Cloud-Based Filesystem with NextCloud

1 Introduction

This project centers around the creation and implementation of a cloud-based file storage system, with a specific focus on utilizing Nextcloud as the selected platform to create the product. The primary objectives include user-friendly functionalities such as file upload, download, and deletion, while ensuring each user has a dedicated private storage space. Critical considerations involve user authentication, file operations, scalability, security measures, and cost-efficiency. The chosen solution, Nextcloud, is examined in detail for its applicability in meeting these requirements.

The deployment plan involves containerization with Docker and docker-compose on a local machine (Mac M1), presenting a practical approach to managing the system. The design documentation encompasses architectural details, component interactions, and security measures taken. Additionally, the cost implications of the design and theoretical considerations for scalability and increased load are discussed. Finally, Docker files and relevant code, including a README file for system deployment and usage instructions, are included in the Github repository as part of the documentation.

2 Creating and Deploying the File Storage System

To create and deploy the file system using Docker Compose it is enough to write a .yml file in which the required services are enlisted (One example of such file, in which username and passwords have been deleted, can be found in the Github repository). The configuration defines two services, Nextcloud and a MariaDB database (plus Locust, which is discussed later in the report), along with associated volumes and networks. The Nextcloud service runs on port 8080, connecting to a MariaDB database service named 'db.' Persistent data storage is achieved through volumes: 'nextcloud_data' for Nextcloud and 'db_data' for MariaDB, the chosen database for the project. The environmental variables set in each service define the necessary database configurations. The file also establishes a network named 'nextcloud_network' to enable communication between the services.

To deploy this file, it must be saved as `docker-compose.yml` and it is enough to simply execute the following command in the terminal within the directory containing the file: `docker-compose up -d`.

This command will pull the required images, create containers based on the defined services, and deploy the Nextcloud file system with its associated database. To access the actual file storage system one can just type 'localhost:8080' and insert their username and password.

3 Key Features

In this section the feature required by the exercise are discussed as they are implemented by the Nextcloud platform.

- **User Authentication:** Nextcloud provides a secure user authentication system, allowing users to sign up, log in, and log out. Different measures can be taken to improve the security of the file system, increase password length and require special characters, enable two-factor authorization, as well as encryption on stored data. This last option comes at the cost of some efficiency, and should therefore only be used when it is really needed.
- **User Authorization:** Users can be easily added and deleted, together with their data. Different user roles, such as regular users and admins, are supported for effective user management. Admins have the ability to manage and delete users, and are the only ones who can modify the overall settings of the system.
- **File Upload:** Users can easily upload files to their private storage space.
- **File Download:** Users have the possibility to download files from their private storage.
- **File Deletion:** Users can also delete files from their private storage space.
- **Private Storage Space:** Regular users are allocated their private storage space to ensure data privacy. Different storage limits can be imposed to avoid any particular user from exploiting the service and preserve storage.

4 Performances

In this section the current performances of the file storage system are discussed, together with some ideas on how to design and what tools to use to deal with an increasing demand of the system.

4.1 Performance Evaluation with Curl

Curl, which stands for Client for URLs, is a powerful command line library that allows users to perform a variety of tasks, such as downloading or uploading files, sending and receiving data from web services, and testing API endpoints. It supports a broad range of options and protocols, making it a versatile and popular tool for interacting with the web from command line.

To assess the performance of the file storage system, a number of files of different sizes has been created using the `dd` command. Then these files have been uploaded to and downloaded from the system, taking the time needed to conclude the operations. The command to download the data is `curl -k -u "USERNAME:PASSWORD" -O "localhost:8080/remote.php/webdav/REMOTE_FILEPATH"`, while the one to upload the files is `curl -k -u "USERNAME:PASSWORD" -T "LOCAL_FILEPATH" "localhost:8080/remote.php/webdav/REMOTE_FILEPATH"` and in both of them the real username and password have been omitted.

The following plots show the time needed to perform the download and the upload operation versus the size of the of the file to be transferred.

