Università degli studi di Trieste

CLOUD-BASED FILE STORAGE SYSTEM

Student

FU Ziyang
SM3800011

MSc in Data Science & Artificial Intelligence

Foundations of Artificial Intelligence and Machine Learning

Trieste, 19th March 2024

Contents

1 Introduction

As organizations and individuals continue to generate and manipulate vast amounts of data, the need for a robust cloud-based file storage system has never been more critical. In response to this demand, the implementation of a cloud-based file storage system capable of facilitating file upload, download, and deletion functionalities while ensuring user privacy, scalability, security, and cost-effectiveness becomes imperative.

This report outlines the approach taken to address this requirement, focusing on the employment of Nextcloud as the chosen solution.
This latter has been the designated solution thanks to several reasons:

- extensive built-in applications that facilitates the configuration;
- seamless UX combined with an easily undestandable UI;
- ease of deployment through Docker containers.

Later in this report the details of the listed topics will be discussed.

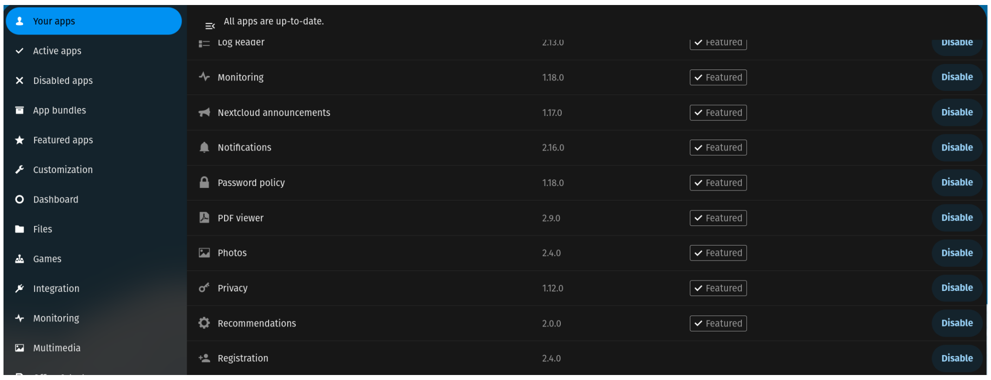
2 User Authentication

This section of the report is designed to provide an overview over the registration and the regular user space action in Nextcloud.

We will explain the following sub-section under the point of view of the **administrator** of the system.

2.1 Registration

Nextcloud present a built-in feature that enables user's registration in a seamless manner. As administrator we are simply asked to carry out one simple task: **enable the registration application from the featured applications**: we just need to navigate to the **Apps page** by clicking on the profile avatar or in the top-right corner and then select "Apps" from the dropdown menu and install the Registration app by searching it.



All apps are up-to-date.

Log Reader	2.13.0	✓ Featured	Disable
Monitoring	1.18.0	✓ Featured	Disable
Nextcloud announcements	1.17.0	✓ Featured	Disable
Notifications	2.16.0	✓ Featured	Disable
Password policy	1.18.0	✓ Featured	Disable
PDF viewer	2.9.0	✓ Featured	Disable
Photos	2.4.0	✓ Featured	Disable
Privacy	1.12.0	✓ Featured	Disable
Recommendations	2.0.0	✓ Featured	Disable
Registration	2.4.0		Disable

Figure 2.1: *Built-in application dashboard*

Once this has been done as admins we may need to configure different users groups and/or users roles, allowing to allocate the suitable levels of privileges.
Additional to granting different privileges levels, we can provide different users a specified private storage quota.

As before Nextcloud's UI grants to administrators a seamless experience in managing the system users via an intuitive dashboard:

1. navigate to the **Users** page by clicking on the profile avatar or in the top-right corner;
2. on the left-side menu we can manage the users' settings.

