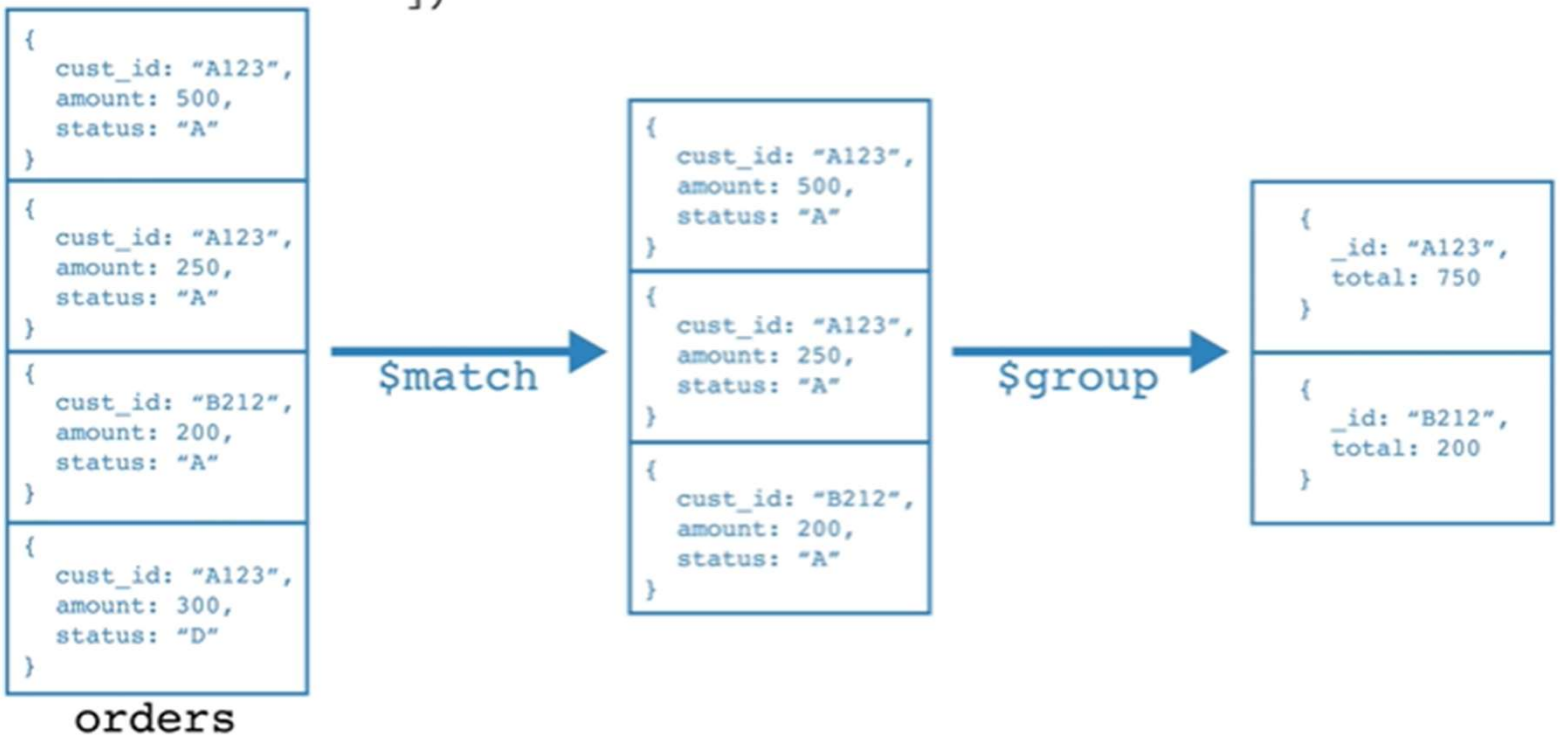
- db.COLLECTION_NAME.aggregate(AGGREGATION PIPELINE)
- An aggregation pipeline consists of one or more stages that process documents

- For example,

```
db.<<CollectionName>>.aggregate([ <<stage1>>, <<stage2>> ])
```

Collection

```
db.orders.aggregate( [
  $match stage ──→ { $match: { status: “A” } },
  $group stage ──→ { $group: { _id: “$cust_id”,total:  {$sum: “$amount” } } }
                   ])
```



MongoDB – aggregate functions: sum, avg

- `db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)`

```
hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}}])
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}}])
[
  { _id: 'lionking', total_likes: 1000 },
  { _id: 'Hikaru', total_likes: 250 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$avg: "$likes"}}}])
[
  { _id: 'lionking', total_likes: 333.3333333333333 },
  { _id: 'Hikaru', total_likes: 125 }
]
```

MongoDB – aggregation with sorting

- `db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)`

```
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}}, {$sort: {total_likes:1}}])
[
  { _id: 'Hikaru', total_likes: 250 },
  { _id: 'lionking', total_likes: 1000 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}}, {$sort: {total_likes:-1}}]
)
[
  { _id: 'lionking', total_likes: 1000 },
  { _id: 'Hikaru', total_likes: 250 }
]
```

MongoDB – aggregate function $push

- `db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)`

```
hive> db.post.aggregate([{$group: {_id:"$owner", urls: {$push: "$url"}}}])
[
  {
    _id: 'lionking',
    urls: [
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
      'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'
    ]
  },
  {
    _id: 'Hikaru',
    urls: [
      'http://mysql.com/mongodb_tutorial.pdf',
      'http://compass.com/tutorial'
    ]
  }
]
```

MongoDB – aggregate function $addToSet

- db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", urls: {$addToSet: "$url"}}}])
[
  {
    _id: 'Hikaru',
    urls: [
      'http://compass.com/tutorial',
      'http://mysql.com/mongodb_tutorial.pdf'
    ]
  },
  {
    _id: 'lionking',
    urls: [
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
      'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'
    ]
  }
]
```

MongoDB – aggregate function $last

- `db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)`

```
hive> db.post.aggregate([{$group: {_id:"$owner", urls: {$last: "$url"}}}])
[
  { _id: 'Hikaru', urls: 'http://compass.com/tutorial' },
  {
    _id: 'lionking',
    urls: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'
  }
]
```

MongoDB – having

- db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}}])
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}},{$match:{num_content:{$gte:2}}}])
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
```

MongoDB – aggregation expressions

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf

$lookup

post2

```
{
   "title" : "my first post",
   "author" : "Jim",
   "likes" : 5
},
{
   "title" : "my second post",
   "author" : "Jim",
   "likes" : 2
},
{
   "title" : "hello world",
   "author" : "Joe",
   "likes" : 3
}
```

comment2

```
{
   "postTitle" : "my first post",
   "comment" : "great read",
   "likes" : 3
},
{
   "postTitle" : "my second post",
   "comment" : "good info",
   "likes" : 0
},
{
   "postTitle" : "my second post",
   "comment" : "i liked this post",
   "likes" : 12
},
{
   "postTitle" : "hello world",
   "comment" : "not my favorite",
   "likes" : 8
},
{
   "postTitle" : "my last post",
   "comment" : null,
   "likes" : 0
}
```

https://www.stackchief.com/tutorials/%24lookup%20Examples%20%7C%20MongoDB

```
hive> db.post2.insertMany([
... {
...     "title" : "my first post",
...     "author" : "Jim",
...     "likes" : 5
... },
... {
...     "title" : "my second post",
...     "author" : "Jim",
...     "likes" : 2
... },
... {
...     "title" : "hello world",
...     "author" : "Joe",
...     "likes" : 3
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("652d31a183c372d7ba29d0be"),
    '1': ObjectId("652d31a183c372d7ba29d0bf"),
    '2': ObjectId("652d31a183c372d7ba29d0c0")
  }
}
```

```
hive> db.comment2.insert([
... {
...     "postTitle" : "my first post",
...     "comment" : "great read",
...     "likes" : 3
... },
... {
...     "postTitle" : "my second post",
...     "comment" : "good info",
...     "likes" : 0
... },
... {
...     "postTitle" : "my second post",
...     "comment" : "i liked this post",
...     "likes" : 12
... },
... {
...     "postTitle" : "hello world",
...     "comment" : "not my favorite",
...     "likes" : 8
... },
... {
...     "postTitle" : "my last post",
...     "comment" : null,
...     "likes" : 0
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("652d329f83c372d7ba29d0c1"),
    '1': ObjectId("652d329f83c372d7ba29d0c2"),
    '2': ObjectId("652d329f83c372d7ba29d0c3"),
    '3': ObjectId("652d329f83c372d7ba29d0c4"),
    '4': ObjectId("652d329f83c372d7ba29d0c5")
  }
}
```

$lookup

```
db.post2.aggregate([
   { $lookup:
      {
        from: "comment2",
        localField: "title",
        foreignField: "postTitle",
        as: "comments"
      }
   }
]).pretty()
```

```
{
"_id" : ObjectId("6176a821795674f634e37a6d"),
"title" : "my first post",
"author" : "Jim",
"likes" : 5,
"comments" : [
{
"_id" : ObjectId("6176a883795674f634e37a6e"),
"postTitle" : "my first post",
"comment" : "great read",
"likes" : 3
}
]
}
```

$lookup

```
hive> db.post2.aggregate([
...     { $lookup:
...         {
...             from: "comment2",
...             localField: "title",
...             foreignField: "postTitle",
...             as: "comments"
...         }
...     }
... ])
```

```
[
  {
    _id: ObjectId("652d31a183c372d7ba29d0be"),
    title: 'my first post',
    author: 'Jim',
    likes: 5,
    comments: [
      {
        _id: ObjectId("652d329f83c372d7ba29d0c1"),
        postTitle: 'my first post',
        comment: 'great read',
        likes: 3
      }
    ]
  },
  {
    _id: ObjectId("652d31a183c372d7ba29d0bf"),
    title: 'my second post',
    author: 'Jim',
    likes: 2,
    comments: [
      {
        _id: ObjectId("652d329f83c372d7ba29d0c2"),
        postTitle: 'my second post',
        comment: 'good info',
        likes: 0
      },
      {
        _id: ObjectId("652d329f83c372d7ba29d0c3"),
        postTitle: 'my second post',
        comment: 'i liked this post',
        likes: 12
      }
    ]
  },
  {
    _id: ObjectId("652d31a183c372d7ba29d0c0"),
    title: 'hello world',
    author: 'Joe',
    likes: 3,
    comments: [
      {
        _id: ObjectId("652d329f83c372d7ba29d0c4"),
        postTitle: 'hello world',
        comment: 'not my favorite',
        likes: 8
      }
    ]
  }
]
```

Next topics

- MongoDB backup and restore
- MongoDB Atlas
- MongoDB compass
- MongoDB authentication and authorization
- MongoDB connection via mongoose

MongoDB - create backup (via system shell)

- mongodump

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongodump
2023-10-16T20:03:42.760+0700    writing admin.system.version to dump/admin/system.version.bson
2023-10-16T20:03:42.769+0700    done dumping admin.system.version (1 document)
2023-10-16T20:03:42.770+0700    writing hive.post to dump/hive/post.bson
2023-10-16T20:03:42.770+0700    writing test.post to dump/test/post.bson
2023-10-16T20:03:42.770+0700    writing hive.post2 to dump/hive/post2.bson
2023-10-16T20:03:42.771+0700    writing hive.comment2 to dump/hive/comment2.bson
2023-10-16T20:03:42.772+0700    done dumping hive.comment2 (5 documents)
2023-10-16T20:03:42.772+0700    done dumping hive.post (5 documents)
2023-10-16T20:03:42.773+0700    writing hive.post_w_options to dump/hive/post_w_options.bson
2023-10-16T20:03:42.773+0700    done dumping hive.post2 (3 documents)
2023-10-16T20:03:42.773+0700    done dumping hive.post_w_options (0 documents)
2023-10-16T20:03:42.774+0700    done dumping test.post (4 documents)
(base) ~/Desktop/2110322_DBSYS_2566_1 $
```

จาก current directory ที่เรียกคำสั่ง mongodump จะมี directory dump เกิดขึ้น

MongoDB create backup (cont.)

