Report Cloud Basic

Edoardo Cortolezzis

May 2024

1 Introduction

The focus of this project is centered around building and deploying a scalable, secure and efficient cloud system; leveraging Docker for containerization, Nextcloud for cloud storage and user management, and Locust for performance testing. The system should allow file upload, download, and management functionalities.

2 Managing User Authentication and Authorization and File Operations

Nextcloud serves as the backbone of our cloud system, providing robust capabilities for user authentication, authorization, and file operations. As an open-source, self-hosted cloud platform, Nextcloud offers a comprehensive suite of features designed to empower users with secure and efficient file storage solutions. Leveraging Nextcloud's functionalities, I have developed and implemented user management and file operation features to create a seamless and intuitive user experience within the cloud environment.

2.1 User Authentication and Authorization

These are fundamental aspects of the cloud system, ensuring security and a smooth user experience. Users can sign up, log in, and log out either through the Nextcloud interface or via a shell script (e.g., *create_users.sh* and *delete_users.sh*). Additionally, users can manage their passwords and profile settings, enhancing control over their accounts. Depending on the type of user, they have different kind of permissions.

2.2 File Operations

File operations such as uploading, downloading, and deleting files from private storage are seamlessly facilitated in the system. Users can perform these operations through the Nextcloud interface or via a shell script (e.g., *locust_test.py* and *clear_storage.sh*). This flexibility allows users to interact with their files

conveniently, whether through the graphical user interface or automated scripts for efficiency and automations.

3 Addressing Scalability

3.1 Design for Growth

The system is designed to accommodate a growing number of users and files while maintaining optimal performance and reliability. During testing, scalability was evaluated using Locust, a load testing tool, where the number of users increased incrementally and a set of tasks are tested. This allowed for the observation of system behavior and performance under varying loads.

3.2 Handling Increased Load and Traffic

In theory, to handle increased load and traffic beyond the tested capacity, several strategies can be implemented:

- **Horizontal Scaling**: Deploying multiple instances of the application across multiple servers allows for distributing the load and handling increased traffic effectively. Load balancers can be employed to evenly distribute incoming requests among these instances.
- **Vertical Scaling**: Increasing the resources of individual servers can help handle increased load. However, there may be limits and it may not be as cost-effective as horizontal scaling in the long run.
- **Caching Mechanisms**: Implementing caching mechanisms can help reduce the load on the backend servers by serving frequently accessed data directly from cache memory. This improves response times and reduces the overall load on the system.
- **Optimized Database Queries**: Optimizing database queries and indexes can improve database performance, allowing the system to handle more concurrent requests efficiently.

4 Addressing Security

4.1 Secure File Storage and Transmission

Within the system, users are categorized into two main groups: "admin" and "users" (regular users). Regular users are allocated 3GB of private storage space, isolated from other users, ensuring privacy and data protection. Admin users have the capability to adjust the storage quota for each user, managing the system's capacity and ensuring fair use across all accounts. Additionally, server-side encryption can be implemented through the Nextcloud interface for an added layer of security.

4.2 Secure User Authentication

The system employs a secure login system, requiring a username and password for access. Password constraints, such as the use of numbers, capital letters, and special characters, can be enforced to enhance password strength and security. Furthermore, the system supports two-factor authentication (2FA), providing an additional layer of security by requiring users to provide a second form of verification. Additionally, the implementation of Nextcloud's Registration app enables users to create their accounts securely with email verification.

4.3 Measures to Prevent Unauthorized Access

To prevent unauthorized access, the system employs several measures:

- **User Roles and Permissions**: Users are categorized into regular and admin roles, each with distinct permissions and access levels. Admin users have the authority to manage user permissions, ensuring appropriate access control within the system.
- **File Access Control**: The File access control app has been implemented to allow admins to further manage user permissions over file access.
- **Brute Force Protection**: Nextcloud includes a brute force protection feature that detects and slows down login attempts after a certain number of failures. This helps mitigate the risk of unauthorized access by preventing automated and unauthorized login attempts.

5 Discussing Cost-Efficiency

5.1 Cost Implications of the Design

Given that the project was developed on my local machine and utilized free services, there were no actual costs associated with it. However, if the project were to be deployed online, various costs would arise, including hosting, domain registration, storage, and potentially other costs related to external services. Specific costs vary depending on the chosen hosting provider and service configurations, so it's essential to consider these factors when planning for deployment.

5.2 System Optimization for Cost Efficiency

To optimize the system for cost efficiency, several strategies can be employed:

- **Pay-On-Demand Services**: Utilizing pay-on-demand services such as AWS S3 or Google Cloud Storage allows to pay only for the storage and bandwidth you use. This flexible pricing model is well suited to scale the storage needs as the project grows, minimizing unnecessary costs.

