MongoDB Security Setup for Pets Collection

This guide demonstrates how to enable authentication in MongoDB, create users and roles to control read/write access on the **pets** collection, plus allow read-only access on the **types** collection. All examples use `password1` as the password.

---

1. Enable Authentication

In `/etc/mongod.conf`, add:

```
security:
  authorization: "enabled"
```

Restart MongoDB:

```
sudo pkill mongod
sudo mongod --config /etc/mongod.conf --fork
```

---

2. Create the Admin User

Start the shell without auth (localhost exception):

```
mongosh
use admin

db.createUser({
  user: "siteAdmin",
  pwd:  "password1",
  roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
})
exit
```

---

3. Create Custom Role: petManager

Login as admin:

```
mongosh -u siteAdmin -p "password1" --authenticationDatabase admin
use admin

db.createRole({
  role: "petManager",
  privileges: [
    {
      resource: { db: "my_database", collection: "pets" },
      actions: [ "find", "insert", "update", "remove" ]
    },
    {
      resource: { db: "my_database", collection: "types" },
      actions: [ "find" ]
    }
  ],
  roles: []
})
exit
```

This gives full CRUD on pets and read-only on types.

---

4. Create Application User: appUser

In the admin shell:

```
mongosh -u siteAdmin -p "password1" --authenticationDatabase admin
use my_database

db.createUser({
  user: "appUser",
  pwd:  "password1",
  roles: [ { role: "petManager", db: "admin" } ]
})
exit
```

---

5. Test Permissions as appUser

```
mongosh -u appUser -p "password1" --authenticationDatabase my_database
use my_database

// Allowed: CRUD on pets
db.pets.insertOne({ name: "Rex", type: "dog", age: 4 })
db.pets.find().pretty()

// Allowed: read-only on types
db.types.find().pretty()

// Denied: write on types
db.types.insertOne({ _id: "hamster", food: "pellets" })
// This should return an unauthorized error.
```

End of pets security setup.

```

```