MongoDB

Subject (namespace)

Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Status | By | Changes |
| 1.0 | 05-08-2014 | Draft | Komyos C. | Initial version. |

Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Status | To |
| 1.0 | 08-05-2014 | Draft | Team |

References

|  |  |
| --- | --- |
| Document number | Document title |
| PS-NW-TD-XXXX |  |

**Table of Contents**

[1 Preface 1](#_Toc411933604)

[1.1 Who should read this document 1](#_Toc411933605)

[2 Glossary 1](#_Toc411933606)

[3 About MongoDB 2](#_Toc411933607)

[4 Installation (local) 2](#_Toc411933608)

[5 Mongo Configuration Options 7](#_Toc411933609)

[5.1 Core Options 8](#_Toc411933610)

[6 Install MongoDB Services Single Machine 27](#_Toc411933611)

[7 Sharding 31](#_Toc411933612)

[7.1 Overview 31](#_Toc411933613)

[7.2 Setting up step 32](#_Toc411933614)

[7.2.1 Setting Config Server 33](#_Toc411933615)

[7.2.2 Setting Router Server 35](#_Toc411933616)

[7.2.3 Initial Shard Server 38](#_Toc411933617)

[7.2.4 Adding mongodb as shard to the router 41](#_Toc411933618)

[7.3 Scalability 43](#_Toc411933619)

[7.3.1 Connect to mongo router 43](#_Toc411933620)

[7.3.2 Add shard 43](#_Toc411933621)

[7.4 --run command router server 43](#_Toc411933622)

[8 Replication 43](#_Toc411933623)

[8.1 Introduction 43](#_Toc411933624)

[8.1.1 Replication in MongoDB 43](#_Toc411933625)

[8.2 Replica Set Installation 46](#_Toc411933626)

[8.2.1 Prepare Configuration 47](#_Toc411933627)

[8.2.2 Install replica services 47](#_Toc411933628)

[8.2.3 Add member of replica set 48](#_Toc411933629)

[8.2.4 Initial Replica Set 48](#_Toc411933630)

[8.2.5 Add Secondary 48](#_Toc411933631)

[8.2.6 Add Arbiter 48](#_Toc411933632)

[8.2.7 Check Replication Status 48](#_Toc411933633)

[9 Sharding with Replication 48](#_Toc411933634)

[10 Enable Authentication in Shared Cluster 48](#_Toc411933635)

[10.1 Enable authentication on each component in the cluster 49](#_Toc411933636)

[10.2 Create user 49](#_Toc411933637)

[50](#_Toc411933638)

[10.2.1 Built-in Roles 50](#_Toc411933639)

[11 How to Build MongoDB with SSL 56](#_Toc411933640)

[11.1 BuildingStep 57](#_Toc411933641)

[11.2 Requirement software for server 58](#_Toc411933642)

[12 Problem on setting up 58](#_Toc411933643)

[12.1 cannot start Routing server 58](#_Toc411933644)

[12.2 I’ve enabled sharding and added a second shard, but all the data is still on one server. Why? 58](#_Toc411933645)

[12.3 Run command “shardCollection” 59](#_Toc411933646)

[12.4 How to solve “The specified service has been marked for deletion” error 60](#_Toc411933647)

[13 Appendices 61](#_Toc411933648)

# Preface

[Introduction to the document]

## Who should read this document

[Describe the intended audience]

# Glossary

**NoSQL** A database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

**Mongos** Is a routing service for MongoDB shard configurations that processes queries from the application layer, and determines the location of this data in the sharded cluster.

**Mongod** Is the primary daemon process for the MongoDB system. It handles data requests, manages data access, and performs background management operations

**MongoDB** An open-source document-based database system. “MongoDB” derives from the word “humongous” because of the database’s ability to scale up with ease and hold very large amounts of data. MongoDB stores documents in collections within databases.

**OpLog** A capped collection that stores an ordered history of logical writes to a MongoDB database. The oplog is the basic mechanism enabling replication in MongoDB.

**Replica Set** A cluster of MongoDB servers that implements master-slave replication and automated failover. MongoDB’s recommended replication strategy

**YAML** Is a human friendly data serialization standard for all programming language, The MongoDB use this format for configuration file.

**Primary** In a replica set, the primary member is the current master instance, which receives all write operations.

**Secondary** A replica set member that replicates the contents of the master database. Secondary members may handle read requests, but only the primary members can handle write operations.

**Set name** The arbitrary name given to a replica set. All members of a replica set must have the same name specified with the replSetName setting or the --replSet option.

**Shard** A single mongod instance or replica set that stores some portion of a sharded cluster’s total data set. In production, all shards should be replica sets.

**Shard Key** The field MongoDB uses to distribute documents among members of a sharded cluster.

# About MongoDB

\*\*\*In this document we use the Mongo version 2.6.7

**MongoDB** (from humongous) is a cross-platform document-oriented database. Classified as a **NoSQL** database, **MongoDB** eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (**MongoDB** calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, **MongoDB** is free and open-source software.

1. Document Database
2. Documents (objects) map nicely to programming language data types.
3. Embedded documents and arrays reduce need for joins.
4. Dynamic schema makes polymorphism easier.
5. High Performance
6. Embedding makes reads and writes fast.
7. Indexes can include keys from embedded documents and arrays.
8. Optional streaming writes (no acknowledgments).
9. High Availability
10. Replicated servers with automatic master failover.
11. Easy Scalability
12. Automatic sharding distributes collection data across machines.
13. Eventually-consistent reads can be distributed over replicated servers.
14. Advanced Operations
15. With **MongoDB** Management Service (MMS) **MongoDB** supports a complete backup solution and full deployment monitoring.

# Installation (local)

**PLATFORM SUPPORT**

Starting in version 2.2, **MongoDB** does not support Windows XP. Please use a more recent version of Windows to use more recent releases of **MongoDB**.

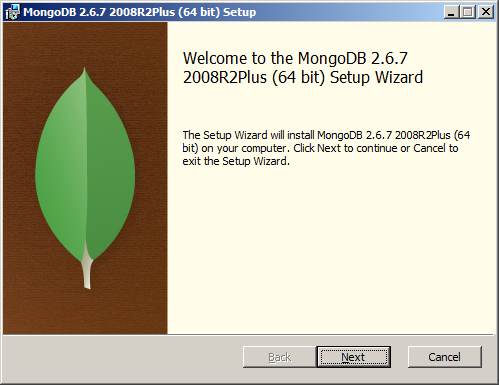
**IMPORTANT**

If you are running any edition of Windows Server 2008 R2 or Windows 7, please install a hotfix to resolve an issue with memory mapped files on Windows.

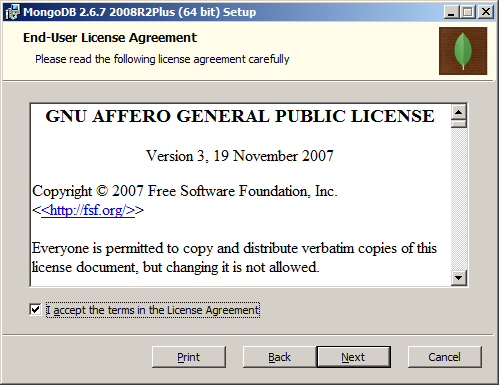
**Requirements**

On Windows **MongoDB** requires Windows Server 2008 R2, Windows Vista, or later. The MSI installer includes all other software dependencies.

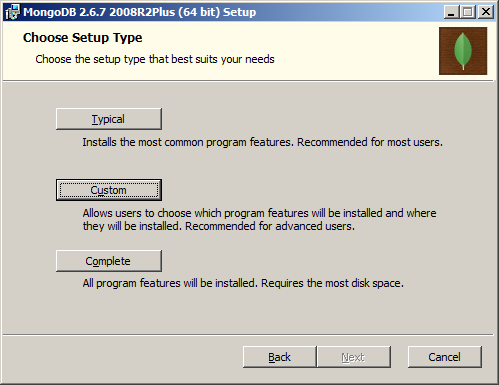
Link for download **MongoDB** setup we choose 64 bit platform <http://www.mongodb.org/downloads?_ga=1.104025928.912530068.1422589415>



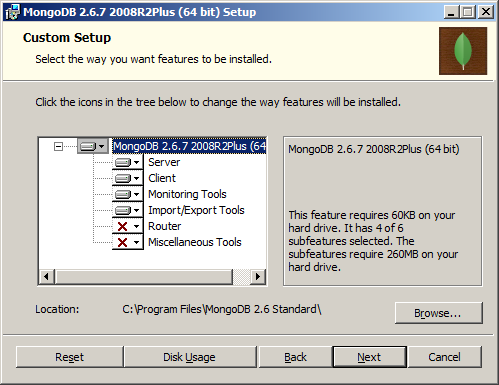
Figure



Figure

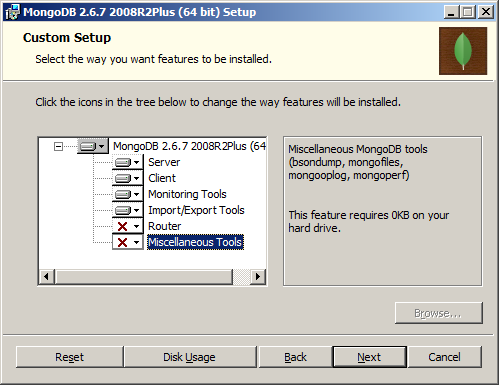


Figure

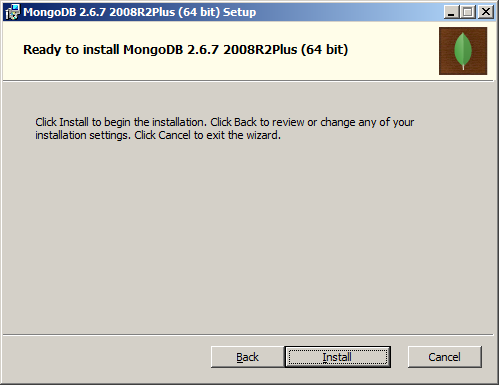


Figure

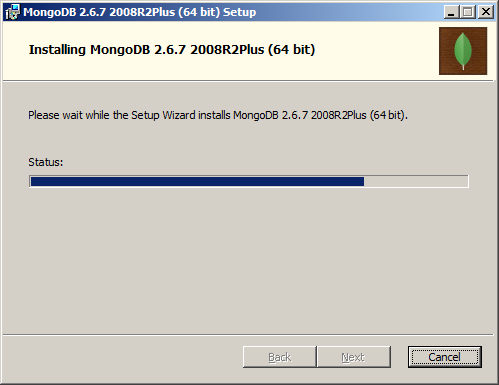
The options Router is need when we need to create the “Sharding” and this machine provide to **Query Router Server**. If you don’t need it can leave it for now.



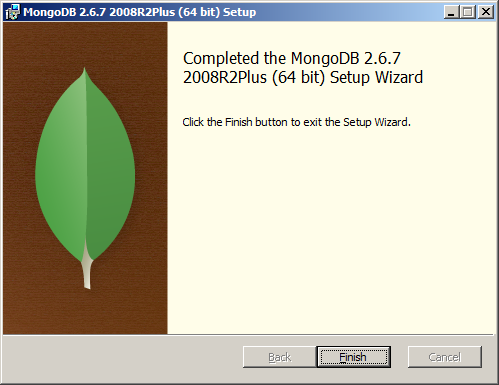
Figure



Figure



Figure



Figure

# Mongo Configuration Options

Before we installation and setup database of mongo, we must understand basic structure of mongo configuration. After we run installation of mongo setup file on windows. In this document we use default location of mongo configuration that is “C:\Program Files\MongoDB 2.6 Standard” , The important directory is “bin”, it contains all MongoDB binary files. For more details can see in table below.

