



MULTI-CLOUD FILE STORAGE SYSTEM

*Efficient.
Secure. Green.*



Talent Exhibition Project: Multi-Cloud File Storage



ARISTIDE KATAGARUKA

BTS CLOUD COMPUTING
LYCÉE GUILLAME KROLL

SUBMISSION DATE: JULY 2ND 2025

Contents

Certification and Declaration of Honor	3
1. Overview.....	4
2. Features	4
3. Project Structure	5
4. Development Environment Setup (3 hours, June 3, 2025)	6
Objective:	6
Steps	6
Troubleshooting	10
5. Cloud Account Setup (2 hours, June 3, 2025).....	10
Objective:	10
Steps	10
Troubleshooting	23
6. Web App Frontend Development (8 hours, June 5, 2025)	23
Objective:	23
Steps	23
Troubleshooting	31
7. Cloud Storage API Integration (10 hours, June 7, 2025)	31
Objective:	31
Steps	31
Troubleshooting	37
8. Application Containerization (6 hours, June 9, 2025)	37
Objective:	37
Steps	37
Troubleshooting	40
9. Multi-Cloud Simulation with Load Balancing (6 hours, June 10, 2025).....	40
Objective:	40
Steps	40
Troubleshooting	42
10. Sustainability and Alerts Implementation (8 hours, June 12, 2025)	42
Objective:	42
Steps	42
Troubleshooting	58

11. Testing File Operations and Demo Scenarios (6 hours, June 15, 2025).....	59
Objective:	59
Steps	59
Troubleshooting	60
12. Performance Optimization	61
Objective:	61
Steps	61
Troubleshooting	63
13. Documentation and Presentation (9 hours, June 17, 2025).....	64
Objective:	64
Steps	64
14. Technical Architecture	67
15. Key Artifacts	68
Flask Backend (backend/app.py)	68
16. Video of the project overview	70
17. Project Management.....	70
18. Conclusion	71
19. Learning Resources	71
19.Resource Links for Multi-Cloud Storage System Project.....	71
20. Table of Figures	73

Certification and Declaration of Honor

I, the undersigned, hereby declare that the work contained in this project report titled "**Multi-Cloud File Storage System**" is my own and has been completed as part of the PROMA and TBUCO course requirements for the BTS Cloud Computing program at Lycée Guillaume Kroll.

I affirm that:

1. I have conducted the project ethically and honestly, adhering to the principles of integrity and academic conduct.
2. All external references, sources, and contributions by others have been appropriately acknowledged in the report.
3. The report has not been plagiarized, and no part of this work has been submitted previously for any other assignment, certification, or purpose.

Furthermore, I acknowledge that failure to comply with the rules of academic honesty may result in disqualification of this project and the revocation of grades awarded for the PROMA and TBUCO course.

Signatures of Team Members:

- Aristide KATAGARUKA



Date: 2/07/2025

1. Overview

This project implements a multi-cloud file storage system leveraging free-tier services from Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). The system is designed to demonstrate cost optimization and sustainability for businesses while serving as a zero-budget prototype for a talent exhibition. It includes a React-based frontend, a Flask backend, Dockerized environments, and Nginx for load balancing, with features like file uploads, sustainability metrics, and API usage alerts.

2. Features

- **File Operations:** Upload/download files across AWS S3, Azure Blob Storage, and GCP Cloud Storage.
 - **Batch Upload:** Support for uploading multiple files simultaneously.
 - **Demo Scenarios:** Mock object (backup), block (VM storage), and file (sharing) storage use cases.
 - **Dashboard:** Displays storage usage, simulated cost savings, API limit alerts, and carbon footprint.
 - **Provider Comparison:** Table comparing pricing, free-tier limits, and egress fees.
 - **Storage Info:** Technical details and use cases for object, block, and file storage.
 - **Sustainability:** Energy estimation (0.01 kWh/GB), lifecycle policies (Glacier, Archive, Coldline).
 - **Alerts:** Notifications for API usage limits (80% of free-tier thresholds).
 - **Deployment:** Containerized with Docker, orchestrated with Docker Compose, load-balanced with Nginx.
-

3. Project Structure

Multi-Cloud-Storage/

— backend/		
— app.py	# Flask backend main application	
— file_upload.py	# Cloud storage upload logic	
— energy_estimator.py	# Energy usage calculation	
— requirements.txt	# Python dependencies	
— temp/	# Temporary file storage	
— key.json	# GCP service account key	
— docs/		
— business-case.md	# Business case documentation	
— technical-docs.md	# Technical documentation	
— user-guide.md	# User guide for setup and usage	
— storage-app/		
— package.json	# React app config and dependencies	
— Dockerfile	# Dockerfile for frontend	
— tailwind.config.js	# Tailwind CSS configuration	
— src/		
— components/		
— FileUpload.js	# File upload UI and logic	
— StorageDashboard.js	# Dashboard with usage, savings,	
— StorageInfo.js	# Info and use cases for storage	
— ProviderComparison.js	# Cloud provider comparison table	
— pages/		
— Home.js	# Main page composing dashboard and	
— services/		
— api.js	# API call utilities	
— App.js	# App entry point	
— index.js	# React entry point	
— index.css	# Tailwind CSS styles	
— docker-compose.yml	# Orchestration for backend, frontend,	
— nginx.conf	# Nginx load balancer configuration	
— .gitignore	# Git ignore rules	

4. Development Environment Setup (3 hours, June 3, 2025)

Objective: Configure development tools and version control.

Steps

1. Install Visual Studio Code (1 hour):

- Download VS Code from code.visualstudio.com.
- Install extensions (Ctrl+Shift+X):
 - **Python:** Linting, debugging, IntelliSense.
 - **JavaScript (ES6) snippets:** React development.
 - **Docker:** Dockerfile and Docker Compose support.
 - **GitLens:** Git commit history and blame.
- Configure Python interpreter: Ctrl+Shift+P > “Python: Select Interpreter” > Python 3.9+.

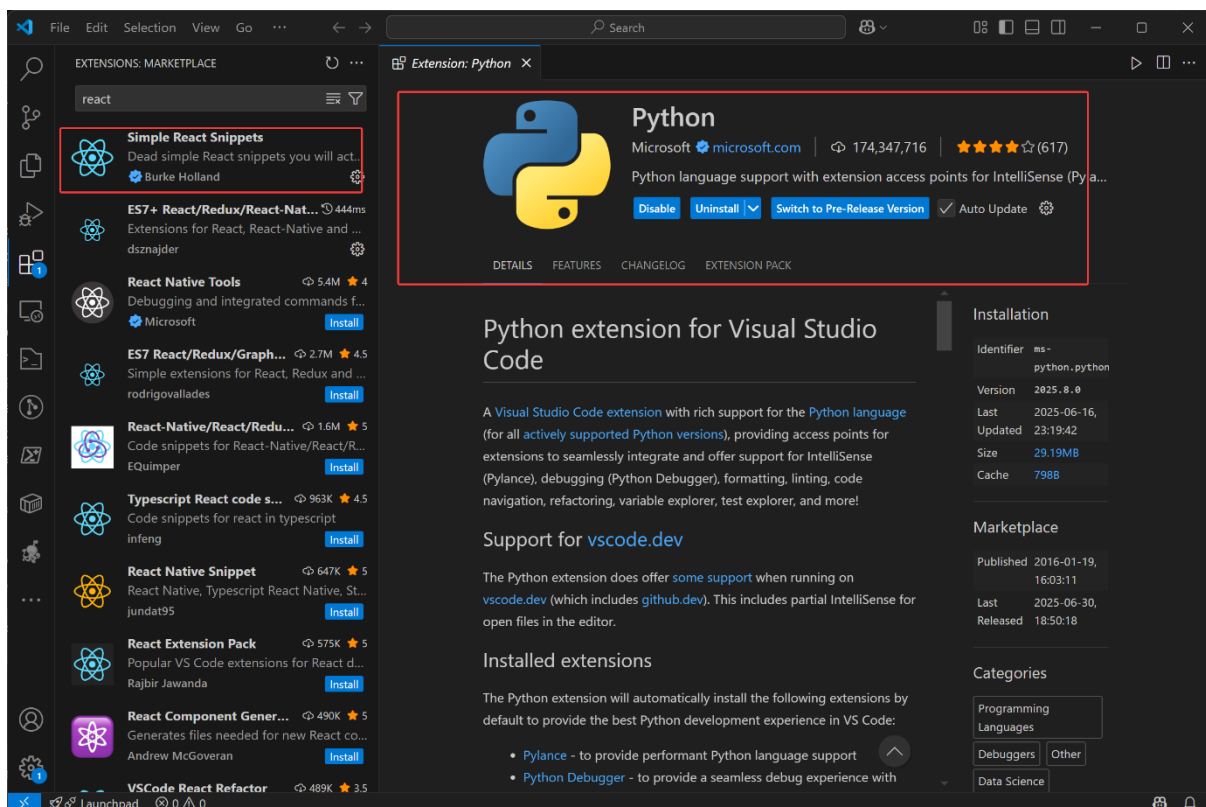


Figure 1: VS Code Extensions panel with installed extensions, Python file open

- **Why:** VS Code provides a lightweight, extensible IDE for Python and React.

2. Install Python and Node.js

- Download Python 3.9+ from python.org (e.g., 3.9.13).

- Windows: Check “Add Python to PATH.”
- macOS/Linux: Use brew install python or sudo apt install python3.
- Download Node.js 16+ (LTS) from nodejs.org.
- Verify:

```
bash

python --version # Check Python version
node --version   # Check Node.js version
npm --version    # Check npm version
```

Figure 2: Verifying versions

- Expected: Python 3.9.x, Node.js v16.x.x, npm 8.x.x.

```
PS C:\Users\akata\Desktop\Talent_Exhibition_Project_Aristide> python --version # Check Python version
Python 3.13.5
PS C:\Users\akata\Desktop\Talent_Exhibition_Project_Aristide> node --version # Check Node.js version
v22.16.0
PS C:\Users\akata\Desktop\Talent_Exhibition_Project_Aristide> npm --version # Check npm version
10.9.2
```

Figure 3: Terminal showing version outputs

- **Why:** Python for Flask backend, Node.js for React frontend.

3. Set Up React Project:

- **Create React app:**
 - npx create-react-app storage-app # Initialize React project
 - cd storage-app # Navigate to project directory
 - npm install tailwindcss postcss autoprefixer axios # Install dependencies
 - npx tailwindcss init -p # Initialize Tailwind CSS with PostCSS
- **Update storage-app/tailwind.config.js:**

```
module.exports = { // Export Tailwind configuration
  content: ["./src/**/*.{js,jsx,ts,tsx}"], // Specify files for Tailwind to scan
  theme: { extend: {} }, // Extend Tailwind theme (empty for now)
  plugins: [], // Add Tailwind plugins (none for now)
};
```
- **Update storage-app/src/index.css:**

```
@tailwind base; /* Apply Tailwind base styles */
@tailwind components; /* Apply Tailwind component styles */
@tailwind utilities; /* Apply Tailwind utility classes */
```

- Add proxy to storage-app/package.json for backend API calls:

```
"proxy": "http://localhost:5000" // Proxy API requests to
Flask backend
```

- Start app:

```
npm start # Start development server
```



Figure 4: Browser showing default React app with Tailwind styling

- Verify at <http://localhost:3000>.
- **Why:** React for dynamic UI, Tailwind for responsive styling, axios for API calls.

4. Configure GitHub Repository :

- Create repository multi-cloud-storage on github.com.
- Initialize Git:


```
git init # Initialize Git repository
git add . # Add all files to staging
git commit -m "Initial commit with React setup" # Commit
changes
git remote add origin <repository-url> # Set remote repository
URL
git push -u origin main # Push to main branch
```
- Create .gitignore:


```
node_modules # Ignore Node.js dependencies
build # Ignore React build output
venv # Ignore Python virtual environment
__pycache__ # Ignore Python bytecode
*.pyc # Ignore Python compiled files
temp # Ignore temporary files
```

- o backend/key.json # Ignore GCP service account key
- o backend/.env # Ignore environment variables
- o **Set up GitHub Actions in .github/workflows/ci.yml:**
- o name: CI # Name of the workflow
- o on: # Trigger events
- o push: # Run on push to main branch
- o branches: [main]
- o jobs: # Define jobs
- o build: # Build job
- o runs-on: ubuntu-latest # Run on Ubuntu latest
- o steps: # Define steps
- o - uses: actions/checkout@v3 # Checkout code
- o - name: Set up Node.js # Step to set up Node.js
- o uses: actions/setup-node@v3
- o with:
- o node-version: '16' # Use Node.js version 16
- o - name: Install dependencies # Step to install dependencies
- o run: npm install # Run npm install
- o - name: Build # Step to build React app
- o run: npm run build # Run build script
- o - name: Test # Step to run tests
- o run: npm test --if-present # Run tests if available
- o **Commit and push:**
- o git add . # Add all changes
- o git commit -m "Add GitHub Actions CI" # Commit CI workflow
- o git push # Push to GitHub

```

1  name: CI
2  on: [push]
3  jobs:
4    build:
5      runs-on: ubuntu-latest
6      defaults:
7        run:
8          working-directory: ./storage-app
9      steps:
10     - uses: actions/checkout@v3
11       with:
12         path: storage-app
13     - uses: actions/setup-node@v3
14       with:
15         node-version: '16'
16         cache: 'npm'
17     - run: npm install
18     - run: npm run build

```

Figure 5: GitHub "Actions" tab showing successful CI run

- o **Why:** GitHub for version control, Actions for CI/CD.

Troubleshooting

- **npm start fails:** Delete node_modules and package-lock.json, run npm install, then npm start. Check port conflicts (lsof -i :3000).
- **Git errors:** Verify repository URL (git remote -v), ensure SSH/HTTPS authentication.
- **Extension issues:** Restart VS Code after installing extensions.

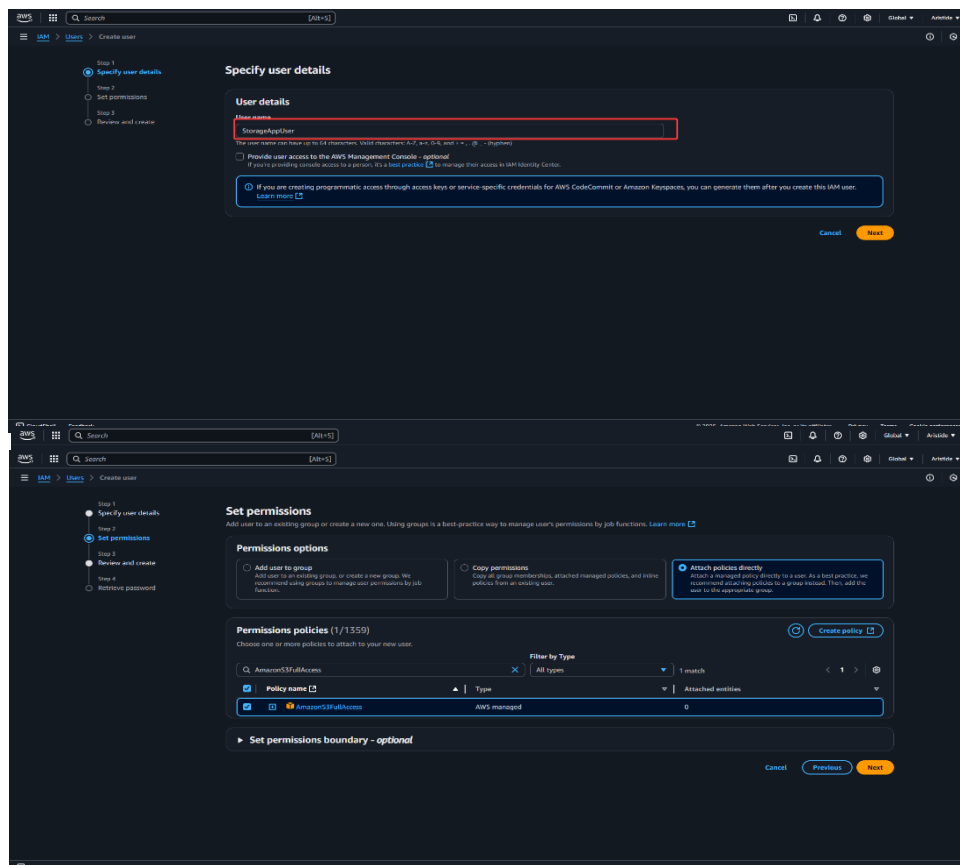
5. Cloud Account Setup (2 hours, June 3, 2025)

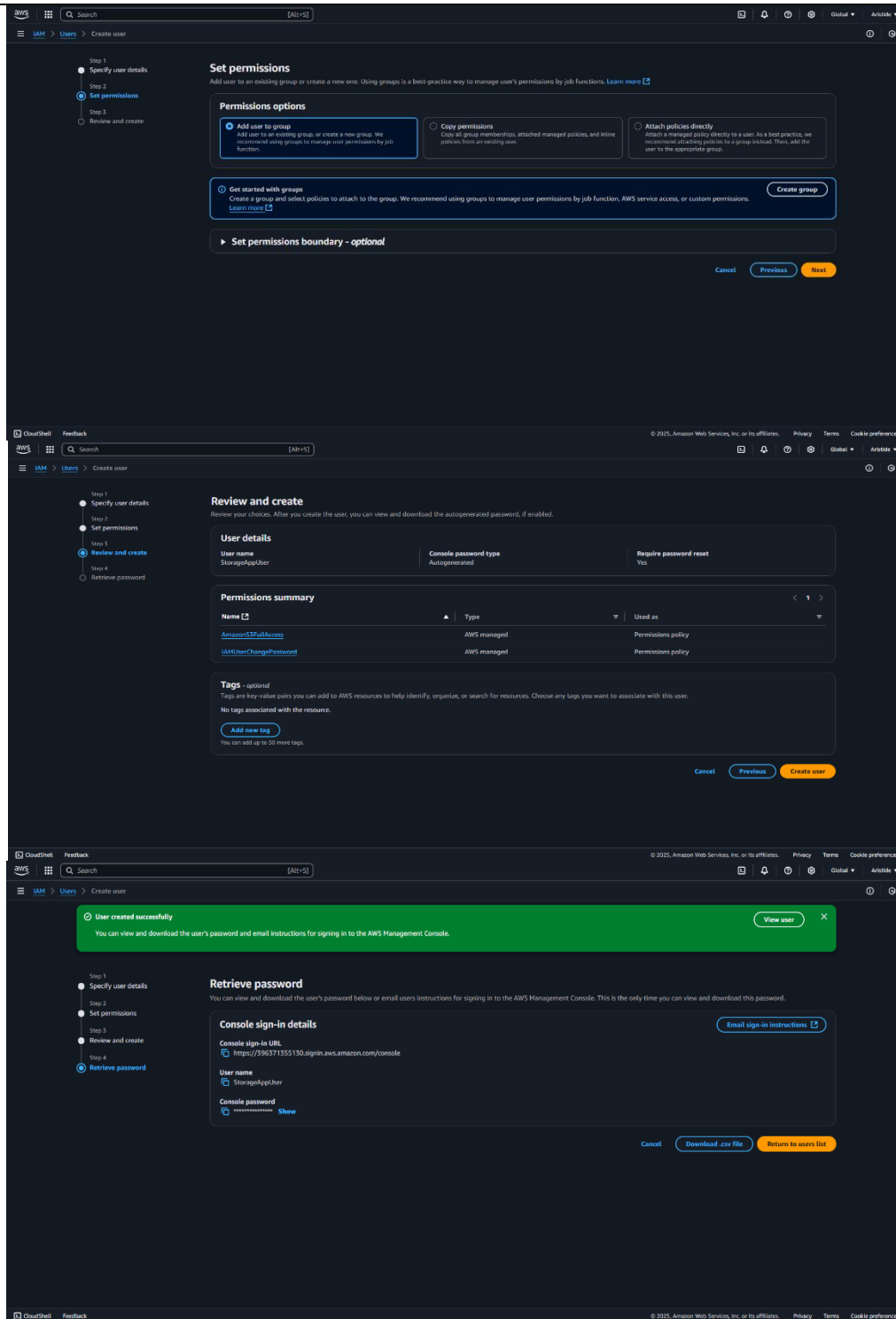
Objective: Register and configure free-tier accounts for AWS, Azure, and GCP.

Steps

5. AWS Free Tier :

- Go to aws.amazon.com/free, click “Create a Free Account.”
- Use unique email, provide payment info (no charges within 5GB S3, 20,000 GET requests).
- Select “Basic Plan (Free).”
- In AWS Console:
 - IAM > Users > Add User: storage-user, programmatic access.





The screenshot displays the AWS IAM console interface for creating a new user. The process is divided into four steps: 1. Specify user details, 2. Set permissions, 3. Review and create, and 4. Retrieve password. The 'Set permissions' step is currently active, showing options to add the user to a group, copy permissions, or attach policies directly. The 'Review and create' step follows, displaying user details (StorageAppUser), permissions summary (AmazonS3FullAccess, IAM:iam:ChangePassword), and tags. A green success message indicates the user was created successfully. The final step, 'Retrieve password', shows the console sign-in URL and the generated password.

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

- ☒ **Add user to group**
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- ☐ **Copy permissions**
Copy all group memberships, attached managed policies, and inline policies from an existing user.
- ☐ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. [Learn more](#)

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name StorageAppUser	Console password type Autogenerated	Require password reset Yes
-----------------------------	--	-------------------------------

Permissions summary

Name	Type	Used as
AmazonS3FullAccess	AWS managed	Permissions policy
IAM:iam:ChangePassword	AWS managed	Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

You can add up to 50 more tags.

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Console sign-in URL
<https://236371355130.signin.aws.amazon.com/console>

User name
StorageAppUser

Console password
[REDACTED]

Figure 6: Specify user creation details(AWS)

- Attach AmazonS3FullAccess policy.

