



# mongoDB

## SEGURIDAD Y DURABILIDAD (DURABILITY)

Enrique Barra



# SEGURIDAD

# SECURITY

- MongoDB proporciona varias características, como autenticación, control de acceso, encriptación, para asegurar los despliegues
- Algunas de las principales características de seguridad son:
- Autenticación
  - <https://docs.mongodb.com/manual/core/authentication/>
  - <https://docs.mongodb.com/manual/core/security-scram/>
  - <https://docs.mongodb.com/manual/core/security-x.509/>
- Autorización
  - <https://docs.mongodb.com/manual/core/authorization/>
- Cifrado del transporte (TLS/SSL)
  - <https://docs.mongodb.com/manual/core/security-transport-encryption/>
- Cifrado del contenido
  - <https://docs.mongodb.com/manual/core/security-client-side-encryption/>

# AUTENTICACIÓN

- La autenticación requiere que los clientes proporcionen una **prueba de identidad** para acceder a la base de datos.
- MongoDB proporciona 3 tipos de mecanismos de autenticación:
  1. Autenticación basada en **contraseña** - el cliente verifica su identidad probando la posesión de un valor secreto predeterminado. Esto incluye SCRAM-SHA-1 y SCRAM-SHA-256
    1. SCRAM: Salted Challenge Response Authentication Mechanism
    2. <https://www.mongodb.com/blog/post/improved-password-based-authentication-mongodb-30-scram-explained-part-1>
    3. <https://www.mongodb.com/blog/post/improved-password-based-authentication-mongodb-30-scram-explained-part-2>
  2. Autenticación basada en **certificados**: el cliente verifica su identidad mediante un certificado x.509 firmado por una autoridad de certificación de confianza. Requiere conexión TLS/SSL
  3. Autenticación **externa** - el cliente verifica su identidad utilizando un servicio externo, como Kerberos o LDAP.

# AUTENTICACIÓN

- Opción 1: Por línea de comandos (Shell) al conectarse a una instancia mongod o mongos
  - --username, --password, y --authenticationDatabase
  - Ejemplo:
  - `mongo --port 27017 --authenticationDatabase "admin" -u "myUserAdmin" -p`
- Opción 2: Primero conectamos a mongod o mongos y luego ejecutamos el comando de autenticación o el método `db.auth()` contra la base de datos de autenticación
  - Ejemplo:
  - `mongo --port 27017`
  - `use admin`
  - `db.auth("myUserAdmin", passwordPrompt())` // or cleartext password

# AUTORIZACIÓN

- MongoDB no tiene el control de acceso por defecto.
- Podemos habilitar la autorización utilizando la opción `--auth` o `security.authorization`.
  - Ejemplo:
  - `mongod --auth --port 27017 --dbpath /var/lib/mongodb`
- Una vez habilitado el control de acceso, los usuarios deben autenticarse y es conveniente definir Roles
- Los **roles** otorgan privilegios para realizar las acciones (<https://docs.mongodb.com/manual/reference/privilege-actions/>) especificadas sobre recursos concretos (una bbdd, una colección, un conjunto de colecciones, un cluster)
- Se definen al crear el usuario (con `db.createUser`) o una vez creado con `db.grantRolesToUser`
  - Built in Roles: <https://docs.mongodb.com/manual/reference/built-in-roles/>
  - User defined Roles: <https://docs.mongodb.com/manual/core/security-user-defined-roles/>

# PRINCIPLE OF LEAST PRIVILEGE (PoLP)

- Principio del mínimo privilegio
- <https://us-cert.cisa.gov/bsi/articles/knowledge/principles/least-privilege>
- “Every program and every user of the system should operate using the least set of privileges necessary to complete the job. Primarily, this principle limits the damage that can result from an accident or error”