**Components**

|  |  |
| --- | --- |
| Binary Files | Description |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongod.exe | The Database process, use it for create mongo db service, Shrading, Arbiter, Replicate |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongos.exe | Sharding Controller, use it for router |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongo.exe | The Database shell (uses interactive JavaScript), we use it to configuration sharding, Replicate |

**Utilities**

| Binary Files | Description |
| --- | --- |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongodump.exe | MongoDB dump tool , use it for backups, snapshots. |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongorstore.exe | MongoDB restore a dump. |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongoexport.exe | Export a single collection to test (JSON, CSV). |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongoimport.exe | Import from JSON or CSV. |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongofiles.exe | Utility for putting and getting files from MongoDB GridFS. |
| C:\Program Files\MongoDB 2.6 Standard\bin\mongostat | Show performance statistics. |

Before we can setup database we will create directories for store configuration, db files and logging.

|  |  |
| --- | --- |
| Directory | Description |
| C:\mongodb\Config | This directory contains configuration of mongodb such as router conig, database config |
| C:\mongo\db | This directory contains database files |
| C:\mongo\log | This directory contains logging file of mongodb |

We can control mongod and mongos instances at runtime using a configuration file. The configuration file contains settings that are functionally equivalent to the mongod and mongos command-line arguments but are easier to manage, especially on large-scale deployments. Configuration files allow commenting to describe the reasoning behind a server’s settings.

The MongoDB Configuration using **YAML** Standard Format.

For example

systemLog:

destination: file

path: "/var/log/mongodb/mongodb.log"

logAppend: true

storage:

journal:

enabled: true

processManagement:

fork: true

net:

bindIp: 127.0.0.1

port: 27017

...

## Core Options

**systemLog**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| verbosity | integer | Increases the amount of internal reporting returned on standard output or in log files. To increase the amount of information reported, increase the number. |  |
| quiet | boolean | Increases the amount of internal reporting returned on standard output or in log files. To increase the amount of information reported, increase the number. |  |
| traceAllExceptions | boolean | Prints verbose information for debugging. Used for additional logging in support-related troubleshooting |  |
| syslogFacility | string | Specifies the facility level used when logging messages to syslog. The value you specify must be supported by your operating system’s implementation of syslog. To use this option, you must enable the [--syslog](http://docs.mongodb.org/manual/reference/program/mongos/#cmdoption--syslog) option | Default: user |
| path | string | Sends all diagnostic logging information to a log file instead of to standard output or to the host’s [syslog](http://docs.mongodb.org/manual/reference/glossary/#term-syslog) system. MongoDB creates the log file at the path you specify. | By default, MongoDB overwrites the log file when the process restarts. To instead append to the log file, set the [--logappend](http://docs.mongodb.org/manual/reference/program/mongos/#cmdoption--logappend) option. |
| logAppend | boolean | Appends new entries to the end of the log file rather than overwriting the content of the log when the [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) instance restarts. | Default: False |
| destination | string | Specify either file or syslog. If you specify file you must also specify systemLog.path. If you do not specify systemLog.destination, MongoDB will send all log output lo standard output |  |
| timeStampFormat | string | The time format for timestamps in log messages. Specify one of the following values:  ctime = displays timestamps as Wed Dec 31 18:17:54.811  iso8601-utc = Displays timestamps in Coordinated Universal Time (UTC) in the ISO-8601 format. For example, for New York at the start of the Epoch : 1970-01-01T00:00:00.000Z  iso8601-local = Displays timestamps in local time in the ISO-8601 format. For example, for New York at the start of the Epoch: 1969-12-31T19:00:00.000+0500 |  |

**processManagement**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| pidFilePath | string | Specifies a file location to hold the process ID of the [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) process where [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) will write its PID. This is useful for tracking the [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) process in combination with the [--fork](http://docs.mongodb.org/manual/reference/program/mongos/#cmdoption--fork) option. Without a specified processManagement.pidFilePath option, the process creates no PID file. |  |
| fork | boolean | Enables a [daemon](http://docs.mongodb.org/manual/reference/glossary/#term-daemon) mode that runs the [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) process in the background. By default [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) does not run as a daemon: typically you will run [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) or [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) as a daemon, either by using processManagement.fork or by using a controlling process that handles the daemonization process (e.g. as with upstart and systemd) | Default: False |

**net**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| port | integer | Specifies the TCP port on which the MongoDB instance listens for client connections. | Default: 27017 |
| bindIp | string | Specifies the IP address that mongos or mongod binds to in order to listen for connections from applications. You may attach mongos or mongod to any interface. When attaching mongos or mongod to a publicly accessible interface, ensure that you have implemented proper authentication and firewall restrictions to protect the integrity of your database  To bind to multiple IP addresses, enter a list of comma separated values. | Default: All interfaces. |
| maxIncommingConnections | integer | The maximum number of simultaneous connections that mongos or mongod will accept. This setting has no effect if it is higher than your operating system’s configured maximum connection tracking threshold.  This is particularly useful for a mongos if you have a client that creates a number of connections but allows them to timeout rather than close the connections. When you set this option, ensure the value is slightly higher than the size of the connection pool or the total number of connections, to prevent erroneous connection spikes from propagating to the members of a sharded cluster. | Default: 1000000 |
| wireObjectCheck | boolean | Forces the mongod or mongos instance to validate all requests from clients upon receipt to prevent clients from inserting malformed or invalid BSON into a MongoDB database.  For objects with a high degree of sub-document nesting, net.wireObjectCheck can have a small impact on performance. | Default: True |
| http.enabled | boolean | Enables the HTTP interface. Enabling the interface can increase network exposure.  Leave the HTTP interface disabled for production deployments. If you do enable this interface, you should only allow trusted clients to access this port. | Default: False |
| unixDomainSocket.enabled | boolean | Disables listening on the UNIX domain socket. The mongos or mongod process always listens on the UNIX socket unless one of the following is true:  net.unixDomainSocket.enabled is set  bindIp is not set  bindIp does not specify 127.0.0.1 | Default: False |
| unixDomainSocket.pathPrefix | string | The path for the UNIX socket. If this option has no value, the mongos or mongod process creates a socket with /tmp as a prefix. MongoDB creates and listens on a UNIX socket unless one of the following is true:  --nounixsocket is set  bindIp is not set  bindIp does not specify 127.0.0.1 | Default: /tmp |
| ipv6 | boolean | Enables IPv6 support and allows the mongos or mongod to connect to the MongoDB instance using an IPv6 network. All MongoDB programs and processes disable IPv6 support by default. | Default: False |
| http.JSONPEnabled | boolean | Permits JSONP access via an HTTP interface. Enabling the interface can increase network exposure. The net.http.JSONPEnabled option enables the HTTP interface, even if the HTTP interface option is disabled.  The net.http.JSONPEnabled option is available only for mongod. | Default: False |
| http.RESTInterfaceEnabled | boolean | Enables the simple REST API. Enabling the REST API enables the HTTP interface, even if the HTTP interface option is disabled, and as a result can increase network exposure.  This option is available only for mongod. | Default: False |
| ssl.sslOnNormalPorts | boolean | With this opention, a mongos or mongod requires SSL encryption for all connections on the default MongoDB port, or the port specified by --port. By default, --sslOnNormalPorts is disabled.  The default distribution of MongoDB does not contain support for SSL. For more information on MongoDB and SSL, see Configure mongod and mongos for SSL. | Deprecated since version 2.6. |
| ssl.mode | string | Enables SSL or mixed SSL used for all network connections. The argument to the net.ssl.mode option can be one of the following:  Disable = The server does not use SSL  allowSSL = Connections between servers do not use SSL. For incoming connections the server accepts both SSL and non-SSL.  preferSSL = Connections between servers use SSL. For incoming connections, the server uses and accepts only SSL and non-SSL  require = The Server uses  The default distribution of MongoDB does not contain support for SSL. For more information on MongoDB and SSL | The default distribution of MongoDB does not contain support for SSL. To use SSL, you must either build MongoDB locally passing the --ssl option to scons or use MongoDB Enterprise.  see http://docs.mongodb.org/manual/tutorial/configure-ssl/ for more information. |
| ssl.PEMKeyFile | string | Specifies the .pem file that contains both the SSL certificate and key. Specify the file name of the .pem file using relative or absolute paths.  When SSL is enabled, you must specify net.ssl.PEMKeyFile. |  |
| ssl.PEMKeyPassword | string | Specifies the password to de-crypt the certificate-key file (i.e. --sslPEMKeyFile). Use the net.ssl.PEMKeyPassword option only if the certificate-key file is encrypted. In all cases, the mongos or mongod will redact the password from all logging and reporting output.  Changed in version 2.6: If the private key in the PEM file is encrypted and you do not specify the net.ssl.PEMKeyPassword option, the mongos or mongod will prompt for a passphrase. See SSL Certificate Passphrase. |  |
| ssl.clusterFile | string | Specifies the .pem file that contains the x.509 certificate-key file for membership authentication for the cluster or replica set.  If net.ssl.clusterFile does not specify the .pem file for internal cluster authentication, the cluster uses the .pem file specified in the PEMKeyFile option. |  |
| ssl.clusterPassword | string | Specifies the password to de-crypt the x.509 certificate-key file specified with --sslClusterFile. Use the net.ssl.clusterPassword option only if the certificate-key file is encrypted. In all cases, the mongos or mongod will redact the password from all logging and reporting output.  If the x.509 key file is encrypted and you do not specify the net.ssl.clusterPassword option, the mongos or mongod will prompt for a passphrase. See SSL Certificate Passphrase. |  |
| ssl.CAFile | string | Specifies the .pem file that contains the root certificate chain from the Certificate Authority. Specify the file name of the .pem file using relative or absolute paths. | Warning  If the --sslCAFile option and its target file are not specified, x.509 client and member authentication will not function. mongod, and mongos in sharded systems, will not be able to verify the certificates of processes connecting to it against the trusted certificate authority (CA) that issued them, breaking the certificate chain. |
| ssl.CRLFile | string | Specifies the .pem file that contains the Certificate Revocation List. Specify the file name of the .pem file using relative or absolute paths. |  |
| ssl.weakCertificateValidation | boolean | Disables the requirement for SSL certificate validation that --sslCAFile enables. With the this option, the mongos or mongod will accept connections when the client does not present a certificate when establishing the connection.  If the client presents a certificate and the mongos or mongod has net.ssl.weakCertificateValidation enabled, the mongos or mongod will validate the certificate using the root certificate chain specified by --sslCAFile and reject clients with invalid certificates.  Use the this option if you have a mixed deployment that includes clients that do not or cannot present certificates to the mongos or mongod. |  |
| ssl.allowInvalidCertificates | boolean | Bypasses the validation checks for SSL certificates on other servers in the cluster and allows the use of invalid certificates. When using the allowInvalidCertificates setting, MongoDB logs as a warning the use of the invalid certificate. |  |
| ssl.FIPSMode | boolean | Directs the mongos or mongod to use the FIPS mode of the installed OpenSSL library. Your system must have a FIPS compliant OpenSSL library to use the net.ssl.FIPSMode option. |  |
| setParameter | documents | Specifies one of the MongoDB parameters described in MongoDB Server Parameters.( http://docs.mongodb.org/manual/reference/parameters/) You can specify multiple setParameter fields |  |

