**MongoDB Documentation**

**1. What is MongoDB?**

**MongoDB** is a popular NoSQL database that stores data in a flexible, JSON-like format called **BSON**. Unlike relational databases, MongoDB doesn't use rows and columns, but stores data as documents inside collections. It is designed for **high performance**, **scalability**, and **ease of development**, making it ideal for modern web, real-time analytics, and big data applications.

**Key Features**

- **Document-Oriented:** Data is stored in collections of BSON documents, making it easy to model complex, hierarchical relationships.
- **Schema-Less:** Collections can hold documents with different fields, supporting evolving data requirements.
- **High Performance:** Fast read/write operations optimized for large-volume workloads.
- **Horizontal Scaling:** Integrated support for sharding and distributed architectures.
- **Rich Query Language:** Supports powerful filtering, sorting, aggregation, and indexing.

**Example MongoDB Document**

```
{
  "name": "Ruthra",
  "age": 21,
  "city": "Avinashi"
}
```

**2. MongoDB Architecture**

- **MongoDB Server (mongod):** Core engine handling data storage, access, and background tasks.
- **MongoDB Shell (mongosh):** Interactive JavaScript-based client to query and manage databases.
- **Database:** Logical container for collections.
- **Collection:** Group of BSON documents (like tables in SQL).
- **Document:** Atomic unit of data (like rows in SQL), stored as flexible BSON objects.

*Insight: MongoDB's architecture is developed to facilitate **distributed data storage** and management, with easy migration between deployment environments (local, cloud, on-premises).*