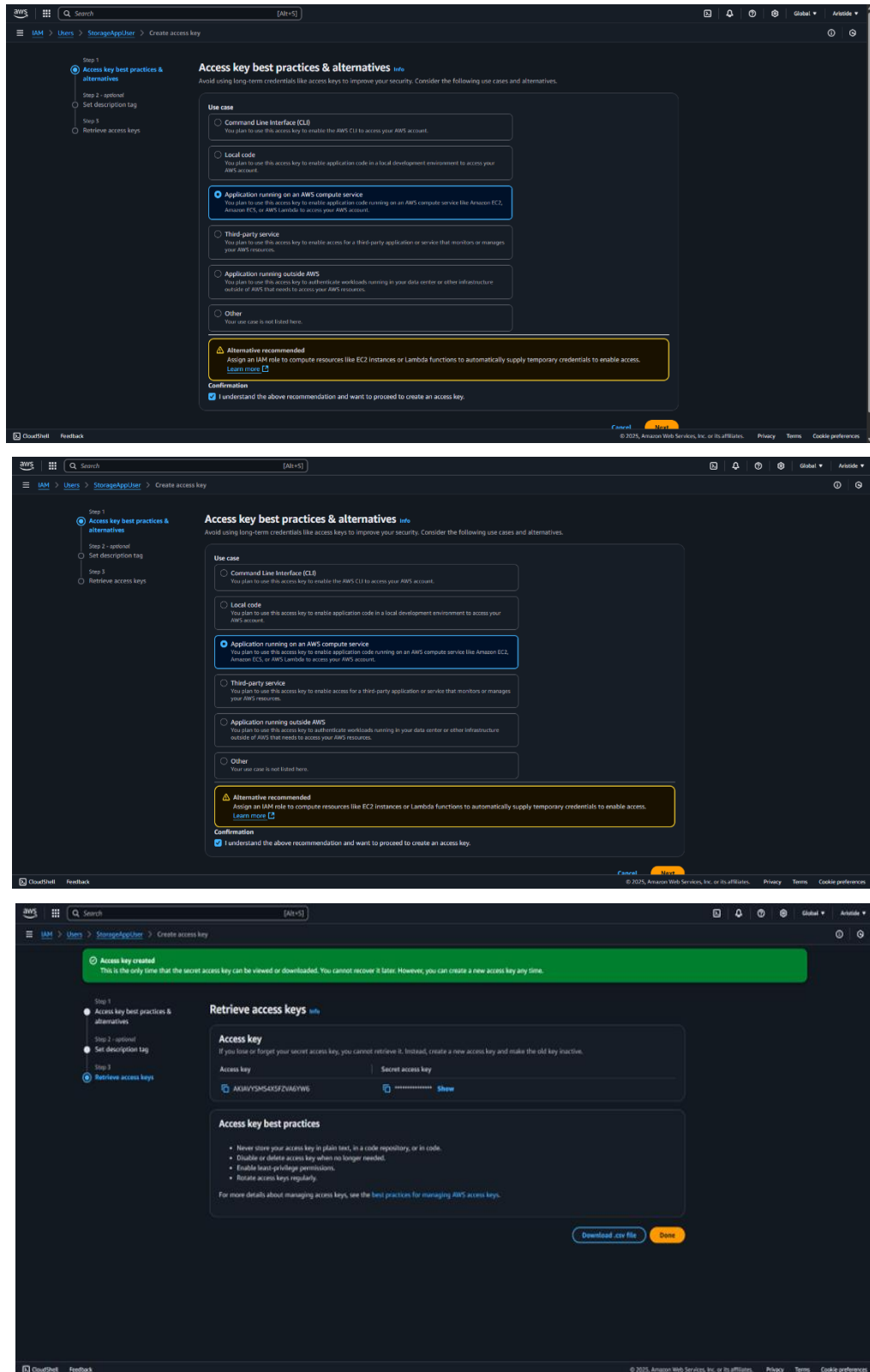


Figure 7: Creating an Access ID and Secret Key

- Save Access Key ID and Secret Access Key in backend/.env:

- `AWS_ACCESS_KEY_ID=your-access-key` # AWS access key for S3
- `AWS_SECRET_ACCESS_KEY=your-secret-key` # AWS secret key for S3

6. Azure Free Tier:

- Visit azure.microsoft.com/free, sign up with Microsoft account.
- Provide payment info for free services (5GB Blob Storage).
- In Azure Portal:
 - Create Storage Account: mystorageaccount, select “Locally-redundant storage” (LRS).
 - Go to Access keys, copy connection string to backend/.env:

`AZURE_STORAGE_CONNECTION_STRING=your-connection-string` # Azure Blob Storage connection string

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a 'Copilot' button. Below the header, the 'Azure services' section is visible, with 'Storage accounts' highlighted by a red box. The 'Resources' section shows a table with columns for Name, Type, and Last Viewed, but it's empty with a message 'No resources have been viewed recently'. The 'Navigate' section at the bottom includes links to Subscriptions, Resource groups, All resources, and Dashboard. Below the main content area, the 'Storage accounts' page is displayed, showing a 'Create' button highlighted with a red box. The page also includes a message 'No storage accounts to display' and a 'Create' button.

Home > Storage accounts >

Create a storage account

Basics Advanced Networking Data protection Encryption Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Azure subscription 1

Resource group *

(New) StorageDemoRG

Create new

Instance details

Storage account name * ⓘ

aristidestorage2025

Region * ⓘ

(Europe) France Central

Deploy to an Azure Extended Zone

Primary service ⓘ

Azure Blob Storage or Azure Data Lake Storage Gen 2

Performance * ⓘ

☒ **Standard:** Recommended for most scenarios (general-purpose v2 account)
 ☐ **Premium:** Recommended for scenarios that require low latency.

Redundancy * ⓘ

Locally-redundant storage (LRS)

Previous Next Review + create

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

aristataganuka2025@ou...

DEFAULT DIRECTORY

Home > Storage accounts >

Create a storage account

Secure transfer

Enabled

Blob anonymous access

Disabled

Allow storage account key access

Enabled

Default to Microsoft Entra authorization in the Azure portal

Disabled

Minimum TLS version

Version 1.2

Permitted scope for copy operations (preview)

From any storage account

Networking

Network connectivity

Public endpoint (all networks)

Default routing tier

Microsoft network routing

Data protection

Point-in-time restore

Disabled

Blob soft delete

Enabled

Blob retention period in days

7

Container soft delete

Enabled

Container retention period in days

7

File share soft delete

Enabled

File share retention period in days

7

Versioning

Disabled

Blob change feed

Disabled

Version-level immutability support

Disabled

Encryption

Encryption type

Microsoft-managed keys (MMK)

Enable support for customer-managed keys

Blobs and files only

Enable infrastructure encryption

Disabled

Previous

Next

Create

Initializing deployment...

Initializing template deployment to resource group 'StorageDemoRG'.

Give feedback

Microsoft Azure | Search resources, services, and docs (G+V) | Copilot

Home > aristidestorage2025_1750081140983 | Overview

Deployment

Search

Overview

Inputs

Outputs

Template

✓ **Your deployment is complete**

Deployment name: aristidestorage2025_1750081140983
Subscription: Azure subscription 1
Resource group: StorageDemoRG

Start time: 16/06/2025, 15:39:53
Correlation ID: bdbb14b6-c032-4397-826c-740e7c5cebcd

Deployment details

Next steps

[Go to resource](#)

Give feedback

Tell us about your experience with deployment

Deployment succeeded
Deployment 'aristidestorage2025_1750081140983' to resource group 'StorageDemoRG' was successful.
[Go to resource](#) [Pin to dashboard](#)

Cost Management
Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

Microsoft Defender for Cloud
Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

Free Microsoft tutorials
Start learning today >

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
[Find an Azure expert >](#)

Microsoft Azure | Search resources, services, and docs (G+V) | Copilot

Home > aristidestorage2025_1750081140983 | Overview > aristidestorage2025

Storage account

Search

Container

Change access level

Restore containers

Refresh

Delete

Give feedback

Search containers by prefix

Show deleted containers

Name	Last modified	Anonymous access level	Lease state
☐ Slogs	16/06/2025, 15:40:16	Private	Available

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage browser

Storage Mover

Partner solutions

Resource visualizer

Data storage

Containers

File shares

Queues

Tables

Security + networking

Data management

Settings

Monitoring

Monitoring (classic)

Automation

Help

<https://portal.azure.com/#@aristidestorage2025@outlook.com/microsoft.com/resource/subscriptions/16ac257b-63bb-4d14-b877-41cd1159f159/resourcegroups/StorageDemoRG/providers/Microsoft.Storage/storageAccounts/aristidestorage2025/containers/1>

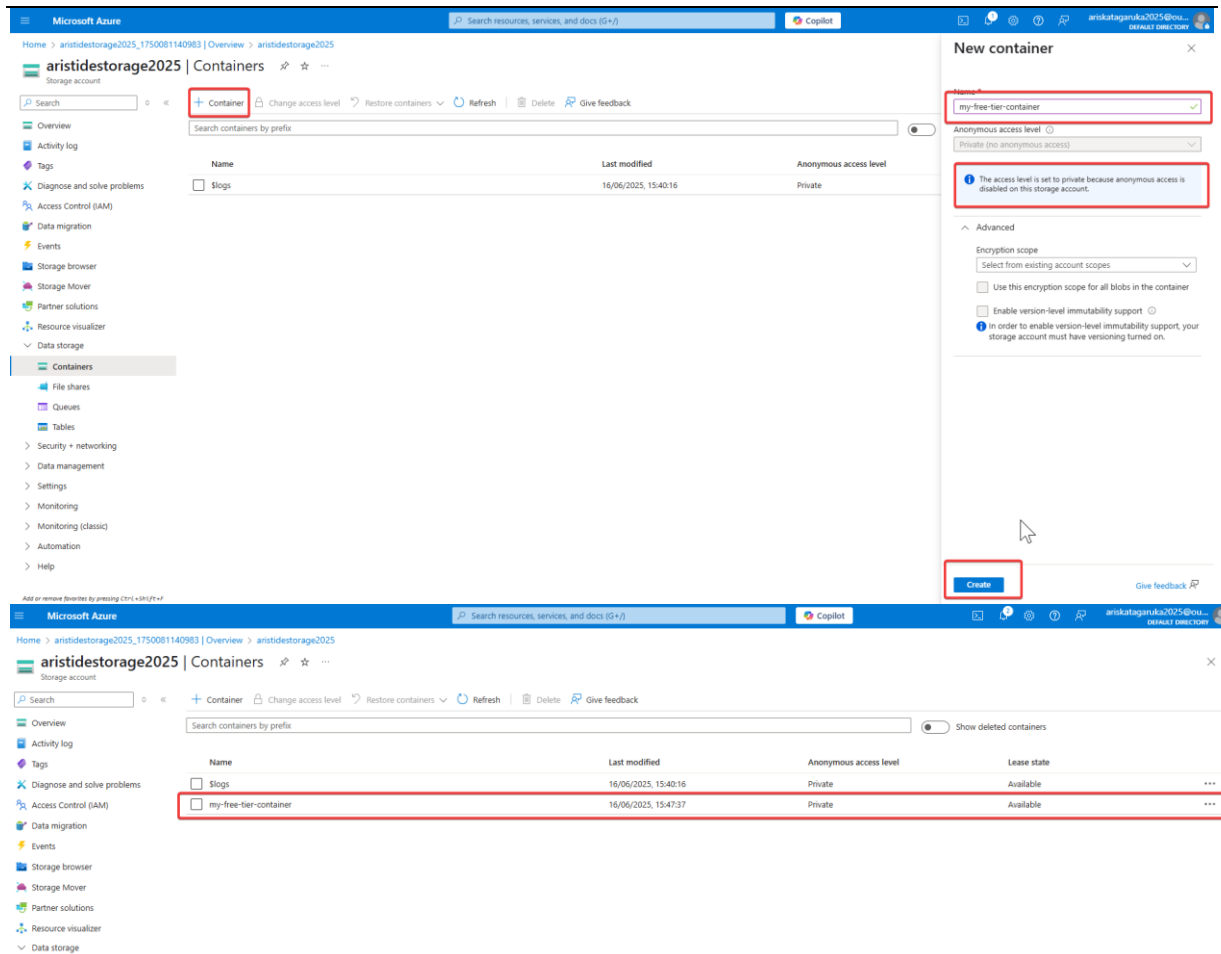


Figure 8: Azure Portal showing connection string

7. GCP Free Tier (0.6 hour):

- Sign up at cloud.google.com/free with \$300 credit (90 days).
- In Google Cloud Console:
 - Create project: multi-cloud-storage.
 - Enable Cloud Storage API (APIs & Services > Library).
 - Create service account: storage-service-account, assign "Storage Admin" role, download JSON key as backend/key.json.
 - Set in backend/.env:

```
GOOGLE_APPLICATION_CREDENTIALS=/app/key.json # Path to
GCP service account key
```

The screenshot shows the Google Cloud console interface. On the left, the navigation menu is open, highlighting 'APIs & Services'. A dropdown menu is visible under 'APIs & Services', showing options like 'Library', 'Credentials', 'OAuth consent screen', and 'Page usage agreements'. The main content area displays the 'Welcome, Aristide Katagaruka' message and a 'You're in Free Trial' banner. Below this, there are recommended products based on interest in 'Web / Mobile Apps'. The 'API Library' section is active, showing a search bar with 'cloud storage' entered. The search results are displayed in a grid, listing various APIs such as 'Maps SDK for Android', 'Maps SDK for iOS', 'Maps JavaScript API', 'Places API', 'Roads API', 'Directions API', 'Dialogflow API', 'Cloud Vision API', 'Cloud Natural Language API', 'Cloud Speech-to-Text API', 'Cloud Translation API', and 'AI Platform Training & Prediction API'. The 'Cloud Storage' API is highlighted in the results.

Google Cloud

Project

Welcome, Aristide Katagaruka

You're in Free Trial

0 out of €265 credits used

Expires September 15, 2025

What happens when trial ends?

Activate full account

You're working on project **My First Project**

Number: 37895144852 ID: high-empire-463106-v4

Add people to your project

Set up budget alerts

Review product spend

Try Gemini 2.0 Flash

Try Gemini

Recommended based on your interest in **Web / Mobile Apps**

Products

Create a VM

Compute Engine

Deploy an application

Cloud Run

Build a mobile or web app

Firebase

Create a Kubernetes cluster

Kubernetes Engine

Create a storage bucket

Cloud Storage

Distribute network traffic

Cloud Load Balancing

Accelerate content delivery

Cloud CDN

Monitor your workloads

Cloud Observability

API Library

Welcome to the API Library

The API Library has documentation, links, and a smart search experience.

cloud storage

Filter

Visibility

Public (489)

Private (2)

Category

Analytics (11)

Big data (22)

Databases (7)

Machine learning (16)

Maps (31)

DevOps (24)

Compute (13)

Maps

Maps SDK for Android

Google

Maps for your native Android app.

Maps SDK for iOS

Google

Maps for your native iOS app.

Maps JavaScript API

Google

Maps for your website

Places API

Google Enterprise API

Get detailed information about 100 million places

Roads API

Google Enterprise API

Snap-to-road functionality to accurately trace GPS breadcrumbs.

Directions API

Google Enterprise API

Directions between multiple locations.

Machine learning

Dialogflow API

Google Enterprise API

Builds conversational interfaces

Cloud Vision API

Google Enterprise API

Image Content Analysis

Cloud Natural Language API

Google Enterprise API

Provides natural language understanding technologies, such as sentiment analysis, entity recognition, and topic classification.

Cloud Speech-to-Text API

Google Enterprise API

Speech recognition

Cloud Translation API

Google Enterprise API

Integrates text translation into your website or application.

AI Platform Training & Prediction API

Google Enterprise API

An API to enable creating and using machine learning models.

API Library

cloud storage api

15 results

Cloud Storage

Google Enterprise API

Google Cloud Storage is a RESTful service for storing and accessing your data on Google's infrastructure.

Google Cloud Storage JSON API

Google Enterprise API

Lets you store and retrieve potentially-large, immutable data objects.

Cloud Storage API

Google Enterprise API

Lets you store and retrieve potentially-large, immutable data objects.

Cloud Storage for Firebase API

Google

The Cloud Storage for Firebase API enables programmatic management of Cloud Storage buckets for use in Firebase projects

Storage Transfer API

Google Enterprise API

Transfers data from external data sources to a Google Cloud Storage bucket or between Google Cloud Storage buckets.

Storage Insights API

The screenshot displays the Google Cloud console interface. At the top, the 'Cloud Storage API' is shown as 'API Enabled'. Below this, the 'Overview' section provides details about the API, including its type (SaaS & APIs), last product update (4/11/23), category (Google Enterprise APIs), and service name (storage.googleapis.com). A sidebar on the left lists various Google Cloud products, with 'APIs & Services' highlighted. The main content area shows the 'API Service Details' for the Cloud Storage API, including its status (Enabled) and documentation links. Below this, the 'Service Accounts' section is visible, showing a table with columns for Email, Status, Name, Description, Key ID, Key creation date, OAuth 2 Client ID, and Actions. A '+ Create service account' button is prominently displayed. The right sidebar contains a 'Recommended for you' section with links to service accounts overview, creation, and management guides.

The screenshot displays the Google Cloud IAM & Admin console interface for creating a service account. The left sidebar shows the navigation menu with 'Service Accounts' selected. The main content area is titled 'Create service account' and consists of three steps:

- 1 Create service account**: This step includes fields for 'Service account name' (filled with 'storage-app-user'), 'Service account ID' (filled with 'storage-app-user'), 'Email address' (displaying 'storage-app-user@high-empire-463106-v4.iam.gserviceaccount.com'), and 'Service account description'. A 'Create and continue' button is highlighted.
- 2 Permissions (optional)**: This step prompts the user to grant permissions. A 'Select a role' dialog box is open, showing a list of roles. The 'Storage Admin' role is selected and highlighted, with a tooltip indicating it 'Grants full control of buckets and objects'. The 'Cloud Storage' role is also visible in the list.
- 3 Principals with access (optional)**: This step is currently empty.

At the bottom of the console, a notification states 'Service account created'.

The first screenshot shows the 'Service accounts' page for project 'My First Project'. A table lists service accounts, with 'storage-app-user@high-empire-463106-vt.iam.gserviceaccount.com' highlighted. A message states: 'No change - principal already exists on the policy'.

The second screenshot shows the 'Keys' page for the 'storage-app-user' service account. It displays a warning about security risks and provides instructions on how to add a new key or upload an existing one.

The third screenshot shows the 'Add key' dropdown menu, with 'Create new key' and 'Upload existing key' options visible.

The first screenshot shows the Google Cloud IAM & Admin console. A modal dialog titled "Create private key for 'storage-app-user'" is open. The "Key type" is set to "JSON" (Recommended). The "Create" button is highlighted.

The second screenshot shows the Google Cloud Cloud Storage console. The "cloud storage" tab is selected. The "Create bucket" button is highlighted.

The third screenshot shows the "Create a bucket" wizard. The "Get Started" step is active. The "Pick a globally unique, permanent name" field contains "aristide-free-tier-2025". The "Labels (optional)" dropdown is expanded. The "Choose where to store your data" section shows "Multi-region" selected, with "eu (multiple regions in European Union)" chosen from the dropdown. The "Choose how to store your data" section is partially visible.

The figure consists of three screenshots from the Google Cloud Console, illustrating the process of creating a new Cloud Storage bucket.

Top Screenshot: 'Create a bucket' - Choose how to store your data
 This screen shows the initial configuration options. The 'Set a default class' option is selected and highlighted with a red box. Below it, the 'Standard' storage class is chosen. The 'Public access prevention' is set to 'On' and 'Access control' is set to 'Uniform'. The 'Continue' button is visible at the bottom.

Middle Screenshot: 'Create a bucket' - Choose how to protect object data
 This screen shows the data protection options. The 'Soft delete policy' is enabled. The 'Use default retention duration' option is selected. A red arrow points to the 'Create' button at the bottom left, with the text 'click "Create"!' next to it.

Bottom Screenshot: 'Bucket details' for 'aristide-free-tier-2025'
 This screen shows the details of the newly created bucket. The bucket name 'aristide-free-tier-2025' is highlighted with a red box. The 'Folder browser' tab is active, showing a list of objects (currently empty). A large circular icon with a plus sign is displayed in the center of the bucket view.

Figure 9:: GCP Console showing enabled API and downloaded key

Troubleshooting

- **Account conflicts:** Use unique emails, clear cookies.
- **Payment issues:** Try virtual card or contact support.
- **API not enabled:** Verify Cloud Storage API in GCP Console.

6. Web App Frontend Development (8 hours, June 5, 2025)

Objective: Build a React interface with components for file operations, dashboard, storage info, and provider comparison.

Steps

8. Set Up React Structure:

- Create folder structure:
- `cd storage-app # Navigate to React project`
`mkdir src/components src/pages src/services # Create component, pages, and services directories`
- Create `storage-app/src/pages/Home.js`:
- `import React from 'react'; // Import React library`
- `function Home() { // Define Home component`
- `return ( // Render JSX`
- `<div className="p-4 bg-gray-100 min-h-screen"> //`
Container with padding and background
- `<h1 className="text-2xl font-bold text-blue-600">Multi-`
Cloud Storage Demo`</h1> // Main title`
- `<p className="text-gray-700">Manage files across AWS,`
Azure, and GCP.`</p> // Description`
- `</div>`
- `);`
- `}`
- `export default Home; // Export Home component`
- Update `storage-app/src/App.js`:
- `import Home from '../pages/Home'; // Import Home component`
- `function App() { // Define App component`
- `return ( // Render JSX`
- `<div className="App"> // App container`
- `<Home /> // Render Home component`
- `</div>`
- `);`
- `}`
- `export default App; // Export App component`
- Test: `npm start`, verify at `http://localhost:3000`.

