Lecture 10 - MongoDB security

Instructor: Suresh Melvin Sigera
Email: suresh.sigera@cuny.csi.edu

Agenda

- Role based access control
- Roles
- Creating a user
- Build-in roles
- Assigning roles to user and databases
- QA

Role based access control

Most RDBMSs feature elaborate security subsystems, allowing authorization of users and groups with fine-grained control over permissions. By contrast, MongoDB v2.0 supports only a simple, per-database authentication mechanism.

This makes the security of the machines on which MongoDB is run all the more important.

Secure environments

MongoDB, like all databases, should be run in a secure environment. Production users of MongoDB must take advantage of the security features of **modern operating systems** to ensure the safety of their data. Probably the most important of these features is the firewall.

The only potential difficulty in using a firewall with MongoDB is knowing which machines need to communicate with each other. Fortunately, the communication rules are simple. With a replica set, each node must be able to reach every other node. In addition, all database clients must be able to connect with every replica set node that the client might conceivably talk to.

Setting up authentication

One of the first priorities for systems administrators is to ensure their system is secure. The best way to handle security with MongoDB is to run it in a trusted environment, ensuring that only trusted machines are able to connect to the server. That said, **MongoDB supports per-connection authentication**, albeit with a fairly coarse-grained permissions scheme.

Authentication basics

Each database in a MongoDB **instance can have any number of users**. When security is enabled, only authenticated users of a database are able to perform read or write operations.

There are two special databases: users in the **admin** and **local** databases can perform operations on any database. A user that belongs to either one of these databases can be thought of as a superuser. After authenticating, admin users are able to read or write from any database and are able to perform certain admin-only commands, such as listDatabases or shutdown.

Before starting the database with security turned on, it's important that at least one admin user has been added. Let's run through a quick example, starting from a shell connected to a server without authentication turned on:

Mongo roles simplified

Four main built-in roles to use

- read
- readWrite
  - Includes read
- dbOwner
  - Includes readWrite, dbAdmin, userAdmin (db level)
- dbAdminAnyDatabase
  - dbAdmin on all DBs but not user admin or readWrite
- root
  - Super user, but with limits*

The missing roles and their issues

Root is not root

- Unable to change system collection!
- Some commands are still not available to this role
- Need an “any/any” role for true “Super User” (but never let applications use it)

Authentication basics

Before starting the database with security turned on, it’s important that at least one admin user has been added. Let’s run through a quick example, starting from a shell connected to a server without authentication turned on.

use **admin**
switched to db admin

db.addUser("root", "abcd");
{
    "user" : "root",
    "readOnly" : false,
    "pwd" : "1a0f1c3c3aa1d592f490a2addc559383"
}

```
> use test
switched to db test
> db.addUser("test_user", "efgh");
{
    "user" : "test_user",
    "readOnly" : false,
    "pwd" : "6076b96fc3fe6002c810268702646eec"
}
> db.addUser("read_user", "ijkl", true);
{
    "user" : "read_user",
    "readOnly" : true,
    "pwd" : "f497e180c9dc0655292fee5893c162f1"
}
```

Authentication basics

Here we've added an admin user, root, and two users on the test database. One of those users, "read_only", has read permissions only and cannot write to the database. From the shell, a read-only user is created by passing true as the third argument to addUser. To call addUser, you must have write permissions for the database in question; in this case we can call addUser on any database because we have not enabled security yet.

Authentication basics

Note: The addUser method is useful for more than just adding new users: it can be used to change a user’s password or read-only status. Just call addUser with the username and a new password or read-only setting for the user.

Different types of DB users

Administrators	Developer/Your app	Data scientist
Must be able to manage DB, and create users	Must be able to perform CRUD	Must be able to fetch data
Does not need to be able insert data, fetch data	Does not need to be able create users or manage DB configuration	Does not need to be able create users or manage the database configuration or insert, edit or delete

Creating an admin user

Creating a user administrator in MongoDB is done by using the createUser method. The following example shows how this can be done.

```
use admin

db.createUser( {
        user: "Suresh", pwd: "Sigera",
        roles:[{role: "userAdminAnyDatabase" ,
        db:"admin"}
]})
```

Creating an admin user

```
Successfully added user: {
        "user" : "Suresh",
        "roles" : [
                {
                        "role" : "userAdminAnyDatabase",
                        "db" : "admin"
                }
        ]
}
```

Creating an admin user

- The first step is to specify the "username" and "password" which needs to be created
- The second step is to assign a role for the user. Since it needs to be a database administrator in which case we have assigned to the "userAdminAnyDatabase" role. This role allows the user to have administrative privileges to all databases in MongoDB
- The db parameter specifies the admin database which is a special Meta database within MongoDB which holds the information for this user

Creating user for single database

To create a user who will manage a single database, we can use the same command as mentioned above but we need to use the "**userAdmin**" option only. The following example shows how this can be done;

```
db.createUser( {
        user: "Employeeadmin", pwd: "password",
        roles:[{role: "userAdmin" , db:"Employee"}]
})
```

Creating user for single database

```
db.createUser( {
        user: "Employeeadmin", pwd: "password",
        roles:[{role: "userAdmin" , db:"Employee"}]
})
```

- The first step is to specify the "username" and "password" which needs to be created
- The second step is to assign a role for the user which in this case since it needs to be a database administrator is assigned to the "userAdmin" role. This role allows the user to have administrative privileges only to the database specified in the db option
- The db parameter specifies the database to which the user should have administrative privileges on

Managing users

First understand the roles which you need to define. There is a whole list of role available in MongoDB. For example, there is a the "read role" which only allows read only access to databases and then there is the "readwrite" role which provides read and write access to the database , which means that the user can issue the insert, delete and update commands on collections in that database.

```
db.createUser( {
        user: "Tom", pwd: "Cruise",
        roles:[
                { role: "read" , db:"MI3"},
                { role: "readWrite" , db:"JackReacher"}
         ]
 })
```

The above code snippet shows that a user called Tom is created, and he is assigned multiple roles in multiple databases. In the above example, he is given read only permission to the "MI3" database and readWrite permission to the "JackReacher" database.

```
db.createUser( {
        user: "Tom", pwd: "Cruise",
        roles:[
                { role: "read" , db:"MI3"},
                { role: "readWrite" , db:"JackReacher"}
         ]
 })
```

The above code snippet shows that a user called Tom is created, and he is assigned multiple roles in multiple databases. In the above example, he is given read only permission to the "MI3" database and readWrite permission to the "JackReacher" database.

Build-in roles

Database User
read readWrite

Database Admin
dbAdmin userAdmin dbOwner

All database roles
readAnyDatabase readWriteAnyDatabase userAdminAnyDatabase dbAdminAnyDatabase

Cluster Admin
clusterManager clusterMonitor hostManager clusterAdmin

Backup/ Restore
backup restore

Superuser
dbOwner (admin) userAdmin (admin) userAdminAnyDatabase root

```
db.createUser({
        user: "student", pwd: "student",
        roles:[
                "readWrite"
        ]
 })

db.auth('student,'student')
```

QA