**Security**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| keyfile | string | Specifies the path to a key file that stores the shared secret that MongoDB instances use to authenticate to each other in a sharded cluster or replica set. keyFile implies security.authorization. See Authentication Between MongoDB Instances for more information. | We use openssl to generate keyfile :  openssl rand -base64 741 > mongodb-keyfile |
| clusterAuthMode | string | The authentication mode used for cluster authentication. If you use, specify so here. This option can have one of the following values:  keyFile = use a keyfile for authentication. Accept only keyfile.  sendKeyFile = For rolling upgrade purposes. Send a keyfile for authentication but can accept both keyfiles and x.509 certificates.  sendX509 = For rolling upgrade purposes. Send the x.509 certificate for authentication but can accept both keyfiles and x.509 certificates.  X509 = Recommended. Send the x.509 certificate for authentication and accept only x.509 certificates. |  |
| authorization | string | Enables Role-Based Access Control (RBAC) to govern each user’s access to database resources and operations.  Set this option to one of the following:  Enabled = A user can access only the database resources and actions for which they have been granted privileges  Disabled = A user can access my database and perform any action. | See  http://docs.mongodb.org/ manual/core/authorization/  for more information  This option is available only for mongod. |
| sasl.hostName | string | A fully qualified server domain name for the purpose of configuring SASL and Kerberos authentication. The SASL hostname overrides the hostname only for the configuration of SASL and Kerberos.  For mongo shell and other MongoDB tools to connect to the new hostName, see the gssapiHostName option in the mongo shell and other tools |  |
| sasl.serviceName | string | Registered name of the service using SASL. This option allows you to override the default Kerberos service name component of the Kerberos principal name, on a per-instance basis. If unspecified, the default value is mongodb.  MongoDB permits setting this option only at startup. The setParameter can not change this setting | This option is available only in MongoDB Enterprise. |
| sasl.saslauthdSocket Path | string | The path to the UNIX domain socket file for saslauthd. |  |
| javascriptEnabled | boolean | Enables or disables the JavaScript execution engine. When disabled, $where, mapReduce, group and any other operation that requires a mongod instance to execute JavaScript. | Default = true |

**operationProfiling**

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Type | Description | Remark |
| slowOpThresholdMs | integer | The threshold in milliseconds at which the database profiler considers a query slow. MongoDB records all slow queries to the log, even when the database profiler is off. When the profiler is on, it writes to the system.profile collection. See the profile command for more information on the database profiler. | This option is available only for mongod. |
| mode | string | Changes the level of database profiling, which inserts information about operation performance into standard output or a log file. Specify one of the following levels:  off = Off. No Profiling  slowOp = On. Only includes slow operations  all = On. Includes all operations | This option is available only for mongod. |

**Storage**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| dbPath | string | The directory where the [mongod](http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod) instance stores its data. | Default: \data\db on Windows  This option is available only for mongod |
| directoryPerDB | boolean | Stores each database’s files in its own folder in the data directory. When applied to an existing system, the directoryPerDB option alters the storage pattern of the data directory.  Use this option in conjunction with your file system and device configuration so that MongoDB will store data on a number of distinct disk devices to increase write throughput or disk capacity. | Default: False |
| indexBuildRetry | boolean | Selects whether mongod rebuilds incomplete indexes on the next start up. This applies in cases where mongod restarts after it has shut down or stopped in the middle of an index build. In such cases, mongod always removes any incomplete indexes, and then, by default, attempts to rebuild them. To stop mongod from rebuilding indexes, set this option to false. | This option is available only for mongod. |
| preallocDataFiles | boolean | Enables or disables data file preallocation.  Do **not** disable data file preallocation in production systems. Only use this option for testing and with small data sets where you frequently drop databases. | Default = True |
| nsSize | integer | Specifies the default size for namespace files, which are files that end in .ns. Each collection and index counts as a namespace.  Use this setting to control size for newly created namespace files. This option has no impact on existing files. The maximum size for a namespace file is 2047 megabytes. The default value of 16 megabytes provides for approximately 24,000 namespaces | Default = 16  This Option is available only for mongod. |
| quota.enforced | boolean | Enables a maximum limit for the number data files each database can have. When running with the this option, MongoDB has a maximum of 8 data files per database. Adjust the quota with storage.quota.maxFilesPerDB. | Default = false  This option is available only for mongod. |
| quota.maxFilesPerDB | integer | Modifies the limit on the number of data files per database. storage.quota.maxFilesPerDB option requires that you set storage.quota.enforced. | Default = 8  This option available only for mongod |
| smallFiles | boolean | Sets MongoDB to use a smaller default file size. The storage.smallFiles option reduces the initial size for data files and limits the maximum size to 512 megabytes. storage.smallFiles also reduces the size of each journal file from 1 gigabyte to 128 megabytes. Use storage.smallFiles if you have a large number of databases that each holds a small quantity of data.  This option can lead the mongod instance to create a large number of files, which can affect performance for larger databases. | This option is available only for mongod |
| syncPeriodSecs | number | Controls how much time can pass before MongoDB flushes data to the data files via an fsync operation.  Do not set this value on production systems.  In almost every situation, you should use the default setting. | Default = 60  This option is available only for mongod. |
| repairPath | string | Specifies a working directory that MongoDB will use during the --repair operation. After --repair completes, the data files in dbPath and the storage.repairPath directory is empty. | Default: A \_tmp directory within the path specified by the dbPath option.  This option is available only for mongod. |
| journal.enabled | boolean | Enables the durability journal to ensure data files remain valid and recoverable. This option applies only when you specify the --dbpath option | Default: true on 64-bit systems, false on 32-bit systems  This option is available only for mongod |
| journal.debugFlags | integer | Provides functionality for testing. Not for general use, and will affect data file integrity in the case of abnormal system shutdown. | This option is available only for mongod. |
| journal.commitIntervalMs | number | The maximum amount of time the mongod process allows between journal operations. Values can range from 2 to 300 milliseconds. Lower values increase the durability of the journal, at the expense of disk performance.  The default journal commit interval is 100 milliseconds if a single block device (e.g. physical volume, RAID device, or LVM volume) contains both the journal and the data files.  If the journal is on a different block device than the data files the default journal commit interval is 30 milliseconds.  To force mongod to commit to the journal more frequently, you can specify j:true. When a write operation with j:true is pending, mongod will reduce commitIntervalMs to a third of the set value. | Default: 100 or 30  This option is available only for mongod. |

**Replication**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| oplogSizeMB | integer | Specifies a maximum size in megabytes for the replication operation log (i.e., the oplog). The mongod process creates an oplog based on the maximum amount of space available. For 64-bit systems, the oplog is typically 5% of available disk space. Once the mongod has created the oplog for the first time, changing the replication.oplogSizeMB option will not affect the size of the oplog. | This option is available only for mongod |
| replSetName | string | Configures replication. Specify a replica set name as an argument to this set. All hosts in the replica set must have the same set name.  If your application connects to more than one replica set, each set should have a distinct name. Some drivers group replica set connections by replica set name. | This option is available only for mongod |
| secondaryIndexPrefetch | string | Determines which indexes secondary members of a replica set load into memory before applying operations from the oplog. By default secondaries load all indexes related to an operation into memory before applying operations from the oplog. This option can have one of the following values:  none= Secondaries do not load indexes into memory  all= Secomdaries load all indexes related to an operation  \_id\_only= Secondaries load no additional indexes into memory beyond the already existing \_id index. | Default = all  This option is available only for mongod |

**Sharding**

|  |  |  |  |
| --- | --- | --- | --- |
| Setting | Type | Description | Remark |
| clusterRole | string | Selects the role the mongod instance has in the sharded cluster. Set this option to one of the following:  configsvr = start this instance as a config server. The instance starts on port 27019  Shardsvr = Start this instance as a shard. The instance starts in port 27018 by default. |  |
| archiveMovedChunks | boolean | When true, the sharding.archiveMovedChunks option forces the mongod instances to save all documents migrated from this shard during chunk migrations to the moveChunk directory of the storage.dbPath. MongoDB does not delete data stored in moveChunk. |  |

**mongos-only Options**

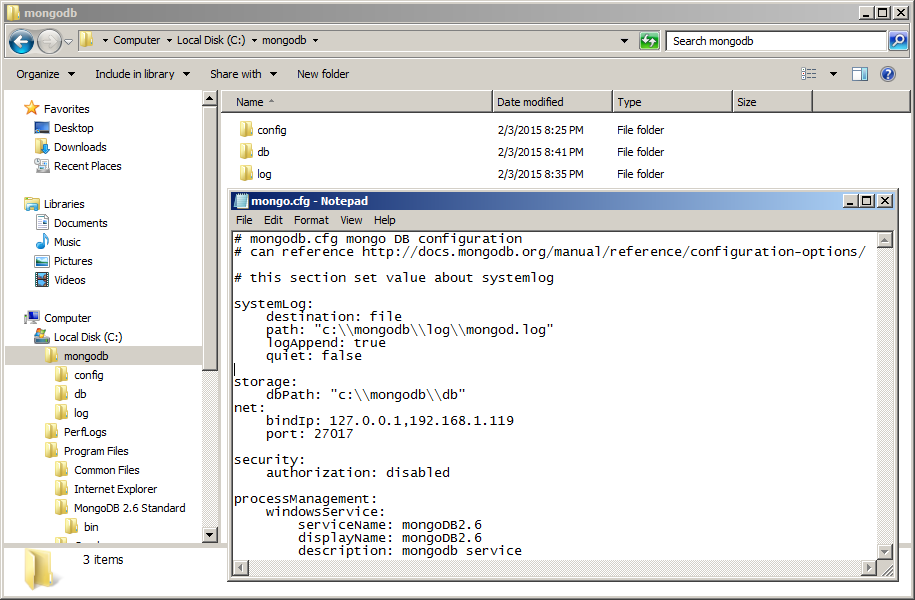
| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| localPingThresholdMs | integer | Affects the logic that mongos uses when selecting replica set members to pass read operations from clients. Specify a value in milliseconds. The default value of 15 corresponds to the default value in all of the client drivers.  When mongos receives a request that permits reads to secondary members, the mongos will:  • Find the member of the set with the lowest ping time.  • Construct a list of replica set members that is within a ping time of 15 milliseconds of the nearest suitable member of the set.  If you specify a value for the replication.localPingThresholdMs option, mongos will construct the list of replica members that are within the latency allowed by this value.  • Select a member to read from at random from this list.  The ping time used for a member compared by the replication.localPingThresholdMs setting is a moving average of recent ping times, calculated at most every 10 seconds. As a result, some queries may reach members above the threshold until the mongos recalculates the average | Default = 15 |
| autoSplit | boolean | Prevents mongos from automatically inserting metadata splits in a sharded collection. If set on all mongos instances, this prevents MongoDB from creating new chunks as the data in a collection grows.  Because any mongos in a cluster can create a split, to totally disable splitting in a cluster you must set sharding.autoSplit on all mongos. |  |
| configDB | string | Specifies the [configuration database](http://docs.mongodb.org/manual/reference/glossary/#term-config-database) for the [sharded cluster](http://docs.mongodb.org/manual/reference/glossary/#term-sharded-cluster). You must specify either 1 or 3 configuration servers, in a comma separated list. **Always** use 3 config servers in production environments.  All [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) instances **must** specify the exact same value for sharding.configDB  If your configuration databases reside in more that one data center, order the hosts so that first config sever in the list is the closest to the majority of your [mongos](http://docs.mongodb.org/manual/reference/program/mongos/#bin.mongos) instances. |  |
| chunkSize | integer | Determines the size in megabytes of each chunk in the sharded cluster. A size of 64 megabytes is ideal in most deployments: larger chunk size can lead to uneven data distribution; smaller chunk size can lead to inefficient movement of chunks between nodes.  This option affects chunk size only when you initialize the cluster for the first time. If you later modify the option, the new value has no effect. See the Modify Chunk Size in a Sharded Cluster procedure if you need to change the chunk size on an existing sharded cluster. | Default=64 |

