Bakthat Documentation

Release 0.6.0

Thomas Sileo

May 20, 2015

Contents

Release v0.6.0.

Bakthat is a MIT licensed backup framework written in Python, it's both a command line tool and a Python module that helps you manage backups on Amazon S3/Glacier and OpenStack Swift. It automatically compress, encrypt (symmetric encryption) and upload your files.

Here are some features:

- Compress with tarfile
- Encrypt with beefish (**optional**)
- Upload/download to S3 or Glacier with boto
- Local backups inventory stored in a SQLite database with peewee
- Delete older than, and Grandfather-father-son backup rotation supported
- Possibility to sync backups database between multiple clients via a centralized server
- Exclude files using .gitignore like file
- Extendable with plugins

You can restore backups **with** or **without** bakthat, you just have to download the backup, decrypt it with Beefish command-line tool and untar it.

You may also check out BakServer, a self-hosted Python server, to help you manage backups anywhere and keep multiple bakthat client synchronized across servers.

CHAPTER 1

Requirements

Bakthat requirements are automatically installed when installing bakthat, but if you want you can install them manually:

```
$ pip install -r requirements.txt
```

- aaargh
- pycrypto
- beefish
- boto
- GrandFatherSon
- peewee
- byteformat
- pyyaml
- sh
- requests
- events

If you want to use OpenStack Swift, following additional packages are also required.

- python-swiftclient
- python-keystoneclient

CHAPTER 2

Overview

2.1 Bakthat command line tool

```
$ pip install bakthat

$ bakthat configure

$ bakthat backup mydir
Backing up mydir
Password (blank to disable encryption):
Password confirmation:
Compressing...
Encrypting...
Uploading...
Upload completion: 0%
Upload completion: 100%

or

$ cd mydir
$ bakthat backup

$ bakthat show
2013-03-05T19:36:15 s3  3.1 KB  mydir.20130305193615.tgz.enc

$ bakthat restore mydir
Restoring mydir.20130305193615.tgz.enc
Password:
Downloading...
Decrypting...
Uncompressing...

$ bakthat delete mydir.20130305193615.tgz.enc
Deleting mydir.20130305193615.tgz.enc
```

2.2 Bakthat Python API

```
import logging
import sh
logging.basicConfig(level=logging.INFO)
```

```
from bakthat.helper import BakHelper

BACKUP_NAME = "myhost_mysql"
BACKUP_PASSWORD = "mypassword"
MYSQL_USER = "root"
MYSQL_PASSWORD = "mypassword"

with BakHelper(BACKUP_NAME, password=BACKUP_PASSWORD, tags=["mysql"]) as bh:
    sh.mysqldump("-p{0}".format(MYSQL_PASSWORD),
                 u=MYSQL_USER,
                 all_databases=True,
                 _out="dump.sql")
    bh.backup()
    bh.rotate()
```

CHAPTER 3

Installation

With pip/easy_install:

```
$ pip install bakthat
```

From source:

```
$ git clone https://github.com/tsileo/bakthat.git
$ cd bakthat
$ sudo python setup.py install
```

Next, you need to set your AWS credentials:

```
$ bakthat configure
```

CHAPTER 4

User Guide

4.1 User Guide

Everything you need to know as a user.

4.1.1 Getting Started

Basic usage, "bakthat -h" or "bakthat <command> -h" to show the help.

If you haven't configured bakthat yet, you should run:

```
$ bakthat configure
```

Note: Even if you have set a default destination, you can use a different destination using the `-d`/`--destination` parameter, for example, if S3 is the default destination, to use Glacier just add "-d glacier" or "–destination glacier".

4.1.2 Backup

```
$ bakthat backup --help
usage: bakthat backup [-h] [-d DESTINATION] [--prompt PROMPT] [-t TAGS]
                      [-p PROFILE] [-c CONFIG] [-k KEY]
                      [filename]

positional arguments:
  filename

optional arguments:
  -h, --help            show this help message and exit
  -d DESTINATION, --destination DESTINATION
                        s3|glacier|swift
  --prompt PROMPT       yes|no
  -t TAGS, --tags TAGS  space separated tags
  -p PROFILE, --profile PROFILE
                        profile name (default by default)
  -c CONFIG, --config CONFIG
                        path to config file
  -k KEY, --key KEY     Custom key for periodic backups (works only with
                        BakManager.io hook.)
```

When backing up file, bakthat store files in gzip format, under the following format: **originaldirname.utctime.tgz**, where utctime is a UTC datetime (%Y%m%d%H%M%S).

