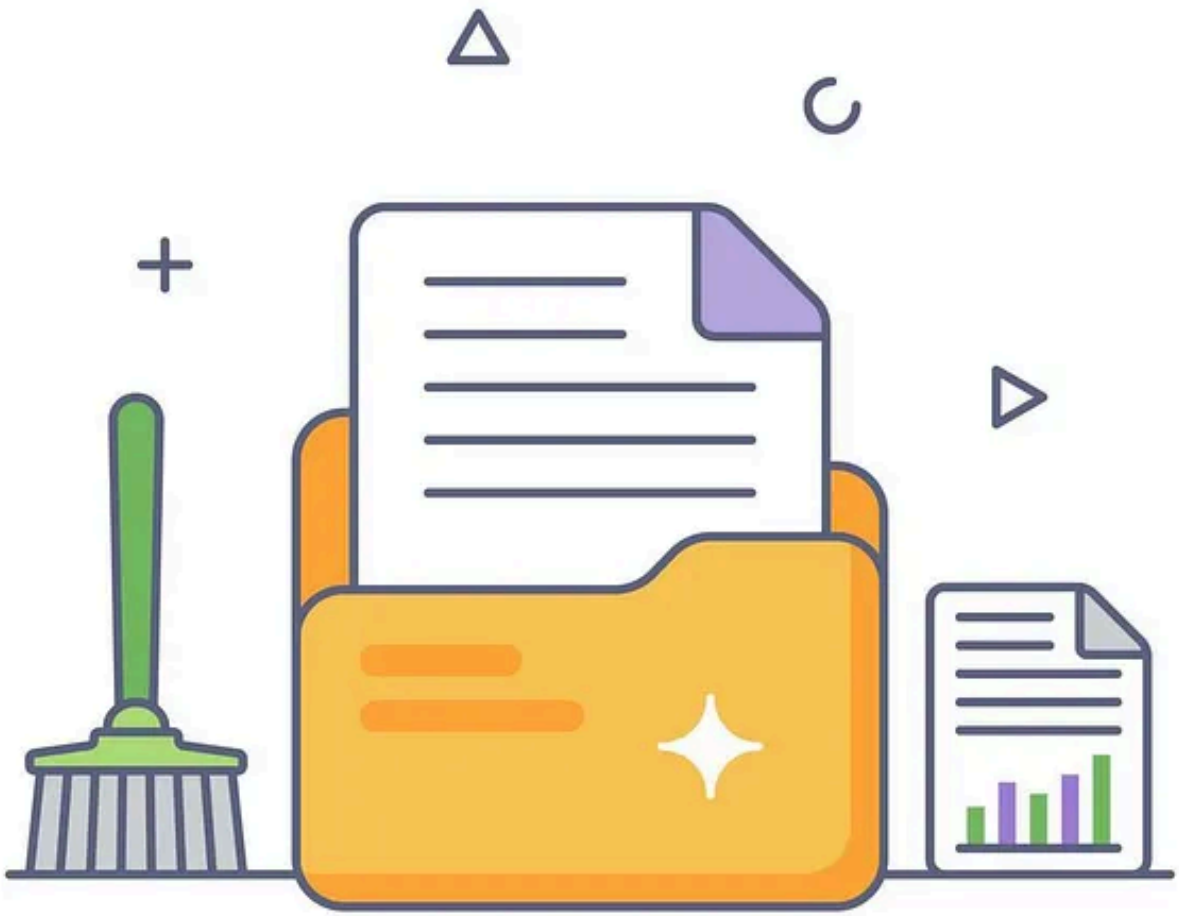


Data Cleaning

Cleaning AWS S3: How I Saved Big Bucks with a Little Housekeeping

Anurag Sharma 5 min read · Jun 2, 2025

Picture this: you're a data engineer in a fast-paced analytics company. Your team is churning through gigabytes and terabytes of data daily, building reporting pipelines and analytics dashboards. But there's a silent problem in the process: your AWS S3 bill. The storage costs are piling up faster than you can think.

As a data engineer surrounded by 30–40 data engineering colleagues, I noticed something: our S3 buckets were turning into data dump yards. Old files, forgotten test datasets, and "just-in-case" backups were taking up storage. The solution? A little housekeeping in S3 that saved us serious cash.

