01 Autenticación

Authentication Methods

To authenticate as a user, you must provide a username, password, and the authentication database associated with that user.

To authenticate using the mongo shell, either:

- Use the mongo command-line authentication options (--username, --password, and --authenticationDatabase) when connecting to the mongod or mongos instance, or

- Connect first to the mongod or mongos instance, and then run the authenticate command or the db.auth() method against the authentication database.

Authentication Mechanisms

MongoDB supports a number of authentication mechanisms that clients can use to verify their identity. These mechanisms allow MongoDB to integrate into your existing authentication system.

MongoDB supports multiple authentication mechanisms:

- SCRAM (Default)

- x.509 Certificate Authentication.

In addition to supporting the aforementioned mechanisms, MongoDB Enterprise also supports the following mechanisms:

- LDAP proxy authentication, and

- Kerberos authentication.

Internal Authentication

In addition to verifying the identity of a client, MongoDB can require members of replica sets and sharded clusters to authenticate their membership to their respective replica set or sharded cluster.

Authentication on Sharded Clusters

In sharded clusters, clients generally authenticate directly to the mongos instances. However, some maintenance operations may require authenticating directly to a specific shard.

Users

Authentication Database

When adding a user, you create the user in a specific database. This database is the authentication database for the user.

A user can have privileges across different databases; that is, a user's privileges are not limited to their authentication database. By assigning to the user roles in other databases, a user created in one database can have permissions to act on other databases.

The user's name and authentication database serve as a unique identifier for that user. That is, if two users have the same name but are created in different databases, they are two separate users. If you intend to have a single user with permissions on multiple databases, create a single user with roles in the applicable databases instead of creating the user multiple times in different databases.

Authenticate a User

To authenticate as a user, you must provide a username, password, and the authentication database associated with that user.

IMPORTANT Authenticating multiple times as different users does not drop the credentials of previously-authenticated users. This may lead to a connection having more permissions than intended by the user, and causes operations within a logical session to raise an error.

Método createUser()

Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user error if the user already exists on the database.

The db.createUser() method has the following syntax:

Field	Type	Description
user	document	The document with authentication and access information

about the user to create.

writeConcern document Optional. The level of write concern for the creation operation.

The writeConcern document takes the same fields as the getLastError command.

The user document defines the user and has the following form:

TIP Starting in version 4.2 of the mongo shell, you can use the passwordPrompt() method in conjunction with various user authentication/management methods/commands to prompt for the password instead of specifying the password directly in the method/command call. However, you can still specify the password directly as you would with earlier versions of the mongo shell.

```
{
  user: "<name>",
  pwd: passwordPrompt(),      // Or  "<cleartext password>"
  customData: { <any information> },
  roles: [
    { role: "<role>", db: "<database>" } | "<role>",
    ...
  ],
  authenticationRestrictions: [
     {
      clientSource: ["<IP>" | "<CIDR range>", ...],
      serverAddress: ["<IP>" | "<CIDR range>", ...]
     },
     ...
  ],
  mechanisms: [ "<SCRAM-SHA-1|SCRAM-SHA-256>", ... ],
  passwordDigestor: "<server|client>"
}
```

Práctica

```
use maraton

> db.createUser({
... user: "pedro",
... pwd: "pedro4321",
... customData: {dni: "90876452P"},
... roles: [
... {role: "clusterAdmin", db: "admin"},
... "readWrite"
... ]
... })
```

```
Successfully added user: {
	"user" : "pedro",
	"customData" : {
		"dni" : "90876452P"
	},
	"roles" : [
		{
			"role" : "clusterAdmin",
			"db" : "admin"
		},
		"readWrite"
	]
}
```

