

## Lab 14:

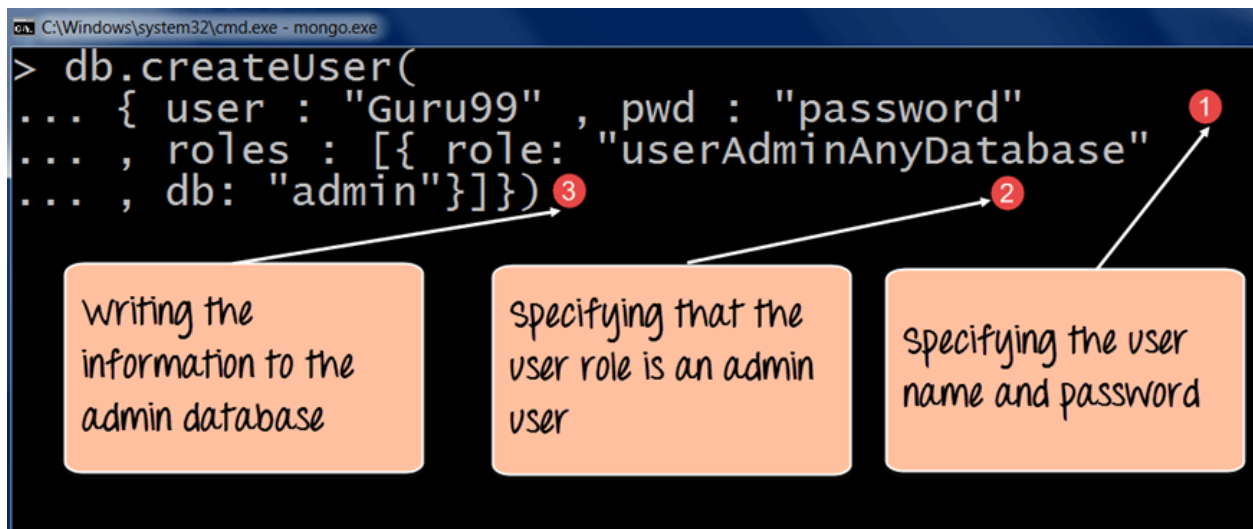
### Create User & add Role in MongoDB

#### MongoDB Create Administrator 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: "demo",
  pwd: "abc123",

  roles:[{role: "userAdminAnyDatabase" , db:"admin"}, "readWriteAnyDatabase"]}
)
```



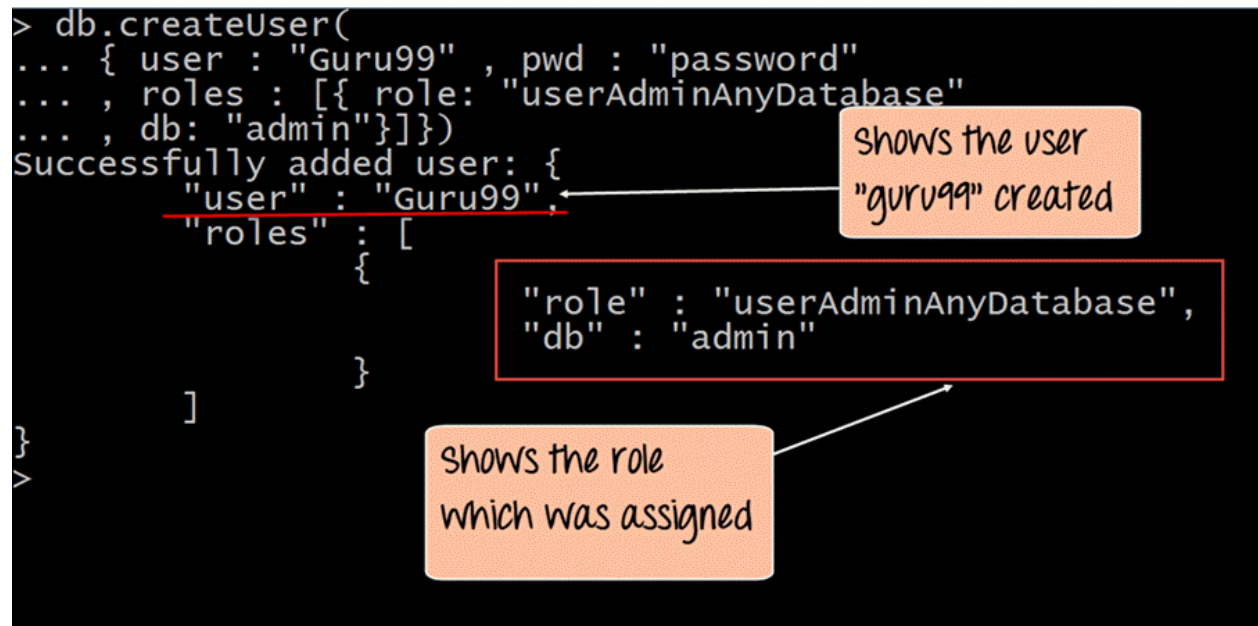
#### Code Explanation:

1. The first step is to specify the "username" and "password" which needs to be created.
2. 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.
3. The db parameter specifies the admin database which is a special Meta database within MongoDB which holds the information for this user.

If the command is executed successfully, the following Output will be shown:

### Output:

