

# MongoDB Lab

INFSCI 2711

Fan Yang

<https://github.com/yfamy123/infsci2711-lab>

# MongoDB installation



- MongoDB local: <https://www.mongodb.com/download-center#community>
  - Mac <https://treehouse.github.io/installation-guides/mac/mongo-mac.html>
  - Windows <https://www.mkyong.com/mongodb/how-to-install-mongodb-on-windows/>
- MongoDB Atlas Cluster: <https://www.mongodb.com/cloud/atlas>

P.S. If you cannot install it on local, try cluster

# MongoDB installation Steps



- Local - Mac

- Open the Terminal app and type `brew update`.
- After updating Homebrew `brew install mongodb`
- After downloading Mongo, create the “db” directory. This is where the Mongo data files will live. You can create the directory in the default location by running
  - `sudo mkdir -p /data/db`
- Make sure that the /data/db directory has the right permissions by running
  - `sudo chown -R `id -un` /data/db`
- Run the Mongodbd daemon, in one of your terminal windows run `mongod`. This should start the Mongodbd server.
- Run the Mongo shell, with the Mongo daemon running in one terminal, type `mongo` in another terminal window. This will run the Mongo shell which is an application to access data in MongoDB.
- To exit the Mongo shell run `quit()`
- To stop the Mongo daemon hit `ctrl-c`

# MongoDB installation Steps



- Cluster

- Create an Atlas User Account
- Create an Atlas Free Tier Cluster ----- Estimated completion time: 10 minutes
  - For Instance Size, select M0
  - Create an administrative Username and Password.
  - Click Deploy to deploy the cluster.
  - Add your current IP address to the Group IP Whitelist by clicking Add Current IP Address. This allows you to connect to the cluster from your current machine. Or add all IP to the whitelist.

# What is MongoDB



- Scalable High-Performance Open-source, Document-orientated database.
- Built for **Speed**.
- Rich Document based queries for **Easy readability**.
- Full Index Support for **High Performance**.
- Replication and Failover for **High Availability**.
- Auto Sharding for **Easy Scalability**.
- Map / Reduce for **Aggregation**.

# Why use MongoDB



- SQL was invented in the 70's to store **data**.
- MongoDB stores **documents (or) objects**.
- Now-a-days, everyone works with **objects** (Python/Ruby/Java/etc.)
- And we need Databases to persist our **objects**. Then why not store **objects** directly ?
- Embedded documents and arrays reduce need for joins. **No Joins** and No-multi document **transactions**.

# Terminology and Concepts



SQL Terms/Concepts	MongoDB Terms/Concepts
database	<a href="#">database</a>
table	<a href="#">collection</a>
row	<a href="#">document</a> or <a href="#">BSON document</a>
column	<a href="#">field</a>
index	<a href="#">index</a>
table joins	<a href="#">\$lookup</a> , embedded documents
primary key	<a href="#">primary key</a>
Specify any unique column or column combination as primary key.	In MongoDB, the primary key is automatically set to the <a href="#">_id</a> field.
aggregation (e.g. group by)	aggregation pipeline
	See the <a href="#">SQL to Aggregation Mapping Chart</a> .

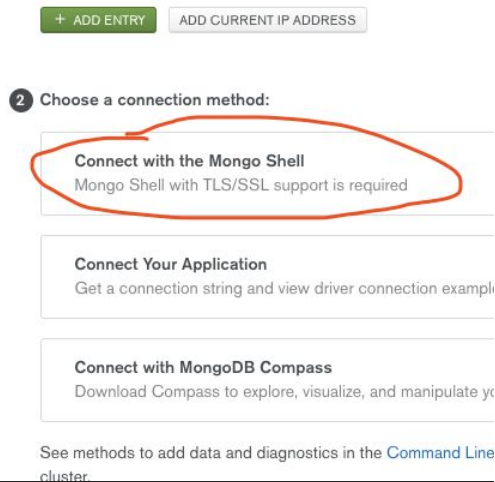
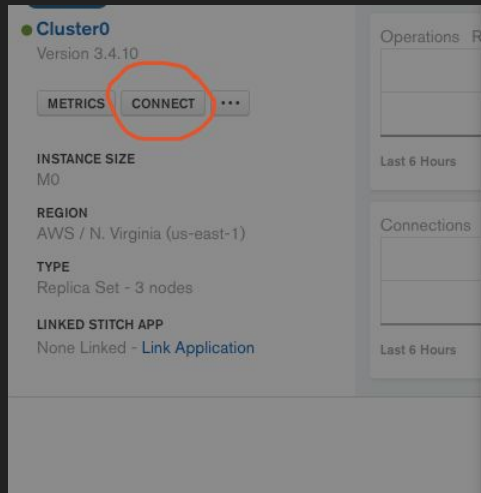
# Connect to MongoDB