Note: If you try to backup a file already gziped, bakthat will only rename it (change extention to .tgz and append utctime).

And you can also disable compression by setting `compress: false` in you configuration file (~/.bakthat.yml by default).

Bakthat let you tag backups to retrieve them faster, when backing up a file, just append the `--tags/-t` argument, tags are space separated, when adding multiple tags, just quote the whole string (e.g. `--tags "tag1 tag2 tag3"`)

Since version **0.5.2**, you can set the password with BAKTHAT_PASSWORD environment variable.

```
$ BAKTHAT_PASSWORD=mypassword bakthat backup myfile
```

If you don't specify a filename/dirname, bakthat will backup the current working directory.

```
$ cd /dir/i/want/to/bak
backup to S3
$ bakthat backup
or
$ bakthat backup /dir/i/want/to/bak

$ bakthat backup /my/dir -t "tag1 tag2"

you can also backup a single file
$ bakthat backup /home/thomas/mysuperfile.txt

backup to Glacier
$ bakthat backup myfile -d glacier

set the password with BAKTHAT_PASSWORD environment variable
$ BAKTHAT_PASSWORD=mypassword bakthat backup myfile

disable password prompt
$ bakthat backup myfile --prompt no
```

Excluding files

New in version 0.5.5.

Bakthat use a ".gitignore style" way to exclude files using Unix shell-style wildcards.

There is two way to exclude files:

- by creating a **.bakthatexclude** file at the root of the directory you want to backup.
- by specifying a file directly with the `--exclude-file` argument.

By default when performing a backup, if no exclude file is specified, it will look for either a **.bakthatexclude** file or a **.gitignore** file. So you backup a git repository, it will use the existing .gitignore if available.

Here is an example **.bakthatexclude** file, wich exlude all .pyc and .log files, and both tmp and cache directory.

```
*.pyc
*.log
tmp
cache
```

Reduced redundancy using S3

New in version 0.5.5.

If you backup to S3, you can active the reduced redundancy by using the `--s3-reduced-redundancy` flag.

```
bakthat backup --s3-reduced-redundancy
```

Temp directory

You can change the temp directory location by setting the TMPDIR, TEMP or TMP environment variables if the backup is too big to fit in the default temp directory.

```
$ export TMP=/home/thomas
```

4.1.3 Restore

```
$ bakthat restore --help
usage: bakthat restore [-h] [-d DESTINATION] [-p PROFILE] [-c CONFIG]
                       filename

positional arguments:
  filename

optional arguments:
  -h, --help            show this help message and exit
  -d DESTINATION, --destination DESTINATION
                        s3|glacier|swift
  -p PROFILE, --profile PROFILE
                        profile name (default by default)
  -c CONFIG, --config CONFIG
                        path to config file
```

When restoring a backup, you can:

- specify **filename**: the latest backups will be restored
- specify **stored filename** directly, if you want to restore an older version.

```
$ bakthat restore bak

if you want to restore an older version
$ bakthat restore bak20120927
or
$ bakthat restore bak20120927.tgz.enc

restore from Glacier
$ bakthat restore bak -d glacier
```

Note: When restoring from Glacier, the first time you call the restore command, the job is initiated, then you can check manually whether or not the job is completed (it takes 3-5h to complete), if so the file will be downloaded and restored.

4.1.4 Listing backups

Let's start with the help for the show subcommand:

```
$ bakthat show --help
usage: bakthat show [-h] [-d DESTINATION] [-t TAGS] [-p PROFILE]
                    [-c CONFIG]
                    [query]

positional arguments:
  query                 search filename for query

optional arguments:
  -h, --help            show this help message and exit
  -d DESTINATION, --destination DESTINATION
                        glacier|s3|swift, show every destination by default
  -t TAGS, --tags TAGS  tags space separated
  -p PROFILE, --profile PROFILE
                        profile name (all profiles are displayed by default)
  -c CONFIG, --config CONFIG
                        path to config file
```

So when listing backups, you can:

- filter by query (filename/stored filename)
- filter by destination (either glacier or s3)
- filter by tags
- filter by profile (if you manage multiple AWS/bucket/vault)

Example:

```
show everything
$ bakthat show

search for a file stored on s3:
$ bakthat show myfile -d s3
```

4.1.5 Delete

If the backup is not stored in the default destination, you have to specify it manually.