```
> db.createUser(
... { user : "Guru99" , pwd : "password"
... , roles : [{ role: "userAdminAnyDatabase"
... , db: "admin"}]})
Successfully added user: {
  "user" : "Guru99",
  "roles" : [
    {
      "role" : "userAdminAnyDatabase",
      "db" : "admin"
    }
  ]
}
```

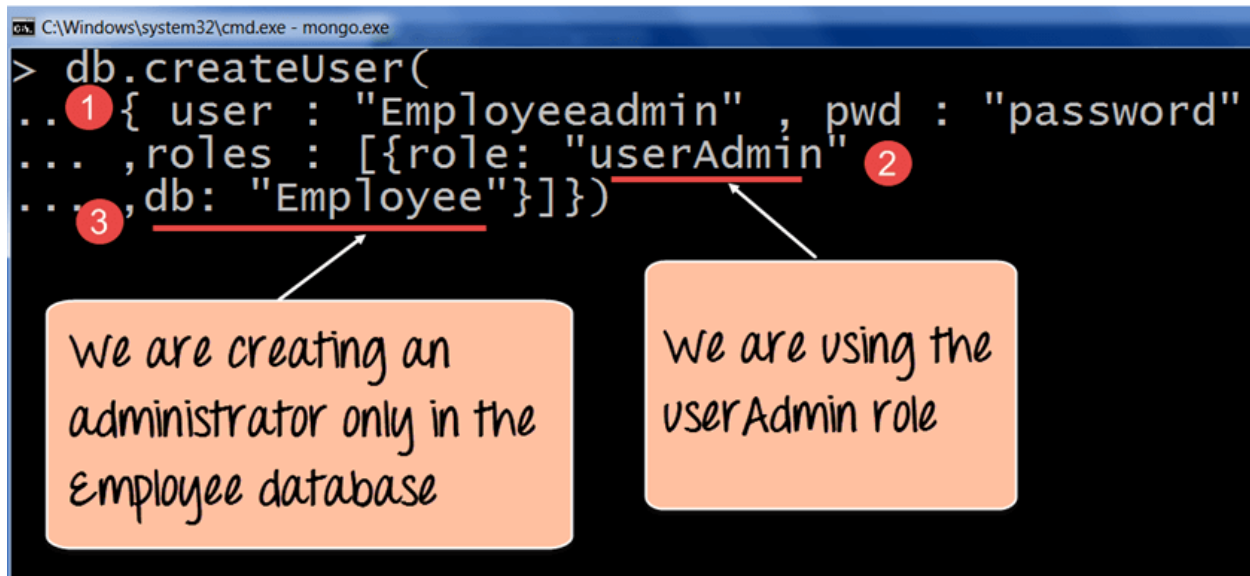


The output shows that a user called "Guru99" was created and that user has privileges over all the databases in MongoDB.

### MongoDB Create 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;



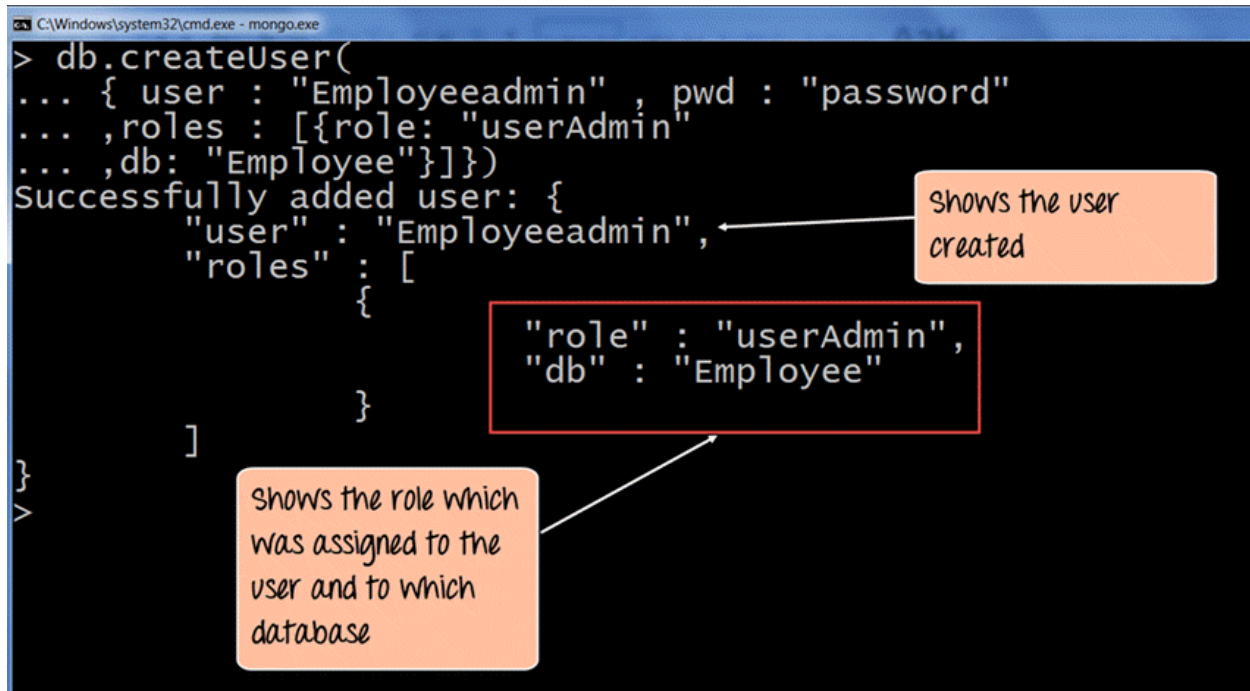
```
db.createUser(  
{  
    user: "Employeeadmin",  
  
