**Creating user :**

use test -- db

db.createUser(

{

user: "myTester",

pwd: passwordPrompt(), // or cleartext password

roles: [ { role: "readWrite", db: "test" },

{ role: "read", db: "reporting" } ]

}

)

## Encryption at-rest

Encryption at-rest is a database-level protection layer to guarantee that the written files and data are encrypted while stored. MongoDB Enterprise Advanced (EA) has implemented the at-rest encryption in WiredTiger, the database storage engine, using [AES-256](https://www.mongodb.com/docs/v7.0/core/security-encryption-at-rest/" \t "https://www.mongodb.com/products/capabilities/security/_self). You can configure at-rest encryption in MongoDB EA with a [KMIP-enabled](https://www.mongodb.com/docs/v7.0/tutorial/configure-encryption/" \t "https://www.mongodb.com/products/capabilities/security/_self) key provider.

customer data is encrypted at-rest by default using AES-256 to secure all volume (disk) data. The process is automated by the transparent disk encryption of your selected cloud provider, and the cloud provider fully manages the encryption keys. You may also choose to enable database-level encryption, which allows you to bring your own encryption keys in AWS Key Management Service (KMS), Google Cloud KMS, or Azure Key Vault.

## In-Use Encryption

Data is encrypted client-side with customer-controlled encryption keys, before being sent, stored, or retrieved from the database. The benefits of this approach are:

**Data encrypted throughout its lifecycle**  
The strongest technical control to ensure that data always remains encrypted in-use, in backups, at-rest, and in-transit.

**Faster application development cycle**  
MongoDB takes the complexity out of developing applications for sensitive workloads. Developers don’t have to be security or cryptography experts to build encryption into their applications.

**Address critical data privacy use cases**  
Helps customers meet strict data privacy requirements such as HIPAA, GDPR, PCI, CCPA and more.

MongoDB has two features for encryption in-use to meet your data protection needs.

### Client-Side Field Level Encryption :

Client-Side Field Level Encryption (CSFLE) is an in-use encryption capability that enables a client application to encrypt sensitive data before storing it in the MongoDB database. Sensitive data is transparently encrypted, remains encrypted throughout its lifecycle, and is only decrypted on the client side.

### Queryable Encryption :

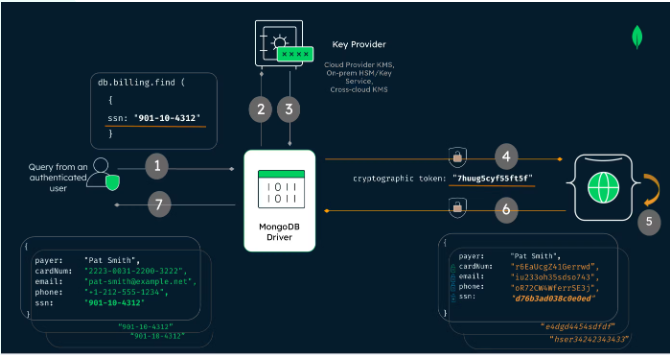
Queryable Encryption is an in-use encryption capability that enables an application to encrypt sensitive data from the client-side, store the encrypted data in the MongoDB database, and run expressive queries on the encrypted data.

**Steps :**

1.An user submit the query and mongodb driver analyses the query

2.The driver recognises the query and reuest the key from key provider

1. The driver gets the key from the key provider
2. The driver submits the query to server
3. The mongodb server process the query on fulley encryped data
4. The mongodb server the encrypted data to driver
5. The query result are decrypted along with key on client side



# Restrict Write Operations to MongoDB[IMG_256](https://www.mongodb.com/docs/compass/current/settings/read-only/#restrict-write-operations-to-mongodb)

You can use the readOnly option to prevent users from performing write operations to your MongoDB deployment through Compass. If you enable the readOnly option, users cannot modify documents, create indexes, or specify validation rules.