**Windows Service Options**

**processManagement**

| Setting | Type | Description | Remark |
| --- | --- | --- | --- |
| windowsService.serviceName | string | Set the service name of mongos or mongod when running as a Windows Service. Use this name with the net start <name> and net stop <name> operations.  You must use processManagement.windowsService.serviceName in conjunction with either the --install or --remove install option. | Default: MongoDB |
| windowsService.displayName | string | Sets the name listed for MongoDB on the Services administrative application. | Default: MongoDB |
| windowsService.description | string | You must use processManagement.windowsService.description in conjunction with the --install option. | Default: MongoDB Server |
| windowsService.serviceUser | string | Runs the mongos or mongod service in the context of a certain user. This user must have “Log on as a service” privileges.  You must use processManagement.windowsService.serviceUser in conjunction with the --install option |  |
| windowsService.servicePassword | string | Sets the password for <user> for mongos or mongod when running with the --serviceUser option.  You must use processManagement.windowsService.servicePassword in conjunction with the --install option. |  |

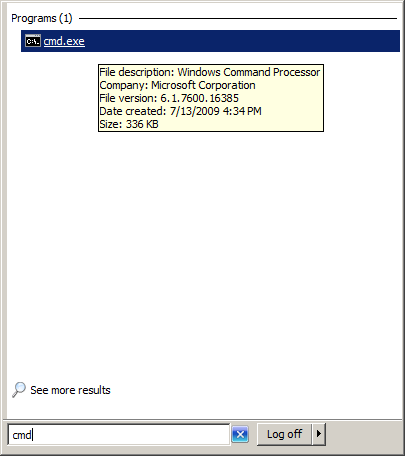
# Install MongoDB Services Single Machine



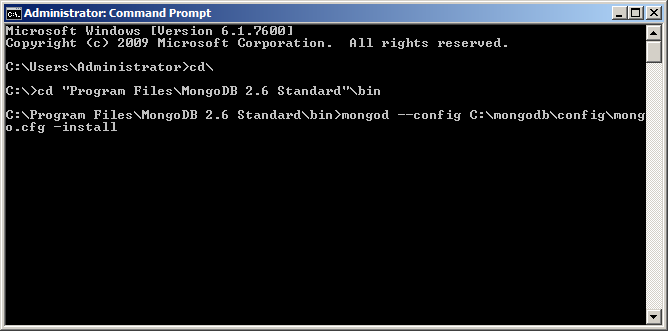
Figure

After installed MongoDB we will create the folder’s structure for contains the data of MongoDB. For the example we can create folder like this

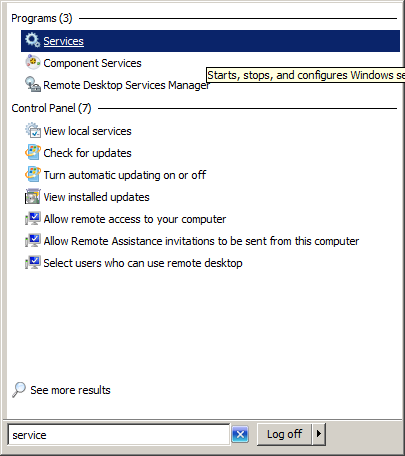
* c:\mongodb\config



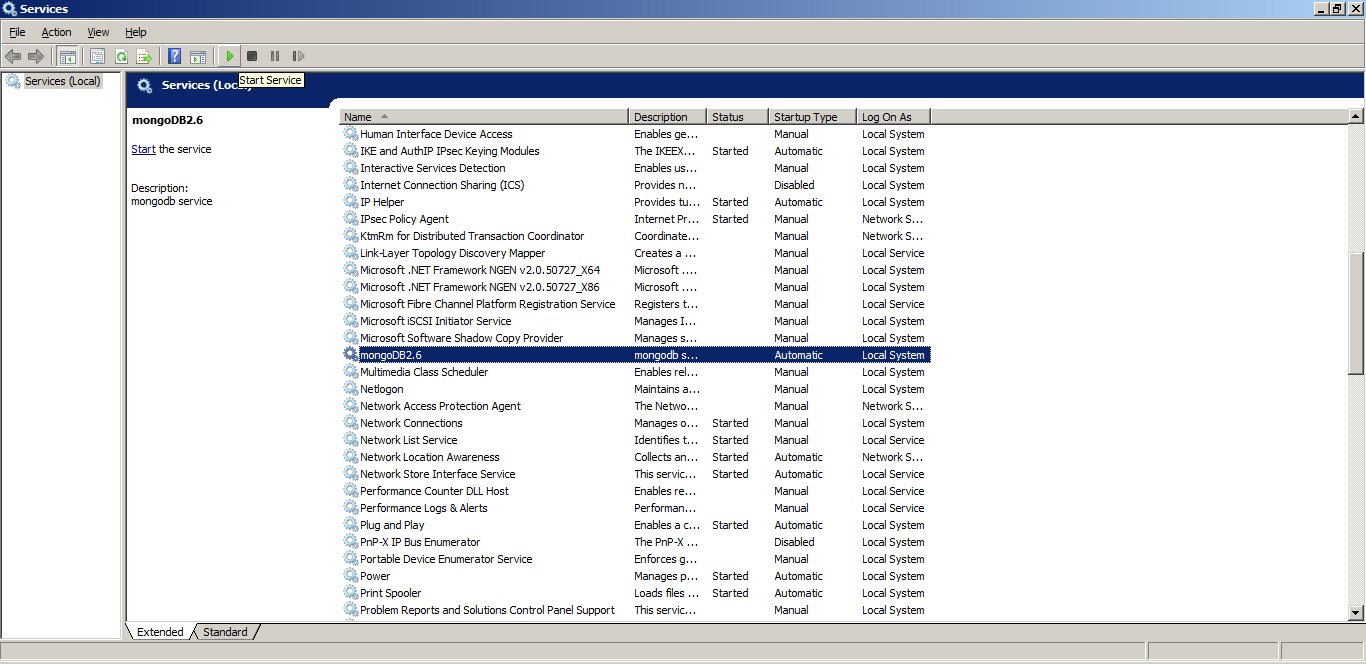
Figure



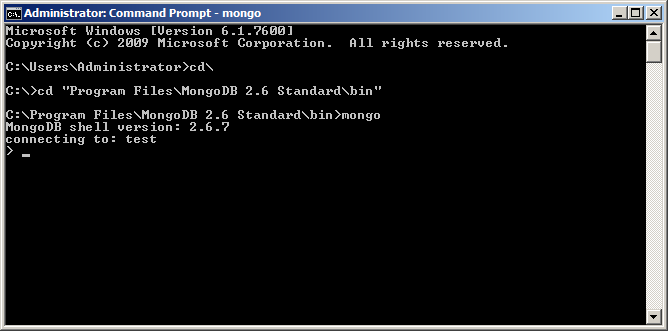
Figure



Figure



Figure



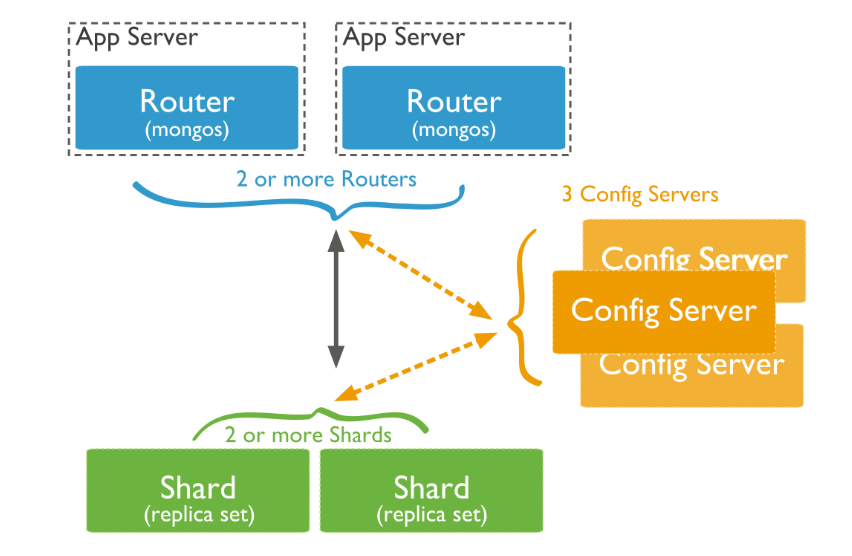
Figure

# Sharding

Sharding is a method for storing data across multiple machines, MongoDB uses sharding to support deployments with very large data sets, support scalability.

## Overview

MongoDB supports sharding through the configuration of a sharded clusters. Sharded c;uster has the following components: shards query routers and the config server.

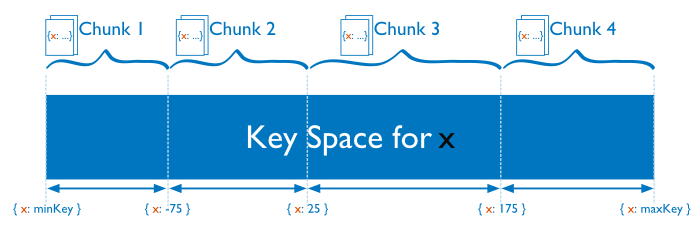


Figure

**Shard** store the data, to provide high availability and data consistency, in a production shard cluster, each shard is a replica set. For more information on replica sets, set next replica chapter

**Query Routers**, or mongso instances, interface with client applications and direct operations to the appropriate shard or shards. The query router processes and targets operations to shards and then returns results to the clients. A sharded cluster can contain more than one query router to divide the client request load. A client sends requests to one query router. Most sharded clusters have many query routers

**Config servers,** store the cluster’s metadata. This data contains a mapping of the cluster’s data set to the shards. The query router uses this metadata to target operations to specific shards. Production sharded clusters have exactly 3 config servers.



Figure

## Setting up step

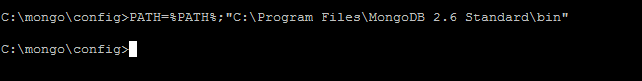
In this document we have 1 **Query Routers Server**, 3 **Config Server** and 2 **Shard Server** ….

Picture of architecture

We will create Windows Service of mongo db, before we can run windows service we have to run mongo command to install windows service.

Before you can run mongo command, you have to add mongo binary path to system path,

c:\>PATH=%PATH%;”C:\Program Files\MongoDB 2.6 Standard\bin”



or go to mongo binary path to run mongo command in this document.