Note: Remember that the delete command delete only the most recent matching backup.

```
$ bakthat delete bak

$ bakthat delete bak -d glacier
```

4.1.6 Delete older than

Delete backup older than the given string interval, like 1M for 1 month and so on.

- **s** seconds
- **m** minutes

- **h** hours
- **D** days
- **W** weeks
- **M** months
- **Y** Years

```
$ bakthat delete_older_than bakname 3M

$ bakthat delete_older_than bakname 3M2D8h20m5s

$ bakthat delete_older_than bakname 3M -d glacier
```

4.1.7 Backup rotation

If you make automated with baktaht, it makes sense to rotate your backups.

Bakthat allows you to rotate backups using Grandfather-father-son backup rotation, you can set a default rotation configuration.

```
$ bakthat configure_backups_rotation
```

Now you can rotate a backup set:

```
$ bakthat rotate_backups bakname
```

Note: Bakthat rely on the GrandFatherSon module to compute rotations, so if you need to setup more complex rotation scheme (like hourly backups), refer to the docs and change the rotation settings manually in your configuration file.

4.1.8 Accessing bakthat Python API

Check out the *Developer's Guide*.

4.1.9 Configuration

Bakthat stores configuration in YAML format, to have the same configuration handling for both command line and Python module use.

You can also handle **multiples profiles** if you need to manage multiple AWs account or vaults/buckets.

By default, your configuration is stored in **~/.bakthat.yml**, but you can specify a different file with the `-c/--config` parameter.

To get started, you can run `bakthat configure`.

```
$ bakthat configure
```

Here is what a configuration object looks like:

```
access_key: YOUR_ACCESS_KEY
secret_key: YOUR_SECRET_KEY
region_name: us-east-1
glacier_vault: myvault
s3_bucket: mybucket
```

The **region_name** key is optionnal is you want to use **us-east-1**.

Managing profiles

Here is how profiles are stored, you can either create them manually or with command line.

```
default:
  access_key: YOUR_ACCESS_KEY
  secret_key: YOUR_SECRET_KEY
  region_name: us-east-1
  glacier_vault: myvault
  s3_bucket: mybucket
myprofile:
  access_key: YOUR_ACCESS_KEY
  secret_key: YOUR_SECRET_KEY
  region_name: us-east-1
  glacier_vault: myvault
  s3_bucket: mybucket
```

To create a profile from command line with bakthat:

```
$ bakthat configure --profile mynewprofile

$ bakthat configure -h
usage: bakthat configure [-h] [-p PROFILE]

optional arguments:
  -h, --help            show this help message and exit
  -p PROFILE, --profile PROFILE
                        profile name (default by default)
```

Once your profile is configured, you can use it with `--profile`/`-p` argument.

```
$ bakthat backup -p myprofile
$ bakthat show -p myprofile
```

OpenStack Swift support

New in version 0.5.0.

If you use OpenStack Swift as backend, **auth_version** and **auth_url** key are required in configuration. Following are sample configurations both temp_auth and keystone auth.

```
temp_auth:
  access_key: ACCOUNT:USER
  secret_key: YOUR_SECRET_KEY
  region_name:
  glacier_vault:
  s3_bucket: mybucket
  default_destination: swift
  auth_url: https://<SWIFT_FQDN>/auth/v1.0
  auth_version: '1'
keystone:
  access_key: ACCOUNT:USER
  secret_key: YOUR_SECRET_KEY
  region_name:
  glacier_vault:
```

```
  s3_bucket: mybucket
  default_destination: swift
  auth_url: https://<KEYSTONE_FQDN>/v2.0
  auth_version: '2'
```

4.1.10 Stored metadata

Batkthat stores some data about your backups in a SQLite database (using peewee as wrapper) for few reasons:

- to allow you to filter them efficiently.
- to avoid making a lot of requests to AWS.
- to let you sync your bakthat data with multiple servers.

Here is a example of data stored in the SQLite database:

```
{u'backend': u's3',
 u'backend_hash': u'9813aa99062d7a226f3327478eff3f63bf5603cd86999a42a2655f5d460e8e143c63822cb8e2f899
 u'backup_date': 1362508575,
 u'filename': u'mydir',
 u'is_deleted': 0,
 u'last_updated': 1362508727,
 u'metadata': {u'is_enc': True},
 u'size': 3120,
 u'stored_filename': u'mydir.20130305193615.tgz.enc',
 u'tags': []}
```

All the keys are explicit, except **backend_hash**, which is the hash of your AWS access key concatenated with either the S3 bucket, either the Glacier vault. This key is used when syncing backups with multiple servers.