- Local

- Start the Mongo server : `mongod`
- Run the Mongo shell : `mongo`

- Cluster

- 



1 If you are not running MongoDB 3.6.2+ with TLS/SSL, click below to download the latest Mongo Shell



2 Connect via the Mongo Shell

[View detailed instructions](#)

I am using shell 3.6 or later

I am using shell 3.4 or earlier

```
mongo "mongodb+srv://cluster0-fnmen.mongodb.net/test" --username yfamy123
```

[COPY](#)

You will be prompted for the password for the `yfamy123` user. Please ensure you are using a MongoDB 3.6.1+ shell with TLS/SSL.

[View your list of users or reset a password](#)



# Create database

- Check the databases list: `show dbs`
- Create a database with name "test": `use test`
- Check the current database: `db`
- Create collection: `db.createCollection('class')`
- Insert data: `db.class.insert({className: "Advanced Database", classNumber: "2711"})`
- Import data (primer-dataset.json):

```
Cluster0-shard-0:PRIMARY> show dbs
admin  0.000GB
local  0.586GB
```

```
Cluster0-shard-0:PRIMARY> use test
switched to db test
```

```
Cluster0-shard-0:PRIMARY> db
test
```

```
Cluster0-shard-0:PRIMARY> db.createCollection('class')
{ "ok" : 1 }
```

```
Cluster0-shard-0:PRIMARY> db.class.insert({className: "Advanced Database", classNumber: "2711"})
WriteResult({ "nInserted" : 1 })
```

P.S. run this command line under the folder of mongodb/.../bin

(Local) `mongoimport --db test --collection restaurants --drop --file <path>`

(Mongo cluster) `mongoimport --host <HOST> --ssl --username <USERNAME> --password <PASSWORD> --authenticationDatabase admin --db <DATABASE> --collection <COLLECTION> --file <FILEPATH>`

# Explore data

- Show one sample: `db.restaurants.findOne()`
- Show 5 samples after 10 samples:  
`db.restaurants.find().limit(5).skip(10)`
- Find samples with single constraints:
  - `SELECT * FROM restaurants WHERE name = "Wendy'S"`
    - `db.restaurants.find( { "name": "Wendy'S" } )`

```
Cluster0-shard-0:PRIMARY> db.restaurants.findOne( { "name": "Wendy'S" } )
{
  "_id" : ObjectId("5a6feb68d706d9a91f3175c2"),
  "address" : {
    "building" : "469",
    "coord" : [
      -73.961704,
      40.662942
    ],
    "street" : "Flatbush Avenue",
    "zipcode" : "11225"
  },
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "grades" : [
    {
      "date" : ISODate("2014-12-30T00:00:00Z"),
      "grade" : "A",
      "score" : 8
    },
    {
      "date" : ISODate("2014-07-01T00:00:00Z"),
      "grade" : "B",
      "score" : 23
    },
    {
      "date" : ISODate("2013-04-30T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2012-05-08T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    }
  ],
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
```

- `SELECT restaurant_id FROM restaurants`  
`WHERE name = "Wendy'S"`

- `db.restaurants.find( { "name":  
"Wendy'S" } , {"restaurant_id":1, _id:0  
})`

```
Cluster0-shard-0:PRIMARY> db.res  
{ "restaurant_id" : "30112340" }  
{ "restaurant_id" : "40386062" }  
{ "restaurant_id" : "40388016" }  
{ "restaurant_id" : "40529991" }  
{ "restaurant_id" : "40568945" }  
{ "restaurant_id" : "40568949" }  
{ "restaurant_id" : "40568953" }
```

- Nested field constraint

- `db.restaurants.find( {  
"address.zipcode": "10305" } )`

```
{ "_id" : ObjectId("5a6feb68d706d9a91f3175f2"), "address" : { "b  
uilding" : "1111", "coord" : [ -74.0796436, 40.598783399999999 ],  
"street" : "Hylan Boulevard", "zipcode" : "10305" }, "borough"  
: "Staten Island", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices"  
, "grades" : [ { "date" : ISODate("2014-04-24T00:00:00Z"), "grad  
e" : "A", "score" : 12 }, { "date" : ISODate("2013-02-26T00:00:0  
0Z"), "grade" : "A", "score" : 5 }, { "date" : ISODate("2012-02-  
02T00:00:00Z"), "grade" : "A", "score" : 2 } ], "name" : "Carvel  
Ice Cream", "restaurant_id" : "40363834" }
```

