Setting up admin and user accounts In MongoDB

Step:1

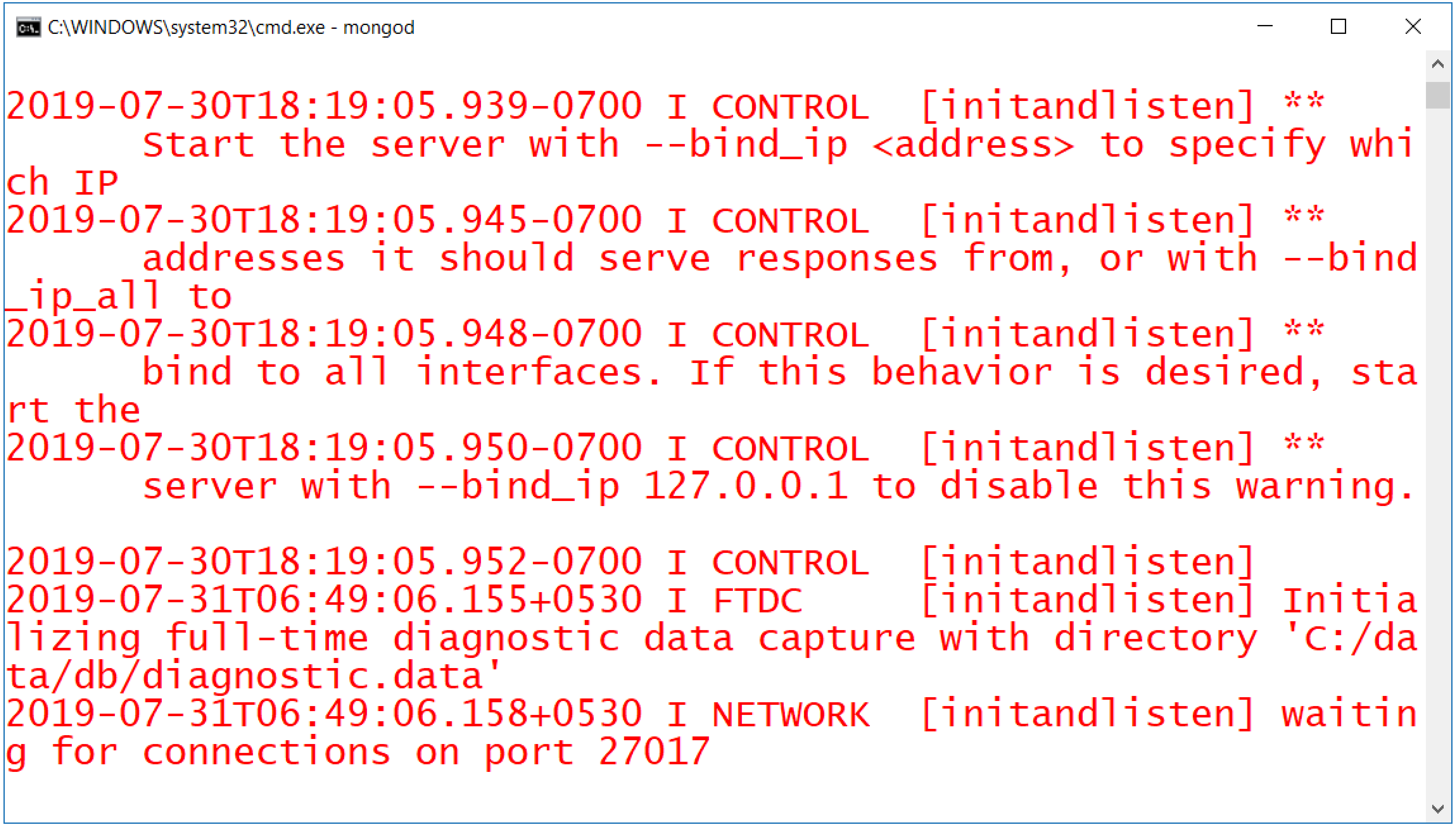
Type in the following command to start the process.

```
> Mongod
```

Or

```
>sudo service mongod start
```

- You will be able to see some of the log messages on the screen similar to this.
- *Don’t worry if it shows any error, it is because most probably the service will be already running on the same port.*



