Cloud-Based File Storage System

Gaia Alessio

25 March 2024

1 Introduction

The objective of this project was to deploy a Cloud File System enabling users to upload, download, and delete files while preserving the privacy of their storage spaces. This system was developed using containerization with Docker and Docker-Compose.

2 Nextcloud

Nextcloud emerged as the preferred platform for various reasons. It is an open-source project with a large community of developers and users, along with comprehensive documentation. It offers ease of implementation with a user-friendly interface, strong security features, and scalability.

The availability of a preconfigured Docker image ensured a straightforward and lightweight deployment process, facilitating easy and efficient usage.

3 User Authentication and Authorization

Nextcloud facilitates user authentication and authorization with its built-in features.

- User Registration: Nextcloud provides a streamlined user registration process, enabling users to easily sign up, log in, or log out.
- Role-Based Access: Nextcloud supports various roles, including admins and regular users, each with appropriate privileges. Admins are granted necessary permissions to manage users, and in the absence of existing admins, the first registered user assumes admin status.
- Private Storage Space: Each regular user automatically receives a private storage space that by defaults amounts to 512MB.
- Ability of Admins to Manage Users: Through an intuitive dashboard, admins can create, modify, and delete user accounts and permissions. This functionality is crucial for maintaining system security.
- File Operations: Users can upload, download, and delete files from their private storage through the Nextcloud interface.

Authentication and user registration are key parts of Nextcloud's security setup. After logging in successfully, Nextcloud gives clients an access token for future requests. This token must be treated carefully and only kept by the client that requested it.

Nextcloud also keeps user passwords safe by encrypting them in its database. This additional protection ensures that passwords are protected from unauthorized access.

While Nextcloud doesn't allow clients to sign up independently, admins can create accounts for them. However, users can still register by themselves using a common email-based signup method.

4 Address Scalability

To manage an increasing amount of users and files, one strategy could be to expand the virtual disk of the Docker container. This allows you to increase the space available for file storage without necessarily having to change the underlying architecture.

Another approach is to deploy a distributed architecture to distribute the load across multiple servers. For example, you can use an architecture where each service runs on separate servers. This approach distributes database load and ensures optimal performance as the number of users and files increases.

To effectively manage increased load and traffic, you can use a horizontal approach, configuring multiple instances of Nextcloud and distributing the load between them to ensure a fair distribution of resources and greater system reliability.

5 Address Security

To ensure the security of the Nextcloud cloud, some security measures can be implemented for both file storage and user authentication.

First, server-side encryption is used to protect file storage. Activation can be done directly from the administration security settings:

Administrator settings -> Administrator security -> Server-side encryption.

This encryption allows files uploaded to the server to be encrypted, providing greater data security. However, it may be noted that enabling encryption can affect system performance, as it increases the computational load required to manage the files.

Similarly, stricter restrictions can be applied to the password policy, mandating the entry of special numbers and characters or requiring that the password must be changed every few days. To prevent unauthorized access and ensure the security of user authentication, you can also enable two-factor authentication, which requires in addition to the password, a second method of identity verification such as a code generated by a Time-Based One-Time Password (TOTP) application. This additional measure improves account security by reducing the likelihood of unauthorized access even if the password is compromised.

6 Cost-Efficiency

In a local setup, the costs are minimal and mostly related to the server's hardware, like storage devices, CPU, RAM, and other parts needed to run the system. Also, there are operational costs to consider, such as electricity usage and hardware maintenance over time.

On the other hand, online deployment involves the use of third-party cloud services such as AWS. These services work on a "pay-as-you-go" basis, where customers only pay for the computing and storage resources they actually use. There's no need for long-term contracts or upfront purchases.

Under this model, customers are charged based on the actual amount of resources used during a given time period, such as hours of use, amount of storage used, outgoing network traffic, and other resources consumed. Although they eliminate the need for up-front investment on hardware and reduce maintenance costs, they incur ongoing operating expenses based on actual usage.

