python-ezdb

Release 0.0.1.r32.aa0634c

GeorgeRaven

Jul 15, 2021

TABLE OF CONTENTS

Python-ezdb is a database abstraction library, currently supporting MongoDB. This library seeks to make it easy to quickly set-up and use databases, without having to go through very low level steps to do so.

CHAPTER

ONE

THE OPEN SOFTWARE LICENSE 3.0 (OSL-3.0)

This Open Software License (the "License") applies to any original work of authorship (the "Original Work") whose owner (the "Licensor") has placed the following licensing notice adjacent to the copyright notice for the Original Work:

Licensed under the Open Software License version 3.0

1) Grant of Copyright License. Licensor grants You a worldwide, royalty-free, non-exclusive, sublicensable license, for the duration of the copyright, to do the following:

a) to reproduce the Original Work in copies, either alone or as part of a collective work;

b) to translate, adapt, alter, transform, modify, or arrange the Original Work, thereby creating derivative works ("Derivative Works") based upon the Original Work;

c) to distribute or communicate copies of the Original Work and Derivative Works to the public, with the proviso that copies of Original Work or Derivative Works that You distribute or communicate shall be licensed under this Open Software License;

d) to perform the Original Work publicly; and

e) to display the Original Work publicly.

2) Grant of Patent License. Licensor grants You a worldwide, royalty-free, non-exclusive, sublicensable license, under patent claims owned or controlled by the Licensor that are embodied in the Original Work as furnished by the Licensor, for the duration of the patents, to make, use, sell, offer for sale, have made, and import the Original Work and Derivative Works.

3) Grant of Source Code License. The term "Source Code" means the preferred form of the Original Work for making modifications to it and all available documentation describing how to modify the Original Work. Licensor agrees to provide a machine-readable copy of the Source Code of the Original Work along with each copy of the Original Work that Licensor distributes. Licensor reserves the right to satisfy this obligation by placing a machine-readable copy of the Source Code in an information repository reasonably calculated to permit inexpensive and convenient access by You for as long as Licensor continues to distribute the Original Work.

4) Exclusions From License Grant. Neither the names of Licensor, nor the names of any contributors to the Original Work, nor any of their trademarks or service marks, may be used to endorse or promote products derived from this Original Work without express prior permission of the Licensor. Except as expressly stated herein, nothing in this License grants any license to Licensor's trademarks, copyrights, patents, trade secrets or any other intellectual property. No patent license is granted to make, use, sell, offer for sale, have made, or import embodiments of any patent claims other than the licensed claims defined in Section 2. No license is granted to the trademarks of Licensor even if such marks are included in the Original Work. Nothing in this License shall be interpreted to prohibit Licensor from licensing under terms different from this License any Original Work that Licensor otherwise would have a right to license.

5) External Deployment. The term "External Deployment" means the use, distribution, or communication of the Original Work or Derivative Works in any way such that the Original Work or Derivative Works may be used by anyone other than You, whether those works are distributed or communicated to those persons or made available

as an application intended for use over a network. As an express condition for the grants of license hereunder, You must treat any External Deployment by You of the Original Work or a Derivative Work as a distribution under section 1(c).

6) Attribution Rights. You must retain, in the Source Code of any Derivative Works that You create, all copyright, patent, or trademark notices from the Source Code of the Original Work, as well as any notices of licensing and any descriptive text identified therein as an "Attribution Notice." You must cause the Source Code for any Derivative Works that You create to carry a prominent Attribution Notice reasonably calculated to inform recipients that You have modified the Original Work.

7) Warranty of Provenance and Disclaimer of Warranty. Licensor warrants that the copyright in and to the Original Work and the patent rights granted herein by Licensor are owned by the Licensor or are sublicensed to You under the terms of this License with the permission of the contributor(s) of those copyrights and patent rights. Except as expressly stated in the immediately preceding sentence, the Original Work is provided under this License on an "AS IS" BASIS and WITHOUT WARRANTY, either express or implied, including, without limitation, the warranties of non-infringement, merchantability or fitness for a particular purpose. THE ENTIRE RISK AS TO THE QUALITY OF THE ORIGINAL WORK IS WITH YOU. This DISCLAIMER OF WARRANTY constitutes an essential part of this License. No license to the Original Work is granted by this License except under this disclaimer.

