MongoDB Security (Users & Roles)

•••

MongoDB User Group
22 March 2017, Madrid

Who am I

Juan Roy

Twitter: @juanroycouto

Email: juanroycouto@gmail.com

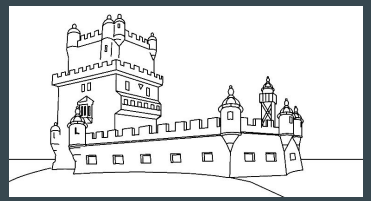
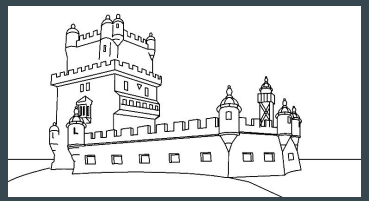
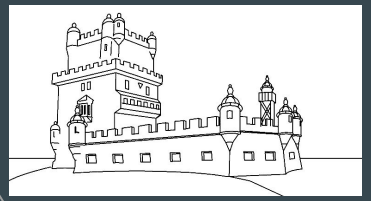
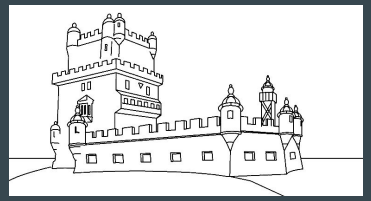
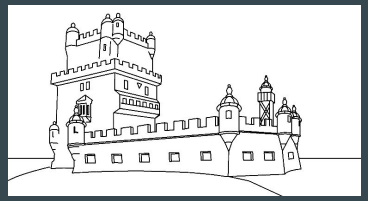
MongoDB DBA at Grupo Undanet

MUG Madrid 22 March 2017

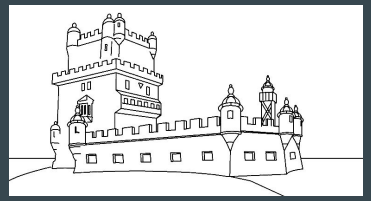
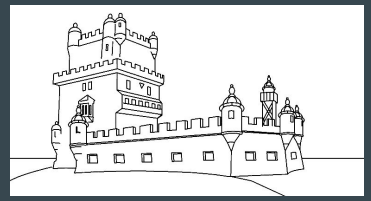
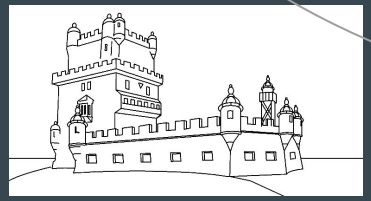
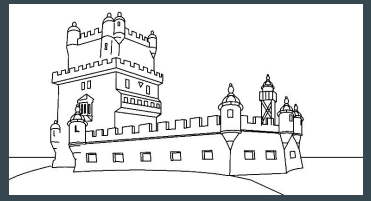
MongoDB - Characters

- The Kingdom
- The Castle
- The King
- The Collaborators
- The Castle Goods
- The Emissary
- The Moat of the Castle
- The Visitors
- The Auditor
- The Monitor
- The King without Kingdom

MUG Madrid 22 March 2017

The MongoDB Kingdom

mongoDB.

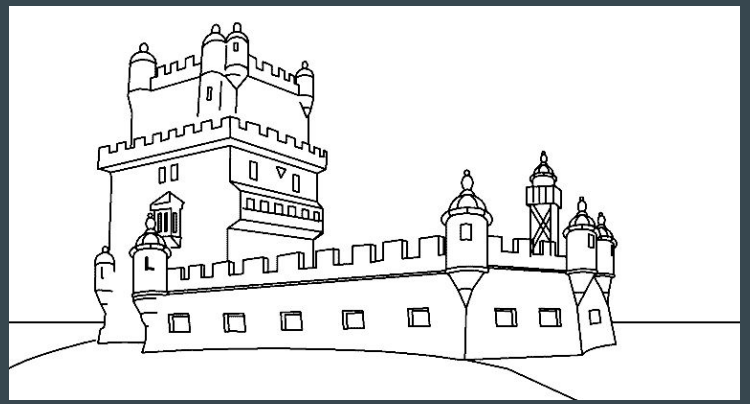
MUG Madrid 22 March 2017

MongoDB - The Castle

- If the security is not enabled at the castle everybody will be able to get into the rooms and take any goods.