# CIFRADO DEL TRANSPORTE (TLS/SSL)

- TLS/SSL: Transport Layer Security/Secure Sockets Layer
- Sólo permite el uso de cifrados fuertes TLS/SSL con un mínimo de 128 bits de longitud de clave para todas las conexiones.
- MongoDB puede utilizar cualquier certificado TLS/SSL válido emitido por una autoridad de certificación, o un certificado autofirmado.
  - **Si usamos un certificado autofirmado**, aunque el canal de comunicación estará encriptado para evitar la escucha de la conexión, no habrá validación de la identidad del servidor. Esto lo deja vulnerable a un ataque de tipo “man in the middle”.
  - **Si usamos un certificado firmado por una autoridad de certificación de confianza** permitirá a los controladores de MongoDB verificar la identidad del servidor.
- <https://docs.mongodb.com/manual/tutorial/configure-ssl/>
- En el fichero de configuración indicamos el certificado:

```
net:  
  tls:  
    mode: requireTLS  
    certificateKeyFile: /etc/ssl/mongodb.pem
```



# CIFRADO DE TODOS LOS DATOS

- En MongoDB solo está disponible para la version Enterprise
- <https://docs.mongodb.com/manual/core/security-encryption-at-rest/>
- El cifrado ocurre de modo transparente usando AES256-CBC (256-bit Advanced Encryption Standard in Cipher Block Chaining mode) via OpenSSL
- Todos los datos se cifran
- Cuidado con los logs que pueden ir en claro
  - Para ello existe security.redactClientLogData
  - <https://docs.mongodb.com/manual/reference/configuration-options/#security.redactClientLogData>
- Pero siempre podremos cifrar todo el disco o volumen por si se produce el robo físico

# CIFRADO DE CAMPOS CONCRETOS

```
{  
  "name" : "John Doe",  
  "address" : {  
    "street" : "1234 Main Street",  
    "city" : "MongoDBVille",  
    "zip" : 99999  
  },  
  "phone" : "949-555-1212",  
  "ssn" : "123-45-6789"  
}
```



```
{  
  "name" : "John Doe",  
  "address" : {  
    "street" : "1234 Main Street",  
    "city" : "MongoDBVille",  
    "zip" : 99999  
  },  
  "phone" : BinData(6,"U2FsdGVkX1+CGIDGUnGgtS46+c7R5u17SwPDEmzyCbA="),  
  "ssn" : BinData(6,"AaIoEw285E3AnfjP+r8ph2YCvMI1+rWzpZK97tV6iz0jx")  
}
```

# MÉTODOS DE CIFRADO SOPORTADOS EN MONGODB

- Cifrado manual (explícito) de campos

- `ClientEncryption.encrypt(encryptionKeyId, value, encryptionAlgorithm)`
- `encryptionAlgorithms`:
  - `AEAD_AES_256_CBC_HMAC_SHA_512-Deterministic`
  - `AEAD_AES_256_CBC_HMAC_SHA_512-Random`
  - <https://docs.mongodb.com/manual/core/security-client-side-encryption/#field-level-encryption-algorithms>
  - Ejemplo:
  - ```
db.employees.insertOne({  
    "name" : "J. Doe",  
    "taxid" : clientEncryption.encrypt(  
        UUID("64e2d87d-f168-493c-bbdf-a394535a2cb9"),  
        "123-45-6789",  
        "AEAD_AES_256_CBC_HMAC_SHA_512-Deterministic" )  
    })
```

- Cifrado automático de campos

- En MongoDB Enterprise y Atlas
- <https://docs.mongodb.com/manual/core/security-automatic-client-side-encryption/>