C:\>cd “C:\Program Files\MongoDB 2.6 Standard\bin”

1. Run command with all parameters, such as

mongod.exe --install --dbpath c:\mongo\config\ --port 20000 --configsvr --logpath c:\mongo\log\configlog.txt --bind\_ip 192.168.1.64

\*\*\*In this documents we donot use this way\*\*\*

2. Run command to read parameter from the configuration file, such as

mongod.exe --install --config c:\mongo\config\router.cfg

\*\*\*In this documents we use this way because it flexible modify later\*\*\*

**Most important all mongo server should same time zone for all server, we suggest sync them to same NTP server.**

### Setting Config Server

The first step of building mongodb with cluster server, we have to create configuration server first. For example for configuration like table below.

# mongo DB configuration

# can reference http://docs.mongodb.org/manual/reference/configuration-options/

# systemlog section, this will be allow to config value relate logging

systemLog:

destination: file

path: "C:\\mongo\\log\\mongoconfig1.log"

logAppend: true

quiet: false

traceAllExceptions: true

timeStampFormat: iso8601-utc

storage:

dbPath: "C:\\mongo\\db\\config1"

directoryPerDB: true

net:

bindIp: 192.168.1.64

port: 28000

# security

security:

authorization: disabled

# Sharding Configuration.

sharding:

clusterRole: configsvr

# windows service section

processManagement:

windowsService:

serviceName: MongoConfig01

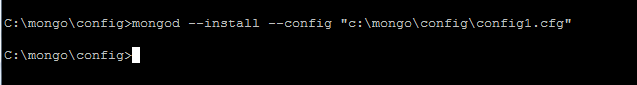
displayName: MongoConfig01

description: MongoConfig Service 01

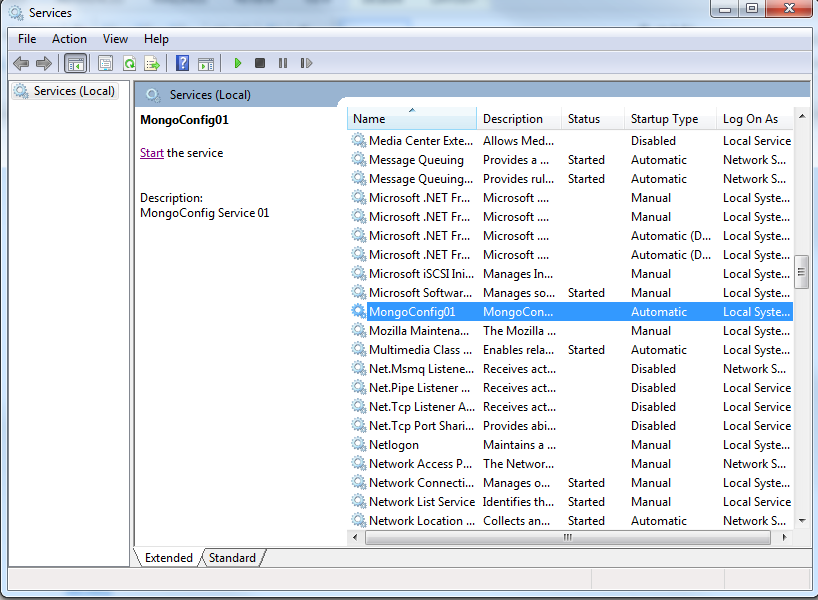
When we got the configuration file, in this case the configuration files save as “config01.cfg”, we can command to install windows service

mongod.exe --install --config config01.cfg

The output should show like picture below.

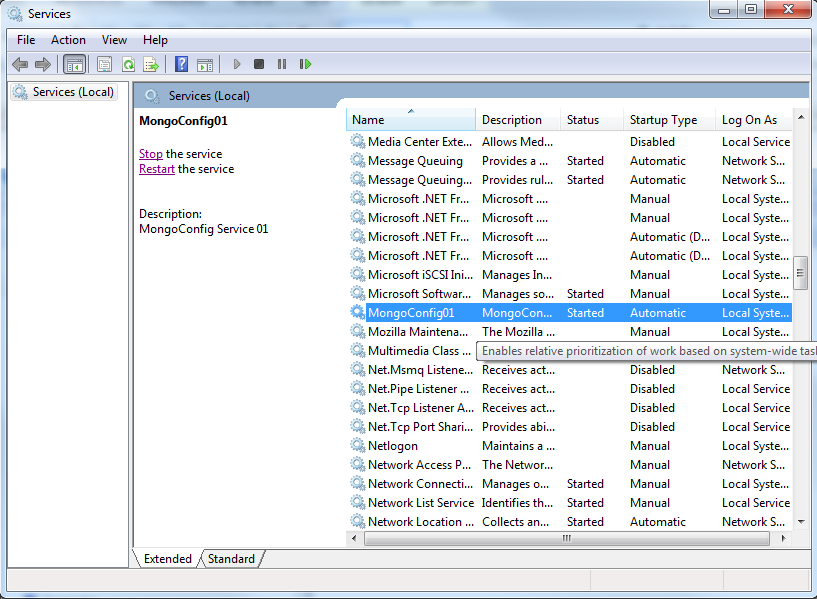


Then go to service explorer by click on start menu -> Administrator Tools -> Services. Find the Service name that was create in previous step, it was in configuration file in this example is “MongoConfig01



Figure

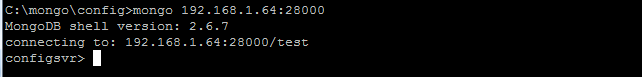
Click start to start MongoConfig01 Service.



Figure

After we start mongo configuration service, before going to next step we have to test connect to mongo configuration server be sure it working.

mongo 192.168.1.64:28000



Repeat install mongo configuration more 2 times.

### Setting Router Server

After we config the mongo configuration server, we have to create new mongo router by create config file for router like table below

# router.cfg mongo DB configuration for router

# can reference http://docs.mongodb.org/manual/reference/configuration-options/

# this section set value about systemlog

systemLog:

destination: file

path: "c:\\mongo\\log\\routerLog.log"

logAppend: true

quiet: false

net:

bindIp: 192.168.1.64

port: 30000

sharding:

configDB: "192.168.1.19:28000,192.168.1.204:28000,192.168.1.83:28000"

processManagement:

windowsService:

serviceName: "MongoRouter"

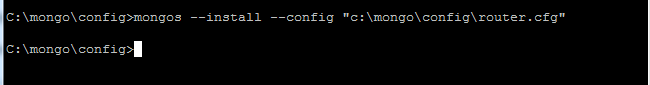
displayName: "MongoRouter"

description: "MougoRouter Service"

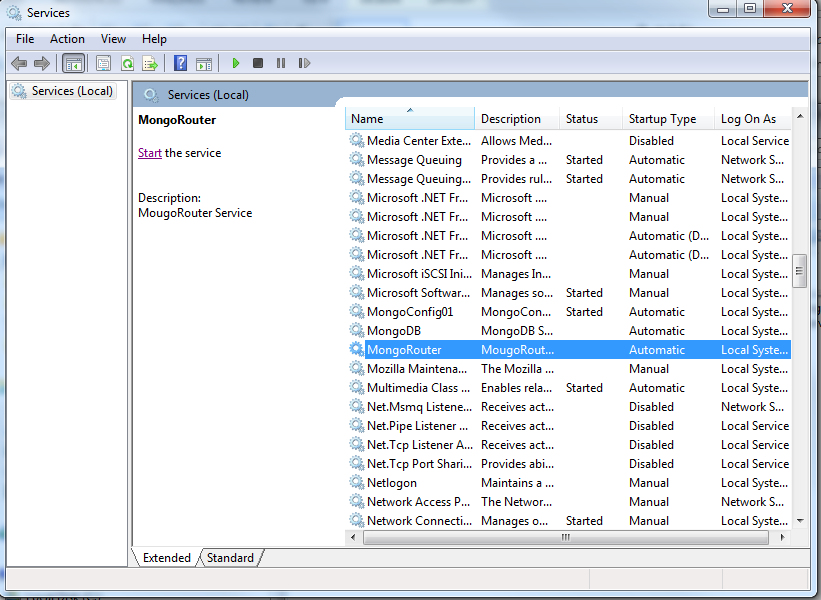
When we got the configuration file, in this case the configuration files save as “router.cfg”, we can command to install windows service

mongos.exe --install --config router.cfg

The output should show like picture below.

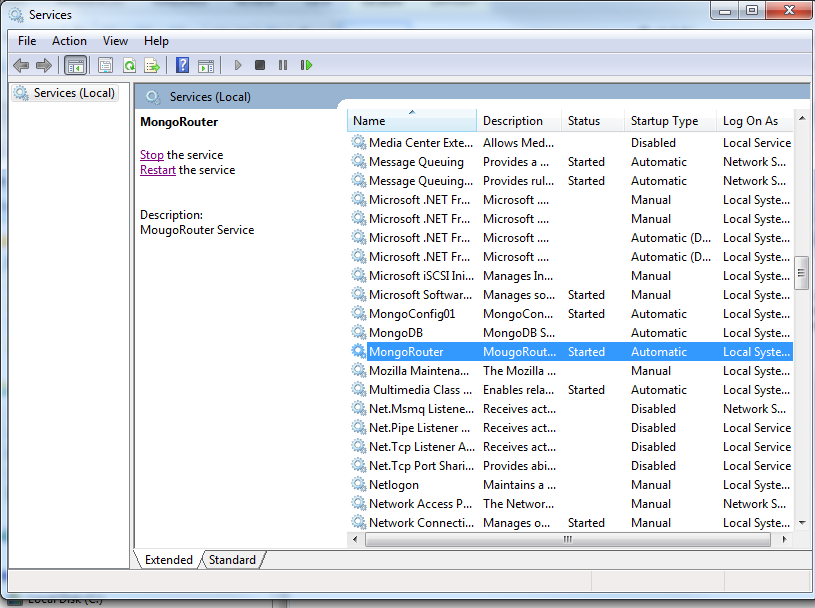


Then go to service explorer by click on start menu -> Administrator Tools -> Services. Find the Service name that was create in previous step, it was in configuration file in this example is “MongoRouter”



Figure

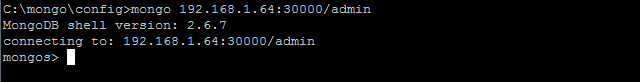
Click start on MongoRouter to start mongo router



Figure

After we start mongo configuration service, before going to next step we have to test connect to mongo Router server be sure it working.

mongo 192.168.1.64:30000/admin



### Initial Shard Server

Before we can add Shard to our mongo cluster server, we need to create mongo server.

#### Setting ShardServer

We have to create configuration file like table below.