```
cd dump/hive/
ls
```

```
comment2.bson                 post.bson                 post2.bson                 post_w_options.bson
comment2.metadata.json        post.metadata.json        post2.metadata.json        post_w_options.metadata.json
```

Syntax	Description	Example
mongodump --host HOST_NAME –port PORT_NUMBER	This commmand will backup all databases of specified mongod instance.	mongodump --host tutorialspoint.com –port 27017
mongodump --dbpath DB_PATH --out BACKUP_DIRECTORY		mongodump --dbpath /data/db/ --out /data/backup/
mongodump --collection COLLECTION --db DB_NAME	This command will backup only specified collection of specified database.	mongodump --collection mycol --db test

http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf

MongoDB export a specific collection (via system shell)

- `mongoexport`

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongoexport --collection=post --db=hive --out=post.json
2023-10-16T20:08:44.533+0700    connected to: mongodb://localhost/
2023-10-16T20:08:44.541+0700    exported 5 records
```

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongostat
insert query update delete getmore command dirty used flushes vsize   res qrw arw net_in net_out conn                time
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   111b   67.3k   27 Oct 16 20:09:38.461
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   221b   68.1k   27 Oct 16 20:09:39.462
    *0    *0     *0     *0       0     3|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   222b   68.2k   27 Oct 16 20:09:40.462
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   111b   67.5k   27 Oct 16 20:09:41.462
    *0    *0     *0     *0       0     1|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   112b   67.6k   27 Oct 16 20:09:42.461
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   111b   67.5k   27 Oct 16 20:09:43.462
    *0    *0     *0     *0       0     4|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   514b   68.5k   27 Oct 16 20:09:44.461
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   246b   67.9k   27 Oct 16 20:09:45.461
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   111b   67.5k   27 Oct 16 20:09:46.462
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   174b   68.1k   27 Oct 16 20:09:47.459
insert query update delete getmore command dirty used flushes vsize   res qrw arw net_in net_out conn                time
    *0    *0     *0     *0       0     1|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   259b   67.7k   27 Oct 16 20:09:48.462
    *0    *0     *0     *0       0     3|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   222b   68.2k   27 Oct 16 20:09:49.462
    *0    *0     *0     *0       0     2|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   221b   68.2k   27 Oct 16 20:09:50.462
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0  393G 31.0M 0|0 0|0   111b   67.5k   27 Oct 16 20:09:51.462
```

MongoDB deployment (via system shell)

- `mongostat`

แสดงจำนวน ของคำสั่งใน DB inserts, queries, updates, deletes, และ cursors

ตัวอย่างการทำงานของ cursors

ใน MongoDB shell ลองคำสั่งต่อไปนี้

```
var myCursor = db.post.find()

while (myCursor.hasNext()) {
      printjson(myCursor.next());
}
```

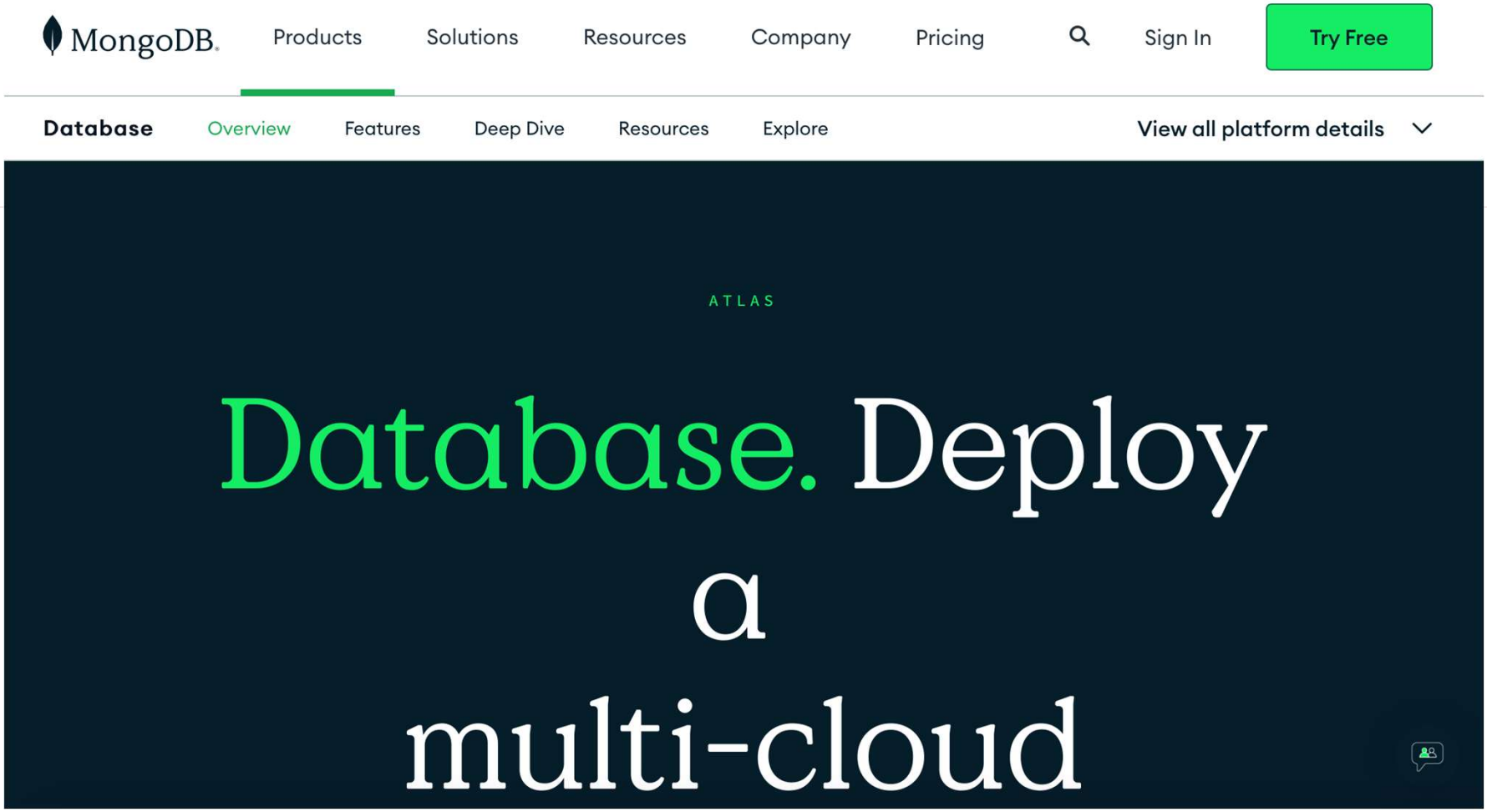
```
                    ns    total    read    write    2023-10-16T20:11:38+07:00
  admin.$cmd.aggregate      0ms     0ms      0ms
        admin.atlascli      0ms     0ms      0ms
  admin.system.version      0ms     0ms      0ms
    config.collections      0ms     0ms      0ms
config.system.sessions      0ms     0ms      0ms
   config.transactions      0ms     0ms      0ms
         hive.comment2      0ms     0ms      0ms
             hive.post      0ms     0ms      0ms
            hive.post2      0ms     0ms      0ms
   hive.post_w_options      0ms     0ms      0ms

                    ns    total    read    write    2023-10-16T20:11:39+07:00
  admin.$cmd.aggregate      0ms     0ms      0ms
        admin.atlascli      0ms     0ms      0ms
  admin.system.version      0ms     0ms      0ms
    config.collections      0ms     0ms      0ms
config.system.sessions      0ms     0ms      0ms
   config.transactions      0ms     0ms      0ms
         hive.comment2      0ms     0ms      0ms
             hive.post      0ms     0ms      0ms
            hive.post2      0ms     0ms      0ms
   hive.post_w_options      0ms     0ms      0ms

