Cloud Computing Exercise

Sara Carpenè

March 2024

1 Introduction

The aim of this exercise is to identify, deploy and implement a cloud-based file storage system. As suggested it has been used Nextcloud as cloud file storage and sharing software, the implementation consists of one Docker file which allows to create the Nextcloud container. For the part concerned testing and assessing the cloud it has been used Locust.

2 Docker

It has been decided to use Docker and Docker Compose to work with Nextcloud thanks to their's simplicity and maintainability. The availability of the Nextcloud image in Docker allows to manages the installation in a more immediate way, without dealing with dependencies or specificity of the underlying architecture.

They have been created two containers with the *docker-compose.yml* file available in the repository of this project.

The two containers are:

1. **db**: This service is based on the *mariadb:10.6* image and it is configured to run a *MariaDB* database.

   ```
   image: mariadb:10.6
   volumes:
     - db:/var/lib/mysql
   ```

   The volumes option mounts the db volume to persist the database data. This container will be used by Nextcloud to store various types of data, including user accounts, file metadata, sharing permissions, app configurations, and more. *MariaDB* is used here because it is efficient, reliable, and well-suited for handling the database needs of a Nextcloud instance, especially as the instance grows and handles more users and data.

2. **app**: This service is based on the *nextcloud* image and it's configured to run the Nextcloud application.

```
    image: nextcloud
    ports:
      - 8080:80
    links:
      - db
    volumes:
      - nextcloud:/var/www/html
```

The ports option maps port 8080 on the host to port 80 in the container, making Nextcloud accessible on port 8080. The links option links this service to the db service, allowing Nextcloud to communicate with the database. The volumes option mounts the nextcloud volume to persist Nextcloud data. The Nextcloud container is responsible for handling user requests, managing user accounts, file storage, and providing the web interface for users to interact with their files.

The separation of these components into two containers allows for better scalability, flexibility, and maintainability. Each container can be scaled, updated, or modified independently without affecting the other. This setup also facilitates easier backup and recovery processes, as data can be backed up or restored on a per-container basis. Additionally, using Docker containers for both the application and the database allows for a consistent and reproducible environment, which is crucial for development, testing, and deployment processes

3 Nextcloud

Containers can be started with the command:

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

This allows to go to the *http://localhost:8080* in which by signing in with the admin credentials will prompt the Nextcloud interface. Nextcloud allows to automatically let users to register in the cloud service, sign up, log in, and log out.

All users have their private storage space, it's dimension can be set from admin's profile (in this exercise it has been fixed to 1 GB). In their storage space users can upload, download and delete files with a user-friendly and intuitive interaction with the platform. From admin's profile users can be created and deleted, they can also be organized in groups and it can be set a default group in which new users will be automatically signed in.

3.1 Security

Regarding the security of the cloud system Nextcloud allows to enable different options.

Server-side encryption can be enabled from administrator setting in the admin's profile. This feature is important in addressing the security of the cloud service because it ensures that the content of files on external storage locations and on the server itself is protected while at rest. This means that even if unauthorized persons gain access to the storage, they will only see encrypted data, not the actual content

To secure user authentication it can be enabled also two factor authentication, this would add an additional layer of security to the authentication process by making it harder for attackers to gain access to users' files because, even if the victim's password is hacked, a password alone is not enough to pass the authentication check. Despite that, for the purpose of this exercise this option would complicate the management of users in testing phase so it has not been enabled.

Nextcloud offers the possibility to set various requirements for user passwords to enhance security and encourage users to adopt strong passwords, in order to prevent unauthorized access to the private storage. These requirements can be configured in the security section of the administrator settings, they are:

1. Minimal Length: It can be set a minimum length for passwords, which by default is 10 characters. This ensures that passwords are long enough to be secure.

2. Password Expiration Period: It can be enforced a period after which passwords must be changed. This helps in reducing the risk of long-term password exposure.

3. Password History: This feature allows to specify how many previous passwords must be different from the current one. This prevents users from reusing recent passwords.

4. Lockout Policy: This policy defines how many failed login attempts are allowed before the account is locked. This prevents brute-force attacks.

5. Forbid Common Passwords: Nextcloud can be configured to disallow commonly used passwords like 'password' or 'login', which are often targeted by attackers.

6. Enforce Upper and Lower Case Characters: Requiring passwords to include both uppercase and lowercase letters adds complexity and makes them harder to guess or crack.

7. Enforce Numeric Characters: Including numbers in passwords increases their complexity and strength.

8. Enforce Special Characters: Requiring special characters (e.g., ! or :) adds another layer of complexity to passwords.

9. Check Against Breached Passwords: Nextcloud can check passwords against a list of breached passwords from *haveibeenpwnd.com*. This helps in preventing the use of passwords that have been compromised in data breaches.