# mongodb.cfg mongo DB configuration

# can reference http://docs.mongodb.org/manual/reference/configuration-options/

# this section set value about systemlog

systemLog:

destination: file

path: "c:\\mongo\\log\\mongod.log"

logAppend: true

quiet: false

storage:

dbPath: "c:\\mongo\\db"

net:

bindIp: 127.0.0.1,192.168.1.44

port: 27017

security:

authorization: enabled

sharding:

clusterRole: shardsvr

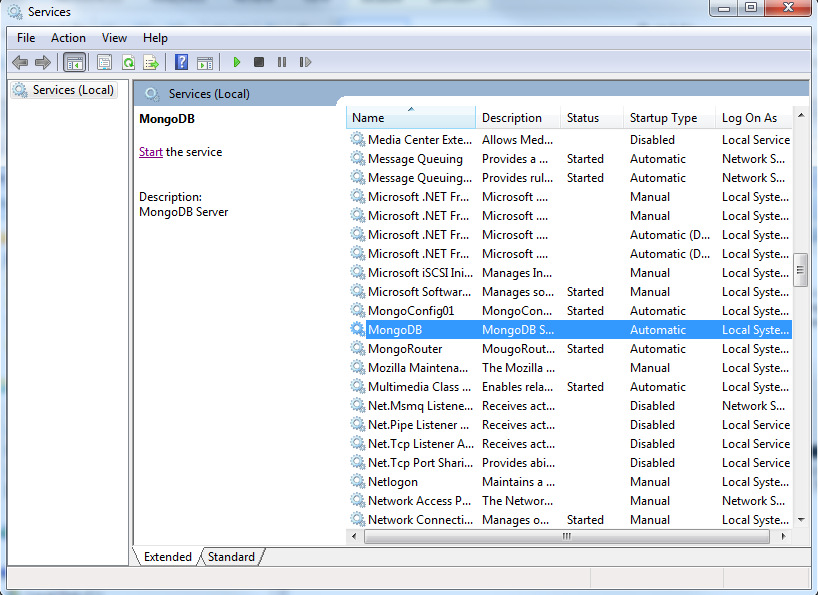
When we finish create mongo server, we have to install it as windows service by command below

mongod.exe --install –config mongo.cfg

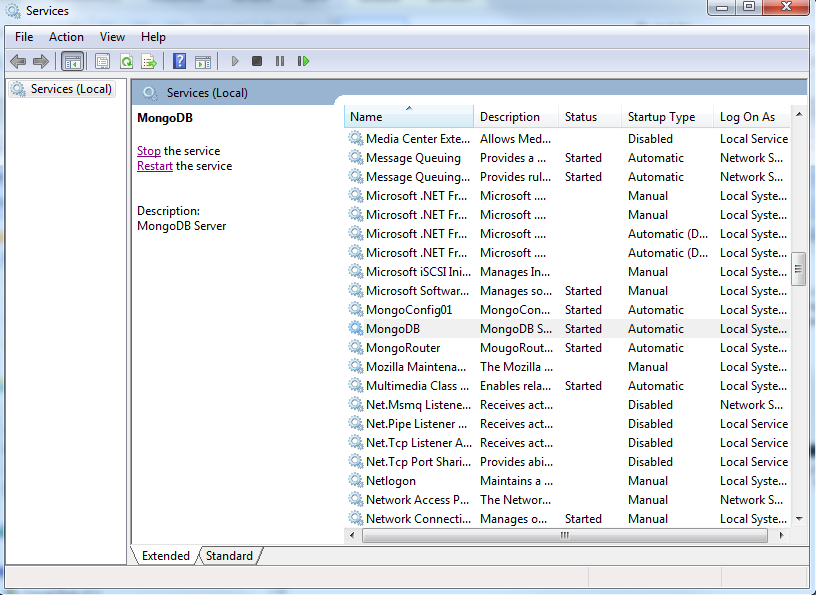
C:\Users\Jorn\Desktop\mongo sharding\install_mongo_service.png

Then we go to service explorer by click on start menu -> Administrator Tools -> Services

Click on start on MongoDB to start mongo db service



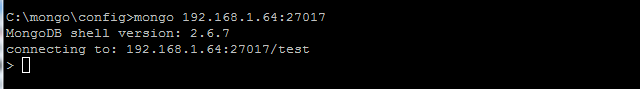
Figure



Figure

After we start mongo configuration service, before going to next step we have to test connect to mongo Router server be sure it working.

mongo 192.168.1.64:27017



### Adding mongodb as shard to the router

The mongo command can run only on mongo shell, we have to connect to mongo router and run those command on mongo shell, the command to connect mongo shell like table below

C:\mongo>mongo 192.168.1.64:30000/admin

MongoDB shell version: 2.6.7

connecting to: 192.168.1.64:30000/admin

mongos>

#### Add sharding

mongos>db.runCommand({addshard : "192.168.1.166:28000" })

*{ "shardAdded" : "shard0000", "ok" : 1 }*

#### Check sharding

mongos>sh.status()

*--- Sharding Status ---*

*sharding version: {*

*"\_id" : 1,*

*"version" : 4,*

*"minCompatibleVersion" : 4,*

*"currentVersion" : 5,*

*"clusterId" : ObjectId("54d205008c3761f1a787ef4f")*

*}*

*shards:*

*{ "\_id" : "shard0000", "host" : "192.168.1.166:28000" }*

*databases:*

*{ "\_id" : "admin", "partitioned" : false, "primary" : "config" }*

#### Enable Sharding on database

mongos> db.runCommand( {enablesharding: "psndatainfo"} )

{ "ok" : 1 }

#### Enable Sharding on Collection

mongos> db.runCommand( {shardCollection: "psndatainfo.metadata", key: {SessionId:1} })

{ "collectionsharded" : "psndatainfo.metadata", "ok" : 1 }

Finally we finish set sharing on mongo router for checking status by command below

mongos>sh.status()

The result should be like

sh.status()

--- Sharding Status ---

sharding version: {

"\_id" : 1,

"version" : 4,

"minCompatibleVersion" : 4,

"currentVersion" : 5,

"clusterId" : ObjectId("54d309e9ffd9495038de9daa")

}

shards:

{ "\_id" : "shard0000", "host" : "192.168.1.166:28000" }

{ "\_id" : "shard0001", "host" : "192.168.1.96:28000" }

databases:

{ "\_id" : "admin", "partitioned" : false, "primary" : "config" }

{ "\_id" : "psndatainfo", "partitioned" : true, "primary" : "shard0000" }

psndatainfo.metadata

shard key: { "SessionId" : 1 }

chunks:

shard0001 2

shard0000 1

{ "SessionId" : { "$minKey" : 1 } } -->> { "SessionId" : "327968337" } on : shard0001 Timestamp(2, 0)

{ "SessionId" : "327968337" } -->> { "SessionId" : "999625835" } on : shard0000 Timestamp(3, 1)

{ "SessionId" : "999625835" } -->> { "SessionId" : { "$maxKey" : 1 } } on : shard0001 Timestamp(3, 0)

## Scalability

After we finish create mongo cluster, we can add shard later by connect to mongo router and run command on mongo shell

### Connect to mongo router

C:\>mongo.exe 192.168.1.64:30000/admin

### Add shard

mongos>db.runCommand({addshard : "192.168.1.166:28000" })

*{ "shardAdded" : "shard0000", "ok" : 1 }*

## --run command router server

# Replication

## Introduction

Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions. With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup.

In some cases, you can use replication to increase read capacity. Clients have the ability to send read and write operations to different servers. You can also maintain copies in different data centers to increase the locality and availability of data for distributed applications.

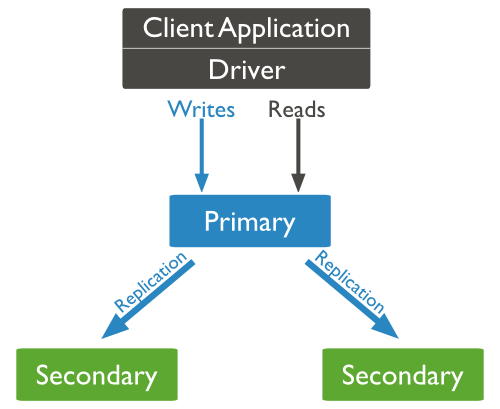
**Why Replication?**

* To keep your data safe
* High (24\*7) availability of data
* Disaster Recovery
* No downtime for maintenance (like backups, index rebuilds, compaction)
* Read scaling (extra copies to read from)
* Replica set is transparent to the application

### Replication in MongoDB

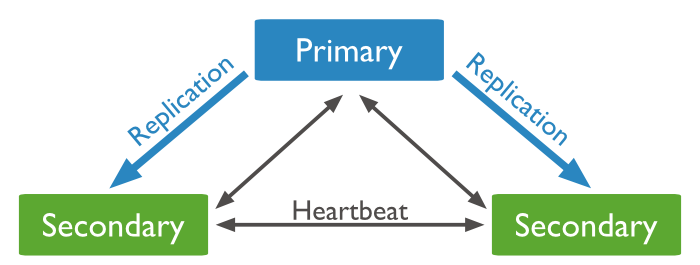
A replica set is a group of mongod instances that host the same data set. One mongod, the primary, receives all write operations. All other instances, secondaries, apply operations from the primary so that they have the same data set.

The **Primary** accepts all write operations from clients. Replica set can have only one **Primary**. Because only one member can accept write operations, replica sets provide strict consistency for all reads from the **Primary**. To support replication, the **Primary** logs all changes to its data sets in its oplog. See **Primary** for more information.



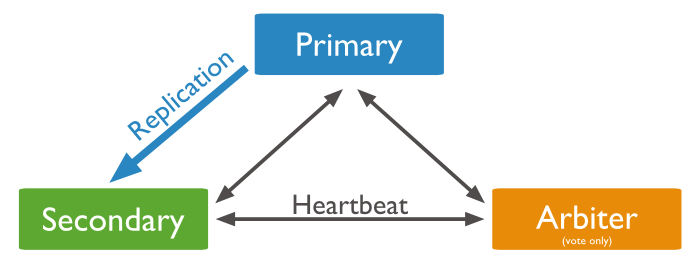
Figure

The **Secondaries** replicate the **Primary**’s oplog and apply the operations to their data sets. **Secondaries**’ data sets reflect the **Primary**’s data set. If the primary is unavailable, the replica set will elect a **Secondary** to be **Primary**. By default, clients read from the **Primary**, however, clients can specify a read preferences to send read operations to **Secondaries**. Reads from **Secondaries** may return data that does not reflect the state of the **Primary**. See **Secondaries** for more information.



Figure

You may add an extra mongod instance to a replica set as an arbiter. **Arbiters** do not maintain a data set. **Arbiters** only exist to vote in elections. If your replica set has an even number of members, add an **Arbiter** to obtain a majority of votes in an election for **Primary**. **Arbiters** do not require dedicated hardware. See **Arbiter** for more information.



Figure

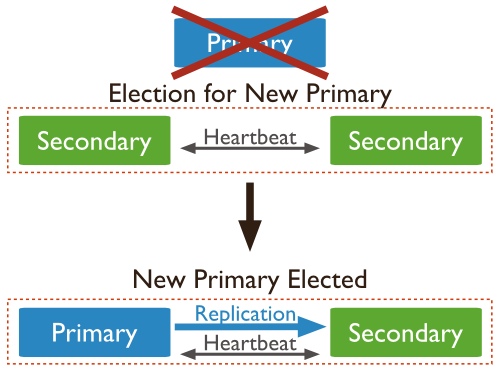
An **Arbiter** will always be an **Arbiter**. A **Primary** may step down and become a **Secondary**. A **Secondary** may become the **Primary** during an election.

#### Asynchronous Replication

**Secondaries** apply operations from the **Primary** asynchronously. By applying operations after the **Primary**, sets can continue to function without some members. However, as a result **Secondaries** may not return the most current data to clients.

#### Automatic Failover

When a **Primary** does not communicate with the other members of the set for more than 10 seconds, the replica set will attempt to select another member to become the new **Primary**. The first secondary that receives a majority of the votes becomes **Primary**.