- **Comparison of Pricing**: It's essential to compare the pricing of different providers to find the most cost-effective solution for the project's needs. Factors to consider include storage capacity, bandwidth, and data transfer costs.

6 Deployment of the Cloud System

6.1 Deployment Plan

To deploy the system in a containerized environment on your laptop using Docker and Docker Compose, follow these steps:

1. **Installation of Dependencies**: Ensure that Docker and Docker Compose are installed on your laptop.

2. **Repository Setup**: Clone the repository containing the project files to your local machine.

3. **Configuration**: Review the configuration files, particularly the *docker-compose.yaml* file, and modify them according to your specific needs. This includes adjusting ports, volumes, and services as necessary. Note that the project utilizes Nextcloud and MariaDB as services, so modifications may be required if different services are desired.

4. **Deployment**: Navigate to the root directory of the repository and run the command "docker compose -f docker-compose.yaml up -d". This command will start the services defined in the *docker-compose.yaml* file and create the containers for the system. Once deployed, you can access the Nextcloud instance at http://localhost:8081, log in as an admin using the credentials defined in the *docker-compose.yaml* file, and begin using the system. To shut down the system, use the command "docker compose down".

5. **Performance Testing**: For performance testing, additional scripts are available. Use the *docker-compose-locust.yaml* file to deploy a Locust instance that simulates users accessing the Nextcloud instance. Access the Locust dashboard at http://localhost:8089. Utilize the *create_users.sh* script to create users in the Nextcloud instance and run the *locust_tests.py* script to perform the actual tests.

6.2 Monitoring and Management

Monitoring and managing the deployed system can be done through the Nextcloud interface, which provides a dashboard with information about the system's status, performance, and usage. Additionally, Docker provides built-in monitoring tools, such as Docker Stats and Docker Events, which allow you to monitor container resource usage and events in real-time.

6.3 Choice of Cloud Provider

If we were to deploy online this project, the preferred choice would be AWS (Amazon Web Services). AWS offers a comprehensive suite of services, including EC2 instances, S3 storage, and RDS databases, essential for online deploying a cloud system. With its robust infrastructure, high availability, and scalability, AWS is well-suited for hosting production environments. Furthermore, AWS has a strong reputation in the industry, excellent customer support, and comprehensive documentation, facilitating the deployment and management of the system in the cloud.

7 Testing and Results

The system underwent comprehensive testing to evaluate its performance in terms of load and I/O operations. Various file operations, including file uploads of different sizes (ranging from 1KB to 1GB) and image uploads, file listing, and file reading, were conducted to assess the system's functionality. Load testing using Locust was employed to simulate concurrent user activity, with the number of users incrementally increased up to 30. The results, shown in Fig.1, demonstrated that the system can handle up to 30 users with low failure rates, indicating its scalability and robustness under moderate to heavy loads. Performance metrics such as Total Requests per Second and Response Time were monitored during load testing, providing insights into the system's responsiveness and efficiency under different load conditions. Overall, the testing and results confirm the system's effectiveness in meeting its performance objectives and providing a reliable and efficient file storage solution.

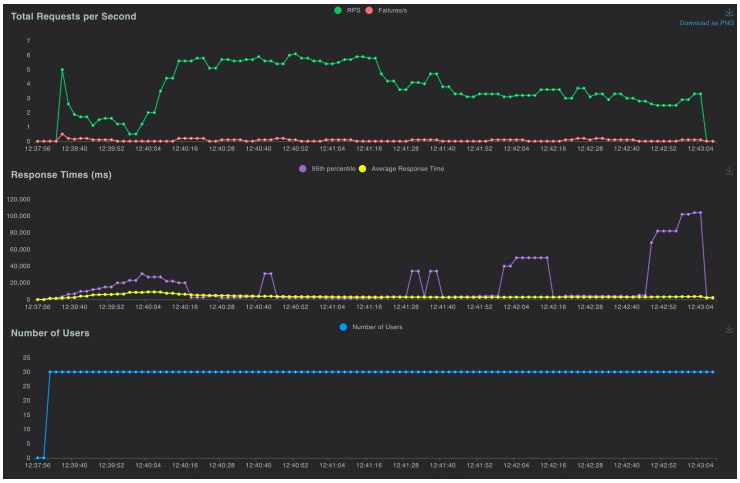

Figure 1: Tests Performances

8 Conclusions

In conclusion, the development and deployment of the cloud-based file storage system has been a success. The functioning system, run and tested on the local machine, demonstrated the feasibility and effectiveness of the chosen technologies and methodologies. Throughout the project, valuable insights were gained into various aspects of cloud computing, containerization, and system deployment, enhancing my understandings and making me gain skills in these areas. The project has provided valuable hands-on experience in designing, implementing, and testing a scalable, secure, and efficient cloud system, contributing to my professional growth and development.