

# MongoDB

## 1. Install MongoDB

---

```
sudo apt update -y
sudo apt upgrade -y
sudo apt install -y mongodb
sudo systemctl status mongod
sudo vi /etc/mongodb.conf
# bind_ip = 0.0.0.0
# auth = true
# 비밀번호 설정
$ mongo
> use admin
> db.createUser({ user: "rada", pwd: "radapw", roles: [ "root" ] })
> quit()
$ sudo systemctl restart mongod
# aws 인스턴스 27017 포트 접속 허용
```

---

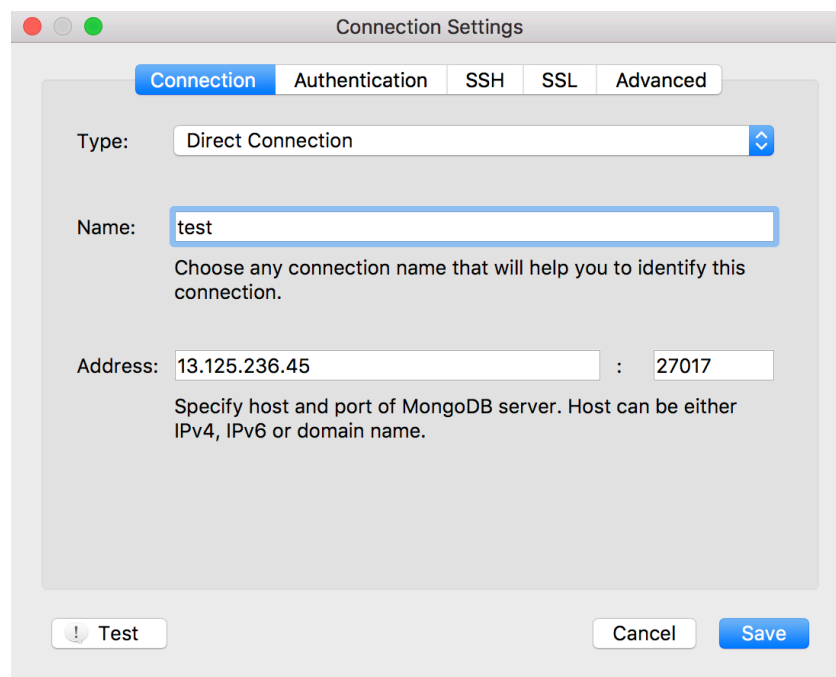
mongo 명령어를 입력하면 mongo shell에 접속됩니다.

## 2. Install Robomongo

<https://robomongo.org/> 페이지에서 경로에서 ROBO 3T 다운로드 후 설치 합니다.

## 3. Connection

MongoDB Connections에서 ip를 입력하여 서버의 mongoDB에 접속합니다.



The image shows the 'Connection Settings' dialog box with the 'Connection' tab selected. The 'Type' is set to 'Direct Connection'. The 'Name' is 'test'. The 'Address' is '13.125.236.45 : 27017'. There are 'Test', 'Cancel', and 'Save' buttons at the bottom.