                    ns    total    read    write    2023-10-16T20:11:40+07:00
  admin.$cmd.aggregate      0ms     0ms      0ms
        admin.atlascli      0ms     0ms      0ms
  admin.system.version      0ms     0ms      0ms
    config.collections      0ms     0ms      0ms
config.system.sessions      0ms     0ms      0ms
   config.transactions      0ms     0ms      0ms
         hive.comment2      0ms     0ms      0ms
             hive.post      0ms     0ms      0ms
            hive.post2      0ms     0ms      0ms
   hive.post_w_options      0ms     0ms      0ms
```

MongoDB deployment

- `mongotop NUMBER`

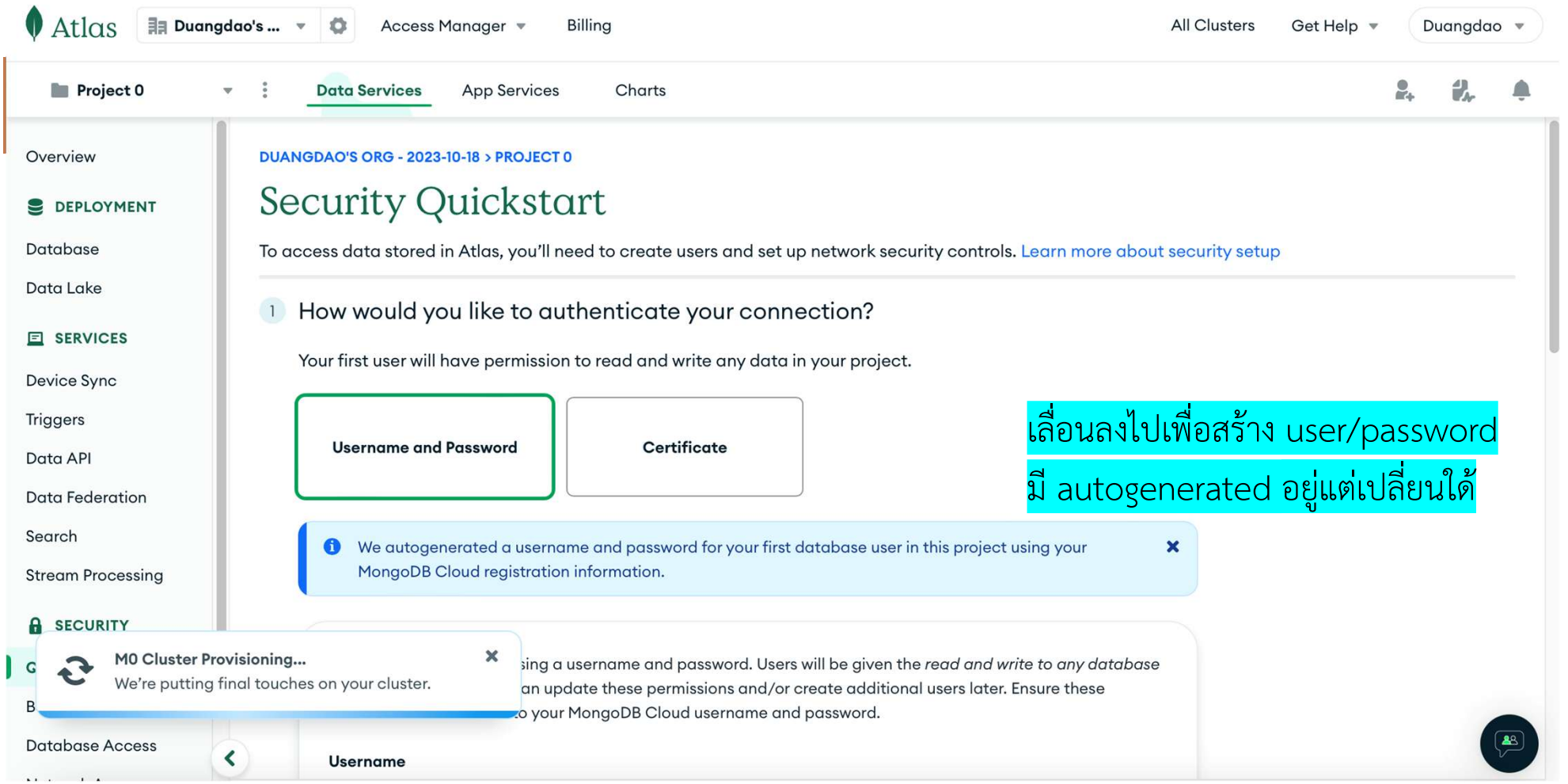
แสดง stats ของแต่ละคอลเลคชัน โดย `NUMBER` คือจำนวนวินาทีที่จะอัพเดตข้อมูลแต่ละรอบ

MongoDB Atlas



https://www.mongodb.com/cloud/atlas

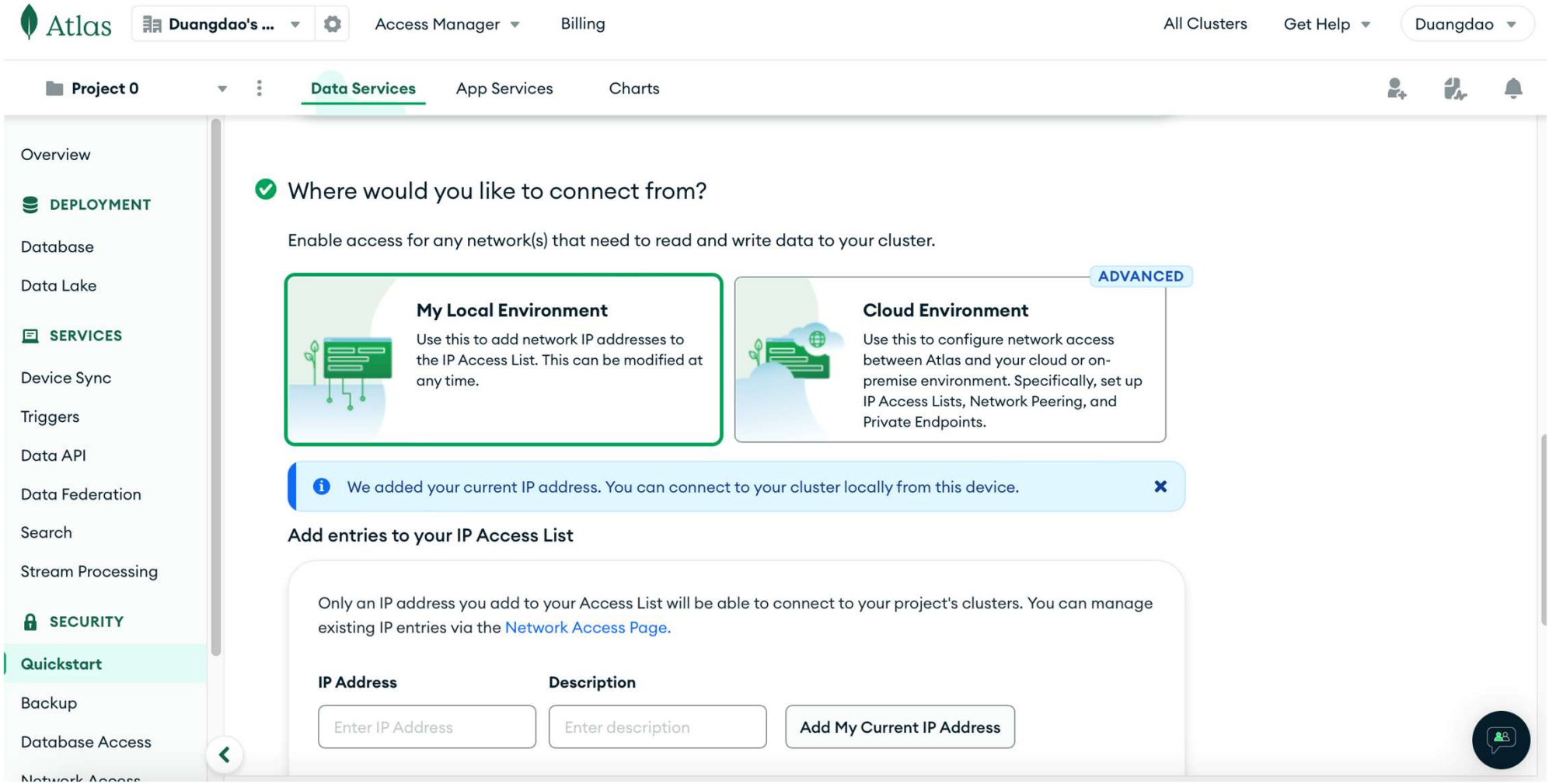
MongoDB Atlas



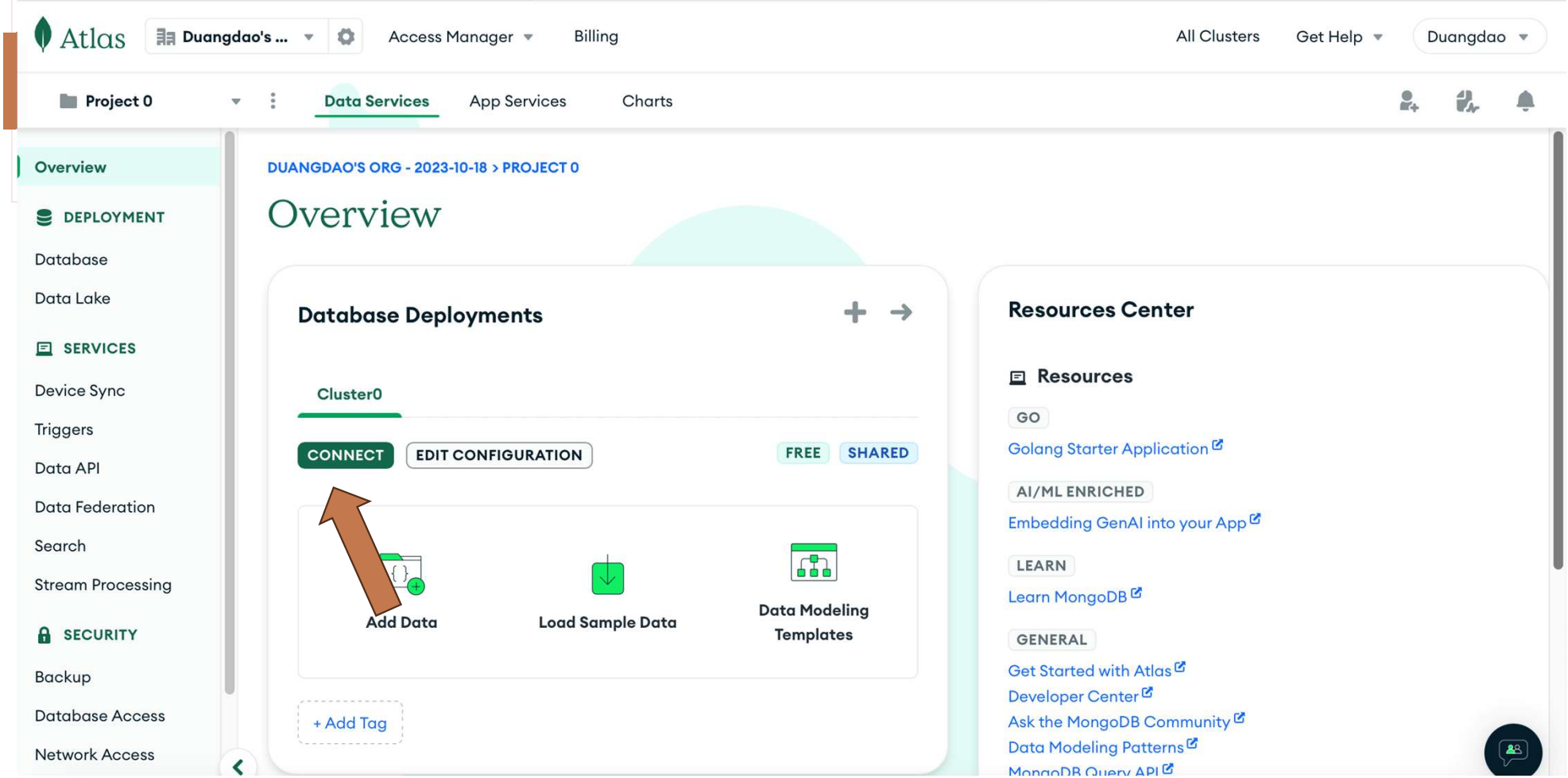
เลื่อนลงไปเพื่อสร้าง user/password

มี autogenerated อยู่แต่เปลี่ยนได้

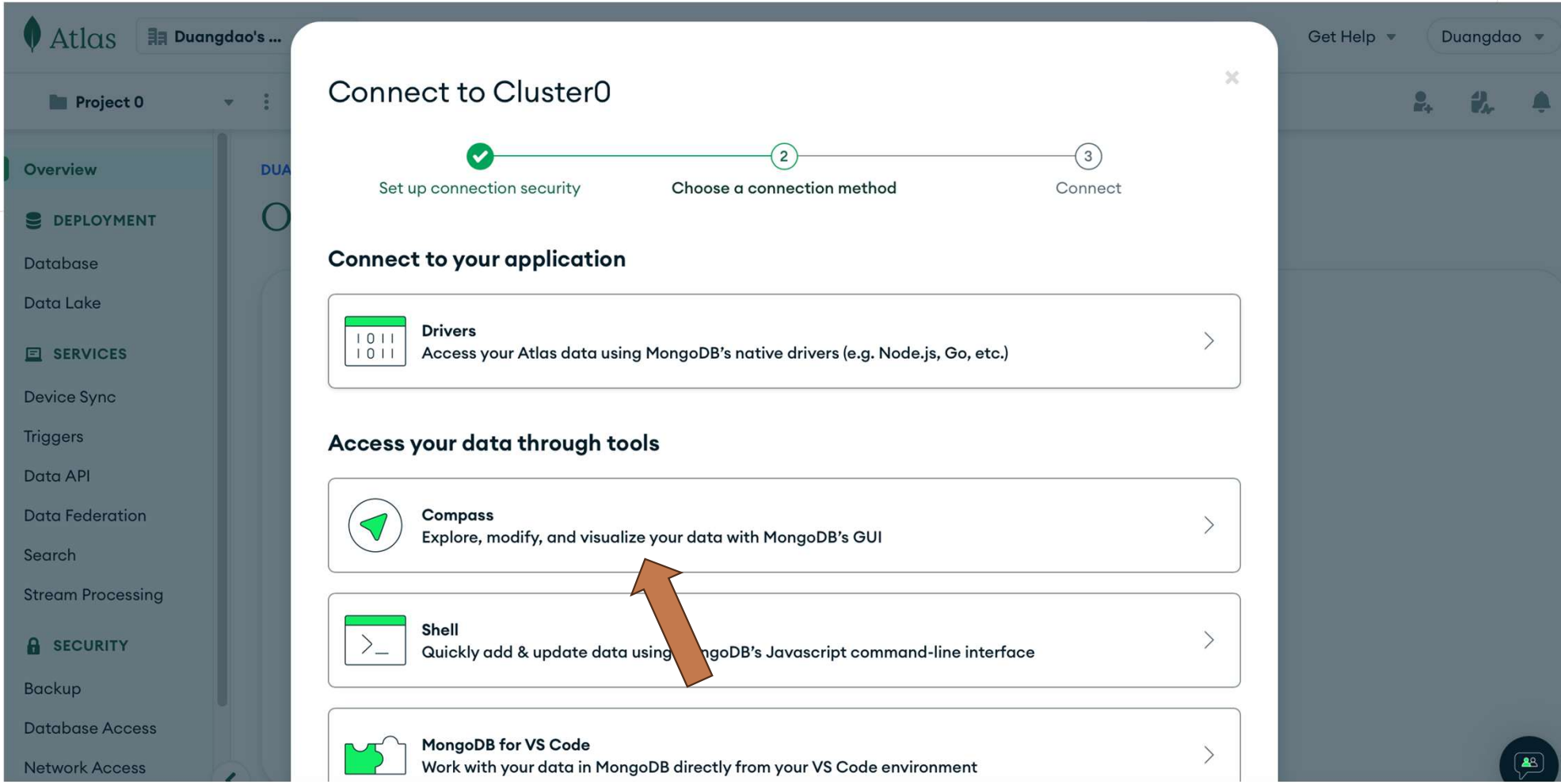
MongoDB Atlas



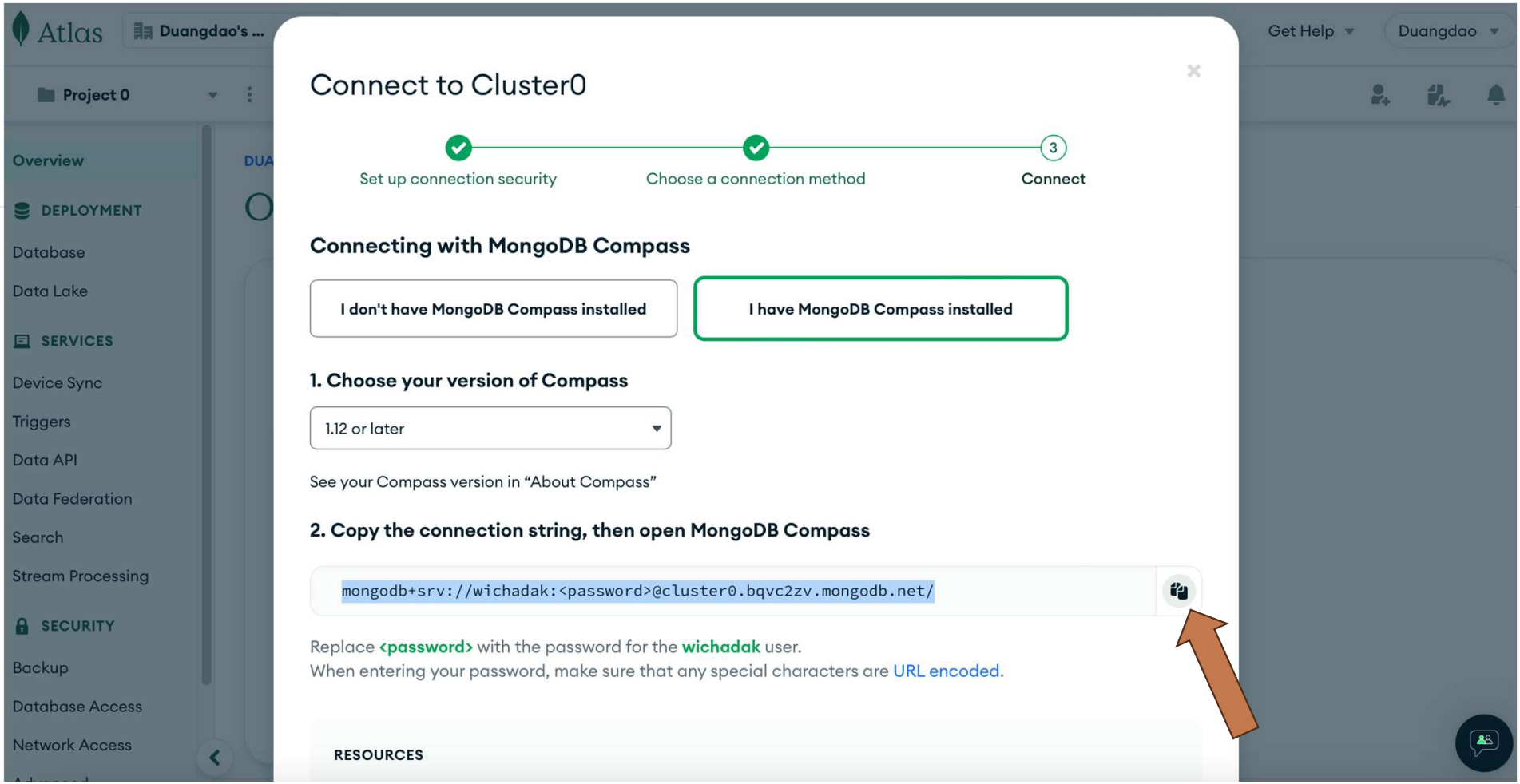
MongoDB Atlas



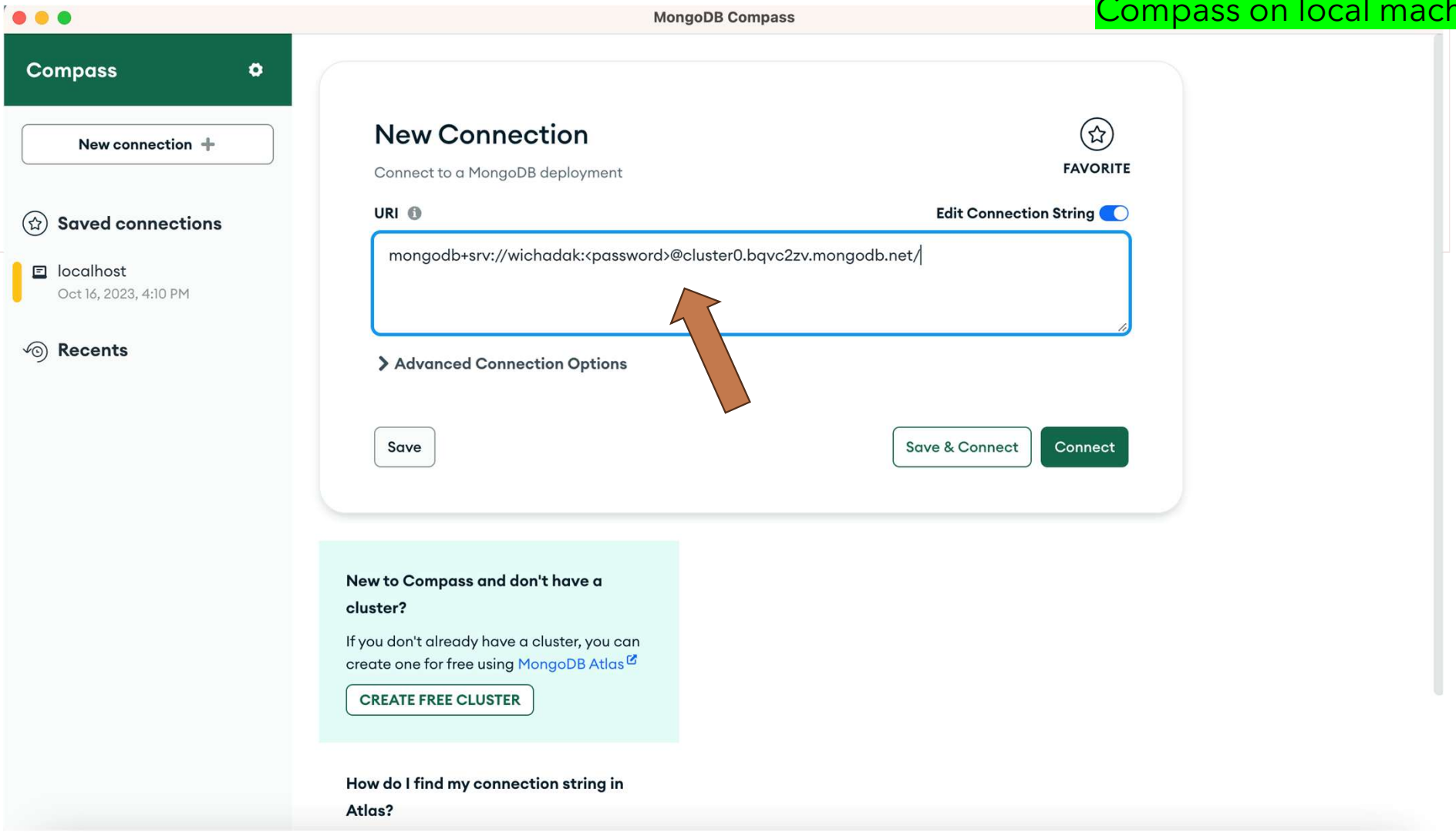
MongoDB Atlas



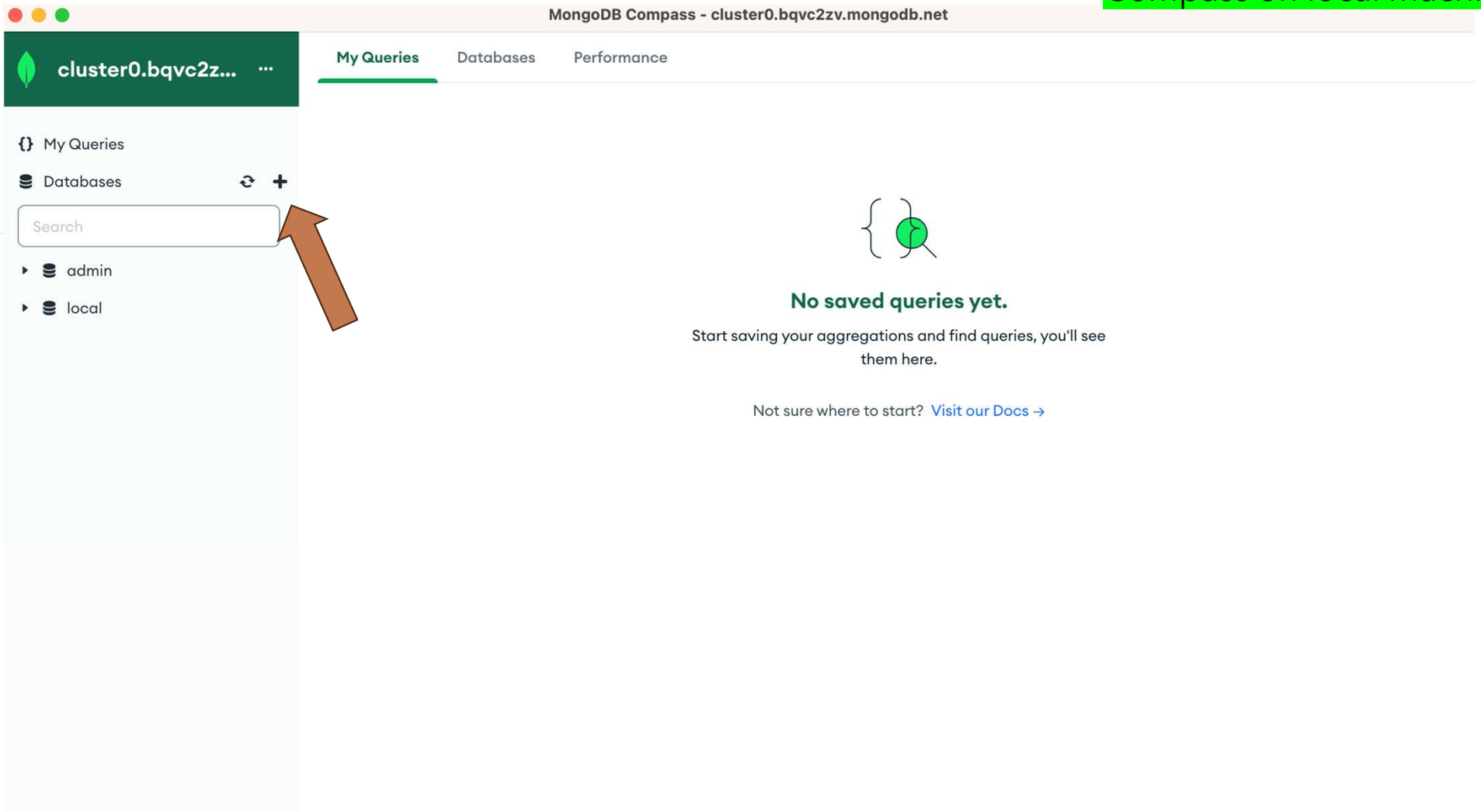
MongoDB Atlas

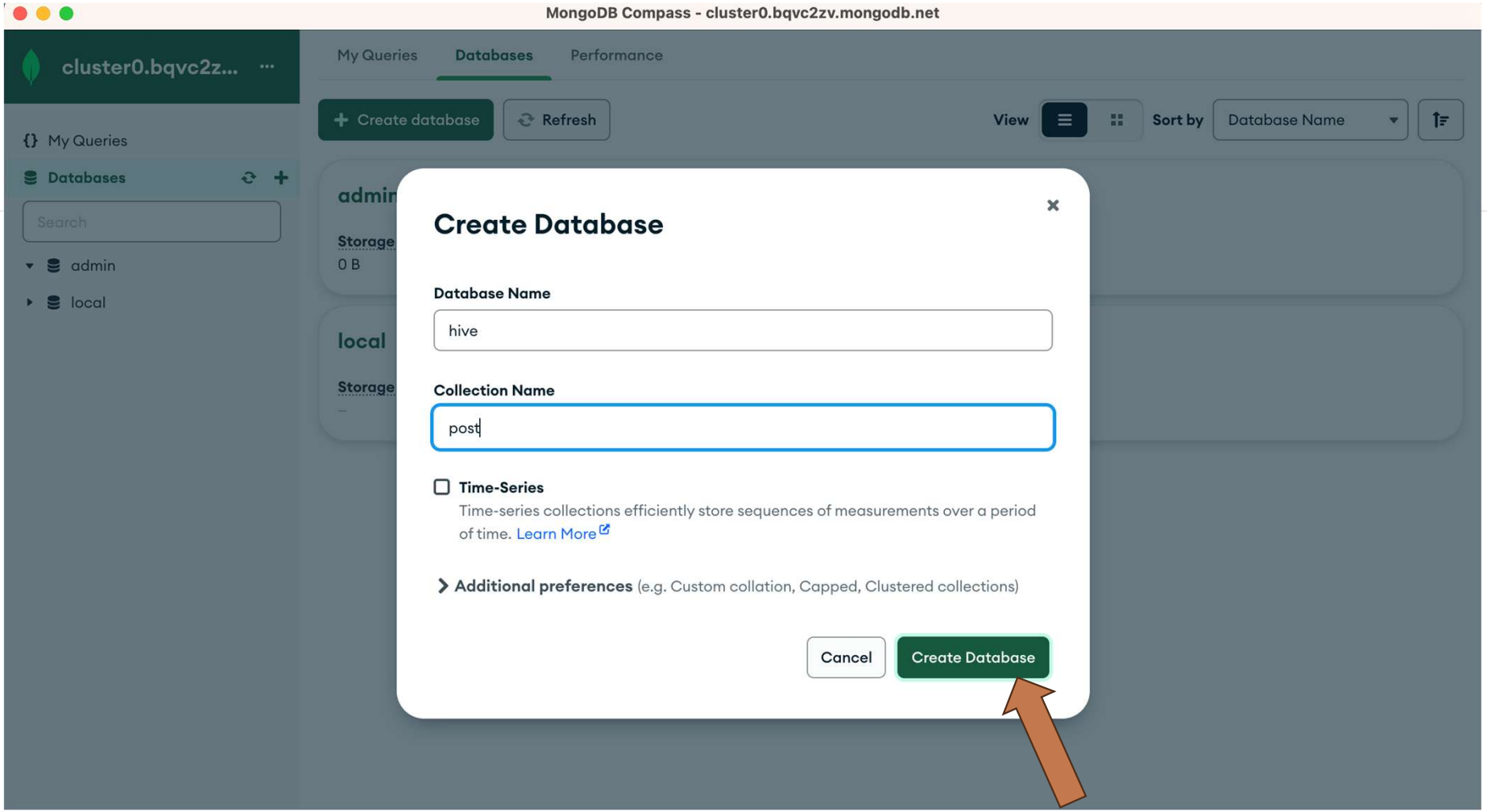