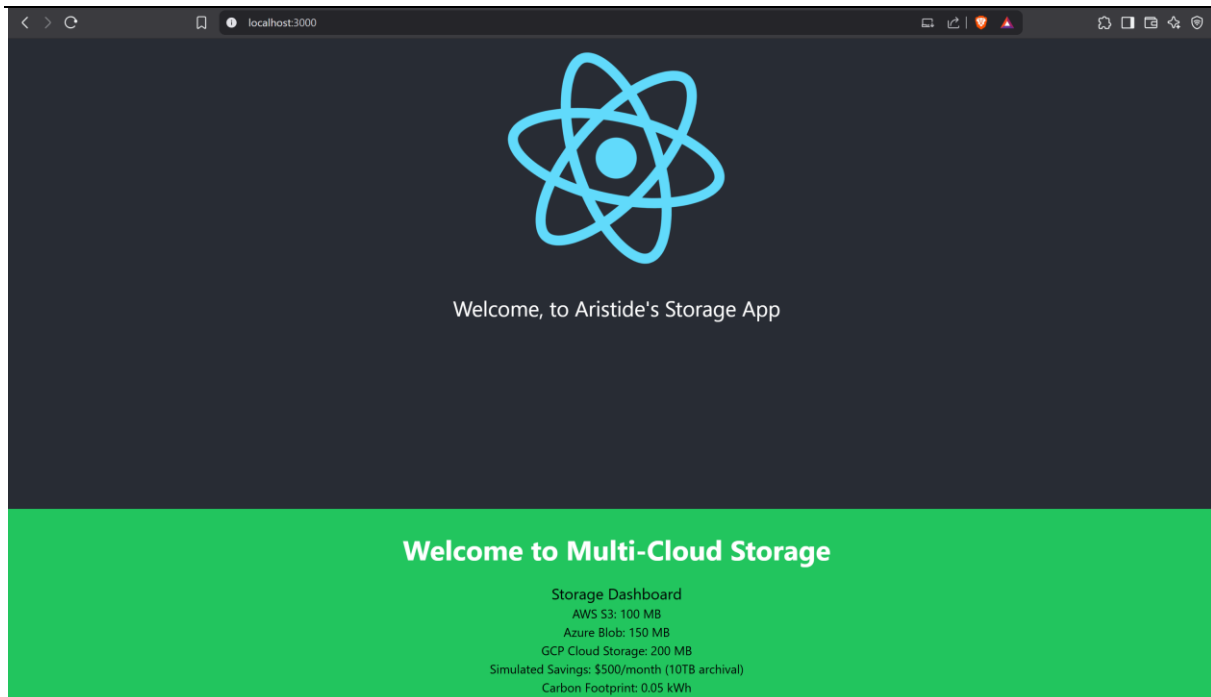


Figure 10: Browser showing Home.js with styled header

9. Implement File Upload/Download UI:

- Create storage-app/src/components/FileUpload.js:
- `import React, { useState } from 'react'; // Import React and useState hook`
- `import { uploadFile } from '../services/api'; // Import API utility for file upload`
- `function FileUpload() { // Define FileUpload component`
- `const [file, setFile] = useState(null); // State for selected file`
- `const [provider, setProvider] = useState('s3'); // State for selected cloud provider`
- `const [status, setStatus] = useState(''); // State for upload status message`
- `const handleFileChange = (e) => { // Handle file input change`
- `if (e.target.files.length > 0) { // Check if file is selected`
- `setFile(e.target.files[0]); // Update file state`
- `}`
- `};`
- `const handleUpload = async () => { // Handle file upload`
- `if (!file) { // Check if file is selected`
- `setStatus('No file selected'); // Set error status`
- `return;`
- `}`
- `setStatus('Uploading...'); // Set uploading status`
- `try { // Handle API call`
- `await uploadFile(file, provider); // Call upload API`
- `setStatus(`Uploaded to ${provider.toUpperCase()}`); // Set success status`
- `} catch (error) { // Handle errors`

```

o      setStatus(`Upload failed: ${error.message}`); // Set
o      error status
o    }
o  };
o  return ( // Render JSX
o    <div className="m-4 bg-white p-4 rounded shadow"> //
o      Container with styling
o      <h2 className="text-xl font-semibold text-gray-800">File
o      Upload</h2> // Section title
o      <select
o        value={provider} // Bind provider state
o        onChange={ (e) => setProvider(e.target.value)} //
o      Update provider state
o        className="mb-2 border rounded p-2" // Styling for
o      dropdown
o      >
o        <option value="s3">AWS S3</option> // AWS option
o        <option value="azure">Azure Blob</option> // Azure
o      option
o        <option value="gcp">GCP Cloud Storage</option> // GCP
o      option
o      </select>
o      <input
o        type="file" // File input
o        onChange={handleFileChange} // Bind file change
o      handler
o        className="mb-2 border rounded p-2" // Styling for
o      input
o      />
o      <button
o        onClick={handleUpload} // Bind upload handler
o        className="
o        bg-blue-500 text-white p-2 rounded hover:bg-blue-600"
o      // Styling for button
o      >
o        Upload File // Button text
o      </button>
o      <p className="mt-2 text-gray-700">{status}</p> //
o      Display status message
o    </div>
o  );
o }
o export default FileUpload; // Export FileUpload component

```

```

o Create storage-app/src/services/api.js:
o import axios from 'axios'; // Import axios for API calls
o export const uploadFile = async (file, provider) => { //
o   Function to upload single file
o   const formData = new FormData(); // Create FormData for file
o   upload
o   formData.append('file', file); // Append file to FormData
o   const response = await axios.post(`/upload/${provider}`,
o   formData); // Send POST request
o   return response.data; // Return response data
o };
o export const uploadBatch = async (files) => { // Function to
o   upload multiple files
o   const formData = new FormData(); // Create FormData for file
o   upload

```

```

o   for (let i = 0; i < files.length; i++) { // Loop through
o     formData.append('files', files[i]); // Append each file to
o     formData
o   }
o   const response = await axios.post('/upload/batch', formData);
o   // Send POST request
o   return response.data; // Return response data
o };

```

10. Implement Storage Dashboard:

```

o   Create storage-app/src/components/StorageDashboard.js:
o   import React, { useState, useEffect } from 'react'; // Import
o   React, useState, and useEffect
o   import axios from 'axios'; // Import axios for API calls
o   function StorageDashboard() { // Define StorageDashboard
o     component
o     const [data, setData] = useState({ // Initialize state for
o       dashboard data
o       usage: { s3: 0, azure: 0, gcp: 0 }, // Storage usage for
o       each provider
o       savings: 0, // Cost savings
o       alerts: [], // API limit alerts
o       energy: 0 // Carbon footprint
o     });
o     useEffect(() => { // Fetch data on component mount
o       axios.get('/dashboard') // Send GET request to dashboard
o       endpoint
o       .then(response => setData(response.data)) // Update
o       state with response data
o       .catch(error => console.error('Error fetching dashboard
o       data:', error)); // Log errors
o     }, []); // Empty dependency array for single fetch
o     return ( // Render JSX
o       <div className="m-4 bg-white p-4 rounded shadow"> //
o       Container with styling
o       <h2 className="text-xl font-semibold text-gray-
o       800">Storage Dashboard</h2> // Section title
o       <p className="text-gray-700">AWS S3: {data.usage.s3}
o       MB</p> // Display S3 usage
o       <p className="text-gray-700">Azure Blob:
o       {data.usage.azure} MB</p> // Display Azure usage
o       <p className="text-gray-700">GCP Cloud Storage:
o       {data.usage.gcp} MB</p> // Display GCP usage
o       <p className="text-green-600">Savings:
o       ${data.savings}/month (10TB archival)</p> // Display savings
o       <p className="text-gray-700">Carbon Footprint:
o       {data.energy} kWh</p> // Display energy usage
o       <div className="bg-red-200 p-2 mt-2 rounded"> //
o       Container for alerts
o       {data.alerts.map((alert, i) => ( // Map through alerts
o       <p key={i} className="text-red-700">{alert}</p> //
o       Display each alert
o       )))
o       </div>
o     </div>
o   );
o }

```

```
export default StorageDashboard; // Export StorageDashboard component
```

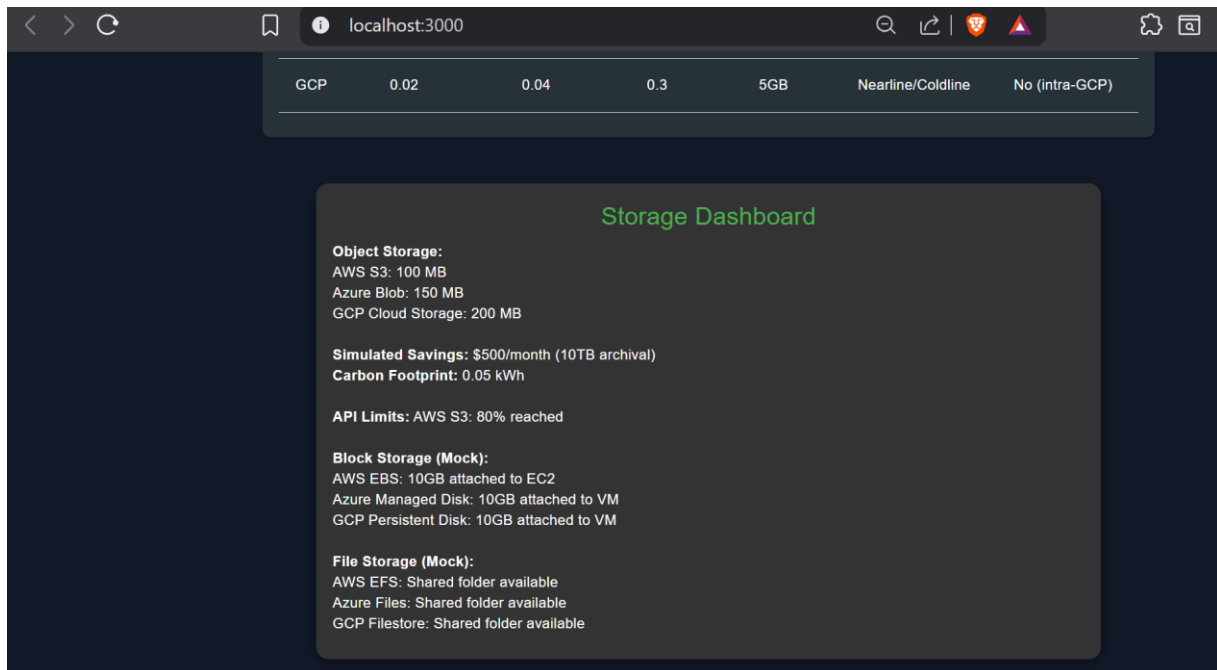


Figure 11: Browser showing dashboard with mock data

11. Implement Storage Info and Use Cases:

- o Create storage-app/src/components/StorageInfo.js:
- o `import React, { useState, useEffect } from 'react'; // Import React, useState, and useEffect`
- o `import axios from 'axios'; // Import axios for API calls`
- o `function StorageInfo() { // Define StorageInfo component`
- o `const [blockData, setBlockData] = useState({}); // State for block storage data`
- o `const [fileData, setFileData] = useState({}); // State for file storage data`
- o `useEffect(() => { // Fetch data on component mount`
- o `axios.get('/demo/block') // Send GET request for block storage data`
- o `.then(response => setBlockData(response.data)) // Update block data state`
- o `.catch(error => console.error('Error fetching block data:', error)); // Log errors`
- o `axios.get('/demo/file') // Send GET request for file storage data`
- o `.then(response => setFileData(response.data)) // Update file data state`
- o `.catch(error => console.error('Error fetching file data:', error)); // Log errors`
- o `}, []); // Empty dependency array for single fetch`
- o `return ( // Render JSX`
- o `<div className="m-4 bg-white p-4 rounded shadow"> // Container with styling`
- o `<h2 className="text-xl font-semibold text-gray-800">Storage Types & Use Cases</h2> // Section title`
- o `<div className="mt-2"> // Container for storage types`

```

o      <h3 className="text-lg font-medium">Object Storage</h3>
o      // Object storage title
o      <p className="text-gray-700">Ideal for backups,
o      archives (e.g., AWS S3 Glacier).</p> // Object storage
o      description
o      <h3 className="text-lg font-medium mt-2">Block
o      Storage</h3> // Block storage title
o      <p className="text-gray-700">AWS: {blockData.aws}</p>
o      // Display AWS block data
o      <p className="text-gray-700">Azure:
o      {blockData.azure}</p> // Display Azure block data
o      <p className="text-gray-700">GCP: {blockData.gcp}</p>
o      // Display GCP block data
o      <h3 className="text-lg font-medium mt-2">File
o      Storage</h3> // File storage title
o      <p className="text-gray-700">AWS: {fileData.aws}</p>
o      // Display AWS file data
o      <p className="text-gray-700">Azure:
o      {fileData.azure}</p> // Display Azure file data
o      <p className="text-gray-700">GCP: {fileData.gcp}</p>
o      // Display GCP file data
o      </div>
o      </div>
o      );
o    }
o    export default StorageInfo; // Export StorageInfo component
  
```

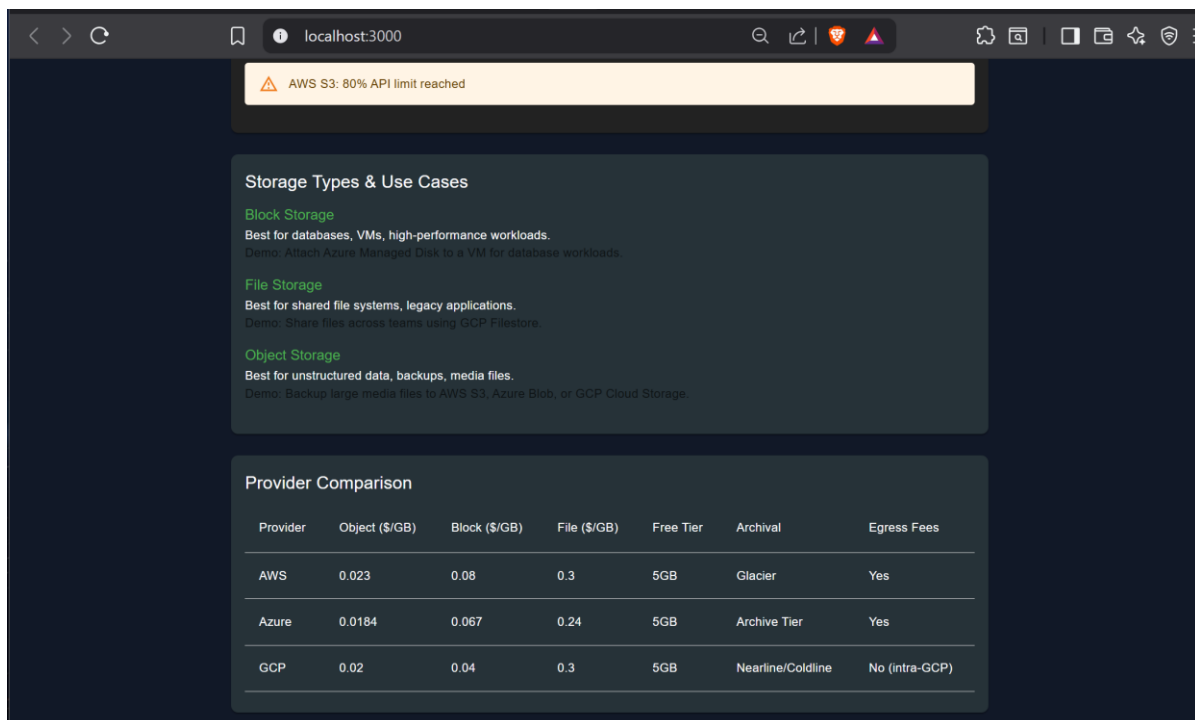


Figure 12: Browser showing storage types and use cases

12. Implement Provider Comparison:

- o Create storage-app/src/components/ProviderComparison.js:
- o import React, { useState, useEffect } from 'react'; // Import React, useState, and useEffect

```

o import axios from 'axios'; // Import axios for API calls
o function ProviderComparison() { // Define ProviderComparison
  component
o   const [providers, setProviders] = useState([]); // State for
  provider data
o   useEffect(() => { // Fetch data on component mount
o     axios.get('/provider/comparison') // Send GET request for
  comparison data
o     .then(response => setProviders(response.data)) // Update
  providers state
o     .catch(error => console.error('Error fetching comparison
  data:', error)); // Log errors
o   }, []); // Empty dependency array for single fetch
o   return ( // Render JSX
o     <div className="m-4 bg-white p-4 rounded shadow"> //
  Container with styling
o     <h2 className="text-xl font-semibold text-gray-
  800">Provider Comparison</h2> // Section title
o     <table className="w-full mt-2 text-left"> // Table for
  comparison data
o       <thead> // Table header
o         <tr className="bg-gray-200"> // Header row styling
o           <th className="p-2">Provider</th> // Provider
  column
o           <th className="p-2">Cost</th> // Cost column
o           <th className="p-2">Free Tier</th> // Free Tier
  column
o           <th className="p-2">Egress Fees</th> // Egress
  Fees column
o         </tr>
o       </thead>
o       <tbody> // Table body
o         {providers.map((provider, i) => ( // Map through
  providers
o           <tr key={i} className="border-t"> // Table row
  with border
o             <td className="p-2">{provider.name}</td> //
  Display provider name
o             <td className="p-2">{provider.cost}</td> //
  Display cost
o             <td className="p-2">{provider.freeTier}</td> //
  Display free tier
o             <td className="p-2">{provider.egress}</td> //
  Display egress fees
o           </tr>
o         )}}
o       </tbody>
o     </table>
o   </div>
o );
o }
o export default ProviderComparison; // Export
  ProviderComparison component

o Update Home.js to include all components:
o import React from 'react'; // Import React library
o import FileUpload from '../components/FileUpload'; // Import
  FileUpload component

```

```

o import StorageDashboard from '../components/StorageDashboard';
  // Import StorageDashboard component
o import StorageInfo from '../components/StorageInfo'; // Import
  StorageInfo component
o import ProviderComparison from
  '../components/ProviderComparison'; // Import
  ProviderComparison component
o function Home() { // Define Home component
o   return ( // Render JSX
o     <div className="p-4 bg-gray-100 min-h-screen"> //
      Container with styling
o       <h1 className="text-2xl font-bold text-blue-600">Multi-
        Cloud Storage Demo</h1> // Main title
o       <FileUpload /> // Render file upload component
o       <StorageDashboard /> // Render dashboard component
o       <StorageInfo /> // Render storage info component
o       <ProviderComparison /> // Render provider comparison
        component
o     </div>
o   );
o }
export default Home; // Export Home component
  
```

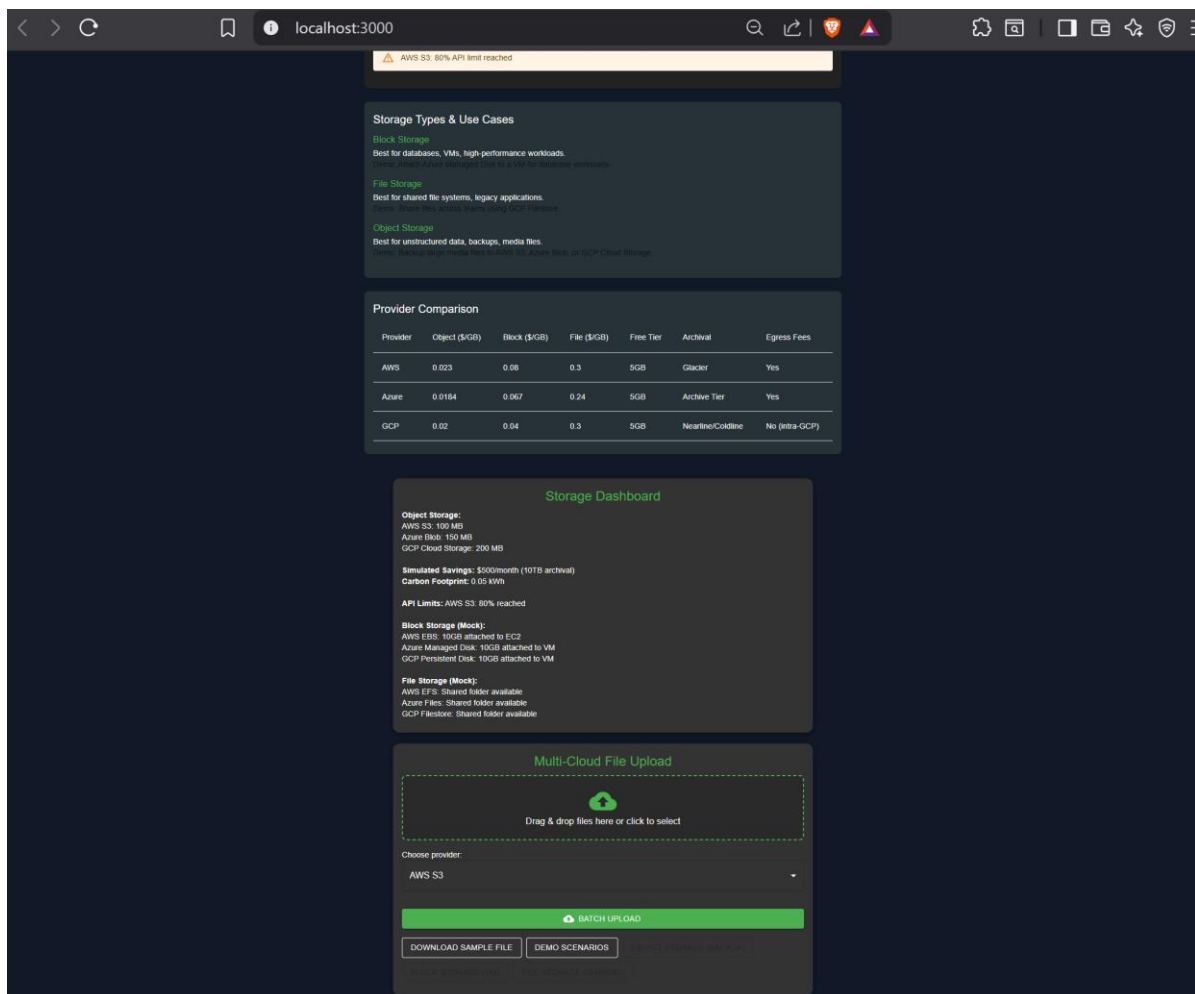


Figure 13: Browser showing all components (upload, dashboard, info, comparison).

Troubleshooting

- **UI not rendering:** Check browser console (Ctrl+Shift+J), verify imports.
- **Tailwind issues:** Ensure tailwind.config.js includes src/**/*.{js,jsx}.
- **API call failures:** Verify package.json proxy and backend running.

7. Cloud Storage API Integration (10 hours, June 7, 2025)

Objective: Develop Flask backend with endpoints for file uploads, mock scenarios, dashboard data, and provider comparison.

Steps

13. Set Up Flask Backend:

- **Create backend folder:**

```
mkdir backend # Create backend directory
cd backend # Navigate to backend directory
python -m venv venv # Create Python virtual environment
source venv/bin/activate # Activate virtual environment
# Windows: venv\Scripts\activate
pip install flask flask-cors boto3 azure-storage-blob google-cloud-storage redis # Install dependencies
```
- **Create backend/requirements.txt:**

```
flask==2.0.1 # Flask framework for backend
flask-cors==3.0.10 # CORS support for Flask
boto3==1.24.0 # AWS SDK for Python
azure-storage-blob==12.13.0 # Azure Blob Storage SDK
google-cloud-storage==2.5.0 # GCP Cloud Storage SDK
redis==3.5.3 # Redis client for caching
```
- **Create backend/app.py:**

```
from flask import Flask, request # Import Flask for web server and request handling
from flask_cors import CORS # Import CORS for cross-origin requests
app = Flask(__name__) # Initialize Flask app
CORS(app) # Enable CORS for frontend communication
@app.route('/') # Define root endpoint
def home(): # Home route handler
    return 'Backend is running!' # Return status message
if __name__ == '__main__': # Check if script is run directly
    app.run(debug=True, port=5000) # Start Flask server on port 5000
```
- **Create temp folder for temporary file storage:**

```
mkdir temp # Create temporary file storage directory
```

- **Start backend:**
- `pip install -r requirements.txt` # Install dependencies
- `python app.py` # Run Flask app
- Verify at `http://localhost:5000`.