Here is how I did it, and how you can too, with a sprinkle of Python, some analytics.

The Problem: S3 Buckets as Cloud Clutter Collectors
In an Analytics-Driven company, speed is king. We are talking about aggressive deadlines, relentless reporting, and analytics pipelines that need to be up and running yesterday. But with great data comes great responsibility — and a lot of junk. Our team was generating massive datasets daily: raw files, intermediate outputs, test data, and those mysterious "customer_temp_data_experiment_v42.csv" files no one dared to touch.

These files were bloating our S3 buckets, and with S3 pricing based on storage volume, every unused gigabyte was costing the company a significant amount. The worst part? No one was cleaning up.

Step 1: The Cleanup Mission with Python and Boto3

To tackle this, I needed to understand what was in our S3 buckets.

Were we retaining test files that were six months old?

Were there duplicates?

And most importantly, how much was this digital clutter costing us?

Enter Python and Boto3, I wrote a Python script to scan our S3 buckets and gather info. The goal was simple: create a summary of every file with its name, size, and last modified date.

Here's a simplified version of the script I used:

```
import boto3
import pandas as pd

s3 = boto3.client('s3')
bucket_name = 'my-data-bucket'

# List objects in the bucket
objects = s3.list_objects_v2(Bucket=bucket_name).get('Contents', [])
data = [{'file_name': obj['Key'], 'size_mb': obj['Size'] / 1024**2, 'last_modified':
obj['LastModified']}
for obj in objects]
```

```
# Create a DataFrame
df = pd.DataFrame(data)
df.to_csv('s3_files_data_information.csv', index=False)
```

Let me break down the script step by step in short sentences:

Import Libraries:

- The script imports boto3.
- It also imports pandas as pd.
- boto3 is for AWS services.
- pandas is for data manipulation.

Set Up S3 Client:

- Creates an S3 client using boto3.client('s3').
- Stores it in the variable s3.
- This allows interaction with AWS S3.

Define Bucket Name:

- Sets bucket_name to 'my-data-bucket'.

- This is the S3 bucket to access.

List Objects in Bucket:

- Calls s3.list_objects_v2(Bucket=bucket_name).
- Retrieves objects with .get('Contents', []).
- Stores objects in objects.
- Uses empty list [] if no objects exist.

Extract Object Details:

- Creates a list data.
- Loops through each object in objects.
- For each object, extracts:
- File name as obj['Key'].
- Size in MB using obj['Size'] / 1024**2.
- Last modified date as obj['LastModified'].
- Stores these as dictionaries in data.

Create DataFrame:

- Uses pd.DataFrame(data) to create a DataFrame.
- Stores it in df.
- The DataFrame contains file names, sizes, and last modified dates.

Save to CSV:

- Saves df to a CSV file named 's3_files_data_information.csv'.
- Uses index=False to exclude row indices in the CSV.

This script catalogs every file and spits out a CSV with all the details. Suddenly, I had a dataset showing exactly where the clutter was hiding.

Step 2: Turning Data into Visualization

Raw data is great, but it does not show where "cost savings" is to Data Team Management or DevOps. To make my case, using Python's visualization libraries — Matplotlib and Seaborn — I created some charts to show the scale of the problem and the potential savings.