8) Limitation of Liability. Under no circumstances and under no legal theory, whether in tort (including negligence), contract, or otherwise, shall the Licensor be liable to anyone for any indirect, special, incidental, or consequential damages of any character arising as a result of this License or the use of the Original Work including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses. This limitation of liability shall not apply to the extent applicable law prohibits such limitation.

9) Acceptance and Termination. If, at any time, You expressly assented to this License, that assent indicates your clear and irrevocable acceptance of this License and all of its terms and conditions. If You distribute or communicate copies of the Original Work or a Derivative Work, You must make a reasonable effort under the circumstances to obtain the express assent of recipients to the terms of this License. This License conditions your rights to undertake the activities listed in Section 1, including your right to create Derivative Works based upon the Original Work, and doing so without honoring these terms and conditions is prohibited by copyright law and international treaty. Nothing in this License is intended to affect copyright exceptions and limitations (including "fair use" or "fair dealing"). This License shall terminate immediately and You may no longer exercise any of the rights granted to You by this License upon your failure to honor the conditions in Section 1(c).

10) Termination for Patent Action. This License shall terminate automatically and You may no longer exercise any of the rights granted to You by this License as of the date You commence an action, including a cross-claim or counterclaim, against Licensor or any licensee alleging that the Original Work infringes a patent. This termination provision shall not apply for an action alleging patent infringement by combinations of the Original Work with other software or hardware.

11) Jurisdiction, Venue and Governing Law. Any action or suit relating to this License may be brought only in the courts of a jurisdiction wherein the Licensor resides or in which Licensor conducts its primary business, and under the laws of that jurisdiction excluding its conflict-of-law provisions. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any use of the Original Work outside the scope of this License or after its termination shall be subject to the requirements and penalties of copyright or patent law in the appropriate jurisdiction. This section shall survive the termination of this License.

12) Attorneys' Fees. In any action to enforce the terms of this License or seeking damages relating thereto, the prevailing party shall be entitled to recover its costs and expenses, including, without limitation, reasonable attorneys' fees and costs incurred in connection with such action, including any appeal of such action. This section shall survive the termination of this License.

13) Miscellaneous. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.

14) Definition of "You" in This License. "You" throughout this License, whether in upper or lower case, means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with you. For purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

15) Right to Use. You may use the Original Work in all ways not otherwise restricted or conditioned by this License or by law, and Licensor promises not to interfere with or be responsible for such uses by You.

16) Modification of This License. This License is Copyright © 2005 Lawrence Rosen. Permission is granted to copy, distribute, or communicate this License without modification. Nothing in this License permits You to modify this License as applied to the Original Work or to Derivative Works. However, You may modify the text of this License and copy, distribute or communicate your modified version (the "Modified License") and apply it to other original works of authorship subject to the following conditions: (i) You may not indicate in any way that your Modified License is the "Open Software License" or "OSL" and you may not use those names in the name of your Modified License; (ii) You must replace the notice specified in the first paragraph above with the notice "Licensed under <insert your license name here>" or with a notice of your own that is not confusingly similar to the notice in this License; and (iii) You may not claim that your original works are open source software unless your Modified License has been approved by Open Source Initiative (OSI) and You comply with its license review and certification process.

CHAPTER

TWO

INSTALLATION

Note: Certain distributions link `python` to `python2` and others link it to `python3`. For disambiguation python, pip, and virtualenv shall mean their python v3 versions here, i.e. `python3`, `pip3`, `virtualenv3`.

Warning: You will need to have git, and python installed for any of the below methods to work. You will also need MongoDB if you intend to create a local database, (more than likely), but python-ezdb can still connect to already running databases without it if you happen to have one already.