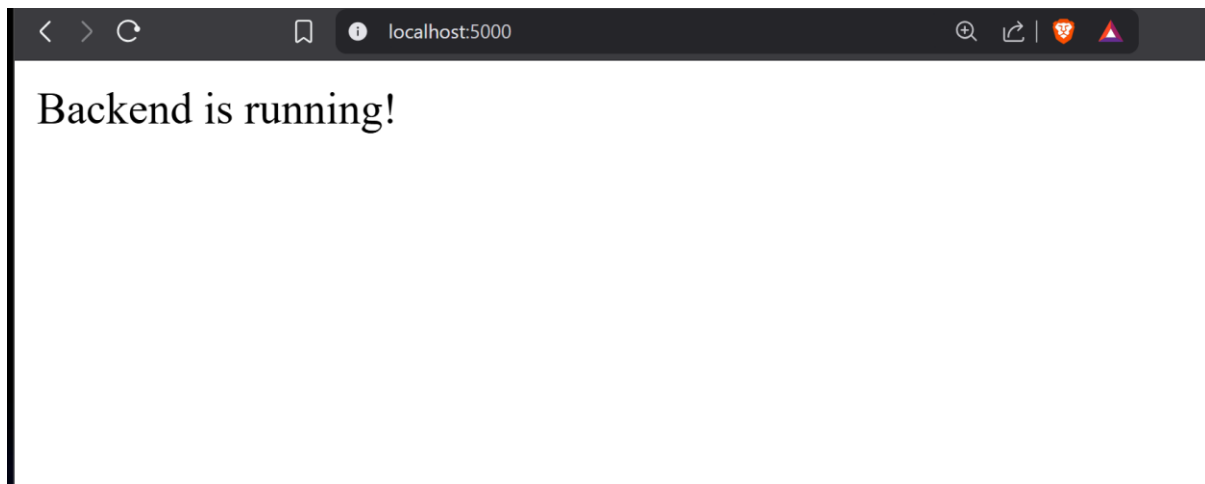


Figure 14: Browser showing "Backend is running!".

14. Integrate AWS S3:

- **Create backend/.env:**
- `AWS_ACCESS_KEY_ID=your-access-key` # AWS access key for S3
- `AWS_SECRET_ACCESS_KEY=your-secret-key` # AWS secret key for S3
- `AZURE_STORAGE_CONNECTION_STRING=your-connection-string` # Azure Blob Storage Connection string
- `GOOGLE_APPLICATION_CREDENTIALS=/app/key.json` # Path to GCP service account key
- **Create backend/file_upload.py (see Section 15).**
- **Update app.py with S3 upload endpoint:**
- `from file_upload import upload_to_s3` # Import S3 upload function
- `from energy_estimator import estimate_energy` # Import energy estimation function
- `import redis` # Import Redis for caching
- `cache = redis.Redis(host='redis', port=6379)` # Initialize Redis client
- `def get_cached(key):` # Function to get cached data
- `return cache.get(key)` # Retrieve value from Redis
- `def set_cached(key, value, ttl=300):` # Function to set cached data
- `cache.setex(key, ttl, value)` # Store value in Redis with TTL
- `@app.route('/upload/s3', methods=['POST'])` # Define S3 upload endpoint
- `def upload_s3():` # S3 upload handler
- `if 'file' not in request.files:` # Check if file is included
- `return 'No file provided', 400` # Return error if no file
- `file = request.files['file']` # Get uploaded file

- ```
o file_path = f'temp/{file.filename}' # Define temporary
 file path
o cache_key = f'upload_s3_{file.filename}' # Create cache
 key for file
o if get_cached(cache_key): # Check if file is already
 uploaded
o return 'File already uploaded', 200 # Return cached
 response
o file.save(file_path) # Save file to temporary path
o upload_to_s3(file_path, 'my-free-tier-bucket',
 file.filename) # Upload to S3
o set_cached(cache_key, 'Uploaded') # Cache upload status
o energy = estimate_energy(file.content_length / (1024 *
 1024)) # Calculate energy usage
 return f'File uploaded to S3: {file.filename}, Energy:
 {energy:.4f} kWh', 200 # Return success message
```
- o In AWS Console: Create bucket my-free-tier-bucket (us-east-1, disable public access block).
  - o Test:

```
curl -X POST -F "file=@test.txt"
http://localhost:5000/upload/s3 # Test S3 upload
```

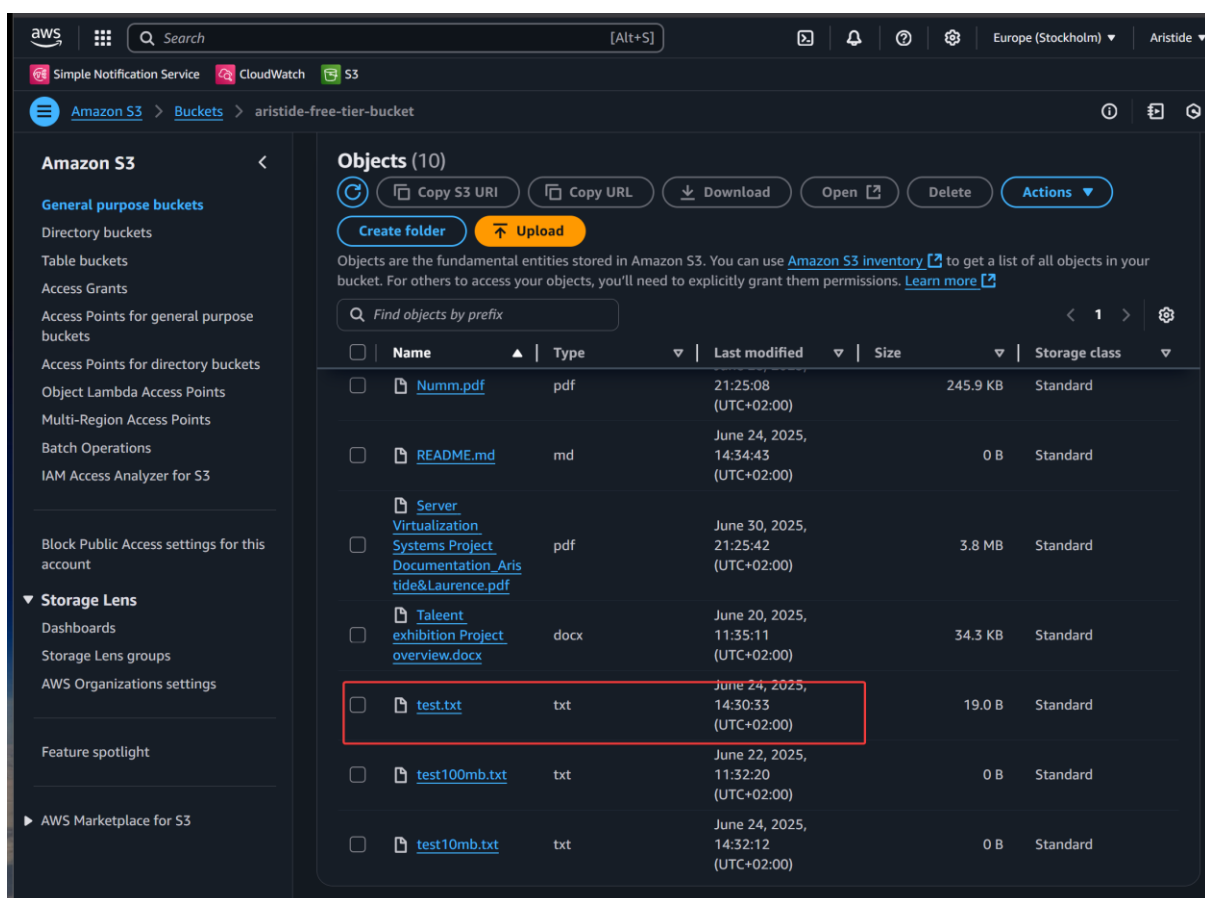


Figure 15: AWS S3 Console showing my-free-tier-bucket with test.txt

## 15. Integrate Azure Blob Storage:

- o Add to app.py:
- o 

```
from file_upload import upload_to_azure # Import Azure upload function
```
- o 

```
@app.route('/upload/azure', methods=['POST']) # Define Azure upload endpoint
```
- o 

```
def upload_azure(): # Azure upload handler
```
- o 

```
 if 'file' not in request.files: # Check if file is included
```
- o 

```
 return 'No file provided', 400 # Return error if no file
```
- o 

```
 file = request.files['file'] # Get uploaded file
```
- o 

```
 file_path = f'temp/{file.filename}' # Define temporary file path
```
- o 

```
 cache_key = f'upload_azure_{file.filename}' # Create cache key for file
```
- o 

```
 if get_cached(cache_key): # Check if file is already uploaded
```
- o 

```
 return 'File already uploaded', 200 # Return cached response
```
- o 

```
 file.save(file_path) # Save file to temporary path
```
- o 

```
 upload_to_azure(file_path, 'my-free-tier-container', file.filename) # Upload to Azure
```
- o 

```
 set_cached(cache_key, 'Uploaded') # Cache upload status
```
- o 

```
 energy = estimate_energy(file.content_length / (1024 * 1024)) # Calculate energy usage
```
- o 

```
 return f'File uploaded to Azure: {file.filename}, Energy: {energy:.4f} kWh', 200 # Return success message
```
- o In Azure Portal: Create container my-free-tier-container.
- o Test:

```
curl -X POST -F "file=@test100mb.txt"
http://localhost:5000/upload/azure # Test Azure upload
```

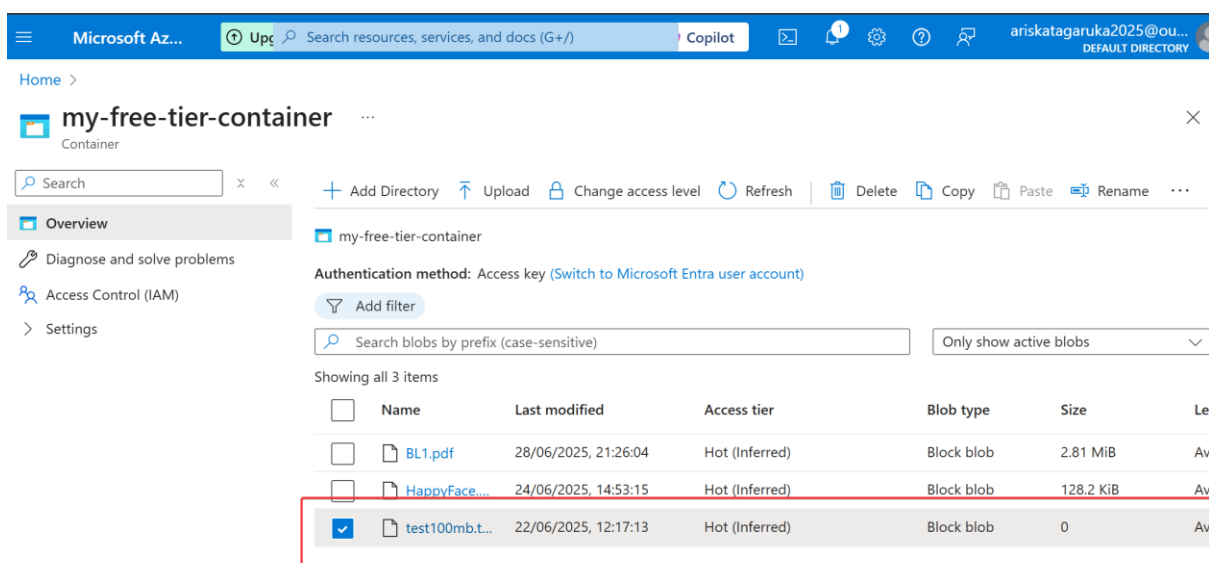


Figure 16: Azure Portal showing my-free-tier-container with test100mb.txt.

## 16. Integrate GCP Cloud Storage:

- Add to app.py:
- ```
from file_upload import upload_to_gcp # Import GCP upload function
```
- ```
@app.route('/upload/gcp', methods=['POST']) # Define GCP upload endpoint
```
- ```
def upload_gcp(): # GCP upload handler
```
- ```
 if 'file' not in request.files: # Check if file is included
```
- ```
        return 'No file provided', 400 # Return error if no file
```
- ```
 file = request.files['file'] # Get uploaded file
```
- ```
    file_path = f'temp/{file.filename}' # Define temporary file path
```
- ```
 cache_key = f'upload_gcp_{file.filename}' # Create cache key for file
```
- ```
    if get_cached(cache_key): # Check if file is already uploaded
```
- ```
 return 'File already uploaded', 200 # Return cached response
```
- ```
    file.save(file_path) # Save file to temporary path
```
- ```
 upload_to_gcp(file_path, 'my-free-tier-bucket', file.filename) # Upload to GCP
```
- ```
    set_cached(cache_key, 'Uploaded') # Cache upload status
```
- ```
 energy = estimate_energy(file.content_length / (1024 * 1024)) # Calculate energy usage
```
- ```
    return f'File uploaded to GCP: {file.filename}, Energy: {energy:.4f} kWh', 200 # Return success message
```
- In GCP Console: Create bucket my-free-tier-bucket (Standard class).
- Test:

```
curl -X POST -F "file=@test.txt"
http://localhost:5000/upload/gcp # Test GCP upload
```

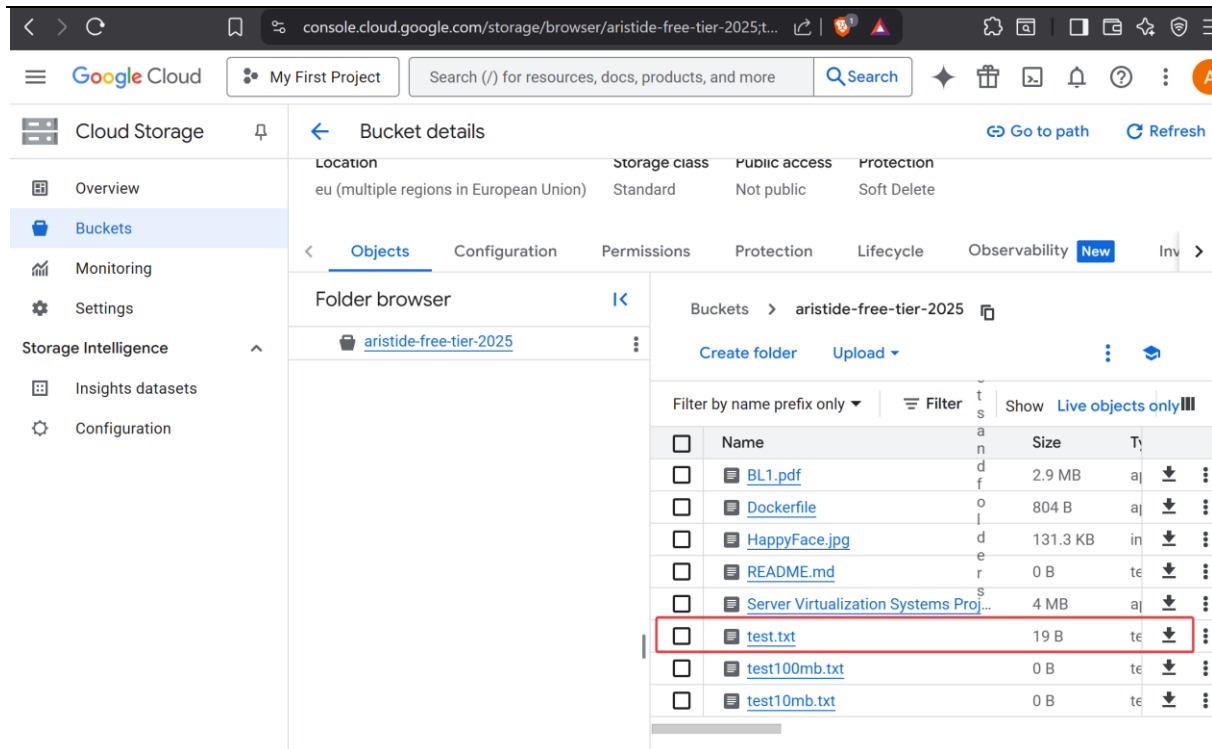


Figure 17: GCP Console showing my-free-tier-bucket with test.txt.

17. Add Mock Endpoints and Dashboard Data:

```

o Add to app.py:
o @app.route('/demo/block', methods=['GET']) # Define block
  storage demo endpoint
o def demo_block(): # Block storage demo handler
  return { # Return mock block storage data
o     'aws': 'EBS: 10GB volume attached for VM',
o     'azure': 'Managed Disk: 10GB for VM',
o     'gcp': 'Persistent Disk: 10GB for VM'
o   }, 200
o @app.route('/demo/file', methods=['GET']) # Define file
  storage demo endpoint
o def demo_file(): # File storage demo handler
  return { # Return mock file storage data
o     'aws': 'EFS: Shared folder created',
o     'azure': 'Azure Files: Shared folder created',
o     'gcp': 'Filestore: Shared folder created'
o   }, 200
o @app.route('/dashboard', methods=['GET']) # Define dashboard
  data endpoint
o def dashboard_data(): # Dashboard data handler
  return { # Return mock dashboard data
o     'usage': {'s3': 100, 'azure': 150, 'gcp': 200},
o     'savings': 500,
o     'alerts': ['AWS S3: 80% API limit reached'],
o     'energy': 0.05
o   }, 200
o @app.route('/provider/comparison', methods=['GET']) # Define
  provider comparison endpoint
o def comparison_data(): # Provider comparison handler

```

```
o     return [ # Return mock comparison data
o       {'name': 'AWS S3', 'cost': '$0.023/GB', 'freeTier':
'5GB', 'egress': 'Yes'},
o       {'name': 'Azure Blob', 'cost': '$0.0018/GB (Archive)',
'freeTier': '5GB', 'egress': 'Yes'},
o       {'name': 'GCP Cloud Storage', 'cost': '$0.020/GB',
'freeTier': '5GB', 'egress': 'No'}
    ], 200
```

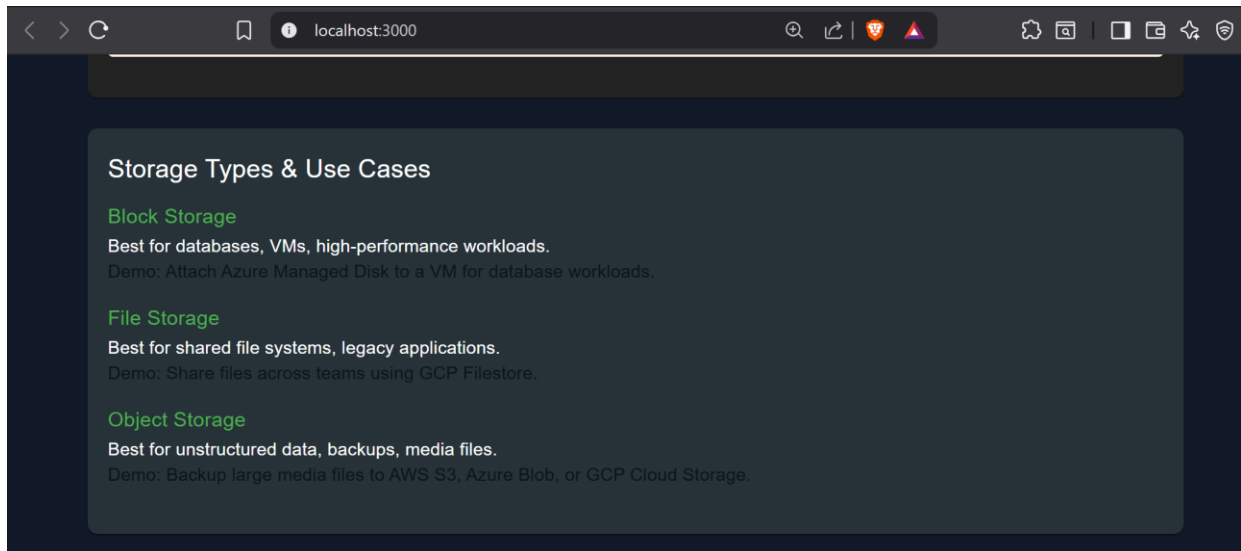


Figure 18: Browser showing mock endpoint responses in StorageInfo.js

Troubleshooting

- **CORS errors:** Ensure flask-cors is installed and enabled in app.py.
- **API failures:** Check backend logs (python app.py), verify bucket/container names.
- **Permission issues:** Confirm IAM roles (AmazonS3FullAccess, Storage Blob Data Contributor, Storage Admin).
- **GCP auth errors:** Ensure key.json is correctly placed in backend.

8. Application Containerization (6 hours, June 9, 2025)

Objective: Containerize Flask backend and React frontend.

Steps

18. Install Docker:

- o Download Docker Desktop from docker.com.
- o Enable Docker service (Linux: systemctl start docker).
- o Verify:

```
docker --version # Check Docker version
```

```

✓ TERMINAL
● PS C:\Users\akata\Desktop\Talent_Exhibition_Project_Aristide> docker -v
  Docker version 28.1.1, build 4eba377
  
```

Figure 19: Terminal showing docker --version, Docker Desktop UI.

19. Dockerfile for Flask:

- Create backend/Dockerfile:
 - FROM python:3.9-slim # Use Python 3.9 slim base image
 - WORKDIR /app # Set working directory
 - COPY . . # Copy all files to container
 - RUN pip install -r requirements.txt # Install Python dependencies
 - CMD ["python", "app.py"] # Run Flask app
- Create backend/.dockerignore:
 - venv # Ignore virtual environment
 - __pycache__ # Ignore Python bytecode
 - *.pyc # Ignore compiled Python files
 - temp # Ignore temporary files
 - key.json # Ignore GCP key
- Build and test:
 - cd backend # Navigate to backend directory
 - docker build -t flask-backend . # Build Docker image
 - docker run -p 5000:5000 --env-file .env -v \$(pwd)/key.json:/app/key.json flask-backend # Run container
- Verify at <http://localhost:5000>.

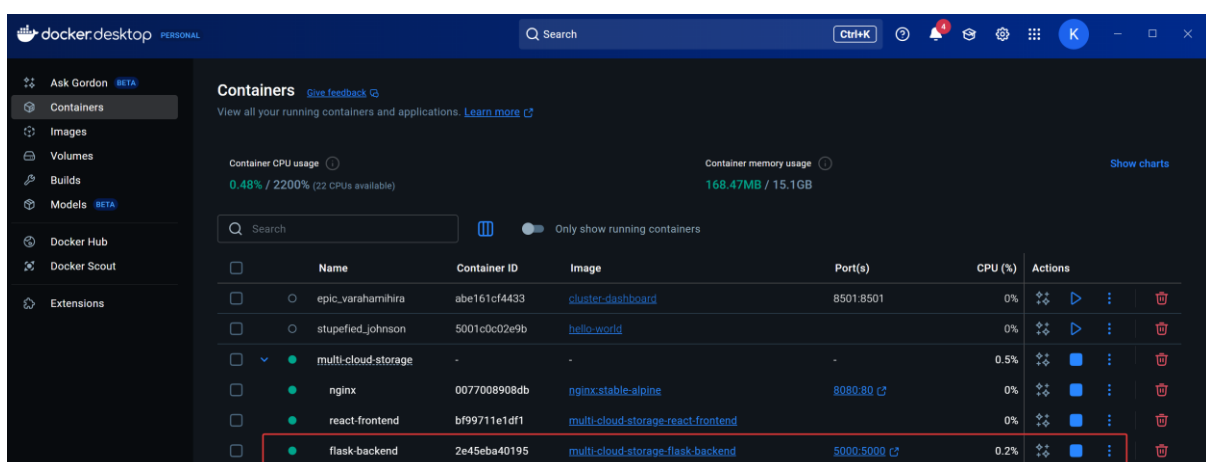


Figure 20: Docker desktop showing Flask backend on port:5000:5000

20. Dockerfile for React:

- Create storage-app/Dockerfile:
 - FROM node:16 # Use Node.js 16 base image
 - WORKDIR /app # Set working directory
 - COPY . . # Copy all files to container
 - RUN npm install # Install Node.js dependencies
 - RUN npm run build # Build React app
 - FROM nginx:alpine # Use Nginx alpine base image
 - COPY --from=0 /app/build /usr/share/nginx/html # Copy React build to Nginx
 - EXPOSE 80 # Expose port 80
 - CMD ["nginx", "-g", "daemon off;"] # Run Nginx
- Create storage-app/.dockerignore:
 - node_modules # Ignore Node.js dependencies
 - build # Ignore React build output
- Build and test:
 - cd storage-app # Navigate to React project
 - docker build -t react-frontend . # Build Docker image
 - docker run -p 80:80 react-frontend # Run container
- Verify at <http://localhost:80>.