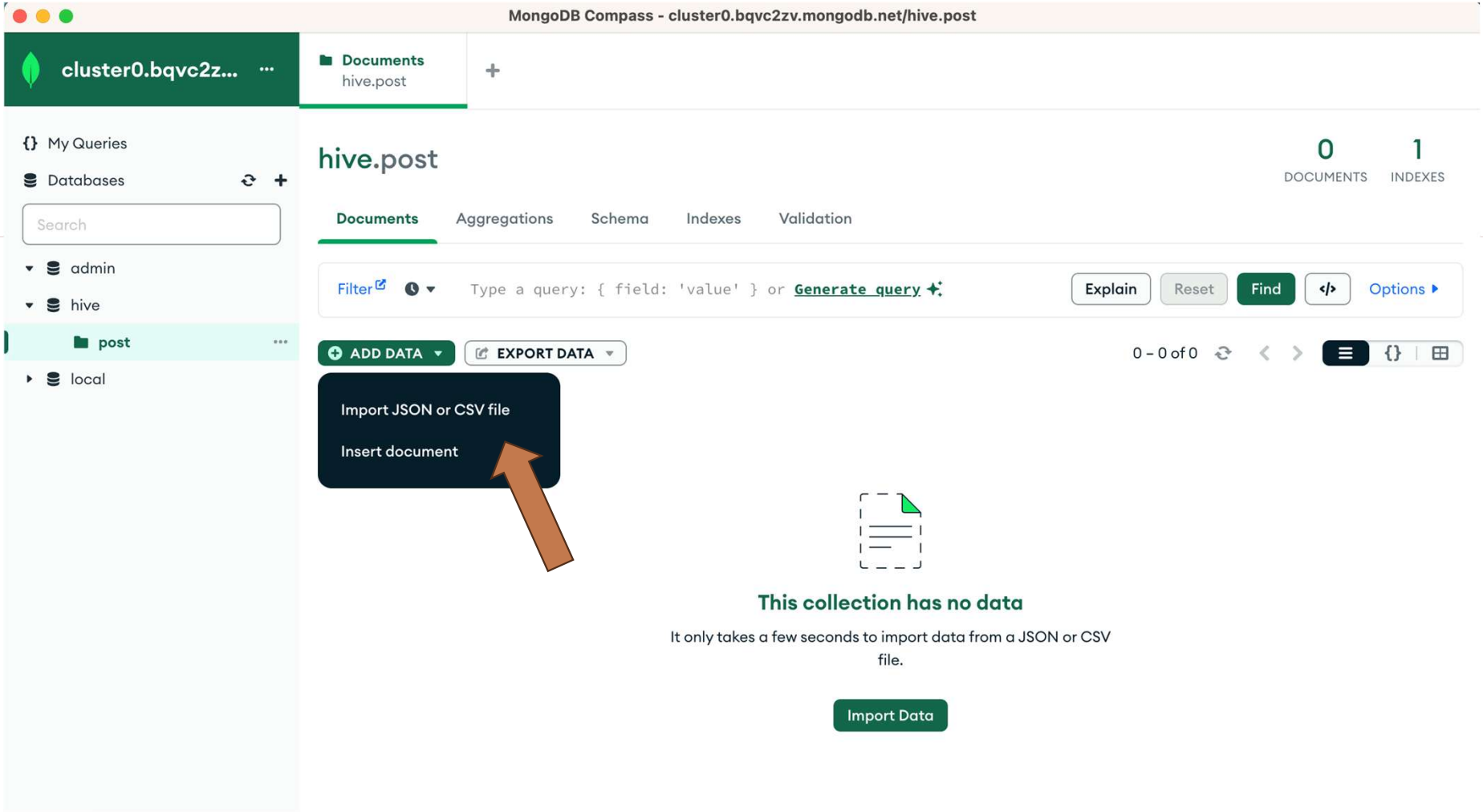
Compass on local machine

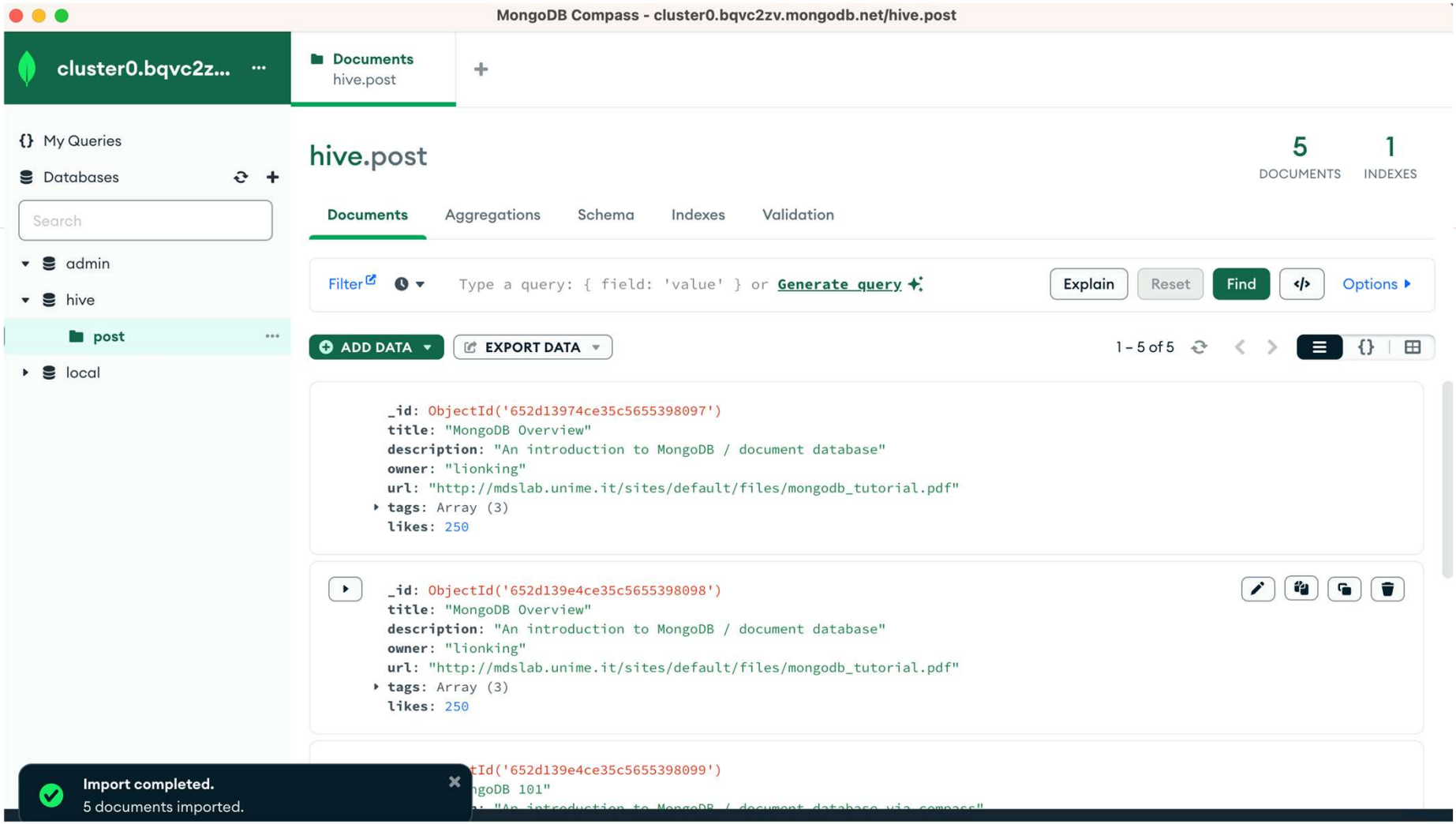


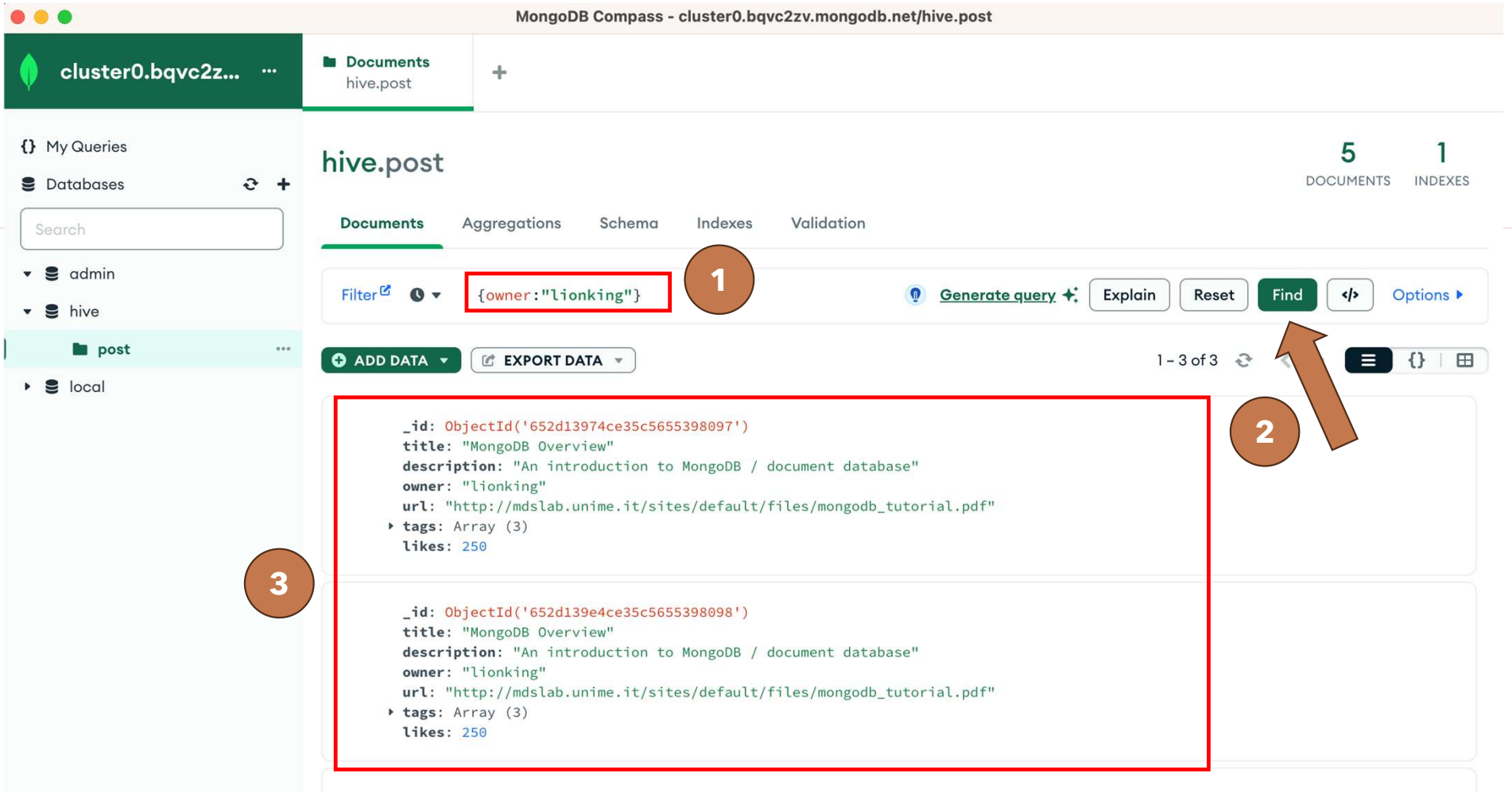
Compass on local machine

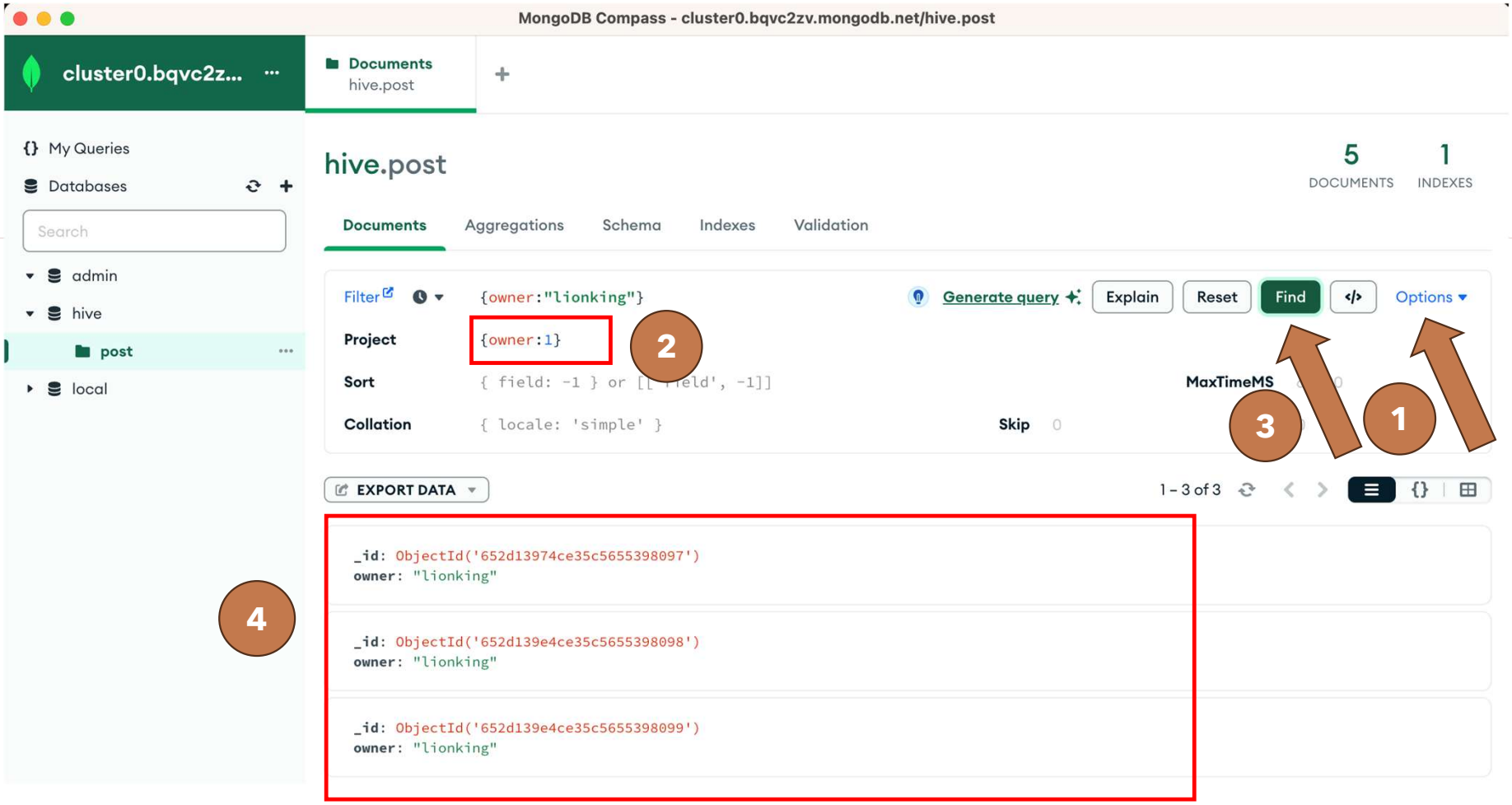


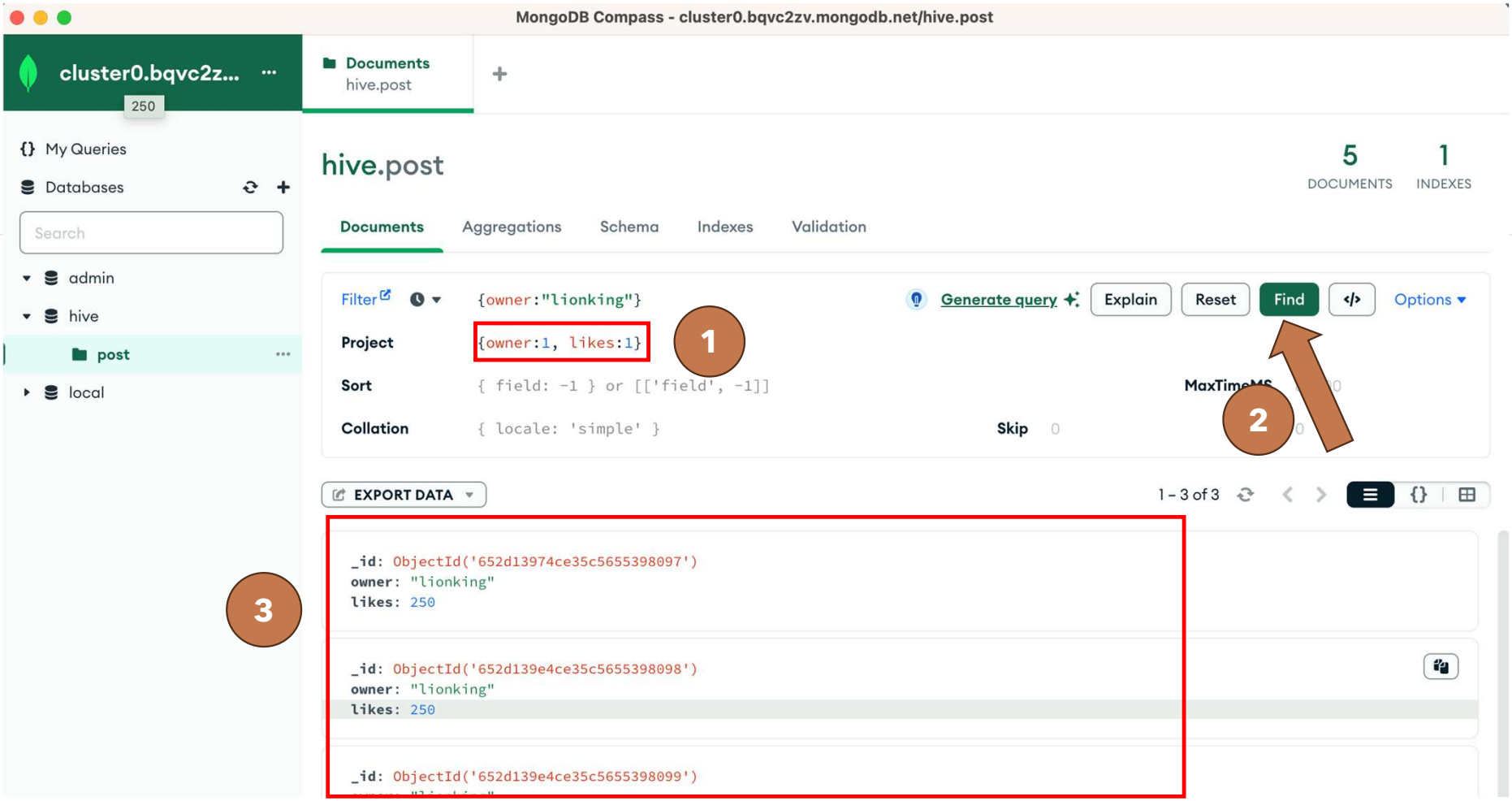

MongoDB Compass - cluster0.bqvc2zv.mongodb.net
cluster0.bqvc2z...
My Queries
Databases
Performance
Create database
Refresh
View
Sort by
Database Name
My Queries
Databases
Search
admin
local
admin
Storage
0 B
local
Storage
Create Database
Database Name
hive
Collection Name
post
Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More
Additional preferences (e.g. Custom collation, Capped, Clustered collections)
Cancel
Create Database

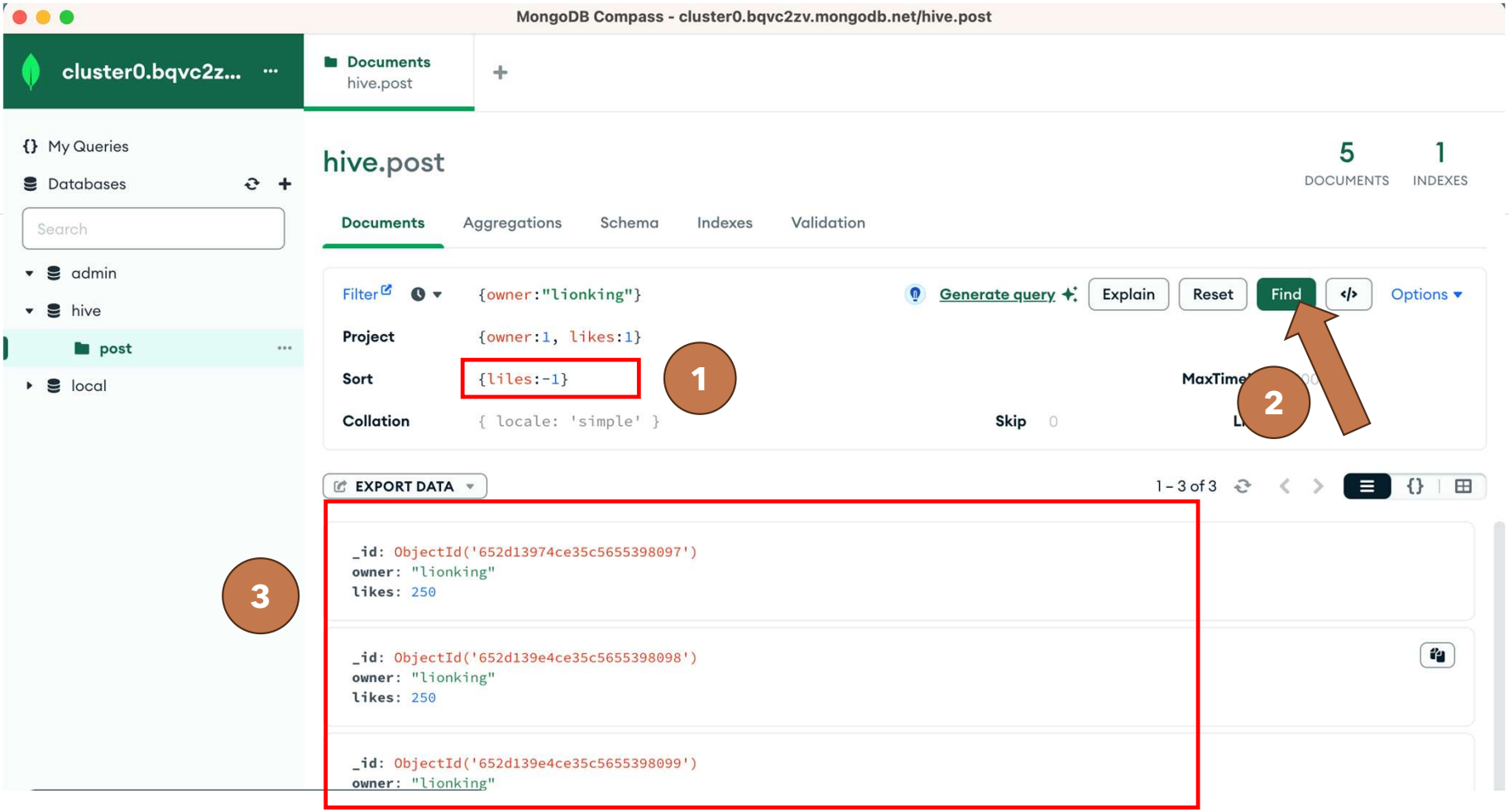

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
0
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
0 – 0 of 0
Import JSON or CSV file
Insert document
This collection has no data
It only takes a few seconds to import data from a JSON or CSV file.
Import Data

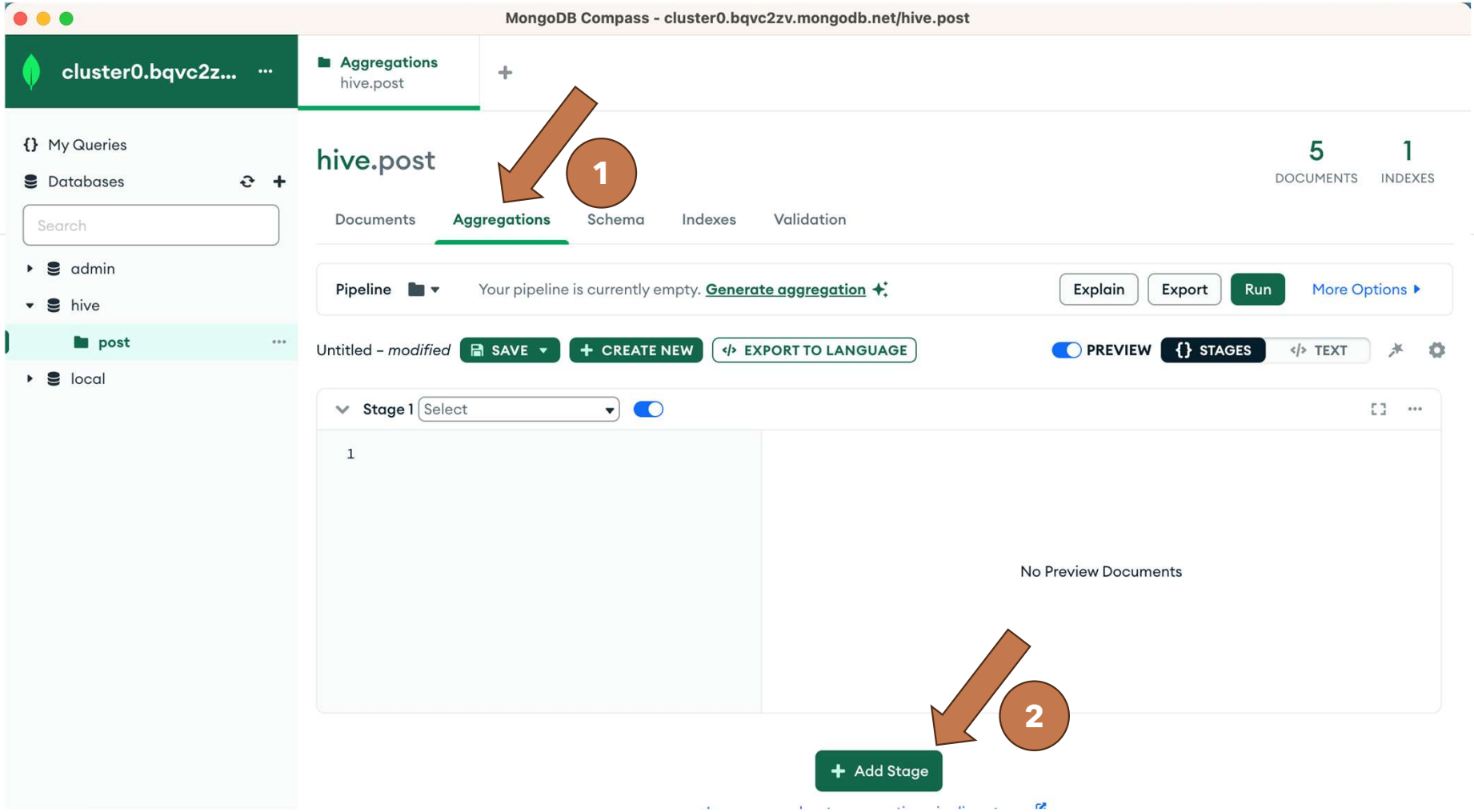

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 – 5 of 5
_id: ObjectId('652d13974ce35c5655398097')
title: "MongoDB Overview"
description: "An introduction to MongoDB / document database"