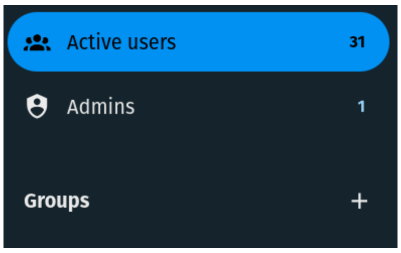
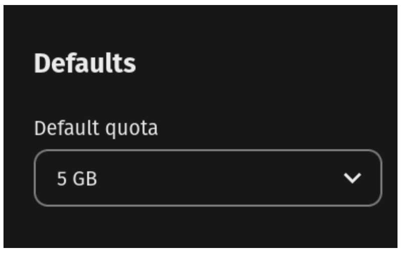


Figure 2.2: *Users groups*


Defaults

Default quota: 5 GB

Figure 2.3: *Storage quota assignment*

2.2 Authentication system

Users can be added using two approaches:

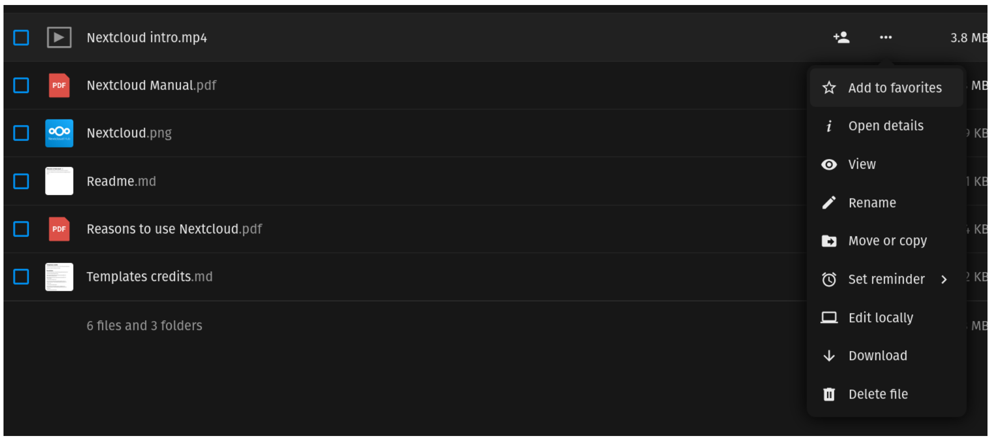
- administrator may manually add the desired users from the Users page;
- thanks to the registration app, users can register themselves through the web interface entering the classical registration components: a valid email address, username and a password that satisfy the set requirements.

How secure user authentication and prevention of unauthorized accesses are and could be implemented will be discussed in later sections of the report.

2.3 File operations managements

The basic file operations that one user could possibly carry out in a Cloud-Based File Storage System are all made possible again thanks to Nextcloud built-in features, that combined with its intuitive user-interface allows users to

- upload files to their private storage;
- download files from their private storage;
- delete files from their private storage.



Nextcloud intro.mp4 3.8 MB
Nextcloud Manual.pdf
Nextcloud.png
Readme.md
Reasons to use Nextcloud.pdf
Templates credits.md
6 files and 3 folders

Figure 2.4: *Private user file management tools*

3 Security

3.1 Users' security

Firstly the users authentication should be addressed: we need to set up a server email account to provide the verification links and enforce proper passwords etiquette

Nextcloud does not contain a full email server, but rather connects to an existing mail server, so an already existing and functioning mail server should be at our disposal.
Once checked this condition a few last steps are required:

1. navigate to the **Administration settings** page by clicking on the profile avatar or in the top-right corner;
2. scroll down on the left-side menu until the section **Basic settings**;
3. fill the Email server configurations form;

Secondly we need to set up a proper password etiquette: we simply need to configure the **Password policy** section of the security page that is also in the Administration settings.

A final, but crucial security measure is to implement a secure file storage to prevent undesired files reading operations: Nextcloud allows to do so by enabling file encryption on the server side.
This will reduce the performance of the system but will prevent an intruder that gains access to the data to read it. All this can be done from the same page of password policy.

From this same page we can enable a further security measure, adopted today by almost every internet services,**Two-factor authentication**.
This measure combined with Nextcloud default configuration in the config.php file of the instance

```
'auth.bruteforce.protection.enabled' => true,
```

should prevent unauthorized accesses:

3.2 Clients-Serve interaction

Although not implemented in this assignment, production-wise is mandatory to enforce HTTPS protocol for clients-server communication in order to encrypt the communication and to prevent man in the middle attacks and data snooping.

