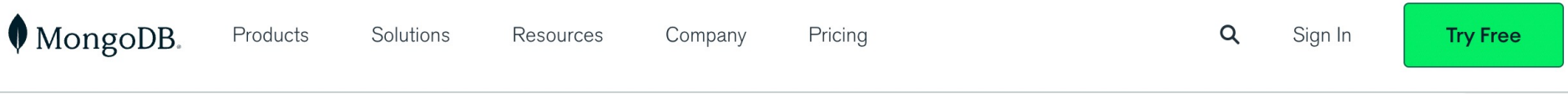
MongoDB Understanding

Dr. A. Suresh, INSOFE

Introduction



What Is MongoDB?

MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need

Introduction

- Open Source
- Document DB
- NoSQL
- Key value Pair
- High Performance
- High Available
- Highly Scalable
- Flexible

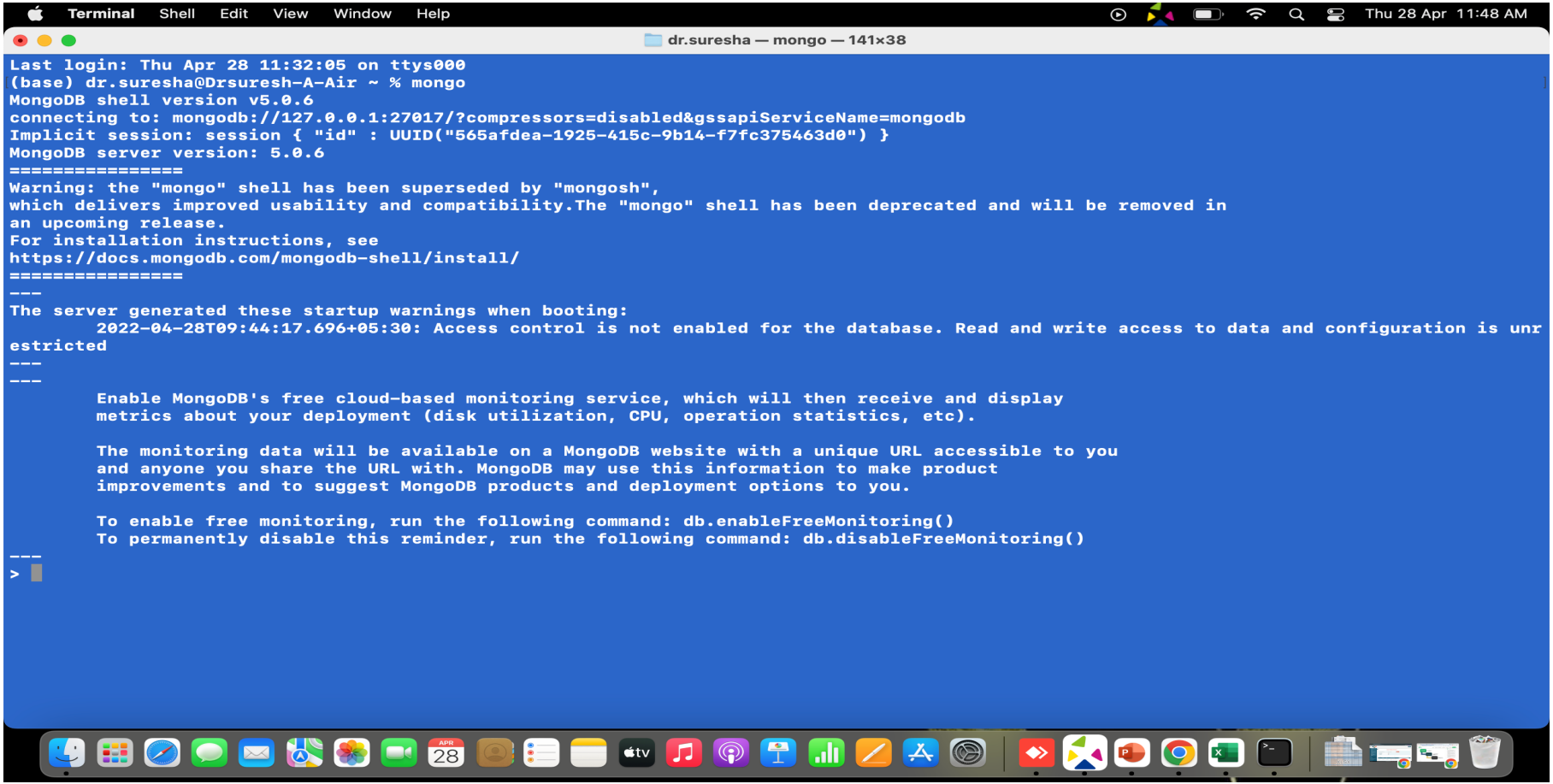
MYSQL	MONGODB
Database	Database
Tables	Collections
Tuples	Document
Column	Fields
Primary key	_id is the default primary key generated

MongoDB Hierarchy

Agenda

Mongo Installation
Create Database
Drop Database
Create Collection
Drop collection
Data types
Insert Document to Collection
Query Document

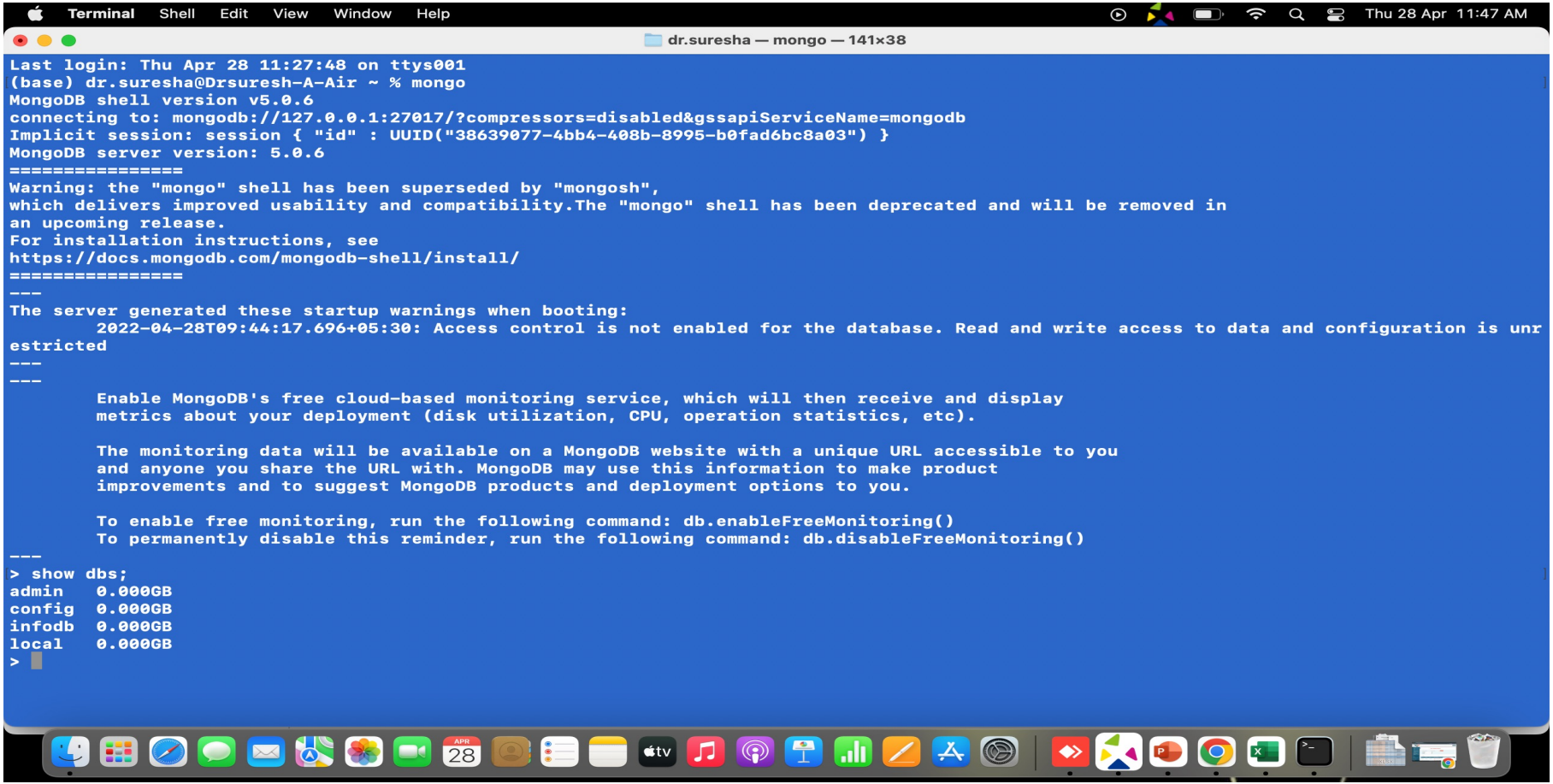
Update Document
Delete Document
Indexing
Aggregation
Replication
Sharding
Deployment


Terminal Shell Edit View Window Help
Thu 28 Apr 11:48 AM
dr.suresha — mongo — 141×38
Last login: Thu Apr 28 11:32:05 on ttys000
(base) dr.suresha@Drsuresh-A-Air ~ % mongo
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("565afdea-1925-415c-9b14-f7fc375463d0") }
MongoDB server version: 5.0.6
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================

The server generated these startup warnings when booting:
2022-04-28T09:44:17.696+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unr
estricted

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

>


Terminal Shell Edit View Window Help
Thu 28 Apr 11:47 AM
dr.suresha — mongo — 141×38
Last login: Thu Apr 28 11:27:48 on ttys001
(base) dr.suresha@Drsuresh-A-Air ~ % mongo
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("38639077-4bb4-408b-8995-b0fad6bc8a03") }
MongoDB server version: 5.0.6
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
The server generated these startup warnings when booting:
2022-04-28T09:44:17.696+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unr
estricted
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
> show dbs;
admin 0.000GB
config 0.000GB
infodb 0.000GB
local 0.000GB
>

```
> show dbs;
admin   0.000GB
config  0.000GB
infodb  0.000GB
local   0.000GB
> use infodb;
switched to db infodb
> show collections;
emp
first
first1
infodb1
> db.first.find()
> db.first1.find();
> db.infodb1.find();
{ "_id" : ObjectId("62678729e4949926d32da30a"), "name" : "admin" }
```

```
> use insofedb;
switched to db insofedb
> show dbs;
admin    0.000GB
config   0.000GB
infodb   0.000GB
local    0.000GB
> db.createCollections('emp1');
uncaught exception: TypeError: db.createCollections is not a function :
@(shell):1:1
> db.createCollection('emp1');
{ "ok" : 1 }
> show dbs;
admin      0.000GB
config     0.000GB
infodb     0.000GB
insofedb   0.000GB
local      0.000GB
>
```

```
> db.emp2.insert({empid:101,name:'aaa'})
WriteResult({ "nInserted" : 1 })
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
> show dbs
admin     0.000GB
config    0.000GB
infodb    0.000GB
insofedb  0.000GB
local     0.000GB
> show collections;
emp1
emp2
> db.emp1.find()
>
```

```
> db.createcollection('emp3',{capped:true,size:1024000,max:5})
uncaught exception: TypeError: db.createcollection is not a function :
@(shell):1:1
> db.createCollection('emp3',{capped:true,size:1024000,max:5})
{ "ok" : 1 }
> show collections;
emp1
emp2
emp3
> db.emp3.insert({empid:101,name:'aaa'})
WriteResult({ "nInserted" : 1 })
> db.emp3.find()
{ "_id" : ObjectId("626afd2f8353db8626310e59"), "empid" : 101, "name" : "aaa" }
>
```

```
> db.createCollection('emp4',{capped:true,size:5242880,max:3})
{ "ok" : 1 }
> db.emp4.insert({empid:101,name:'aaa'})
WriteResult({ "nInserted" : 1 })
> db.emp4.find()
{ "_id" : ObjectId("626b03988353db8626310e5c"), "empid" : 101, "name" : "aaa" }
> db.emp4.insert({empid:102,name:'bbb'})
WriteResult({ "nInserted" : 1 })
> db.emp4.find()
{ "_id" : ObjectId("626b03988353db8626310e5c"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b03b88353db8626310e5d"), "empid" : 102, "name" : "bbb" }
> db.emp4.insert({empid:102,name:'bbb'})
WriteResult({ "nInserted" : 1 })
```

```
> db.emp4.find()
{ "_id" : ObjectId("626b03988353db8626310e5c"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b03b88353db8626310e5d"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b03c68353db8626310e5e"), "empid" : 102, "name" : "bbb" }
> db.emp4.insert({empid:103,name:'ccc'})
WriteResult({ "nInserted" : 1 })
> db.emp4.find()
{ "_id" : ObjectId("626b03b88353db8626310e5d"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b03c68353db8626310e5e"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b03da8353db8626310e5f"), "empid" : 103, "name" : "ccc" }
> db.emp4.totalSize()
73728
> db.emp4.dataSize()
153
```

```
> db.emp4.stats()
{
        "ns" : "insofedb.emp4",
        "size" : 153,
        "count" : 3,
        "avgObjSize" : 51,
        "storageSize" : 36864,
        "freeStorageSize" : 16384,
        "capped" : true,
        "max" : 3,
        "maxSize" : 5242880,
        "wiredTiger" : {
                "metadata" : {
                        "formatVersion" : 1
                },
                }
        },
        "nindexes" : 1,
        "indexBuilds" : [ ],
        "totalIndexSize" : 36864,
        "totalSize" : 73728,
        "indexSizes" : {
                "_id_" : 36864
        },
        "scaleFactor" : 1,
        "ok" : 1
}
>
```

```
> db.emp4.stats({scale:1024000})
{
        "ns" : "insofedb.emp4",
        "size" : 0,
        "count" : 3,
        "avgObjSize" : 51,
        "storageSize" : 0,
        "freeStorageSize" : 0,
        "capped" : true,
        "max" : 3,
        "maxSize" : 5,
        "wiredTiger" : {
                "metadata" : {
                        "formatVersion" : 1
                },
        },
        "nindexes" : 1,
        "indexBuilds" : [ ],
        "totalIndexSize" : 0,
        "totalSize" : 0,
        "indexSizes" : {
                "_id_" : 0
        },
        "scaleFactor" : 1024000,
        "ok" : 1
}
```

```
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
> db.emp2.insert({empid:"102",name:"bbb",domain:{dept:"DS",experience:"2 years"}})
WriteResult({ "nInserted" : 1 })
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
>
```

```
> db.emp2.insert({_id:003,empid:"103",name:"ccc",domain:{dept:"DS",experience:"2 years"}})
WriteResult({ "nInserted" : 1 })
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : 3, "empid" : "103", "name" : "ccc", "domain" : { "dept" : "DS", "experience" : "2 years" } }
>
```

```
> db.emp2.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

