About this guide

This guide covers:

- Connecting to MongoDB
- Tuning database connection (concurrency level, automatic reconnection, timeouts, etc)
- Connecting to MongoDB using connection URI
- Connecting to replica sets
- Connecting in PaaS environments, for example, MongoHQ add-on on Heroku

What version of Monger does this guide cover?

This guide covers Monger 3.1 (including preview releases).

Overview

Before using Monger, you need to connect to MongoDB and choose a database to work with. Monger supports working with multiple connections and databases.

To connect, use `monger.core/connect` function which returns a connection:

```
(ns my.service.server
  (:require [monger.core :as mg])
  (:import [com.mongodb MongoOptions ServerAddress]))

;; localhost, default port
(let [conn (mg/connect)])

;; given host, default port
(let [conn (mg/connect {:host "db.megacorp.internal"})])


;; given host, given port
(let [conn (mg/connect {:host "db.megacorp.internal" :port 7878})])
```

To choose a database, use `monger.core/get-db`:

```
(ns my.service.server
  (:require [monger.core :as mg]))

(let [conn (mg/connect)
      db   (mg/get-db conn "monger-test")])
```

Connecting To Mongodb Using Connection Options

Passing no arguments to `monger.core/connect` is very common for development but production environments typically require connection options tweaks.

This is done by using 2-arity form of `monger.core/connect!` that takes a server address and connection options. Because MongoDB connections have a good dozen of tunable parameters, Monger provides a function that takes a map of them and produces an object that can be used as connection options:

```
(ns my.service.server
  (:require [monger.core :as mg])
  (:import [com.mongodb MongoOptions ServerAddress]))

;; using MongoOptions allows fine-tuning connection parameters,
;; like automatic reconnection (highly recommended for production environment)
(let [^MongoOptions opts (mg/mongo-options :threads-allowed-to-block-for-connection-multiplier 300)
      ^ServerAddress sa  (mg/server-address "127.0.0.1" 27017)
      conn               (mg/connect sa opts)]
  )
```

Supported Connection Options

:connections-per-host (default: 10)

The maximum number of connections allowed per host for this Mongo instance. Those connections will be kept in a pool when idle. Once the pool is exhausted, any operation requiring a connection will block waiting for an available connection.

:threads-allowed-to-block-for-connection-multiplier (default: 5)

This multiplier, multiplied with the connectionsPerHost setting, gives the maximum number of threads that may be waiting for a connection to become available from the pool. All further threads will get an exception right away. For example if connectionsPerHost is 10 and threadsAllowedToBlockForConnectionMultiplier is 5, then up to 50 threads can wait for a connection.

:max-wait-time (default: 120,000)

The maximum wait time in milliseconds that a thread may wait for a connection to become available. A value of 0 means that it will not wait. A negative value means to wait indefinitely.

:connect-timeout (default: 0)

The connection timeout in milliseconds. Only used for new connections. A value of 0 means no timeout.

:socket-timeout (default: 0)

The socket timeout in milliseconds. A value of 0 means no timeout.

:socket-keep-alive (default: false)

This flag controls the socket keep alive feature that keeps a connection alive through firewalls.

auto-connect-retry (default: true)

If true, the client will keep trying to connect to the same server in case that the socket cannot be established.

max-auto-connect-retry-time (default: 15 seconds)

he maximum amount of time in milliseconds to spend retrying to open connection to the same server.

w

`:w` value of the default write concern of the connection. With Monger, setting default write concern using `monger.core/set-default-write-concern!` largely eliminates the need for this option.

w-timeout

`:w-timeout` value of the default write concern of the connection. With Monger, setting default write concern using `monger.core/set-default-write-concern!` largely eliminates the need for this option.

fsync

`:fsync` value of the default write concern of the connection: should all operations wait for server to sync changes to disk? With Monger, setting default write concern using `monger.core/set-default-write-concern!` largely eliminates the need for this option.

Connecting Using URI (Heroku, CloudFoundry, etc)

In certain environments, for example, Heroku or other PaaS providers, the only way to connect to MongoDB is via connection URI.

Monger provides `monger.core/connect-via-uri` function that combines `monger.core/connect`, `monger.core/get-db`, and `monger.core/authenticate` and works with string URIs like `mongodb://userb71148a:0da0a696f23a4ce1ecf6d11382633eb2049d728e@cluster1.mongohost.com:27034/app81766662`.

