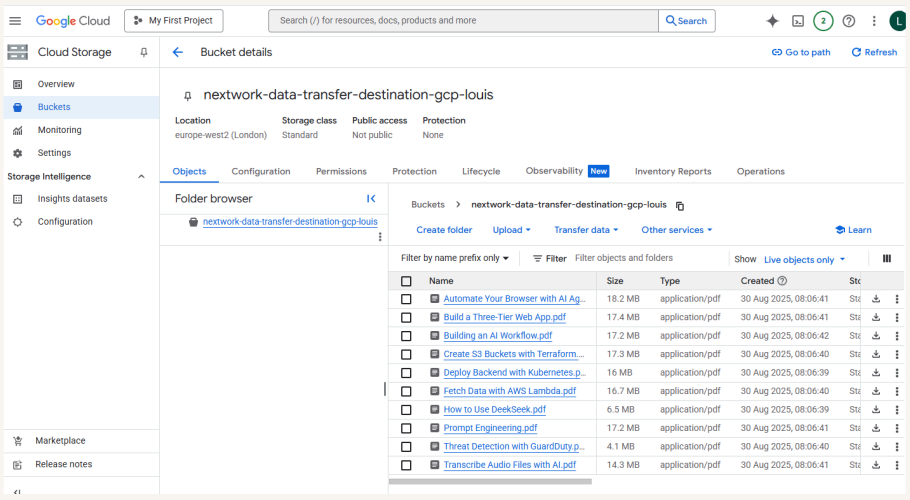
nextwork.org

# Multi-Cloud Data Transfer with AWS and GCP

L Louis Moyo



**Louis Moyo**
NextWork Student



# Introducing Today's Project!

In this project, I will demonstrate how to back up files from an AWS S3 bucket into Google Cloud Storage using the GCP Storage Transfer Service. I'm doing this project to learn how to integrate multiple cloud providers, configure IAM roles for cross-cloud access, and implement reliable, real-time data transfers. This will help me build skills in multi-cloud engineering, which is essential for designing resilient, flexible, and cost-efficient cloud architectures.

## Tools and concepts

Services I used were Amazon S3, AWS IAM, Google Cloud Storage, and Storage Transfer Service. Key concepts I learnt include identity federation via accounts.google.com with a subjectId, least-privilege IAM, region and storage class choices, and manifest-based selective transfers.

## Project reflection

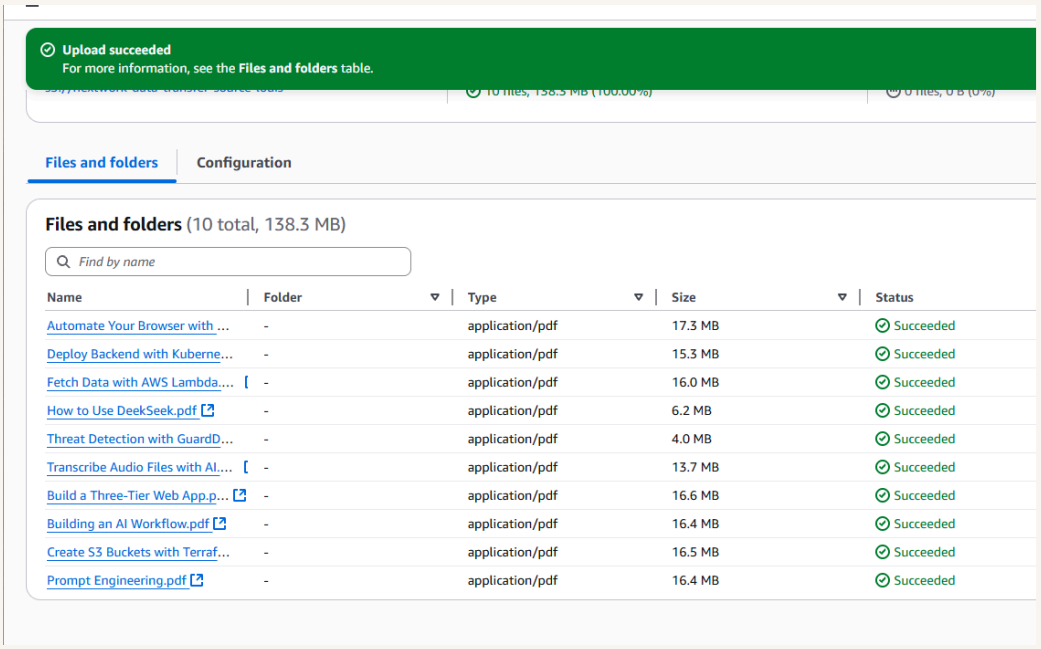
This project took me approximately 1.5-2 hours. The most challenging part was nailing the IAM trust policy and manifest format. It was most rewarding to see files land in GCS and understand secure cross-cloud access without long-lived keys.