This approach offers flexibility and scalability to customers, allowing them to easily adapt resources to specific workload needs.

To optimize cost efficiency, several strategies should be adopted such as:

- Ensure that allocated resources, such as CPU, RAM, and storage, match the actual workload requirements. Avoid oversizing to reduce unnecessary costs
- Take advantage of the automatic scaling capabilities offered by cloud platforms to automatically adjust resources according to demand. This ensures that you only pay for the resources you use, optimizing cost efficiency.
- Continuously monitor resource utilization to identify areas for optimization.

Hybrid deployment is an infrastructure model that combines the use of both local and cloud resources. Some workloads and data are managed locally within your enterprise infrastructure, while others run and store in the cloud.

This approach lets companies benefit from both methods. They can adjust their resources easily to match current needs, especially during busy times, without needing to buy new hardware. By keeping some resources and data in their own data center, they can also have more control over security. Hybrid deployment helps companies make their IT infrastructure costs more efficient. They can use the cloud for temporary tasks and keep critical resources on-site for tighter control.

In summary, hybrid distribution provides a balance between flexibility, control and cost reduction, allowing companies to make the most of the opportunities offered by both infrastructures.

7 Deployment

To provide a deployment plan for the system in a containerized environment on your laptop based on docker and docker-compose the first step is clearly to install Docker and Docker Compose on your laptop if they are not already installed.

Then create a ‘Docker-compose.yml‘ file which is a configuration document that defines the environment for running Nextcloud, a file storage application, along with a MariaDB database. This environment consists of two main services, each corresponding to a separate container:

The first service, named db, starts a container using the MariaDB version 10.6 image. This container represents the MariaDB database used by Nextcloud to store data.

The second service, named app, starts another container using the Nextcloud image.. It is configured to map port 8080 of the container to the local host on port 80, allowing access to the application through the Web browser.

Both services are configured with the restart:always option, which means that the containers will be automatically restarted in the event of an error or unexpected shutdown.

In summary, through the Docker Compose file, two separate containers are created: one for the MariaDB database and one for the Nextcloud application. These containers work together to provide a complete execution environment for Nextcloud.

Ensure that containers are started and initialized properly by running ‘Docker-compose -d‘ in the directory containing the file ‘Docker-compose.yml‘ . This command creates and starts the containers. Finally, access Nextcloud through the Nextcloud URL http://localhost:8080.

For Nextcloud production deployment, Amazon Web Services (AWS) is a suitable choice due to its robust infrastructure, scalability, and comprehensive set of cloud services. AWS has an extensive global data center network, allowing you to deploy Nextcloud instances closer to users for lower latency and

better performance. AWS offers scalable compute, storage, and networking resources that can be dynamically adjusted to fit variable workloads. You can easily scale up or down according to demand without incurring any initial costs.

AWS provides managed services such as Amazon S3 for object storage that simplify the deployment and management of Nextcloud infrastructure.

AWS adheres to industry-leading security standards and compliance certifications, ensuring that your data is protected and meets regulatory requirements. AWS offers a pay-as-you-go pricing model, allowing you to pay only for resources consumed without long-term contracts or initial commitments.

By deploying Nextcloud on AWS, you can benefit from a highly available, scalable, and secure cloud infrastructure that meets your organization's performance, reliability, and cost efficiency needs.

8 Testing

When assessing system performance using Nextcloud and Locust, it's essential to account for both the generated workload and the input/output (IO) operations involved in data access and management.

Employing Locust to simulate user load on the Nextcloud platform allows for evaluating system behavior across various load levels. Tasks with diverse operations can be defined, and performance metrics can be monitored during these simulations to gauge the system's workload management capabilities.

IO operations, like file uploads and downloads from Nextcloud, significantly impact overall system performance. It's critical to assess processing speed and latency associated with these operations. Conducting specific tests, such as measuring upload/download times for files of different sizes, aids in evaluating IO performance.

Upon identifying any system weaknesses through load testing and IO operation assessments, appropriate optimizations can be applied. Continuous performance monitoring post-optimization ensures sustained performance levels.