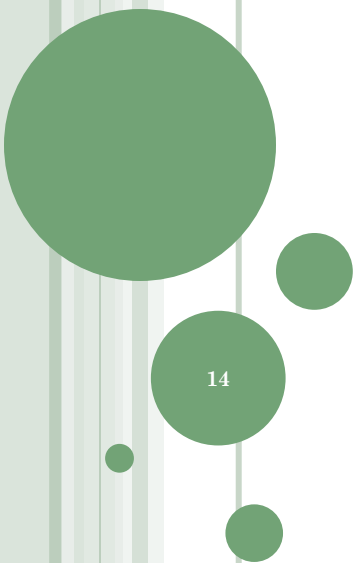
# CIFRADO EN EL ODM

- Se puede cifrar a nivel de aplicación con el ODM, en este caso MongoDB ni se entera que está guardando datos cifrados
- Inconveniente: con la shell no podremos acceder a dichos campos en claro. Solo con el ODM que es el que tiene las funciones cifrar y descifrar.

```
var mySchema = new Schema({
  name: {
    type: String,
    default: '',
    trim: true,
    required: 'Please enter group name',
    unique: true,
    get: decryptFunction,
    set: encryptFunction
  }
});
mySchema.set('toObject', {getters: true});
mySchema.set('toJSON', {getters: true});
```

# SEGURIDAD – RESUMEN

- MongoDB también proporciona la Lista de Seguridad con una lista de acciones recomendadas para proteger un despliegue de MongoDB.
  - <https://docs.mongodb.com/manual/administration/security-checklist/>
- 1. Habilitar el control de acceso y forzar autenticación
- 2. Configurar control de acceso basado en Roles
  - 1. Principio de mínimo privilegio
- 3. Cifrar las Comunicaciones (TLS/SSL)
- 4. Cifrar los datos almacenados (cuidado con replicas y backups)
  - 1. Todo el disco
  - 2. Campos concretos
  - 3. Ambos
- 5. Limitar la exposición de red
  - 1. No escuchar en 0.0.0.0, porque escuchará en todas las interfaces de red
  - 2. Configurar adecuadamente el firewall
- 6. Correr MongoDB con un usuario dedicado, no como root
- 7. Comprobar a menudo la configuración y el despliegue
- 8. Comprobar fallos de seguridad (CVE)
  - 1. <https://www.mongodb.com/alerts>



# DURABILIDAD (DURABILITY, LA D DE ACID)

# DURABILITY

- La durabilidad es la garantía de que los datos escritos se han guardado y sobrevivirán permanentemente.
- Las bases de datos NoSQL como MongoDB dan a los desarrolladores un **control detallado** sobre la durabilidad de sus queries de escritura
- Esto permite a los desarrolladores elegir **diferentes modelos de durabilidad, seguridad y rendimiento** para diferentes clases de datos.
- Sin embargo, esto también pone la carga sobre el desarrollador para discernir y comprender los matices de las diferentes opciones de seguridad de escritura
- Esto se llama **Write Concern**
  - Varía entre **weak** y **strong**
  - **Weak** da más rendimiento pero menos seguridad
  - **Strong** da más seguridad pero menos rendimiento



# JOURNAL DE OPERACIONES

- Journal es un diario de operaciones donde se apuntan todas las modificaciones de los datos entre checkpoints (escrituras a la bbdd propiamente dichas)
- Por defecto
  - Se escriben los datos a disco cada 60 segundos (checkpoint)
  - Configurable con el parámetro `storage.syncPeriodSecs` o con `syncdelay`
  - Se escribe el journal a disco cada 100ms
  - Configurable entre 1 y 500ms con `storage.journal.commitIntervalMs` o con `journalcommitinterval`
- Nota: esto aplica para el motor wiredtiger que es el por defecto, existe in-memory (Enterprise) <https://docs.mongodb.com/manual/core/inmemory/>



