mongotop

**mongotop** returns the amount of time a MongoDB instance spends performing read and write operations. It is broken down by collection (namespace). This allows you to make sure there is no unexpected activity and see where resources are consumed. All active namespaces are reported.

## Commands

MongoDB provides several commands that can be used to collect the different metrics from your database presented in [Part 1](https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger). Here are the most useful ones.

### serverStatus

**serverStatus** (db.serverStatus() if run from the mongo shell) is the most complete native metrics-gathering command for MongoDB. It provides a document with statistics from most of the key metrics categories we talked about in [Part 1](https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger): connections, operations, journaling, background flushing, locking, cursors, memory, asserts, etc. You can find the full list of metrics it can return [here](https://docs.mongodb.com/manual/reference/command/serverStatus/#output).

This command is used by most [third party monitoring tools](https://www.datadoghq.com/blog/collecting-mongodb-metrics-and-statistics/#production-monitoring) to collect MongoDB metrics along with the dbStats and replSetGetStatus commands that are still necessary to collect storage metrics and statistics about your replica sets (see next paragraphs).

### dbStats

**dbStats** (db.stats() in the mongo shell) provides metrics about storage usage of the database: number of objects, or memory taken by documents and padding in the database (see memory metrics in [Part 1](https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger) of this series). [Here](https://docs.mongodb.com/manual/reference/command/dbStats/#output) is the full list of metrics it returns.

### collStats

**collStats** (db.collection.stats() in the shell) returns metrics similar to the dbStats output but  for a specified collection: size of a collection, number of objects inside it, average size of objects, number of indexes in the collection, etc. See the full list [here](https://docs.mongodb.com/manual/reference/command/collStats/#output).

For example the following command runs collStats on the “movie” collection, with a scale of 1024 bytes:

db.runCommand( { collStats : “restaurant”, scale: 1024 } )

### getReplicationInfo

getReplicationInfo (db.printReplicationInfo() in the shell) returns metrics about oplogs of the different members of a replica set like the oplog size or the oplog window. See the list of output fields [here](https://docs.mongodb.com/manual/reference/method/db.printReplicationInfo/#output-fields).

### replSetGetStatus

**replSetGetStatus** (rs.status() from the shell) reports metrics about members of your replica set: state, metrics required to calculate replication lag. [See Part 1](https://www.datadoghq.com/blog/monitoring-mongodb-performance-metrics-wiredtiger) for more info about these metrics. This command is used to check the health of a replica set’s members and make sure replication is correctly configured. You can find the full list of metrics of the output [here](https://docs.mongodb.com/manual/reference/command/replSetGetStatus/#output).

### sh.status

Sh.status (sh.status() from the shell) provides metrics about sharding configuration and existing chunks (contiguous range of shard key values in a specific [shard](https://docs.mongodb.com/manual/reference/glossary/#term-shard)) for a sharded cluster. The full list of metrics of the output is available [here](https://docs.mongodb.com/manual/reference/method/sh.status/#output-fields).

### getProfilingStatus

getProfilingStatus (db.getProfilingStatus() in the shell) returns the current [profile](https://docs.mongodb.com/manual/reference/command/profile/#dbcmd.profile) level and the defined threshold above which the profiler considers a query slow (slowOpThresholdMs).