4.1.11 Backup/Restore Glacier inventory

Bakthat automatically backups the local Glacier inventory (a dict with filename => archive_id mapping) to your S3 bucket under the "bakthat_glacier_inventory" key.

You can retrieve bakthat custom inventory without waiting:

```
$ bakthat show_glacier_inventory
```

or

```
$ bakthat show_local_glacier_inventory
```

You can trigger a backup mannualy:

```
$ bakthat backup_glacier_inventory
```

And here is how to restore the glacier inventory from S3:

```
$ bakthat restore_glacier_inventory
```

4.1.12 S3 and Glacier IAM permissions

```
{
    "Statement": [
    {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::S3_BUCKET_NAME*"
    },
    {
        "Effect": "Allow",
        "Action": "glacier:*"
        "Resource": "arn:aws:glacier:AWS_REGION:AWS_ACCOUNT_ID:vaults/GLACIER_VAULT_NAME",
    }
    ]
}
```

CHAPTER 5

Developer's Guide

5.1 Developer's Guide

5.1.1 Low level API

You can access low level API (the same used when using bakthat in command line mode) from **bakthat** root module.

```
import bakthat

# roration is optional
bakthat_conf = {'access_key': 'YOURACCESSKEY',
                'secret_key': 'YOURSECRETKEY',
                'glacier_vault': 'yourvault',
                's3_bucket': 'yours3bucket',
                'region_name': 'eu-west-1',
                'rotation': {'days': 7,
                             'first_week_day': 5,
                             'months': 6,
                             'weeks': 6}}

bakthat.backup("/dir/i/wanto/bak", conf=bakthat_conf)

bakthat.backup("/dir/i/wanto/bak", conf=bakthat_conf, destination="glacier")

# or if you want to have generated the configuration file with "bakthat configure" or created ~/.bakt
bakthat.backup("/dir/i/wanto/bak")

bakthat.ls()

# restore in the current working directory
bakthat.restore("bak", conf=bakthat_conf)
```

Event Hooks

New in version 0.6.0.

You can configure hook to be executed on the following events:

- before_backup
- on_backup

- before_restore
- on_restore
- before_delete
- on_delete
- before_delete_older_than
- on_delete_older_than
- before_rotate_backups
- on_rotate_backups

So, **before_** events are executed at the beginning of the action, and **on_** events are executed just before the end.

For each action, a **session_id** (an uuid4) is assigned, so you can match up **before_** and **on_** events.

Every callback receive the session_id as first argument, and for **on_** callbacks, you can retrieve the result of the function, most of the time a Backup object or a list of Backup object, depending of the context.

```
from bakthat import backup, events

def before_backup_callback(session_id):
    print session_id, "before_backup"

def on_backup_callback(session_id, backup):
    print session_id, "on_backup", backup

events.before_backup += before_backup_callback
events.on_backup += on_backup_callback

bakthat.backup("/home/thomas/mydir")
```

Bakthat makes use of Events to handle all the "event things".

5.1.2 Plugins

New in version 0.6.0.

You can create plugins to extend bakthat features, all you need to do is to subclass `bakthat.plugin.Plugin` and implement an `activate` (and optionally `deactivate`, executed just before exiting) method.

The `activate` and `deactivate` method is called only once. `activate` is called when the plugin is initialized, and `deactivate` (you can see it like a cleanup function) is called at exit.

Note: For now, you can't create new command yet with plugin (maybe in the future).

By default, plugins are stored in **~/.bakthat_plugins/**, but you can change the plugins location by setting the `plugins_dir` setting in your configuration file.

```
default:
  plugins_dir: /home/thomas/.bakthat_plugins
```

And to enable plugins, add it to the `plugins` array:

```
default:
  plugins: [test_plugin.TestPlugin, filename.MyPlugin]
```

You can access **raw profile configuration** using `self.conf`, and **bakthat logger** using `self.log` (e.g. `self.log.info("hello")`) and in any methods. You can also hook events directly on `self`, like `self.on_backup += mycallback`.