```
> db.emp2.createIndex({empid:101},{unique:true})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

```
> db.emp2.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "empid" : 101
                },
                "name" : "empid_101",
                "unique" : true
        }
```

```
> db.emp2.dropIndexes()
{
        "nIndexesWas" : 2,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
> db.emp2.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

```
> db.emp2.createIndex({empid:101},{unique:true})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

```
> db.emp2.insert({empid:"101",name:"aaa",domain:{dept:"DS",experience:"2 years"}})
WriteResult({ "nInserted" : 1 })
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : 3, "empid" : "103", "name" : "ccc", "domain" : { "dept" : "DS", "experience" : "2 years" } }
{ "_id" : ObjectId("626b112c8353db8626310e61"), "empid" : "101", "name" : "aaa", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
> db.emp2.insert({empid:"101",name:"aaa",domain:{dept:"DS",experience:"2 years"}})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "E11000 duplicate key error collection: insofedb.emp2 index: empid_101 dup key
: { empid: \"101\" }"
        }
```

```
> db.getName()
test
> show dbs
admin       0.000GB
config      0.000GB
infodb      0.000GB
insofedb    0.000GB
local       0.000GB
> use insofedb
switched to db insofedb
> show collections
emp1
emp2
emp3
emp4
```

```
> db.emp1.find()
> var new_data=[{empid:101,name:'aaa'},{empid:102,name:'bbb'},{empid:103,name:'ccc'}]
> db.emp1.insert(new_data)
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
>
```

```
> var new_data=[{empid:104,name:'aaa'},{empid:105,name:'bbb'},{empid:106,name:'ccc'}]
> db.emp1.insert(new_data)
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
>
```

```
> db.emp1.update[{name:"aaa"},{$set:{"name":"aaaa"}}]
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
> db.emp1.update({name:"aaa"},{$set:{"name":"aaaa"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
>
```

```
> db.emp1.update({name:"bbb"},{$set:{name:"bbbb"}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
>
```

```
> db.emp1.update({name:"ddd"},{$set:{"name":"dddd"}},{upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("626b71557fe1e5bfec470e27")
})
> db.emp1.update({name:"ddd"},{$set:{"name":"eee"}},{upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("626b71637fe1e5bfec470e2a")
})
```

```
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
{ "_id" : ObjectId("626b71557fe1e5bfec470e27"), "name" : "dddd" }
{ "_id" : ObjectId("626b71637fe1e5bfec470e2a"), "name" : "eee" }
> db.emp1.update({name:"eee"},{$set:{"name":"eeee"}},{upsert:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
{ "_id" : ObjectId("626b71557fe1e5bfec470e27"), "name" : "dddd" }
{ "_id" : ObjectId("626b71637fe1e5bfec470e2a"), "name" : "eeee" }
```

```
> db.emp1.remove({name:'eeee'})
WriteResult({ "nRemoved" : 1 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
{ "_id" : ObjectId("626b697886713d015d293ca0"), "empid" : 106, "name" : "ccc" }
{ "_id" : ObjectId("626b71557fe1e5bfec470e27"), "name" : "dddd" }
```

```
> db.emp1.remove({name:'bbbb'})
WriteResult({ "nRemoved" : 2 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
>
```

```
> db.emp1.remove({empid:106})
WriteResult({ "nRemoved" : 1 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
{ "_id" : ObjectId("626b71557fe1e5bfec470e27"), "name" : "dddd" }
> db.emp1.remove({"_id" : ObjectId("626b71557fe1e5bfec470e27")})
WriteResult({ "nRemoved" : 1 })
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9c"), "empid" : 102, "name" : "bbbb" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b697886713d015d293c9f"), "empid" : 105, "name" : "bbbb" }
```

```
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({name:'ccc'})
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
> db.emp1.find().limit(2)
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
> db.emp1.find().sort({empid:-1})
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
> db.emp1.find().sort({empid:1})
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
>
```

```
> db.emp2.find()
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : 3, "empid" : "103", "name" : "ccc", "domain" : { "dept" : "DS", "experience" : "2 years" } }
{ "_id" : ObjectId("626b112c8353db8626310e61"), "empid" : "101", "name" : "aaa", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
> db.emp2.find({domain.dept:"DS"})
uncaught exception: SyntaxError: missing : after property id :
@(shell):1:20
> db.emp2.find({"domain.dept":"ds"})
> db.emp2.find({"domain.dept":"DS"})
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : 3, "empid" : "103", "name" : "ccc", "domain" : { "dept" : "DS", "experience" : "2 years" } }
{ "_id" : ObjectId("626b112c8353db8626310e61"), "empid" : "101", "name" : "aaa", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
>
```

```
> db.emp2.find().sort({"domain.experience":-1})
{ "_id" : ObjectId("626b0ed28353db8626310e60"), "empid" : "102", "name" : "bbb", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : 3, "empid" : "103", "name" : "ccc", "domain" : { "dept" : "DS", "experience" : "2 years" } }
{ "_id" : ObjectId("626b112c8353db8626310e61"), "empid" : "101", "name" : "aaa", "domain" : { "dept" : "D
S", "experience" : "2 years" } }
{ "_id" : ObjectId("626afb9a8353db8626310e58"), "empid" : 101, "name" : "aaa" }
>
```

```
> db.emp1.find()
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({empid:{$in:[101,104]}})
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({name:{$in:['aaaa','ccc']}});
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
```

```
> db.emp1.find();
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({empid:{$lt:102}});
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
> db.emp1.find({empid:{$gt:102}});
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({empid:{$gte:102}});
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
> db.emp1.find({empid:{$gte:101}});
{ "_id" : ObjectId("626b670086713d015d293c9b"), "empid" : 101, "name" : "aaaa" }
{ "_id" : ObjectId("626b670086713d015d293c9d"), "empid" : 103, "name" : "ccc" }
{ "_id" : ObjectId("626b697886713d015d293c9e"), "empid" : 104, "name" : "aaa" }
>
```

```
> db.emp1.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
> db.emp1.explain("executionStats").find({name:aaa})
uncaught exception: ReferenceError: aaa is not defined :
@(shell):1:41
> db.emp1.explain("executionStats").find({name:"aaa"})
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "insofedb.emp1",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "name" : {
```

```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 1,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 3,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "name" : {
                                "$eq" : "aaa"
                        }
```

```
> db.emp1.createIndex({name:1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> db.emp1.explain("executionStats").find({name:"aaa"})
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "insofedb.emp1",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "name" : {
                                "$eq" : "aaa"
```

```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 5,
        "totalKeysExamined" : 1,
        "totalDocsExamined" : 1,
        "executionStages" : {
                "stage" : "FETCH",
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 0,
                "works" : 2,
                "advanced" : 1,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
                "docsExamined" : 1,
                "alreadyHasObj" : 0,
```

```
> db.emp1.createIndex({empid:-1})
{
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

```
> db.emp1.dropIndexes()
{
        "nIndexesWas" : 3,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
> db.emp1.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```

```
> db.emp1.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "name" : 1
                },
                "name" : "name_1"
        },
        {
                "v" : 2,
                "key" : {
                        "empid" : -1
                },
                "name" : "empid_-1"
        }
```

```
insofedb> db.emp1.find();
[
  {
    _id: ObjectId("626b670086713d015d293c9b"),
    empid: 101,
    name: 'aaaa'
  },
  {
    _id: ObjectId("626b670086713d015d293c9d"),
    empid: 103,
    name: 'ccc'
  },
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
```

```
insofedb> db.emp1.aggregate([{$match:{name:"aa"}}])

insofedb> db.emp1.aggregate([{$match:{name:"aaa"}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
insofedb> db.emp1.aggregate([{$match:{name:"aaaa"}}])
[
  {
    _id: ObjectId("626b670086713d015d293c9b"),
    empid: 101,
    name: 'aaaa'
  }
]
```

```
insofedb> db.emp1.aggregate([{$match:{$or:[{name:"aaa"},{empid:"104"}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:104}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:"104"}]}}])
```

```
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:{$gte:104}}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:{$gte:104,$lt:106}}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
```

```
insofedb> db.emp1.insert({empid:106,name:'aaa'})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62720ecbbc4cc625bca09c2c") }
}
insofedb> db.emp1.find()
```

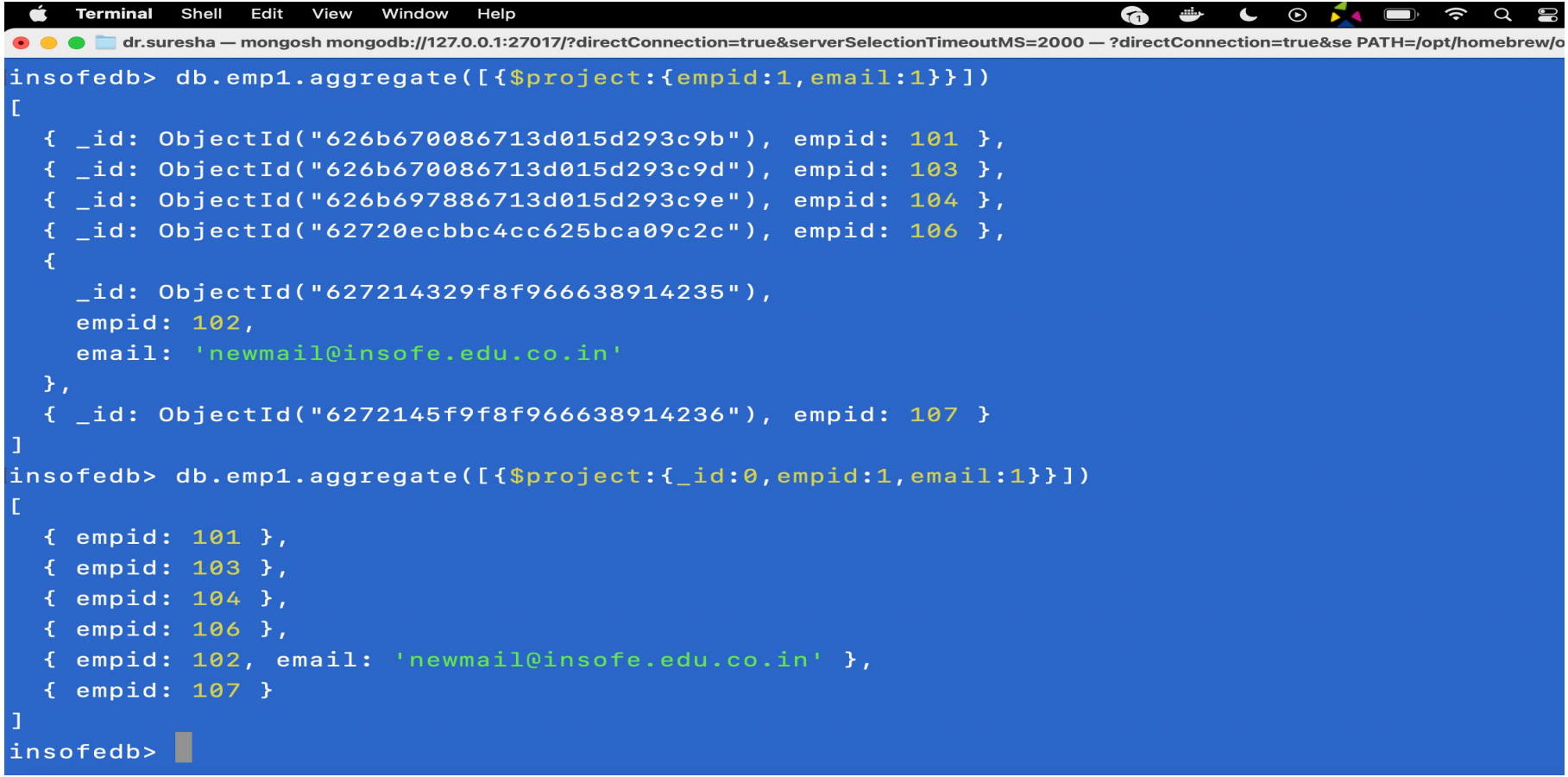
```
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:{$gte:104,$lt:106}}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  }
]
insofedb> db.emp1.aggregate([{$match:{$and:[{name:"aaa"},{empid:{$gte:104,$lte:106}}]}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  },
  {
    _id: ObjectId("62720ecbbc4cc625bca09c2c"),
    empid: 106,
    name: 'aaa'
  }
]
```

```
insofedb> db.emp1.insert({empid:102,name:'aaa',email:"newmail@insofe.edu.co.in"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("627214329f8f966638914235") }
}
```

```
insofedb> db.emp1.insert({empid:107})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6272145f9f8f966638914236") }
}
```

```
insofedb> db.emp1.aggregate([{$project:{empid:1,name:1}}])
[
  {
    _id: ObjectId("626b670086713d015d293c9b"),
    empid: 101,
    name: 'aaaa'
  },
  {
    _id: ObjectId("626b670086713d015d293c9d"),
    empid: 103,
    name: 'ccc'
  },
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  },
  {
    _id: ObjectId("62720ecbbc4cc625bca09c2c"),
    empid: 106,
    name: 'aaa'
  },
  {
    _id: ObjectId("627214329f8f966638914235"),
    empid: 102,
    name: 'aaa'
  },
  { _id: ObjectId("6272145f9f8f966638914236"), empid: 107 }
]
```

```
insofedb> db.emp1.aggregate([{$project:{empid:1,name:1,email:1}}])
[
  {
    _id: ObjectId("626b670086713d015d293c9b"),
    empid: 101,
    name: 'aaaa'
  },
  {
    _id: ObjectId("626b670086713d015d293c9d"),
    empid: 103,
    name: 'ccc'
  },
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  },
  {
    _id: ObjectId("62720ecbbc4cc625bca09c2c"),
    empid: 106,
    name: 'aaa'
  },
  {
    _id: ObjectId("627214329f8f966638914235"),
    empid: 102,
    name: 'aaa',
    email: 'newmail@insofe.edu.co.in'
  },
  { _id: ObjectId("6272145f9f8f966638914236"), empid: 107 }
]
```