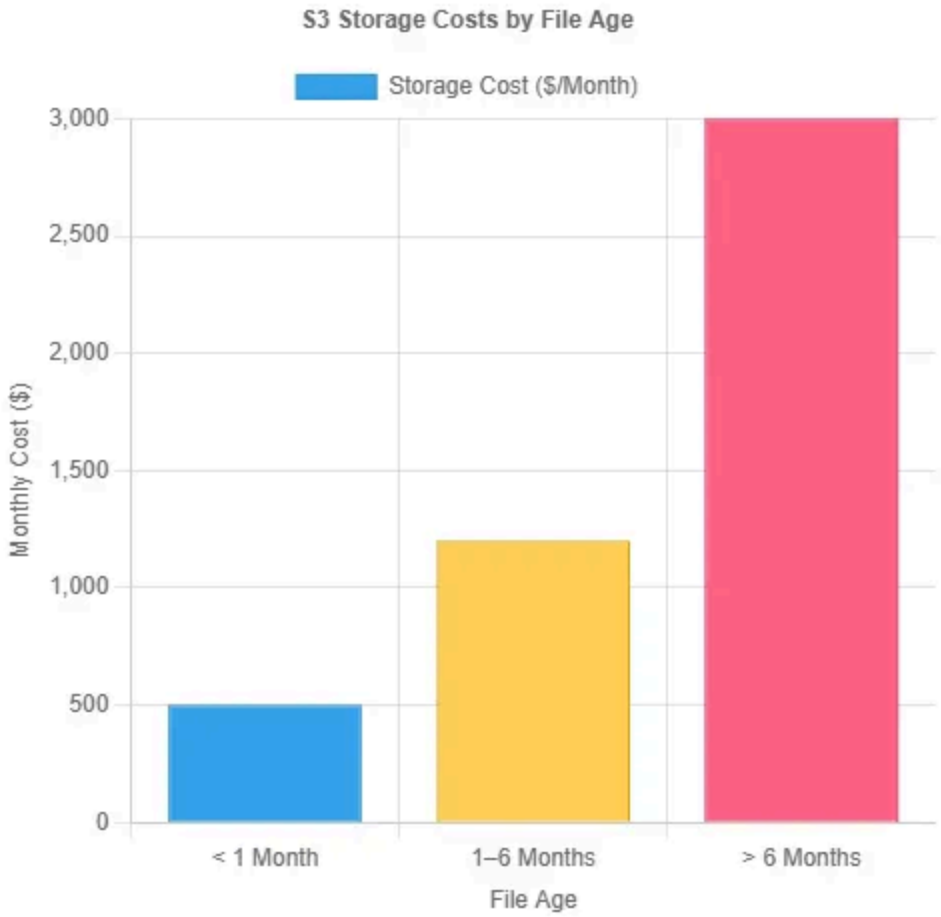
Check what I did:

• **Grouped files by age:** I bucketed files into categories like "less than 1 month old," "1–6 months old," and "older than 6 months."

• **Calculated storage costs:** Using AWS's S3 pricing (roughly $0.023 per GB per month for Standard storage in us-east-1), I estimated the monthly cost of each file category.

• **Visualized the savings:** I created a bar chart showing how much we could save by deleting files older than 1 month or 6 months.

The chart I presented to the team, which got everyone's attention faster:



Step 3: The Cleanup Plan

• **Delete files older than 6 months:** These were mostly test data and outdated backups. We confirmed with the Data Manager/Data Architect that they were not needed.

• **Archive files aged 1–6 months:** Move them to S3 Glacier for cheaper storage (at ~$0.004 per GB/month).

• **Automate the process:** Set up a recurring Python job using AWS Lambda to scan buckets monthly and flag files for deletion or archival.

To make the case, forecast the savings:

• Deleting 6-month-old files saved ~$3,000/month.

• Archiving 1–6-month-old files to Glacier saved ~$800/month.

• Total annual savings? A cool $46,800.

Step 4: Making It Stick

The cleanup was not a one-and-done deal. To keep our S3 buckets clean for a long duration, we:

• **Set lifecycle policies:** Configured S3 to automatically move files to Glacier after 90 days and delete them after 180 days.

• **Monitored savings:** Added a monthly report to track storage costs and celebrate our wins.

Takeaways for Your Own S3 Spring Cleaning

Want to save some cash on your AWS bill?

Here is the playbook:

1. Audit your buckets: Use Boto3 to catalog what's in your S3 storage.

2. Visualize the impact: Turn data into compelling charts to rally support.

3. Clean strategically: Delete what's safe, archive what's needed, and automate the rest.

4. Spread the word: Get your team on board to prevent future clutter.

By treating your S3 buckets like a well-organized closet, you will save money *and* sleep better knowing your data is tidy.

Data Analysis | Amazon | Data Engineer | Data Cleaning | Data Analytics

Written by Anurag Sharma

82 followers · 3 following

Edit profile

Data Engineering Specialist with 10+ exp. Passionate about optimizing pipelines, data lineage, and Spark performance and sharing insights to empower data pros!