This section will outline various methods for installation of python-ezdb, and its dependencies. Not all methods are equal there are slight variations between them, which are outlined in the respective sections below, along with instructions for each method:

2.1 Files-only/ development

2.2 Automated

This section discusses the more automated and repeatable installation methods for Nemesyst, but they do not contain all the files needed to learn, and begin developing Nemesyst integrated applications, rather this includes just the bare-bones Nemesyst ready for your deployment.

2.2.1 pip

For now you can use pip via:

```
pip install git+https://github.com/DreamingRaven/python-ezdb.git#branch=master
```

2.2.2 Archlinux

Install python-ezdb-git[AUR].

2.3 Manual

This section outlines the manual methods of installing python-ezdb, for maximum control at the cost of time and repeatability.

2.3.1 setup.py

```
git clone https://github.com/DreamingRaven/python-ezdb
cd python-ezdb
python setup.py install
```

2.3.2 Archlinux

```
git clone https://github.com/DreamingRaven/python-ezdb
cd python-ezdb/.archlinux/
makepkg -si
```

2.3.3 Virtual env

To create the python-virtualenv:

```
virtualenv venv
```

If python 3 is not the default python for your virtualenvironment, simply delete the new directory `venv` and instead use the following to generate a new one with python3:

```
virtualenv -p python3 venv
```

To then use the newly created virtual environment:

```
source venv/bin/activate
```

OR if you are using a terminal like fish:

```
source venv/bin/activate.fish
```

To install Nemesyst and all its dependencies into a virtual environment while it is being used (activated):

```
pip install git+https://github.com/DreamingRaven/python-ezdb.git#branch=master
```

To exit the virtual environment:

```
deactivate
```

CHAPTER

THREE

SERVING WITH MONGODB

MongoDB is an object based database system. Python-ezdb provides a nice higher level interface "*Mongo*" by using PyMongo and os commands to make managing MongoDB more streamlined and less reliant on direct connection management to MongoDB.

3.1 Creating a basic database

Disambiguation: we define a basic database as a standalone MongoDB instance with one universal administrator and one read/write user with password authentication.

While it is possible it is highly discouraged to use Nemesyst to create the users you require as this is quite complicated to manage and may lead to more problems than its worth compared to simply creating a database and adding a user manually using something like the following:

3.1.1 Manual creation of MongoDB

Files-only/ development **creation of database example:**

```
mongod --config ./examples/configs/basic_mongo_config.yaml
```

This will create a database with all the MongoDB defaults as it is an empty yaml file. If you would instead want a more complex setup please take a look at `examples/configs/authenticated_replicaset.yaml` instead, but you will need to generate certificates and keys for this so it is probably a poor place to start but will be what you will want to use in production as a bare minimum security.

3.1.2 Docker-Compose creation of MongoDB

Docker-Compose, ***Files-only/ development*** **creation of database example:**

```
docker-compose up
```

This similar to the *Manual creation of MongoDB* creation uses a simple config file to launch the database. This can be changed in `docker-compose.yaml`. At this point you will need to connect to the running MongoDB instance (see: *Connecting to a running database*) to create your main administrator user, with "userAdminAnyDatabase" role. After this you can use the following to close the Docker container with the database:

Docker-Compose, ***Files-only/ development*****, closing Docker-Compose database example:**

```
docker-compose down
```

Note: Don't worry we set our docker-compose.yaml to save its files in `/containers/mongodb` so they are persistent between runs of docker-compose. If you need to delete the MongoDB database that is where you can find them.

3.1.3 Connecting to a running database

To be able to fine tune, create users, update etc it will be necessary to connect to MongoDB in one form or another. Nemesyst can help you log in or you can do it manually.

Note: If there is no userAdmin or userAdminAnyDatabase then unless expressly configured there will be a localhost exception which will allow you to log in and create this user. If this user exists the localhost exception will close. Please ensure you configure this user as they can grant any role or rights to anyone and would be a major security concern along with making it very difficult to admin your database.

