MongoDB Commands – Complete Deep Explanation

==========

(This PDF includes every explanation: why each command is used, implementation, and practical examples.)

1. listDatabases

Why:

Lists all databases on the server. Useful for exploration, debugging, auditing.

Implementation:

{ listDatabases: 1 }

Practical:

Check which databases exist on a production server.

2. listCommands

Why:

Shows all available runCommand operations. Good for discovery.

Implementation:

{ listCommands: 1 }

Practical:

Check if your version supports a command like 'profile'.

3. dbStats

Why:

Shows storage size, indexes, number of collections.

Implementation:

{ dbStats: 1 }

Practical:

Used for disk usage monitoring.

4. dropDatabase

Why:

Removes entire database.

Implementation:

{ dropDatabase: 1 }

Practical:

Delete temporary dev database.

5. create (collection)

Why:

Create collection intentionally with options.

Implementation:

{ create: "myCollection" }

Practical:

Create capped or validated collection.

6. drop (collection)

Why:

Delete collection.

Implementation:

{ drop: "myCollection" }

Practical:

Cleanup temp data.

7. listCollections

Why:

See all collections in DB.

Implementation:

{ listCollections: 1 }

Practical:

Explore unknown database.

8. renameCollection

Why:

Rename collection without copying.

Implementation:

{ renameCollection: "db.old", to: "db.new" }

Practical:

Rename tempUsers → users.

9. validate

Why:

Check corruption, index integrity.

Implementation:

{ validate: "collection" }

Practical:

After crash, validate integrity.

10. insert

Why:

Low-level insert.

Implementation:

{ insert: "col", documents: [...] }

Practical:

Use in custom drivers or scripts.

11. find

Why:

Low-level query.

Implementation:

{ find: "col", filter: {...} }

Practical:

Used by drivers internally.

12. update

Why:

Bulk update logic, low-level.

Implementation:

{ update: "col", updates: [...] }

Practical:

Custom migration scripts.

13. delete

Why:

Delete docs using command-level control.

Implementation:

{ delete: "col", deletes: [...] }

Practical:

Bulk delete logs.

14. createIndexes

Why:

Create indexes for performance.

Implementation:

{ createIndexes: "col", indexes: [...] }

Practical:

Fix slow queries by adding index.

15. listIndexes

Why:

See all indexes.

Implementation:

{ listIndexes: "col" }

Practical:

Check if index already exists.

16. dropIndexes

Why:

Remove unnecessary indexes.

Implementation:

{ dropIndexes: "col", index: "name_index" }

Practical:

Delete unused index to improve memory.

17. replSetGetStatus

Why:

See replica set health.

Implementation:

{ replSetGetStatus: 1 }

Practical:

Debug replication lag.

18. replSetGetConfig

Why:

View replica set configuration.

Implementation:

{ replSetGetConfig: 1 }

Practical:

Inspect or modify priorities.

19. replSetStepDown

Why:

Force new primary election.

Implementation:

{ replSetStepDown: 60 }

Practical:

Maintenance on primary.

20. enableSharding

Why:

Enable database for sharding.

Implementation:

{ enableSharding: "db" }

Practical:

Prepare large DB for scaling.

21. shardCollection

Why:

Shard specific collection.

Implementation:

{ shardCollection: "db.col", key: {_id:1} }

Practical:

Distribute large dataset across shards.

22. getShardMap

Why:

See shard distribution.

Implementation:

{ getShardMap: 1 }

Practical:

Audit cluster layout.

23. serverStatus

Why:

Full metrics snapshot.

Implementation:

{ serverStatus: 1 }

Practical:

Analyze performance issues.

24. hostInfo

Why:

See OS-level server info.

Implementation:

{ hostInfo: 1 }

Practical:

Verify server architecture.

25. currentOp

Why:

List running operations.

Implementation:

{ currentOp: 1 }

Practical:

Find stuck queries.

26. killOp

Why:

Kill long-running op.

Implementation:

{ killOp: 1, op: }

Practical:

Stop dangerous COLLSCAN.

27. logRotate

Why:

Rotate MongoDB logs.

Implementation:

{ logRotate: 1 }

Practical:

Prevent disk from filling.

28. createUser

Why:

Create database users.

Implementation:

{ createUser: "admin", pwd: "...", roles:[...] }

Practical:

Add app-level user with readWrite.

29. dropUser

Why:

Remove database user.

Implementation:

{ dropUser: "admin" }

Practical:

Revoke access when employee leaves.

30. usersInfo

Why:

List all users.

Implementation:

{ usersInfo: 1 }

Practical:

Security audits.

31. grantRolesToUser

Why:

Add new roles.

Implementation:

{ grantRolesToUser: "user", roles:["read"] }

Practical:

Give analytics user read-only access.

32. getLog

Why:

Fetch logs (global, startupWarnings).

Implementation:

{ getLog: "global" }

Practical:

Read warnings without SSH.

33. getParameter

Why:

Read server config parameters.

Implementation:

{ getParameter: "*" }

Practical:

Check logLevel, cache configs.

This PDF contains all core commands, explanations, implementation formats, and use-case examples.