MimicDB Documentation

Release 1.0.1

Nathan Cahill

April 20, 2015

Contents

The Python implementation of MimicDB works with the Boto library and several different backends including Redis.

CHAPTER 1

Installation

By default, MimicDB requires Redis (although other backends can be used instead).

```
$ pip install boto
$ pip install redis
$ pip install mimicdb
```

CHAPTER 2

Quickstart

If you're using Boto already, replace `boto` imports with `mimicdb` imports.

Change:

```
from boto.s3.connection import S3Connection
from boto.s3.key import Key
```

To:

```
from mimicdb.s3.connection import S3Connection
from mimicdb.s3.key import Key
```

Additionally, import the MimicDB object itself, and initiate the backend:

```
from mimicdb import MimicDB
MimicDB()
```

After establishing a connection for the first time, sync the connection to save the metadata locally:

```
conn = S3Connection(KEY, SECRET)
conn.sync()
```

Or sync only a couple buckets from the connection:

```
conn.sync('bucket1', 'bucket2')
```

After that, upload, download and list as you usually would. API calls that can be responded to locally will return instantly without hitting S3 servers. API calls that are made to S3 using MimicDB will be mimicked locally to ensure consistency with the remote servers.

Pass `force=True` to most functions to force a call to the S3 API. This also updates the local database.

CHAPTER 3

Contents

3.1 Installation

By default, MimicDB requires Redis (although other backends can be used instead).

```
$ pip install boto
$ pip install redis
$ pip install mimicdb
```

3.2 Quickstart

If you're using Boto already, replace `boto` imports with `mimicdb` imports.

Change:

```
from boto.s3.connection import S3Connection
from boto.s3.key import Key
```

To:

```
from mimicdb.s3.connection import S3Connection
from mimicdb.s3.key import Key
```

Additionally, import the MimicDB object itself, and initiate the backend:

```
from mimicdb import MimicDB
MimicDB()
```

After establishing a connection for the first time, sync the connection to save the metadata locally:

```
conn = S3Connection(KEY, SECRET)
conn.sync()
```

Or sync only a couple buckets from the connection:

```
conn.sync('bucket1', 'bucket2')
```

After that, upload, download and list as you usually would. API calls that can be responded to locally will return instantly without hitting S3 servers. API calls that are made to S3 using MimicDB will be mimicked locally to ensure consistency with the remote servers.

Pass `force=True` to most functions to force a call to the S3 API. This also updates the local database.

3.3 Backends

Besides the default Redis backend, MimicDB has SQLite and in-memory backends available. To use a different backend, pass an instance of it to the MimicDB __init__ function:

```
from mimicdb import MimicDB
from mimicdb.backends.sqlite import SQLite

MimicDB(backend=SQLite())
```

Keep in mind that the default database for the SQLite backend is an in-memory database. It, along with the in-memory backend, will be destroyed when the process finishes running. For persistent data, use Redis or a custom backend.

class mimicdb.backends.**Backend**

Base class for MimicDB backends. Extendable to support custom backends. A custom backend **must** implement each of the functions of the base class.

```
from mimicdb.backends import Backend

class MyAwesomeBackend(Backend):
    def __init__(self):

    etc.
```

class mimicdb.backends.default.**Redis**(**args*, ***kwargs*)

Default backend for MimicDB. Initiated with identical parameters as `redis.StrictRedis`.

Parameters ***args, **kwargs** (*args*) – StrictRedis.__init__() parameters

```
from mimicdb.backends.default import Redis

redis = Redis(host='localhost', port=6379, db=0)
```

class mimicdb.backends.sqlite.**SQLite**(**args*, ***kwargs*)

SQLite backend. Pass `sqlite3.connect()` parameters to __init__. If no parameters are passed, `:memory:` is chossen as the default database.

Parameters ***args, **kwargs** (*args*) – sqlite3.connect() parameters

```
from mimicdb.backends.sqlite import SQLite

sqlite = SQLite('mimicdb.db')
```

class mimicdb.backends.memory.**Memory**

In-Memory backend. A good example for building a custom backend.

```
from mimicdb.backends.memory import Memory

memory = Memory()
```

3.4 Syncing S3 State

If there are buckets or keys on S3 before starting to use MimicDB, it's important to sync the connection to ensure consistency with the S3 API in future calls.

Syncing only has to be run once. Be aware that buckets with large numbers of keys can take a long time to sync. It's best to not use S3Connection.sync() in frequently used code paths.

All MimicDB data in the synced buckets is cleared before syncing to ensure a clean slate.

```
from mimicdb.s3.connection import S3Connection

conn = S3Connection(KEY, SECRET)
conn.sync()
```

S3Connection.**sync**(**buckets*)
Sync either a list of buckets or the entire connection.

Force all API calls to S3 and populate the database with the current state of S3.

Parameters ***buckets** (**string*) – Buckets to sync

Bucket.**sync**()
Sync a bucket.

Force all API calls to S3 and populate the database with the current state of S3.

3.5 Forcing S3 API Calls

All of Boto's S3 classes and functions are wrapped in MimicDB functions to provide access to the local data instead of the S3 API. This means that changes to buckets and keys on S3 (outside of MimicDB), won't be reflected in MimicDB unless the API call is forced.

Forcing an S3 API call also stores the response in MimicDB, so the forced call only has to happen once to retrieve new changes.

In most cases, passing `force=True` to the function will cause the S3 API call to be executed.

> **Warning:** `Bucket.__iter__` can not have arguments passed to it, so it defaults to not calling the API.

The following functions can be forced:

S3Connection.**get_bucket**(*bucket_name*, *validate=True*, *headers=None*, *force=None*)
Return a bucket from MimicDB if it exists. Return a S3ResponseError if the bucket does not exist and validate is passed.

Parameters **force** (*boolean*) – If true, API call is forced to S3

S3Connection.**get_all_buckets**(**args*, ***kwargs*)
Return a list of buckets in MimicDB.

Parameters **force** (*boolean*) – If true, API call is forced to S3

Bucket.**get_key**(**args*, ***kwargs*)
Return the key from MimicDB.

Parameters **force** (*boolean*) – If true, API call is forced to S3

Bucket.**get_all_keys**(**args*, ***kwargs*)
Return a list of keys from MimicDB.

Parameters **force** (*boolean*) – If true, API call is forced to S3

Bucket.**list**(**args*, ***kwargs*)
Return an iterable of keys from MimicDB.

Parameters **force** (*boolean*) – If true, API call is forced to S3

m

B

G

L

M

R

S