Mongo

To connect to an non-sharded database with autnentication but no TLS/SSL:

Bash shell example:

```
mongo hostname:port -u username --authenticationDatabase database name
```

To connect to a slightly more complicated scenario with authentication, TLS, and sharding enabled:

Bash shell example:

```
mongo hostname:port -u username --authenticationDatabase database name --tls␣
↪--tlsCAFile path to ca file --tlsCertificateKeyFile path to cert key file
```

3.1.4 Creating database users

You will absolutely need a user with at least "userAdminAnyDatabase" role. Connect to the running database see *Connecting to a running database*.

Mongo shell create a new role-less user:

```
db.createUser({user: "username", pwd: passwordPrompt(), roles: []})
```

Mongo shell grant role to existing user example:

```
db.grantRolesToUser(
"username",
[
  { role: "userAdminAnyDatabase", db: "admin" }
])
```

Mongo shell create user and grant `userAdminAnyDatabase` in one:

```
db.createUser({user: "username", pwd: passwordPrompt(), roles: [
↪{role:"userAdminAnyDatabase", db: "admin"}]})
```

Note: Since this user belongs to `admin` in the previous examples that means the authenticationDatabase is `admin` when authenticating as this user as per the instructions in "*Connecting to a running database*".

3.2 From basic database to replica sets

This section will outline how to take a currently standard database and turn it into a replica set

3.2.1 MongoDB config file setup for replica sets

Files-only/ development **example `./examples/mongod.d/replica.yaml`:**

```
security:
  keyFile: mongo.key
  authorization: enabled
replication:
  replSetName: rs0
processManagement:
  fork: true
net:
  bindIp: 0.0.0.0
  port: 65535
systemLog:
  path: mongolog.log
  destination: file
storage:
  dbPath: /data/db
  directoryPerDB: true
```

3.2.2 Checking the current status of the replica sets

The replica sets should not be initialized which we can check.

Mongo shell **Check the current status of replica sets:** Command:

```
rs.status()
```

Out:

```
{
    "operationTime" : Timestamp(0, 0),
    "ok" : 0,
    "errmsg" : "no replset config has been received",
    "code" : 94,
    "codeName" : "NotYetInitialized",
    "$clusterTime" : {
            "clusterTime" : Timestamp(0, 0),
            "signature" : {
                    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                    "keyId" : NumberLong(0)
            }
    }
}
```

There should be no config present also, which we can also check.

Mongo shell **Check the current status of replica set config:** Command:

```
rs.conf()
```

Out:

```
2020-03-12T13:43:46.998+0000 E  QUERY    [js] uncaught exception: Error:␣
↪Could not retrieve replica set config: {
    "operationTime" : Timestamp(0, 0),
    "ok" : 0,
    "errmsg" : "no replset config has been received",
    "code" : 94,
    "codeName" : "NotYetInitialized",
    "$clusterTime" : {
            "clusterTime" : Timestamp(0, 0),
            "signature" : {
                    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                    "keyId" : NumberLong(0)
            }
    }
} :
rs.conf@src/mongo/shell/utils.js:1531:11
@(shell):1:1
```

If the config does not yet exist like above, or is not initialized we should initialize it.

3.2.3 Initializing and populating the replica set config

Mongo shell Initialize the config: Command:

```
rs.initiate()
```

Now the rs.conf should exist so we are free to add members to the replica set.

Mongo shell Add a member to the config: Command:

```
rs.add({host: "hostname:port"})
```

3.3 From plaintext database to TLS/SSL

First it is necessary to generate a key and a certificate file for our use. For now these can be self signed but in future you may want to look at getting them signed by a certificate authority such as LetsEncrypt.

3.3.1 Generating a certificate authority key, and then a self signed certificate

This example shows generating an encrypted RSA key. If you would instead prefer it to be plaintext remove `-aes-256-cbc`.