**Louis Moyo**
NextWork Student



# Setting up Data in S3

I started this project by setting up a new S3 bucket in AWS, which I named nextwork-data-transfer-source-louis to ensure it had a unique identifier. This bucket will act as the source location for the data we'll transfer into Google Cloud Storage later in the project. I then uploaded several files into the bucket to serve as test data. By doing this, I now have a working S3 environment with files ready for multi-cloud backup, making it possible to validate the transfer process end to end.



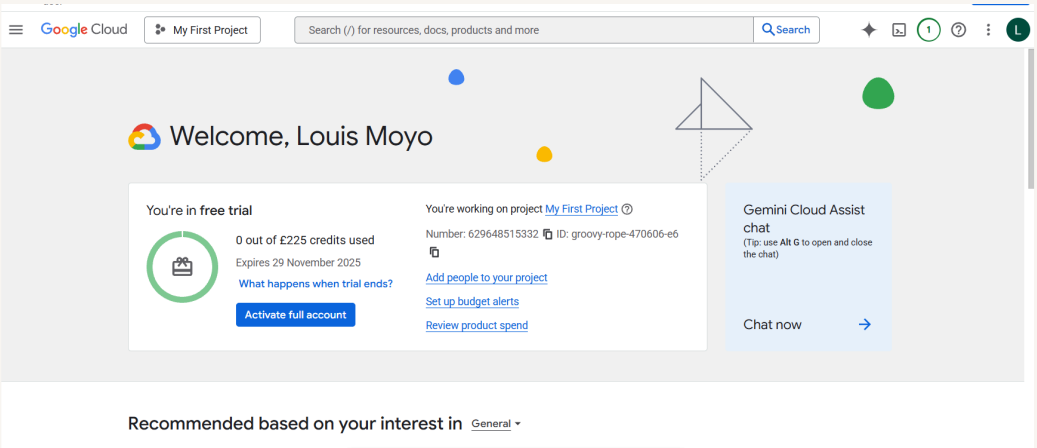
**Louis Moyo**
NextWork Student



# Setting up GCP

Google Cloud Platform is Google's suite of cloud services that provides infrastructure, storage, databases, machine learning, and networking tools, similar to AWS and Azure. In this project, I set up a GCP account to gain access to Cloud Storage, which will serve as the backup destination for my AWS S3 data, and to use the Storage Transfer Service, which will manage the actual transfer between AWS and GCP.

GCP's free tier includes access to more than 25 services such as Cloud Storage for data, Compute Engine for virtual machines, BigQuery for analytics, and Cloud Run for containerised apps - all up to their monthly free limits. On top of that, when I create a new account, I also get $300 in trial credit that I can spend on any GCP service that isn't fully covered under the free tier. This credit lasts for the first 90 days of my account, giving me the ability to explore, test, and complete projects like this one without incurring costs.



L
**Louis Moyo**
NextWork Student


# Storage Transfer

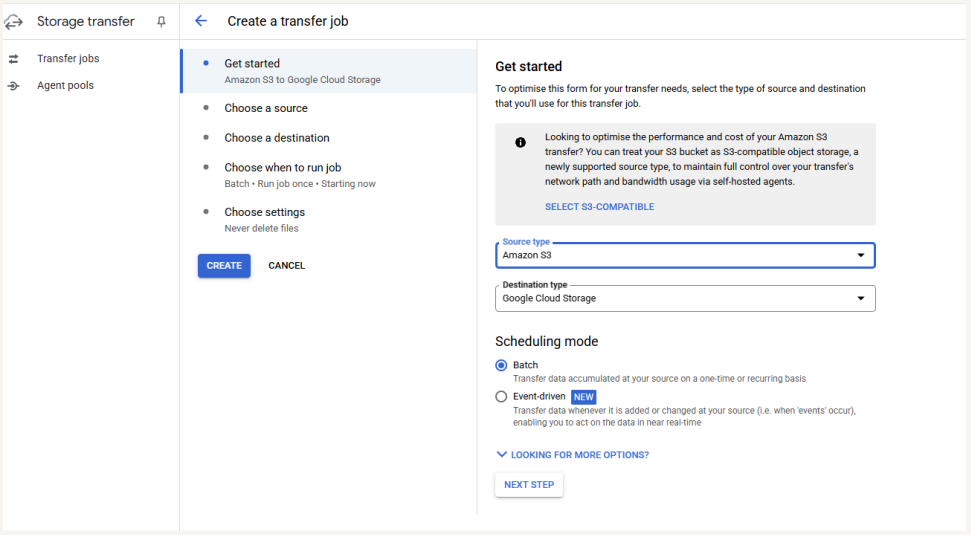
Data transfers are important for ensuring that critical files are securely backed up and available across different platforms, reducing the risk of data loss and downtime. By creating a transfer job, we automate the process of moving data from our AWS S3 bucket into GCP Cloud Storage, making it reliable and repeatable. Using multiple cloud providers for the transfer has benefits such as higher resilience (if one provider has an outage, the other still holds the data), cost optimisation (taking advantage of each provider's pricing), and greater flexibility (leveraging the unique strengths of AWS and GCP for different workloads).