In summary, considering workload generation and IO operations is pivotal for evaluating and optimizing Nextcloud system performance using Locust. By meticulously evaluating and implementing suitable optimizations, the system can effectively address user needs and maintain high performance standards.

Before starting the tests, it is necessary to create a set of user accounts. This is done using a bash script executed with the command

chmod +x adduser.sh

and then

./adduser.sh

Next you can create the locustfile.py file to run the tests. Finally, the locust web interface can be accessed from http://localhost:8089/.

8.1 Testing with small file

This test was performed with 30 simultaneous users for 2 minutes. The graph shows a gradual increase to reach 30 users. 100% of the requests were made by users using Nextcloud, with actions equally divided between "PROPFIND" and "uploadSmall".

There were 2,514 requests with no failures, indicating that the application is stable under the tested workload. The number of requests per second (current RPS) indicates how many requests per second the system can handle. An average of 38.5 RPS indicates good traffic handling capacity under the test load.

Response time statistics provide details on various HTTP requests (HEAD, PROPFIND, DELETE, GET, PUT) with corresponding percentiles.

- HEAD with 30 requests shows a high average response time (8673.31 ms).
- PROPFIND with 297 requests with an average response time of 2000.87 ms, showing better performance than HEAD requests, but still with room for optimizations, especially considering the maximum response time (45646 ms).
- DELETE with 302 requests has an average response time of 1606.68 ms, showing good efficiency in handling file deletion requests.
- GET with 1,564 requests shows the lowest average response time (616.19 ms), indicating that file retrieval operations are relatively fast.
- PUT with 321 requests has an average response time of 2618.35 ms, suggesting that file loading may take time and could benefit from optimizations.

The wide variation in response times, especially in higher percentiles, suggests that there are opportunities to optimize performance.

Figure 1: Small File Test

8.2 Testing with large file

This test was performed with 30 simultaneous users. Response time statistics provide details on various HTTP requests (HEAD, PROPFIND, DELETE, GET, PUT) with corresponding percentiles.

- The HEAD request shows a high median response time of 12 seconds, with a maximum of over 20 seconds. This could indicate a bottleneck in server response for these requests.
- The PROPFIND request has a better average and median response time, but it also reaches a high maximum, suggesting occasional delays.
- The GET and PUT requests for a large file (large.txt) also show a high median and average response times, which is not unusual for large file operations but could point to performance issues with handling large file transactions.

The 50th percentile (median) response times range from 1.2 seconds for PROPFIND to 28 seconds for GET of a large file, indicating the relative performance of each operation. Increasing percentile values for HEAD and PROPFIND requests indicate that a greater percentage of these requests have longer response times, with some PROPFIND requests taking up to 46 seconds.

In summary, the high response times, especially for HEAD and file operations, indicate that these are areas that could benefit from performance optimization.

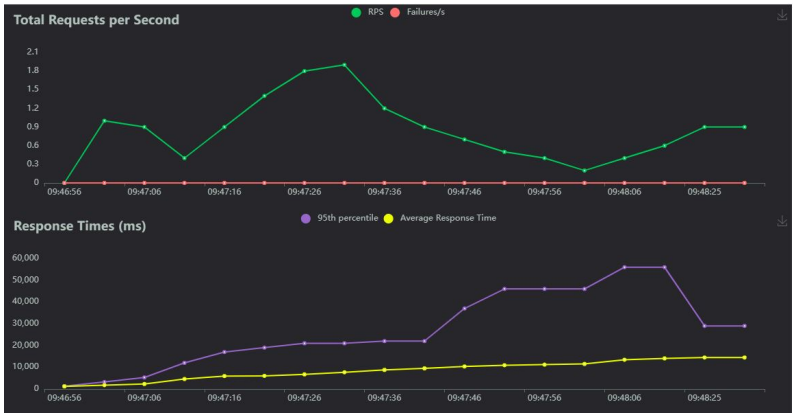

Figure 2: Large File Test