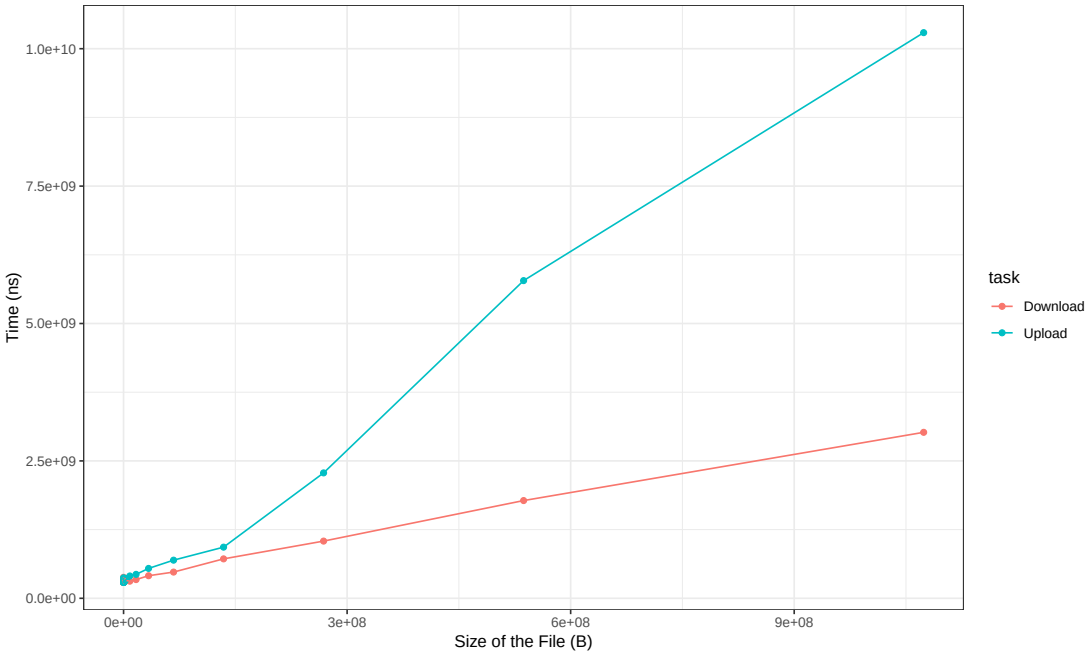

Figure 1: Time vs Size of the File

The plot shows that both operation showcase similar behaviours when the size of the file is kept small, while the download appears to be faster as the size of the file increases.

4.2 Performance Evaluation with Locust

Locust is an open-source, Python-based performance testing tool designed for simulating large numbers of users to assess the scalability and performance of web applications. It works by allowing users to define scenarios that simulate user interactions with the system, and then generating a load based on these scenarios to stress-test the target application or service.

For this exercise, 30 users have been created using the command `docker exec -e OC_PASS=test_password1234! --user www-data ex_3-nextcloud-1 /var/www/html/occ user:add --password-from-env test_user`. As a security measure Nextcloud only accepts requests coming from localhost, and it is therefore needed to add the locust container to the `trusted_domains`. In order to use locust one needs to write some tasks that tell locust which operation to perform. They can be found in the `/locust-tasks/tasks.py` file and have the following purposes:

- **propfind:** Sends a PROPFIND request to list files for the current user.
- **read_file:** Performs a GET request to read the content of the `Readme.md` file for the current user.
- **upload_file_1gb**, **upload_file_1kb**, **upload_file_1mb:** Simulate file upload tasks of sizes 1GB, 1KB, and 1MB, respectively. They use the PUT method to upload files from the local `/test-data/` directory.

Each of the task also possesses a weight which determines the relative frequency with which it is performed by Locust. The following charts show the total number of requests and failed requests at each timestamp, together with the response time.