Connection Settings

Connection Authentication SSH SSL Advanced

Type: Direct Connection

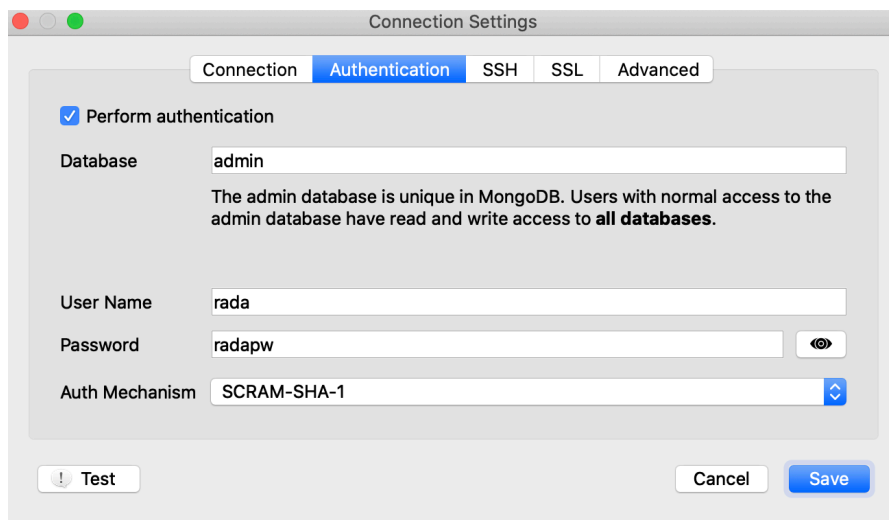
Name: test

Choose any connection name that will help you to identify this connection.

Address: 13.125.236.45 : 27017

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

! Test Cancel Save



The image shows the 'Connection Settings' dialog box with the 'Authentication' tab selected. The 'Perform authentication' checkbox is checked. The 'Database' is 'admin'. The 'User Name' is 'rada'. The 'Password' is 'radapw'. The 'Auth Mechanism' is 'SCRAM-SHA-1'. There are 'Test', 'Cancel', and 'Save' buttons at the bottom.

Connection Settings

Connection Authentication SSH SSL Advanced

☒ Perform authentication

Database: admin

The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to **all databases**.

User Name: rada

Password: radapw

Auth Mechanism: SCRAM-SHA-1

! Test Cancel Save

## 4. Basic Syntax

<https://docs.mongodb.com/v3.6/reference/>

### 4.1 Create Database

# mongo 라는 이름의 데이터 베이스 생성

```
use mongo
```

# 현재 사용중인 데이터 베이스 확인

```
db
```

# database list 확인

```
show dbs
```

- 데이터 베이스를 생성후에 최소 1개이상의 document를 추가해야 생성된 데이터 베이스가 보입니다.

# document 생성

```
use mongo
```

```
db.user.insert({"name":"alice", "age":20, "email":"alice@gmail.com"})
```

### 4.2 Delete Database

# 현재 사용중인 데이터 베이스 삭제

```
db.dropDatabase()
```

### 4.3 Create Collection

reference

- <https://docs.mongodb.com/v3.6/reference/method/db.createCollection/>

# name : collection 이름

`db.createCollection(name, [option])`

option

`capped` : true로 설정하면 collection의 최대 용량을 설정 (최대 용량의 크기는 `size` 옵션으로 설정), 설정된 최대용량 이상으로 데이터가 입력되면 오래된 데이터 부터 자동으로 삭제됩니다.

`autoIndex` : true로 설정하면 `_id` 필드에 index가 자동으로 생성됩니다.

`size` : 숫자 데이터를 사용하며 collection의 최대 사이즈를 byte 단위로 지정

`max` : 숫자 데이터를 사용하며 최대 document 갯수를 설정

# user 컬렉션을 생성

`db.createCollection("user")`

# `autoIndex`와 `max` 옵션을 설정하여 info 컬렉션을 생성

`db.createCollection("info1", { autoIndexId: true, capped: true, size: 500, max:5 })`

`db.createCollection("info2", { autoIndexId: true, capped: true, size: 50, max:5 })`

# `createCollection`을 사용하지 않고 article 컬렉션을 생성

`db.articles.insert( { "title":"data science", "contents":"mongodb" } )`

# 컬렉션 리스트 확인

`show collections`

## 4.4 Delete Collection

```
# articles 컬렉션 삭제  
db.articles.drop()
```

## 4.5 Make Document

```
db.<collection_name>.insert(<document>)
```

```
# info 컬렉션에 document 추가  
db.info1.insert({ "subject":"python", "level":3 })  
db.info1.insert({ "subject":"web", "level":1 })  
db.info1.insert({ "subject":"sql", "level":2 })
```

```
# 한번에 여러개의 document 추가  
# max:5 옵션 제한에 걸려 5개의 데이터가 info1에 들어간다.
```

```
db.info1.insert( [  
    { "subject":"python", "level":3 },  
    { "subject":"web", "level":1 },  
    { "subject":"sql", "level":2 },  
    { "subject":"python", "level":3 },  
    { "subject":"web", "level":1 },  
    { "subject":"sql", "level":2 },  
)
```