Terminal Shell Edit View Window Help
dr.suresha — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — ?directConnection=true&se PATH=/opt/homebrew/o
insofedb> db.emp1.aggregate([{$project:{empid:1,email:1}}])
[
{ _id: ObjectId("626b670086713d015d293c9b"), empid: 101 },
{ _id: ObjectId("626b670086713d015d293c9d"), empid: 103 },
{ _id: ObjectId("626b697886713d015d293c9e"), empid: 104 },
{ _id: ObjectId("62720ecbbc4cc625bca09c2c"), empid: 106 },
{
_id: ObjectId("627214329f8f966638914235"),
empid: 102,
email: 'newmail@insofe.edu.co.in'
},
{ _id: ObjectId("6272145f9f8f966638914236"), empid: 107 }
]
insofedb> db.emp1.aggregate([{$project:{_id:0,empid:1,email:1}}])
[
{ empid: 101 },
{ empid: 103 },
{ empid: 104 },
{ empid: 106 },
{ empid: 102, email: 'newmail@insofe.edu.co.in' },
{ empid: 107 }
]
insofedb>

```
insofedb> db.emp1.aggregate([{$match:{"name":"aaa"}},{$project:{empid:1,name:1}}])
[
  {
    _id: ObjectId("626b697886713d015d293c9e"),
    empid: 104,
    name: 'aaa'
  },
  {
    _id: ObjectId("62720ecbbc4cc625bca09c2c"),
    empid: 106,
    name: 'aaa'
  },
  {
    _id: ObjectId("627214329f8f966638914235"),
    empid: 102,
    name: 'aaa'
  }
]
```

```
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"}}}])
[
  { _id: { name: 'aaa' } },
  { _id: { name: 'aaaa' } },
  { _id: { name: 'ccc' } },
  { _id: { name: null } }
]
insofedb> db.emp1.aggregate([{$group:{"name":"$name"}}])
MongoServerError: The field 'name' must be an accumulator object
insofedb> db.emp1.aggregate([{$group:{"_id":{"empid":"$empid"}}}])
[
  { _id: { empid: 104 } },
  { _id: { empid: 103 } },
  { _id: { empid: 106 } },
  { _id: { empid: 107 } },
  { _id: { empid: 102 } },
  { _id: { empid: 101 } }
]
```

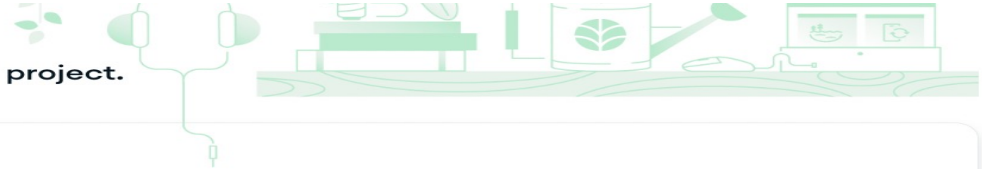
```
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"},"count":{$sum:1},"average_empid":{$avg:"$emp
id"}}}])
[
  { _id: { name: 'ccc' }, count: 1, average_empid: 103 },
  { _id: { name: 'aaa' }, count: 3, average_empid: 104 },
  { _id: { name: 'aaaa' }, count: 1, average_empid: 101 },
  { _id: { name: null }, count: 1, average_empid: 107 }
]
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"},"count":{$sum:1},"average_empid":{$avg:"$emp
id"}}}]).sort({"name":1})
[
  { _id: { name: 'aaa' }, count: 3, average_empid: 104 },
  { _id: { name: 'aaaa' }, count: 1, average_empid: 101 },
  { _id: { name: 'ccc' }, count: 1, average_empid: 103 },
  { _id: { name: null }, count: 1, average_empid: 107 }
]
```

dr.suresha — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — ?directConnection=true&se PATH=/opt/homebrew/opt/node@14/bin:/Use...

```
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"},"count":{$sum:1},"min_empid":{$min:"$empid"}
}}])
[
  { _id: { name: 'aaaa' }, count: 1, min_empid: 101 },
  { _id: { name: 'aaa' }, count: 3, min_empid: 102 },
  { _id: { name: 'ccc' }, count: 1, min_empid: 103 },
  { _id: { name: null }, count: 1, min_empid: 107 }
]
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"},"count":{$sum:1},"first_empid":{$first:"$emp
id"}}}])
[
  { _id: { name: 'aaa' }, count: 3, first_empid: 104 },
  { _id: { name: 'ccc' }, count: 1, first_empid: 103 },
  { _id: { name: 'aaaa' }, count: 1, first_empid: 101 },
  { _id: { name: null }, count: 1, first_empid: 107 }
]
insofedb> db.emp1.aggregate([{$group:{"_id":{"name":"$name"},"count":{$sum:1},"first_empid":{$last:"$empi
d"}}}])
[
  { _id: { name: 'aaa' }, count: 3, first_empid: 102 },
  { _id: { name: 'aaaa' }, count: 1, first_empid: 101 },
  { _id: { name: 'ccc' }, count: 1, first_empid: 103 },
  { _id: { name: null }, count: 1, first_empid: 107 }
```

Welcome to Atlas!

Tell us a few things about yourself and your project.



What is your goal today?

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

- ○ Build a new application
- ○ Migrate an existing application
- ○ Explore what I can build
- ◉ Learn MongoDB

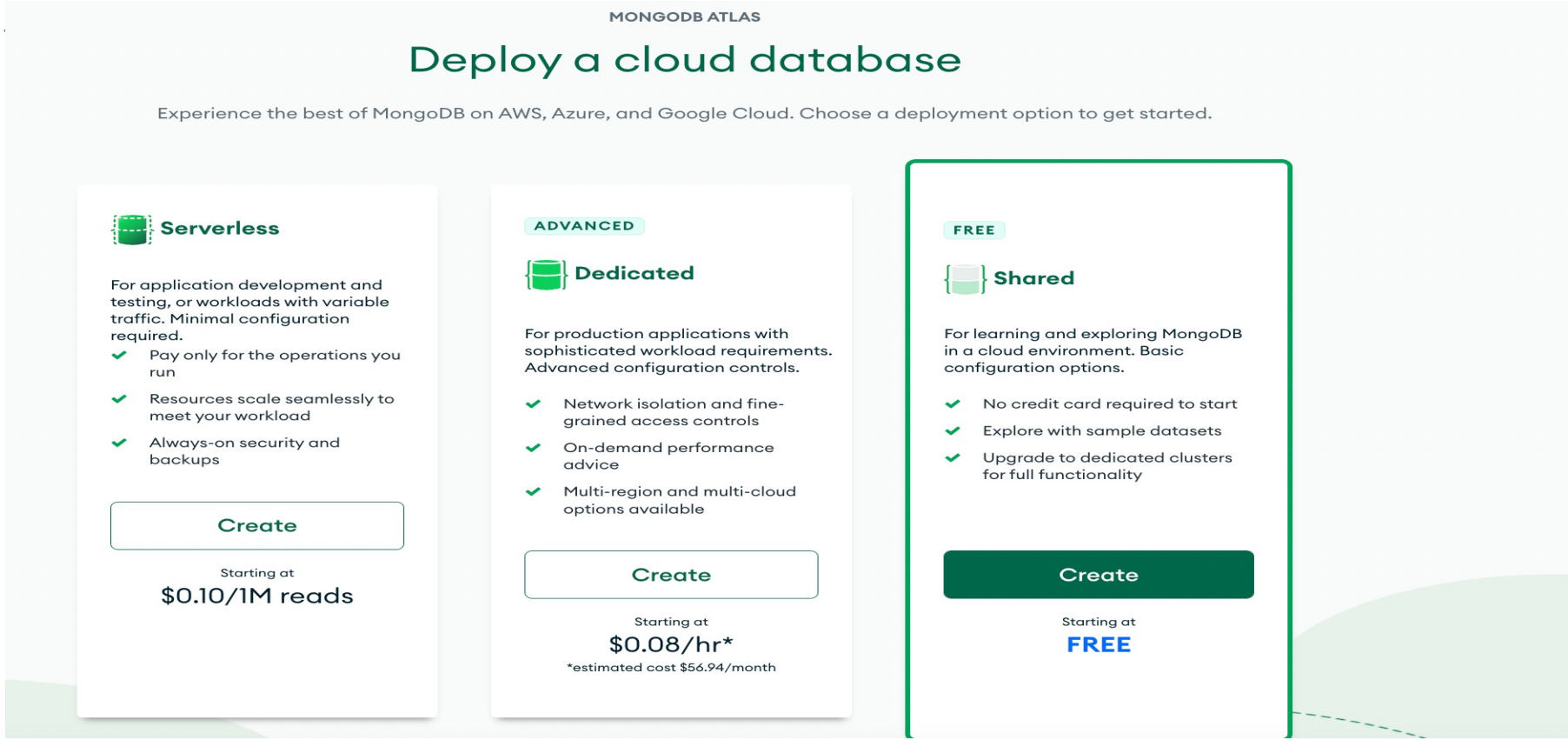
What type of application are you building?

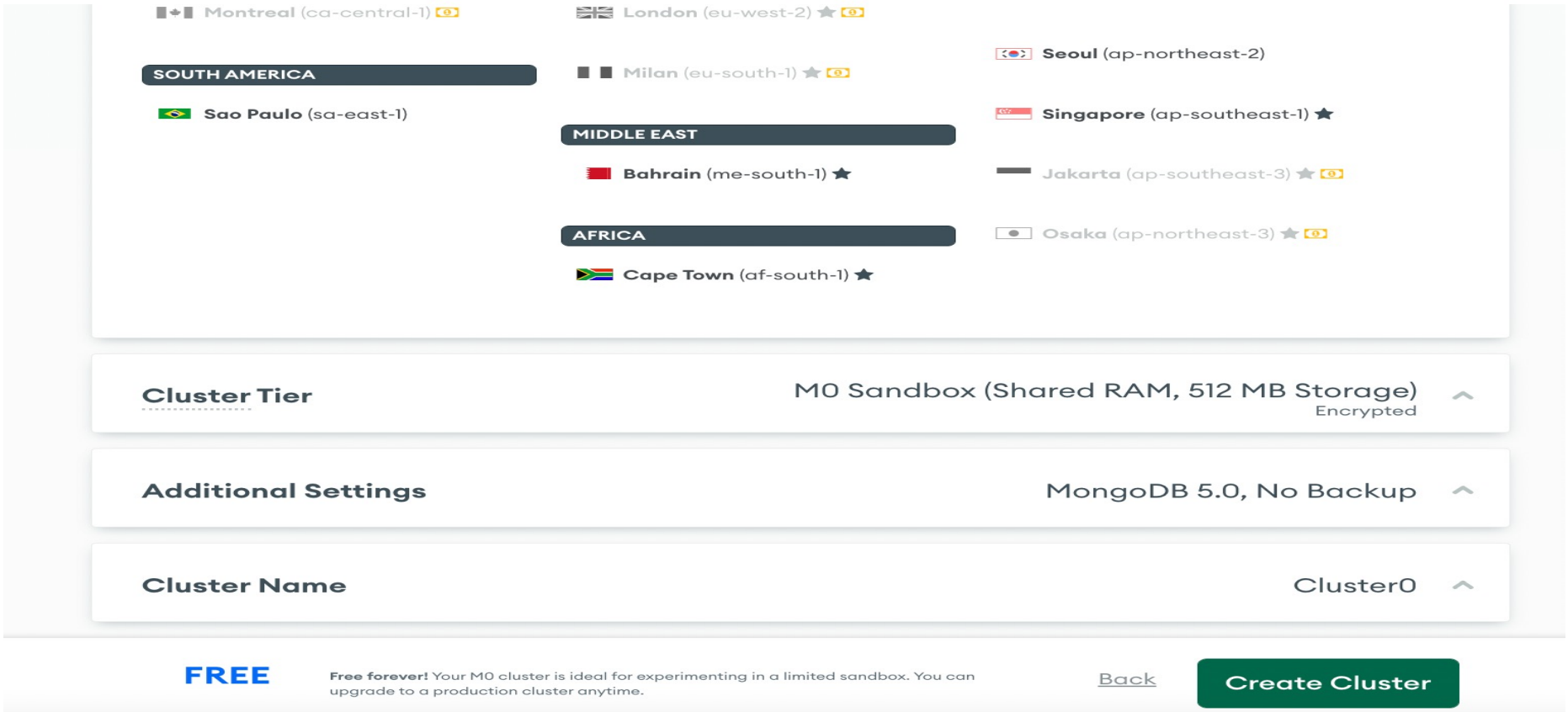
No Application ▾

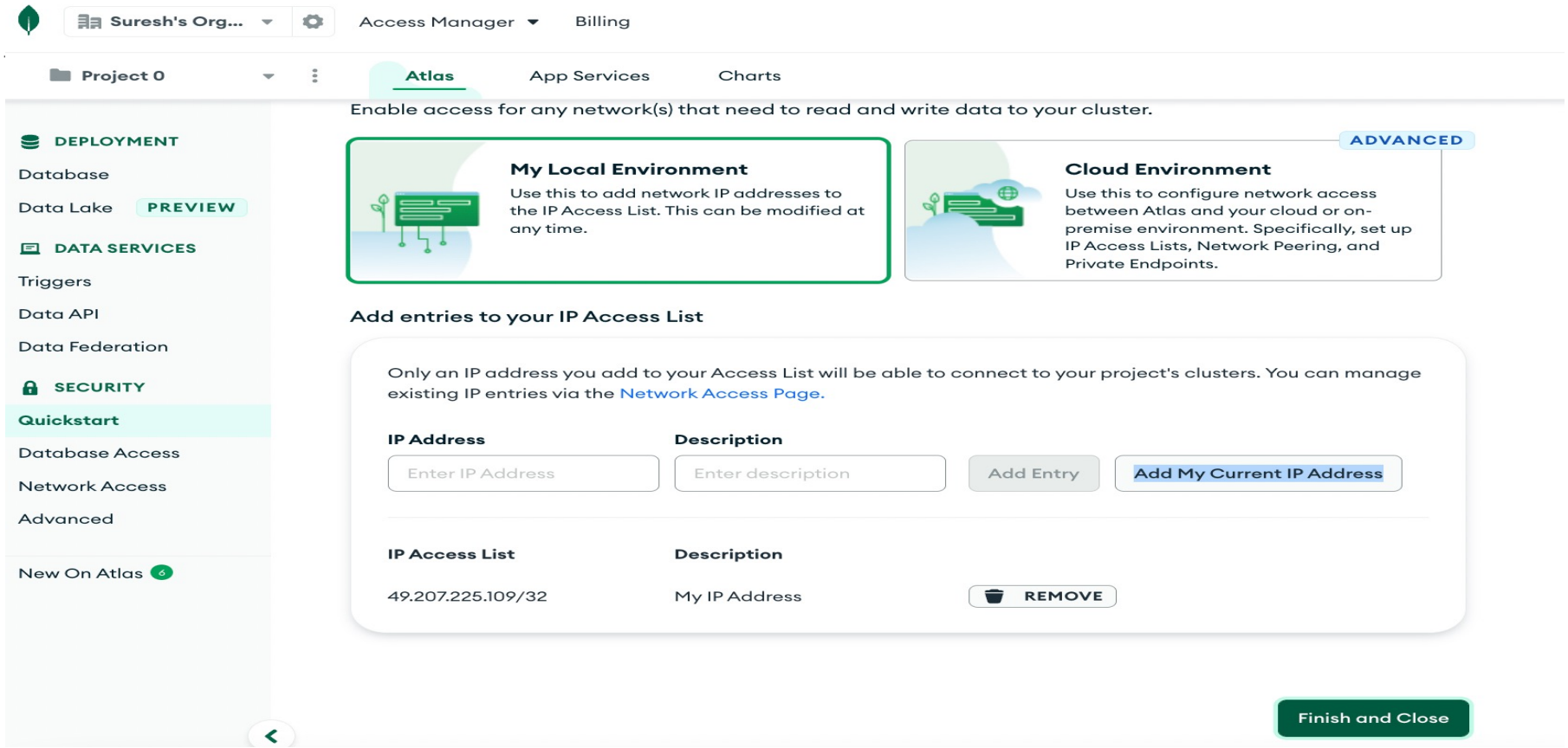
What is your preferred language?