Figure

#### Additional Features

Replica sets provide a number of options to support application needs. For example, you may deploy a replica set with members in multiple data centers, or control the outcome of elections by adjusting the priority of some members. Replica sets also support dedicated members for reporting, disaster recovery, or backup functions.

Table below show the ability to access the MongoDB when the servers crashes.

|  |  |  |  |
| --- | --- | --- | --- |
| Shard (Primary) | Secondary | Arbiter | Insert to Mongo |
| Y | Y | Y | Y |
| Y | X | Y | Y |
| X | Y | Y | Y |
| Y | Y | X | Y |
| Y | X | X | X |
| X | Y | X | X |
| X | X | X | X |

\* Y is online, X is offline

## Replica Set Installation

This procedure creates the initial three-member replica set rs1. The replica set members can be create following these step.

### Prepare Configuration

The step for create the initial three-member replica set is to create the configuration file. For more information about how to configure configuration file will refer to section 5.1 Core Options.

# mongo DB configuration

# can reference http://docs.mongodb.org/manual/reference/configuration-options/

# systemlog section, this will be allow to config value relate logging

systemLog:

destination: file

path: "C:\\mongo\\log\\mongoconfig1.log"

logAppend: true

quiet: false

traceAllExceptions: true

timeStampFormat: iso8601-utc

storage:

dbPath: "C:\\mongo\\db\\config1"

directoryPerDB: true

#for each member, we have to set to its own IP and Port

net:

bindIp: 192.168.1.19

port: 28000

# security

security:

authorization: disabled

# windows service section

processManagement:

windowsService:

serviceName: MongoReplica01

displayName: MongoReplica01

description: MongoReplica01 Service

From the configuration file example above, we have to configure one file per one member.

### Install replica services

This step we have to install each member with “replSet” options by using this command.

>mongod –config “C:\\mongo\\config\\mongoconfig1.conf” –-replSet “rs1” --install

After we have installed all three-member with replica set name “rs1”. We have to start all service of member and prepare which one will be **Primary** and **Secondary** and **Arbiter**.

### Add member of replica set

This step we have to use **Primary** to add **Secondary** and **Arbiter** by connect to **Primary** with this command.

>mongo <PrimaryIP:PrimaryPort>

### Initial Replica Set

>rs.initiate()

### Add Secondary

>rs.add(“SecondaryIP:SecondaryPort”)

### Add Arbiter

>rs.addArb(“ArbiterIP:ArbiterPort”)

### Check Replication Status

>rs.status()

# Sharding with Replication

## Prepare Primary+Shard Server

1. Prepare configuration command for primary machine

systemLog:

destination: file

path: "C:\\Program Files\\MongoDB 2.6 Standard\\log\\mongo.log"

logAppend: true

quiet: false

traceAllExceptions: true

timeStampFormat: iso8601-utc

storage:

dbPath: "C:\\Program Files\\MongoDB 2.6 Standard\\data\\db"

directoryPerDB: true

net:

bindIp: 127.0.0.1,192.168.1.187

port: 28000

security:

authorization: disabled

sharding:

clusterRole: shardsvr

1. Run command to install primary on command prompt

Mongod –config <config path> --relpSet <Replication Name> --install

1. Start service.
2. Connect to mongo via command

Mongo <host:port> (username passwd)

1. Run command

rs.initiate()

## Prepare secondary server

1. Prepare configuration command for secondary server

systemLog:

destination: file

path: "C:\\Program Files\\MongoDB 2.6 Standard\\log\\mongo.log"

logAppend: true

quiet: false

traceAllExceptions: true

timeStampFormat: iso8601-utc

storage:

dbPath: "C:\\Program Files\\MongoDB 2.6 Standard\\data\\db"

directoryPerDB: true

net:

bindIp: 127.0.0.1,192.168.1.188

port: 29000

security:

authorization: disabled

1. Run command to install secondary server on command prompt

Mongod –config <config path> --relpSet <Replication Name> --install

1. Start service.

## Prepare Arbiter Server

1. Prepare configuration command for secondary server

systemLog:

destination: file

path: "C:\\Program Files\\MongoDB 2.6 Standard\\log\\mongo.log"

logAppend: true

quiet: false

traceAllExceptions: true

timeStampFormat: iso8601-utc

storage:

dbPath: "C:\\Program Files\\MongoDB 2.6 Standard\\data\\db"

directoryPerDB: true

net:

bindIp: 127.0.0.1,192.168.1.189

port: 29000

security:

authorization: disabled

1. Run command to install secondary server on command prompt

Mongod –config <config path> --relpSet <Replication Name> --install

1. Start service.

## Add Secondary to Primary Server

1. Connect to mongo of primary server via command

Mongo <host:port> (username passwd)

1. Add secondary server to primary via command

rs.add(“<secondaryIP:secondaryPort>”)

## Add Arbiter to Primary Server

1. Connect to mongo of primary server via command

Mongo <host:port> (username passwd)

1. Add secondary server to primary via command

rs.addArb(“<ArbiterIP:ArbiterPort>”)

# Enable Authentication in Shared Cluster

When authentication is enabled on a sharded cluster every client that accesses the cluster must provide credentials. This includes MongoDB instances that access each other within the cluster.

To enable authentication on a sharded cluster, you must enable authentication individually on each component of the cluster. This means enabling authentication on each mongos and each mongod, including each config server, and all members of a shard’s replica set.

Authentication requires an authentication mechanism and, in most cases, a key file. The content of the key file must be the same on all cluster members.

This section will provides step for create authentication of mongoDB by using a key file.

***Requirement***

1. User with role userAdmin or userAdminAnyDatabase. (in case we did not have any user, we must create user admin by log in to mongoDB and use this command.

db.createUser(

{

user: “admin”,

pwd: “password”

roles:

[

{

role: “userAdminAnyDatabase”,

db: “admin”

}

]

}

)

1. Key file that generate from openssl by use this command. (refer to document: )

openssl rand -base64 741 > mongodb-keyfile

## Enable authentication on each component in the cluster

After we have user administrator and key file we have to copy key file to ever cluster in mongoDB system include router server, config server, shard, replicate and arbiter, then set configuration file for each mongo process to the directory that store key file as a table below.

security:

keyFile: C:\\mongo\keyfile\\mongodb-keyfile

Restart mongoDB Service.

## Create user

1. \*\*\*Connect to mongoDB by using user administrator.
2. Run command for create user as table below. (for user roles, look at section 9.2.1 Built-in Roles)

## 

>use psndatainfo

>db.createUser(

{

user: “psnadmin”,

pwd: “password”,

roles:

[

{

role: “dbOwner”,

db: “psndatainfostore”

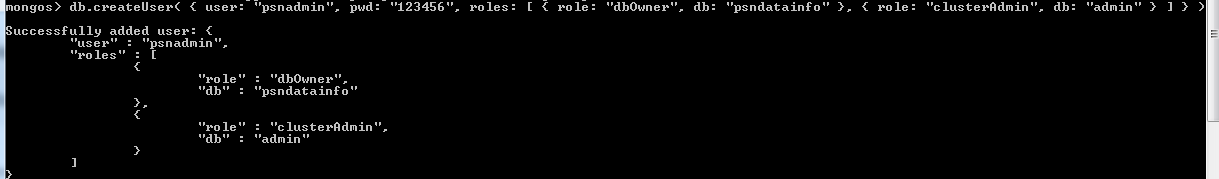
}

]

}

)

1. The result will show as figure below.



\*\*\* In case that we want to use the user that we create for connect to the database directly, we have to select the destination database after we connected to admin database and use “use <destination db>” first, then create user.

### Built-in Roles

MongoDB grants access to data and commands through role-based authorization and provides built-in roles that provide the different levels of access commonly needed in a database system. You can additionally create user-defined roles.

A role grants privileges to perform sets of actions on defined resources. A given role applies to the database on which it is defined and can grant access down to a collection level of granularity.

Each of MongoDB’s built-in roles defines access at the database level for all non-system collections in the role’s database and at the collection level for all system collections.

MongoDB provides the built-in database user and database administration roles on every database. MongoDB provides all other built-in roles only on the admin database.

This section describes the privileges for each built-in role. You can also view the privileges for a built-in role at any time by issuing the rolesInfo command with the showPrivileges and showBuiltinRoles fields both set to true.

#### Database User Roles

Every database includes the following client roles:

| Roles | Description | actions |
| --- | --- | --- |
| read | Provides the ability to read data on all non-system collections and on the following system collections:system.indexes, system.js, and system.namespaces collections. The role provides read access by granting the following actions. | collStats,  dbHash,  dbStats,  find,  killCursors |
| readWrite | Provides all the privileges of the read role plus ability to modify data on all non-system collections and the system.js collection. The role provides the following actions on those collections. | collStats,  convertToCapped,  createCollection,  dbHash,  dbStats,  dropCollection,  createIndex,  dropIndex,  emptycapped,  find,  insert,  killCursors,  remove,  renameCollectionSameDB,  update |

#### Database Administration Roles

Every database includes the following database administration roles.

| Roles | Description | Actions |
| --- | --- | --- |
| dbAdmin | Provides the following actions on the database’s system.indexes, system.namespaces, and system.profile collections. | collStats,  dbHash,  dbStats,  find,  killCursors,  dropCollection and createCollection on system.profile only |
| Provides the following actions on all non-system collections. This role\*does not\* include full read access on non-system collections | collMod,  collStats,  compact,  convertToCapped,  createCollection,  createIndex,  dbStats,  dropCollection,  dropDatabase,  dropIndex,  enableProfiler,  indexStats,  reIndex,  renameCollectionSameDB,  repairDatabase,  storageDetails,  validate |
| dbOwner | The database owner can perform any administrative action on the database. This role combines the privileges granted by the readWrite, dbAdmin and userAdmin roles. |  |
| userAdmin | Provides the ability to create and modify roles and users on the current database. This role also indirectly provides superuser access to either the database or, if scoped to the admin database, the cluster. The userAdmin role allows users to grant any user any privilege, including themselves.  The userAdmin role explicitly provides the following actions. | changeCustomData,  changePassword,  createRole,  createUser,  dropRole,  dropUser,  grantRole,  revokeRole,  viewRole,  viewUser |

#### Cluster Administration Roles

The admin database includes the following roles for administering the whole system rather than just a single database. These roles include but are not limited to replica set and sharded cluster administrative functions.

| Roles | Description | Actions |
| --- | --- | --- |
| clusterAdmin | Provides the greatest cluster-management access. This role combines the privileges granted by the clusterManager, clusterMonitor, and hostManager roles. | Additionally, the role provides the dropDatabase action. |
| clusterManager | Provides management and monitoring actions on the cluster. A user with this role can access the config and local databases, which are used in sharding and replication, respectively.  Provides the following actions on the cluster as a whole. | addShard,  applicationMessage,  cleanupOrphaned,  flushRouterConfig,  listShards,  removeShard,  replSetConfigure,  replSetGetStatus,  replSetStateChange,  resync |
| Provides the following actions on all databases in the cluster. | enableSharding,  moveChunk,  splitChunk,  splitVector |
| On the config database, provides the following actions on all configuration collections and on the system.indexes, system.js, and system.namespaces collections: | collStats,  dbHash,  dbStats,  find,  killCursors |
| On the local database, provides the following actions on the replset collection: | collStats,  dbHash,  dbStats,  find,  killCursors |
| clusterMonitor | Provides read-only access to monitoring tools, such as the MongoDB Management Service (MMS) monitoring agent.  Provides the following actions on the cluster as a whole: | connPoolStats,  cursorInfo,  getCmdLineOpts,  getLog,  getParameter,  getShardMap,  hostInfo,  inprog,  listDatabases,  listShards,  netstat,  replSetGetStatus,  serverStatus,  shardingState,  top |
|  | Provides the following actions on all databases in the cluster. | collStats,  dbStats,  getShardVersion |
|  | Provides the find action on all system.profile collections in the cluster. |  |
|  | Provides the following actions on the config database’s configuration collections and system.indexes, system.js, and system.namespaces collections. | collStats,  dbHash,  dbStats,  find,  killCursors |
| hostManager | Provides the ability to monitor and manage servers.  Provides the following actions on the cluster as a whole. | applicationMessage,  closeAllDatabases,  connPoolSync,  cpuProfiler,  diagLogging,  flushRouterConfig,  fsync,  invalidateUserCache,  killop,  logRotate,  resync,  setParameter,  shutdown,  touch,  unlock |
| Provides the following actions on all databases in the cluster. | killCursors,  repairDatabase |