Your First Plugin

Here is a basic plugin example, a `TimerPlugin` in **test_plugin.py**:

```
import time
from bakthat.plugin import Plugin

class TestPlugin(Plugin):
    def activate(self):
        self.start = {}
        self.stop = {}
        self.before_backup += self.before_backup_callback
        self.on_backup += self.on_backup_callback

    def before_backup_callback(self, session_id):
        self.start[session_id] = time.time()
        self.log.info("before_backup {0}".format(session_id))

    def on_backup_callback(self, session_id, backup):
        self.stop[session_id] = time.time()
        self.log.info("on_backup {0} {1}".format(session_id, backup))
        self.log.info("Job duration: {0}s".format(self.stop[session_id] - self.start[session_id]))
```

Now, we can enable it:

```
default:
  plugins: [test_plugin.TestPlugin]
```

Finally, we can check that our plugin is actually working:

```
$ bakthat backup mydir
before_backup 4028dfc7-7a17-4a99-b3fe-88f6e4879bda
Backing up /home/thomas/mydir
Password (blank to disable encryption):
Compressing...
Uploading...
Upload completion: 0%
Upload completion: 100%
Upload completion: 0%
Upload completion: 100%
on_backup 4028dfc7-7a17-4a99-b3fe-88f6e4879bda <Backup: mydir.20130604191055.tgz>
Job duration: 4.34407806396s
```

Monkey Patching

With plugin, you have the ability to extend or modify everything in the `activate function`.

Here is an example, which update the `Backups` model at runtime:

```
from bakthat.plugin import Plugin
from bakthat.models import Backups
```

```
class MyBackups(Backups):
    @classmethod
    def my_custom_method(self):
        return True


class ChangeModelPlugin(Plugin):
    """ A basic plugin implementation. """
    def activate(self):
        global Backups
        self.log.info("Replace Backups")
        Backups = MyBackups
```

More on event hooks

See **Event Hooks** for more informations and Events documentation.

5.1.3 Helpers

BakHelper

BakHelper is a context manager that makes create backup script with bakthat (and it works well with sh) an easy task.

It takes care of create a temporary directory and make it the current working directory so you can just dump files to backup or call system command line tool lilke mysqldump/mongodump/and so on with the help of sh.

Here is a minimal example.

```
import logging
logging.basicConfig(level=logging.INFO)

from bakthat.helper import BakHelper

with BakHelper("mybackup", tags=["mybackup"]) as bh:

    with open("myfile.txt", "w") as f:
        f.write("mydata")

    bh.backup()
    bh.rotate()
```

Now test the script:

```
$ python mybackupscript.py
INFO:root:Backing up /tmp/mybackup_JVTGOM
INFO:root:Compressing...
INFO:root:Uploading...
INFO:bakthat.backends:Upload completion: 0%
INFO:bakthat.backends:Upload completion: 100%
```

You can also use it like a normal class:

```
import logging
import sh
logging.basicConfig(level=logging.INFO)
```

```
from bakthat.helper import BakHelper

bakthat_conf = {'access_key': 'YOURACCESSKEY',
                'secret_key': 'YOURSECRETKEY',
                'glacier_vault': 'yourvault',
                's3_bucket': 'yours3bucket',
                'region_name': 'eu-west-1',
                'rotation': {'days': 7,
                             'first_week_day': 5,
                             'months': 6,
                             'weeks': 6}}

bh = BakHelper(conf=bakthat_conf)
with open("myfile.txt", "w") as f:
    f.write("mydata")
bh.backup("myfile.txt")
bh.rotate("myfile.txt")
```

Create a MySQL backup script with BakHelper

Here is a MySQL backup script, it makes use of sh to call system **mysqldump**.

See also:

You can also check out a MongoDB backup script example here.

```
import logging
import sh
logging.basicConfig(level=logging.INFO)

from bakthat.helper import BakHelper

BACKUP_NAME = "myhost_mysql"
BACKUP_PASSWORD = "mypassword"
MYSQL_USER = "root"
MYSQL_PASSWORD = "mypassword"

with BakHelper(BACKUP_NAME, password=BACKUP_PASSWORD, tags=["mysql"]) as bh:
    sh.mysqldump("-p{0}".format(MYSQL_PASSWORD),
                 u=MYSQL_USER,
                 all_databases=True,
                 _out="dump.sql")
    bh.backup()
    bh.rotate()
```

KeyValue

New in version 0.4.5.

KeyValue is a simple "key value store" that allows you to quickly store/retrieve strings/objects on Amazon S3. All values are serialized with json, so **you can directly backup any json serializable value**.

It can also takes care of compressing (with gzip) and encrypting (optionnal).

Compression in enabled by default, you can disable it by passing compress=False when setting a key.

