Andy Lewis
andy.lewis@ankesand.com
http://github.com/ankesand/aws_s3_uploader

AWS S3 File Uploader

Simple (Python) program(s) to automate ongoing periodic uploading of (dummy) datafiles from local Linux (Ubuntu) instance to Amazon Web Services (AWS) Simple Storage Solution (S3) bucket.

1. Part One: Scripting

1.1. Local files – dummy data generator

The following comprises a simple dummy datafile generator – to simulate data files being written to local directories:

- "base-dir" & 'locations' directories (including access permissions)

  (Dummy locations: "customer001", "customer002", "customer003")

bash

```
cd /home
```

```
[sudo] mkdir base-dir base-dir/customer001 base-dir/customer002 base-dir/customer003
```

```
[sudo] chmod -R 777 base-dir
```

- "aws_s3_file_uploader" directory (including access permissions)

  (Distinct from "base-dir")

bash

```
cd /home
```

```
[sudo] mkdir aws_s3_file_uploader
```

```
[sudo] chmod -R 777 aws_s3_file_uploader
```

- Python script to generate dummy datafile(s)

  Directory structure: "base-dir/[location]/YYYY/MM/DD/HH/mm-[datafile].dat"; one file per location

generate_dummy_data_files.py

(home/aws_s3_file_uploader/generate_dummy_data_files.py)

```python
import os
from datetime import datetime

# iterate through locations:
for location in os.listdir("/home/base-dir"):

    # treat only directories as locations:
    if os.path.isdir(f"/home/base-dir/{location}"):

        # set timestamp / directory structure variables:
        timestamp = datetime.now()
        YYYY = datetime.strftime(timestamp, "%Y")
        MM = datetime.strftime(timestamp, "%m")
        DD = datetime.strftime(timestamp, "%d")
        HH = datetime.strftime(timestamp, "%H")
```

```
        mm = datetime.strftime(timestamp, "%M")

        # check / update directory structure:
        if not os.path.isdir(f"/home/base-dir/{location}/{YYYY}"):
            os.mkdir(f"/home/base-dir/{location}/{YYYY}")
        if not os.path.isdir(f"/home/base-dir/{location}/{YYYY}/{MM}"):
            os.mkdir(f"/home/base-dir/{location}/{YYYY}/{MM}")
        if not os.path.isdir(f"/home/base-dir/{location}/{YYYY}/{MM}/{DD}"):
            os.mkdir(f"/home/base-dir/{location}/{YYYY}/{MM}/{DD}")
        if not os.path.isdir(f"/home/base-dir/{location}/{YYYY}/{MM}/{DD}"\
                             f"/{HH}"):
            os.mkdir(f"/home/base-dir/{location}/{YYYY}/{MM}/{DD}/{HH}")

        # set filepath / filename variables:
        filepath = f"/home/base-dir/{location}/{YYYY}/{MM}/{DD}/{HH}"
        filename = f"{mm}-datafile.dat"

        # write datafile to local directory:
        with open(f"{filepath}/{filename}", "x") as f:
            f.write(f"dummy data generated at "\
                    f"{datetime.strftime(timestamp, '%Y-%m-%d %H:%M:%S')}")

        # output (print to terminal / write to log):
        print (f"{datetime.strftime(timestamp, '%b %d %H:%M:%S')} "\
               f"location: {location} "\
               f"directory: {filepath} "\
               f"filename: {filename}"
               )
```

- Periodically scheduled (dummy) datafile generation – cron

  generate_dummy_data_files.py – execute every three minutes (all day, every day)

bash

```
cd /var/log
```

```
[sudo] mkdir aws_s3_file_uploader
```

```
[sudo] chmod -R 777 aws_s3_file_uploader
```

bash

```
crontab -e
```

crontab

```
*/3 * * * * python3 /home/aws_s3_file_uploader/generate_dummy_data_files.py
>> /var/log/aws_s3_file_uploader/generated.log
```