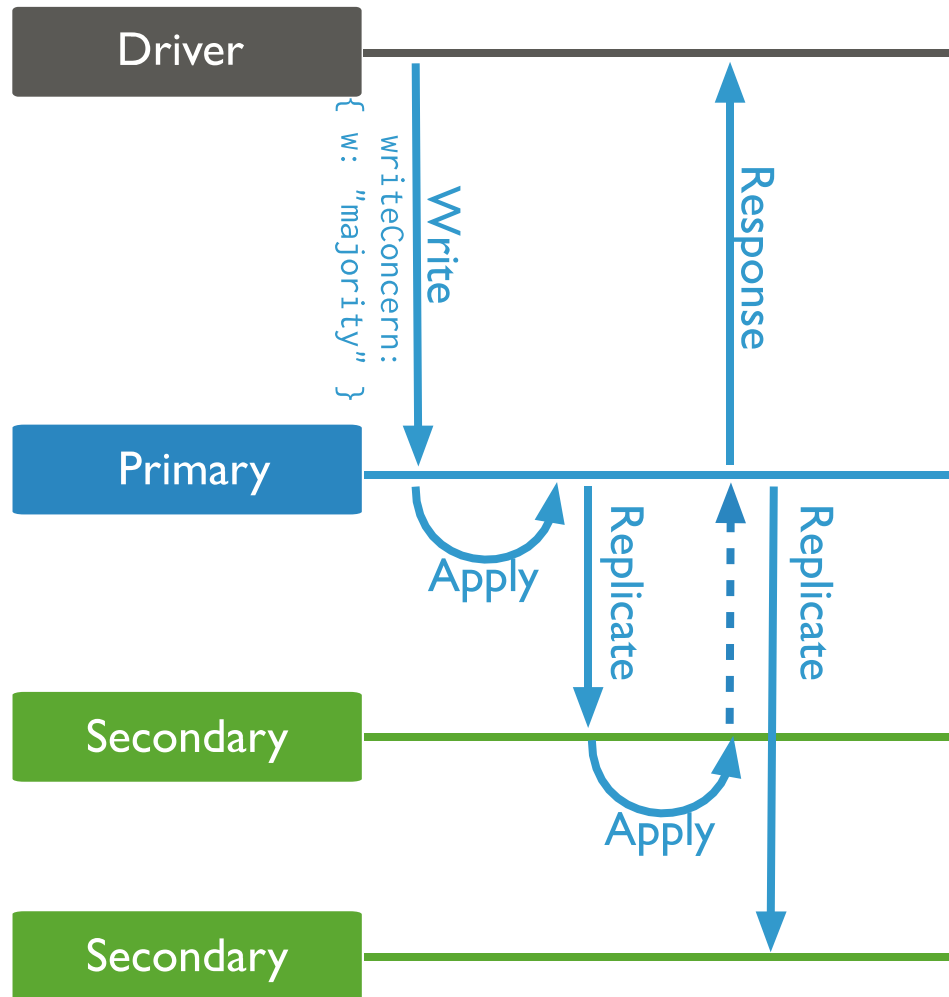
# WRITE CONCERN

- Write concern describe el nivel de reconocimiento solicitado a MongoDB para operaciones de escritura a un mongod o a réplicasets o a sharded clusters.

```
{ w: <value>, j: <boolean>, wtimeout: <number> }
```

- la opción **w** para solicitar el reconocimiento de que la operación de escritura se ha propagado a **un número determinado de instancias** mongod o a instancias mongod con etiquetas especificadas
    - Ejemplo: w: 1 para que solo avise en cuanto el primer mongod tiene la operación, w: “majority” para que espere a que la mayoría la tenga, ...
  - la opción **j** para solicitar el reconocimiento de que la operación de escritura se ha escrito en el diario **en disco** (journal)
    - True: se escribe en el journal en el disco duro
    - False: se escribe en memoria (RAM/cache nivel 1, 2 ...) pero no en disco
  - la opción **wtimeout** para especificar un límite de tiempo para evitar que las operaciones de escritura se bloqueen indefinidamente
- Ver más casuística y ejemplos:
  - <https://docs.mongodb.com/manual/reference/write-concern/>

# EJEMPLO CON 3 NODOS REPLICA Y W: MAJORITY



# EJEMPLOS

```
db.blogs.insert(  
  { title: "Data Durability", length: 1099, topic: "MongoDB" },  
  { writeConcern: { w: 2, wtimeout: 2500 } }  
)
```

```
db.products.insert(  
  { item: "envelopes", qty : 100, type: "Clasp" },  
  { writeConcern: { w: "majority" , wtimeout: 5000 } }  
)
```

```
db.students.insert(  
  { name: "pepe", age : 26, type: "Erasmus" },  
  { writeConcern: { j:1, w: "majority" , wtimeout: 4000 } }  
)
```





mongoDB

# SECURITY AND DURABILITY

Enrique Barra