# Explore data

- Find samples with complex constraints:
  - SELECT \* FROM restaurants WHERE  
name = "Wendy'S" AND zipcode =  
"10305"
    - `db.restaurants.find( { $and: [`  
 `{"name": "Wendy'S" },`  
 `{"address.zipcode": "10305"} ] })`

```
Cluster0-shard-0:PRIMARY> db.restaurants.find( { $and: [ { "name": "Wendy'S" }, { "address.zipcode": "10305" } ] })
{ "_id" : ObjectId("5a6feb6cd706d9a91f318999"), "address" : { "building" : "1661", "coord" : [ -74.09107, 40.5882746 ], "street" : "Hylan Blvd", "zipcode" : "10305" }, "borough" : "Staten Island", "cuisine" : "Hamburgers", "grades" : [ { "date" : ISODate("2014-12-04T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2014-06-24T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-09-06T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-01-16T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-04-19T00:00:00Z"), "grade" : "B", "score" : 15 } ], "name" : "Wendy'S", "restaurant_id" : "41033173" }
```

- SELECT \* FROM restaurants

WHERE name = "Wendy'S" OR  
zipcode = "10305"

- `db.restaurants.find( { $or: [`  
    `{"name": "Wendy'S" },`  
    `{"address.zipcode": "10305"} ]`  
    `})`

```
Cluster0-shard-0:PRIMARY> db.restaurants.find( { $or: [ {"name" : "Wendy'S" }, {"address.zipcode": "10305"} ] }).limit(1)
{ "_id" : ObjectId("5a6feb68d706d9a91f3175c2"), "address" : { "building" : "469", "coord" : [ -73.961704, 40.662942 ], "street" : "Flatbush Avenue", "zipcode" : "11225" }, "borough" : "Brooklyn", "cuisine" : "Hamburgers", "grades" : [ { "date" : ISODate("2014-12-30T00:00:00Z"), "grade" : "A", "score" : 8 }, { "date" : ISODate("2014-07-01T00:00:00Z"), "grade" : "B", "score" : 23 }, { "date" : ISODate("2013-04-30T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2012-05-08T00:00:00Z"), "grade" : "A", "score" : 12 } ], "name" : "Wendy'S", "restaurant_id" : "30112340" }
```

- SELECT \* FROM restaurants WHERE name LIKE "%Burger%"

- db.restaurants.find( { "name": /Burger/ } )
- db.restaurants.find( { "name": { \$regex: /Burger/ } } )
- "Burger%" —— /^Burger/
- "%Burger" —— /\$Burger/
- <https://docs.mongodb.com/manual/reference/operator/query/regex/>

```
Cluster0-shard-0:PRIMARY> db.restaurants.find( { "name": /Burger/ } ).limit(1)
{ "_id" : ObjectId("5a6feb68d706d9a91f3176ea"), "address" : { "building" : "22210", "coord" : [ -73.759249, 40.761574 ], "street" : "Northern Boulevard", "zipcode" : "11361" }, "borough" : "Queens", "cuisine" : "Hamburgers", "grades" : [ { "date" : ISODate("2014-12-18T00:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2013-12-11T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2013-07-22T00:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2013-03-01T00:00:00Z"), "grade" : "C", "score" : 10 }, { "date" : ISODate("2012-02-06T00:00:00Z"), "grade" : "A", "score" : 10 } ], "name" : "Burger King", "restaurant_id" : "40370167" }
```

# Explore data

- Find samples with complex constraints:
  - `SELECT * FROM restaurants WHERE name = "Wendy'S" ORDER BY restaurant_id DESC`
    - `db.restaurants.find( { "name": "Wendy'S" } ).sort( { restaurant_id: -1 } )`
  - `SELECT COUNT(*) FROM restaurants`
    - `db.restaurants.count() / db.restaurants.find().count()`

```
Cluster0-shard-0:PRIMARY> db.restaurants.find( { "name": "Wendy'S" } ).sort( { restaurant_id: -1 } ).limit(1)
{ "_id" : ObjectId("5a6feb7bd706d9a91f31d4cc"), "address" : { "building" : "", "coord" : [ -73.77813909999999, 40.6413111 ], "street" : "Jfk Airport", "zipcode" : "11430" }, "borough" : "Queens", "cuisine" : "Hamburgers", "grades" : [ { "date" : ISODate("2014-11-26T00:00:00Z"), "grade" : "A", "score" : 10 } ], "name" : "Wendy'S", "restaurant_id" : "50012552" }
```

```
Cluster0-shard-0:PRIMARY> db.restaurants.count()
25359
```