An common approach would be to set up a reverse proxy container in Docker for the Nextcloud instance and to run it with a self-signed SSL/TLS certificate or obtain it from certificate authority.
The following code snippet shows a possible yaml file implementation:

Listing 3.1: *compose yaml*

```
app:
  image: nextcloud
  restart: always
  volumes:
    - nextcloud:/var/www/html
  environment:
    -  ... # database settings

nginx: # our designated reverse proxy
  image: nginx:latest
  volumes:
    - ./nginx/cert:/etc/nginx/cert
    - ./nginx/conf.d:/etc/nginx/conf.d
  ports:
    - 443:443 # HTTPS ports mapping
  links:
    - app
```

By specifying **443:443** we indicate that port 443 (the one handling all the HTTPS traffic) on the host machine is being mapped to port 443 on the container, so any traffic directed to port 443 on the host machine will be forwarded to port 443 within the Nginx container.
This mapping allows external clients to access services running inside the container through the specified port.

4

Scalability and deployment

This section is designed to address the performance of the system in terms of load and IO operations and to discuss theoretically how to handle increased load and traffic.

Before entering into the core subject of this section we will just briefly address how the tested system was deployed.

4.1 Deployment

The deployment has been done through the usage of docker and docker-compose. We leverage docker-compose and the compose.yml to connect a MySQL backend database to our nextcloud instance.

To run the two docker containers, we just need to cd to the folder containing the docker-compose.yml file and run: **docker-compose up**

4.2 Performance and Testing

To test the performance of the system it comes handy to design stress tests and see how the infrastructure reacts. The **Locust** python package has been be used to simulate user interactions with the platform, in order to collect performance metrics.

The first thing that needs to be done is to create dummy users that will later be used to flood the system with **PUT** and **GET** requests.
This has been achieved through the following bash command:

```
for i in {1..30}
do
    docker exec -e OC_PASS=ziyang1234! --user www-data nextcloud \
    /var/www/html/occ user:add --password-from-env user$i
done
```

Now we need to define a series a tasks that the newly created dummy users will try to carry out in the system. Specifically the designed tasks are the following:

- **opening** a README.md;
- **uploading** files of size 1kb and 1mb;
- **deleting** the files from the previous bullet point.

These tasks has been defined within a **testing.py** python script, that also takes care to generate the files to be uploaded on the fly.

Locust offers us well designed web interface from which we can carry out the depicted tests and to monitor in real time the performance metrics. The web application can be reached from **http://localhost:8089/**.

The designed performance test generates concurrent user activities, up to thirty concurrent users, on the locally hosted Nextcloud.

The graphs in the following page present two collected metrics:

- **total requests per second** handled by the system;
- **response time** to requests;

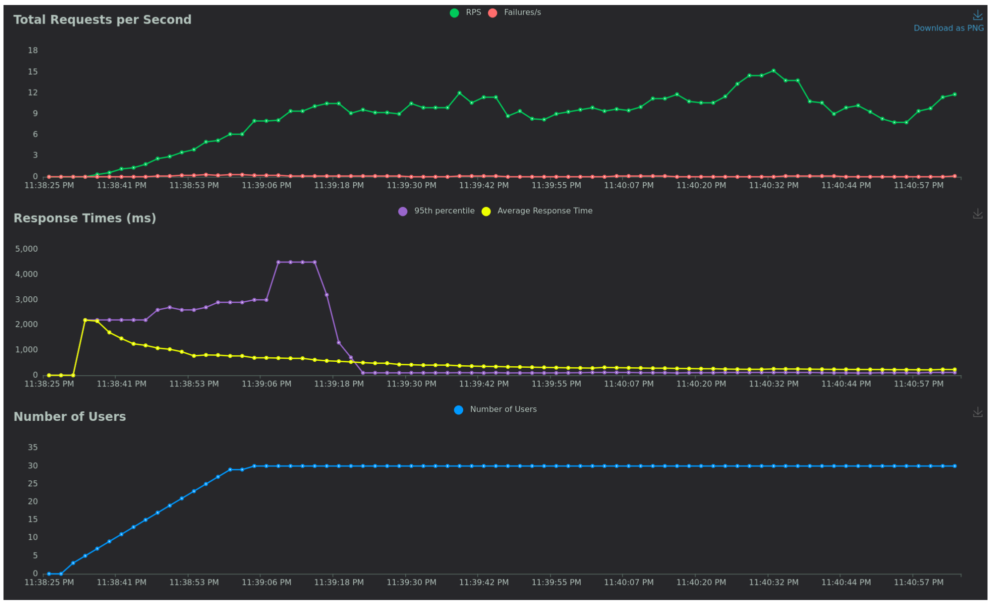


Figure 4.1: *Perfomance metrics*

As one can observe from the fist line chart, the locally hosted Nextcloud deals with no failures the PUT and GET tasks of the thirty concurrent users.