# size:50 옵션 제한에 걸려 4개의 데이터가 info2에 입력된다.

```
db.info2.insert( [  
    { "subject":"python", "level":3 },  
    { "subject":"web", "level":1 },  
    { "subject":"sql", "level":2 },  
    { "subject":"python", "level":3 },  
    { "subject":"web", "level":1 },  
])
```

```
db.info.insert( [  
    { "subject":"python", "level":3 },  
    { "subject":"web", "level":1 },  
    { "subject":"sql", "level":2 },  
    { "subject":"java", "level":3 },  
    { "subject":"html", "level":1 },  
    { "subject":"css", "level":2 },  
])
```

## 4.6 Delete Document

# level2인 데이터 삭제 : 제약조건이 걸려있는 컬렉션의 도큐먼트는 삭제가 안됩니다.

```
db.info.remove( {level:2} )
```

## 4.7 Find

reference

<https://docs.mongodb.com/manual/reference/method/db.collection.find/index.html>

format

`db.collection.find(query, projection)`

query : document 조회 조건을 설정. 모든 document를 조회 할때는 ({} )를 사용

projection : document를 조회할때 보여지는 필드(컬럼)를 정의

### 4.7.1 query

#### 4.7.1.1 기본 document 조회

# info 컬렉션에 있는 모든 document 조회

```
db.info.find()
```

```
db.getCollection('info').find({})
```

# subject가 python인 document 조회

```
db.info.find({"subject": "python"})
```

#### 4.7.1.2 비교 연산자

reference

- <https://docs.mongodb.com/v3.6/reference/operator/query/>

# level이 2 이하인 document를 조회

```
db.info.find({"level": {$lte: 2}})
```

# level이 3 이상인 document를 조회

```
db.info.find({"level": {$gte: 3}})
```

# subject가 java와 python을 포함하는 document 조회

```
db.info.find( {"subject": {$in: ["java", "python"]}} )
```

#### 4.7.1.3 논리 연산자

\$or : 조건중 하나라도 true이면 true

\$and : 모든 조건이 true이면 true

\$not : 조건중 하나라도 false이면 true

\$nor : 모든 조건이 false이면 true (or와 반대 개념)

# subject가 python이고 level이 3이상인 document 조회

```
db.info.find( { $and: [ { "subject": "python" }, { "level": {$gte: 3} } ] } )
```

# subject가 python이아니고 level이 1이하가 아닌 document 조회

```
db.info.find( { $nor: [ { "subject": "python" }, { "level": {$lte: 1} } ] } )
```



# level이 2보다 크지 않은 document 조회 (2 포함)

```
db.info.find({ "level": { $not: { $gt: 2 } } })
```

#### 4.7.1.4 \$where

\$where 연산자를 사용하면 자바스크립트 표현식 사용이 가능합니다.

# level이 1인 document 조회

```
db.info.find( { $where: "this.level == 1" } )
```

### 4.7.2 projection

document를 조회할때 보여지는 필드(컬럼)를 정의합니다.

#### 4.7.2.1 Basic

# subject와 comments만 출력되도록 find

# 설정을 true 값을 설정하던가 false 값을 설정합니다. ( \_id는 따로 설정을 안하면 true )

```
db.info.find({}, {"_id":false, "level":false})
```

```
db.info.find({}, {"subject":true, "level":true})
```

```
db.info.find({}, {"_id":false, "subject":true, "level":true})
```

### 4.8 Find Method

find method를 사용하면 find를 사용한 document의 결과를 가공하여 출력할수 있습니다.