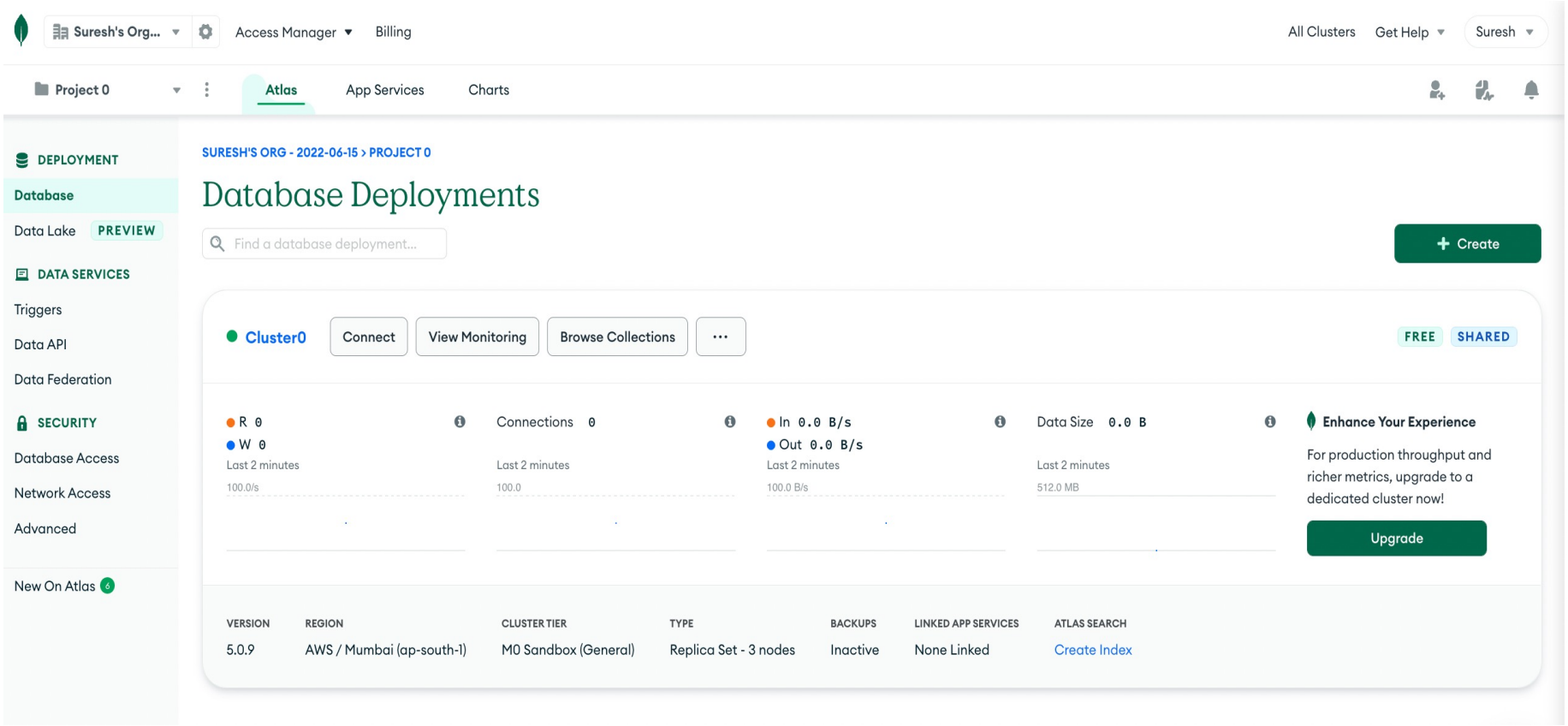
We'll use this to customize code samples and content we share with you. You can always change this later.

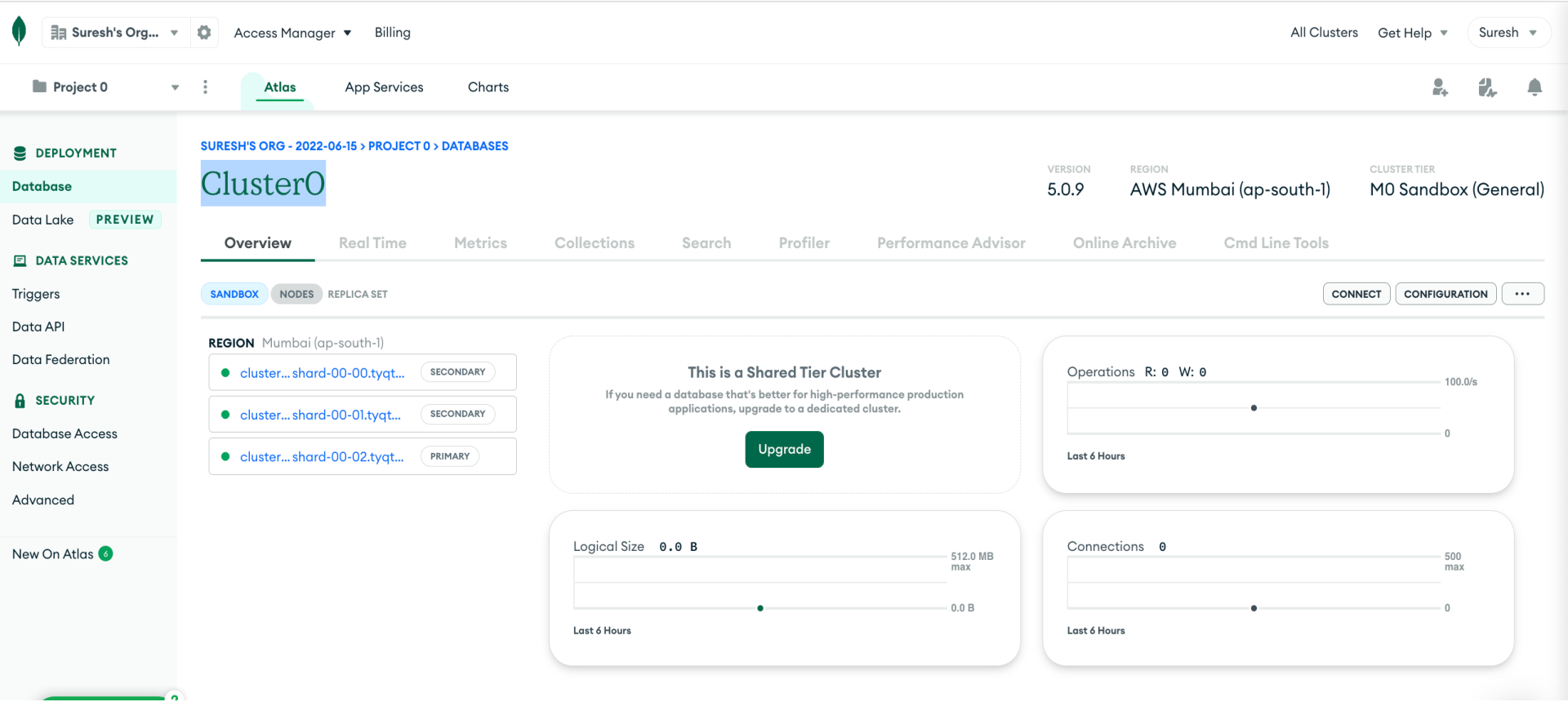
Python ▾

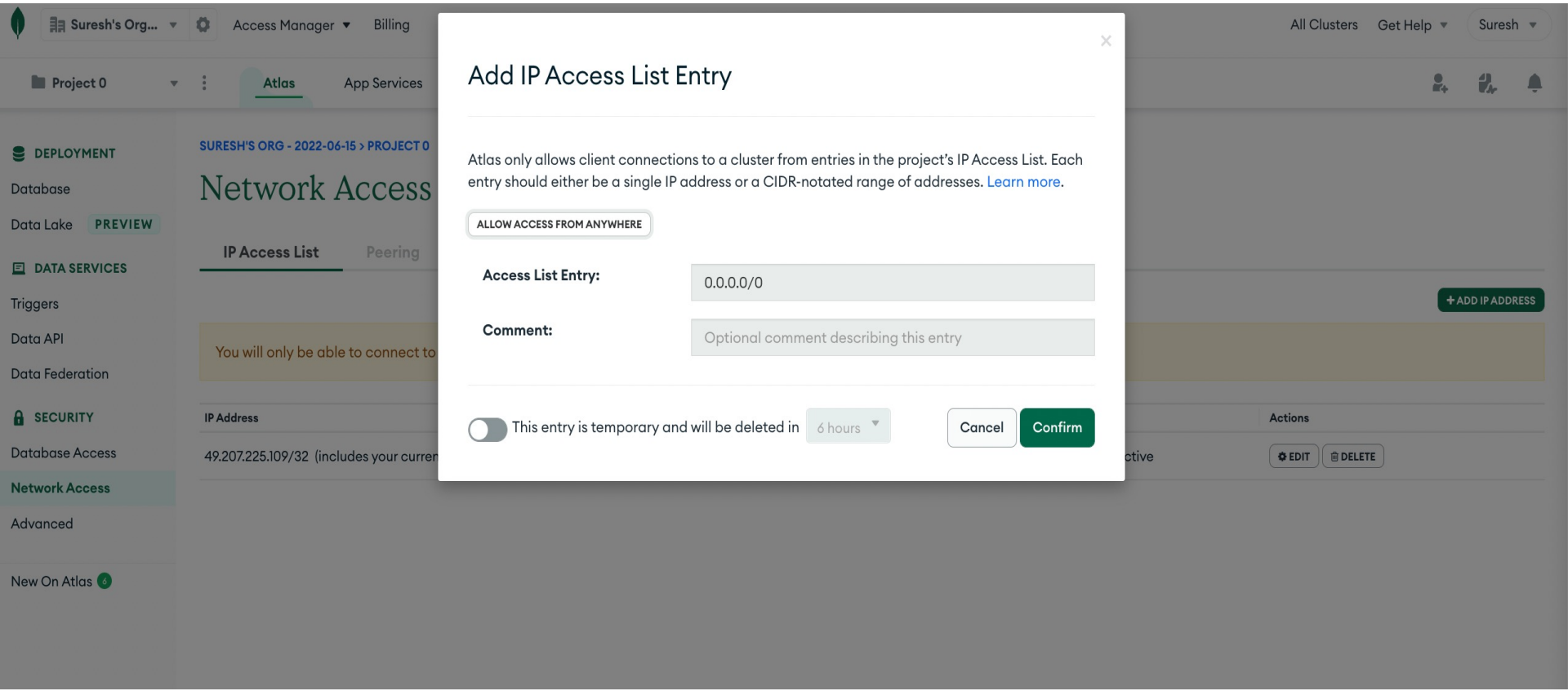

MONGODB ATLAS
Deploy a cloud database
Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.
Serverless
For application development and testing, or workloads with variable traffic. Minimal configuration required.
Pay only for the operations you run
Resources scale seamlessly to meet your workload
Always-on security and backups
Create
Starting at
$0.10/1M reads
ADVANCED
Dedicated
For production applications with sophisticated workload requirements. Advanced configuration controls.
Network isolation and fine-grained access controls
On-demand performance advice
Multi-region and multi-cloud options available
Create
Starting at
$0.08/hr*
*estimated cost $56.94/month
FREE
Shared
For learning and exploring MongoDB in a cloud environment. Basic configuration options.
No credit card required to start
Explore with sample datasets
Upgrade to dedicated clusters for full functionality
Create
Starting at
FREE

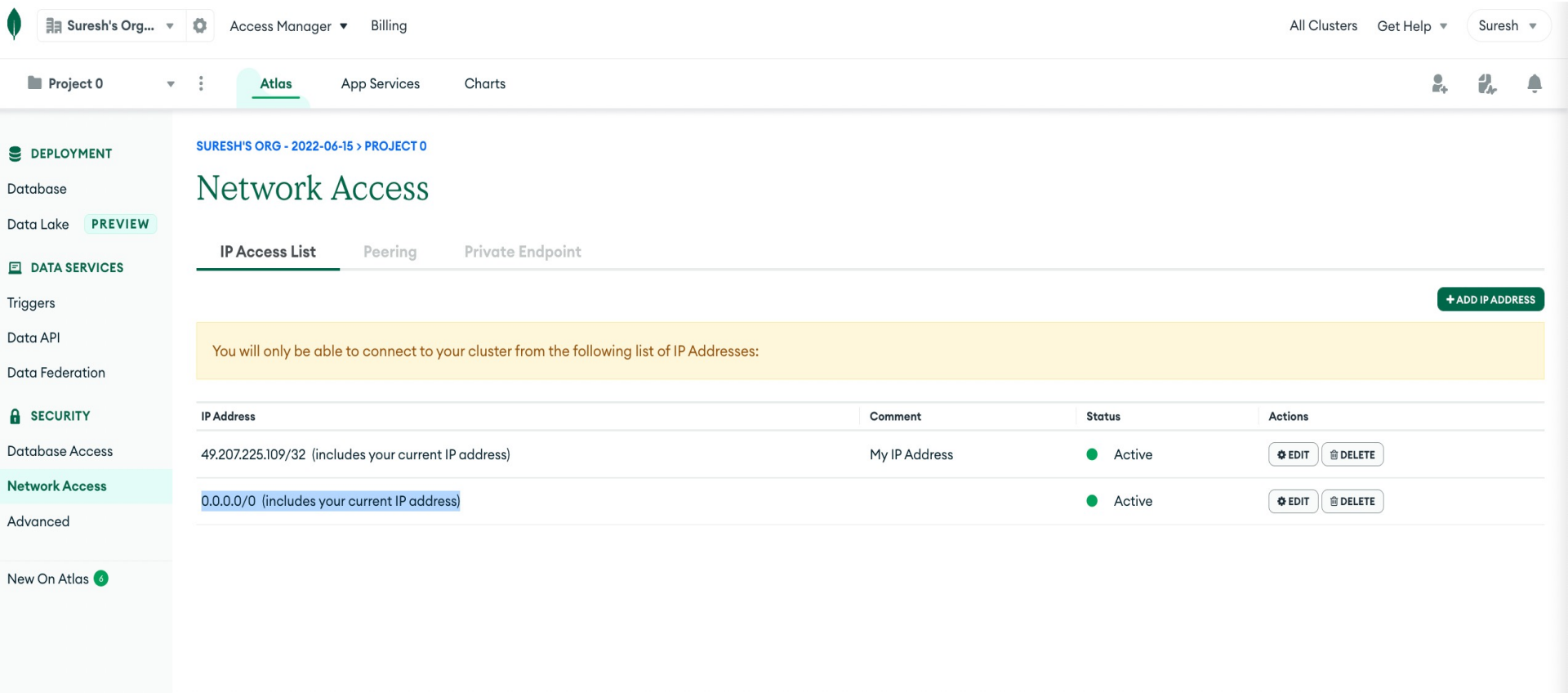

Montreal (ca-central-1)
London (eu-west-2)
SOUTH AMERICA
Milan (eu-south-1)
Seoul (ap-northeast-2)
Sao Paulo (sa-east-1)
Singapore (ap-southeast-1)
MIDDLE EAST
Bahrain (me-south-1)
Jakarta (ap-southeast-3)
AFRICA
Osaka (ap-northeast-3)
Cape Town (af-south-1)
Cluster Tier
M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted
Additional Settings
MongoDB 5.0, No Backup
Cluster Name
Cluster0
FREE
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.
Back
Create Cluster