- SELECT DISTINCT(name) from restaurants

- db.restaurants.distinct("name")

```
"187Th St Pizza",  
"Loi Estiatorio",  
"Mahalo New York Bakery",  
"La Pollera Colorado 3",  
"Northside Bakery",  
"Christine'S Restaurant",  
"Superwings & Things",  
"Mokja",  
"Fairfield Inn Suites Penn Station",  
"Indian Oven",  
"Cold Press'D"
```

- SELECT \* FROM restaurants WHERE

grades.score > 40

- db.restaurants.find({"grades.score" :  
{\$gt:40}})

- > : \$gt                      < : \$lt                      >= : \$gte  
                                 <= : \$lte

```
Cluster0-shard-0:PRIMARY> db.restaurants.find({"grades.score" : {$gt:40}}).limit(1)  
{ "_id" : ObjectId("5a6feb68d706d9a91f3175cc"), "address" : { "building" : "1269", "coord" : [ -73.871194, 40.6730975 ], "street" : "Sutter Avenue", "zipcode" : "11208" }, "borough" : "Brooklyn", "cuisine" : "Chinese", "grades" : [ { "date" : ISODate("2014-09-16T00:00:00Z"), "grade" : "B", "score" : 21 }, { "date" : ISODate("2013-08-28T00:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2013-04-02T00:00:00Z"), "grade" : "C", "score" : 56 }, { "date" : ISODate("2012-08-15T00:00:00Z"), "grade" : "B", "score" : 27 }, { "date" : ISODate("2012-03-28T00:00:00Z"), "grade" : "B", "score" : 27 } ], "name" : "May May Kitchen", "restaurant_id" : "40358429" }
```

- Other Aggregation

<https://docs.mongodb.com/getting-started/shell/aggregation/>



# Fancy Functionality

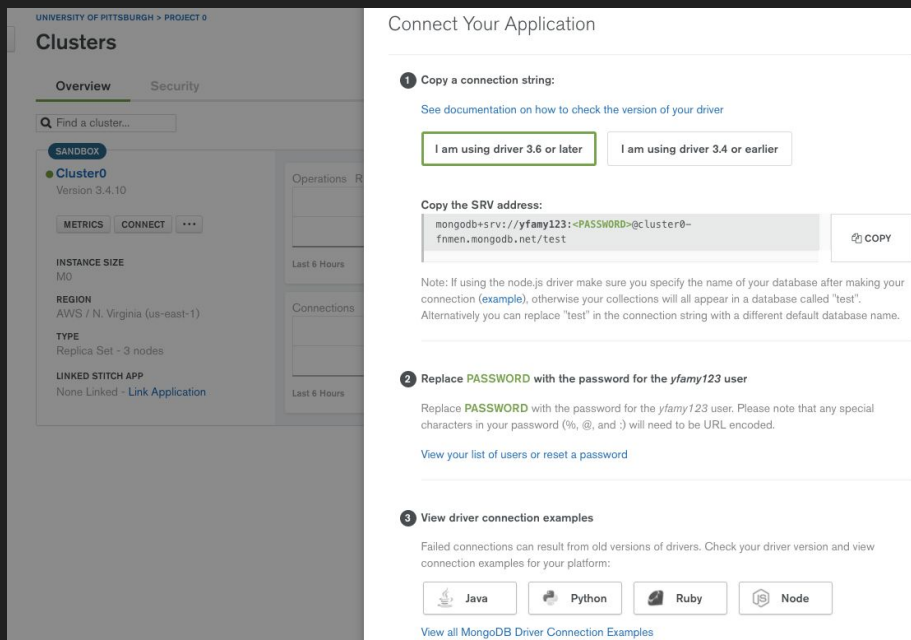
- Text search:
  - Create text index (A collection can only have one text search index, but that index can cover multiple fields.)
    - `db.restaurants.createIndex( { "name": "text", "address.street": "text", "borough": "text", "cuisine": "text" } )`
  - Search pattern within the text index
    - Containing any terms: `db.restaurants.find( { $text: { $search: "Brooklyn Ham" } } )`
    - Exact search: `db.restaurants.find( { $text: { $search: "\"Brooklyn Ham\"" } } )`
    - Exclude search: `db.restaurants.find( { $text: { $search: "Brooklyn -Ham" } } )`
  - Text score (compute a relevance score for each document that specifies how well a document matches the query)
    - `db.restaurants.find({ $text: { $search: "Brooklyn Ham" } }, { textScore: { $meta: "textScore" } })`