3.2 Scalability and cost efficiency

In order to handle increasing load and traffic, an option could be the horizontal scaling. Nextcloud is designed to be easily scalable using standard LAMP stack horizontal scaling technologies (a bundle of four different software technologies that developers use to build websites and web applications it includes Linux, Apache, MySQL and PHP), and it allows to grow the number of users by adding more infrastructures to the physical system. For a private usage this is the most suitable choice since it's simple and effective.

However, adopting this approach comes with a significant cost, as it would require purchasing new hardware to extend the cloud service's capabilities. This cost can be mitigated by using pay-as-you-go models, such as those offered by Amazon AWS. This model allows for payment only for what is used, keeping the organization agile and capable of responding to workload fluctuations without overcommitting the budget. Additionally, with AWS, organizations can benefit from massive cost savings, reducing variable costs compared to using physical infrastructure. This approach enables the elimination of the need to predict infrastructure capacity, increasing the organization's speed and agility.

If the level of required scalability is high, for example as it can be for a commercial company that needs to manage several thousands of users at the same time or that has a specific requirements for data locality, compliance, or performance that necessitate a more distributed architecture, then also Nextcloud Global Scale could be a viable option. The Global Scale architecture is based on a decentralized model where there are no central database, storage, or caching instances, as they may cause a bottleneck. Instead, it relies on independent nodes that can be located anywhere in the world, without the need for fast connections between them. This design allows for the use of commodity hardware, which is significantly cheaper than specialized petabyte servers, leading to lower maintenance costs and no significant license costs beyond the Nextcloud Enterprise subscription. It allows for the distribution of data over multiple hosting centers in different countries or even continents, which can be beneficial for legal requirements on where data is stored, to increase performance by bringing data closer to users, or for cost, security, or auditing reasons.

The architecture is governed by several components, including the Global Site Selector, Lookup Server, and Balancer. The Global Site Selector serves as the primary login instance for all global users, redirecting logins to the node closest to the user's physical location. This ensures minimal latency and a seamless federated user experience. The Lookup Server keeps track of user locations and stores policy data such as storage/quota settings, speed class, and reliability class. The Balancer operates as a dedicated machine, overseeing the storage usage, CPU and RAM loads, network usage, and uptime of every node, and can designate nodes as either online or offline and trigger the relocation of user accounts to alternative nodes if necessary.

4 Locust

Locust.io is an open-source, Python-based performance testing tool that enables developers to evaluate their systems' performance under high user load. It allows for the simulation of multiple users, providing detailed insights into system performance and identifying potential failure points. In this case it has been used to test the performances of the system in term of load and IO operations.

It has been written a python scripts named *filelocust.py* that if run with the locust command allows to open a web interface at the link *http://localhost:8089*. Therefore it is required to set the number of users and the ramp up rate (user/second). For this exercise the number of users has been set to 28 and the ramp up rate to 15.

Before starting the test it has been created 28 users with a simple bash script. The results of performing several tests with different sized file are reported in this section. Each test involves simulating the login of 28 users, with various tasks defined in the *locustfile.py* including propfind, upload and delete of a file, as well as get. Each task is selected by each user with equal probability. When these users run, they pick tasks to execute, sleep for a while, and then pick a new task, and so on.

4.1 Small files

The first test was performed with a file of dimension $< 1KB$. From the data reported by locust it can be seen that the average response time varies widely across different requests, with some requests having very low response times (e.g., GET requests averaging around 390 ms) and others having much higher response times (e.g. the HEAD request has a median response time of 30000 ms. This suggests that this specific request is particularly resource-intensive or that the system is not optimally handling these requests under load). All requests have a failure rate of 0, this suggests that the system is robust under the tested load.