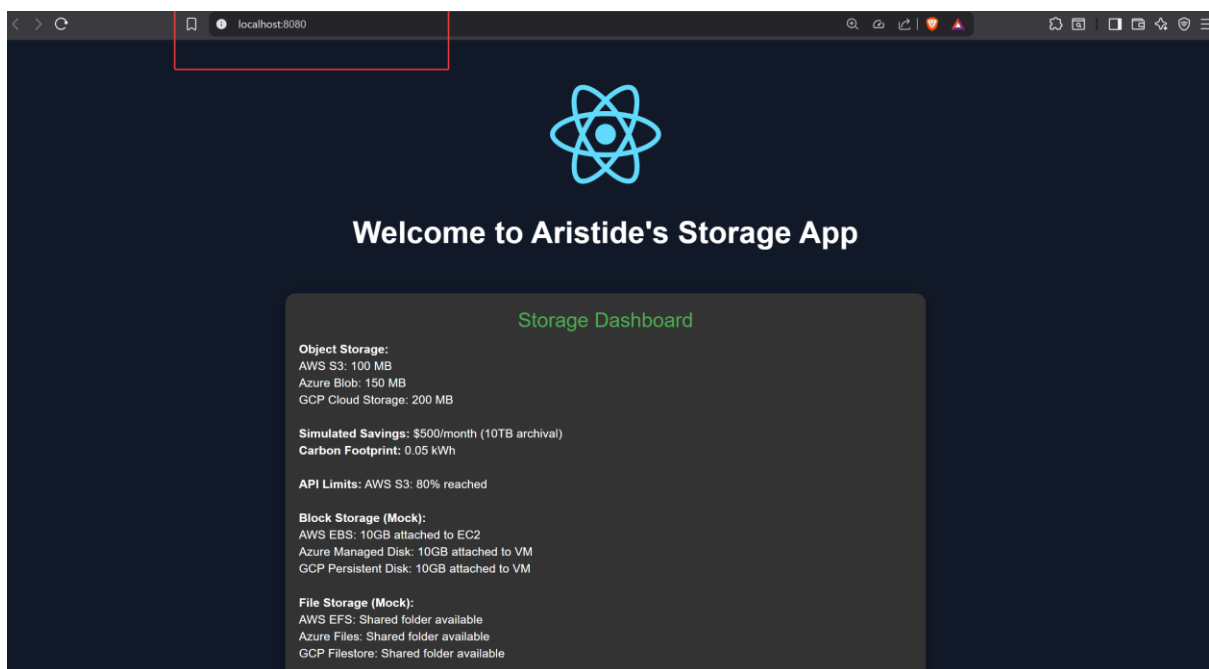


Figure 21: WebUI showing browser at <http://localhost:80>.

21. Test Containers Locally:

- Test Flask and React containers, verify API calls with Postman.

Troubleshooting

- **Port conflicts:** Check with `lsof -i :80`, change ports (e.g., `-p 8080:80`).
- **Build failures:** Verify dependencies in `requirements.txt` or `package.json`.
- **GCP auth:** Ensure `key.json` is mounted correctly in Docker.

9. Multi-Cloud Simulation with Load Balancing (6 hours, June 10, 2025)

Objective: Simulate multi-cloud environments with Docker Compose and Nginx.

Steps

22. Set Up Docker Compose:

```

o Create docker-compose.yml:
o version: '3' # Specify Docker Compose version
o services: # Define services
o   flask-backend: # Backend service
o     build: ./backend # Build from backend directory
o     ports: # Map ports
o       - "5000:5000" # Map host port 5000 to container port
o       5000
o     environment: # Set environment variables
o       - AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} # AWS access
o       key
o       - AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} # AWS
o       secret key
o       -
o       AZURE_STORAGE_CONNECTION_STRING=${AZURE_STORAGE_CONNECTION_STRI
o       NG} # Azure connection string
o       - GOOGLE_APPLICATION_CREDENTIALS=/app/key.json # GCP key
o       path
o     volumes: # Mount volumes
o       - ./backend:/app # Mount backend directory
o       - ./backend/key.json:/app/key.json # Mount GCP key
o   react-frontend: # Frontend service
o     build: ./storage-app # Build from storage-app directory
o     ports: # Map ports
o       - "80:80" # Map host port 80 to container port 80
o   redis: # Redis service
o     image: redis:latest # Use latest Redis image
o     ports: # Map ports
o       - "6379:6379" # Map host port 6379 to container port
o       6379
o   nginx: # Nginx service
o     image: nginx:latest # Use latest Nginx image
o     ports: # Map ports
o       - "8080:80" # Map host port 8080 to container port 80
o     volumes: # Mount volumes

```

- o `- ./nginx.conf:/etc/nginx/nginx.conf # Mount Nginx config`
- o `- ./storage-app/build:/usr/share/nginx/html # Mount React build`
- o `depends_on: # Specify dependencies`
 - `- flask-backend # Ensure backend starts first`
- o **Run:**

`docker-compose up --build # Build and start all services`
- o Access frontend at <http://localhost:8080>, backend at <http://localhost:5000>.

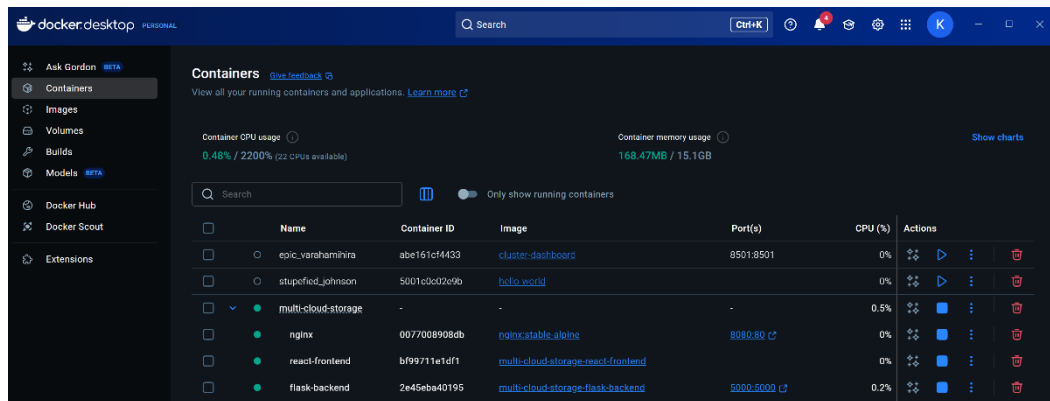


Figure 22: Docker desktop showing docker-compose up at <http://localhost:8080>.

23. Configure Nginx Load Balancing:

- o **Create nginx.conf:**
- o `events {} # Define Nginx events block (empty for default settings)`
- o `http { # Define HTTP server block`
- o `upstream backend { # Define upstream backend server`
- o `server flask-backend:5000; # Route to Flask backend on port 5000`
- o `}`
- o `server { # Define server block`
- o `listen 80; # Listen on port 80`
- o `location /api/ { # Route API requests`
- o `proxy_pass http://backend; # Forward to Flask backend`
- o `proxy_set_header Host $host; # Pass host header`
- o `proxy_set_header X-Real-IP $remote_addr; # Pass client IP`
- o `}`
- o `location / { # Route all other requests`
- o `root /usr/share/nginx/html; # Serve React build`
- o `try_files $uri /index.html; # Fallback to index.html for SPA`
- o `}`
- o `}`
- o `}`
- o Test file uploads via frontend, verify routing to `/upload/s3`, `/upload/azure`, `/upload/gcp`.

Troubleshooting

- **Nginx errors:** Check docker logs nginx, ensure flask-backend is running.
- **404 errors:** Verify nginx.conf paths and service names.
- **CORS issues:** Confirm flask-cors in app.py.

10. Sustainability and Alerts Implementation (8 hours, June 12, 2025)

Objective: Add energy estimation, lifecycle policies, and API usage alerts.

Steps

24. Energy Estimation Script:

- Create backend/energy_estimator.py:
- `def estimate_energy(file_size_mb):` # Function to estimate energy usage
- `# Assume 0.01 kWh/GB upload`
- `energy_kwh = (file_size_mb / 1024) * 0.01` # Convert MB to GB and calculate energy
- `return energy_kwh` # Return estimated energy in kWh

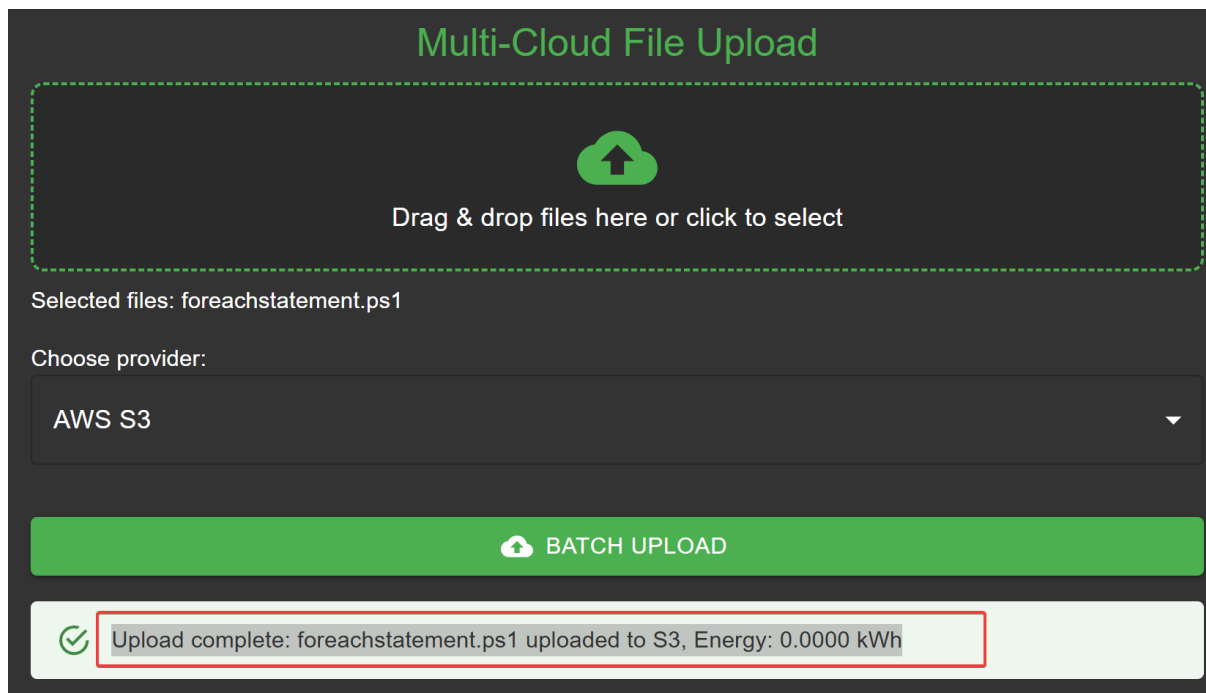
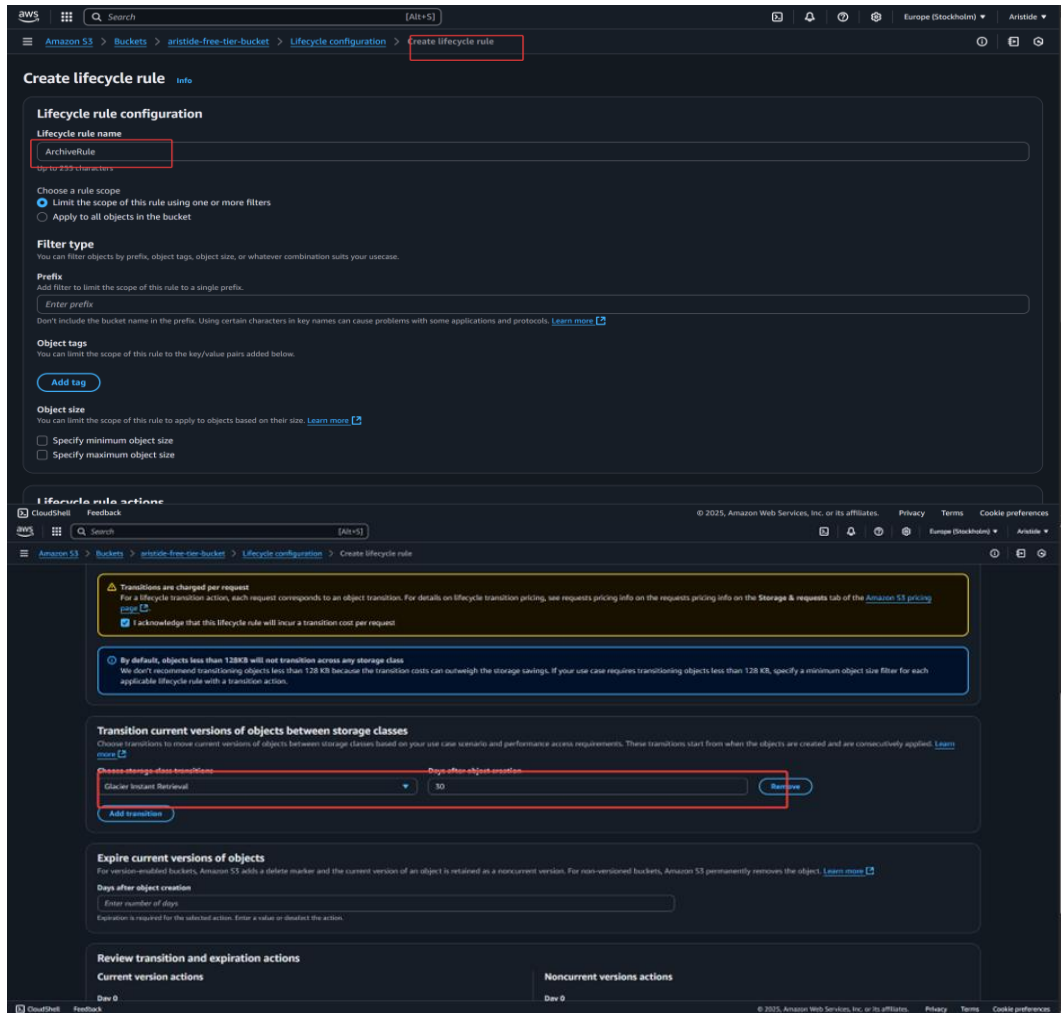


Figure 23: Browser showing upload response with energy estimation.

25. Lifecycle Policies:

- **AWS S3:**
 - In AWS Console: S3 > my-free-tier-bucket > Management > Create lifecycle rule: ArchiveRule, transition to Glacier after 30 days.



Create lifecycle rule

Lifecycle rule configuration

Lifecycle rule name
ArchiveRule

Choose a rule scope
☒ Limit the scope of this rule using one or more filters
☐ Apply to all objects in the bucket

Filter type
You can filter objects by prefix, object tags, object size, or whatever combination suits your usecase.

Prefix
Add filter to limit the scope of this rule to a single prefix.
Enter prefix:

Object tags
You can limit the scope of this rule to the key/value pairs added below.
Add tag

Object size
You can limit the scope of this rule to apply to objects based on their size.
☐ Specify minimum object size
☐ Specify maximum object size

Lifecycle rule actions

Transitions are charged per request
For a lifecycle transition action, each request corresponds to an object transition. For details on lifecycle transition pricing, see requests pricing info on the [Storage & requests tab of the Amazon S3 pricing page](#).
☒ I acknowledge that this lifecycle rule will incur a transition cost per request.

By default, objects less than 128 KB will not transition across any storage class.
We don't recommend transitioning objects less than 128 KB because the transition costs can outweigh the storage savings. If your use case requires transitioning objects less than 128 KB, specify a minimum object size filter for each applicable lifecycle rule with a transition action.

Transition current versions of objects between storage classes
Choose transitions to move current versions of objects between storage classes based on your use case scenario and performance access requirements. These transitions start from when the objects are created and are consecutively applied.

Choose storage class transitions
Days after object creation: 30
 Glacier Instant Retrieval

Expire current versions of objects
For version-enabled buckets, Amazon S3 adds a delete marker and the current version of an object is retained as a noncurrent version. For non-versioned buckets, Amazon S3 permanently removes the object.
 Days after object creation: Enter number of days.

Review transition and expiration actions
 Current version actions: Day 0
 Noncurrent versions actions: Day 0

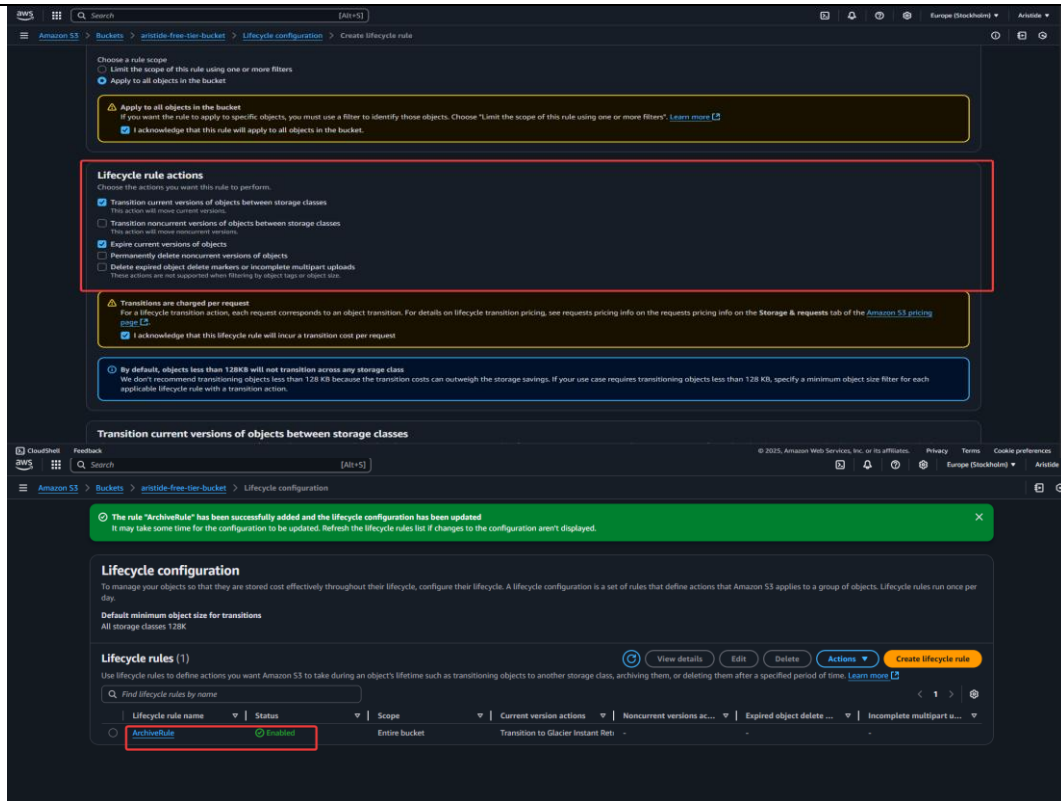
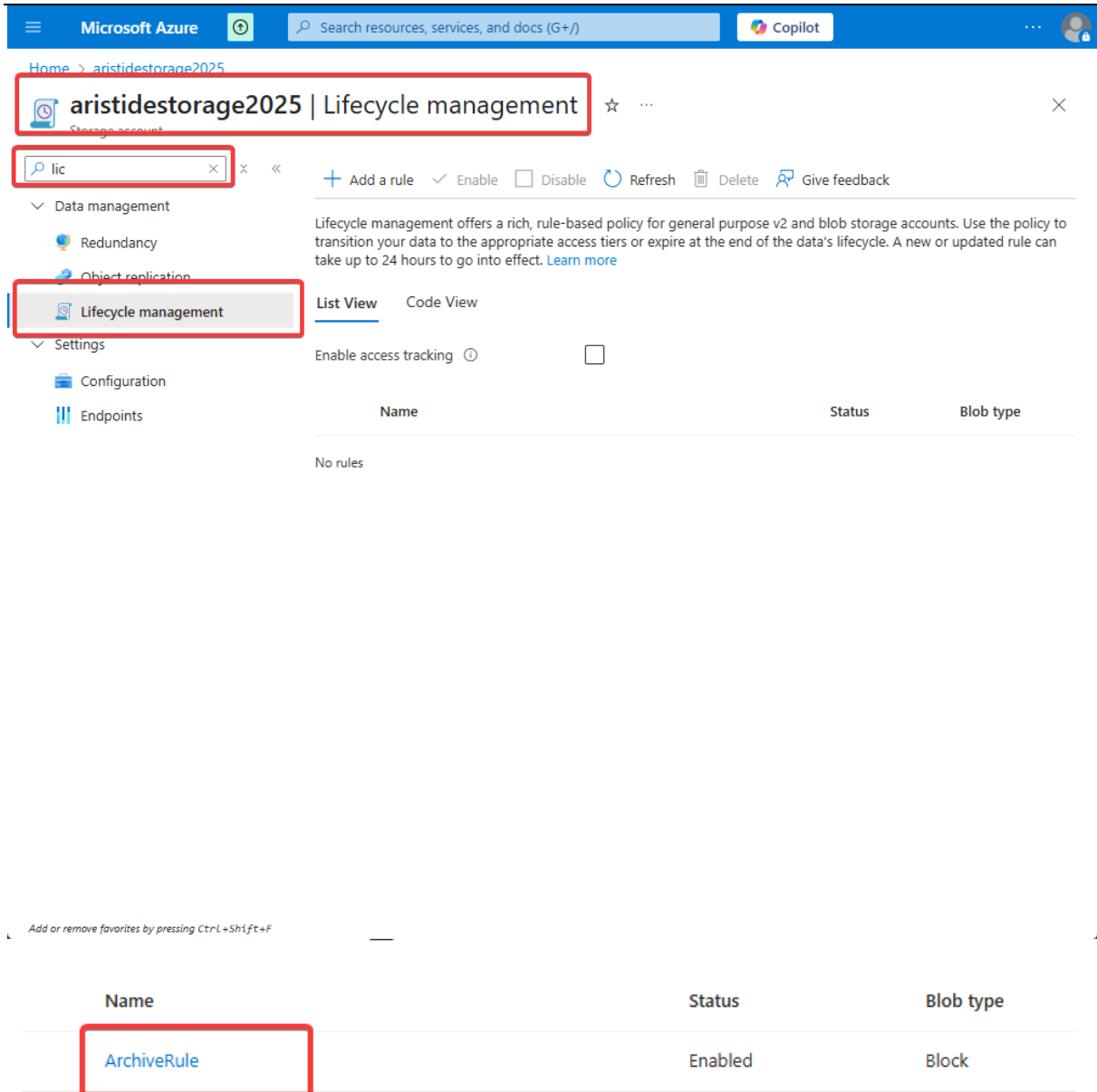


Figure 24:AWS Console showing lifecycle rule.