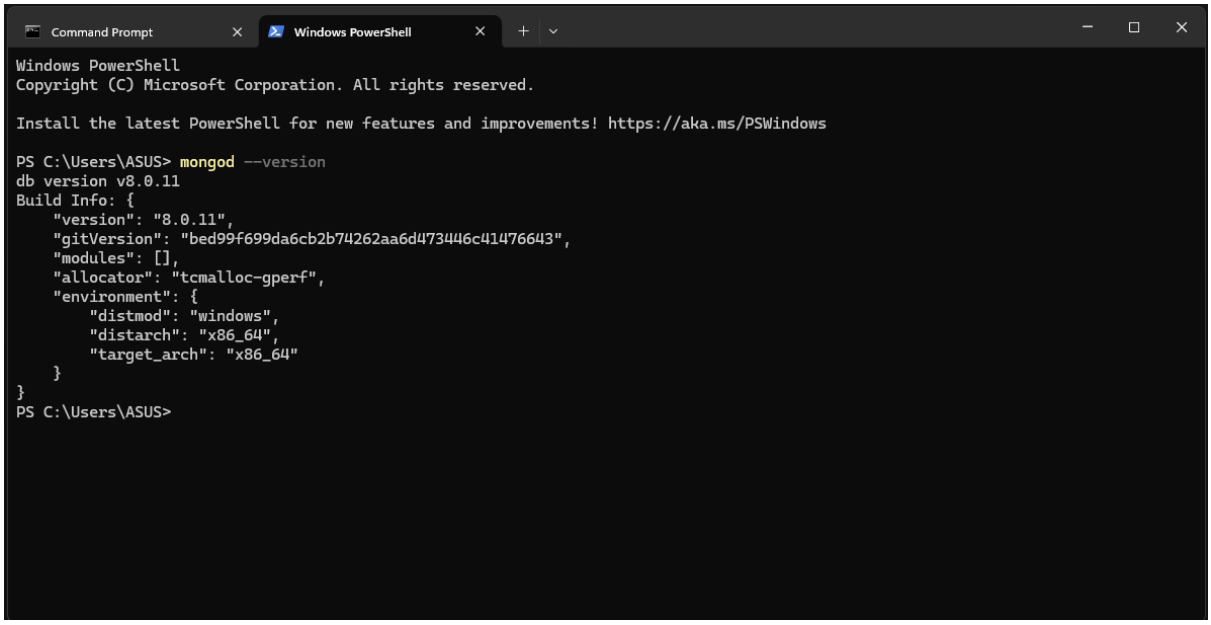
**3. Installation**

**On Windows**

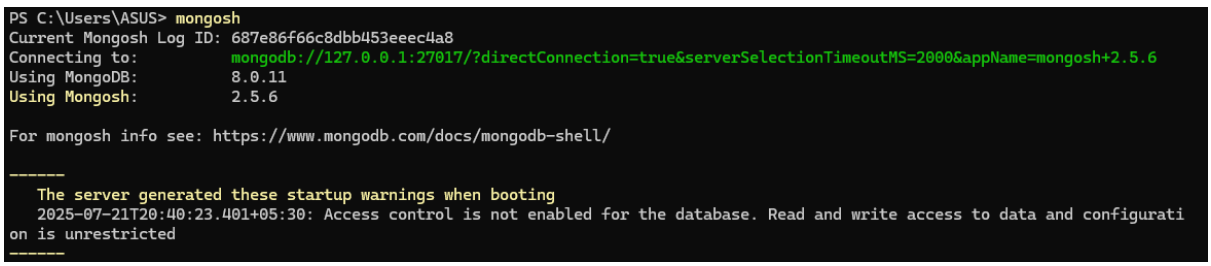
1. Download MongoDB Community Edition from the official site.
2. Choose the **MSI installer** for Windows.

3. Run the installer and select **Complete Setup**.
4. Enable **Install as a Service**—this allows MongoDB to start automatically with Windows.
5. Install **MongoDB Shell (mongosh)** (usually bundled; download separately if missing).
6. (Optional) Install **MongoDB Compass**, a GUI tool for database interaction.
7. **Verify** installation by running mongosh in Command Prompt: this should launch the MongoDB shell.

**Verify Installation:**

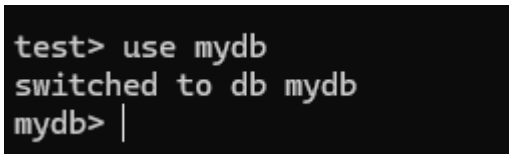



**To Start the MongoDB Server use command 'mongosh'**




**4. Core MongoDB Operations**

**Creating and Switching Databases**




*Switches to (or creates) the database "mydb".*

**Creating Collections and Inserting Documents**

```
mydb> db.createCollection("people")
{ ok: 1 }
mydb> db.people.insertOne({ name: "Alice", age: 25, city: "Chennai" })
{
  acknowledged: true,
  insertedId: ObjectId('687e87936c8dbb453eeec4a9')
}
mydb> db.people.insertMany([
...     { user_id: 1, age: 25, status: "passed" },
...     { user_id: 2, age: 30, status: "failed" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687e87a86c8dbb453eeec4aa'),
    '1': ObjectId('687e87a86c8dbb453eeec4ab')
  }
}
mydb>
```

**Viewing Databases and Collections**

```
mydb> show dbs
admin          40.00 KiB
config         48.00 KiB
hexaware       24.00 KiB
local         112.00 KiB
mydb           40.00 KiB
nvd_database   12.00 KiB
school        100.00 KiB
mydb> show collections
people
mydb>
```

**Fetching Records**

```
mydb> db.people.find()
[
  {
    _id: ObjectId('687e87936c8dbb453eeec4a9'),
    name: 'Alice',
    age: 25,
    city: 'Chennai'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4ab'),
    user_id: 2,
    age: 30,
    status: 'failed'
  }
]
mydb>
```

```
mydb> db.people.find({}, { user_id: 1, status: 1 })       // Specific fields
[
  { _id: ObjectId('687e87936c8dbb453eeec4a9') },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    status: 'passed'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4ab'),
    user_id: 2,
    status: 'failed'
  }
]
mydb>
```

```
mydb> db.people.find({}, { user_id: 1, status: 1, _id: 0 }) // Specific fields, hide _id
[
  {},
  { user_id: 1, status: 'passed' },
  { user_id: 2, status: 'failed' }
]
mydb>
```

```
mydb> db.people.findOne()                                  // Single document
{
  _id: ObjectId('687e87936c8dbb453eeec4a9'),
  name: 'Alice',
  age: 25,
  city: 'Chennai'
}
mydb>
```