The transfer is set up using Storage Transfer Service, which is Google Cloud's managed solution for moving data into and out of Cloud Storage. It simplifies cross-cloud transfers by handling the heavy lifting - authentication, scheduling, retries, and data validation - without requiring us to manually download and re-upload files. We need this service because AWS and GCP don't provide native ways to connect directly to each other, and building a custom transfer pipeline would be far more complex, time-consuming, and error-prone.

There are two different types of transfers you could set up: batch transfers and event-driven transfers. The difference is that batch transfers are one-time or scheduled jobs - they move data either immediately or at recurring times, which is useful for large migrations or periodic backups. Event-driven transfers, on the other hand, automatically trigger whenever a new file is created or an existing file is updated in the source bucket, making them ideal for real-time synchronisation. Batch is simpler to set up, while event-driven ensures continuous updates without manual intervention.

**Louis Moyo**
NextWork Student





**Louis Moyo**
NextWork Student

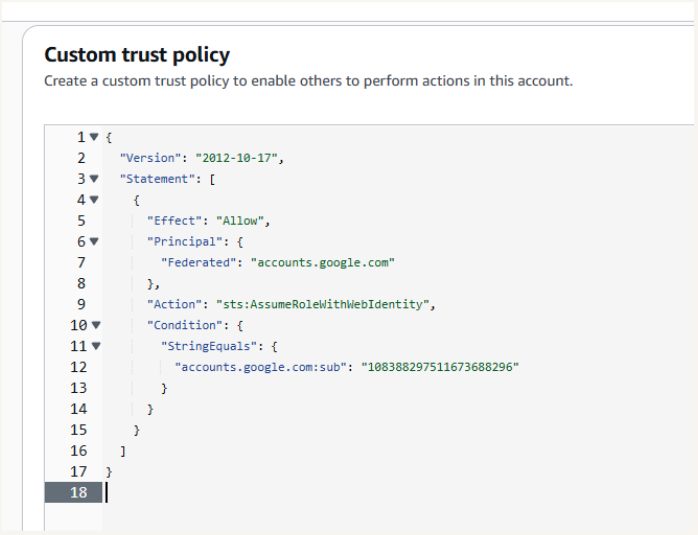


# Granting GCP Access to AWS

To connect AWS and GCP, I'm using identity federation, which works by creating a temporary trust relationship between the two cloud providers. Instead of sharing permanent AWS access keys with GCP, I create an IAM role in AWS that grants S3 permissions, and then allow GCP's Storage Transfer Service to assume that role. When GCP requests access, AWS issues short-lived credentials (usually valid for 15–60 minutes) to perform the transfer. This is more secure than static keys because the credentials expire automatically, reducing the risk of misuse or compromise while still enabling cross-cloud access.

I created a custom IAM role for Google's Storage Transfer Service because it lets me tightly control who can assume the role > only the specific GCP service account for my project. The custom trust policy enforces this by matching the service account's subject ID, so requests from any other Google project are denied. I attached AmazonS3ReadOnlyAccess to follow least-privilege > GCP can list and read objects but cannot change or delete anything. Using this role with web identity federation gives GCP temporary credentials, which expire automatically and avoid risky long-lived access keys.

Within the IAM role, I needed a custom trust policy to control exactly who can assume it > only Storage Transfer from my GCP project. The policy identifies GCP based on a subject ID, which is the unique identifier of the Storage Transfer service account that Google auto-creates for my project. By matching accounts.google.com:sub to that subject ID, AWS will issue temporary credentials only when the caller is that exact service account > blocking access from any other Google project or identity. This enforces least privilege, prevents key sharing, and keeps the cross-cloud link both precise and secure.


Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Federated": "accounts.google.com"
},
"Action": "sts:AssumeRoleWithWebIdentity",
"Condition": {
"StringEquals": {
"accounts.google.com:sub": "108388297511673688296"
}
}
}
]
}

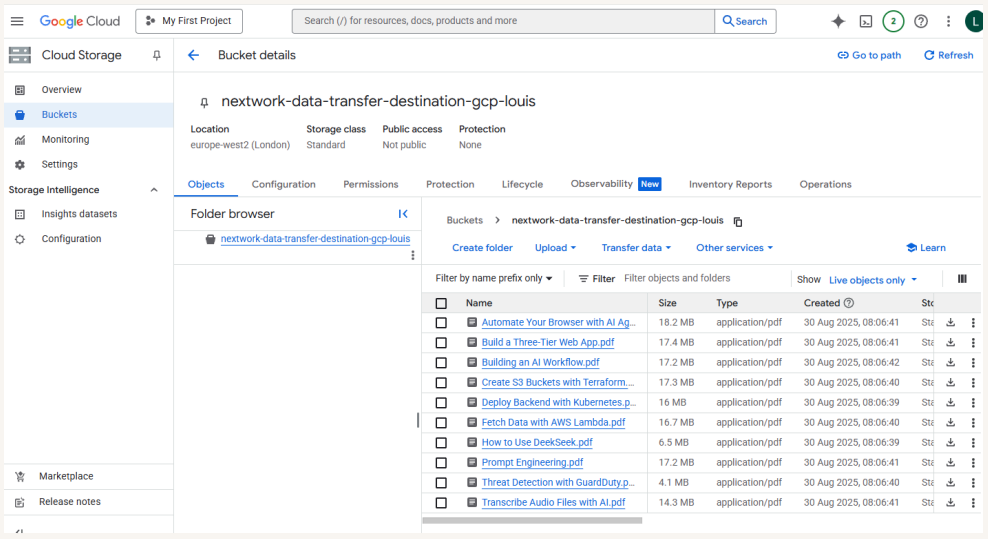
**Louis Moyo**
NextWork Student



# Transferring from S3 to GCS!

To set up my destination GCS bucket, I needed to set up its region, which means choosing the physical Google Cloud location where my data is stored and served to optimise latency, comply with any data residency needs, and reduce cross-region egress, and its storage class, which means selecting the performance and cost tier for the data - I used Standard for frequent access and low latency, though Nearline, Coldline, or Archive would cut storage cost if I accessed the data less often.

I verified my data transfer was successful by watching the Storage Transfer job until the Operation status changed to Success, then opening my destination GCS bucket, hitting Refresh, and confirming the S3 files appeared.



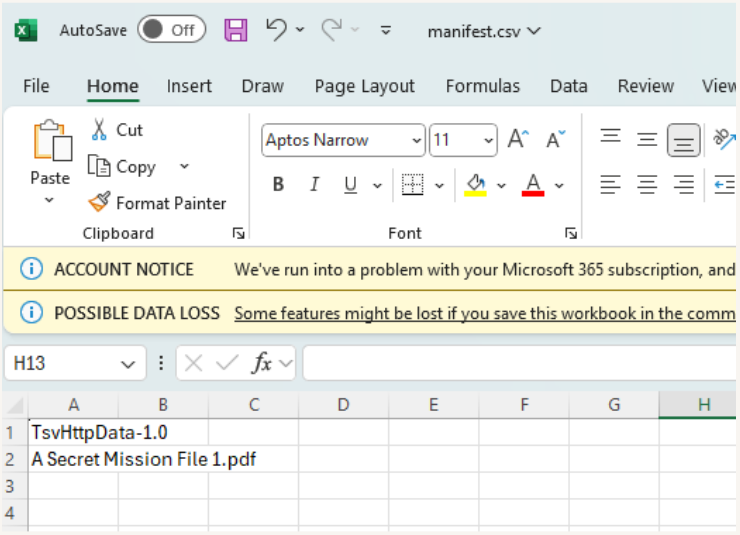
**Louis Moyo**
NextWork Student



# Transfer with a Manifest

In a project extension, I created manifest.csv with header TsvHttpData-1.0, listed exact S3 object keys (one per line), enabled Use a manifest in Storage Transfer, and ran the job. A manifest works by whitelisting only those objects for transfer > ideal for partial moves, retries, and tightly scoped migrations.

I verified my data transfer was successful by watching the Storage Transfer job until the Operation status changed to Success, then opening my destination GCS bucket, hitting Refresh, and confirming the S3 files appeared.



# The place to learn & showcase your skills

Check out nextwork.org for more projects