Also, backups stored with KeyValue can be restored with bakthat restore and show up in bakthat show.

```
from bakthat.helper import KeyValue
import json

bakthat_conf = {'access_key': 'YOURACCESSKEY',
                'secret_key': 'YOURSECRETKEY',
                'glacier_vault': 'yourvault',
                's3_bucket': 'yours3bucket',
                'region_name': 'es-east-1'}

kv = KeyValue(conf=bakthat_conf)

mydata = {"some": "data"}
kv.set_key("mykey", mydata)

mydata_restored = kv.get_key("mykey")

data_url = kv.get_key_url("mykey", 60)  # url expires in 60 secondes

kv.delete_key("mykey")

kv.set_key("my_encrypted_key", "myvalue", password="mypassword")
kv.get_key("my_encrypted_key", password="mypassword")

# You can also disable gzip compression if you want:
kv.set_key("my_non_compressed_key", {"my": "data"}, compress=False)
```

5.1.4 Accessing bakthat SQLite database

Since bakthat stores custom backups metadata (see *Stored metadata*), you can execute custom SQL query.

CHAPTER 6

API Documentation

6.1 Bakthat API

6.1.1 Bakthat

These functions are called when using bakthat in command line mode and are the foundation of the bakthat module.

backup

bakthat.**backup**(*filename='/var/build/user_builds/bakthat/checkouts/latest/docs'*, *destination=None*, *profile='default'*, *config='/home/docs/.bakthat.yml'*, *prompt='yes'*, *tags=[]*, *key=None*, *exclude_file=None*, *s3_reduced_redundancy=False*, ***kwargs*)

Perform backup.

Parameters

- **filename** (*str*) – File/directory to backup.
- **destination** (*str*) – s3|glacier|swift
- **prompt** (*str*) – Disable password promp, disable encryption, only useful when using bakthat in command line mode.
- **tags** (*str or list*) – Tags either in a str space separated, either directly a list of str (if calling from Python).
- **password** (*str*) – Password, empty string to disable encryption.
- **conf** (*dict*) – Override/set AWS configuration.
- **custom_filename** (*str*) – Override the original filename (only in metadata)

Return type dict

Returns A dict containing the following keys: stored_filename, size, metadata, backend and filename.

restore

bakthat.**restore**(*filename*, *destination=None*, *profile='default'*, *config='/home/docs/.bakthat.yml'*, ***kwargs*)

Restore backup in the current working directory.

Parameters

- **filename** (*str*) – File/directory to backup.
- **destination** (*str*) – s3|glacier|swift
- **profile** (*str*) – Profile name (default by default).
- **conf** (*dict*) – Override/set AWS configuration.

Return type bool

Returns True if successful.

info

show

bakthat.**show**(*query=''*, *destination=''*, *tags=''*, *profile='default'*, *config='/home/docs/.bakthat.yml'*)

delete

bakthat.**delete**(*filename*, *destination=None*, *profile='default'*, *config='/home/docs/.bakthat.yml'*, ***kwargs*)

Delete a backup.

Parameters

- **filename** (*str*) – stored filename to delete.
- **destination** (*str*) – glacier|s3|swift
- **profile** (*str*) – Profile name (default by default).
- **conf** (*dict*) – A dict with a custom configuration.
- **conf** – Override/set AWS configuration.

Return type bool

Returns True if the file is deleted.

delete_older_than

bakthat.**delete_older_than**(*filename*, *interval*, *profile='default'*, *config='/home/docs/.bakthat.yml'*, *destination=None*, ***kwargs*)

Delete backups matching the given filename older than the given interval string.

Parameters

- **filename** (*str*) – File/directory name.
- **interval** (*str*) – Interval string like 1M, 1W, 1M3W4h2s... (s => seconds, m => minutes, h => hours, D => days, W => weeks, M => months, Y => Years).
- **destination** (*str*) – glacier|s3|swift
- **conf** (*dict*) – Override/set AWS configuration.

Return type list

Returns A list containing the deleted keys (S3) or archives (Glacier).

rotate_backups

bakthat.**rotate_backups**(*filename*, *destination=None*, *profile='default'*, *config='/home/docs/.bakthat.yml'*, ***kwargs*)

Rotate backup using grandfather-father-son rotation scheme.

Parameters

- **filename** (*str*) – File/directory name.
- **destination** (*str*) – s3|glacier|swift
- **conf** (*dict*) – Override/set AWS configuration.
- **days** (*int*) – Number of days to keep.
- **weeks** (*int*) – Number of weeks to keep.
- **months** (*int*) – Number of months to keep.
- **first_week_day** (*str*) – First week day (to calculate wich weekly backup keep, saturday by default).