owner: "lionking"
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"
tags: Array (3)
likes: 250
_id: ObjectId('652d139e4ce35c5655398098')
title: "MongoDB Overview"
description: "An introduction to MongoDB / document database"
owner: "lionking"
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"
tags: Array (3)
likes: 250
Import completed.
5 documents imported.

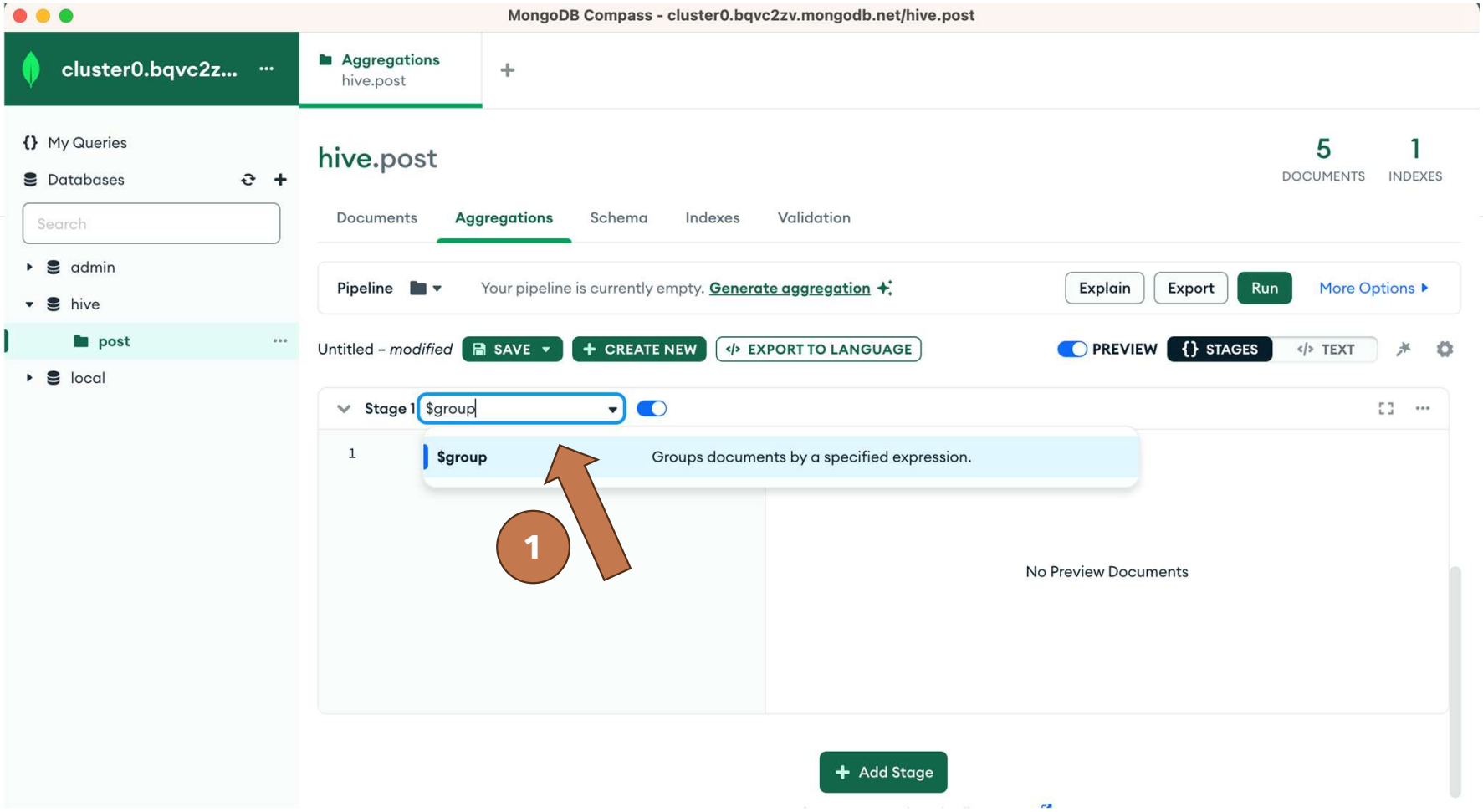

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
{owner:"lionking"}
1
Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 – 3 of 3
2
3
_id: ObjectId('652d13974ce35c5655398097')
title: "MongoDB Overview"
description: "An introduction to MongoDB / document database"
owner: "lionking"
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"
tags: Array (3)
likes: 250
_id: ObjectId('652d139e4ce35c5655398098')
title: "MongoDB Overview"
description: "An introduction to MongoDB / document database"
owner: "lionking"
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"
tags: Array (3)
likes: 250

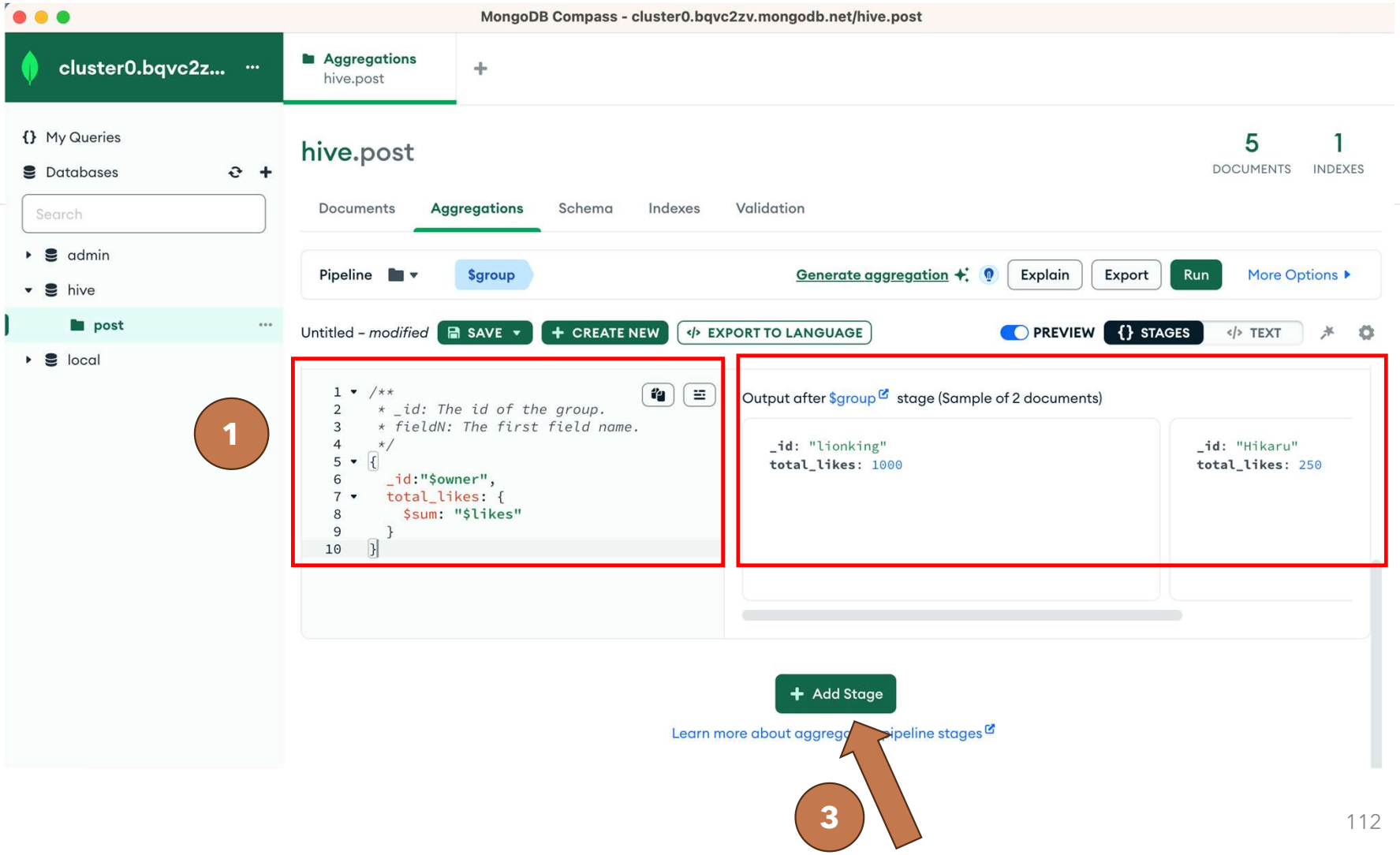

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
{owner:"lionking"}
Generate query
Explain
Reset
Find
Options
Project
{owner:1}
Sort
{ field: -1 } or [['field', -1]]