4.3 Scalability

Before diving into the discussion of how to theoretically deal with an increased load and traffic, we take a look at a simple proposed solution to minimally scale the system locally.

We set up a multi-container (precisely two in this assignment) environment for running two Nextcloud instances, a MySQL database, and an Nginx web server as a reverse proxy.
Hypothetically with multiple Nextcloud instances, there could be improved concurrency and parallelism for handling user requests, leading to better responsiveness under high load conditions, as requests can be distributed across multiple instances.
Additionally the reserve proxy provides built-in **load balancing routines** that can better the measured metrics. For the scope of this assignment the **Least Connections** routine has been made use of: it redirects a request to the server with the least number of active connections.

Following we can see the performance metrics of this implementation:

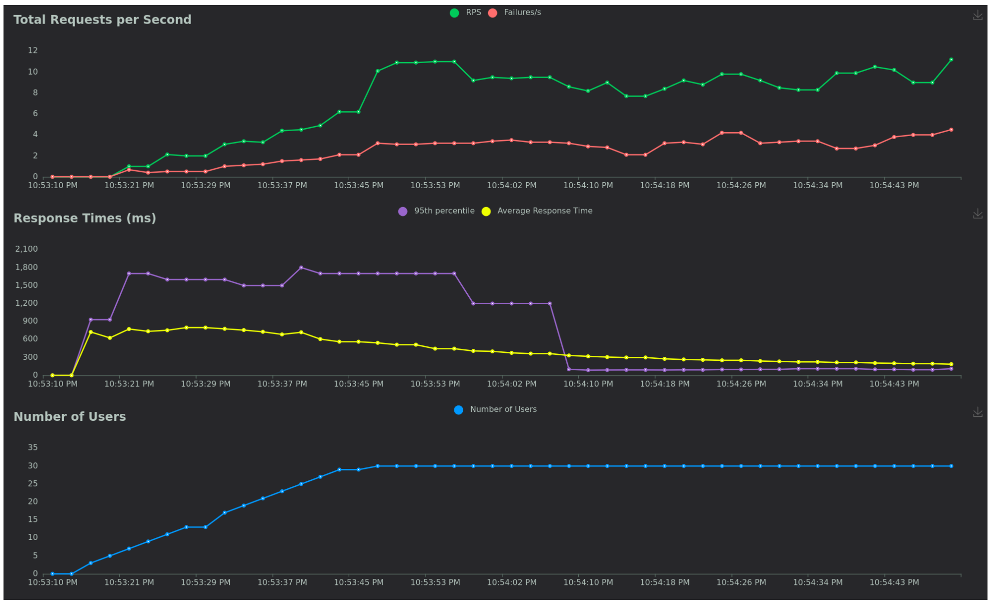


Figure 4.2: *Perfomance metrics*

As we can expect, by observing the second line chart we can notice that the response time has slightly improved compared with the single instance solution, but turning our attention to the first graph, the system now starts to present failures and the number of requests per second handled is not improving.

Now we can discuss how to theoretically handle an increasing load and amount of traffic.
As any task in this world we can either do it by ourselves or outsource it to more resourceful people.

4.3.1 Internal Solution

With in an organization that is already in possess of a cluster of nodes, it could be feasible to develop an internal solution. In this prospective tools like **Docker Swarm** really come handy.

Docker Swarm is a container orchestration tool for clustering and scheduling Docker containers. With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system. Docker Swarm enables developers to join multiple physical or virtual machines into a cluster.

So the basic idea is to use Docker Swarm to deploy and manage Nextcloud as a set of Docker containers within the already existing nodes infrastructure. Docker Swarm will handle tasks like distributing Nextcloud containers across different servers, ensuring high availability, and scaling the application based on demand.