Bash shell generate encrypted RSA certificate authority private key example:

```
openssl genpkey -algorithm RSA -aes-256-cbc -pkeyopt rsa_keygen_bits:4096 -out␣
↪ssl_key
```

Bash shell generate x509 certificate file valid for 365 days example:

```
openssl req -key ssl_key -x509 -new -days 365 -out signed_certificate
```

Note: It should be noted that MongoDB does hostname validation using this certificate file. The things we are aware of are the hostname must match, and in the case of replicas one thing like organization name must match between the communicating replicas if they use SSL/TLS. It should also be noted that Pymongo unlike mongo does not interpret between hostname and ip address the same way, an example can be found in *MongoDB/ Serving Issues*.

This should now leave you with two files, an `ssl_key` and a `signed_certificate`. We can now combine these two together to create a .pem file with both to provide to MongoDB. This new file will is the certificate-key file.

Bash shell a `ckfile.pem` file example:

```
cat signed_certificate > ckfile.pem
cat ssl_key >> ckfile.pem
```

3.3.2 Using our certificate and key as the server

Almost all of the required changes take place in the mongodb config file/ how you call mongod itself.

***Files-only/ development* `mongod.conf`/ `mongod.yaml` example:**

```
net:
  bindIp: 127.0.0.1
  port: 27017
  tls:
    mode: requireTLS
    certificateKeyFile: ckfile.pem # this should be a path to this file
    certificateKeyFilePassword: password
    allowConnectionsWithoutCertificates: true
```

An example TLS/SSL enabled replica set database config file can be seen below. This however requires a few additional files for authenticating the databases and certificates for SSL/TLS that you will need to generate.

***Files-only/ development* example `./examples/mongod.d/authenticated_replicaset.yaml`:**

```
security:
  keyFile: mongo.key
  authorization: enabled
replication:
  replSetName: rs0
processManagement:
  fork: true
net:
  bindIp: 0.0.0.0
  port: 65535 # the highest number port possible to use
  tls:
    mode: requireTLS
    certificateKeyFile: ckfile.pem # path to ckfile.pem
    certificateKeyFilePassword: passwordIfItHasOneAtAll # password for␣
↪ckfile
    allowConnectionsWithoutCertificates: true
systemLog:
  path: mongolog.log
  destination: file
```

(continues on next page)

(continued from previous page)

```
storage:
  dbPath: /data/db
  directoryPerDB: true
```

3.3.3 Using our certificate and key as the client

Self signed certificates are just as valid, and as good as any other certificate, with one exception; only machines we can install our certificate on will trust us, unless we disable this layer of trust entirely. Thus if our certificate is self signed then the certificate file in our case `signed_certificate` must be installed on each machine that we desire to trust our MongoDB instance.

3.4 Troubleshooting

Please see *MongoDB/ Serving Issues*

3.5 Further reading

MongoDB core:

- config file options
- user management

Replica sets:

- rs.initiate
- add members

TLS/SSL:

- arch wiki tls

CHAPTER

FOUR

MONGO

4.1 ezdb.mongo.Mongo

Nemesyst MongoDB abstraction/ Handler. This handler helps abstract some pymongo functionality to make it easier for us to use a MongoDB database for our deep learning purposes.

4.2 Example usage

This unit test also briefly shows how to use gridfs by dumping tuple items in the form (dict(), object), where the dict will become the files metadata and the object is some form of the data that can be sequentialized into the database.

Warning: Mongo uses subprocess.Popen in init, start, and stop, since these threads would otherwise lock up ezdb.mongo.Mongo, with time.sleep() to wait for the database to startup, and shutdown. Depending on the size of your database it may be necessary to extend the length of time time.sleep() as larger databases will take longer to startup and shutdown.