C:\WINDOWS\system32\cmd.exe - mongod

```
2019-07-30T18:19:05.939-0700 I CONTROL  [initandlisten] **
      Start the server with --bind_ip <address> to specify whi
ch IP
2019-07-30T18:19:05.945-0700 I CONTROL  [initandlisten] **
      addresses it should serve responses from, or with --bind
_ip_all to
2019-07-30T18:19:05.948-0700 I CONTROL  [initandlisten] **
      bind to all interfaces. If this behavior is desired, sta
rt the
2019-07-30T18:19:05.950-0700 I CONTROL  [initandlisten] **
      server with --bind_ip 127.0.0.1 to disable this warning.

2019-07-30T18:19:05.952-0700 I CONTROL  [initandlisten]
2019-07-31T06:49:06.155+0530 I FTDC     [initandlisten] Initia
lizing full-time diagnostic data capture with directory 'C:/da
ta/db/diagnostic.data'
2019-07-31T06:49:06.158+0530 I NETWORK  [initandlisten] waitin
g for connections on port 27017
```

Step:2

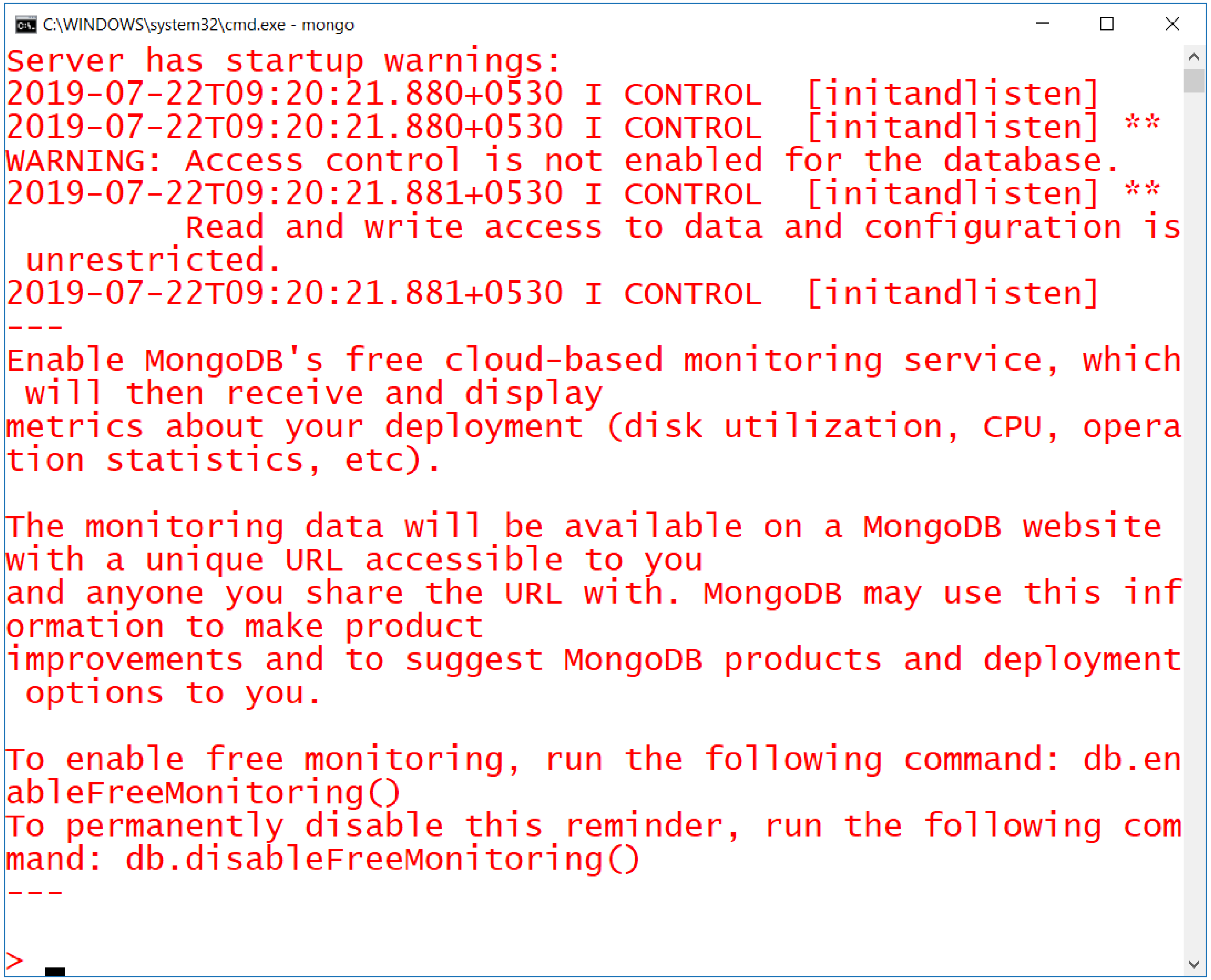
Getting started with Mongo Shell

The mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations.

To start the Mongo Shell, simply type the following command on a new terminal.

```
>mongo
```

You will see a few messages including the current MongoDB and Mongo Shell version and prompt > waiting for any commands.



```
Server has startup warnings:
2019-07-22T09:20:21.880+0530 I CONTROL  [initandlisten]
2019-07-22T09:20:21.880+0530 I CONTROL  [initandlisten] **
WARNING: Access control is not enabled for the database.
2019-07-22T09:20:21.881+0530 I CONTROL  [initandlisten] **
         Read and write access to data and configuration is
 unrestricted.
2019-07-22T09:20:21.881+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which
 will then receive and display
metrics about your deployment (disk utilization, CPU, opera
tion 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 inf
ormation to make product
improvements and to suggest MongoDB products and deployment
 options to you.

To enable free monitoring, run the following command: db.en
ableFreeMonitoring()
To permanently disable this reminder, run the following com
mand: db.disableFreeMonitoring()
---

>
```

Now, we'll start off by creating users and restricting unauthorized access to our database.

Since we already started the mongod service without access control [default], we can continue to the next step.

Creating the user Administrator

- We need to create an admin user in the admin database with userAdminAnyDatabase privilege.
- To do this we'll first use the admin database.
- Use the following code in the mongo shell to use the admin database, if the database doesn't exist, MongoDB will create the particular database and use it.