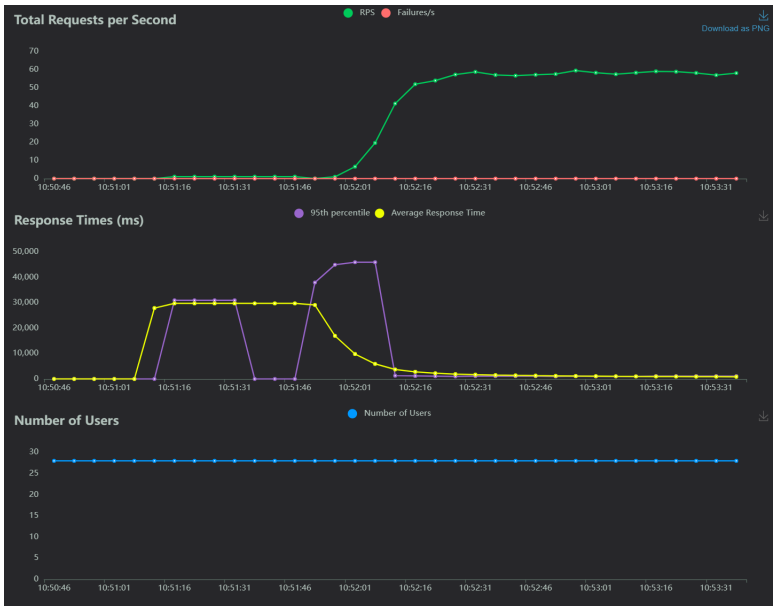

Figure 1: File of dimension $< 1KB$

4.2 Medium files

The median response time for most requests is higher than the one of small file, with several requests exceeding 8000 ms (8 seconds). This indicates that the system may struggle under high load, as users experience significant delays in receiving responses. Also in this case the HEAD request is the one requiring the most time to complete. From the figure 2 it can be seen that the average response time is quite stable, while the 95th percentile reaches a pick an then decrease.

4.3 Big files

For big sized messages the underlying architecture (my Lenovo laptop with processor: Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz, RAM: 8,00 GB (7,80 GB utilizzabile), OS: Windows 11 pro) wasn't able to manage 30 users so the test have been performed with 4 users. The results show that the response time for every operation is significantly higher than the one of the previous test and it's value doesn't decrease during the time. Another interesting thing to notice is that the number of failures is still 0, suggesting that the system is good enough to handle quite big sized files.

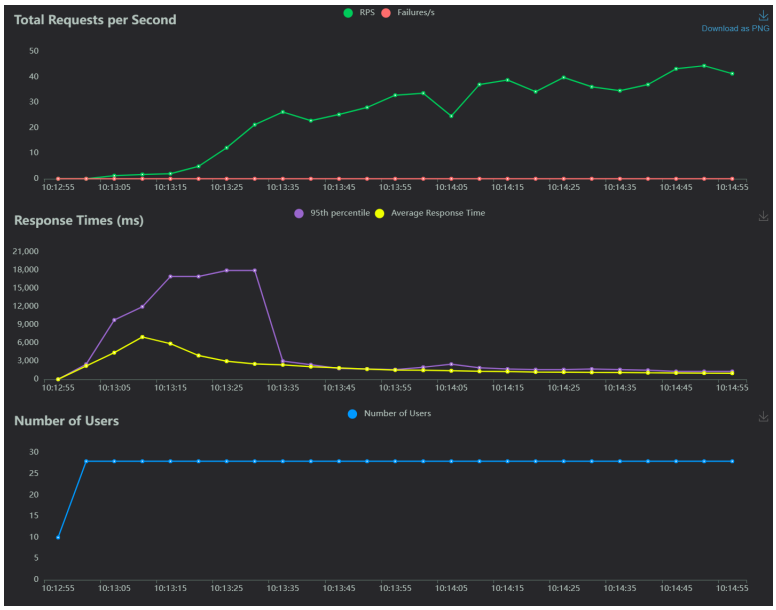

Figure 2: File of dimension 1MB

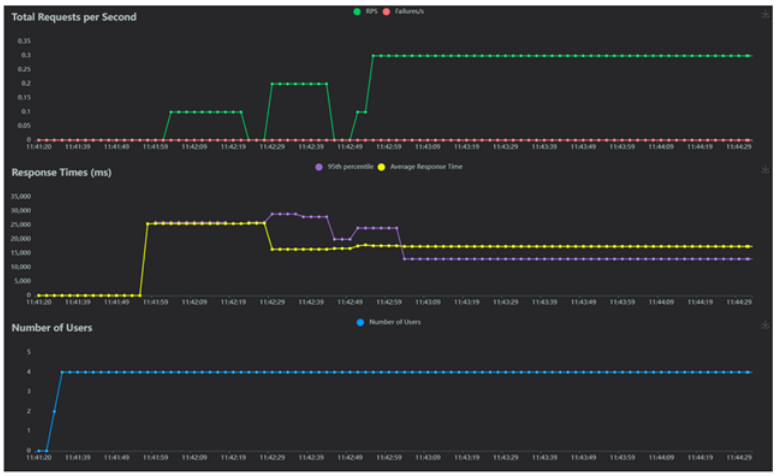

Figure 3: File of dimension 50MB

.

5 Bibliograpy

1. https://docs.nextcloud.com/server/latest/admin_manual/configuration_user/user_password_policy.html
2. https://docs.locust.io/en/stable/writing-a-locustfile.html#writing-a-locustfile
3. https://nextcloud.com/blog/nextcloud-global-scale-how-it-works/
4. https://nextcloud.com/media/wp135098u/GlobalScale-Whitepaper-WebVersion-072018.pdf
5. https://jtlreporter.site/blog/how-to-analyze-locustio-report