- If the security is enabled nobody will be able to get into the castle before identifying. Companies outside the kingdom can deal with this.

- Without the secret key no castle will be able to join to the Kingdom.



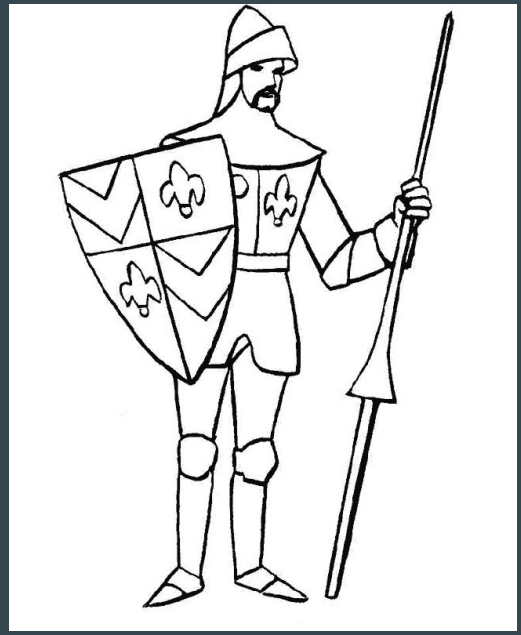
MongoDB - The King

userAdminAnyDatabase

- He decides his collaborators (users) to do the tasks (actions) needed in each room of the castle.
- Creates, grants and revokes roles to its collaborators

MongoDB - The Collaborators

- They must do specific tasks.
- They are not allowed to do any tasks that are not in their role.
- They only can work in their workplaces



```
> db.createUser({
   user: "uuuu",
   pwd: "pppp",
   roles: [ { role: "roleName", db: "dbName" } ]
});

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

mongoDB®

MongoDB - The Roles

- Roles must be standard. When a new collaborator is named assumes the role's tasks of the last one.
- In the role are written down the tasks to do and the places the tasks must be done by the King's collaborator (Kingdom, castle, database, etc).
- MongoDB offers built-in roles and the possibility to create new ones depending on our needs.

mongoDB®

MongoDB - The Roles

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

> db.grantRolesToUser(‘uuuu’,
    [ { role : ‘roleName’, db : ‘dbname’ } ]);

> db.revokeRolesFromUser(‘uuuu’,
    [ { role : ‘roleName’, db : ‘dbname’ } ]);
```

mongoDB®

MongoDB - The Castle Goods

- Castle assets (data) are guarded in rooms where nobody knows what's inside (disk encryption).
- To access the goods is necessary a key that has to match which the guardian has.

mongoDB®

MongoDB - The Emissary

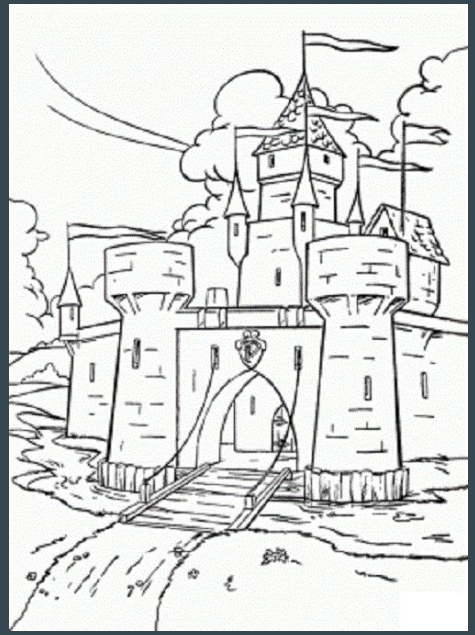
- He carries the messages from one castle to another (Replica Set).
- These messages must be encrypted (network encryption).

mongoDB.

MongoDB - The Moat of the Castle

- Firewalls
  - Limit incoming traffic on a specific port to specific systems and limit incoming traffic from untrusted hosts.

- Virtual Private Networks
  - VPNs make possible to link two networks over an encrypted and limited-access trusted network.



mongoDB®

MongoDB - The Visitors

read-only views

- The castle's visitors (physical persons or apps) are allowed to view only the goods the King is interested in.

```
> db.createView(
    'viewName',
    'originalCollection',
    [ { aggregationStages } ]
);
```

mongoDB.