- **Azure Blob Storage:**
 - In Azure Portal: Storage Account > Lifecycle Management > Add rule: ArchiveRule, move to Archive after 90 days.



Home > aristidestorage2025

aristidestorage2025 | Lifecycle management

Storage account

lic

+ Add a rule ✓ Enable ☐ Disable ↻ Refresh 🗑 Delete 🗨 Give feedback

Data management
 Redundancy
 Object replication
Lifecycle management
 Settings
 Configuration
 Endpoints

Lifecycle management offers a rich, rule-based policy for general purpose v2 and blob storage accounts. Use the policy to transition your data to the appropriate access tiers or expire at the end of the data's lifecycle. A new or updated rule can take up to 24 hours to go into effect. [Learn more](#)

List View Code View

Enable access tracking ☐

Name	Status	Blob type
ArchiveRule	Enabled	Block

No rules

Add or remove favorites by pressing Ctrl+Shift+F

Figure 25: Azure Portal showing lifecycle rule.

- **GCP Cloud Storage:**
 - In GCP Console: Cloud Storage > my-free-tier-bucket > Lifecycle > Add rule: ArchiveRule, transition to Coldline after 60 days.

☰

Google Cloud

My First Project

Search (/) for resources, docs, products, and more

Cloud Storage

Overview

Buckets

Monitoring

Settings

Storage Intelligence

Insights datasets

Configuration

←

Add object lifecycle rule

After you add or edit a rule, it may take up to 24 hours to take effect.

- Select an action

☐ Set storage class to Nearline
Best for backups and data accessed less than once a month

☒ Set storage class to Coldline
Best for disaster recovery and data accessed less than once a quarter

☐ Set storage class to Archive
Best for long-term digital preservation of data accessed less than once a year

☐ Delete object

☐ Delete multi-part upload
Sets a time limit and removes unfinished or idle multi-part uploads

Continue
- Select object conditions

Create Cancel

Coldline if there are rules for both).

Rules [Add a rule](#) [Delete all](#)

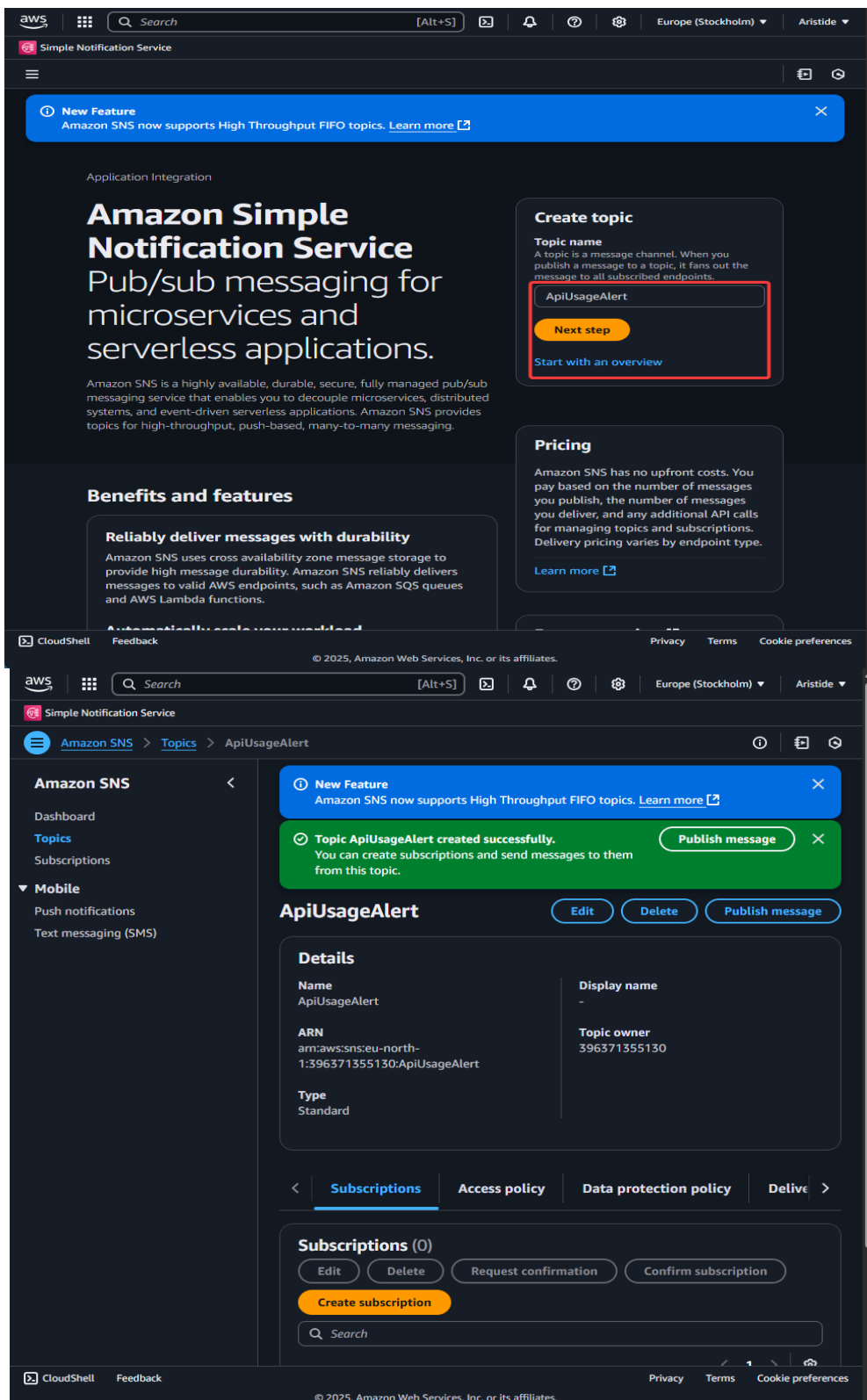
Action	Object condition	Works with
Set to Coldline	60+ days since object was created Name matches prefix 'ArchiveRule'	

Figure 26: GCP Console showing lifecycle rule.

26. API Usage Alerts:

- **AWS SNS/CloudWatch:**
 - In AWS Console: SNS > Topics > Create Topic: ApiUsageAlert, add email subscription.

- In CloudWatch: Create alarm for S3 NumberOfRequests > 16,000 (80% of 20,000 GET limit), notify ApiUsageAlert.



The screenshot displays the AWS Management Console for the Amazon Simple Notification Service (SNS). The top section shows the 'Create topic' page with the topic name 'ApiUsageAlert' entered. The bottom section shows the 'ApiUsageAlert' topic details page, including the 'Subscriptions' tab with a 'Create subscription' button.

Amazon Simple Notification Service
Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

Benefits and features

- Reliably deliver messages with durability**
Amazon SNS uses cross availability zone message storage to provide high message durability. Amazon SNS reliably delivers messages to valid AWS endpoints, such as Amazon SQS queues and AWS Lambda functions.

Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

ApiUsageAlert

[Next step](#)

[Start with an overview](#)

Pricing

Amazon SNS has no upfront costs. You pay based on the number of messages you publish, the number of messages you deliver, and any additional API calls for managing topics and subscriptions. Delivery pricing varies by endpoint type.

[Learn more](#)

Amazon SNS

Dashboard
Topics
Subscriptions
▼ Mobile
Push notifications
Text messaging (SMS)

ApiUsageAlert

[Edit](#) [Delete](#) [Publish message](#)

Details

Name ApiUsageAlert	Display name -
ARN arn:aws:sns:eu-north-1:396371355130:ApiUsageAlert	Topic owner 396371355130
Type Standard	

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Deliver](#)

Subscriptions (0)

[Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#)

[Create subscription](#)



Search

[Alt+S]

Europe (Stockholm)

Aristide

Simple Notification Service

Amazon SNS

Subscriptions

Create subscription

New feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Create subscription

Details

Topic ARN

arn:aws:sns:eu-north-1:396371355130:ApiUsageAlert

Protocol

The type of endpoint to subscribe

Email

Endpoint

An email address that can receive notifications from Amazon SNS.

katar711@school.lu

After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)

This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)

Send undeliverable messages to a dead-letter queue.

Cancel

Create subscription

CloudShell

Feedback

Privacy

Terms

Cookie preferences

© 2025, Amazon Web Services, Inc. or its affiliates.



Search

[Alt+S]

Europe (Stockholm)

Aristide

Simple Notification Service

Amazon SNS

Topics

ApiUsageAlert

Subscription: 7c91c5b9-a0dc-4702-8077-6d9271d8c...

New Feature

Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Subscription to ApiUsageAlert created successfully.

The ARN of the subscription is arn:aws:sns:eu-north-1:396371355130:ApiUsageAlert:7c91c5b9-a0dc-4702-8077-6d9271d8cb7d.

Subscription: 7c91c5b9-a0dc-4702-8077-6d9271d8cb7d

Edit

Delete

Details

ARN

arn:aws:sns:eu-north-1:396371355130:ApiUsageAlert:7c91c5b9-a0dc-4702-8077-6d9271d8cb7d

Status

Pending confirmation

Protocol

EMAIL

Endpoint

katar711@school.lu

Topic

ApiUsageAlert

Subscription Principal

arn:aws:iam::396371355130:root

Subscription filter policy

Redrive policy (dead-letter queue)

Subscription filter policy [Info](#)

This policy filters the messages that a subscriber receives.

No filter policy configured for this subscription.

CloudShell

Feedback

Privacy

Terms

Cookie preferences

© 2025, Amazon Web Services, Inc. or its affiliates.

The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, a search bar, and the region 'Europe (Stockholm)'. The left sidebar shows the 'Amazon S3' service, with 'Buckets' selected. The main content area displays 'Bucket metrics' for the bucket 'aristide-free-tier-bucket'. Under 'Request metrics', there is a 'Filters' section. A message states: 'You don't have any filters. To view request metrics, create a filter.' with a 'Create filter' button. The 'Create filter' dialog is open, showing the following fields:

- Name:** Filter name (This can't be edited after the filter is created). The input field contains 'AllRequests'.
- Scope:** Choose a filter scope. The radio button 'Limit the scope of this filter using prefix, object tags, and Access Point, or a combination of all three' is selected.
- Filter type:** Choose at least one filter type. The checkbox 'Prefix' is selected.
- Prefix:** Limit this filter to a single prefix. The input field contains 'AllRequests'.

At the bottom right of the dialog, there are 'Cancel' and 'Create filter' buttons.

The screenshot displays the AWS CloudWatch console during the 'Create alarm' process. The 'Specify metric and conditions' step is active, showing a 'Metric' section with a 'Graph' preview. A 'Select metric' button is visible. Below the main area, a 'Select metric' modal is open, showing a graph titled 'Untitled graph' and a list of metrics. The 'Storage Metrics' category is highlighted with a red box, showing 2 metrics. The modal also includes a search bar, filters, and buttons for 'Add math', 'Add query', 'Graph with SQL', and 'Graph search'.

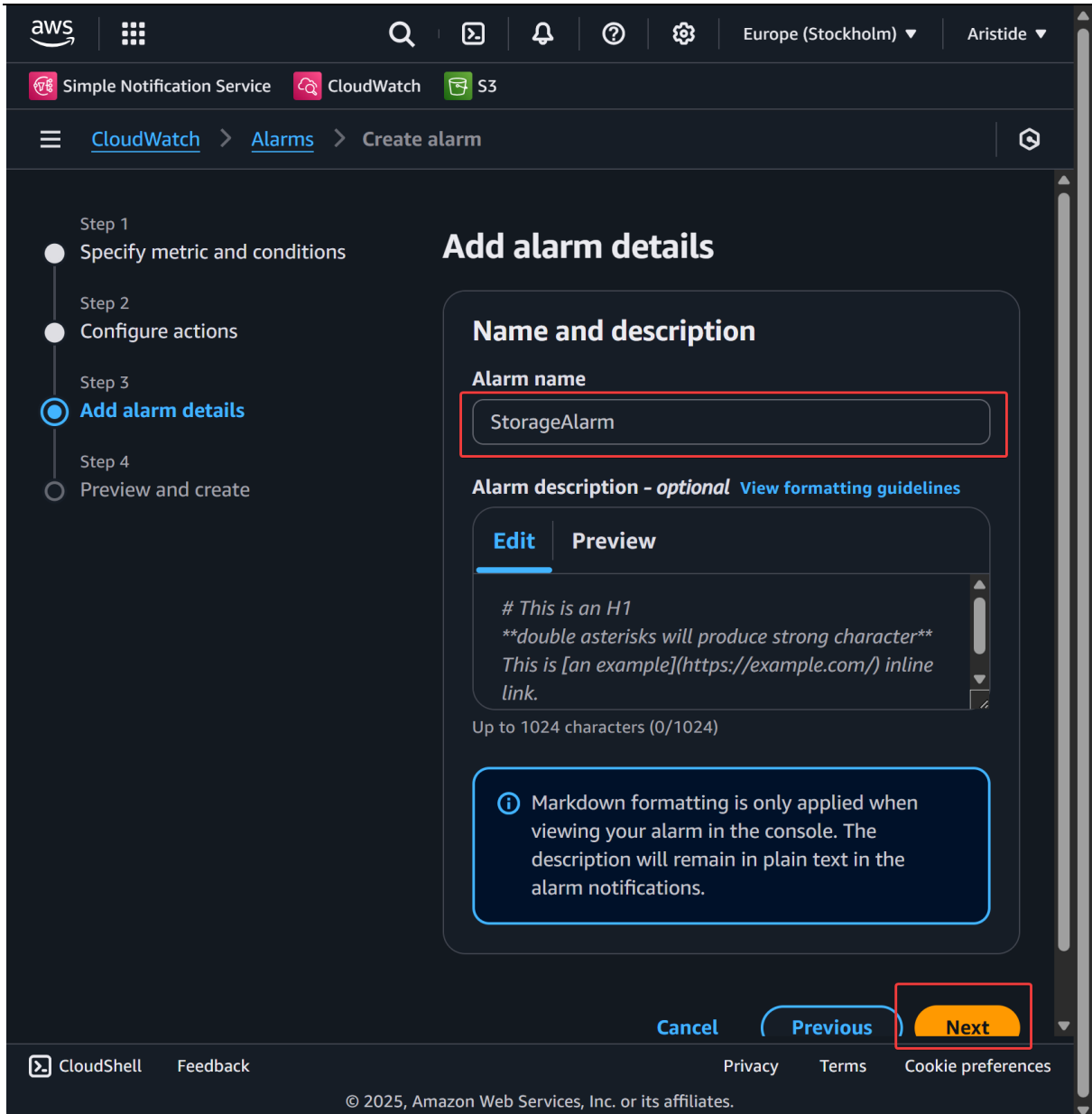
The image displays two screenshots of the AWS CloudWatch console interface for creating a new alarm.

Top Screenshot: "Create alarm" page

- Period:** 1 day
- Conditions:**
 - Threshold type:** Static (selected). Description: Use a value as a threshold.
 - Whenever NumberOfObjects is...**
 - Greater** (selected). Description: > threshold.
 - Greater/Equal: >= threshold
 - Lower/Equal: <= threshold
 - Lower: < threshold
 - than...** (highlighted with a red box): Define the threshold value. Value: 16000. Note: Must be a number.
- Buttons:** Cancel, Next (highlighted with a red box).

Bottom Screenshot: "Specify metric and conditions" page

- Steps:**
 - Step 1: Specify metric and conditions (selected)
 - Step 2: Configure actions
 - Step 3: Add alarm details
 - Step 4: Preview and create
- Metric:**
 - Graph:** This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 day. The graph shows a blue line for "NumberOfObjects" and a red dashed threshold line at 16,000.
 - Namespace:** AWS/S3
 - Metric name:** NumberOfObjects
 - BucketName:** aristide-free-tier-bucket
 - StorageType:** AllStorageTypes
 - Statistic:** Average
 - Period:** 1 day
- Buttons:** Edit



aws

Simple Notification Service CloudWatch S3

CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add alarm details

Step 4
Preview and create

Add alarm details

Name and description

Alarm name

StorageAlarm

Alarm description - optional [View formatting guidelines](#)

Edit Preview

This is an H1
 double asterisks will produce strong character
 This is [an example](https://example.com/) inline link.

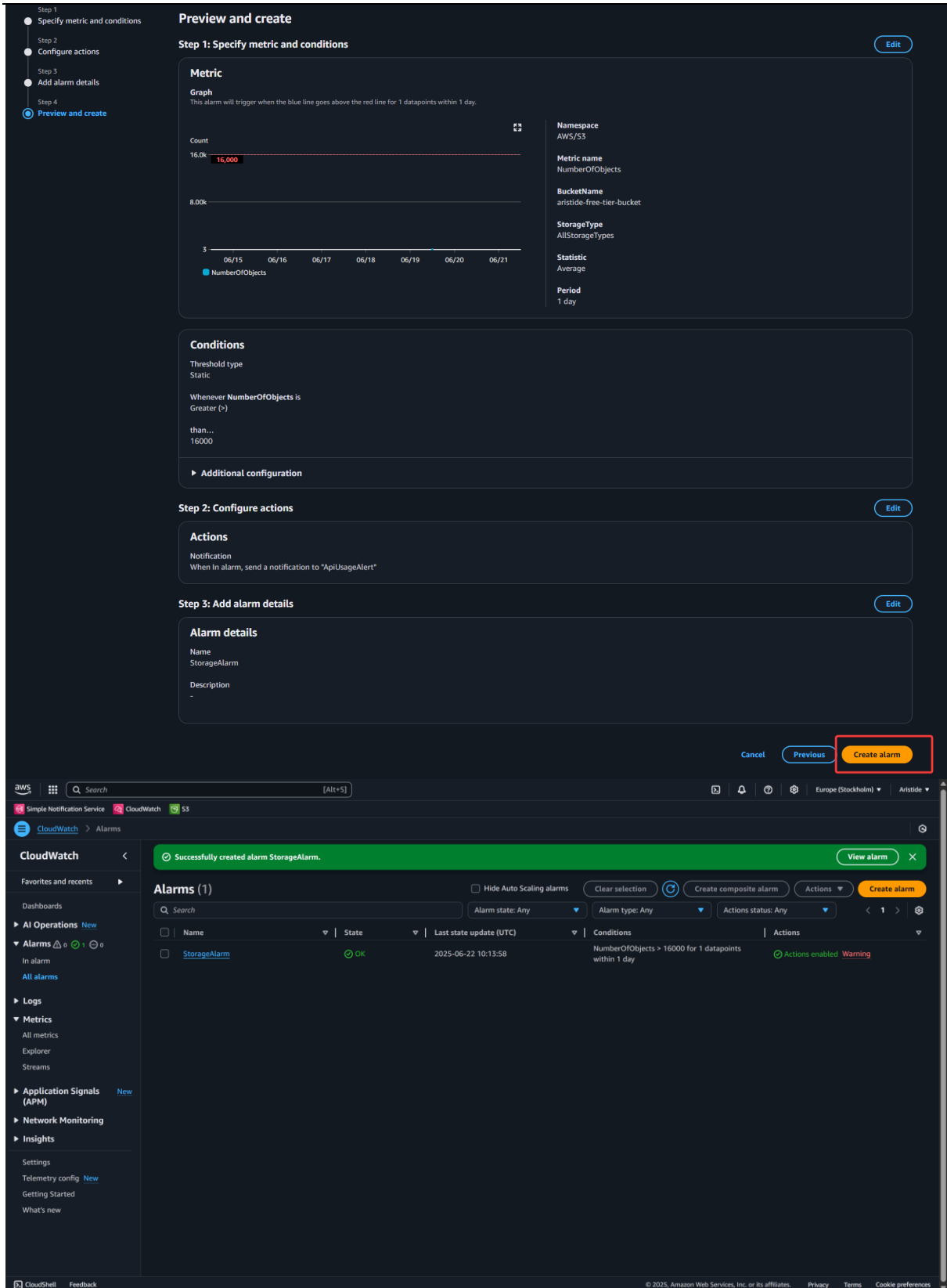
Up to 1024 characters (0/1024)

i Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.

Cancel Previous **Next**

CloudShell Feedback Privacy Terms Cookie preferences

© 2025, Amazon Web Services, Inc. or its affiliates.



Preview and create

Step 1: Specify metric and conditions Edit

Metric

Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 day.

Count

16.0k 16,000

8.00k

3

06/15 06/16 06/17 06/18 06/19 06/20 06/21

NumberOfObjects

Namespace
AWS/S3

Metric name
NumberOfObjects

BucketName
aristide-free-tier-bucket

StorageType
AllStorageTypes

Statistic
Average

Period
1 day

Conditions

Threshold type
Static

Whenever **NumberOfObjects** is
Greater (>)

than...
16000

► Additional configuration

Step 2: Configure actions Edit

Actions

Notification
When in alarm, send a notification to "ApiUsageAlert"

Step 3: Add alarm details Edit

Alarm details

Name
StorageAlarm

Description
-

Cancel Previous **Create alarm**

CloudWatch

Search [Alt+S]

Simple Notification Service CloudWatch S3

CloudWatch > Alarms

CloudWatch

Successfully created alarm StorageAlarm. View alarm

Alarms (1)

Hide Auto Scaling alarms Clear selection Create composite alarm Actions **Create alarm**

Search

Alarm state: Any Alarm type: Any Actions status: Any

Name	State	Last state update (UTC)	Conditions	Actions
StorageAlarm	OK	2025-06-22 10:13:58	NumberOfObjects > 16000 for 1 datapoints within 1 day	Actions enabled Warning

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 27: CloudWatch showing the creation of an alarm.

○ **Azure Monitor:**

- In Azure Portal: Storage Account > Alerts > Create alert rule: Total Transactions > 16,000, email notification.

