Cloud Basic final project

Giulio Fantuzzi [SM3800012]

21st March 2024 exam

Contents

1 Abstract

This project focuses on identification, deployment, and implementation of a cloud-based file storage system, with the primary aim of enabling users to upload, download, and delete files, while ensuring them to have their data through private storage spaces. Key considerations such as scalability, security, and cost-efficiency will be explored to evaluate accurately the deployed solution.

2 Why Nextcloud?

Among the hinted solutions, Nextcloud emerged as the most appealing platform for several reasons. Such decision was primarily influenced by the simplicity in implementing the system, especially when leveraging Docker containers. Nextcloud is supported by an extensive documentation and, being an open-source project, it benefits from a vast community of developers and users, which greatly streamlined the whole implementation process. Additionally, Nextcloud offered a comprehensive set of features beyond basic file storage, allowing me to set robust security measures

and to integrate other software and services, making the system highly versatile and suitable for a variety of use cases.

3 User Authentication and File Operations

Requirements about user authentication and authorization are guaranteed:

- **User Management:** Nextcloud offers a built-in user management system that supports sign-up [1] , login, and logout functionalities. This system is designed to be user-friendly, ensuring users to easily navigate the platform.
- **Role-Based Access Control:** Nextcloud supports role-based access control ($RBAC$), allowing for the creation of different user roles such as regular users and admins. This feature was crucial for managing access levels within the system, ensuring that admins have the necessary permissions to manage users, while regular users have their private storage space.
- **Admin Management:** admins have the ability to manage users through the Nextcloud admin interface. This includes creating, editing, and deleting user accounts, as well as assigning roles and permissions. Such feature is, in general, essential for maintaining the system's security and user management.
- **Private Storage Space:** each user is automatically assigned a private storage space upon account creation. Nextcloud, by default, assigns a 512 MB storage quota to each user, and administrators can adjust individual users' quotas through the web interface. To set a global quota for all the users, instead, it is necessary to modify a `config.php` file (see Section 7.2).

Nextcloud meets the specified criteria also regarding file operations. More precisely, it provides a straightforward and secure method for users to upload files to their private storage. Users can easily navigate to their private storage space and upload files directly from their devices, through an intuitive interface. Downloading files from private storage is just as simple. Users can access their files through the Nextcloud interface and download them to their devices. Ultimately, users are allowed to delete files directly from their private storage. This feature is integrated -once again- into the user interface, facilitating users to manage their files and to keep only the data that they actually need.

4 Address security

Nextcloud offers a comprehensive suite of security features that can be configured from the administrators interface. All the options can be found in `Administration settings/Security/` and can be easily enabled. While experimenting with these security settings for the purpose of this exercise, it became evident that despite they are not strictly necessary for basic functionality, they are highly recommended for real-world deployments. In a real-case scenario, where the security of sensitive data is of prime importance, these settings should be enabled and carefully configured to the highest level of protection.

4.1 Basic password policies

Administrators can set parameters such as the minimum password length, user password history, and the number of days until a user password expires. Additionally, they can specify the number of login attempts before an account is blocked, with the option to set this to zero for no limit. These settings might prevent unauthorized access by enforcing strong password practices among users.

[1] To facilitate self-registration through the web interface, Nextcloud provides a Registration app that can be easily installed and activated. This setup will enable users to sign-up using their email addresses, with a verification link sent to them for confirmation. Notice that such a functionality requires configuring a server email account and setting up an SMTP server.

4.2 Enhanced password security

Besides basic policies, Nextcloud offers an enhanced password security, allowing administrators to enable features such as "forbid common passwords," "enforce upper and lower case characters," "enforce numeric/special characters," and "check passwords against the list of breached passwords from haveibeenpwned.com". These options significantly increase the complexity and security of passwords and prevent unauthorized access, enhancing the overall security and mitigating potential risks to user accounts. Hence, all of them are strongly recommended in a real-world scenario.

4.3 Server-side-encryption

Server-side encryption (SSE) in Nextcloud allows administrators to encrypt files before they are stored on the server. This encryption process is actually transparent to the end-users, so that files are automatically encrypted upon upload and decrypted upon download. Despite some drawbacks in terms of performance, I considered SSE as a fundamental option to enable in order to strengthen the security of data stored in my Nextcloud instance.

4.4 Two-factor authentication (2FA)

Nextcloud offers 2FA as a security feature that adds an additional layer of protection to user accounts beyond traditional password-based authentication. If enabled, it enforces users to provide two forms of identification: a password and a second factor, reducing the risk of unauthorized access to accounts (even if a user's password is compromised). In a real-world scenario, enabling and enforcing 2FA would be a fundamental step in securing the deployed cloud environment. However, I deliberately excluded such security measure from my project (this was done primarily to simplify the testing process with `locust`).