Setting up a basic database, and initializing it with a user.

```
    def setUp(self):
        """Predefined setUp function for preparing tests, in our case
        creating the database."""
        import os
        from ezdb.mongo import Mongo

        # the path and directory we want to use to store the database files
        db_path = "./unit_test_db"
        db = Mongo({"pylog": null_printer, "db_path": db_path,
                    "db_log_path": db_path})
        # initialise the database files and create a basic user
        db.init()
        # start the database with authenticaton
        db.start()

        # for tests only to check db directory is created
        self.assertTrue(os.path.isdir(db_path))
```

Connecting to and dumping data to a database using normal mongodb requests.

```
    def test_dump(self):
        """Test/ example of dump and retrieve from a MongoDB database."""
        from ezdb.mongo import Mongo

        db = Mongo({"pylog": null_printer})
        self.assertIsInstance(db, Mongo)
        db.connect()
        db.dump(db_collection_name="test", data={"success": 1})
        cursor = db.getCursor(db_collection_name="test")
        for batch in db.getBatches(db_data_cursor=cursor):
            self.assertEqual(len(batch), 1)
            for doc in batch:
                self.assertEqual(doc["success"], 1)
```

Using Gridfs to dump larger files in chunks to database.

```
    def test_gridfs(self):
        """Test/ example of gridfs dump and retrieve from MongoDB."""
        from ezdb.mongo import Mongo

        db = Mongo({"pylog": null_printer})
        self.assertIsInstance(db, Mongo)
        db.connect()
        db.dump(db_collection_name="test", data=({"success": 1}, b'success'))
        cursor = db.getCursor(db_collection_name="test.files")
        for batch in db.getFiles(db_data_cursor=cursor):
            for grid in batch:
                # check ids match
                self.assertEqual(grid["_id"], grid["metadata"]["_id"])
                # read file and check is equal to what we put in
                self.assertEqual(grid["gridout"].read(), b'success')
```

Completely removing the database, this completely removes all your data.

```
    def tearDown(self):
        """Predefined tearDown function for cleaning up after tests,
        in our case deleting any generated db files."""
        import os
        import shutil
        from ezdb.mongo import Mongo

        db_path = "./unit_test_db"
        db = Mongo({"pylog": null_printer, "db_path": db_path,
                    "db_log_path": db_path})
        db.stop()
        if(db_path is not None):
            shutil.rmtree(db_path)

        # for tests only to check db directory has been removed
        self.assertFalse(os.path.isdir(db_path))
```

4.3 API

class ezdb.mongo.**Mongo**(*args: Optional[dict] = None*, *logger: Optional[print] = None*)

Python2/3 compatible MongoDb utility wrapper.

This wrapper saves its state in an internal overridable dictionary such that you can adapt it to your requirements, if you should need to do something unique, the caveat being it becomes harder to read.

Parameters

- **args** (*dictionary*) – Dictionary of overides.
- **logger** (*function address*) – Function address to print/ log to (default: print).

Example Mongo({"db_user_name": "someUsername", "db_password": "somePassword"})

Example Mongo()

connect(*db_ip: Optional[str] = None*, *db_port: Optional[str] = None*, *db_authentication: Optional[str] = None*, *db_authentication_database=None*, *db_user_name: Optional[str] = None*, *db_password: Optional[str] = None*, *db_name: Optional[str] = None*, *db_replica_set_name: Optional[str] = None*, *db_replica_read_preference: Optional[str] = None*, *db_replica_max_staleness: Optional[str] = None*, *db_tls: Optional[bool] = None*, *db_tls_ca_file: Optional[str] = None*, *db_tls_certificate_key_file: Optional[str] = None*, *db_tls_certificate_key_file_password: Optional[str] = None*, *db_tls_crl_file: Optional[str] = None*, *db_collection_name: Optional[str] = None*) → pymongo.database.Database

Connect to a specific mongodb database.

This sets the internal db client which is neccessary to connect to and use the associated database. Without it operations such as dump into the database will fail. This is replica set capable.