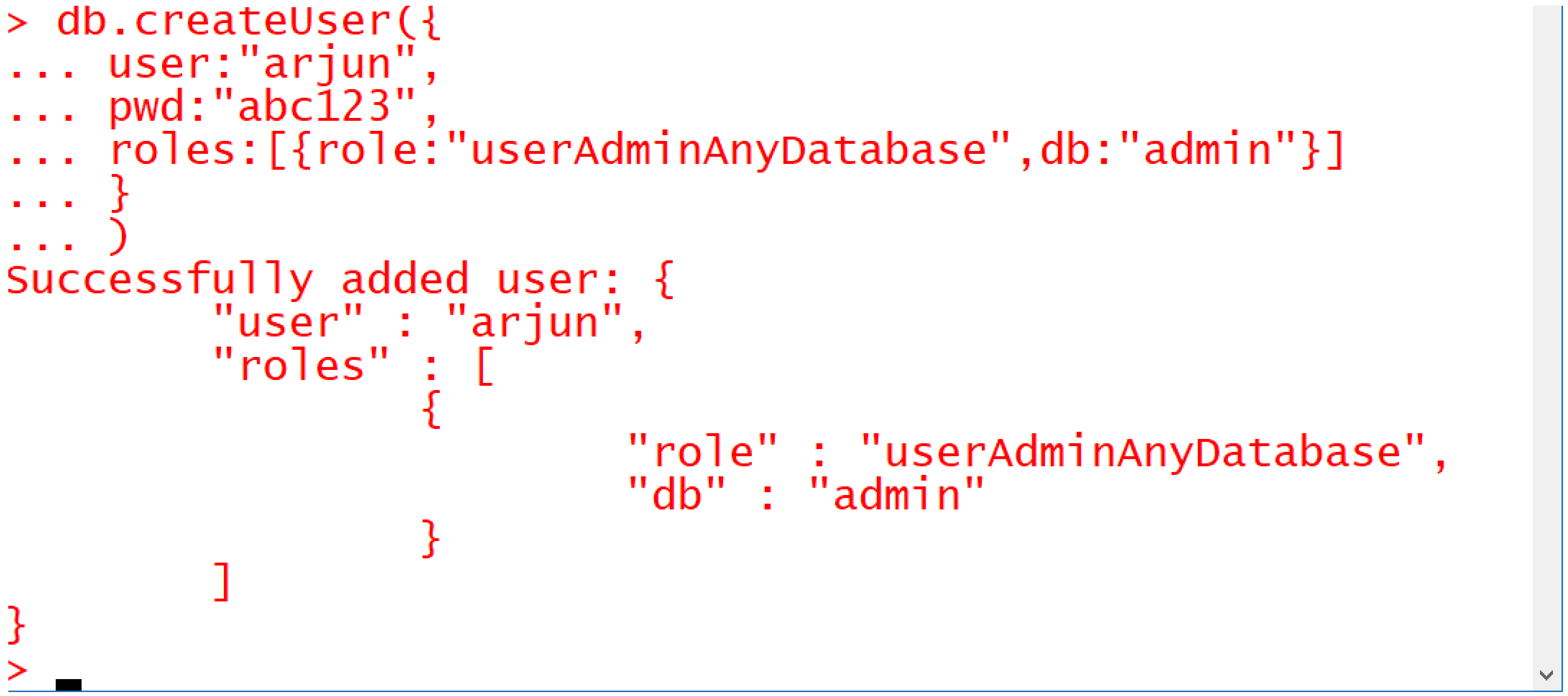
```
>use admin
```

make sure you get the message stating Switched to db admin.

Step:3

- Now, we’ll use the createUser() function to create an admin user.
- Type the following command in the mongo shell to create the admin user.

```
db.createUser(
 {
   user: “arjun",
   pwd: "abc123",
   roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
 }
)
```



```
> db.createUser({
... user:"arjun",
... pwd:"abc123",
... roles:[{role:"userAdminAnyDatabase",db:"admin"}]
... }
... )
Successfully added user: {
        "user" : "arjun",
        "roles" : [
                {
                        "role" : "userAdminAnyDatabase",
                        "db" : "admin"
                }
        ]
}
>
```

Step:4

- Once we finish creating the userAdministrator we can start the mongod instance with access control.
- Now close the shell by typing exit.
- Then restart the mongod instance using the following command in the terminal after closing the running instance on the other terminal using ctrl + C .

```
>mongod --auth --port 27017
```

Start the mongo shell in another terminal using the following command to authenticate as user administrator.

```
>mongod --port 27017 -u “arjun" -p "abc123" --authenticationDatabase "admin“
```

- The userAdministrator user has only permission to create and manage the database users.

- If you try to read or write in the database using userAdministrator user MongoDB will return an error.

- So, we need to create additional users with roles to read and write on the database.

Step:5

Creating database users

We’ll start off by creating a test database, using the following code in mongo shell.

```
>use test
```

As I said earlier, if we use the above code, MongoDB will create the “test” database if it doesn’t exist and switches to that DB.