*Result is shown in an array of JSON-like documents.*

**Querying Data (Filter, Logical)**

```
mydb> db.people.find({ status: "passed" })                  // Filter by value
[
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  }
]
mydb> db.people.find({ age: { $gt: 25 } })                  // Greater than
[
  {
    _id: ObjectId('687e87a86c8dbb453eeec4ab'),
    user_id: 2,
    age: 30,
    status: 'failed'
  }
]
mydb> db.people.find({ $or: [{ status: "passed" }, { age: 25 }] }) // OR condition
[
  {
    _id: ObjectId('687e87936c8dbb453eeec4a9'),
    name: 'Alice',
    age: 25,
    city: 'Chennai'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  }
]
mydb> db.people.find({ status: "passed", age: 25 })         // AND condition
[
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  }
```

### Sorting and Limiting Results

```
mydb> db.people.find().sort({ user_id: 1 })   // Ascending
[
  {
    _id: ObjectId('687e87936c8dbb453eeec4a9'),
    name: 'Alice',
    age: 25,
    city: 'Chennai'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4ab'),
    user_id: 2,
    age: 30,
    status: 'failed'
  }
]
mydb> db.people.find().sort({ user_id: -1 })  // Descending
[
  {
    _id: ObjectId('687e87a86c8dbb453eeec4ab'),
    user_id: 2,
    age: 30,
    status: 'failed'
  },
  {
    _id: ObjectId('687e87a86c8dbb453eeec4aa'),
    user_id: 1,
    age: 25,
    status: 'passed'
  },
  {
    _id: ObjectId('687e87936c8dbb453eeec4a9'),
    name: 'Alice',
    age: 25,
    city: 'Chennai'
  }
]
mydb>
```

db.people.find().limit(5).skip(10)  *// Pagination: skipping and limiting*

### Counting and Distinct

```
mydb> db.people.find().limit(5).skip(10)      // Pagination: skipping and limiting

mydb> db.people.countDocuments({})    // Total records
3
mydb> db.people.countDocuments({ age: { $gt: 30 } })   // With condition
0
mydb> db.people.distinct("status")    // All unique status values
[ 'failed', 'passed' ]
mydb>
```

*Counting helps in analytics and reporting; distinct is used for categorical breakdowns.*

### Updating and Deleting

```
[ 'failed', 'passed' ]
mydb> db.people.updateOne({ user_id: 1 }, { $set: { age: 26 } }) // Update record
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mydb> db.people.deleteOne({ user_id: 1 })                        // Delete record
{ acknowledged: true, deletedCount: 1 }
mydb>
```

**Dropping Collections and Databases**

```
mydb> db.people.drop()      // Remove the collection
true
mydb> db.dropDatabase()     // Remove the whole database
{ ok: 1, dropped: 'mydb' }
mydb>
```

**Explain Query Plans (Performance Insight)**

```
mydb> db.people.find({ status: "A" }).explain()
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'mydb.people',
    parsedQuery: { status: { '$eq': 'A' } },
    indexFilterSet: false,
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: { isCached: false, stage: 'EOF' },
    rejectedPlans: []
  },
  queryShapeHash: 'E4B71C1FC6A4544FA07E582BF9C76628598DFC2DA2B46B02BC8B86D1A595E8BD',
  command: { find: 'people', filter: { status: 'A' }, '$db': 'mydb' },
  serverInfo: {
    host: 'Ruthra-PC',
    port: 27017,
    version: '8.0.11',
    gitVersion: 'bed99f699da6cb2b74262aa6d473446c41476643'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
mydb>
```

*This shows how MongoDB will execute the query, useful for optimizing performance in larger databases.*

**5. Sample CRUD Workflow in MongoDB**

1. **Connect to MongoDB:** Ensure mongod is running, then open mongosh.
2. **Create/Choose Database:** use demodb
3. **Create Collection:** db.createCollection("people")
4. **Insert Data:** Use insertOne or insertMany.
5. **Read Data:** Use find, findOne, with optional filters and projections.
6. **Update Data:** Use updateOne, $set for field updates.
7. **Delete Data:** Use deleteOne or deleteMany.
8. **Drop Collection/Database:** Use drop/dropDatabase as needed.