# Fancy Functionality

- Geospatial:
  - Create a 2dsphere index: `db.restaurants.createIndex( { "address.coord" : "2dsphere" } )`
  - Find samples near one location:
    - `db.restaurants.find({"address.coord":{" $near":{"$geometry: {coordinates: [ -73.9667, 40.78 ]`  
`}}, $minDistance: 1000, $maxDistance: 5000}}))`
    - `db.restaurants.find( { "address.coord": { $geoWithin: { $centerSphere: [ [`  
`-73.96252129999999, 40.7098035 ], 5 / 3963.2 ] } } } )`

# Linking with Mongdb

- Local
- Cluster <https://docs.atlas.mongodb.com/driver-connection/#php-driver-example>



The image shows a screenshot of the MongoDB Atlas interface. On the left, the 'Clusters' page is visible, showing a cluster named 'Cluster0' with version 3.4.10. The 'Overview' tab is selected, and the 'CONNECT' button is highlighted. On the right, the 'Connect Your Application' dialog is open, guiding the user through the connection process.

**Connect Your Application**

- Copy a connection string:**  
See documentation on how to check the version of your driver  
☒ I am using driver 3.6 or later ☐ I am using driver 3.4 or earlier  
**Copy the SRV address:**  
`mongodb+srv://yfamy123:<PASSWORD>@cluster0-fnmen.mongodb.net/test` [COPY](#)  
Note: If using the node.js driver make sure you specify the name of your database after making your connection (example), otherwise your collections will all appear in a database called "test". Alternatively you can replace "test" in the connection string with a different default database name.
- Replace **PASSWORD** with the password for the yfamy123 user**  
Replace **PASSWORD** with the password for the yfamy123 user. Please note that any special characters in your password (% , @ , and .) will need to be URL encoded.  
[View your list of users or reset a password](#)
- View driver connection examples**  
Failed connections can result from old versions of drivers. Check your driver version and view connection examples for your platform:  

[Java](#) [Python](#) [Ruby](#) [Node](#)

  
[View all MongoDB Driver Connection Examples](#)

# Linking with Mongodb

- Python
  - `pip install pymongo`
  - Start mongo server: `mongod`
- Terminal: `python testMongodb.py`

<https://api.mongodb.com/python/current/>

```
from pymongo import MongoClient, errors

try:
    '''Connect with local mongodb'''
    # MongoClient = MongoClient('localhost',27017)

    '''Connect with mongodb cluster'''
    MongoClient = MongoClient("mongodb://yfamy123:<passw

    print("Connect successfully!")
except errors.ConnectionFailure:
    print("Could not connect to MongoDB")

print("Show database", MongoClient.database_names())

'''Change database'''
database = MongoClient.test

'''Get collection name'''
collection = database.collection_names(include_system_co

for collect in collection:
    print(collect)

'''Get collection restaurants'''
restaurants = database.restaurants

'''Print one sample'''
print(restaurants.find_one())

'''Print one sample with constrain'''
print(restaurants.find_one({ "name": "Wendy'S" })))
```

# Linking with Mongodb

- Java

- Download .jar file

<https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver>

- Examples

<https://www.mkyong.com/mongodb/java-mongodb-query-document/>

- API

<http://mongodb.github.io/mongo-java-driver/3.6/javadoc/>

```
import com.mongodb.*;

import java.util.List;
import java.util.Set;

public class testMongodb {

    public static void main(String[] args) {

        try {

            /* Connect to Cluster */
            MongoClientURI uri = new MongoClientURI(
                "mongodb+srv://yfamy123:amy920420@cluster0-fnmen.mongodb.net/test");
            MongoClient mongoClient = new MongoClient(uri);

            /* Connect to Local */
            MongoClient mongoClient = new MongoClient();

            System.out.println("Connect successfully!");

            List<String> databaseNames = mongoClient.getDatabaseNames();

            System.out.println("Database list");
            for(String db : databaseNames) {
                System.out.println(db);
            }

            DB database = mongoClient.getDB( dbName: "test");

            Set<String> collectionNames = database.getCollectionNames();

            System.out.println("Collection list in database test");
            for(String collection : collectionNames) {
                System.out.println(collection);
            }

            DBCollection restaurant = database.getCollection( name: "restaurants");
```