### 4.8.1 sort

document를 정렬시켜 줍니다.

'sort({key: value})' 와 같은 포맷으로 사용을 하며 key는 정렬할 필드명을 작성하고, value는 오름차순은 1, 내림차순을 -1을 넣어주면 됩니다.

# info 컬렉션의 document를 level 오름차순으로 정렬

```
db.info.find().sort({"level":1})
```

# info 컬렉션의 document를 level 내림차순으로 정렬

```
db.info.find().sort({"level":-1})
```

# level을 기준으로 내림차순으로 정렬한 후 subject를 기준으로 오름차순으로 정렬

```
db.info.find().sort({"level":-1, "subject":1})
```

### 4.8.2 limit

limit을 사용하면 document출력 결과의 수를 제한할수 있습니다.

# document의 결과를 3개 까지만 출력

```
db.info.find().limit(3)
```

# document의 결과를 level로 내림차순으로 정렬하고 3개까지만 출력

```
db.info.find().sort({"level":-1}).limit(3)
```

### 4.8.3 skip

skip을 검색한 document의 결과의 시작부분을 설정할때 사용합니다.

# document를 3번째 부터 출력

```
db.info.find().skip(2)
```

\* limit, skip을 함께 사용해서 mysql의 limit과 같이 사용할수 있습니다.

### 4.9 update

reference

- <https://docs.mongodb.com/manual/reference/command/update/index.html>

```
db.collection.update( query, update, { upsert: <bool>, multi: <bool> } )
```

upsert : insert와 update의 합성어 (데이터가 있으면 update, 없으면 insert 한다는 의미)

multi : true로 설정되면 여러개의 document를 수정합니다. 기본값은 false

# 특정 document를 새로운 document로 수정하기

```
db.info.update(  
    { "subject": "html" },  
    { "subject": "sass", "level": 2 }  
)
```

```
db.info.update(  
    { "subject": "less" },  
    { "subject": "less", "level": 2 },
```

```
        { "upsert": true }  
    )
```

#### 4.9.1 \$set, \$unset

\$set을 사용하면 특정 document의 필드를 수정할수 있습니다.

\$unset를 사용하면 특정 document의 필드 제거할수 있습니다.

# python의 level을 3으로 수정 (한개의 데이터만 수정)

```
db.info.update( { subject: "java" }, { $set: { level: 4 } } )
```

# level 2를 level 1로 수정 (여러개의 데이터 수정)

```
db.info.update(  
    { level: 2 },  
    { $set: { level: 1 } },  
    { multi: true }  
)
```

# subject가 sass인 document의 level필드 삭제

```
db.info.update(  
    { subject: "sass" },  
    { $unset: {level: 1} }  
)
```

\* level: 1의 1은 true를 의미합니다.

# level이 20이하인 데이터를 1로 수정하기

```
db.info.update(  
    { level: { $lte: 2 } },
```

```
    { $set: {level: 1} },  
    { multi: 1 }  
)  
  
# level이 없는 데이터 level 추가하기  
db.info.update(  
    { level: {$exists: false} },  
    { $set: {level: 2} },  
    { multi: 1 }  
)
```

#### 4.10 Function

자바스크립트 문법으로 함수 작성이 가능합니다.

```
# skip 함수  
var showSkip = function(start){  
    return db.info.find().skip(start)  
}  
showSkip(3)
```

## 5. Pymongo

파이썬 코드를 사용하여 Mongodb에 데이터를 저장하는 패키지

# zigbang.py 모듈 파일을 서버로 이동

# 코드에서 Mongodb 접속정보로 변경

# python3 설치

\$ sudo apt install python3 python3-pip

# python 패키지 설치

\$ pip3 install requests geohash2 pandas pymongo

# python 코드 실행

\$ python3 zigbang.py 망원동

# 망원동 원룸정보가 크롤링되고 크롤링 된 정보를 Mongodb에 저장