MaxTimeMS
Collation
{ locale: 'simple' }
Skip
0
EXPORT DATA
1 - 3 of 3
_id: ObjectId('652d13974ce35c5655398097')
owner: "lionking"
_id: ObjectId('652d139e4ce35c5655398098')
owner: "lionking"
_id: ObjectId('652d139e4ce35c5655398099')
owner: "lionking"
1
2
3
4

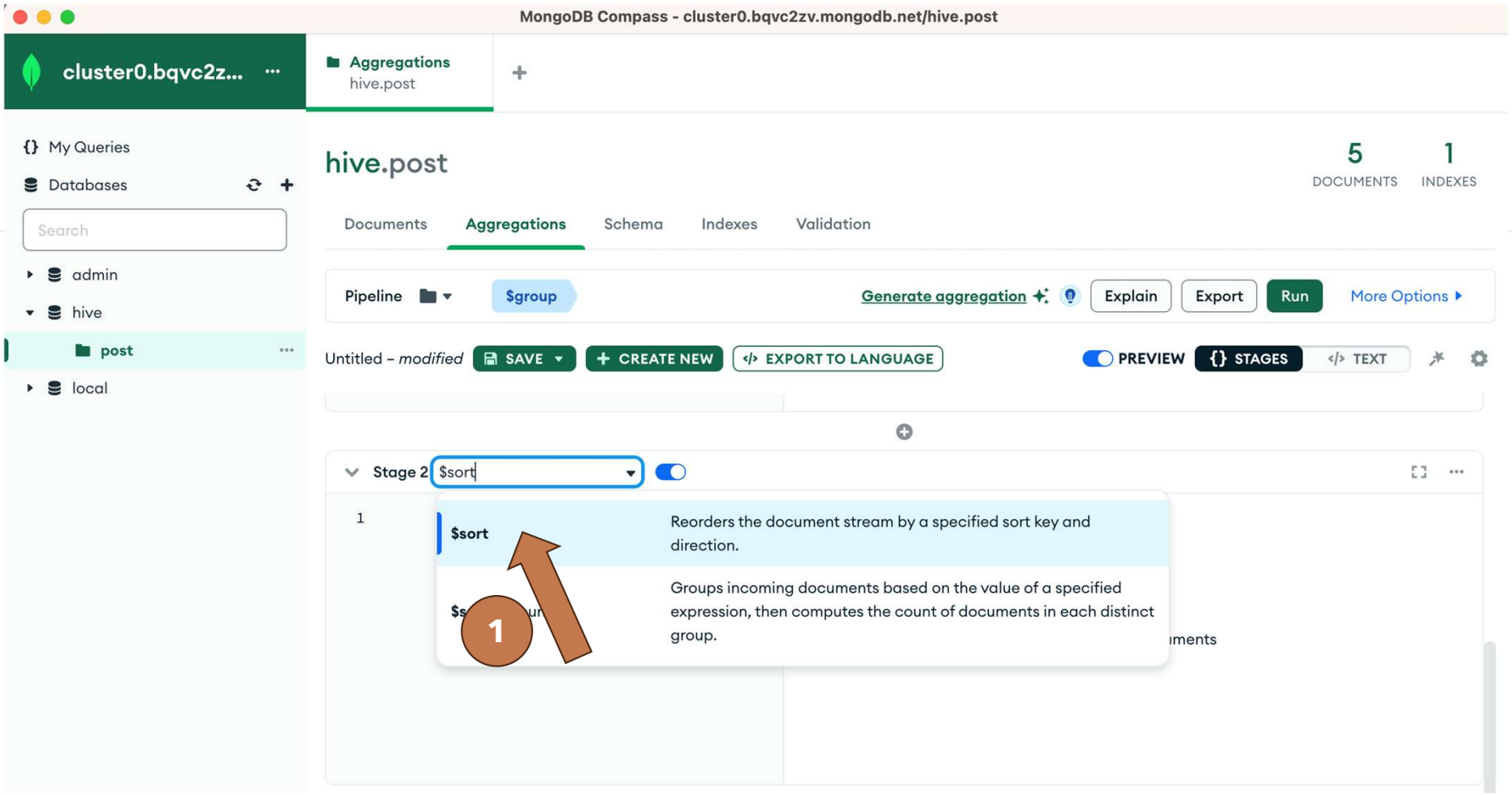

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
250
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
{owner:"lionking"}
Generate query
Explain
Reset
Find
Options
Project
{owner:1, likes:1}
1
Sort
{ field: -1 } or [['field', -1]]
Collation
{ locale: 'simple' }
Skip
0
2
EXPORT DATA
1 - 3 of 3
3
_id: ObjectId('652d13974ce35c5655398097')
owner: "lionking"
likes: 250
_id: ObjectId('652d139e4ce35c5655398098')
owner: "lionking"
likes: 250
_id: ObjectId('652d139e4ce35c5655398099')

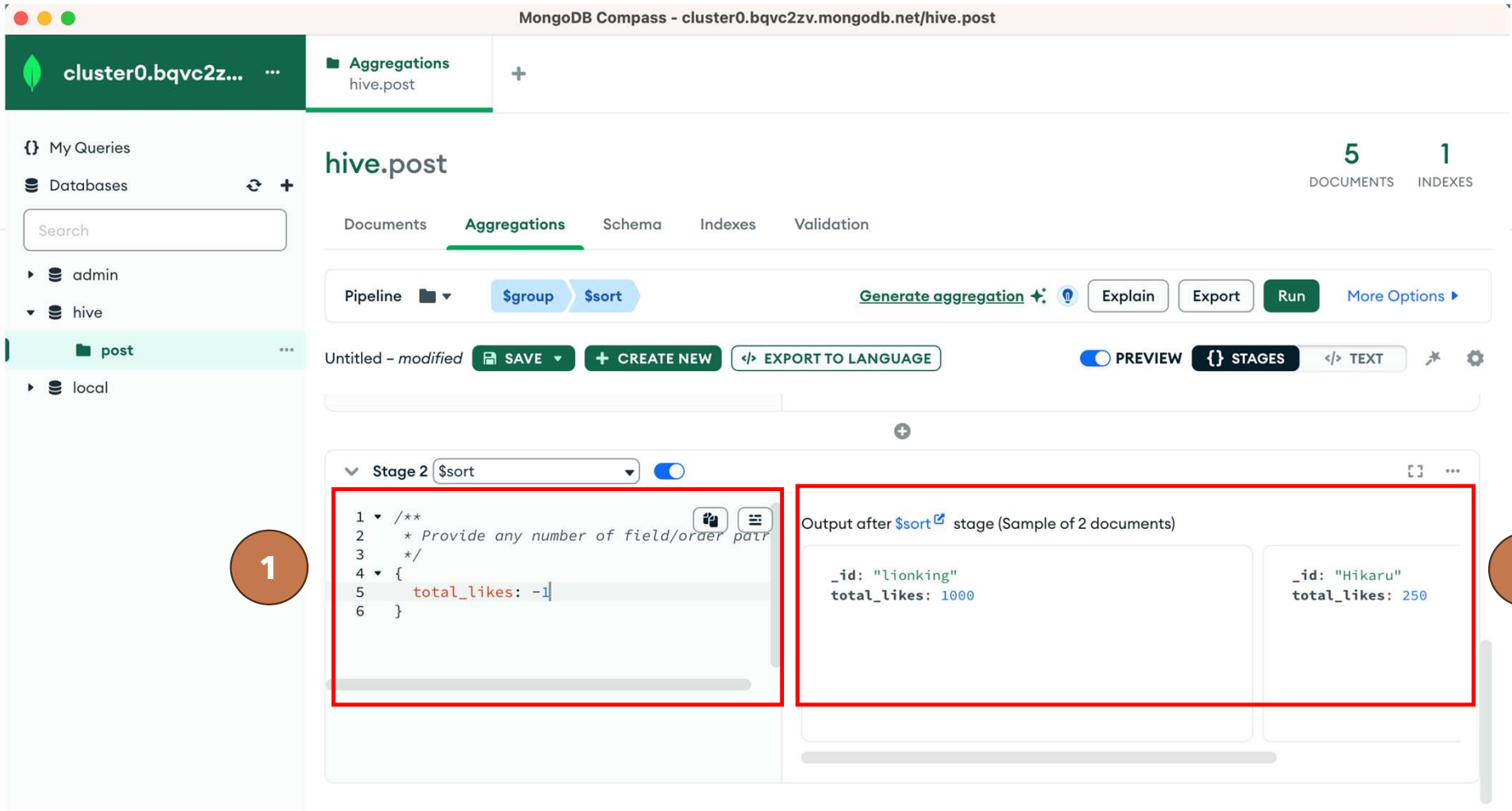

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Documents
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
{owner:"lionking"}
Generate query
Explain
Reset
Find
Options
Project
{owner:1, likes:1}
Sort
{liles:-1}
Collation
{ locale: 'simple' }
Skip
0
1
2
3
EXPORT DATA
1 – 3 of 3
_id: ObjectId('652d13974ce35c5655398097')
owner: "lionking"
likes: 250
_id: ObjectId('652d139e4ce35c5655398098')