[check / verify:]

bash

```
service cron status
```

```
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset:
enabled)
   Active: active (running) [...]
```

bash

```
cat /var/log/aws_s3_file_uploader/generated.log
```

```
...
Oct 25 12:00:01 location: customer001 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 00-datafile.dat
Oct 25 12:00:01 location: customer002 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 00-datafile.dat
Oct 25 12:00:01 location: customer003 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 00-datafile.dat
Oct 25 12:03:01 location: customer001 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 03-datafile.dat
Oct 25 12:03:01 location: customer002 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 03-datafile.dat
Oct 25 12:03:01 location: customer003 directory: /home/base-
dir/customer001/2020/10/25/12 filename: 03-datafile.dat
...
```

bash

```
cd /home/base-dir/
```

```
tree
```

```
.
├── customer001
│   └── 2020
│       └── 10
│           └── 25
│               └── 12
│                   ├── 00-datafile.dat
│                   ├── 03-datafile.dat
...                 ...
├── customer002
│   └── 2020
│       └── 10
│           └── 25
│               └── 12
│                   ├── 00-datafile.dat
│                   ├── 03-datafile.dat
...                 ...
└── customer003
    └── 2020
        └── 10
            └── 25
                └── 12
                    ├── 00-datafile.dat
                    ├── 03-datafile.dat
                    ...
```

1.2. AWS S3 bucket – file uploader

The following comprises a simple program to upload (dummy) files from a given local directory to a given AWS S3 bucket and (subsequently) delete from local directory origin:

- Boto3 installation

  Amazon Web Services (AWS) Software Development Kit (SDK) for Python

  bash

```
pip install boto3
```

- Check environment variables*

  Amazon Web Services (AWS) credentials

  bash

```
printenv
```

```
...
AWS_ACCESS_KEY_ID=[____________________]**
AWS_SECRET_ACCESS_KEY=[________________________________________]**
...
```

  ** – assumes AWS credentials stored as environment variables*
  *** – redacted*

- Python program to upload (dummy) files from local directories to AWS S3 bucket and delete local file (upon successful upload)

  Directory structure: "base-dir/[location]/YYYY/MM/DD/HH/mm-[datafile].dat"
  Arguments: i) 'base-dir'; ii) target S3 bucket name;