Figure 2: Locust Test, 10 users

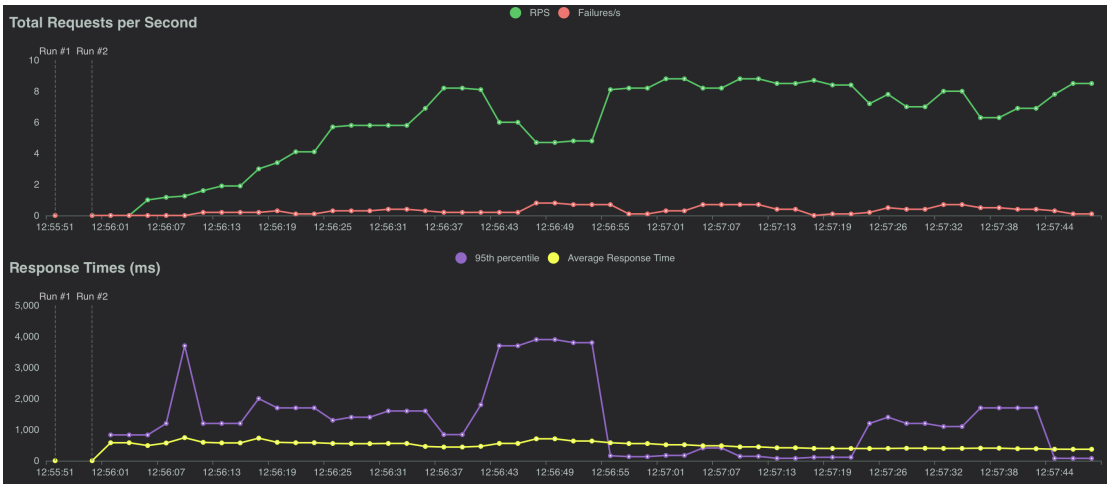

Figure 3: Locust Test, 30 users

While the response time does not show significant differences between the tests performed with 10 and 30 users, one thing to be noticed is that some requests fail in the last scenario. More tests would be needed to pinpoint the reason behind this phenomenon.

5 Scalability

In this section two possible solution to an increase demand of the system are explored and their strengths and weaknesses are explored.

5.1 On-Cluster Deployment

Cluster deployment involves creating a network of interconnected servers to distribute and manage the file system. This approach strengthens scalability and fault tolerance, but it demands significant setup complexity and ongoing maintenance efforts. While it provides high availability and promotes independence, it may be more suitable for enterprises with substantial infrastructure investments who already possess a cluster.

5.1.1 Advantages of On-Cluster Deployment

- **Scalability:** Cluster deployment offers horizontal scalability, allowing organizations to add more nodes to the cluster as storage needs grow, providing virtually unlimited scalability.
- **Data Redundancy:** Distributed filesystems in clusters provide built-in redundancy mechanisms, ensuring data resilience and high availability.
- **Security:** Organization can choose their own privacy policies without relying on third party providers.
- **Control:** Enterprises have full control over the cluster infrastructure.

5.1.2 Disadvantages of On-Cluster Deployment

- **Complexity:** Setting up and managing a cluster deployment requires expertise in distributed systems, networking, and storage technologies, leading to higher complexity and maintenance overhead.
- **Costs:** Initial setup costs for hardware, networking, and infrastructure can be substantial. Additionally, ongoing maintenance and management expenses, including monitoring, troubleshooting, upgrades, and data backup systems contribute to total cost of ownership.

5.2 Cloud Services

Leveraging cloud services like AWS S3 allows organizations to scale storage seamlessly, paying only for the resources they consume (pay as you go services). One key feature of such services is their autoscalability, which refers to the ability of a system or infrastructure to automatically adjust its capacity and resources in response to changing workloads or demand.

5.2.1 Advantages of relying on Cloud Services

- **Scalability:** Cloud services like AWS provide on-demand scalability, allowing organizations to instantly increase or decrease storage capacity as needed, without upfront hardware investments.
- **Global Accessibility:** Cloud services offer global accessibility, allowing users to access data from anywhere in the world, improving collaboration and productivity.
- **Cost Efficiency:** Cloud services operate on a pay-as-you-go model, enabling organizations to pay only for the resources they use, minimizing upfront costs and optimizing resource utilization.

5.2.2 Disadvantages of relying on Cloud Services

- **Data Privacy and Compliance:** Storing sensitive data in the cloud may raise concerns related to data privacy, security, and compliance with regulatory requirements, especially in highly regulated industries.
- **Dependency:** Organizations relying on cloud services are dependent on the availability and performance of the cloud provider's infrastructure. Service outages or disruptions can impact business operations.

5.3 Final Considerations

Comparing the two solutions, a number of factors must be taken into account when deciding which solution should be chosen, among these there are organizational requirements, budget constraints, scalability needs, and data security issues. In regard to costs, for instance, on-cluster deployment has higher upfront infrastructure costs (capex), while the operative expenses (opex) should probably be assessed taking the account the specifics of any particular situation, and can therefore be seen that no silver bullet is at our disposal.