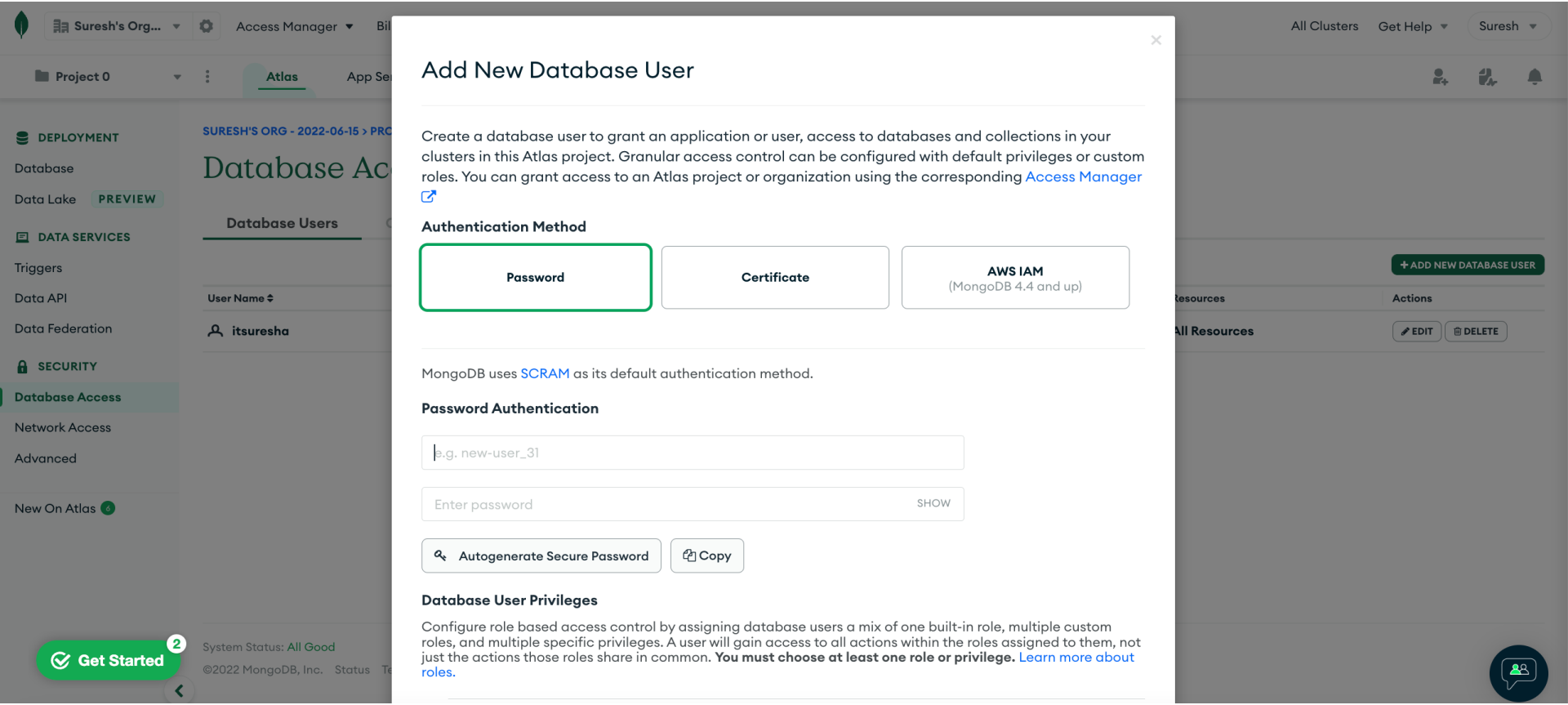

Suresh's Org...
Access Manager
Billing
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Quickstart
Database Access
Network Access
Advanced
New On Atlas
Enable access for any network(s) that need to read and write data to your cluster.
My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.
ADVANCED
Cloud Environment
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.
Add entries to your IP Access List
Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.
IP Address
Enter IP Address
Description
Enter description
Add Entry
Add My Current IP Address
IP Access List
Description
49.207.225.109/32
My IP Address
REMOVE
Finish and Close

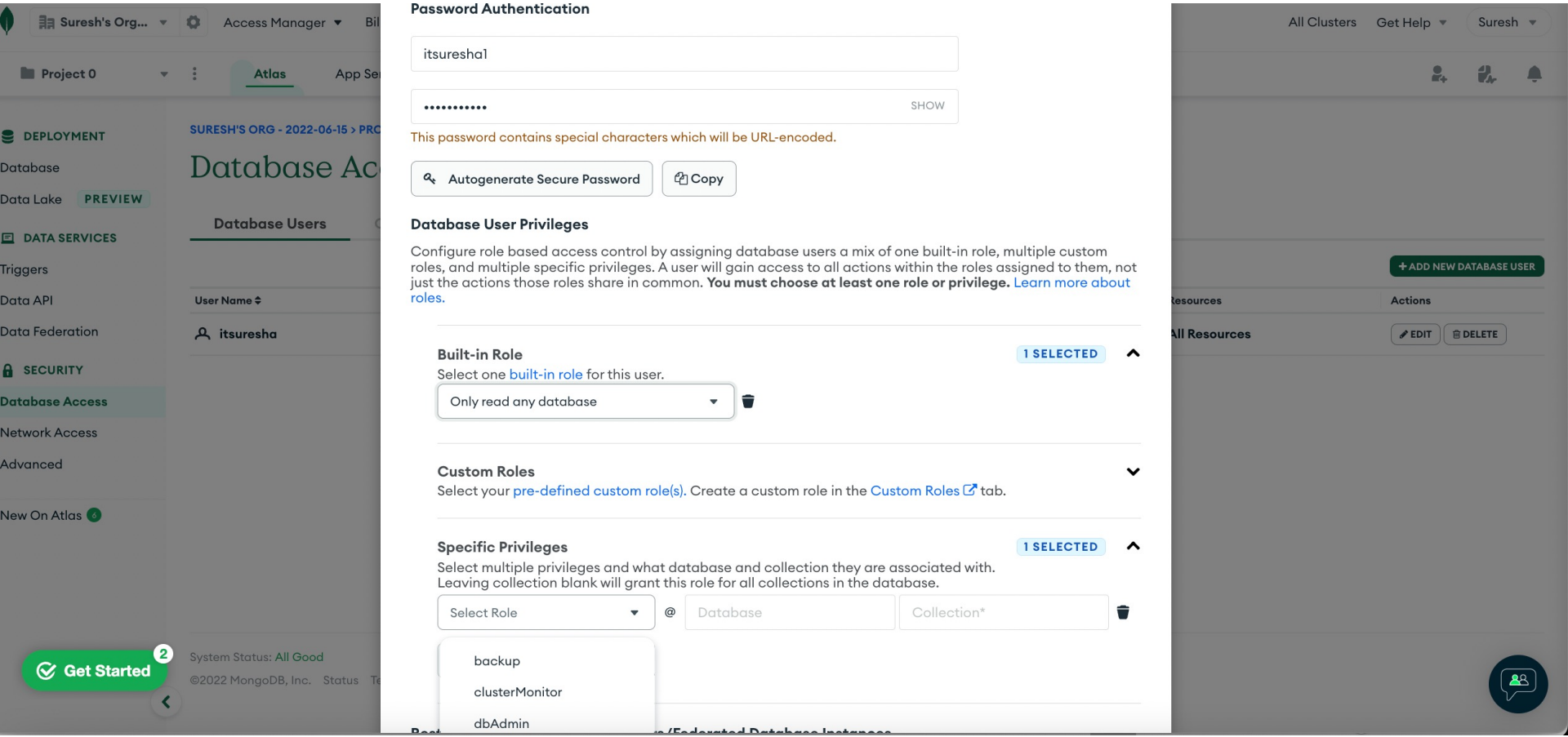

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
6
SURESH'S ORG - 2022-06-15 > PROJECT 0
Database Deployments
Find a database deployment...
+ Create
Cluster0
Connect
View Monitoring
Browse Collections
FREE
SHARED
R 0
W 0
Last 2 minutes
Connections 0
Last 2 minutes
In 0.0 B/s
Out 0.0 B/s
Last 2 minutes
Data Size 0.0 B
Last 2 minutes
Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!
Upgrade
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SEARCH
5.0.9
AWS / Mumbai (ap-south-1)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Create Index

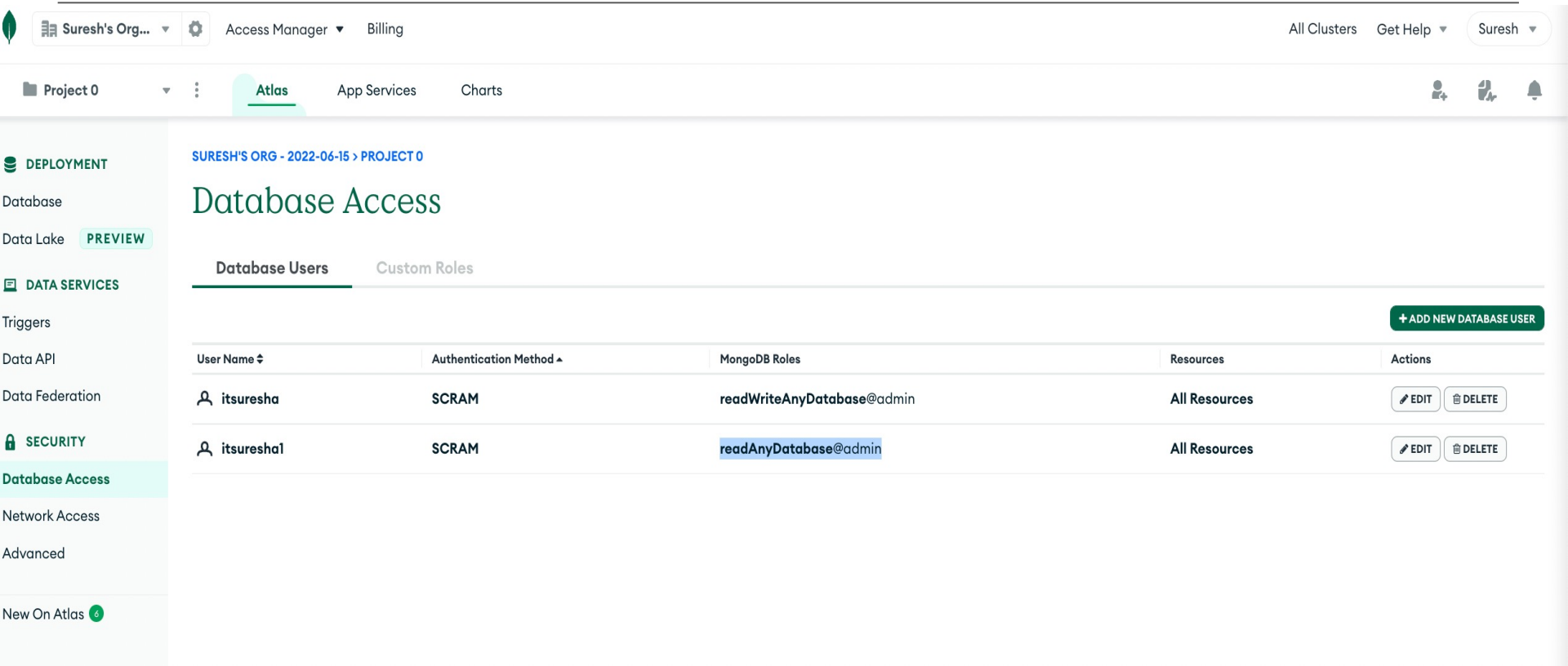

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PROJECT 0 > DATABASES
Cluster0
VERSION
5.0.9
REGION
AWS Mumbai (ap-south-1)
CLUSTER TIER
M0 Sandbox (General)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
SANDBOX
NODES
REPLICA SET
CONNECT
CONFIGURATION
REGION Mumbai (ap-south-1)
cluster... shard-00-00.tyqt...
SECONDARY
cluster... shard-00-01.tyqt...
SECONDARY
cluster... shard-00-02.tyqt...
PRIMARY
This is a Shared Tier Cluster
If you need a database that's better for high-performance production applications, upgrade to a dedicated cluster.
Upgrade
Operations R: 0 W: 0
100.0/s
0
Last 6 Hours
Logical Size 0.0 B
512.0 MB max
0.0 B
Last 6 Hours
Connections 0
500 max
0
Last 6 Hours

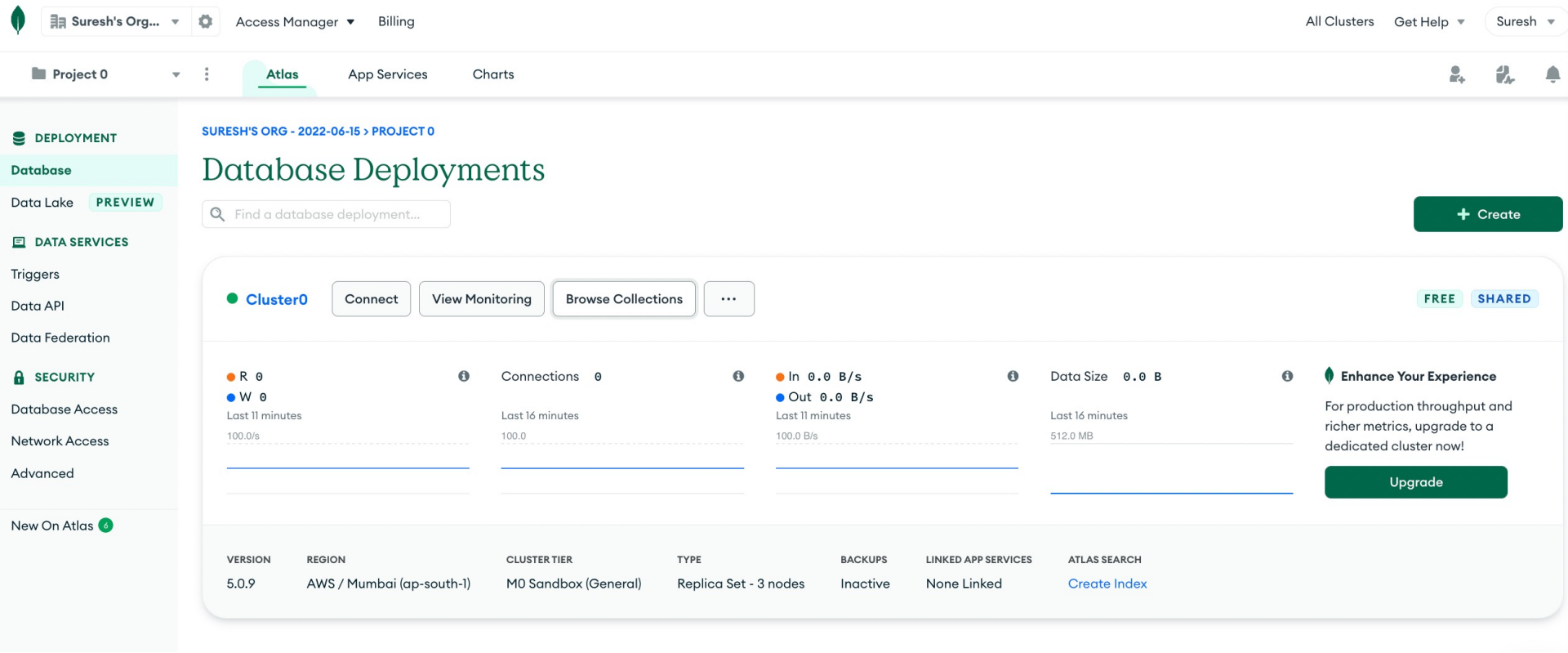

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PROJECT 0
Network Access
IP Access List
Peering
+ ADD IP ADDRESS
You will only be able to connect to
IP Address
Actions
49.207.225.109/32 (includes your curren
ctive
EDIT
DELETE
Add IP Access List Entry
Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. Learn more.
ALLOW ACCESS FROM ANYWHERE
Access List Entry:
0.0.0.0/0
Comment:
Optional comment describing this entry
This entry is temporary and will be deleted in
6 hours
Cancel
Confirm


Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PROJECT 0
Network Access
IP Access List
Peering
Private Endpoint
+ ADD IP ADDRESS
You will only be able to connect to your cluster from the following list of IP Addresses:
IP Address
Comment
Status
Actions
49.207.225.109/32 (includes your current IP address)
My IP Address
Active
EDIT
DELETE
0.0.0.0/0 (includes your current IP address)
Active
EDIT
DELETE


Suresh's Org...
Access Manager
All Clusters
Get Help
Suresh
Project 0
Atlas
DEPLOYMENT
Database
Data Lake PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PRO
Database Ac
Database Users
+ ADD NEW DATABASE USER
User Name
Resources
Actions
itsuresha
All Resources
EDIT
DELETE
Add New Database User
Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding Access Manager
Authentication Method
Password
Certificate
AWS IAM
(MongoDB 4.4 and up)
MongoDB uses SCRAM as its default authentication method.
Password Authentication
e.g. new-user_31
Enter password
SHOW
Autogenerate Secure Password
Copy
Database User Privileges
Configure role based access control by assigning database users a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. Learn more about roles.
Get Started
2
System Status: All Good
©2022 MongoDB, Inc. Status


Password Authentication
itsuresha1
••••••••••
SHOW
This password contains special characters which will be URL-encoded.
Autogenerate Secure Password
Copy
Database User Privileges
Configure role based access control by assigning database users a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. Learn more about roles.
Built-in Role
1 SELECTED
Select one built-in role for this user.
Only read any database
Custom Roles
Select your pre-defined custom role(s). Create a custom role in the Custom Roles tab.
Specific Privileges
1 SELECTED
Select multiple privileges and what database and collection they are associated with.
Leaving collection blank will grant this role for all collections in the database.
Select Role
@
Database
Collection*
backup
clusterMonitor
dbAdmin
Suresh's Org...
Access Manager
Project 0
Atlas
All Clusters
Get Help
Suresh
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
Database Users
ADD NEW DATABASE USER
User Name
Resources
Actions
itsuresha
All Resources
EDIT
DELETE
Get Started
2
System Status: All Good
©2022 MongoDB, Inc.


Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PROJECT 0
Database Access
Database Users
Custom Roles
+ ADD NEW DATABASE USER
User Name
Authentication Method
MongoDB Roles
Resources
Actions
itsuresha
SCRAM
readWriteAnyDatabase@admin
All Resources
EDIT
DELETE
itsuresha1
SCRAM
readAnyDatabase@admin
All Resources
EDIT
DELETE


Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
SURESH'S ORG - 2022-06-15 > PROJECT 0
Database Deployments
Find a database deployment...
Create
Cluster0
Connect
View Monitoring
Browse Collections
FREE
SHARED
R 0
W 0
Last 11 minutes
Connections 0
Last 16 minutes
In 0.0 B/s
Out 0.0 B/s
Last 11 minutes
Data Size 0.0 B
Last 16 minutes
Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!
Upgrade
VERSION
5.0.9
REGION
AWS / Mumbai (ap-south-1)
CLUSTER TIER
M0 Sandbox (General)
TYPE
Replica Set - 3 nodes
BACKUPS
Inactive
LINKED APP SERVICES
None Linked
ATLAS SEARCH
Create Index

Connect to Cluster0

✔ Setup connection security > Choose a connection method > Connect

Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Go Back

Close

Connect to Cluster0

✔ Setup connection security > ✔ Choose a connection method > Connect

I do not have the MongoDB Shell installed | **I have the MongoDB Shell installed**

1 **Select your operating system and download the** `mongosh`

macOS

Install via Homebrew

```
brew install mongosh
```

Homebrew is a package manager for macOS. Install Homebrew

2 **Run your connection string in your command line**

Use this connection string **in your application**:

```
mongosh "mongodb+srv://cluster0.tyqtb.mongodb.net/myFirstDatabase" --apiVersion 1
--username <username>
```

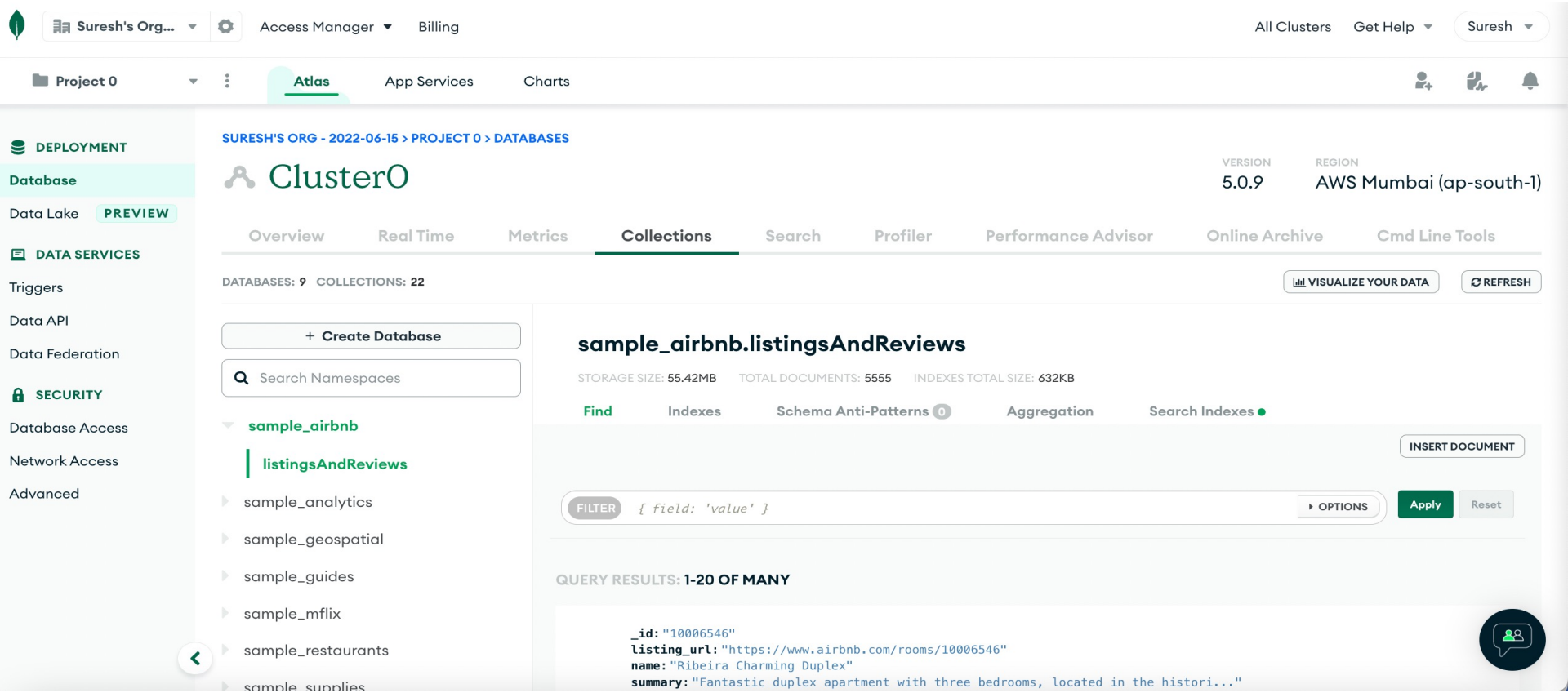
Replace **myFirstDatabase** with the name of the database that connections will use by default. You will be prompted for the password for the Database User, **<username>**. When entering your password, make sure all special characters are URL encoded.

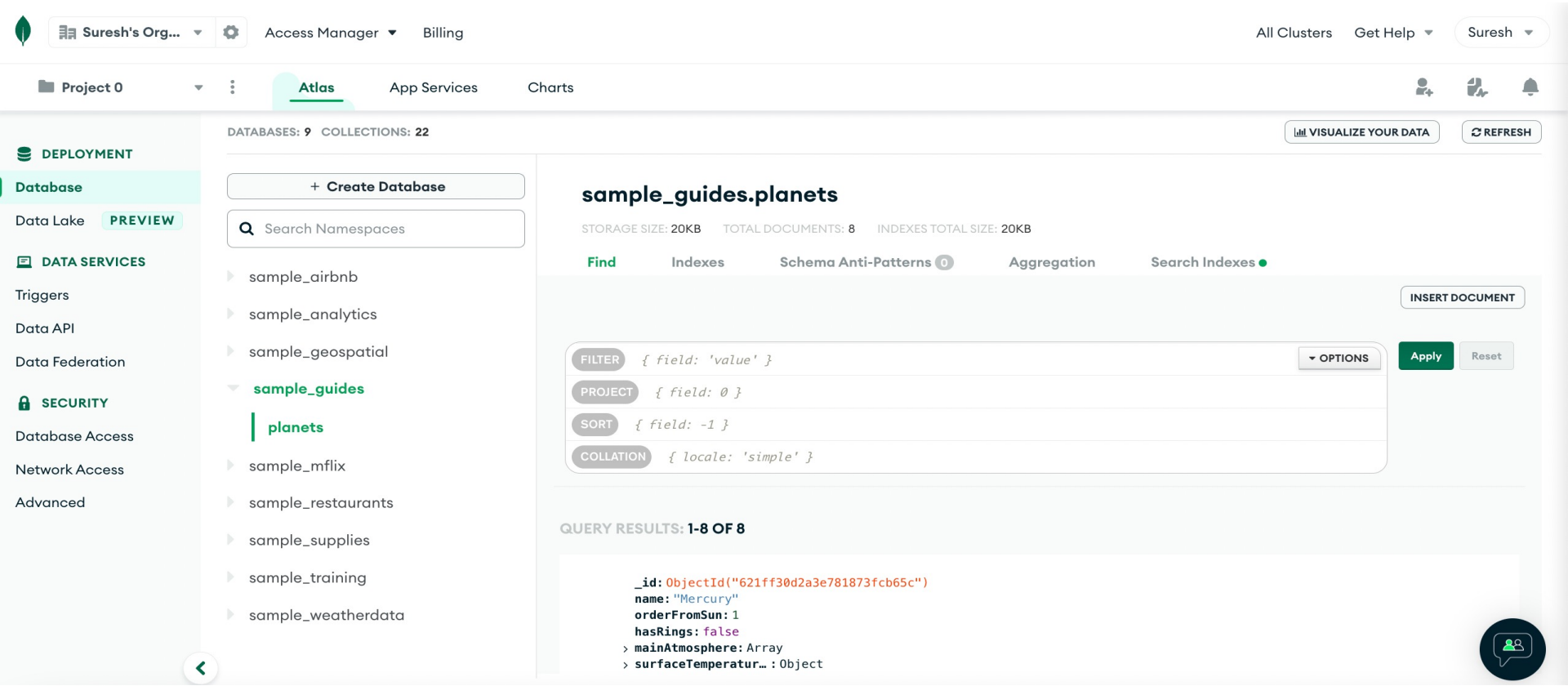
Having trouble connecting? View our troubleshooting documentation

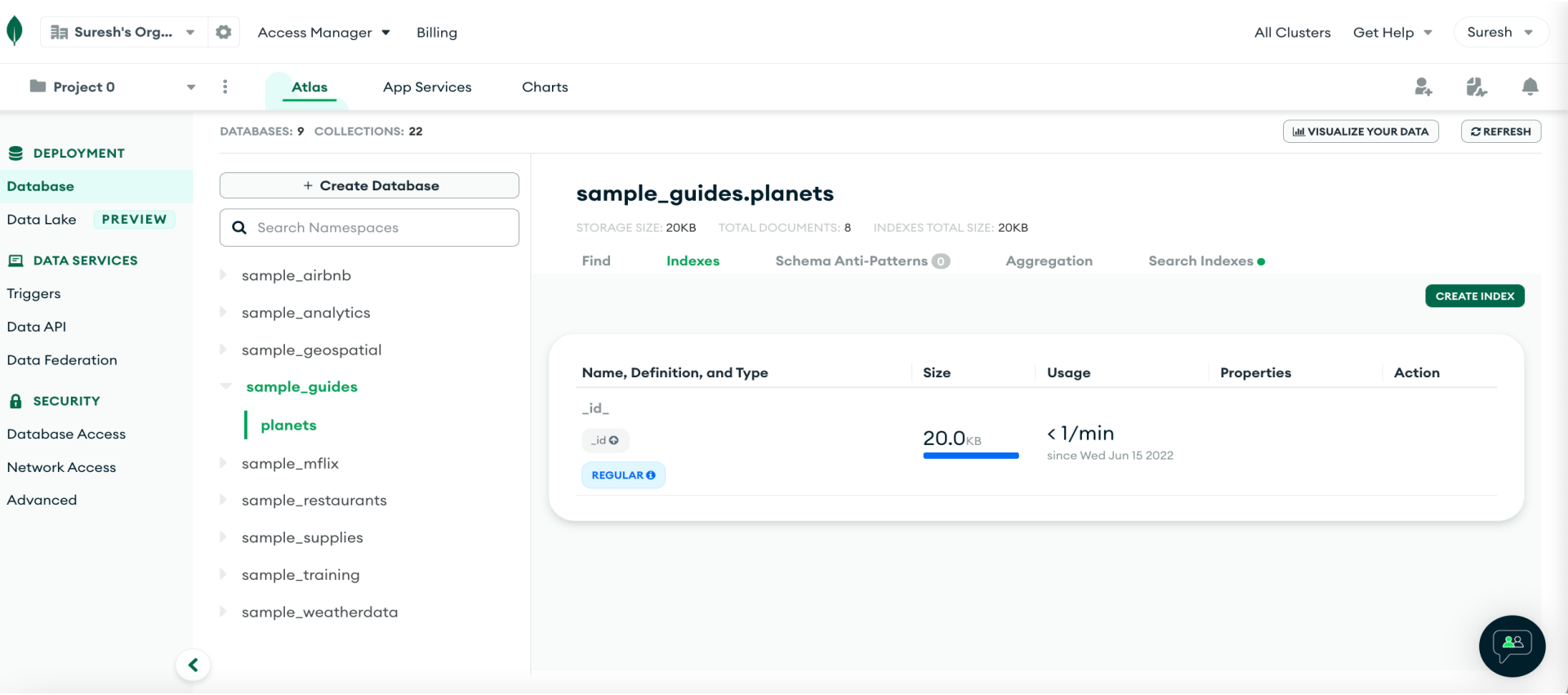
dr.suresha — mongosh mongodb+srv://cluster0.tyqtb.mongodb.net/myFirstDatabase — myFirstDatabase PATH=/opt/homebrew/opt/node@14/bin:/Users/dr.suresha/opt/anaconda3/bin:/U...