4.5 Monitoring the system

Besides enforcing password policies and access controls, proactive monitoring is a crucial aspect for safeguarding the security of a Nextcloud instance. Nextcloud already provides some built-in features and apps for doing that, which are directly accessible from the `Administration settings` : system metrics are available at `System` section, while any suspicious activity or potential security incident may be monitored within the `Logging` section. Further tools to manage and monitor users activity can be downloaded (and enabled) in Nextcloud `Apps` section. More precisely:

- **Ransomware Protection app**, which prevents clients from uploading files with known ransomware file endings. By detecting and blocking potentially malicious files, the Ransomware Protection app helps mitigate the risk of data loss or extortion.
- **User Usage Report app**, which generates usage reports and facilitates administrators to track user behavior, monitor file access patterns and identify potential security risks.

5 Cost-efficiency

5.1 Cost Implications of my Design

For this project I referred to Nextcloud Server, which is the free version of Nextcloud, available to all the users. It can be deployed on a dedicated server infrastructure, making it particularly suitable for small/medium businesses. However, larger organizations with more advanced needs, requiring professional support and scalability, may find it necessary to opt for Nextcloud Enterprise: a pre-configured and optimized version that offers various service tiers (basic, standard, premium, and ultimate), associated to different levels of requirements. The free version of Nextcloud eliminates

the need for a subscription fee, but still hosting costs remain. More precisely, these costs can include both the hardware on which Nextcloud is hosted, both the storage solutions used to store the data. For the purposes of my project, I limited to deploy Nextcloud on my laptop (using Docker) without the integration of any external service or storage solution.

5.2 Optimizing the System for cost-efficiency

Assuming to manage a Nextcloud instance within a corporate environment, I have identified cost-efficiency in storage management as the most intriguing challenge, especially for organizations experiencing growth or fluctuating data usage (like startups). Forecasting the necessary amount of storage often proves to be infeasible, especially when the future trajectory of the business is uncertain. Therefore, I would recommend a balanced approach that incorporates both on-premises and cloud storage solutions, leveraging the advantages of each while mitigating their respective limitations. My personal advice would be to allocate a certain amount of budget towards on-premises storage, so that essential tasks can be performed independently of external providers. Then, a cloud storage solution should be integrated through its *pay-as-you-go* model, enabling the organization to adjust its storage capacity according to the fluctuating demand.

6 Address scalability

6.1 Handle a growing number of users and files

As the number of users and files increases, it becomes essential to implement a scalable file storage architecture. As we previously analyzed in Section 5.2, this may involve a mixed strategy combining on-premises and cloud storage solutions. Regarding the latter, technologies like AWS S3 offer highly scalable and reliable storage options, which we'll delve into further in Section 7.3. Furthermore, deploying a distributed environment that employs horizontal partitioning of data is a viable approach to enhance scalability. Horizontal partitioning divides data into smaller, more manageable chunks, thereby distributing the load across multiple nodes and improving performance. In this particular setup, NoSQL solutions like MongoDB may be intriguing in terms of scalability, flexibility, and performance for handling large volumes of unstructured data. However, the feasibility of this second solution is not trivial at all. Nextcloud does not directly support NoSQL databases, as it primarily works with SQL databases for its core functionalities. To overcome this, either you develop a custom storage backend that interfaces with the NoSQL database, or you refer to some already existing tools, like `n8n`. Notice that this service comes with a cost, so the organization should think carefully about costs and benefits. Ultimately,the transition to a NoSQL database is strictly dependent on some data requirements, including the types of data being stored and accessed within the system (it is crucial to assess whether our company meets the requirements).

6.2 Handle increased load and traffic

As the demand for the services grows, it is fundamental to ensure that our system can effectively manage increased load and traffic without compromising performance or reliability. Some considerations from Section 6.2 are still valid, especially the idea of horizontal data partitioning. In fact, by dividing data into smaller partitions across multiple nodes, we can distribute the load evenly and improve the system's overall performance. Transitioning from theoretical discussions to concrete implementation, I equipped my Nextcloud instance with `Redis`, a caching mechanism which allows to store frequently accessed data into memory, reducing the need to query the primary storage or database for every request. This not only improves response times, but also alleviates the load on the database, enhancing the overall scalability of the system.