Parameters

- **db_ip** (*string*) – Database hostname or ip to connect to.
- **db_port** (*string*) – Database port to connect to.
- **db_authentication** (*string*) – The authentication method to use on db.
- **db_user_name** (*string*) – Username to use for authentication to db_name.
- **db_password** (*string*) – Password for db_user_name in database db_name.
- **db_name** (*string*) – The name of the database to connect to.
- **db_replica_set_name** (*string*) – Name of the replica set to connect to.
- **db_replica_read_preference** (*string*) – What rep type to prefer reads from.
- **db_replica_max_staleness** (*string*) – Max seconds behind is replica allowed.
- **db_tls** (*bool*) – use TLS for db connection.
- **db_tls_ca_file** (*string*) – Certificate authority file path.
- **db_tls_certificate_key_file** (*string*) – Certificate and key file for tls.
- **db_tls_certificate_key_file_password** (*string*) – Cert and key file pass.
- **db_tls_crl_file** (*string*) – Certificate revocation list file path.
- **db_collection_name** (*string*) – GridFS collection to use.

Returns database client object

Return type pymongo.database.Database

debug() → None
Log function to help track the internal state of the class.

Simply logs working state of args dict.

donate(*other*, *other_collection*, *db_collection_name*, *db_data_cursor=None*, *sum: Optional[int] = None*, *frequency: Optional[int] = None*, *count: Optional[int] = None*)
Donate documents to another db collection.

Like giving blood, we are not getting anything back to self, other than maybe gratification.

dump(*db_collection_name: str*, *data: dict*, *db: Optional[pymongo.database.Database] = None*) → None
Import data dictionary into database.

Parameters

- **db_collection_name** (*string*) – Collection name to import into.
- **data** (*dictionary*) – Data to import into database.
- **db** (*pymongo.database.Database*) – Database to import data into.

Example dump(db_collection_name="test", data={"subdict":{"hello": "world"}})

getBatches(*db_batch_size: Optional[int] = None*, *db_data_cursor: Optional[pymongo.command_cursor.CommandCursor] = None*) → list
Get database cursor data in batches.

Parameters

- **db_batch_size** (*integer*) – The number of items to return in a single round.
- **db_data_cursor** (*command_cursor.CommandCursor*) – The cursor to use to retrieve data from db.

Returns yields a list of items requested.

Return type list of dicts

Todo desperateley needs a rewrite and correction of bug. Last value always fails. I want this in a magic function too to make it easy.

getCursor(*db: Optional[pymongo.database.Database] = None*, *db_pipeline: Optional[list] = None*, *db_collection_name: Optional[str] = None*) → pymongo.command_cursor.CommandCursor
Use aggregate pipeline to get a data-cursor from the database.

This cursor is what mongodb provides to allow you to request the data from the database in a manner you control, instead of just getting a big dump from the database.

Parameters

- **db_pipeline** (*list of dicts*) – Mongodb aggregate pipeline data to transform and retrieve the data as you request.
- **db_collection_name** (*str*) – The collection name which we will pull data from using the aggregate pipeline.
- **db** (*pymongo.database.Database*) – Database object to operate pipeline on.

Returns Command cursor to fetch the data with.

Return type pymongo.command_cursor.CommandCursor

getFiles(*db_batch_size: Optional[int] = None*, *db_data_cursor: Optional[pymongo.command_cursor.CommandCursor] = None*, *db_collection_name: Optional[str] = None*, *db: Optional[pymongo.database.Database] = None*) → list

Get gridfs files from mongodb by id using cursor to .files.

Parameters

- **db_batch_size** (*integer*) – The number of items to return in a single round.
- **db_data_cursor** (*command_cursor.CommandCursor*) – The cursor to use to retrieve data from db.
- **db_collection_name** (*str*) – The top level collecton name not including .chunks or .files where gridfs is to operate.
- **db** (*pymongo.database.Database*) – Database object to operate pipeline on.

Returns yields a list of tuples containing (item requested, metadata).

init(*db_path: Optional[str] = None*, *db_log_path: Optional[str] = None*, *db_log_name: Optional[str] = None*, *db_config_path: Optional[str] = None*) → None

Initialise the database.

Includes ensuring db path and db log path exist and generating, creating the DB files, and adding an authentication user. All of this should be done on a localhost port so that the unprotected database is never exposed.