MongoDB - The Auditor

Auditing

- Records the following operations:
  - CRUD Operations.
  - Schema (DDL).
  - Authentication & Authorization.
  - Replica Set & Sharded Cluster.

mongoDB®

MongoDB - The Monitor

Monitoring

- He monitors the goods exchanges and the state of the kingdom's castles.
  - From land (OPS Manager).
  - And also from the air (Cloud Manager).

mongoDB.

MongoDB - The King without Kingdom

- For those Kings without Kingdom exists a paradise (MongoDB Atlas) where they can:
  - Rent the castles they need.
  - With all the security measures.
  - And fully monitored, both goods and castles.

mongoDB®

MongoDB Security Features

- User Access Management
  - MongoDB Authentication
  - MongoDB Authorization
- MongoDB Auditing (forensic analysis)
- MongoDB Encryption (data protection over the network -TLS- and at-rest)
- Environmental & Process Control

MongoDB Authentication

- Designed to confirm the identity of:
  - Users.
  - Administrators, Developers, etc.
  - Software systems (apps, reporting tools, etc).
  - Physical and logical nodes where the database runs on.
- Best practices:
  - Create login credentials for each entity that will need access to the database.
  - Enforce authentication between nodes.
- Supporting in-database (SCRAM-SHA-1) and Centralized User Access Management (LDAP, x.509, Kerberos).

MongoDB Authorization

- Resources<--Actions<--Privileges<--Roles-->Users
- Authorization governs what an User is allowed to do in the resource.
- Best practices:
  - Grant minimal access to users (only to those they need to perform their functions).
  - Group common access privileges into roles rather than having to define them individually for each user.
  - Control access to sensitive data (restrict permissions to individual fields).

MongoDB Auditing

- Auditing can detect:
  - Attempts to access unauthorized data.
  - Changes to database configuration for each entity, recording:
    - Change action.
    - Identity.
    - Timestamp.
  - Changes to data:
    - Capture every query or write operation, filtering only those fields you need.

MongoDB Encryption

- Encryption is the encoding of data, in transit or at rest, enabling only authorised users to read it.
- Encrypt Connections to the Database:
  - Internal communications between castles.
  - Connections via drivers or shell.
  - Access to castles.
- Encrypt Data at Rest (On-disk encryption of the database's data files).
- Sign and Rotate Encryption Keys (Encryption keys for network and disk encryption should be periodically rotated).
- Enforce Strong Encryption.

MongoDB Environmental and Process Control

- Installation of firewalls.
- Network configurations.
- Defining file system permissions.
- Creation of physical access controls to the IT environment.
- DBA and Developer training.
- Database provisioning, monitoring and backup.
- Database maintenance.

Configuration - mongod.conf

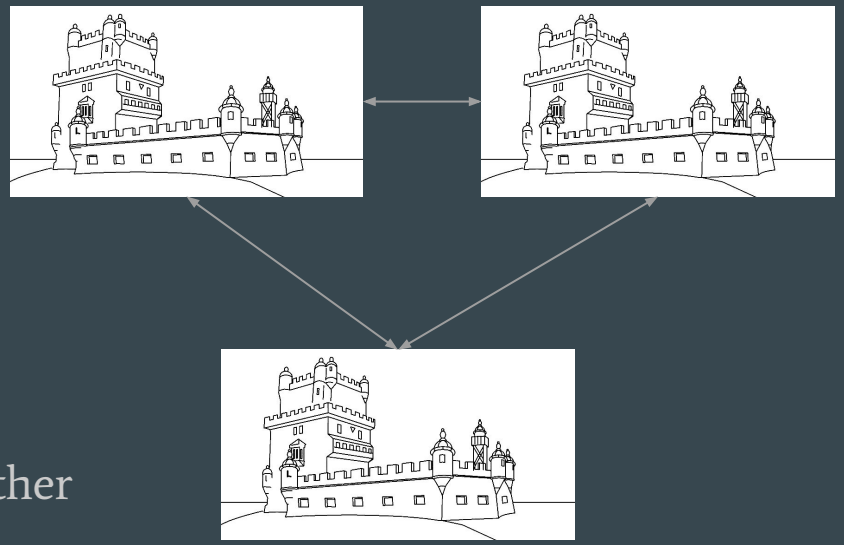
- Auth enables authorization to control users access to castle resources and its actions.

```
...
security:
  keyFile: "/data/key/replicaset.key"
  authorization: "enabled"
...
```

Configuration - Key file

```
$ openssl rand -base64 755 >
/data/key/replicaset.key

$ chmod 400 /data/key/replicaset.key
```



- The key file stores the shared secret that MongoDB castles use to authenticate to each other in a County or Kingdom.

mongoDB®

Roles

- userAdminAnyDatabase
- clusterManager
- clusterMonitor
- backup
- restore
- dbAdmin
- readWrite
- read

Roles - userAdminAnyDatabase

- The King. He can create users, roles and grant or revoke roles to any user.