We can use the db.createUser() function to create an user for this db. Note that, the keyword db refers to the current db we are working on. In this case its the “test” db we switched earlier to.

```
db.createUser(
 {
   user: “mohan",
   pwd: "xyz123",
   roles: [ { role: "readWrite", db: "test" } ]
 }
)
```

```
> db.createUser({
... user:"mohan",
... pwd:"123abc",
... roles:[{role:"readWrite",db:"test"}]
... }
... )
Successfully added user: {
        "user" : "mohan",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "test"
                }
        ]
}
>
```

Step:6

- We can now log in to the database as the “mohan” with read and write access to the “test” DB.
- Close the current mongo shell instance by using exit command and type the following command in the terminal.

```
>mongo --port 27017 -u “mohan" -p “123abc" --authenticationDatabase "test“
```

- To view the current DB you are working in, use the db command in the mongo shell.
- In this case, we should be able to see “test” printed out.
- Or, you can start using the connection string/URI to use this database in applications.

```
mogodb://testUser:xyz123@localhost:27017/test
```

How to Drop a User?

Assuming that you are logged in with a suitable role that allows you to drop users, you will need to change context to the database where it was created.

```
> use admin

> db.dropUser('<userName>’)
```

Where are Users Stored?

To check a user, you must change your context to the database in which the user was created, such as the Admin database.

```
> use '<dbName>'
```

You can then use either

```
> db.system.users.find()
```

or

```
> db.getUsers()
```

But, if you only want to ask for a specific user, use this command:

```
> db.getUser('<userName>')
```

How to Logout?

