Backend Engineering Launchpad - Case Study

Amazon S3 : Simple Cloud Storage

Author: Airtribe

Background & Problem

In today's digital age, cloud storage solutions like Amazon S3 are essential for storing, retrieving, and managing vast amounts of data efficiently and securely. These services offer many important features such as file versioning, searchability, and robust access control. However, understanding how these systems work can be complex, requiring deep knowledge about the principles of distributed systems, data security, file management, and user authentication.

For this capstone project, your goal is to build a simplified clone of Amazon S3, which we'll refer to as "Simple Cloud Storage" (SCS). The SCS will be a web-based storage solution that allows users to store and retrieve files from a personal cloud-based storage space. This project will help you grasp the underlying principles of cloud storage systems while honing your skills in backend development and system design.

Requirements

The SCS should fulfill the following requirements:

1. User Authentication: The system should implement a secure user authentication system. Users should be able to register, login, and manage their accounts independently.

2. File Upload and Download: Users should be able to upload files to their storage space and download them as needed. The integrity and security of the uploaded files should be ensured.

3. File Organization: The system should allow users to organize their files into directories or folders, akin to the bucket and object hierarchy in Amazon S3.

4. File Listing and Search: The system should enable users to view a list of their uploaded files. Additionally, users should be able to search for specific files based on filenames and metadata.

5. Permissions and Access Control: The system should include an access control mechanism that dictates who can access specific files or folders. The system should support different types of access, such as private, public, and shared.

6. File Versioning: Users should have the ability to manage different versions of a file and roll back to previous versions when necessary.

7. Metadata Management: Users should be able to add metadata to their files to enhance searchability and organization. This metadata could include tags, descriptions, or custom attributes.

Add-ons & Extensions

1. File Deduplication: Implement file deduplication to maximize storage efficiency. If a user attempts to upload a duplicate file (an exact copy of a file that already exists in their storage), the system should recognize this and avoid storing multiple copies of the same file.

2. Usage Analytics: Implement APIs using to check analytics related to their storage usage. This could include the amount of storage space used, the types of files stored, the frequency of file access, etc.

Assessment Criteria

- Functionality: Does the system work as intended? Does it meet all the requirements stated above?
- Code Quality: Is your code clean, organized, well-commented, and following best practices?
- Design and Structure: Is the system well-designed? Does it demonstrate a good understanding of system design principles and patterns?
- Documentation: Is your report comprehensive and clear? Does it effectively explain the choices made and how to use the project?
- Presentation: Do you effectively demonstrate and explain your system and the decisions made during its development?

Deliverables

1. The final, functional product.
2. README file outlining how to use the system, API documentation, the design decisions and other necessary information.
3. Public link of the Github repository
4. Explainer video demonstrating your project work