Centralized User Data

For users created in MongoDB, MongoDB stores all user information, including name, password, and the user's authentication database, in the system.users collection in the admin database. Do not access this collection directly but instead use the user management commands.

```
> use admin
switched to db admin
> show collections
system.users
system.version
> db.system.users.find()

> db.system.users.find().pretty()
{
	"_id" : "admin.pedro",
	"user" : "pedro",
	"db" : "admin",
	"credentials" : {
		"SCRAM-SHA-1" : {
			"iterationCount" : 10000,
			"salt" : "BsCGjO2AZkCho8dxLfkyjA==",
			"storedKey" : "0OHJb5bp747CBaJLhbd4be5yU48=",
			"serverKey" : "9Thb493E0Z/hsHMV/js+KDrRGW4="
		}
	},
	"roles" : [
		{
			"role" : "readWrite",
			"db" : "certi"
```

```
                },
                {
                        "role" : "readWriteAnyDatabase",
                        "db" : "admin"
                },
                {
                        "role" : "read",
                        "db" : "certi"
                },
                {
                        "role" : "userAdminAnyDatabase",
                        "db" : "admin"
                }
        ]
}
{
        "_id" : "admin.Su",
        "user" : "Su",
        "db" : "admin",
        "credentials" : {
                "SCRAM-SHA-1" : {
                        "iterationCount" : 10000,
                        "salt" : "loFsjZ45wDue8xETe9xeJA==",
                        "storedKey" : "i5zOLYqT01YnsdJzyN4GpNsZL3I=",
                        "serverKey" : "Hh9mY9M7rbJlAR0MSWk3y+FT6vs="
                }
        },
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "admin"
                }
        ]
}
{
        "_id" : "maraton.pedro",
        "userId" : UUID("b064035f-295b-4712-b068-86151db4ad91"),
        "user" : "pedro",
        "db" : "maraton",
        "credentials" : {
                "SCRAM-SHA-1" : {
                        "iterationCount" : 10000,
                        "salt" : "RB7dkElhzLhyQRauX4KTgQ==",
                        "storedKey" : "y4QMtZEYddh+hi9uoDnsafVeno8=",
                        "serverKey" : "fSRaPhkZhaOgD8+WLwridXVhp4o="
                },
```

```
                "SCRAM-SHA-256" : {
                        "iterationCount" : 15000,
                        "salt" : "TZGmr2GHoRJoW6CyoKzfkAdyF081QX2QFUKnDg==",
                        "storedKey" :
"IIw1d/zK6CDMU/BxlmfH5lV6nUvqv9MzHZzmbcb5/n8=",
                        "serverKey" :
"DZOQHkMpTVWtrC9dJevcgcc/XTIiF69Dus5OslnE4lU="
                }
        },
        "customData" : {
                "dni" : "90876452P"
        },
        "roles" : [
                {
                        "role" : "clusterAdmin",
                        "db" : "admin"
                },
                {
                        "role" : "readWrite",
                        "db" : "maraton"
                }
        ]
}
```


Enable Access Control

Enabling access control on a MongoDB deployment enforces authentication, requiring users to identify themselves. When accessing a MongoDB deployment that has access control enabled, users can only perform actions as determined by their roles.

With access control enabled, ensure you have a user with userAdmin or userAdminAnyDatabase role in the admin database. This user can administrate user and roles such as: create users, grant or revoke roles from users, and create or modify customs roles.

Práctica

use admin

```
> db.createUser({
... user: "superAdmin",
... pwd: "NoHay2sin3",
... roles: [
... {role: "userAdminAnyDatabase", db: "admin"},
```

... "readWriteAnyDatabase"
...]
... })
Successfully added user: {
	"user" : "superAdmin",
	"roles" : [
		{
			"role" : "userAdminAnyDatabase",
			"db" : "admin"
		},
		"readWriteAnyDatabase"
	]
}

Y ahora reiniciamos el servidor con la opción --auth

mongod --auth

Para conectar una shell ahora:

MacBook-Pro-de-Pedro:~ pedro$ mongo --port 27017 --authenticationDatabase "admin" -u "superAdmin" -p

MongoDB shell version v4.2.3
Enter password: ********
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("93e01f37-5b83-4791-92fd-cbd3b71101cf") }
MongoDB server version: 4.2.3
> use maraton
switched to db maraton
> db.participantes.find()
{ "_id" : 0, "nombre" : "Juan", "apellido1"...

También se puede autenticar a posteriori desde la base de datos con el siguiente método:

```
MacBook-Pro-de-Pedro:~ pedro$ mongo
MongoDB shell version v4.2.3
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ed463a97-69a0-4e84-81e7-4cb73c336150") }
MongoDB server version: 4.2.3

> use maraton
switched to db maraton
> db.auth("pedro","pedro4321")
1
```