7 Deployment

7.1 Docker and docker-compose

Aligning with the project requirements, the deployment of Nextcloud has been executed using `docker` and docker Compose. The deployment scripts and configuration files are available here. Below I provide a brief overview of the Docker images:

- **nextcloud**: this image deploys the Nextcloud application and it is configured with environment variables for database connection details, admin user credentials, and other settings. The image exposes port 8080 to allow access to the Nextcloud web interface;
- **mariadb**: this image is used to deploy a MariaDB database server, providing the backend storage for Nextcloud. It is configured with environment variables for setting up the root password, db name and user credentials;
- **redis**: this image sets up a Redis server, which is utilized for caching purposes (see Section 6.2 for details) within the Nextcloud application;
- **locustio/locust**: this image sets up `Locust`, an open-source load testing tool, which is used to simulate user traffic and analyze the performance of the Nextcloud deployment. In particular, it is configured to run tests defined in a `locustfile.py`, targeting the Nextcloud instance.

In summary, these Docker images work together in a containerized environment for deploying and testing the Nextcloud app with all its associated services. To run the docker containers, just move to the `docker-compose.yml` file and do:

```
docker-compose up -d
```

7.2 Main configurations

Once executed the `docker-compose.yml`, the first option that I configured regarded the private storage space. As detailed in Section 3, each user is automatically assigned a 512 MB private storage space upon account creation. However, since the testing phase would have involved stressing the system with files of the order of gigabytes, I adjusted the storage quota per user accordingly, with:

```
docker exec --user www-data nextcloud php occ config:app:set files\
    default_quota --value '2 GB'
```

The second stage of the configuration phase involved setting up security measures, which were easily configurable through the Nextcloud interface (located at the path `Administration Settings/Security/Password policy`). A slightly more complex feature to enable was the SSE (server-side encryption). While the interface provided a dedicated button at `Administration Settings/Security/Server-side encryption`, a preliminary step was required to enable encryption from the web. Specifically, after logging in as an admin, it was necessary to navigate to the Apps section, search for the Default Encryption module app and enable it. Alternatively, it is possible to enable SSE from terminal, through occ command:

```
docker exec --user www-data nextcloud /var/www/html/occ \
    app:enable encryption
docker exec --user www-data nextcloud /var/www/html/occ \
    encryption:enable
```

7.3 Cloud provider to deploy the system in production

For deploying a Nextcloud instance in a production environment, I would opt for AWS (Amazon Web Services), since it emerges as a compelling choice due to its scalability, reliability, and the wide range of services that it offers. In what follows I present some key reasons to justify AWS as the ideal cloud provider choice.

- **Scalability and Performance**: AWS offers a highly scalable infrastructure, allowing to accommodate the growth of my Nextcloud deployment as the user base and the storage needs increase. AWS provides services like Amazon S3, which is highly scalable and can serve thousands of HTTP requests per second, without any changes to the architecture. This aspect, which was somehow anticipated in Section 6.1, is crucial for handling varying loads and ensuring that the Nextcloud instance remains responsive and available to users.
- **Reliability**: AWS is known for its high reliability and availability. With data centers located in various regions worldwide, AWS can provide redundancy capabilities, ensuring that my Nextcloud instance remains accessible and functional despite hardware failures or network disruptions.
- **Cost-Effectiveness**: AWS pricing is based on usage, meaning that we only pay for the resources we consume. This *pay-as-you-go* model, which can lead to significant cost savings, has already been discussed in Section 5.2.
- **Integration**: AWS offers integration with a vast ecosystem of services and tools, allowing us to easily incorporate additional functionalities into Nextcloud.

8 Locust testing

As part of evaluating the performance of Nextcloud infrastructure, I conducted load testing using `Locust`. `Locust` is a load testing tool that allows to simulate concurrent users accessing the system. Developers can define tests in regular Python code, while the locust interface provides detailed reports about the performance of the Nextcloud instance under the simulated load. Prior to initiating any test, it was necessary to create a set of user accounts to mimic interactions with Nextcloud. To perform this task, I designed a specific bash script:

```
#!/bin/bash
new_users=$1
base_username="TestUser_"
base_password="TestPassword_"
for i in $(seq 1 $new_users)
do
    username="${base_username}${i}"
    password="${base_password}${i}"
    docker exec -e OC_PASS="$password" \
        --user www-data nextcloud /var/www/html/occ \
        user:add --password-from-env "$username"
done
```

Such script can be executed as follows:

```
sh create_usr.sh <N>
```

where `N` is a required parameter to specify the number of users to create. Once user creation phase is terminated, the locust web user interface can be accessed from `http://localhost:8089/` (as configured in the `docker-compose.yml` file). One final command is required to perform load tests correctly, to ensure that the requests made by Locust are recognized as coming from a trusted source:

```
docker exec --user www-data nextcloud /var/www/html/occ \
    config:system:set trusted_domains 1 --value=nextcloud
```

All locust tasks were defined in a Python script and executed on my MacBook Pro M2 laptop. In order to assess the system's performance under varying stress conditions, I found it interesting to run two types of test. The first with files of small to medium sizes (from kilobytes to megabytes); the second introducing files with significantly larger dimensions (up to gigabytes). Both of them were done with a pool of 50 users.