  upload_files_to_aws_s3.py
  (home/aws_s3_file_uploader/upload_files_to_aws_s3.py)

```
#!/usr/bin/python3

import sys
import os
from datetime import datetime

arguments_count = len(sys.argv) - 1

# check two required arguments passed:
if arguments_count != 2:

    raise TypeError(f"upload_files_to_aws_s3 takes 2 positional arguments "\
                    f"({arguments_count} given)"
                    )

# proceed with two given arguments passed:
else:

    import boto3
    from botocore.exceptions import ClientError

    # create low-level client interface to AWS:
    s3_client = boto3.client("s3",
                             aws_access_key_id=\
```

```
                            os.environ["AWS_ACCESS_KEY_ID"],
                            aws_secret_access_key=\
                            os.environ["AWS_SECRET_KEY"]
                            )

# command line arguments passed:
base_dir_path = sys.argv[1]
bucket = sys.argv[2]

# directory (path) variables:
base_dir = os.path.basename(base_dir_path)
base_dir_start = base_dir_path.rstrip(base_dir)

# counters:
file_upload_count = 0
file_delete_count = 0

# define S3 bucket 'file-checker' function (file exists - True/False):
def file_in_s3_bucket(s3_client, bucket, key):
    try:
        s3_client.head_object(Bucket=bucket, Key=key)
    except ClientError:
        return False
    else:
        return True

# define custom Exception class (if local and S3 files conflict):
class ValidationError(Exception):
    def __init__(self, msg):
        self.msg = msg

# iterate through base directory children (i.e. 'locations'):
for dirname in os.listdir(base_dir_path):

    # check base directory child is a directory:
    if os.path.isdir(f"{base_dir_path}/{dirname}"):

        # iterate through base directory child files and (sub)folders:
        for root, dirs, files in os.walk(f"{base_dir_path}/{dirname}"):

            # iterate through files in (sub)folders
            # of each base directory child:
            for filename in files:

                # obtain local file path:
                filepath_local = os.path.join(root, filename)

                # set s3 upload file path:
                filepath_s3_upload = os.path.relpath(filepath_local,
                                                     base_dir_start
                                                     )

                # check file doesn't already exist in S3 bucket:
                if not file_in_s3_bucket(s3_client,
                                         bucket,
                                         filepath_s3_upload
                                         ):

                    # upload file to S3 bucket:
                    s3_client.upload_file(filepath_local,
                                          bucket,
                                          filepath_s3_upload
                                          )

                    timestamp_uploaded = datetime.now()
```

```
                    timestamp_log = datetime.\
                                    strftime(timestamp_uploaded,
                                             "%b %d %H:%M:%S"
                                             )

                    # output (print to terminal / write to log):
                    print(f"{timestamp_log} process: upload "\
                          f"bucket: {bucket} path: {filepath_s3_upload}"
                          )

                    file_upload_count += 1

                # check file uploaded to S3 bucket successfully
                # (for delete local file):
                if file_in_s3_bucket(s3_client,
                                     bucket,
                                     filepath_s3_upload
                                     ):

                    # get file info (e.g. byte size) of local file:
                    file_stat = os.stat(filepath_local)
                    # get file info (e.g. byte size) of S3 uploaded file:
                    file_head = s3_client.\
                                head_object(Bucket=bucket,
                                            Key=filepath_s3_upload
                                            )
                    # validate local file is same as S3 uploaded file:
                    if file_head["ContentLength"] == file_stat.st_size:

                        # delete local file:
                        try:
                            os.remove(filepath_local)

                            timestamp_deleted = datetime.now()
                            timestamp_log = datetime.\
                                            strftime(timestamp_deleted,
                                                     "%b %d %H:%M:%S"
                                                     )

                            # output (print to terminal / write to log):
                            print(f"{timestamp_log} "\
                                  f"process: delete "\
                                  f"path: {filepath_local}"
                                  )

                            file_delete_count += 1

                        except Exception as e:
                            print(e)

                    # should uploaded S3 file be diff. to local file:
                    else:
                       print(f"ValidationError: {filename} in S3 "\
                             f"bucket has different file size to "\
                             f"{filename} stored locally"
                             )

                else:
                    raise FileNotFoundError(f"S3 object "\
                                            f"'{filepath_s3_upload}' "\
                                            f"(bucket '{bucket}') "\
                                             "unavailable at call to "\
                                             "head_object()"
                                            )
```

```
    # output (print to terminal / write to log):
    if file_upload_count == 0 and file_delete_count == 0:

        timestamp_no_process = datetime.now()
        timestamp_log = datetime.\
                        strftime(timestamp_no_process,
                                 "%b %d %H:%M:%S"
                                 )
        print(f"{timestamp_log} "\
              f"process: null"
              )
```

- Periodically scheduled (dummy) local file upload to AWS S3 bucket and delete from local directory – cron

  upload_files_to_aws_s3.py – execute every twenty minutes (all day, every day – at minutes 5, 25 & 45)

bash

```
crontab -e
```

crontab

```
*/3 * * * * python3 /home/aws_s3_file_uploader/generate_dummy_data_files.py
>> /var/log/aws_s3_file_uploader/generated.log

13,28,43,58 * * * * /home/aws_s3_file_uploader/upload_files_to_aws_s3.py
/home/base-dir eu-s3-ankesand-001 >>
/var/log/aws_s3_file_uploader/uploaded.log
```

[check / verify:]

bash

```
service cron status
```

```
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset:
enabled)
   Active: active (running) [...]
```

bash

```
cat /var/log/aws_s3_file_uploader/uploaded.log
```

```
...
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer001/2020/10/25/12/00-datafile.dat
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/00-datafile.dat
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer001/2020/10/25/12/03-datafile.dat
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/03-datafile.dat
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer002/2020/10/25/12/00-datafile.dat
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/00-datafile.dat
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer002/2020/10/25/12/03-datafile.dat
```

```
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/03-datafile.dat
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer003/2020/10/25/12/00-datafile.dat
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/00-datafile.dat
Oct 25 12:13:01 process: upload bucket: eu-s3-ankesand-001 path: base-
dir/customer003/2020/10/25/12/03-datafile.dat
Oct 25 12:13:01 process: delete path: /home/base-dir/customer001/2020/10/
25/12/03-datafile.dat
...
```