Return type list

Returns A list containing the deleted keys (S3) or archives (Glacier).

6.1.2 Backends

BakthatBackend

class bakthat.backends.**BakthatBackend**(*conf={}*, *profile='default'*)

Handle Configuration for Backends.

The profile is only useful when no conf is None.

Parameters

- **conf** (*dict*) – Custom configuration
- **profile** (*str*) – Profile name

GlacierBackend

class bakthat.backends.**GlacierBackend**(*conf={}*, *profile='default'*)

Backend to handle Glacier upload/download.

backup_inventory()

Backup the local inventory from shelve as a json string to S3.

delete_job(*filename*)

Delete the job entry for the filename.

Parameters **filename** (*str*) – Stored filename.

download(*keyname*, *job_check=False*)

Initiate a Job, check its status, and download the archive if it's completed.

get_job_id(*filename*)

Get the job_id corresponding to the filename.

Parameters **filename** (*str*) – Stored filename.

load_archives_from_s3()
Fetch latest inventory backup from S3.

restore_inventory()
Restore inventory from S3 to local shelve.

retrieve_archive(*archive_id*, *jobid*)
Initiate a job to retrieve Galcier archive or download archive.

retrieve_inventory(*jobid*)
Initiate a job to retrieve Galcier inventory or output inventory.

S3Backend

class bakthat.backends.**S3Backend**(*conf={}*, *profile='default'*)
Backend to handle S3 upload/download.

cb(*complete*, *total*)
Upload callback to log upload percentage.

SwiftBackend

class bakthat.backends.**SwiftBackend**(*conf={}*, *profile='default'*)
Backend to handle OpenStack Swift upload/download.

cb(*complete*, *total*)
Upload callback to log upload percentage.

RotationConfig

class bakthat.backends.**RotationConfig**(*conf={}*, *profile='default'*)
Hold backups rotation configuration.

6.1.3 Helper

BakHelper

class bakthat.helper.**BakHelper**(*backup_name*, ***kwargs*)
Helper that makes building scripts with bakthat better faster stronger.

Designed to be used as a context manager.

Parameters

- **backup_name** (*str*) – Backup name also the prefix for the created temporary directory.
- **destination** (*str*) – Destination (glacier|s3)
- **password** (*str*) – Password (Empty string to disable encryption, disabled by default)
- **profile** (*str*) – Profile name, only valid if no custom conf is provided
- **conf** (*dict*) – Override profiles configuration
- **tags** (*list*) – List of tags

backup(*filename=None*, ***kwargs*)
Perform backup.

Parameters

- **filename** (*str*) – File/directory to backup.
- **password** (*str*) – Override already set password.
- **destination** (*str*) – Override already set destination.
- **tags** (*list*) – Tags list
- **profile** (*str*) – Profile name
- **conf** (*dict*) – Override profiles configuration

Return type dict

Returns A dict containing the following keys: stored_filename, size, metadata and filename.

delete_older_than (*filename=None*, *interval=None*, ***kwargs*)
Delete backups older than the given interval string.

Parameters

- **filename** (*str*) – File/directory name.
- **interval** (*str*) – Interval string like 1M, 1W, 1M3W4h2s... (s => seconds, m => minutes, h => hours, D => days, W => weeks, M => months, Y => Years).
- **destination** (*str*) – Override already set destination.
- **profile** (*str*) – Profile name
- **conf** (*dict*) – Override profiles configuration

Return type list

Returns A list containing the deleted keys (S3) or archives (Glacier).

enable_sync (*api_url*, *auth=None*)
Enable synchronization with *bakthat.sync.BakSyncer* (optional).

Parameters

- **api_url** (*str*) – Base API URL.
- **auth** (*tuple*) – Optional, tuple/list (username, password) for API authentication.

restore (*filename*, ***kwargs*)
Restore backup in the current working directory.

Parameters

- **filename** (*str*) – File/directory to backup.
- **password** (*str*) – Override already set password.
- **destination** (*str*) – Override already set destination.
- **profile** (*str*) – Profile name
- **conf** (*dict*) – Override profiles configuration

Return type bool

Returns True if successful.

rotate (*filename=None*, ***kwargs*)
Rotate backup using grandfather-father-son rotation scheme.

Parameters

- **filename** (*str*) – File/directory name.
- **destination** (*str*) – Override already set destination.
- **profile** (*str*) – Profile name
- **conf** (*dict*) – Override profiles configuration

Return type list

Returns A list containing the deleted keys (S3) or archives (Glacier).

sync()
Shortcut for calling BakSyncer.