### 6. Practical Example

Suppose you have a database called trainingdb and a collection people:

```
mydb> db.people.insertMany([
...     { user_id: "bc101", status: "A", age: 25 },
...     { user_id: "bc102", status: "B", age: 30 },
...     { user_id: "bc103", status: "A", age: 50 },
...     { user_id: "xy201", status: "A", age: 35 },
...     { user_id: "mn301", status: "B", age: 20 }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687e899e6c8dbb453eeec4ac'),
    '1': ObjectId('687e899e6c8dbb453eeec4ad'),
    '2': ObjectId('687e899e6c8dbb453eeec4ae'),
    '3': ObjectId('687e899e6c8dbb453eeec4af'),
    '4': ObjectId('687e899e6c8dbb453eeec4b0')
  }
}
mydb>
```

- Find all: db.people.find()

```
mydb> db.people.find()
[
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ac'),
    user_id: 'bc101',
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ad'),
    user_id: 'bc102',
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ae'),
    user_id: 'bc103',
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4af'),
    user_id: 'xy201',
    status: 'A',
    age: 35
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4b0'),
    user_id: 'mn301',
    status: 'B',
    age: 20
  }
]
mydb>
```

- Filter: db.people.find({ status: "A" })

```
mydb> db.people.find({ status: "A" })
[
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ac'),
    user_id: 'bc101',
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ae'),
    user_id: 'bc103',
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4af'),
    user_id: 'xy201',
    status: 'A',
    age: 35
  }
]
mydb>
```

- Sort: db.people.find().sort({ user_id: 1 })

```
mydb> db.people.find().sort({ user_id: 1 })
[
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ac'),
    user_id: 'bc101',
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ad'),
    user_id: 'bc102',
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ae'),
    user_id: 'bc103',
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4b0'),
    user_id: 'mn301',
    status: 'B',
    age: 20
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4af'),
    user_id: 'xy201',
    status: 'A',
    age: 35
  }
]
mydb>
```

- Count: db.people.countDocuments({})

```
mydb> db.people.countDocuments({})
5
mydb>
```

- Pattern: db.people.find({ user_id: /^bc/ })

```
mydb> db.people.find({ user_id: /^bc/ })
[
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ac'),
    user_id: 'bc101',
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ad'),
    user_id: 'bc102',
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e899e6c8dbb453eeec4ae'),
    user_id: 'bc103',
    status: 'A',
    age: 50
  }
]
mydb>
```

- Distinct: db.people.distinct("status")

```
mydb> db.people.distinct("status")
[ 'A', 'B' ]
mydb>
```

- Update: db.people.updateOne({ user_id: "bc101" }, { $set: { age: 26 } })

```
mydb> db.people.updateOne({ user_id: "bc101" }, { $set: { age: 26 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mydb>
```

- Delete: db.people.deleteOne({ user_id: "mn301" })

```
mydb> db.people.deleteOne({ user_id: "mn301" })
{ acknowledged: true, deletedCount: 1 }
mydb>
```

*This covers all fundamental MongoDB CRUD operations you'll commonly use in real-world projects.*

**7. Additional Expert Insights**

- **Schema Design:** Though MongoDB is schema-less, plan document structures up front for consistency and efficiency, using embedded documents for “belongs-to” scenarios and references for one-to-many relationships in large datasets.
- **Indexing:** Create indexes on frequently queried fields (db.collection.createIndex()) to significantly improve query performance.
- **Sharding:** For very large datasets, consider sharding collections to distribute data across multiple servers and balance load.
- **Aggregation Framework:** Use db.collection.aggregate() for complex data analysis—grouping, summarizing, and transforming data.
- **Security:** Enable authentication, role-based access, and network-level protections in production deployments.
- **Backup and Restore:** Use mongodump and mongorestore for efficient database backup and recovery.
- **Monitoring:** Use MongoDB’s built-in tools or external solutions (e.g., Ops Manager, Atlas) to monitor performance and get alerts on anomalies.

**8. Conclusion**

MongoDB’s flexible architecture, rich query language and scalability features make it a top choice for developers building data-intensive applications. Mastering the foundational commands, patterns, and best-practices outlined above sets a strong base for both learning and deploying full-scale MongoDB solutions.