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
});
```

Roles - clusterManager

- The King's architect who manage the configuration of the castles (Replica Set & Cluster).

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "clusterManager", db: "admin" } ]
});
```

Roles - clusterMonitor

- Architect of the King who watches over the state of the kingdom (OPS & Cloud Manager).

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "clusterMonitor", db: "admin" } ]
});
```

Roles - backup

- King's Employee who stores its goods in a safe place outside the Kingdom.

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "backup", db: "admin" } ]
});
```

Roles - restore

- King's Employee who restores its goods when it is necessary.

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "restore", db: "admin" } ]
});
```

Roles - dbAdmin

- Database Administrator

```
> use test;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "dbAdmin", db: "test" } ]
});
```

Roles - readWrite

- Castle's visitors who exchange goods.

```
> use test;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "readWrite", db: "test" } ]
});
```

Roles - read

- Castle's visitors who only want to view what there is inside.

```
> use admin;

> db.createUser(
{
    user: "uuuu",
    pwd: "pppp",
    roles: [ { role: "read", db: "test" }, { role: "readWrite", db: "test2" }
]
});
```

Roles

- Each role is scoped to the database in which it has been created.
- A role can only include privileges that apply to its database and can only inherit from other roles in its database.
- A role created in the admin database can include privileges that apply to the admin database, other databases or to the cluster resource, and can inherit from roles in other databases as well as the admin database.

```
> use admin;

> db.system.roles.find();
```

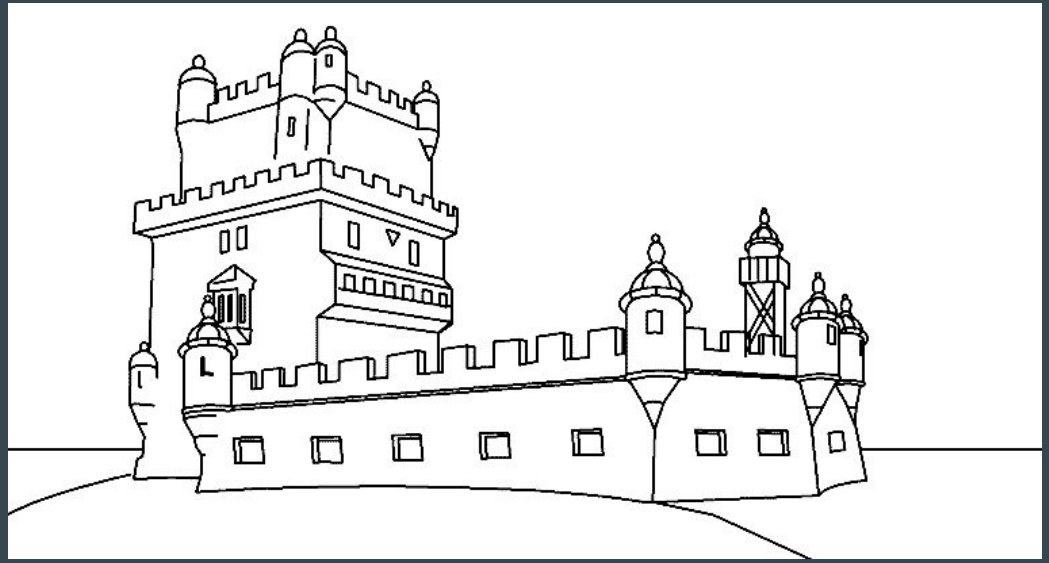
Field-Level Security (Read-Only Views)

- Restrict access to sensitive data.
- Non-materialized views expose only a subset of data from a collection.
- This view is generated from an aggregation over another collection/s or view.
- Permissions granted against the view are specified separately from permissions granted to the underlying collection/s.

```
> db.createView('viewName', 'originalCollection', [ { aggregationStages } ]);

> db.system.views.find();
```

Questions?



mongoDB®

MUG Madrid 22 March 2017

Thank you!

Thank you for your attention!

MongoDB User Group
22 March 2017, Madrid