The key feature to put under our lens is **automated load balancing**: Swarm assigns containers to the physical nodes and optimizes resources by automatically scheduling container workloads to run on the most appropriate host. Such orchestration is key to make sure that containers are only launched on systems with adequate resources to maintain necessary performance.

The benefits of this kind of design can be the followings:

- **total control**: since it's on-premise, we have full control over data and infrastructure;
- **flexibility**: Docker Swarm allows us to easily deploy and manage containers across the existing nodes, scaling the resource consumption to meet the actual demand. Moreover the we can design security measures that are tailored to the organization's needs;

Money-wise this implementation would require significant initial investments: physical nodes and software development need to be put together. If we don't have at our disposal the human resources to build the depicted software-hardware ecosystem we could end in accounting **large initial capital expenditure**. In addition this self-owned solution would need regular maintenance.

4.3.2 Outsourcing: Microsoft Azure with Azure Blob storage

A solution nowadays always more popular is to rely upon an external cloud provider. The market offers us various competitors to choose from, but in this solution I would like to refer to **Microsoft Azure**.

Azure offers the infrastructure capacity to support both vertical and horizontal scaling and embraces modern practises such as an object storage solution: **Blob**. It optimized for storing massive amounts of unstructured data. This combination achieves the desired feature of auto-scaling to handle variable load and traffic.

The **advantages** offered by an external cloud provider could be the following:

- **scalability**: we can easily provision additional compute, storage, and networking resources as needed without the need to purchase and manage additional physical hardware;
- **modern standards**: technological giants like Microsoft offer state of the art features, like **object storage**, a must have for businesses that handle a great variety of data;
- **global reach**: Azure has a global network of data centers, enabling to deploy applications and services closer to our users, resulting in lower latency and better performance. This will results in a better customer experience and could possibly enhance remote workers productivity;
- **a large line-up of databases**: Azure Cosmos DB offers fast, distributed NoSQL and relational database at any scale (PostgreSQL, MongoDB, and Apache Cassandra).

Of course all this comes with costs, mostly:

- **usage-Based Costs:** Cloud providers typically charge based on resource usage, which includes computing resources, storage, and data transfer;
- **Storage costs:** the more data we need to store the greater are the fees we will be charged for.

The optimal choice between an internal solution and outsourcing really depends on multiple factors.

For example if we're planning an exponential customer-base growth, the on-premise solution might not be suitable: the time needed to develop the internal solution is mostly likely greater than the rate at which new customers will require our services, thus we will not be able to match the demand.
In this scenario relying on external providers might be the solution that maximizes the trade-off between growing variable costs and guaranteeing the availability of the service for steady revenue streams.

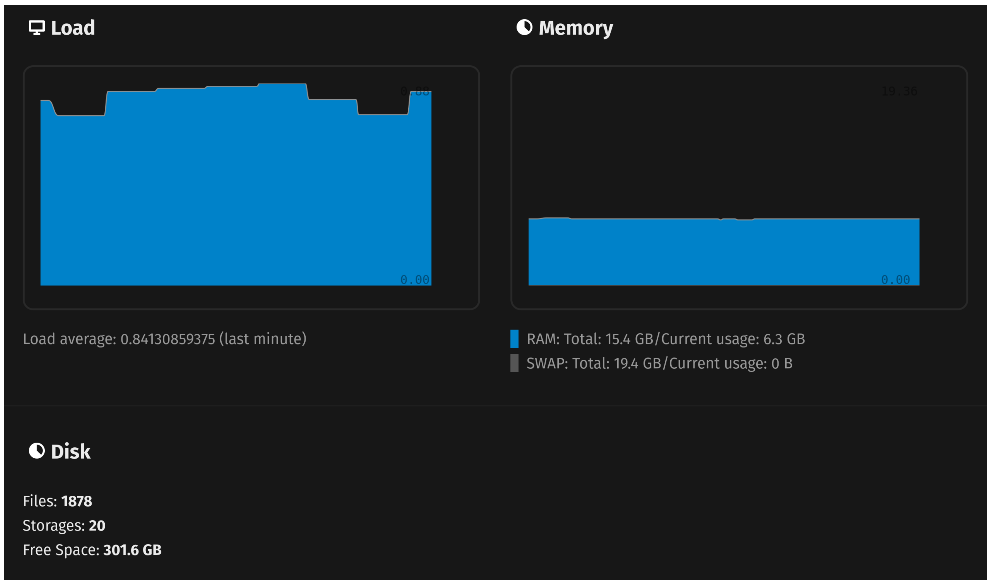
On the contrary a well-established business, or one with a strong local customer-base might not need the degree of scalability provided by an external cloud provider: in the long run variable usage-based and storage costs can overweight the investment needed to develop an on-premise solution.