Parameters

- **db_path** (*string*) – Desired directory of MongoDB database files.
- **db_log_path** (*string*) – Desired directory of MongoDB log files.
- **db_log_name** (*string*) – Desired name of log file.
- **db_config_path** (*string*) – Config file to pass to MongoDB.

login(*db_port: Optional[str] = None*, *db_user_name: Optional[str] = None*, *db_password: Optional[str] = None*, *db_name: Optional[str] = None*, *db_ip: Optional[str] = None*) → None

Log in to database, interrupt, and availiable via cli.

Parameters

- **db_port** (*string*) – Database port to connect to.
- **db_user_name** (*string*) – Database user to authenticate as.
- **db_password** (*string*) – User password to authenticate with.
- **db_name** (*string*) – Database to authenticate to, the authentication db.
- **db_ip** (*string*) – Database ip to connect to.

start(*db_ip: Optional[str] = None*, *db_port: Optional[str] = None*, *db_path: Optional[str] = None*, *db_log_path: Optional[str] = None*, *db_log_name: Optional[str] = None*, *db_cursor_timeout: Optional[int] = None*, *db_config_path: Optional[str] = None*, *db_replica_set_name: Optional[str] = None*) → subprocess.Popen

Launch an on machine database with authentication.

Parameters

- **db_ip** (*list*) – List of IPs to accept connectiongs from.
- **db_port** (*string*) – Port desired for database.
- **db_path** (*string*) – Path to parent dir of database.

- **db_log_path** (*string*) – Path to parent dir of log files.
- **db_log_name** (*string*) – Desired base name for log files.
- **db_cursor_timeout** (*integer*) – Set timeout time for unused cursors.
- **db_path** – Config file path to pass to MongoDB.

Return type subprocess.Popen

Returns Subprocess of running MongoDB.

stop(*db_path=None*) → subprocess.Popen
Stop a running local database.

Parameters **db_path** (*string*) – The path to the database to shut down.

Returns Subprocess of database closer.

Return type subprocess.Popen

userAdd(*username: str*, *password: str*, *roles: list*) → None
Take new credentials and create new user in database

CHAPTER

FIVE

DISAMBIGUATION

5.1 General

5.1.1 IP (address)

An Internet Protocol (IP) address can be thought of as a computer postcode to help identify where a machine can be found, for the purposes of communicating to it; it is difficult to mail a letter without a postcode.

local *IP (address)* **example:** `192.168.1.1`

5.1.2 port

A computers port is usually associated with an *IP (address)*. This port further specifies where specifically to communicate to. Following the post analogy the post code is the *IP (address)*, but the house number is the port. This analogy breaks down when we consider that each computer can and usually does have multiple ports in the thousands, so maybe the computer is a property tycoon with multiple homes that can be contacted.

port **example:** 192.168.1.1 `:22`

5.2 Authentication

5.3 TLS Related

5.4 Shard/ Replica Related

CHAPTER

SIX

TROUBLESHOOTING

6.1 MongoDB/ Serving Issues

Error: not master and slaveOk=false This error means you have attempted to read from a replica set that is not the master. If you would like to read from SECONDARY-ies/ slaves (anything thats not the PRIMARY) you can:

Mongo shell:

```
rs.slaveOk()
```

pymongo.errors.OperationFailure: Authentication failed This error means likely means that your authentication credentials are incorrect, you will want to check the values you are passing to pymongo via Nemesyst to ensure they are what you are expecting. In particular pay special attention to Mongo().connect() as it is the life blood of all connections but since the driver is a lazy driver it wont fail until you attempt to use the connection.

pymongo.errors.ServerSelectionTimeoutError: 192.168.1.10:27017: [SSL: CERTIFICATE_VERIFY_FAILED] certificat This error is a implementation quirk of pymongo not converting between ip addresses and hostname strings implicitly even if the certificate stipulates the desired IP address correctly for other things such as the mongo client. My only recommendation is to either use hostnames even if that only be explicit in `/etc/hosts` or disabling TLS but both are bad options for anything more than testing.

INDEX