The screenshot displays the Microsoft Azure portal interface for the 'aristidestorage2025' storage account. The left-hand navigation pane shows the 'Alerts' tab selected under the 'Monitoring' section. In the main content area, a 'Set up alert rules on this resource' section contains a 'Create alert rule' button. Below this, the 'Create an alert rule' form is visible, showing the 'Condition' tab. The 'Signal name' is set to 'Transactions'. Under 'Alert logic', the 'Threshold type' is 'Static', 'Aggregation type' is 'Total', 'Value is' is 'Greater than', 'Unit' is 'Count', and the 'Threshold' is set to '16000'. A 'Preview' section on the right shows a graph of 'Transactions (Sum)' over the last 6 hours, with a value of 16. The bottom of the form has buttons for 'Review + create', 'Previous', and 'Next: Actions >'.

Microsoft Azure Upgrade Search resources, services, and docs (G+)

Home > **Create an alert rule**

Scope Condition **Actions** Details Tags Review + create

An action group is a set of actions that can be applied to an alert rule. [Learn more](#)

Select actions

☐ Use quick actions (preview)
Select one or more of the quick actions.

☒ Use action groups
Add an existing action group or create a new one.

☐ None

Action groups

Action group name Contains actions

No action group selected yet

Select action groups

Select action groups

Select up to five action groups to attach to this rule.

+ Create action group

Subscription Azure subscription 1

Search

Action group name ↑↓ Resource group ↑↓ Contains actions Location ↑↓

No results to display

Review + create Previous **Next: Details >** Select

Microsoft Azure Upgrade Search resources, services, and docs (G+)

Home > Create an alert rule > **Create action group**

Basics Notifications Actions Tags **Review + create**

This is a summary of your action group. Please review to ensure the information is correct and consider [Azure Monitoring Pricing](#) and the [Azure Privacy Statement](#).

Basics

Subscription	Azure subscription 1	
Resource group	StorageDemoRG	
Region	global	
Action group name	NewActionGroupAlert	
Display name	TestAlert	

Notifications

Notification type	Name	Selected
Email/SMS message/Push/Voice	Aristide	Email

Actions

None

Tags

None

Create Previous

Figure 28: Azure Portal showing the creation alert rule.

- **GCP Cloud Monitoring:**
 - In GCP Console: Monitoring > Alerting > Create policy: storage.googleapis.com/api/request_count > 80% quota, email notification.

The screenshot displays the Google Cloud Monitoring console interface. The top navigation bar includes the Google Cloud logo, project name 'My First Project', and a search bar. The left sidebar shows the 'Monitoring' section with 'Alerting' highlighted. The main content area is titled 'Alerting' and includes a '+ Create policy' button. Below this, there are sections for 'Summary' (showing 0 incidents firing, 0 acknowledged, and 0 policies), 'Incidents' (a table with no rows), 'Snoozes' (a table with no rows), and 'Policies' (a table with no rows). A 'Recommended for you' sidebar on the right lists 'Alerting overview', 'Quickstart', and 'Use cases for Monitoring'. The bottom section shows the 'Create alerting policy' workflow. Under 'ALERT CONDITIONS', 'GCS Bucket - Request count' is selected. The 'Add filters' section is empty. The 'Transform data' section is expanded, showing 'Within each time series' with a 'Rolling window' of '1 min' and a 'Rolling window function' of 'rate'. The 'Across time series' section is also visible. On the right, a chart shows 'GCS Bucket - Request count' over time, with a table of values for 'method' (ListObjects, WriteObject) and 'Value' (0.017 /s). The 'Estimated monthly cost' is shown as '\$0.00' (coming 2026). At the bottom, a notification states '1 file successfully uploaded' and a status bar shows 'Uploads and My First Project operations' as 'Complete'.

Google Cloud | My First Project | alert

Create alerting policy | + Add alert condition | Delete alert condition | View Code

ALERT CONDITIONS

- GCS Bucket - Request count
 - Configure trigger

ALERT DETAILS

- Notifications and name
- Review alert

Configure alert trigger

1 hour 6 hours 1 day 7 days 30 days Custom

Condition Types

- ☒ **Threshold**
Condition triggers if a time series rises above or falls below a value for a specific duration window
- ☐ Metric absence
Condition triggers if any time series in the metric has no data for a specific duration window
- ☐ Forecast [Preview](#)
Condition triggers if any timeseries in the metric is projected to cross the threshold in the near future.

Alert trigger: Any time series violates | Threshold position: Above threshold

Threshold value: 0.1 /s

Advanced Options

Condition name *: GCS Bucket - Request count

[Next](#)

GCS Bucket - Request count

0.12/s

0.1/s

0.08/s

0.06/s

0.04/s

0.02/s

0

UTC+2 10:20 AM 10:30 AM 10:40 AM 10:50 AM 11:00 AM

Filter Enter property name or value

method	Value
ListObjects	0.017 /s
WriteObject	0.017 /s

Estimated monthly cost [Coming 2026](#)

~~\$0.10~~ \$0.00

Pricing will be introduced in 2026. Estimates do not

[Create Policy](#) [Provide feedback](#) [Cancel](#)

Uploads and My First Project operations

[Dockerfile](#) [Complete](#)

Google Cloud | My First Project | alert

Create alerting policy | + Add alert condition | Delete alert condition | View Code

ALERT CONDITIONS

- GCS Bucket - Request count
 - Configure trigger

ALERT DETAILS

- Notifications and name
- Review alert

Configure notifications and finalize alert

Configure notifications Recommended

☒ Use notification channel

Notification Channels: AristideEmailNotification

Notification subject line

We recommend that you create multiple notification channels for redundancy purposes. Google has no control of many of the delivery systems after we have passed the notification to that system. Additionally, a single Google service supports Cloud Console Mobile App, PagerDuty, Webhooks, and Slack. If you use one of these notification channels, then use email, SMS, or Pub/Sub as the redundant channel.

[Learn more](#)

☒ **Notify on incident closure**

Incident autoclose duration: 7 days

If data is absent, select a duration after which incident will automatically close

Application labels

Linking alert policies to App Hub will enable alert policies, and alerts, to be displayed within App Hub pages, and will help link alerts back to the application, without requiring that you create the alert. [Learn more](#)

[Create Policy](#) [Provide feedback](#) [Cancel](#)

Uploads and My First Project operations

[Dockerfile](#) [Complete](#)

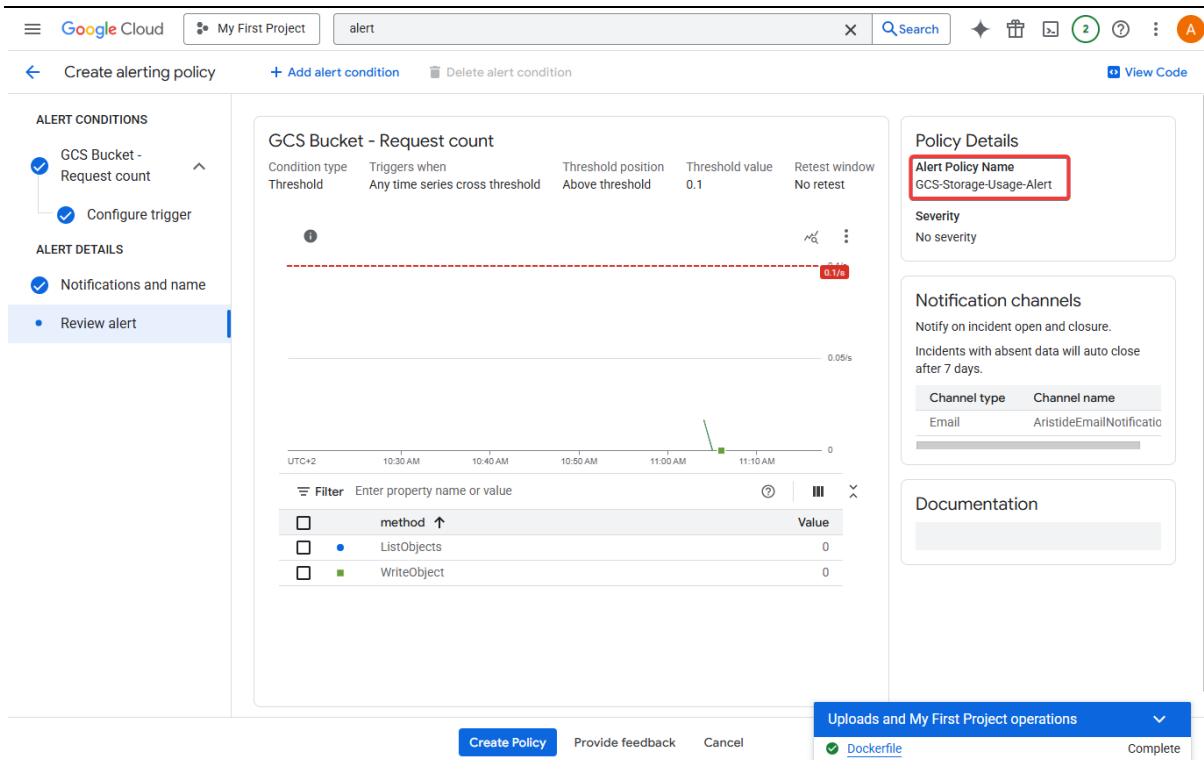


Figure 29: GCP Console showing monitoring policy.

Troubleshooting

- **Alerts not triggering:** Verify thresholds, email subscriptions.
- **Lifecycle issues:** Check rule application in consoles.

11. Testing File Operations and Demo Scenarios (6 hours, June 15, 2025)

Objective: Validate file operations, mock scenarios, and alerts.

Steps

27. Test File Operations:

- Upload 10MB file to AWS S3, Azure Blob, GCP Cloud Storage via frontend.
- Download file using provider-generated links (e.g., S3 pre-signed URL).
- Share file link via API.
- Test mock endpoints (/demo/block, /demo/file) via StorageInfo.js.

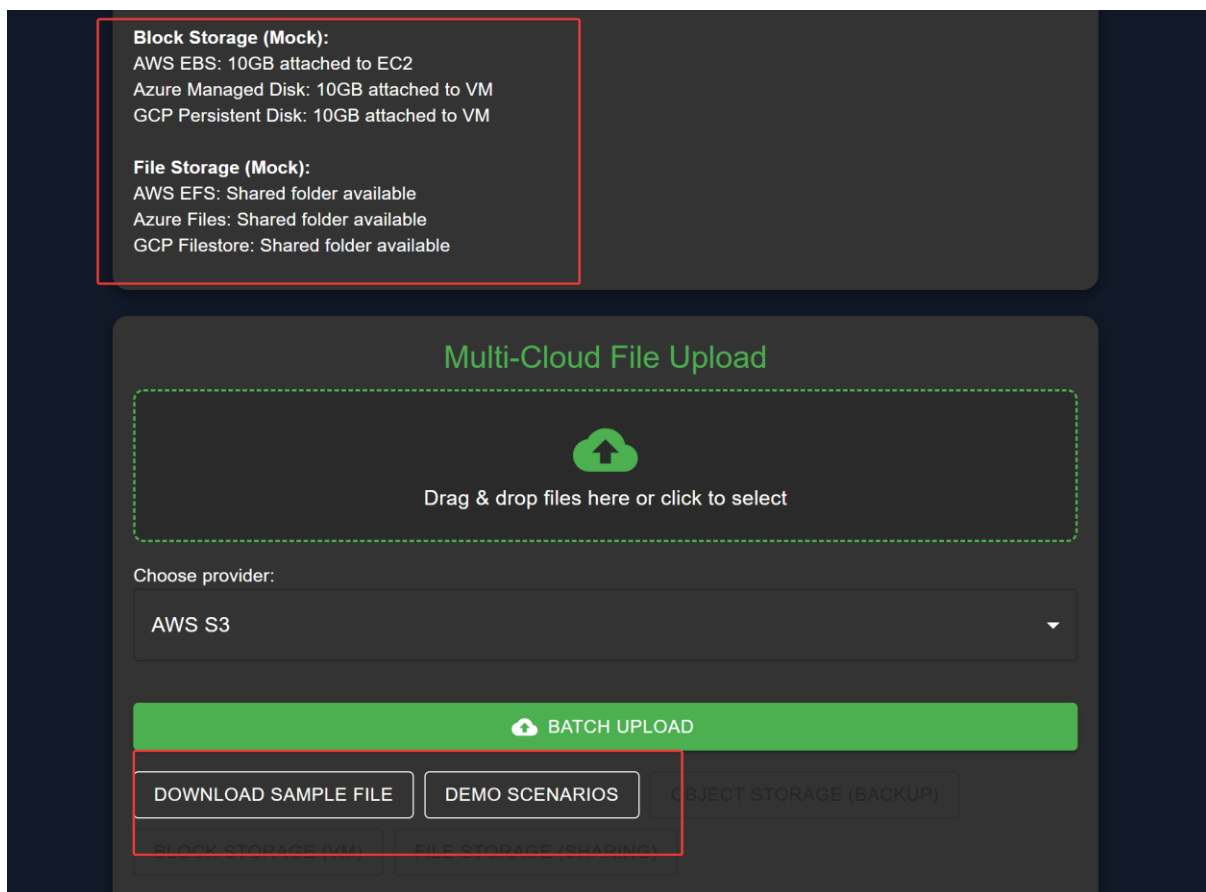


Figure 30: Browser showing upload/download and mock scenario outputs.

28. Test Load Balancing and Alerts:

- Send multiple upload requests, verify Nginx routing.
- Simulate high request counts to trigger alerts (e.g., 16,000 requests).
- Confirm email notifications.

Troubleshooting

- **Upload failures:** Check docker logs flask-backend for errors.
 - **Alerts not received:** Verify console configurations.
-

12. Performance Optimization

Objective: Implement caching and batch API calls.

Steps

29. Local Caching with Redis:

- Install Redis dependency (already in requirements.txt).
- Ensure Redis is included in docker-compose.yml.

☐	Name	Container ID	Image	Port(s)	CPU (%)	Actions
☐	○ epic_varahamihira	abe161cf4433	cluster-dashboard	8501:8501	0%	
☐	○ stupefied_johnson	5001c0c02e9b	hello-world		0%	
☐	▼ ● multi-cloud-storage	-	-	-	0.46%	
☐	● nginx	0077008908db	nginx:stable-alpine	8080:80 ↗	0%	
☐	● react-frontend	bf99711e1df1	multi-cloud-storage-react-frontend		0%	
☐	● flask-backend	2e45eba40195	multi-cloud-storage-flask-backend	5000:5000 ↗	0.2%	
☐	● redis-1	e06970437521	redis:7	6379:6379 ↗	0.26%	

Figure 31: Docker desktop showing Redis container running.

30. Batch API Calls:

- Update app.py:
- `@app.route('/upload/batch', methods=['POST'])` # Define batch upload endpoint
- `def upload_batch():` # Batch upload handler
- `if 'files' not in request.files:` # Check if files are included
- `return 'No files provided', 400` # Return error if no files
- `files = request.files.getlist('files')` # Get list of uploaded files
- `responses = []` # Initialize response list
- `for file in files:` # Iterate through files
- `file_path = f'temp/{file.filename}'` # Define temporary file path
- `cache_key = f'upload_s3_{file.filename}'` # Create cache key for file
- `if get_cached(cache_key):` # Check if file is already uploaded
- `responses.append(f'File {file.filename} already uploaded')` # Add cached response
- `continue` # Skip to next file
- `file.save(file_path)` # Save file to temporary path
- `upload_to_s3(file_path, 'my-free-tier-bucket', file.filename)` # Upload to S3
- `set_cached(cache_key, 'Uploaded')` # Cache upload status
- `responses.append(f'Uploaded {file.filename} to S3')` # Add success response

```
return {'results': responses}, 200 # Return all responses
```

```
o Update FileUpload.js:
o import React, { useState } from 'react'; // Import React and
  useState hook
o import axios from 'axios'; // Import axios for API calls
o function FileUpload() { // Define FileUpload component
o   const [files, setFiles] = useState([]); // State for
    selected files
o   const [provider, setProvider] = useState('s3'); // State for
    selected cloud provider
o   const [status, setStatus] = useState(''); // State for
    upload status message
o   const handleFileChange = (e) => { // Handle file input
    change
o     setFiles(e.target.files); // Update files state
o   };
o   const handleUpload = async () => { // Handle file upload
o     if (files.length === 0) { // Check if files are selected
o       setStatus('No files selected'); // Set error status
o       return;
o     }
o     setStatus('Uploading...'); // Set uploading status
o     try { // Handle API call
o       const formData = new FormData(); // Create FormData for
        file upload
o       for (let i = 0; i < files.length; i++) { // Loop through
        files
o         formData.append('files', files[i]); // Append each
        file
o       }
o       const endpoint = files.length > 1 ? '/api/upload/batch' :
        `/api/upload/${provider}`; // Choose endpoint
o       const response = await axios.post(endpoint, formData);
        // Send POST request
o       setStatus(files.length > 1 ?
        response.data.results.join(', ') : response.data); // Set
        status
o     } catch (error) { // Handle errors
o       setStatus(`Upload failed: ${error.message}`); // Set
        error status
o     }
o   };
o   return ( // Render JSX
o     <div className="m-4 bg-white p-4 rounded shadow"> //
        Container with styling
o       <h2 className="text-xl font-semibold text-gray-800">File
        Upload</h2> // Section title
o       <select
o         value={provider} // Bind provider state
o         onChange={(e) => setProvider(e.target.value)} //
        Update provider state
o         className="mb-2 border rounded p-2" // Styling for
        dropdown
o       >
o         <option value="s3">AWS S3</option> // AWS option
o         <option value="azure">Azure Blob</option> // Azure
        option
```

```

o      <option value="gcp">GCP Cloud Storage</option> // GCP
    option
o      </select>
o      <input
o          type="file" // File input
o          multiple // Allow multiple file selection
o          onChange={handleFileChange} // Bind file change
    handler
o          className="mb-2 border rounded p-2" // Styling for
input
o      />
o      <button
o          onClick={handleUpload} // Bind upload handler
o          className="bg-blue-500 text-white p-2 rounded hover:bg-
blue-600" // Styling for button
o      >
o          Upload Files // Button text
o      </button>
o      <p className="mt-2 text-gray-700">{status}</p> //
Display status message
o      </div>
o      );
o      }
    export default FileUpload; // Export FileUpload component

```

Troubleshooting

- **Redis failures:** Verify Redis container (docker ps).
- **Batch upload issues:** Test with small files, check backend logs (docker logs flask-backend).

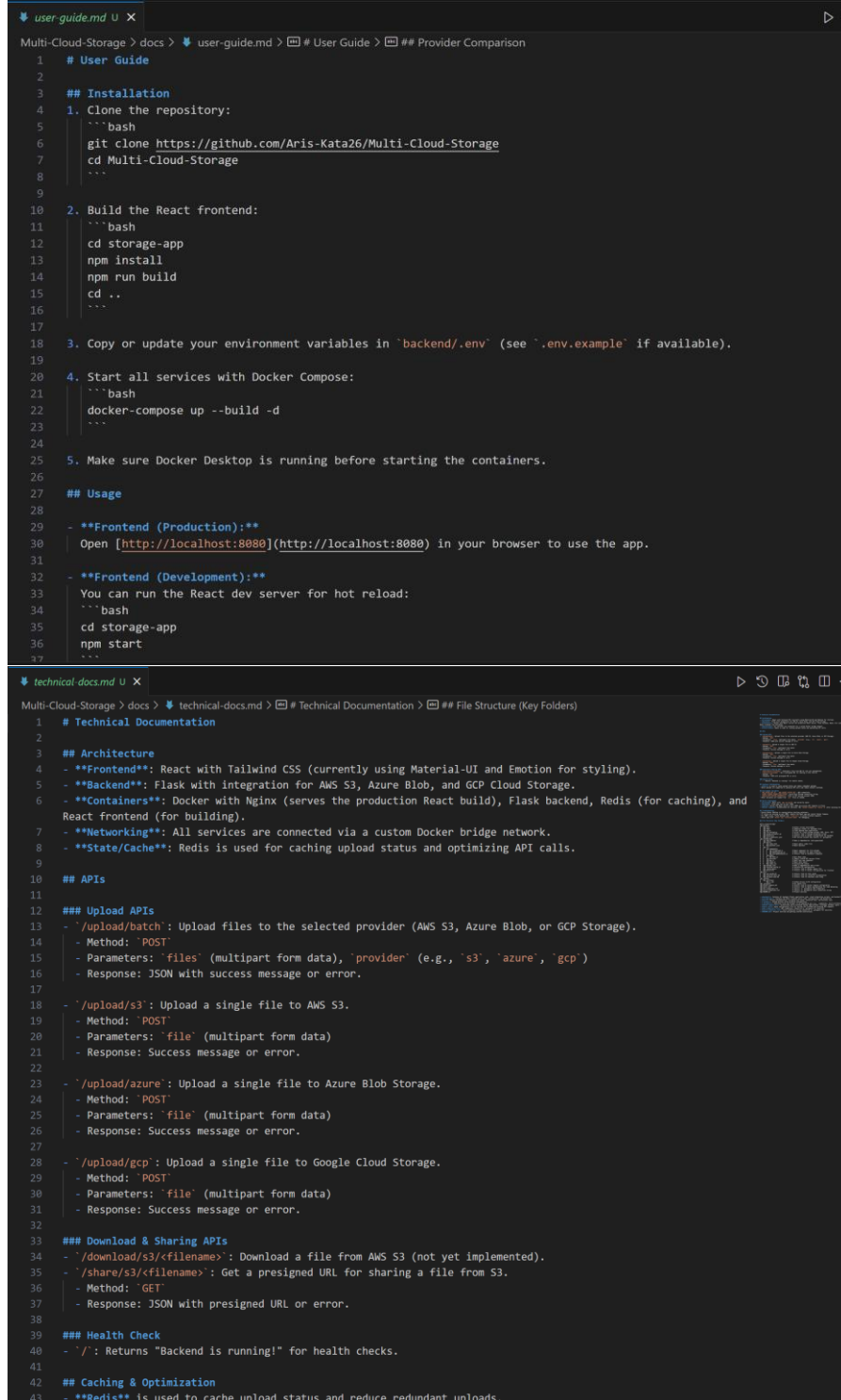
13. Documentation and Presentation (9 hours, June 17, 2025)

Objective: Create comprehensive documentation and presentation materials.