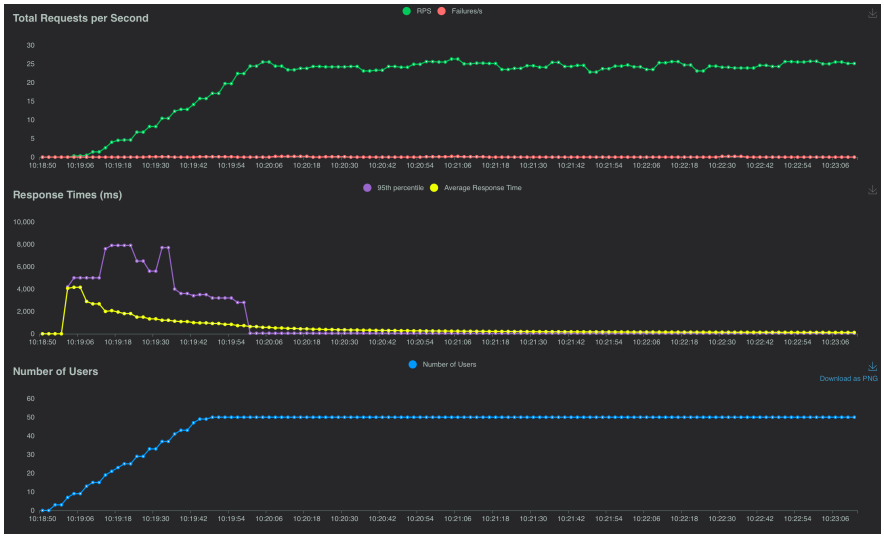

Figure 1: Test 1 with files of small-medium size

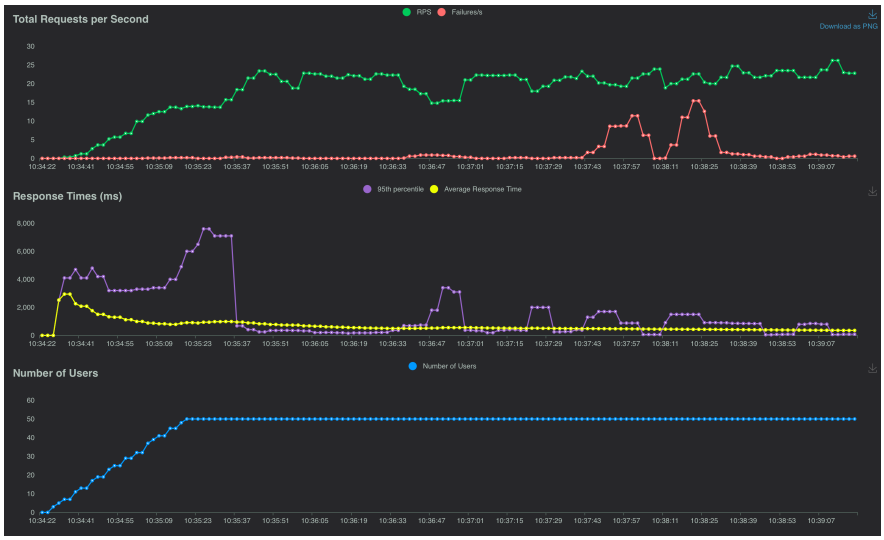

Figure 2: Test 2 with files of large size

Results of the first test (the one with files of small to medium sizes) were highly encouraging. Throughout a monitoring period of over 5 minutes, the system exhibited a remarkable stability with a constant 0% failure rate in request handling. Another interesting behavior can be spotted in the response times variability, particularly during the initial login phase. As users complete the authentication process, such variability seems to mitigate, hinting at an efficient resource allocation of the system once it reaches a stable state. Moving to the second test (the one involving larger file sizes), the system still displayed resilience and stability, despite its (expected) higher failure rate. Particularly interesting was the "post-failure" recovery observed in the second test: despite encountering failures, the system continues to execute subsequent operations successfully without disrupts.