The logout ends the current authentication session. You must do it in the same database that you authenticated to.

```
> use admin
> db.logout()
```

Database Administration Roles

The database administration roles we can use are the following:

dbAdmin – Grant privileges to perform administrative tasks
userAdmin – Allows you to create and modify users and roles on the current database
dbOwner – This role combines the following:

- readWrite
- dbAdmin
- userAdmin

Cluster Administration Roles

Roles at the admin database for administering the whole system.

clusterMonitor – Provides read-only access to monitoring tools
clusterManager – For management and monitoring actions on the cluster
hostManager – To monitor and manage servers
clusterAdmin – Combines the other three roles plus dropDatabase action

Backup and Restoration Roles

This role belongs to the admin database.

backup – Provides the privileges needed for backing up data
restore – Provides the privileges needed to restore data from backups

All-Database Roles

These roles lie on the admin database and provide privileges which apply to all databases.

readAnyDatabase – The same as ‘read’ role but applies to all databases
readWriteAnyDatabase – The same as ‘readWrite’ role but applies to all databases
userAdminAnyDatabase – The same as ‘userAdmin’ role but applies to all databases
dbAdminAnyDatabase – The same as ‘dbAdmin’ role but applies to all databases

Superuser Roles

The following roles are not superuser roles directly but are able to assign any user any privilege on any database, also themselves.

- **userAdmin**
- **dbOwner**
- **userAdminAnyDatabase**

The root role provides full privileges on all resources:

root

How to Grant a Role to a User?

- You can grant a role as you create the user, or retrospectively.

The next command is valid for assigning a role at the same time you create the user:

```
> use '<dbName>’
```

And you can use this command to do it later:

```
> use '<dbName>'
> db.grantRolesToUser(
   '<userName>',
   [ { role : '<roleName>', db : '<dbName>' }, '<roleName>', … ]
)
```

And you can use this command to do it later:

```
> use '<dbName>'
> db.grantRolesToUser(
  '<userName>',
  [ { role : '<roleName>', db : '<dbName>' }, '<roleName>', ... ]
)
```

How to Revoke a Role from a User?

```
> use '<dbName>'
> db.revokeRolesFromUser(
   '<userName>',
   [ { role : '<roleName>', db : '<dbname>' } | '<roleName>' ]
)
```

User-defined Roles

How to Create a Role?

```
> use '<dbName>’

> db.createRole({
   role: "<roleName>",
   privileges: [
     { resource: { db : “<dbName>”,
                   collection : “<collectionName>” },
       actions: [ '<actionName>' ]
     }
   ],
   roles: [ { role : '<fatherRoleName>', db : '<dbName>'} | '<roleName>' ]
})
```

How to Drop a Role?

```
> use '<dbName>'
➢ db.dropRole('<roleName>’)
```

How to Grant or Revoke Privileges to/from a Role
These commands grant or revoke privileges to a user-defined role.

```
> use '<dbName>'
> db.grantPrivilegesToRole(
 '<roleName>',
 [
   { resource : { db : '<dbName>', collection : '<collectionName'> },
     actions : [ '<actionName>',... ]
   },
   ...
 ]
)
```

```
> db.revokePrivilegesFromRole(
 '<roleName>',
 [
   { resource : { db : '<dbName>', collection : '<collectionName'> },
     actions : [ '<actionName>',... ]
   },
   …
 ]
)
```

How to Grant or Revoke Roles to/from a Role?

```
> use '<dbName>'
> db.grantRolesToRole(
 '<roleName>',
 [ { role : '<roleName>', db : '<dbName>' } | <roles> ]
)
> db.revokeRolesFromRole(
 '<roleName>',
 [ { role : '<roleName>', db : '<dbName>' } | <roles> ]
)
```

How to Update a Role

Be careful! As the documentation says: “An update to the privileges or roles array completely replaces the previous array’s values”.

```
> use '<dbName>'
> db.updateRole(
 '<roleName>',
 {
   privileges : [
     {
       resource : { db : '<dbName>', collection : '<collectionName>' },
       actions : [ '<actionName>' ]
     },...
   ],
   roles : [ { role : '<roleName>', db : '<dbName>' } | '<roleName>' ]
 }
)
```