owner: "lionking"
likes: 250
_id: ObjectId('652d139e4ce35c5655398099')
owner: "lionking"

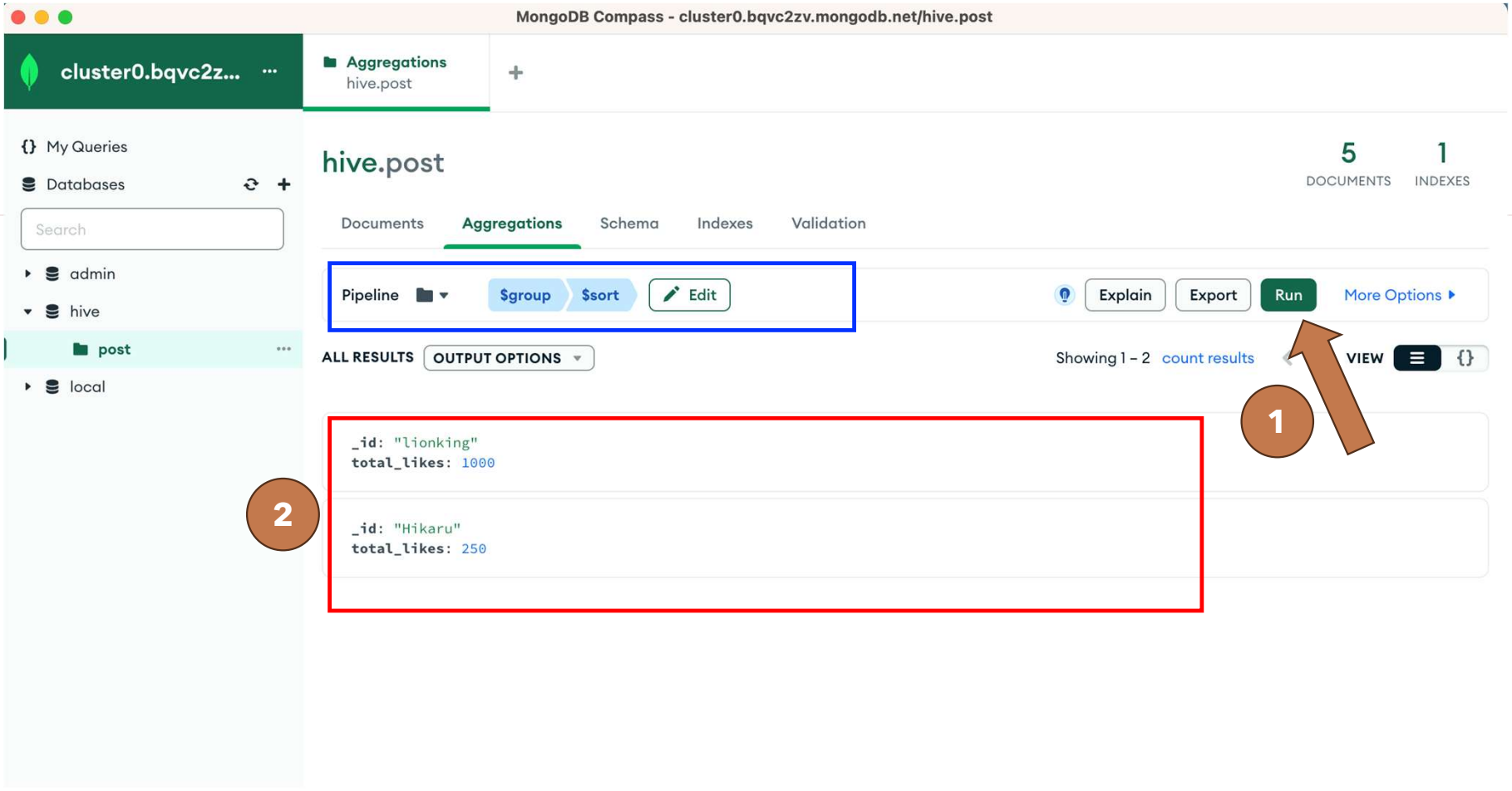

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
1
Pipeline
Your pipeline is currently empty. Generate aggregation
Explain
Export
Run
More Options
Untitled – modified
SAVE
CREATE NEW
EXPORT TO LANGUAGE
PREVIEW
STAGES
TEXT
Stage 1
Select
1
No Preview Documents
2
Add Stage


MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Pipeline
Your pipeline is currently empty. Generate aggregation
Explain
Export
Run
More Options
Untitled – modified
SAVE
CREATE NEW
EXPORT TO LANGUAGE
PREVIEW
STAGES
TEXT
Stage 1
$group
1
$group
Groups documents by a specified expression.
1
No Preview Documents
Add Stage


MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Pipeline
$group
Generate aggregation
Explain
Export
Run
More Options
Untitled – modified
SAVE
CREATE NEW
EXPORT TO LANGUAGE
PREVIEW
STAGES
TEXT
/**
* _id: The id of the group.
* fieldN: The first field name.
*/
{
_id:"$owner",
total_likes: {
$sum: "$likes"
}
}
Output after $group stage (Sample of 2 documents)
_id: "lionking"
total_likes: 1000
_id: "Hikaru"
total_likes: 250
Add Stage
Learn more about aggrega ipeline stages
1
2
3


MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Pipeline
$group
Generate aggregation
Explain
Export
Run
More Options
Untitled – modified
SAVE
CREATE NEW
EXPORT TO LANGUAGE
PREVIEW
STAGES
TEXT
Stage 2
$sort
1
$sort
Reorders the document stream by a specified sort key and direction.
Groups incoming documents based on the value of a specified expression, then computes the count of documents in each distinct group.
1


MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Pipeline
$group
$sort
Generate aggregation
Explain
Export
Run
More Options
Untitled – modified
SAVE
CREATE NEW
EXPORT TO LANGUAGE
PREVIEW
STAGES
TEXT
Stage 2
$sort
1 /**
2 * Provide any number of field/order pair
3 */
4 {
5 total_likes: -1
6 }
Output after $sort stage (Sample of 2 documents)
_id: "lionking"
total_likes: 1000
_id: "Hikaru"
total_likes: 250
1
2


MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post
cluster0.bqvc2z...
Aggregations
hive.post
My Queries
Databases
Search
admin
hive
post
local
hive.post
5
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Pipeline
$group
$sort
Edit
Explain
Export
Run
More Options
ALL RESULTS
OUTPUT OPTIONS
Showing 1 – 2
count results
VIEW
1
2
_id: "lionking"
total_likes: 1000
_id: "Hikaru"
total_likes: 250