#### Backup and Restoration Roles

The admin database includes the following roles for backing up and restoring data.

| Roles | Description | Actions |
| --- | --- | --- |
| backup | Provides minimal privileges needed for backing up data. This role provides sufficient privileges to use the MongoDB Management Service (MMS) backup agent, or to use mongodump to back up an entire mongod instance. |  |
| Provides the following actions on the mms.backup collection in the admin database: | Insert,  update |
| Provides the listDatabases action on the cluster as a whole. |  |
| Provides the find action on the following:  all non-system collections in the cluster  all the following system collections in the cluster: system.indexes, system.namespaces, and system.js  the admin.system.users and admin.system.roles collections  legacy system.users collections from versions of MongoDB prior to 2.6 |  |
| restore | Provides minimal privileges needed for restoring data from backups. This role provides sufficient privileges to use the mongorestore tool to restore an entire mongod instance. |  |
| Provides the following actions on all non-system collections and system.js collections in the cluster; on the admin.system.users and admin.system.roles collections in the admin database; and on legacy system.users collections from versions of MongoDB prior to 2.6 | collMod,  createCollection,  createIndex,  dropCollection,  insert |
| Provides the following additional actions on admin.system.users and legacy system.users collections. | find,  remove,  update |
| Provides the find action on all the system.namespaces collections in the cluster.  Although, restore includes the ability to modify the documents in the admin.system.users collection using normal modification operations, only modify these data using the user management methods. |  |

#### All-Database Roles

The admin database provides the following roles that apply to all databases in a mongod instance and are roughly equivalent to their single-database equivalents

| Roles | Description | Actions |
| --- | --- | --- |
| readAnyDatabase | Provides the same read-only permissions as read, except it applies to all databases in the cluster. The role also provides the listDatabases action on the cluster as a whole. |  |
| readWriteAnyDatabase | Provides the same read and write permissions as readWrite, except it applies to all databases in the cluster. The role also provides the listDatabases action on the cluster as a whole. |  |
| userAdminAnyDatabase | Provides the same access to user administration operations as userAdmin, except it applies to all databases in the cluster. The role also provides the following actions on the cluster as a whole. | authSchemaUpgrade,  invalidateUserCache,  listDatabases |
|  | The role also provides the following actions on the admin.system.users and admin.system.roles collections on the admin database, and on legacy system.users collections from versions of MongoDB prior to 2.6 | collStats,  dbHash,  dbStats,  find,  killCursors,  planCacheRead,  createIndex,  dropIndex |
| dbAdminAnyDatabase | Provides the same access to database administration operations as dbAdmin, except it applies to all databases in the cluster. The role also provides the listDatabases action on the cluster as a whole. |  |

#### Superuser Roles

| Roles | Description |
| --- | --- |
| root | Provides access to the operations and all the resources of the readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabase and clusterAdmin roles combined.  root does not include any access to collections that begin with the system. prefix.  For example, without the ability to insert data directly into the:data:system.users <admin.system.users> and system.roles collections in the admin database. root is not suitable for writing or restoring data that have these collections (e.g. with mongorestore.) To perform these kinds of restore operations, provision users with the restore role. |

#### Internal Role

| Roles | Description |
| --- | --- |
| \_\_system | MongoDB assigns this role to user objects that represent cluster members, such as replica set members and mongos instances. The role entitles its holder to take any action against any object in the database.  Do not assign this role to user objects representing applications or human administrators, other than in exceptional circumstances.  If you need access to all actions on all resources, for example to run the eval or applyOps commands, do not assign this role. Instead, create a user-defined role that grants anyAction on anyResource and ensure that only the users who needs access to these operations has this access. |

Mongo shell command reference: <http://docs.mongodb.org/manual/reference/method/>

# How to Build MongoDB with SSL

The mongo distribution not come with ssl option, from MongoDB Official website they offer to build with ssl option from source code. So this section will show instruction how to do that.

The software that require for build MongoDB

|  |  |  |
| --- | --- | --- |
| Software | Description | Source |
| Python 2.7 | The scripting language that require for building script | https://www.python.org/ |
| Scons 2.3.4 | The building tool | http://www.scons.org/ |
| OpenSSL 1.01l | The SSL library and tools | https://slproweb.com/ products/Win32OpenSSL.html |
| Wix Toolset 3.9 R2 | The Windows Installation Tools (exe file) | http://wixtoolset.org/ |
| MS Visual Studio 2013 | Use it as Compiler | http://www.visualstudio.com/ |
| MongoDB source code | The source code of mongo | https://www.mongodb.org |

Before we can compile mongoDB with SSL. We have to install software tools in above table.

1. install Python 2.7, please check box allow add Python Path to the system otherwise you have to manual add later.

2. Install scons 2.3.4

3. Install openSSL 1.01l, for this document we install on C:\OpenSSL-Win64

4. Install Wix Toolset R3.9R2

5. Mongodb source code.

## BuildingStep

1. Unzip mongodb source code to folder (in this D:\mongo-src, then go inside that folder

2. Run command

scons --ssl --release --64 --extrapath=C:\OpenSSL-Win64

This will compile source code and create mongod.exe in the root folder of the mongodb source code folder (in this example mongo-src)

3. Compile and build component by follow command below

scons --ssl --release --64 --extrapath=C:\OpenSSL-Win64 <Alias Command>

|  |  |
| --- | --- |
| Alias Command | Description |
| all | Builds everything (core, tools, and all tests) |
| tools | Builds the server tools (mongpdump, mongostat, etc) |
| test | Builds the detest program. |
| lint | Runs the code linter (linter = the process of flagging suspicious language usesge) |
| msi | Builds the windows MSI Installer |
| core | Builds mongod, mongos, and the mongo shell. |
| smoke | Runs the “dbtest” test. |
| smokeCppUnitTests | Runs the C++ unit tests. |
| smokeJsCore | Runs (some of) the Javascript integration tests |

In this document we build core, tools and msi

scons --ssl --release --64 --extrapath=C:\OpenSSL-Win64 core

It will create mongod.exe, mongos.exe and mongo.exe on root folder on mongo source code

scons --ssl --release --64 --extrapath=C:\OpenSSL-Win64 tools

This will create mongoDB tools on root folder of mongo source

scons --ssl --release --64 --extrapath=C:\OpenSSL-Win64 msi

This will create msi file for windows installer on <mongo source>/win32/64/release/ssl/msi

## Requirement software for server

For new installation to the server, we need softwares and their dependency like table below.

|  |  |  |
| --- | --- | --- |
| Software | Description | Dependency software |
| MongoDB msi | The mongoDB installer | OpenSSL |
| OpenSSL 1.01l | The SSL library and tools | Microsift Visual C++ 2008 Redistributable |
| Microsift Visual C++ 2008 Redistributable | The C++ Library. |  |

# Problem on setting up

## cannot start Routing server

run command :

> mongos

Error message :

2015-02-03T16:24:28.372+0700 [mongosMain] ERROR: could not verify that config servers are in sync :: caused by :: config servers 192.168.1.19:28000 and 192.168.1.204:28000 differ: {} vs { chunks: "d41d8cd98f00b204e9800998ecf8427e", databases: "0195c739c938fdafd9632279694ea64e", shards: "d41d8cd98f00b204e9800998ecf8427e", version: "b267b05f25abd9801f30de9c4f8f6bfc" }

2015-02-03T16:24:28.372+0700 [mongosMain] configServer connection startup check failed

Solving:

<http://docs.mongodb.org/manual/tutorial/migrate-config-servers-with-same-hostname/>

copy file and folder in storage path “db” from config server 1 to another config servers

## I’ve enabled sharding and added a second shard, but all the data is still on one server. Why?

First, ensure that you’ve declared a shard key for your collection. Until you have configured the shard key, MongoDB will not create chunks, and sharding will not occur.

Next, keep in mind that the default chunk size is 64 MB. As a result, in most situations, the collection needs to have at least 64 MB of data before a migration will occur.

Additionally, the system which balances chunks among the servers attempts to avoid superfluous migrations. Depending on the number of shards, your shard key, and the amount of data, systems often require at least 10 chunks of data to trigger migrations.

You can run db.printShardingStatus() to see all the chunks present in your cluster.

<http://docs.mongodb.org/manual/faq/sharding/>

Solving:

For testing “Shard” is it work or not. We can change the chunk size by command following

*use config*

*db.settings.save( { \_id:"chunksize", value: <sizeInMB> } )*

[*http://docs.mongodb.org/manual/tutorial/modify-chunk-size-in-sharded-cluster/*](http://docs.mongodb.org/manual/tutorial/modify-chunk-size-in-sharded-cluster/)

and then then insert the data to mangodb we will see the data balancing in server 2.

## Run command “shardCollection”

Run command:

mongos> db.runCommand( {shardCollection:"psndatainfo.metadata",key:{"SessionId":

"hashed"}})

Error Message:

{

"ok" : 0,

"errmsg" : "ensureIndex failed to create index on primary shard: write r

esults unavailable from 192.168.1.166:28000 :: caused by :: Location15907 could

not initialize sharding on connection 192.168.1.166:28000 (192.168.1.166) :: cau

sed by :: mongos specified a different config database string : stored : 192.168

.1.19:28000,192.168.1.204:28000,192.168.1.83:28000 vs given : 192.168.1.204:2800

0"

}

Solve:

Delete all database of shard server and config server

## How to solve “The specified service has been marked for deletion” error

<http://stackoverflow.com/questions/20561990/how-to-solve-the-specified-service-has-been-marked-for-deletion-error>

# Appendices

<http://en.wikipedia.org/wiki/NoSQL>

<http://en.wikipedia.org/wiki/MongoDB>

<http://docs.mongodb.org/manual/>

**Replicate & Sharding**

<http://www.tutorialspoint.com/mongodb/mongodb_replication.htm>

<http://www.thegeekstuff.com/2014/02/mongodb-replication/>

<http://alexyu.se/content/2012/04/mongodb-quick-start-replica-sets-and-sharding>

<http://stackoverflow.com/questions/16232025/why-mongodb-config-servers-must-be-one-or-three-only>

**SSL**

<http://info.mongodb.com/rs/mongodb/images/MongoDB_Subscription_Value.pdf>

<http://c343c.org/junko/2015/02/09/mongodb-c-ssl-client-certificate/>

http://pki-tutorial.readthedocs.org/en/latest/simple/index.html#create-pem-bundle