bash

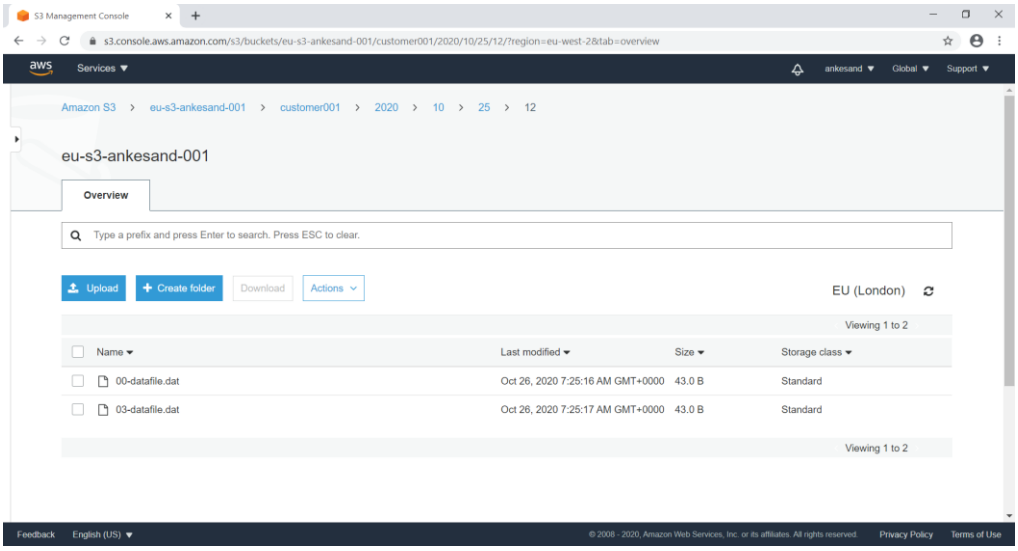
```
cd /home/base-dir/
```

```
tree
```

```
.
├── customer001
│   └── 2020
│       └── 10
│           └── 23
│               ├── 12
│               └── 13
│                   ...
│                   ├── 54-datafile.dat
│                   └── 57-datafile.dat
├── customer002
│   └── 2020
│       └── 10
│           └── 23
│               ├── 12
│               └── 13
│                   ...
│                   ├── 54-datafile.dat
│                   └── 57-datafile.dat
└── customer003
    └── 2020
        └── 10
            └── 23
                ├── 12
                └── 13
                    ...
                    ├── 54-datafile.dat
                    └── 57-datafile.dat
```

AWS

(https://s3.console.aws.amazon.com/s3/buckets/eu-s3-ankesand-001/customer001/2020/10/25/12/?region=eu-west-2&tab=overview)



2. Part Two: System Design

The process presented in Part One above provides a simple solution to demonstrate the principles of automated file transfer – it is not necessarily to be considered 'production ready'.

2.1. Potential risks & issues

Prior to deployment there are many further considerations, including:

- Integration
- Security ('InfoSec')
- Failover / backup
- Infrastructure / provisioning
- Performance (incl. monitoring / diagnostics)
  (etc.)

2.2. Proposal for improvements – management / maintenance of reliable service

Initial recommendations considered high priority would include:

- **Backup (local)**

  To maintain a continuous service (without loss of data) an additional mitigation would be to implement an appropriate backup policy / system.

  Local backups to onsite network attached storage (NAS) would add an additional source from which to potentially restore lost data.

  The volume of data, its importance and any associated service levels agreed as to attending to any system issues may inform the frequency and retention of these local backups.

- **Encryption (HTTPS)**

  To provide a more secure service any external connections ought to be encrypted.

  Use of HTTPS (SSL / TLS) communication between the local system(s) and the AWS S3 bucket(s) would provide such additional level of security.