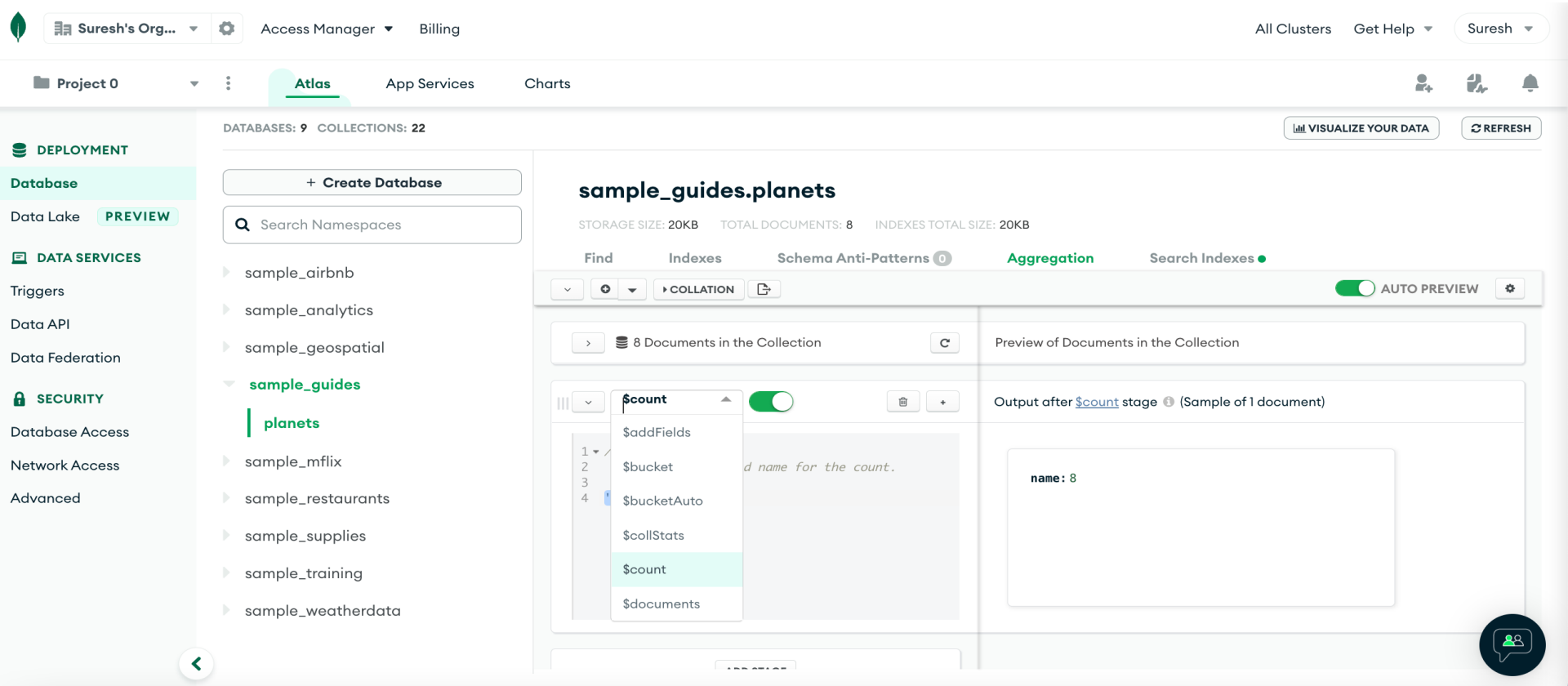
```
Last login: Mon Jun 13 08:58:00 on ttys000
(base) dr.suresha@Drsuresh-A-MacBook-Air ~ % mongosh "mongodb+srv://cluster0.tyqtb.mongodb.net/myFirstDatabase" --apiVersion 1 --username itsuresha
Enter password: ***********
Current Mongosh Log ID: 62aa6404445f5f2b6014942c
Connecting to:          mongodb+srv://cluster0.tyqtb.mongodb.net/myFirstDatabase?appName=mongosh+1.3.1
Using MongoDB:          5.0.9 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
Atlas atlas-mhgt6m-shard-0 [primary] myFirstDatabase>
```

```
dr.suresha — mongosh mongodb+srv://cluster0.tyqtb.mongodb.net/myFirstDatabase — myFirstDatabase PATH=/opt/homebrew/opt/node@14/bin:/Users/dr.suresha/opt/anaconda3/bin:/U...
Atlas atlas-mhgt6m-shard-0 [primary] sample_airbnb> show dbs
sample_airbnb       58.8 MB
sample_analytics    9.57 MB
sample_geospatial   1.45 MB
sample_guides         41 kB
sample_mflix        52.8 MB
sample_restaurants  9.03 MB
sample_supplies     1.17 MB
sample_training     60.4 MB
sample_weatherdata  2.92 MB
admin                344 kB
local               3.32 GB
Atlas atlas-mhgt6m-shard-0 [primary] sample_airbnb> use sample_airbnb
already on db sample_airbnb
Atlas atlas-mhgt6m-shard-0 [primary] sample_airbnb> show collections
listingsAndReviews
Atlas atlas-mhgt6m-shard-0 [primary] sample_airbnb> use sample_guides
switched to db sample_guides
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides> show collections
planets
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides> db.planets.find()
[
  {
    _id: ObjectId("621ff30d2a3e781873fcb65c"),
```


Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
SURESH'S ORG - 2022-06-15 > PROJECT 0 > DATABASES
Cluster0
VERSION
5.0.9
REGION
AWS Mumbai (ap-south-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 9 COLLECTIONS: 22
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
sample_airbnb
listingsAndReviews
sample_analytics
sample_geospatial
sample_guides
sample_mflix
sample_restaurants
sample_supplies
sample_airbnb.listingsAndReviews
STORAGE SIZE: 55.42MB
TOTAL DOCUMENTS: 5555
INDEXES TOTAL SIZE: 632KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER
{ field: 'value' }
OPTIONS
Apply
Reset
QUERY RESULTS: 1-20 OF MANY
_id: "10006546"
listing_url: "https://www.airbnb.com/rooms/10006546"
name: "Ribeira Charming Duplex"
summary: "Fantastic duplex apartment with three bedrooms, located in the histori..."

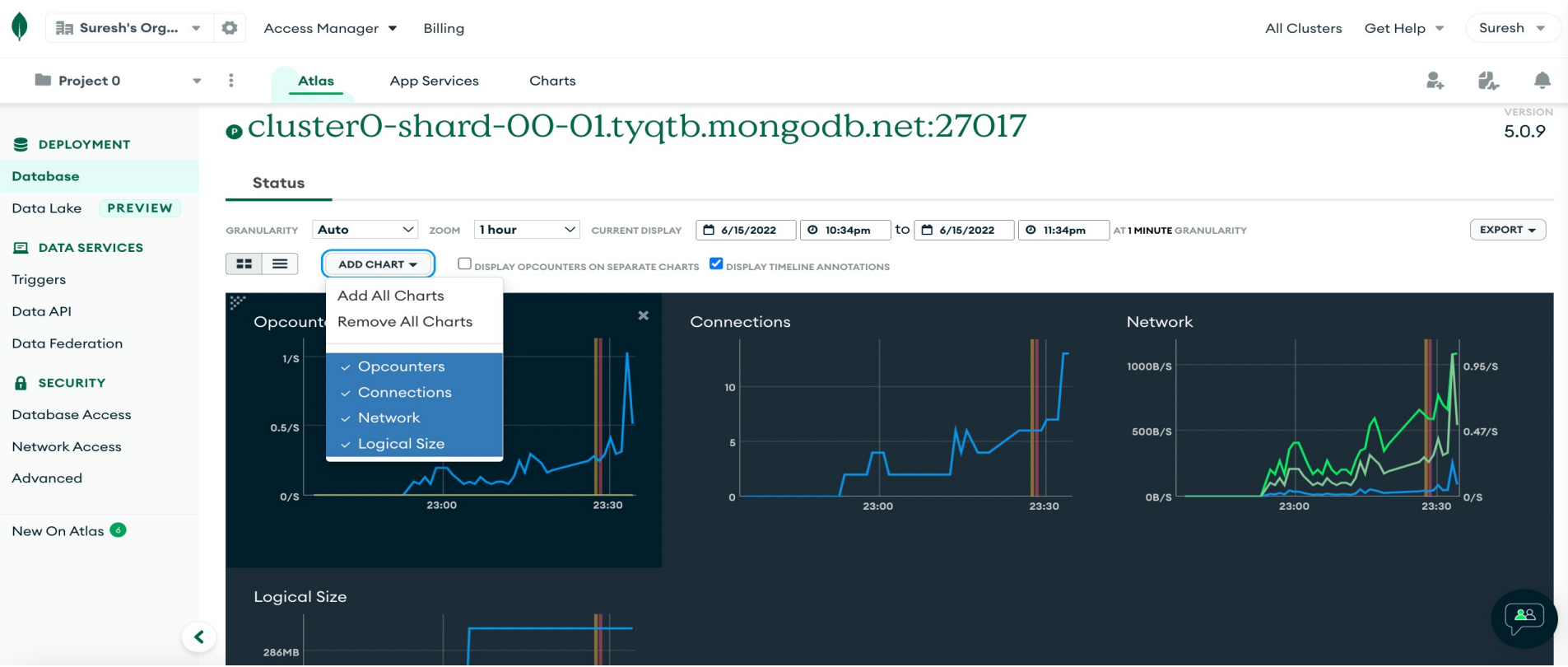

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
DATABASES: 9
COLLECTIONS: 22
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
planets
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
sample_guides.planets
STORAGE SIZE: 20KB
TOTAL DOCUMENTS: 8
INDEXES TOTAL SIZE: 20KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER { field: 'value' }
OPTIONS
PROJECT { field: 0 }
SORT { field: -1 }
COLLATION { locale: 'simple' }
Apply
Reset
QUERY RESULTS: 1-8 OF 8
_id: ObjectId("621ff30d2a3e781873fcb65c")
name: "Mercury"
orderFromSun: 1
hasRings: false
mainAtmosphere: Array
surfaceTemperatur… : Object

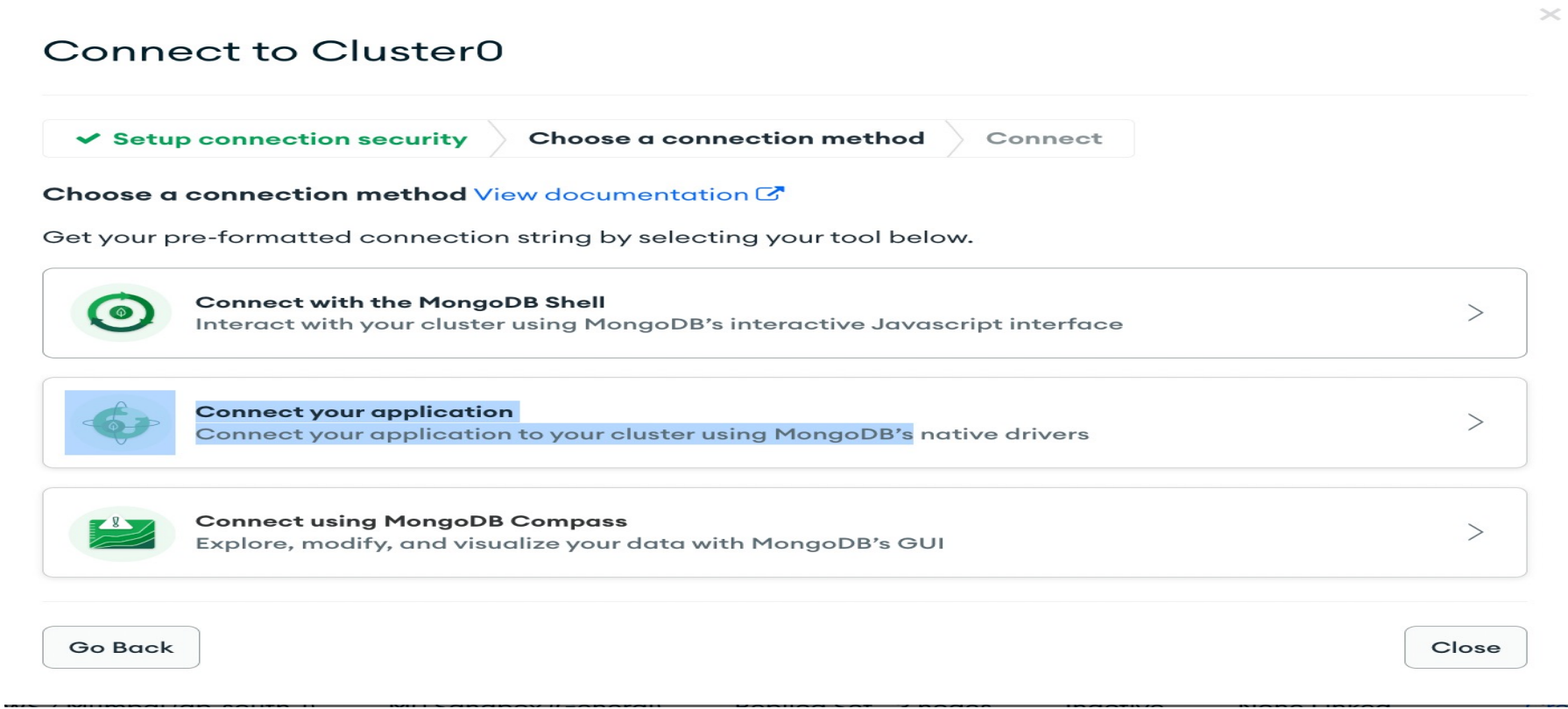

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
DATABASES: 9 COLLECTIONS: 22
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
planets
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
sample_guides.planets
STORAGE SIZE: 20KB
TOTAL DOCUMENTS: 8
INDEXES TOTAL SIZE: 20KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
CREATE INDEX
Name, Definition, and Type
Size
Usage
Properties
Action
_id_
_id
REGULAR
20.0KB
< 1/min
since Wed Jun 15 2022