KeyValue

class bakthat.helper.**KeyValue**(*conf={}*, *profile='default'*)
A Key Value store to store/retrieve object/string on S3.

Data is gzipped and json encoded before uploading, compression can be disabled.

delete_key(*keyname*)
Delete the given key.

Parameters **keyname** (*str*) – Key name

get_key(*keyname*, ***kwargs*)
Return the object stored under keyname.

Parameters

- **keyname** (*str*) – Key name
- **default** (*str*) – Default value if key name does not exist, None by default

Return type str

Returns The key content as string, or default value.

get_key_url(*keyname*, *expires_in*, *method='GET'*)
Generate a URL for the keyname object.

Be careful, the response is JSON encoded.

Parameters

- **keyname** (*str*) – Key name
- **expires_in** (*int*) – Number of the second before the expiration of the link
- **method** (*str*) – HTTP method for access

Rtype str

Returns The URL to download the content of the given keyname

set_key(*keyname*, *value*, ***kwargs*)
Store a string as keyname in S3.

Parameters

- **keyname** (*str*) – Key name
- **value** (*bool*) – Value to save, will be json encoded.
- **compress** – Compress content with gzip, True by default

6.1.4 Sync

BakSyncer

class bakthat.sync.**BakSyncer**(*conf=None*)
Helper to synchronize change on a backup set via a REST API.

No sensitive information is transmitted except (you should be using https): - API user/password - a hash (hashlib.sha512) of your access_key concatened with

your s3_bucket or glacier_vault, to be able to sync multiple client with the same configuration stored as metadata for each bakckupyy.

Parameters **conf** (*dict*) – Config (url, username, password)

register()
Register/create the current host on the remote server if not already registered.

sync()
Draft for implementing bakthat clients (hosts) backups data synchronization.

Synchronize Bakthat sqlite database via a HTTP POST request.

Backups are never really deleted from sqlite database, we just update the is_deleted key.

It sends the last server sync timestamp along with data updated since last sync. Then the server return backups that have been updated on the server since last sync.

On both sides, backups are either created if they don't exists or updated if the incoming version is newer.

sync_auto()
Trigger sync if autosync is enabled.

bakmanager_hook

bakthat.sync.**bakmanager_hook**(*conf*, *backup_data*, *key=None*)
First version of a hook for monitoring periodic backups with BakManager (https://bakmanager.io).

Parameters

- **conf** (*dict*) – Current profile config
- **backup_data** (*dict*) – Backup data (size)
- **key** (*str*) – Periodic backup identifier

6.1.5 Utils

bakthat.utils.**_timedelta_total_seconds**(*td*)
Python 2.6 backward compatibility function for timedelta.total_seconds.

Parameters **td** (*timedelta object*) – timedelta object

Return type float

Returns The total number of seconds for the given timedelta object.

bakthat.utils.**_interval_string_to_seconds**(*interval_string*)
Convert internal string like 1M, 1Y3M, 3W to seconds.

Parameters **interval_string** (*str*) – Interval string like 1M, 1W, 1M3W4h2s... (s => seconds, m => minutes, h => hours, D => days, W => weeks, M => months, Y => Years).

Return type int

Returns The conversion in seconds of interval_string.

6.1.6 Models

class bakthat.models.**Backups**(**args*, ***kwargs*)
Backups Model.

class bakthat.models.**Inventory**(**args*, ***kwargs*)
Filename => archive_id mapping for glacier archives.

class bakthat.models.**Jobs**(**args*, ***kwargs*)
filename => job_id mapping for glacier archives.

classmethod **get_job_id**(*filename*)
Try to retrieve the job id for a filename.

Parameters **filename** (*str*) – Filename

Return type str

Returns Job Id for the given filename

classmethod **update_job_id**(*filename*, *job_id*)
Update job_id for the given filename.

Parameters

- **filename** (*str*) – Filename
- **job_id** (*str*) – New job_id

Returns None

class bakthat.models.**Config**(**args*, ***kwargs*)
key => value config store.

CHAPTER 7

Articles

- Bakthat 0.5.0 Released With OpenStack Swift Support and BakManager Integration
- Backing Up MongoDB to Amazon Glacier/S3 With Python Using Sh and Bakthat
- Bakthat 0.4.5 Released, Introducing a New Helper: KeyValue
- Bakthat 0.2.0 Released Adding Amazon Glacier Support

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

b

Symbols

B

C

D

E

G

I

J

K

L

R

S

U