Lastpass Backup Documentation

Release 0.1

Rick Henry

Nov 17, 2018

Contents:

CHAPTER 1

Installation

You first need to install the lastpass commandline tool for your platform. It is used internally for accessing the lastpass api.

Then clone the repo and run `python3 setup.py install`

CHAPTER 2

Usage

```
from lp_backup import Runner

# create backup runner
example_backup_runner = Runner("/home/YOUR_USER/.config/lp_backup.yml")
# run backup
backup_file_name = example_backup_runner.backup()
print(backup_file_name)

# restore backup to /tmp/example-full-restore.csv (which is PLAIN TEXT, be sure to␣
↪delete after use)
backup_file_name.restore(backup_file_name, "/tmp/test-full-restore.csv")
```

2.1 Installation

You must install the lastpass cli and have it available in your path.

You will need Python 3.6 or later installed, as well as pip.

For now, clone the repo and run

```
$ python3 setup.py install
```

from inside it. You may want to do it in a virtual environment.

2.2 Configuration

2.2.1 Getting Started

A great starting point is the example configuration file.

2.2.2 What You'll Need

- Your lastpass username (email) and password.
- The lastpass command line tool.
- Somewhere to put the backups. This can be a local directory, S3 or S3 compatible service, or any builtin in filesystem of pyfilesystem2. If pyfilesystem2 has an extension for the filesystem you want to use, it should work but you will need to install the extension separately.

2.2.3 Configuration File

The configuration file is in YAML syntax. If you are unfamiliar, this seems helpful. There are several configuration options available:

- `Email:` *Required* The email address for your lastpass account
- `Trust:` Whether to trust this computer after the backup in the lastpass cli
- `Encryption Key:` This can be set to `null` or `generate` for your first run. It is highly recommended that it be left set to `generate`. On first run, an encryption key will be generated and saved into the configuration file (don't lose it). This file should be kept safe.
- `Compression:` Whether to compress the data. If `true`, the data will be lzma compressed and saved with a `.xz` extension.
- `Date`: Whether to include the date in filenames.
- `Prefix`: Path prefix (folders) to put the backup file in.
- `Backing Store:` List of locations to put backups. Specify a uri as documented at pyfilesystem2 for osfs, mountfs, ftpfs. You can use non-native filesystems (e.g. sshfs) but you will need to install the corresponding extensions to pyfilesystem2. Obviously this must be accessible to the backup program when it runs. You may also specify S3 for Amazon Simple Storage Service or s3 compatible service, as well as a WebDav server.
  - Mixed Example:

```
Backing Store:
- URI: /home/YOURUSER/lastpass_backups
- Type: S3
  Bucket: lastpass_backup_bucket
- Type: webdav
  Base URL: https://example.com
  Root: /remote.php/webdav
  Username: YOURUSER
  Password: $WEBDAV_PASSWORD
```

  - `S3`: It is recommended that you use external authentication for S3 (env vars or IAM roles), but you may specify Key ID and Secret Key, preferably through env vars. Be sure to specify Endpoint URL and ensure authentication for S3 compatible services Specify Date: True for adding date information to the backup file/folders.
    - Example S3 configuration:

```
Backing Store:
  - Type: S3
    Bucket: mybackupbucket
```

    - Example S3 compatible service (digital ocean spaces):

```
Backing Store:
  - Type: S3
    Bucket: mybackupbucket
    Endpoint URL: https://nyc3.digitaloceanspaces.com
    Key ID: $DOKEY
    Secret Key: $DOSECRET
```

- `webdav`: You can use any webdav service with password authentication. You can store the relevant information in the config file, or specify environment variables The webdav library in use separates the base part of the url from the root of the webdav server, so split your url accordingly.

  Example Configuration:

```
Backing Store:
 - Type: webdav
   Base URL: https://mynextcloudserver.com
   Root: /remote.php/webdav
   Username: john
   Password: my-single-app-password
```

  IF YOU DO THIS: please please please don't use your main password, set up multi-factor authentication and generate a single app password. This is a plain text file protected only by filesystem permissions (which should probably be 600, by the way). Even better, make a dedicated user account for this purpose, so that none of your other data is put at risk.

2.3 Usage

2.3.1 Creating Backups

Create a simple python script, for instance `backup.py` with the contents:

```
from lp_backup import Runner

run = Runner(path='/path/to/config/file.yaml')
backup_file_name = run.backup()
```

2.3.2 Restoring from Backups

This will create a plain text csv file that you can import into lastpass or another password manager.

```
from lp_backup import Runner
run = Runner(path='/path/to/config/file.yml')
run.restore(backup_file_name, output_file_name)
```

It is crucial that you use the exact configuration file used to create the initial backup or your data might be garbled.

2.4 Api Documentation

This section contains more granular documentation of the classes and functions available to the api.

2.4.1 Automated Runner

class Runner(*path*, *, *filesystem=None*)

This class handles orchestration of downloading and storing the backup. Options are set in a yaml configuration file. There is an `example` you can use as a starting point.

Parameters

- **path** – absolute path to the file on the system or relative to the FS object supplied in the filesystem parameter
- **filesystem** (*keyword*) – a pyfilesystem2 FS object where the yaml config file is located.

backup()

Using the configuration from the file, create the backup.

restore(*infilename*, *new_file*)

Restore backup to a plain text csv file for uploading to password manager.

Parameters

- **infilename** – the name of the backup file
- **new_file** – the filename to save the data to

2.4.2 File I/O

read_backup(*backing_store_fs*, *infile*, *prefix=''*)

Read a backup file from some pyfilesystem.

Parameters

- **backing_store_fs** – The pyfilesystem object where the file is located
- **infile** – the name of the file
- **prefix** (*optional*) – the prefix before the filename

Returns raw file data

write_out_backup(*backing_store_fs*, *data*, *outfile*, *prefix=''*)

Write the backup data to its final location. A backing store is required and either a filepath to the packaged backup or the tmp filesystem is required.

Parameters

- **backing_store_fs** – a pyfilesystem2 object to be the final storage location of the backup. (should be *OSFS*, *S3FS*, *FTPFS*, etc.) Can be a single object or list of filesystem objects for copying to multiple backing stores.
- **data** – the byte stream that needs to be written to the file on the backing store fs.
- **outfile** – the name of the file to write out to.
- **prefix** (*optional*) – a parent directory for the files to be saved under. This is can be a good place to encode some information about the backup. A slash will be appended to the prefix to create a directory or pseudo-directory structure.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index