**Full Media Backup & Compression Pipeline**

**1️⃣ Overview / Goal**

**Goal:**  
Automate backup, compression, and storage optimization for iPhone media while minimizing iCloud storage usage.

**Devices / Systems:**

* Primary iPhones (64GB) → taking photos/videos
* iCloud → current backup (over 500GB)
* NAS (1TB) → central storage
* Pixel 1 → unlimited Google Photos storage
* Proxmox Server / Home Assistant → runs automation scripts
* Spare iPhone 6 → optional for syncing iCloud if needed

**Objectives:**

1. Reduce iCloud storage usage.
2. Ensure all media is safely backed up on NAS and Pixel.
3. Maintain metadata, compression history, sync status.
4. Automate deletion from iCloud after sync/compression.
5. Keep modular, auditable logs and database history.

**2️⃣ Folder & DB Structure**

**Folder Layout (on NAS/Proxmox):**

/mnt/wd\_all\_pictures/

incoming/ # freshly downloaded from iCloud

processed/ # after Pixel sync + compression

delete\_pending/ # after folder movement, ready for iCloud deletion

logs/ # all logs per module

**Database (media.db)** – SQLite:

| **Column** | **Description** |
| --- | --- |
| id | Primary key |
| filename | File name |
| icloud\_id | Unique iCloud ID |
| created\_date | Original creation date |
| local\_path | Full path in NAS |
| status | downloaded / processed / error |
| synced\_google | yes/no |
| compressed | yes/no |
| initial\_size | File size at download |
| current\_size | Updated size after compression |
| last\_compressed | Timestamp of last compression |
| deleted\_icloud | yes/no |
| album\_moved | 0/1 |
| last\_updated | Auto timestamp |

**3️⃣ Scripts & Pipeline**

| **Step** | **Script** | **Function** |
| --- | --- | --- |
| 0 | sync\_icloud.py | Download iCloud media → insert metadata into DB. |
| 1 | sync\_pixel.py | Sync to Pixel / Google Photos → mark synced\_google='yes'. |
| 2 | compress\_media.py | Tiered compression based on file age → update current\_size & last\_compressed. |
| 3 | cleanup\_icloud.py | Move files to processed/ → delete\_pending/ → add to iCloud album → mark album\_moved=1. |
| 4 | delete\_icloud.py | Delete files from delete\_pending/ and iCloud album → mark deleted\_icloud='yes'. |
| 5 | Optional | report\_db.py → generate summary / CSV of DB for auditing. |

**Flow:**

iPhone → iCloud → NAS/incoming

│

▼

sync\_pixel.py

│

▼

compress\_media.py (tiered)

│

▼

cleanup\_icloud.py

│

▼

delete\_icloud.py

**4️⃣ Master / Main Script**

You can create a run\_pipeline.py to execute all modules sequentially:

import subprocess

import logging

LOG\_FILE = "logs/main\_pipeline.log"

logging.basicConfig(level=logging.INFO, format="%(asctime)s [%(levelname)s] %(message)s",

handlers=[logging.FileHandler(LOG\_FILE), logging.StreamHandler()])

logger = logging.getLogger("main\_pipeline")

modules = [

"sync\_icloud.py",

"sync\_pixel.py",

"compress\_media.py",

"cleanup\_icloud.py",

"delete\_icloud.py"

]

def run\_module(module):

logger.info(f"--- Running {module} ---")

result = subprocess.run(["python3", module])

if result.returncode != 0:

logger.error(f"{module} failed! Return code: {result.returncode}")

else:

logger.info(f"{module} completed successfully.")

def main():

for mod in modules:

run\_module(mod)

logger.info("✅ Full pipeline complete.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

This gives a **single entry point** for the entire workflow and keeps each module independent.

**5️⃣ AI-Friendly Prompt / Description**

**Prompt for AI Assistance:**  
“I am building a modular media backup pipeline for iPhones using iCloud, NAS, and a Pixel phone.  
The pipeline includes: downloading iCloud media, syncing to Pixel, tiered compression by file age, folder organization, and automated deletion from iCloud.  
Each step updates a SQLite DB (media.db) with metadata, compression size, sync status, album movement, and deletion flags.  
Logs are kept separate for each module (logs/).  
Scripts are modular (sync\_icloud.py, sync\_pixel.py, compress\_media.py, cleanup\_icloud.py, delete\_icloud.py) and a master script (run\_pipeline.py) executes them sequentially.  
Please provide recommendations, improvements, or code for missing steps while ensuring database integrity, compression strategy, and automated deletion are safely implemented.”

**6️⃣ Recommendations / Next Steps**

1. **Automation & Scheduling**
   * Use cron or systemd timers on Proxmox / Home Assistant to run run\_pipeline.py nightly or hourly.
   * Ensure each script logs success/failure independently.
2. **Monitoring & Dashboard**
   * Connect **sqlite-web** or **DB Browser** for visualizing media.db.
   * Optionally, integrate Home Assistant dashboard for storage usage, compression stats, and sync status.
3. **Error Handling & Recovery**
   * If any module fails, run\_pipeline.py logs error; can send alert (email/notification).
   * Retry strategy for iCloud / Pixel network errors.
4. **Tiered Compression**
   * Already implemented; consider adding adjustable parameters (light/medium/aggressive).
   * Can also implement **archival tier**: older than X years → archive to cold storage, delete from main NAS.
5. **Testing**
   * Run scripts manually first on small batch of files to ensure DB updates, compression, folder moves, and deletion all work correctly.
   * Keep backups of NAS before testing deletions.
6. **Security & 2FA**
   * Store iCloud credentials securely (environment variables or keyring).
   * Handle 2FA tokens safely to avoid repeated prompts.
7. **Extensions for AI / Automation**
   * AI can recommend: automated compression settings, storage alerts, backup verification, and even smart deletion rules based on file age/duplicates.
   * AI can also help generate reports from media.db in CSV/Excel/PDF formats for auditing.

**✅ Summary**

* **All scripts are modular** → safe, auditable, extendable.
* **Database tracks everything** → compression, sync, deletion.
* **Master script** allows full sequential execution.
* **Folder structure + logs** maintain clarity and reduce human errors.
* **AI can take over** optimization, report generation, or advanced automation