    pwd: "password",  
  
    roles:[{role: "userAdmin" , db:"Employee"}]})
```

### Code Explanation:

1. The first step is to specify the "username" and "password" which needs to be created.
2. 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.
3. The db parameter specifies the database to which the user should have administrative privileges on.

If the command is executed successfully, the following Output will be shown:

### Output:



```
C:\Windows\system32\cmd.exe - mongo.exe
> db.createUser(
... { user : "Employeeadmin" , pwd : "password"
... ,roles : [{role: "userAdmin"
... ,db: "Employee"}]})
Successfully added user: {
  "user" : "Employeeadmin",
  "roles" : [
    {
      "role" : "userAdmin",
      "db" : "Employee"
    }
  ]
}
```

Shows the user created

Shows the role which was assigned to the user and to which database

The output shows that a user called "Employeeadmin" was created and that user has privileges only on the "Employee" database.

## 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: "Mohan",  
    pwd: "password",  
    roles:[  
      {  
        role: "read" , db:"Marketing"},  
        role: "readwrite" , db:"Sales"}  
      ]  
    }  
  }  
)
```

Specifying the different roles  
for the user

```
db.createUser(  
  {  
    user: "Mohan",  
    pwd: "password",  
    roles:[  
      {  
        role: "read" , db:"Marketing"},  
        role: "readwrite" , db:"Sales"}  
      ]  
    }  
  }  
)
```

```

        role: "readWrite" , db:"Sales"}
    }
    ]
})

```

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

## Exercise

Add a user with the userAdminAnyDatabase role in the admin database.

The following creates the user myUserAdmin in the admin database with the userAdminAnyDatabase role and the readWriteAnyDatabase role.

```

use admin
db.createUser(
{
  user: "myUserAdmin",
  pwd: "abc123", // or cleartext password
  roles: [ { role: "userAdminAnyDatabase", db: "admin" },
"readWriteAnyDatabase" ]
}
)

```

## Re-start the MongoDB instance with access control.

Start the mongod with access control enabled.

If you start the mongod using a configuration file (bin/mongod.cfg), add the security.authorization configuration file setting:

```
security:
  authorization: enabled
```

Clients that connect to this instance must now authenticate themselves as a MongoDB user.  
Clients can only perform actions as determined by their assigned roles.

### **Connect and authenticate as the user administrator:**

Using the mongo shell, you can:

Connect with authentication by passing in user credentials,

Start a mongo shell with the -u <username>, -p, and the --authenticationDatabase <database> command line options:

```
mongo --port 27017 --authenticationDatabase "admin" -u "myUserAdmin" -p
```

Enter your password when prompted.

Create additional users as needed for your deployment

Once authenticated as the user administrator, use db.createUser() to create additional users. You can assign any built-in roles or user-defined roles to the users.

The following operation adds a user myTester to the cities database who has readWrite role in the cities database as well as read role in the reporting database.

```
use cities
db.createUser(
  {
    user: "myTester",
    pwd: test123,
    roles: [ { role: "readWrite", db: "cities" },
             { role: "read", db: "reporting" } ]
  }
)
```

After creating the additional users, disconnect the mongo shell.

## Connect to the instance and authenticate as myTester.

After disconnecting the mongo shell as myUserAdmin, reconnect as myTester. You can:

Connect with authentication by passing in user credentials,

```
mongo --port 27017 -u "myTester" --authenticationDatabase "cities" -p
```

Enter the password for the user when prompted.

Insert a document as myTester.

As myTester, you have privileges to perform read and write operations in the cities database (as well as perform read operations in the reporting database). Once authenticated as myTester, insert a document into a collection in cities database. For example, you can perform the following insert operation in the test database:

```
db.foo.insert( { x: 1, y: 1 } )
```

## References:

<https://www.guru99.com/mongodb-create-user.html>

<https://docs.mongodb.com/manual/tutorial/enable-authentication/>