5 Monitoring

Nextcloud provides a built-in monitoring tool to track the performance of our system.
All we have to do is just following these simple steps to enable the monitoring app:

1. login as admin;
2. navigate to the **Apps page:** click on your profile avatar or username in the top-right corner, then select "Apps" from the dropdown menu;
3. install the Monitoring app: in the Apps page scroll down to find the "Monitoring" app and enable it.

Once done this simple procedure we can monitor the system's performances easily by navigating to **Administration setting**, scrolling down on the side menu until reaching the section **System**.
This section offers us a comprehensive dashboard, that comprises a series of metrics.



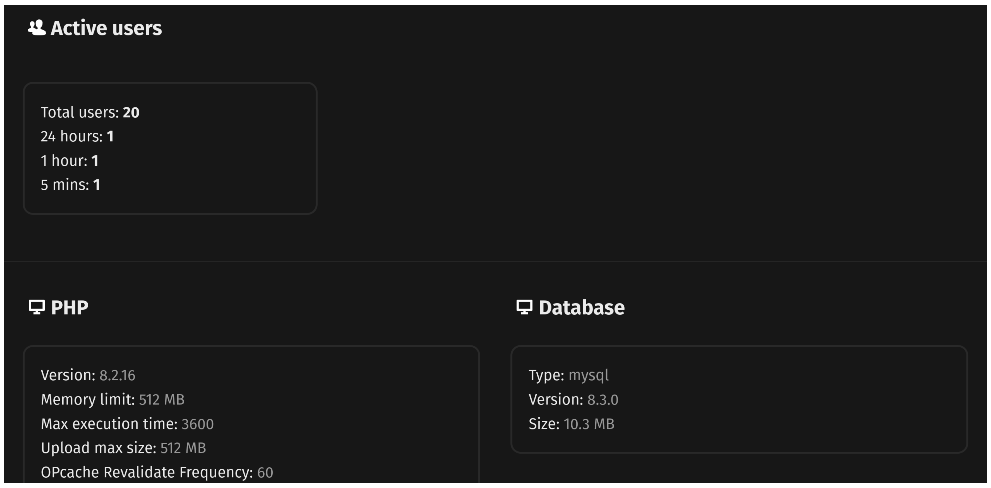
Load average: 0.84130859375 (last minute)

🟦 RAM: Total: 15.4 GB/Current usage: 6.3 GB
⬛ SWAP: Total: 19.4 GB/Current usage: 0 B

Disk

Files: **1878**
Storages: **20**
Free Space: **301.6 GB**

Figure 5.1: *Monitoring dashboard figure 1*



Active users

Total users: **20**
24 hours: **1**
1 hour: **1**
5 mins: **1**

PHP

Version: 8.2.16
Memory limit: 512 MB
Max execution time: 3600
Upload max size: 512 MB
OPcache Revalidate Frequency: 60

Database

Type: mysql
Version: 8.3.0
Size: 10.3 MB

Figure 5.2: *Monitoring dashboard figure 2*

6 Conclusions

We explored a possible implementation of a cloud-based file storage system using Nextcloud: its extensive set of features allowed us to be compliant to the following assignment requirements:

- be able to manage User Authentication and Authorization;
- allow users to manage File Operations;
- security measures, among which secure file storage thanks to server-side encryption;
- monitoring the performances of the deployed system.

Additionally Docker container significantly simplifies the development and deployment process of the system in a containerized environment on the local laptop.

The deployed system has been tested to highlight its performances and then finally two diametrically opposed deployment solutions have been discussed: internal solution leveraging on on-premise clusters and choosing an external cloud provider.
From this section we stated how an absolute optimal solution does not exist, it all depends on the business current development stage and needs.

This project highlights the importance of how a the design of a comprehensive solution and system really needs to take on an holistic approach: features, security, scalability, and cost-effectiveness are all need to be considered.

This page intentionally left blank.