Steps

31. User Guide and Technical Docs:

- Create docs/user-guide.md:
 - # User Guide
 - ## Installation
 - 1. Clone repository: ``git clone https://github.com/yourusername/Multi-Cloud-Storage.git`` # Clone project
 - 2. Set environment variables in ``backend/.env``:
`AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY,`
`AZURE_STORAGE_CONNECTION_STRING,`
`GOOGLE_APPLICATION_CREDENTIALS.` # Configure credentials
 - 3. Place GCP key in ``backend/key.json``. # Add GCP service account key
 - 4. Run: ``docker-compose up --build`` # Build and start services
 - ## Usage
 - - Access frontend: `http://localhost:8080` # Open web interface
 - - Upload files to AWS, Azure, GCP. # Use file upload component
 - - View dashboard, storage info, and provider comparison. # Explore UI components
 - - Run mock block/file storage scenarios. # Test demo endpoints
- Create docs/technical-docs.md:
 - # Technical Documentation
 - ## Architecture
 - - ****Frontend****: React with Tailwind CSS # Responsive UI framework
 - - ****Backend****: Flask with AWS S3, Azure Blob, GCP Cloud Storage APIs # API server
 - - ****Containers****: Docker with Nginx, Redis # Containerized services
 - ## APIs
 - - ``/upload/s3`, `/upload/azure`, `/upload/gcp``: File uploads # Upload endpoints
 - - ``/upload/batch``: Batch upload # Batch upload endpoint
 - - ``/demo/block`, `/demo/file``: Mock scenarios # Demo endpoints
 - - ``/dashboard``: Usage, savings, alerts, energy # Dashboard data
 - - ``/provider/comparison``: Provider comparison data # Comparison data



```

user-guide.md
Multi-Cloud-Storage > docs > user-guide.md > # User Guide > ## Provider Comparison
1 # User Guide
2
3 ## Installation
4 1. Clone the repository:
5 ```bash
6 git clone https://github.com/Aris-Kata26/Multi-Cloud-Storage
7 cd Multi-Cloud-Storage
8 ```
9
10 2. Build the React frontend:
11 ```bash
12 cd storage-app
13 npm install
14 npm run build
15 cd ..
16 ```
17
18 3. Copy or update your environment variables in `backend/.env` (see `.env.example` if available).
19
20 4. Start all services with Docker Compose:
21 ```bash
22 docker-compose up --build -d
23 ```
24
25 5. Make sure Docker Desktop is running before starting the containers.
26
27 ## Usage
28
29 - **Frontend (Production):**
30 Open [http://localhost:8080](http://localhost:8080) in your browser to use the app.
31
32 - **Frontend (Development):**
33 You can run the React dev server for hot reload:
34 ```bash
35 cd storage-app
36 npm start
37 ```
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
259
```

- o - Leveraging GCP's no-egress-fee policy. # Reduce data transfer costs
 - o - Setting budget alerts at \$800 for \$1,000/month budget. # Monitor spending
- Energy reduction: 20-30% via archival tiers, supporting ESG compliance. # Sustainability benefits

```
business-case.md U X
Multi-Cloud-Storage > docs > business-case.md > # Business Case Study > ## Live Demonstration & Alerts
1 # Business Case Study
2
3 ## Executive Summary
4 A company managing 10TB of data achieves significant cost savings, operational flexibility, and sustainability
  improvements by adopting a multi-cloud storage strategy using AWS S3, Azure Blob, and Google Cloud Storage.
5
6 ## Key Benefits
7
8 - **Cost Optimization**:
9   By moving 80% of infrequently accessed data (8TB) to Azure Archive ($0.0018/GB), monthly storage costs drop
  from $184 (S3 Standard at $0.023/GB) to $14.40. The remaining 2TB stays on AWS S3 for high availability and
  performance.
10
11 - **Leveraging GCP**:
12   GCP's no-egress-fee policy for data transfers allows the business to move and access data across clouds
  without incurring additional costs, supporting hybrid and multi-cloud workflows.
13
14 - **Budget Alerts**:
15   Automated budget alerts are set at $800 for a $1,000/month budget, enabling proactive cost management and
  preventing overruns.
16
17 - **Energy Reduction & ESG Compliance**:
18   Using archival and cold storage tiers reduces energy consumption by 20-30%, supporting ESG goals. Estimated
  carbon footprint reduction is 0.05 kWh per operation.
19
20 - **Operational Resilience**:
21   Multi-cloud storage ensures data redundancy and business continuity. If one provider experiences downtime,
  data remains accessible via other clouds.
22
23 - **Vendor Flexibility**:
24   Avoids vendor lock-in by distributing data and workloads across AWS, Azure, and GCP, allowing the business to
  negotiate better rates and adapt to changing requirements.
25
26 ## Implementation Overview
27
28 - **Architecture**:
29   The solution uses a React frontend and Flask backend, containerized with Docker. Nginx serves the frontend,
  and Redis provides caching for performance.
30
31 - **APIs**:
32   Unified upload and batch endpoints allow seamless file transfers to any provider.
33
34 - **Monitoring**:
35   Dashboard displays usage, savings, and energy metrics. Budget alerts and usage thresholds are integrated.
```

Figure 33: VS code showing business case

14. Technical Architecture

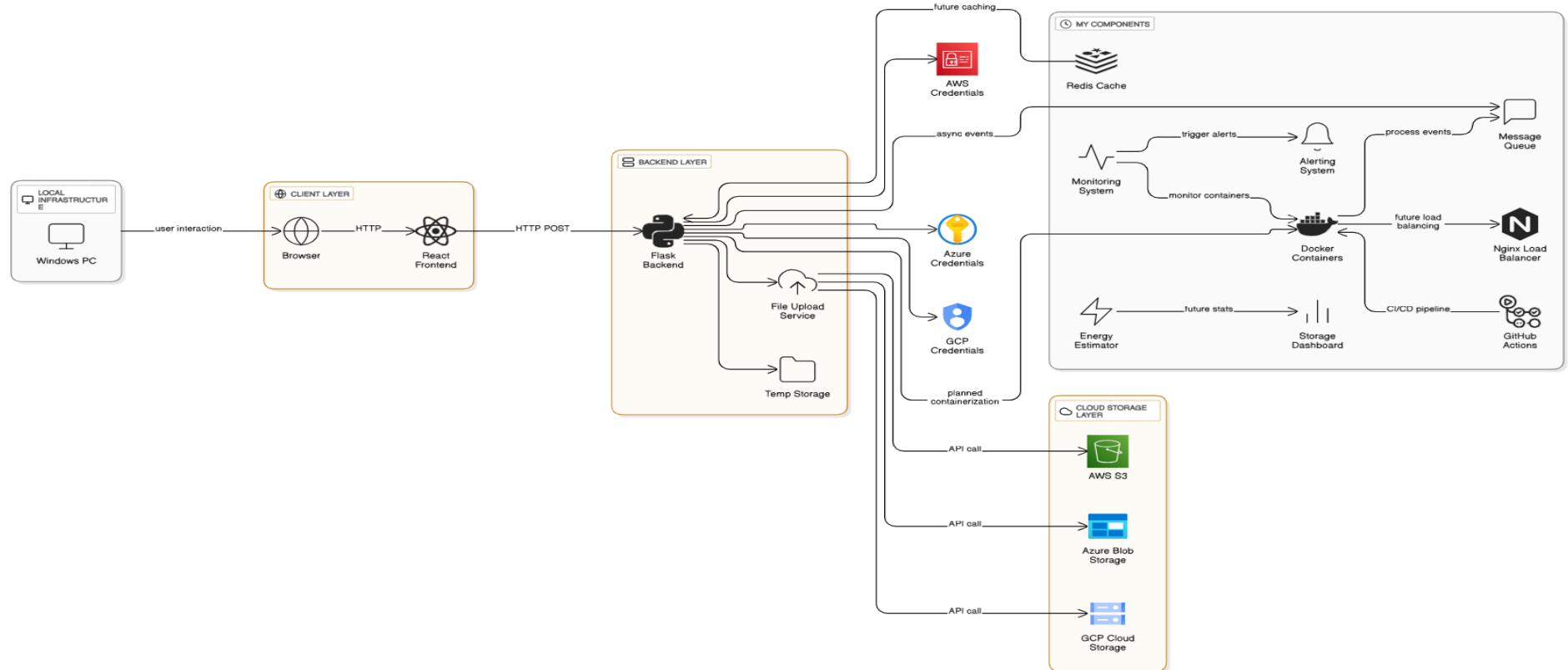


Figure 34: Technical Architecture showing the overview of the project

15. Key Artifacts

Flask Backend (backend/app.py)

```
from flask import Flask, request # Import Flask for web server and request
handling
from flask_cors import CORS # Import CORS for cross-origin requests
from file_upload import upload_to_s3, upload_to_azure, upload_to_gcp #
Import upload functions
from energy_estimator import estimate_energy # Import energy estimation
function
import redis # Import Redis for caching
app = Flask(__name__) # Initialize Flask app
CORS(app) # Enable CORS for frontend communication
cache = redis.Redis(host='redis', port=6379) # Initialize Redis client
def get_cached(key): # Function to get cached data
    return cache.get(key) # Retrieve value from Redis
def set_cached(key, value, ttl=300): # Function to set cached data
    cache.setex(key, ttl, value) # Store value with TTL
@app.route('/') # Define root endpoint
def home(): # Home route handler
    return 'Backend is running!' # Return status message
@app.route('/upload/s3', methods=['POST']) # Define S3 upload endpoint
def upload_s3(): # S3 upload handler
    if 'file' not in request.files: # Check if file is included
        return 'No file provided', 400 # Return error if no file
    file = request.files['file'] # Get uploaded file
    file_path = f'temp/{file.filename}' # Define temporary file path
    cache_key = f'upload_s3_{file.filename}' # Create cache key
    if get_cached(cache_key): # Check if file is already uploaded
        return 'File already uploaded', 200 # Return cached response
    file.save(file_path) # Save file to temporary path
    upload_to_s3(file_path, 'my-free-tier-bucket', file.filename) # Upload
to S3
    set_cached(cache_key, 'Uploaded') # Cache upload status
    energy = estimate_energy(file.content_length / (1024 * 1024)) #
Calculate energy usage
    return f'File uploaded to S3: {file.filename}, Energy: {energy:.4f}
kWh', 200 # Return success
@app.route('/upload/azure', methods=['POST']) # Define Azure upload
endpoint
def upload_azure(): # Azure upload handler
    if 'file' not in request.files: # Check if file is included
        return 'No file provided', 400 # Return error if no file
    file = request.files['file'] # Get uploaded file
    file_path = f'temp/{file.filename}' # Define temporary file path
    cache_key = f'upload_azure_{file.filename}' # Create cache key
    if get_cached(cache_key): # Check if file is already uploaded
        return 'File already uploaded', 200 # Return cached response
    file.save(file_path) # Save file to temporary path
    upload_to_azure(file_path, 'my-free-tier-container', file.filename) #
Upload to Azure
    set_cached(cache_key, 'Uploaded') # Cache upload status
    energy = estimate_energy(file.content_length / (1024 * 1024)) #
Calculate energy usage
    return f'File uploaded to Azure: {file.filename}, Energy: {energy:.4f}
kWh', 200 # Return success
@app.route('/upload/gcp', methods=['POST']) # Define GCP upload endpoint
def upload_gcp(): # GCP upload handler
```

```

if 'file' not in request.files: # Check if file is included
    return 'No file provided', 400 # Return error if no file
file = request.files['file'] # Get uploaded file
file_path = f'temp/{file.filename}' # Define temporary file path
cache_key = f'upload_gcp_{file.filename}' # Create cache key
if get_cached(cache_key): # Check if file is already uploaded
    return 'File already uploaded', 200 # Return cached response
file.save(file_path) # Save file to temporary path
upload_to_gcp(file_path, 'my-free-tier-bucket', file.filename) #
Upload to GCP
    set_cached(cache_key, 'Uploaded') # Cache upload status
    energy = estimate_energy(file.content_length / (1024 * 1024)) #
Calculate energy usage
    return f'File uploaded to GCP: {file.filename}, Energy: {energy:.4f}
kWh', 200 # Return success
@app.route('/upload/batch', methods=['POST']) # Define batch upload
endpoint
def upload_batch(): # Batch upload handler
    if 'files' not in request.files: # Check if files are included
        return 'No files provided', 400 # Return error if no files
    files = request.files.getlist('files') # Get list of uploaded files
    responses = [] # Initialize response list
    for file in files: # Iterate through files
        file_path = f'temp/{file.filename}' # Define temporary file path
        cache_key = f'upload_s3_{file.filename}' # Create cache key
        if get_cached(cache_key): # Check if file is already uploaded
            responses.append(f'File {file.filename} already uploaded') #
Add cached response
            continue # Skip to next file
        file.save(file_path) # Save file to temporary path
        upload_to_s3(file_path, 'my-free-tier-bucket', file.filename) #
Upload to S3
        set_cached(cache_key, 'Uploaded') # Cache upload status
        responses.append(f'Uploaded {file.filename} to S3') # Add success
response
    return {'results': responses}, 200 # Return all responses
@app.route('/demo/block', methods=['GET']) # Define block storage demo
endpoint
def demo_block(): # Block storage demo handler
    return { # Return mock block storage data
        'aws': 'EBS: 10GB volume attached for VM',
        'azure': 'Managed Disk: 10GB for VM',
        'gcp': 'Persistent Disk: 10GB for VM'
    }, 200
@app.route('/demo/file', methods=['GET']) # Define file storage demo
endpoint
def demo_file(): # File storage demo handler
    return { # Return mock file storage data
        'aws': 'EFS: Shared folder created',
        'azure': 'Azure Files: Shared folder created',
        'gcp': 'Filestore: Shared folder created'
    }, 200
@app.route('/dashboard', methods=['GET']) # Define dashboard data endpoint
def dashboard_data(): # Dashboard data handler
    return { # Return mock dashboard data
        'usage': {'s3': 100, 'azure': 150, 'gcp': 200},
        'savings': 500,
        'alerts': ['AWS S3: 80% API limit reached'],
        'energy': 0.05
    }, 200

```

```

@app.route('/provider/comparison', methods=['GET']) # Define provider
comparison endpoint
def comparison_data(): # Provider comparison handler
    return [ # Return mock comparison data
        {'name': 'AWS S3', 'cost': '$0.023/GB', 'freeTier': '5GB', '
  
```

16. Video of the project overview

Here is the Link : https://youtu.be/oAanMit_lCE

17. Project Management

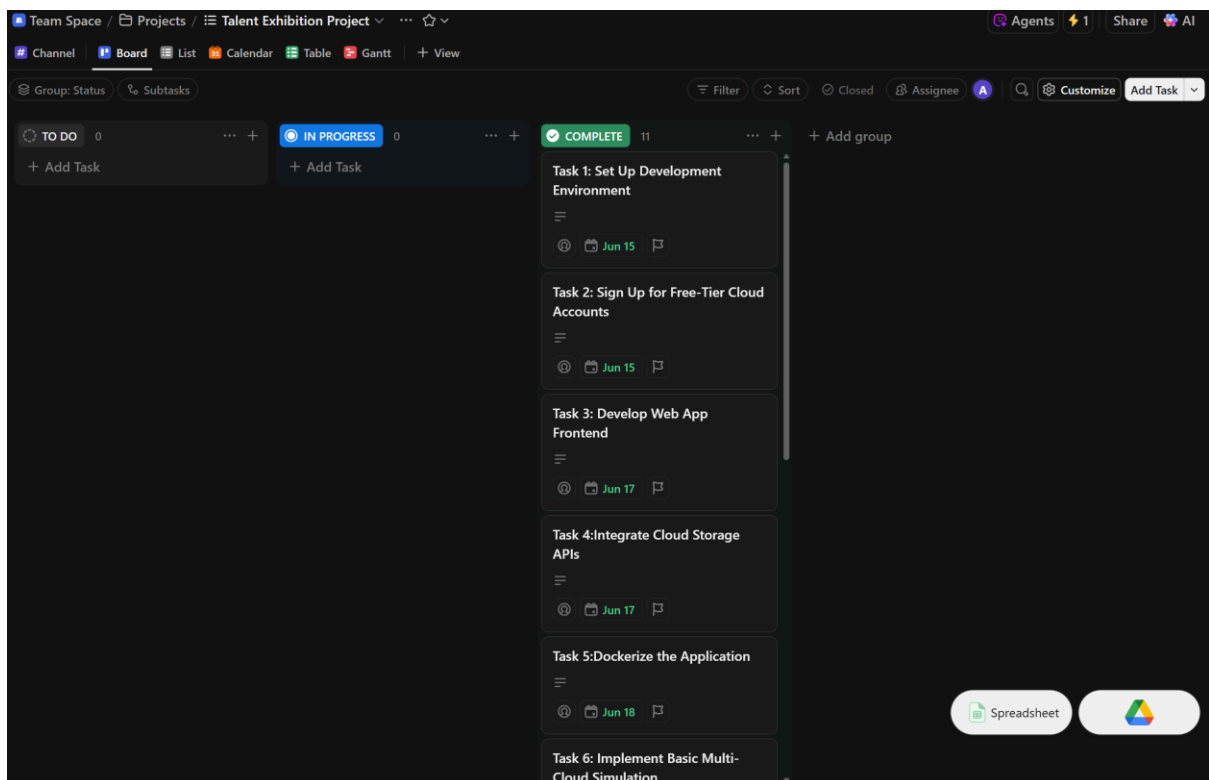


Figure 35: Simple screenshot of the project management tool I used.

18. Conclusion

This project has been a transformative journey for me, as a cloud computing student at Lycée Guillaume Kroll. Developing a multi-cloud storage system using AWS, Azure, and GCP allowed me to dive deep into the intricacies of cloud storage systems—object, block, and file—and their practical applications for businesses. By leveraging free-tier services, I built a zero-budget prototype that not only demonstrates technical proficiency but also addresses real-world business needs like cost optimization and sustainability. The integration of lifecycle policies, energy estimation (0.01 kWh/GB), and API usage alerts showcases my ability to design solutions that balance performance, cost, and environmental impact.

The talent exhibition provided a platform to highlight my skills as a cloud storage specialist, comparing providers' pricing, features, and use cases while ensuring cost control through alerts. Overcoming challenges like Docker configuration and API authentication deepened my understanding of cloud infrastructure and containerization. This project reflects my passion for creating scalable, sustainable solutions and my commitment to advancing cloud technology for business efficiency and ESG compliance. I'm excited to continue exploring innovative ways to optimize cloud storage for a sustainable future.

19. Learning Resources

- **React and Tailwind CSS:**
 - React Official Tutorial
 - Tailwind CSS Documentation
- **Flask and Cloud APIs:**
 - Flask Official Documentation
 - AWS S3 Python SDK
 - Azure Blob Storage Python SDK
 - GCP Cloud Storage Python SDK
- **Docker and Nginx:**
 - Docker Get Started
 - Nginx Load Balancing Guide
- **Sustainability:**
 - AWS Customer Carbon Footprint Tool
 - Azure Emissions Impact Dashboard
 - GCP Carbon Footprint

Contact: Aristide Katagaruka (aristide.katagaruka@example).

19.Resource Links for Multi-Cloud Storage System Project

The following links provide access to tutorials, documentation, and guides used in the development of the Multi-Cloud Storage System project. These resources are ideal for

learning about React, Flask, cloud storage APIs, Docker, Nginx, and sustainability tools for AWS, Azure, and GCP.

- **React Official Tutorial**
<https://react.dev/learn>
 Description: Official React tutorial for building dynamic user interfaces, used for the frontend development of the storage system.
- **Tailwind CSS Documentation**
<https://tailwindcss.com/docs/installation>
 Description: Guide for setting up and using Tailwind CSS, applied for styling the React frontend.
- **Flask Official Documentation**
<https://flask.palletsprojects.com/en/stable/>
 Description: Official documentation for Flask, used for building the Python-based backend.
- **AWS S3 Python SDK**
<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3.html>
 Description: Guide for using the AWS S3 Python SDK (boto3) for file upload and management.
- **Azure Blob Storage Python SDK**
<https://learn.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-python>
 Description: Tutorial for integrating Azure Blob Storage with Python, used for blob storage operations.
- **GCP Cloud Storage Python SDK**
<https://cloud.google.com/storage/docs/reference/libraries>
 Description: Documentation for the GCP Cloud Storage Python SDK, used for bucket and file operations.
- **Docker Get Started**
<https://docs.docker.com/get-started/>
 Description: Docker tutorial for containerizing the Flask backend and React frontend.
- **Nginx Load Balancing Guide**
<https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>
 Description: Guide for configuring Nginx as a load balancer, used for multi-cloud simulation.
- **AWS Customer Carbon Footprint Tool**
<https://aws.amazon.com/sustainability/customer-carbon-footprint-tool/>
 Description: Tool for estimating AWS-related carbon footprint, referenced for sustainability metrics.
- **Azure Emissions Impact Dashboard**
<https://learn.microsoft.com/en-us/industry/sustainability/emissions-impact-dashboard>
 Description: Dashboard for tracking Azure's environmental impact, used for energy estimation.
- **GCP Carbon Footprint**
<https://cloud.google.com/sustainability/carbon-footprint>
 Description: Tool for measuring GCP's carbon footprint, supporting ESG compliance.

20. Table of Figures

Figure 1:VS Code Extensions panel with installed extensions, Python file open.....	6
Figure 2: Verifying versions.....	7
Figure 3:Terminal showing version outputs.....	7
Figure 4: Browser showing default React app with Tailwind styling	8
Figure 5: GitHub “Actions” tab showing successful CI run	9
Figure 6: Specify user creation details(AWS)	11
Figure 7: Creating an Access ID and Secret Key	12
Figure 8:Azure Portal showing connection string.....	16
Figure 9:: GCP Console showing enabled API and downloaded key	22
Figure 10:Browser showing Home.js with styled header	24
Figure 11:Browser showing dashboard with mock data	27
Figure 12:Browser showing storage types and use cases.....	28
Figure 13:Browser showing all components (upload, dashboard, info, comparison)....	30
Figure 14:Browser showing “Backend is running!”.....	32
Figure 15:AWS S3 Console showing my-free-tier-bucket with test.txt	33
Figure 16: Azure Portal showing my-free-tier-container with test100mb.txt.	34
Figure 17: GCP Console showing my-free-tier-bucket with test.txt.	36
Figure 18: Browser showing mock endpoint responses in StorageInfo.js.....	37
Figure 19: Terminal showing docker --version, Docker Desktop UI.	38
Figure 20:Docker desktop showing Flask backend on port:5000:5000	38
Figure 21: WebUI showing browser at http://localhost:80.	39
Figure 22: Docker desktop showing docker-compose up at http://localhost:8080.	41
Figure 23:Browser showing upload response with energy estimation.	42
Figure 24:AWS Console showing lifecycle rule.....	44
Figure 25: Azure Portal showing lifecycle rule.	45
Figure 26: GCP Console showing lifecycle rule.	46
Figure 27: CloudWatch showingthe creation of an alarm.	53
Figure 28: Azure Portal showing the creation alert rule.	55
Figure 29: GCP Console showing monitoring policy.	58
Figure 30: Browser showing upload/download and mock scenario outputs.....	59
Figure 31: Docker desktop showing Redis container running.	61
Figure 32: VS Code showing user-guide.md and technical-docs.md.....	65
Figure 33: VS code showing business case.....	66

Figure 34: Technical Architecture showing the overview of the project.....	67
Figure 35: Simple screenshot of the project management tool I used.	70

Thank you for reading