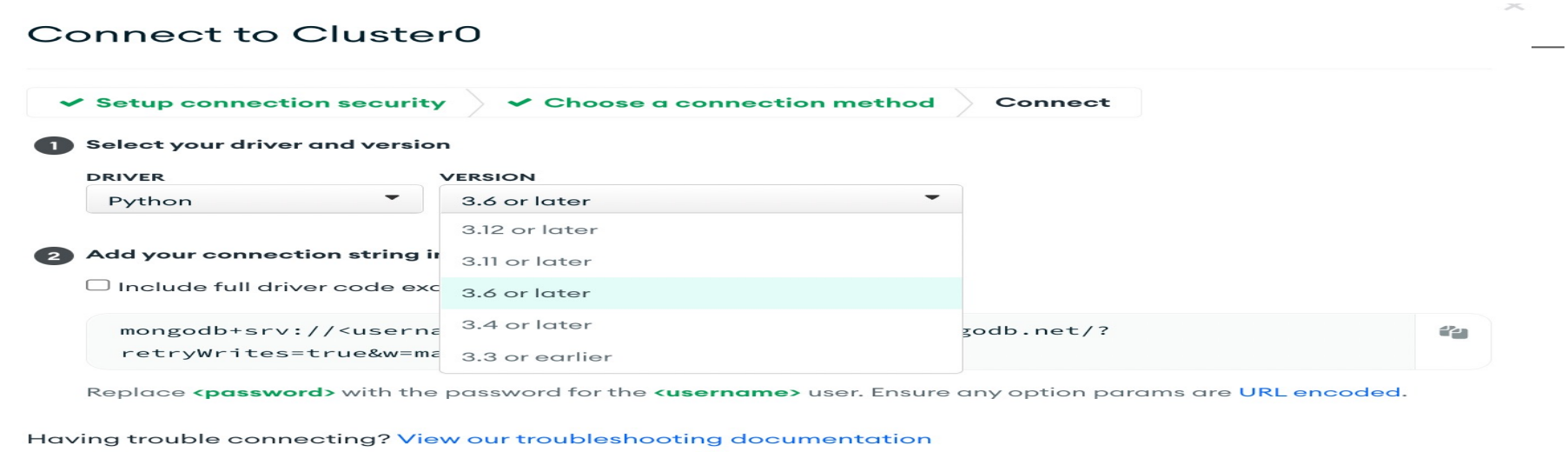

Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
DATABASES: 9 COLLECTIONS: 22
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
planets
sample_mflix
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
sample_guides.planets
STORAGE SIZE: 20KB
TOTAL DOCUMENTS: 8
INDEXES TOTAL SIZE: 20KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
COLLATION
AUTO PREVIEW
8 Documents in the Collection
Preview of Documents in the Collection
$count
$addFields
$bucket
$bucketAuto
$collStats
$count
$documents
Output after $count stage (Sample of 1 document)
name: 8

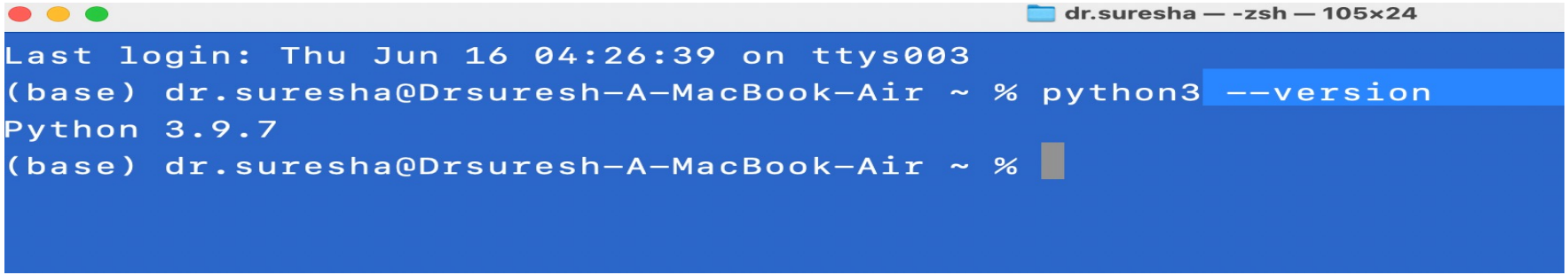
```
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides> db.planets.insert({name:"new_name"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62aa6a44445f5f2b6014942d") }
}
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides> db.planets.insert({name:"new_name"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62aa6a7a445f5f2b6014942e") }
}
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides>
```

```
Atlas atlas-mhgt6m-shard-0 [primary] sample_guides> db.planets.aggregate([{$group:{"_id":{"name":"$name"}}
,"count":{$sum:1},"first_doc":{$first:"$name"}}}])
[
  { _id: { name: 'Earth' }, count: 1, first_doc: 'Earth' },
  { _id: { name: 'Mars' }, count: 1, first_doc: 'Mars' },
  { _id: { name: 'new_name' }, count: 2, first_doc: 'new_name' },
  { _id: { name: 'Uranus' }, count: 1, first_doc: 'Uranus' },
  { _id: { name: 'Saturn' }, count: 1, first_doc: 'Saturn' },
  { _id: { name: 'Jupiter' }, count: 1, first_doc: 'Jupiter' },
  { _id: { name: 'Neptune' }, count: 1, first_doc: 'Neptune' },
  { _id: { name: 'Venus' }, count: 1, first_doc: 'Venus' },
  { _id: { name: 'Mercury' }, count: 1, first_doc: 'Mercury' }
```


Suresh's Org...
Access Manager
Billing
All Clusters
Get Help
Suresh
Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas
clusterO-shard-00-01.tyqtb.mongodb.net:27017
VERSION
5.0.9
Status
GRANULARITY
Auto
ZOOM
1 hour
CURRENT DISPLAY
6/15/2022
10:34pm
to
6/15/2022
11:34pm
AT 1 MINUTE GRANULARITY
EXPORT
ADD CHART
DISPLAY OPCOUNTERS ON SEPARATE CHARTS
DISPLAY TIMELINE ANNOTATIONS
Add All Charts
Remove All Charts
Opcounters
Connections
Network
Logical Size
Opcount
1/S
0.5/S
0/S
23:00
23:30
Connections
10
5
0
23:00
23:30
Network
1000B/S
500B/S
0B/S
0.95/S
0.47/S
0/S
23:00
23:30
Logical Size
286MB


Connect to Cluster0
Setup connection security
Choose a connection method
Connect
Choose a connection method View documentation
Get your pre-formatted connection string by selecting your tool below.
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface
Connect your application
Connect your application to your cluster using MongoDB's native drivers
Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI
Go Back
Close


Connect to Cluster0
Setup connection security
Choose a connection method
Connect
1 Select your driver and version
DRIVER
Python
VERSION
3.6 or later
3.12 or later
3.11 or later
3.6 or later
3.4 or later
3.3 or earlier
2 Add your connection string i
Include full driver code exc
mongodb+srv://<userna
godb.net/?
retryWrites=true&w=ma
Replace <password> with the password for the <username> user. Ensure any option params are URL encoded.
Having trouble connecting? View our troubleshooting documentation


dr.suresha — -zsh — 105×24
Last login: Thu Jun 16 04:26:39 on ttys003
(base) dr.suresha@Drsuresh-A-MacBook-Air ~ % python3 --version
Python 3.9.7
(base) dr.suresha@Drsuresh-A-MacBook-Air ~ %

MySQL vs Mongo DB

MySQL

MySQL is a popular, free-to-use, and open-source relational database management system (RDBMS) developed by Oracle. As with other relational systems, MySQL stores data using tables and rows, enforces referential integrity and uses structured query language (SQL) for data access. When users need to retrieve data from a MySQL database, they must construct an SQL query that joins multiple tables together to create the view on the data they require.

Database schemas and data models need to be defined ahead of time, and data must match this schema to be stored in the database. This rigid approach to storing data offers some degree of safety, but trades this for flexibility. If a new type or format of data needs to be stored in the database, schema migration must occur, which can become complex and expensive as the size of the database grows.

MySQL vs MongoDB

MongoDB

MongoDB is also free to use and open source; however, its design principles differ from traditional relational systems. Often styled as a non-relational (or NoSQL) system, MongoDB adopts a significantly different approach to storing data, representing information as a series of JSON-like documents (actually stored as binary JSON, or BSON), as opposed to the table and row format of relational systems.

MongoDB documents consist of a series of key/value pairs of varying types, including arrays and nested documents; however, the key difference is that the structure of the key/value pairs in a given collection can vary from document to document. This more flexible approach is possible because documents are self-describing.

NoSQL DB

What is a NoSQL database?

When people use the term “NoSQL database,” they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

Brief history of NoSQL databases

NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

NoSQL DB

Types of NoSQL databases

Over time, four major types of NoSQL databases emerged: document databases, key-value databases, wide-column stores, and graph databases.

- **Document databases** store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.
- **Key-value databases** are a simpler type of database where each item contains keys and values.
- **Wide-column stores** store data in tables, rows, and dynamic columns.
- **Graph databases** store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

NoSQL DB

When should NoSQL be used?

When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:

- Fast-paced Agile development
- Storage of structured and semi-structured data
- Huge volumes of data
- Requirements for scale-out architecture
- Modern application paradigms like microservices and real-time streaming

See When to Use NoSQL Databases and Exploring NoSQL Database Examples for more detailed information on the reasons listed above.

NoSQL database misconceptions

Over the years, many misconceptions about NoSQL databases have spread throughout the developer community. In this section, we'll discuss two of the most common misconceptions:

- Relationship data is best suited for relational databases.
- NoSQL databases don't support ACID transactions.

When to choose?

The core differences between these two database systems are significant. Choosing which one to use is really a question of approach rather than purely a technical decision.

MySQL is a mature relational database system, offering a familiar database environment for experienced IT professionals.

MongoDB is a well-established, non-relational database system offering improved flexibility and horizontal scalability, but at the cost of some safety features of relational databases, such as referential integrity.

Which one should you choose?

In the following sections, we're going to look at some of the different considerations when deciding between MongoDB and MySQL.

User Friendliness?

MongoDB is an attractive option to developers. Its data storage philosophy is simple and immediately understandable to anybody with programming experience.

MongoDB stores data in collections with no enforced schema. This flexible approach to storing data makes it particularly suitable for developers who may not be database experts, yet want to use a database to support the development of their applications.

Compared to MySQL, this flexibility is a significant advantage: To get the best out of a relational database, you must first understand the principles of normalization, referential integrity, and relational database design.

With the ability to store documents of varying schemas, including unstructured data sets, MongoDB provides a flexible developer interface for teams that are building applications that don't need all of the safety features offered by relational systems. A common example of such an application is a web application that doesn't depend on structured schemas; it can easily serve unstructured, semi-structured, or structured data, all from the same MongoDB collection.

Scalability

A key benefit of the MongoDB design is that the database is extremely easy to scale. Configuring a **sharded cluster** allows a portion of the database, called a shard, to also be configured as a replica set. In a sharded cluster, data is distributed across many servers. This highly flexible approach allows MongoDB to horizontally scale both read and write performance to cater to applications of any scale.

A **replica set** is the replication of a group of MongoDB servers that hold the same data, ensuring high availability and disaster recovery.

With a MySQL database system, options for scalability are much more limited. Typically, you have two choices: vertical scalability, or adding read replicas. Scaling vertically involves adding more resources to the existing database server, but this has an inherent upper limit.

Read replication involves adding read-only copies of the database to other servers. However, this is typically limited to five replicas in total, which can only be used for read operations. This can cause issues with applications that are either write-heavy, or write and read regularly for the database, since it's common for replicas to lag behind the write master. Multi-master replication support has been added to MySQL, but its implementation is more limited than the functionality available in MongoDB.

Performance

Assessing the performance of two completely different database systems is very difficult, since both management systems approach the task for data storage and retrieval in completely different ways. While it's possible to directly compare two SQL databases with a set of standard SQL benchmarks, achieving the same across non-relational and relational databases is much more difficult and subjective.

For example: MySQL is optimized for high performance joins across multiple tables that have been appropriately indexed. In MongoDB, joins are supported with the **$lookup** operation, but they are less needed due to the way MongoDB documents tend to be used; they follow a hierarchical data model and keep most of the data in one document, therefore eliminating the need for joins across multiple documents.

MongoDB is also optimized for write performance, and features a specific **insertMany()** API for rapidly inserting data, prioritizing speed over transaction safety wherein MySQL data needs to be inserted row by row.

Observing some of the high-level query behaviors of the two systems, we can see that MySQL is faster at selecting a large number of records, while MongoDB is significantly faster at inserting or updating a large number of records.

Flexibility

This is an easy one, and a hands down win for MongoDB. The schemaless design of MongoDB documents makes it extremely easy to build and enhance applications over time, without needing to run complex and expensive schema migration processes as you would with a relational database.

With MongoDB, there are more dynamic options for updating the schema of a collection, such as creating new fields based on an **aggregation pipeline** or updating nested array fields. This benefit is particularly important as databases grow in size. In contrast, larger MySQL databases are slower to migrate schemas and stored procedures that can be dependent on the updated schemas. MongoDB's flexible design makes this much less of a concern.

It's worth pointing out that both databases have a lot in common. Both are free to get started with, both are easy to install on Linux and Windows, and both have wide programming language support for popular languages like Java, **node.js**, and Python.

In addition, MongoDB offers MongoDB Atlas, a managed cloud solution which is also forever free to use for exploratory purposes, while for a MySQL managed cloud version, you would need to have an account with one of the major public cloud providers and fall within their free tier terms in order to not pay.

Security & Conclusion

MongoDB leverages the popular role-based access control model with a flexible set of permissions. Users are assigned to a role, and that role grants them specific permissions over data sets and database operations. All communication is encrypted with TLS, and it's possible to write encrypted documents to MongoDB data collections using a master key which is never available to MongoDB, achieving encryption of data at rest.

MySQL supports the same encryption features as MongoDB; its authentication model is also similar. Users can be granted roles but also privileges, giving them permissions over particular database operations and against particular data sets.

Why is using MongoDB better than using MySQL?

Which query language does each database use?

Is MongoDB faster than MySQL?

Does MySQL support JSON documents?

References

https://www.mongodb.com/what-is-mongodb

https://www.mongodb.com/compare/mongodb-mysql

https://studio3t.com/academy/lessons/mongodb-basics/

Q&A ?

Thank You