`monger.core/connect-via-uri` returns a map with two keys:

- `:conn`
- `:db`

It can be used to connect with or without authentication, for example:

```
;; connect without authentication
(let [uri               "mongodb://127.0.0.1/monger-test4"
      {:keys [conn db]} (mg/connect-via-uri uri)])

;; connect with authentication
(let [uri               "mongodb://clojurewerkz/monger!:monger!@127.0.0.1/monger-test4"
      {:keys [conn db]} (mg/connect-via-uri "mongodb://127.0.0.1/monger-test4")])

;; connect using connection URI stored in an env variable, in this case, MONGOHQ_URL
(let [uri               (System/genenv "MONGOHQ_URL")
      {:keys [conn db]} (mg/connect-via-uri "mongodb://127.0.0.1/monger-test4")])
```

It is also possible to pass connection options as query parameters:

```
(let [uri               "mongodb://localhost/test?maxPoolSize=128&waitQueueMultiple=5;waitQueueTimeoutMS=150;socketTimeoutMS=5500&autoConnectRetry=true;safe=false&w=1;wtimeout=2500;fsync=true"
      {:keys [conn db]} (mg/connect-via-uri "mongodb://127.0.0.1/monger-test4")])
```

Connecting to a Replica Set

Monger supports connecting to replica sets using one or more seeds when calling `monger.core/connect` with a collection of server addresses instead of just a single one:

```
(ns my.service
  (:require monger.core :as mg))

;; Connect to a single MongoDB instance
(mg/connect (mg/server-address "127.0.0.1" 27017) (mg/mongo-options))

;; Connect to a replica set
(mg/connect [(mg/server-address "127.0.0.1" 27017)
             (mg/server-address "127.0.0.1" 27018)]
         (mg/mongo-options))
```

`monger.core/connect!` function works exactly the same way.

Authentication

Monger supports authenticated connections. Authentication uses a set of credentials and happens against a database. `monger.core/connect-with-credentials` is the function that opens a connection with authentication. Some `monger.core/connect` overloads accept a set of credentials as well.

Starting with version 3.0, MongoDB Java client supports multiple credential types. Monger provides convenient helpers for two most commonly used ones:

- Plan (username and password)
- x509 certificates

`monger.credentials` is the namespace with functions that return credentials accepted by `monger.core/connect-with-credentials`.

Plain Credentials (Username and Password)

In the example above, a connection is opened with a set of plain credentials: username `"username"`, password `"password"`, and database `"some-db"`:

```
(ns monger.docs.examples
  (:require [monger.core :as mg]
            [monger.credentials :as mcred]))

(let [db   "some-db"
      u    "username"
      p    "password"]
```

```
  (mg/connect-with-credentials "127.0.0.1" (mcred/create u db p))
```

`monger.core/connect-with-credentials` will return a connection if authentication succeeds and throw an exception otherwise.

To connect to a replicate set that requires authentication with Monger, use `monger.core/connect` with 3 arguments: a set of endpoints, connection optoins, and a set of credentials. See the "Connecting to a Replica Set" section above.

Disconnecting

To disconnect, use `monger.core/disconnect`:

```
(ns my.service.server
  (:require [monger.core :as mg]))

(let [conn (mg/connect)]
  (mg/disconnect conn))
```

What to Read Next

The documentation is organized as a number of guides, covering all kinds of topics.

We recommend that you read the following guides first, if possible, in this order:

- Inserting documents
- Querying & finders
- Updating documents
- Deleting documents
- Indexing and other collection operations
- Integration with 3rd party libraries
- Map/Reduce
- GridFS support
- Using MongoDB Aggregation Framework
- Using MongoDB commands

Tell Us What You Think!

Please take a moment to tell us what you think about this guide on Twitter or the Monger mailing list

Let us know what was unclear or what has not been covered. Maybe you do not like the guide style or grammar or discover spelling mistakes. Reader feedback is key to making the documentation better.

comments powered by Disqus

This website was developed by ClojureWerkz team.

Follow us on Twitter: ClojureWerkz, Michael Klishin, Alex P

Artwork by zuk13