Table of Contents

[**How the sync works (in your script)** 1](#_Toc206008683)

[**What that means in plain English** 1](#_Toc206008684)

[**How to prove it (for your reviewers)** 2](#_Toc206008685)

**How the sync works (in your script)**

1. **Discover everything dynamically**
   * list\_subdirs(BASE\_URL) finds all folders (pc/, pr/, sm/, …).
   * list\_files(dir\_url) grabs every file in each folder.  
     ➜ No hard‑coded filenames.
2. **Compare source vs S3 before uploading**
   * For each source file, the script issues a **HEAD** (or GET fallback) to read:
     + Last-Modified
     + Content-Length
   * It maps that URL to an S3 key via s3\_key\_for(...) (e.g., bls/pc/pc.series).
   * It checks S3 with head\_object:
     + If **ContentLength** and **Metadata['src-last-modified']** match the source, it **SKIPs** (no re‑upload).
     + Otherwise it downloads and uploads the object with **SSE‑S3** and stores:
       - src-last-modified
       - src-content-length
       - src-url
3. **Delete what the source no longer has (removals)**
   * Builds a set of **current source keys** (source\_map.keys()).
   * Lists **all S3 keys under your prefix** (list\_s3\_keys(...)).
   * Computes **orphans = S3 − Source** and calls delete\_keys(...) to remove them.  
     ➜ This keeps your mirror clean when BLS removes or renames files.
4. **Idempotent by design**
   * Running it again uploads **nothing** unless a file changed at the source.
   * If the source adds/removes files, your next run reflects that in S3.

**What that means in plain English**

* **Added at source?** → Appears in S3 next run.
* **Changed at source?** → Re‑uploaded (metadata/size change detected).
* **Deleted at source?** → Removed from S3 next run.
* **Unchanged?** → Skipped; no duplicate uploads.

**How to prove it (for your reviewers)**

* **Show SKIP logs:** run twice and point to “SKIP unchanged …” lines.
* **Simulate a delete:** manually delete a file in S3 under bls/, run the script → it will **recreate** it because it still exists at source.  
  Then simulate a **source removal** by temporarily excluding a folder (set BASE\_URL to a subfolder) and show the script **deletes orphans**; revert after.
* **Inspect metadata:** in S3 Console → select any object → **Metadata** tab shows src-last-modified, src-content-length.

Initial load from Laptop

$env:BLS\_S3\_BUCKET="dataquest-gov-bls-timeseries"

$env:BLS\_S3\_PREFIX="bls"

$env:BLS\_BASE\_URL="https://download.bls.gov/pub/time.series/"

$env:AWS\_REGION="us-east-1"

$env:BLS\_USER\_AGENT="dataquest-bls-sync/1.0 (+https://github.com/tracyanderson213/rearc-dataquest-bls; contact: tracy.anderson@outlook.com)"

$env:BLS\_RATE\_LIMIT\_RPS="3"

## Local Run

```powershell

python -m venv .venv

# Windows PowerShell

.\.venv\Scripts\Activate

pip install -r requirements.txt

# Set environment variables

$env:BLS\_S3\_BUCKET="dataquest-gov-bls-timeseries"

$env:BLS\_S3\_PREFIX="bls"

$env:BLS\_BASE\_URL="https://download.bls.gov/pub/time.series/"

$env:AWS\_REGION="us-east-1"

$env:BLS\_USER\_AGENT="dataquest-bls-sync/1.0 (+https://github.com/tracyanderson213/rearc-dataquest-bls; contact: tracy.anderson@outlook.com)"

$env:BLS\_CONTACT="tracy.anderson@outlook.com"

$env:BLS\_RATE\_LIMIT\_RPS="3"

python sync\_bls.py