DATA SCIENCE &
ARTIFICIAL INTELLIGENCE

CLOUD-BASED FILE STORAGE SYSTEM DEPLOYMENT

Roshanak Behrouz
DSAI
Cloud basics
University of Trieste
March 2024

TECHNOLOGIES USED

1. **Docker**
2. **Docker Compose**
3. **Nextcloud**
4. **Nginx**
5. **Locust**
6. **MariaDB**

Why?

DEPLOYMENT

- I've created a Docker Compose setup (version 3) to orchestrate a system comprising two MariaDB databases, two Nextcloud instances, and an Nginx server functioning as a load balancer.
- Databases (db1 and db2): Utilizes MariaDB images to establish two separate containers for databases, with persistent volumes for data storage. Environment variables configure database credentials.
- Nextcloud Instances (app1 and app2): Two distinct Nextcloud services are set up, each automatically restarting on failure. They're linked to their respective databases through environment variables for seamless database connectivity. Dependency settings ensure they start only after their respective databases are ready. Ports 8081 and 8082 are mapped to make the Nextcloud interfaces accessible externally.
- Nginx: Configured as a load balancer, it directs incoming host traffic on port 80 to the appropriate Nextcloud instance. The configuration for load balancing is provided through a mounted nginx.conf file. It's set to start only after both Nextcloud services are operational.
- Volumes (db1 and db2): Named volumes for the MariaDB containers ensure data persistence across restarts or removals.
- Network (nextcloud_network): A custom network facilitates communication among containers using their names, simplifying the orchestration of inter-service interactions.

USER MANAGEMENT AND SECURITY AND COST-EFFICIENCY

- Registration
- User generation
- create different file size
- Users privileges'
- User roles
- Admins

- Strong Password policy
- Usage server report
- Server-side file Encryption
- Updates and Backups

- Opensource Technologies

Scalability?

LOAD TESTING WITH LOCUST

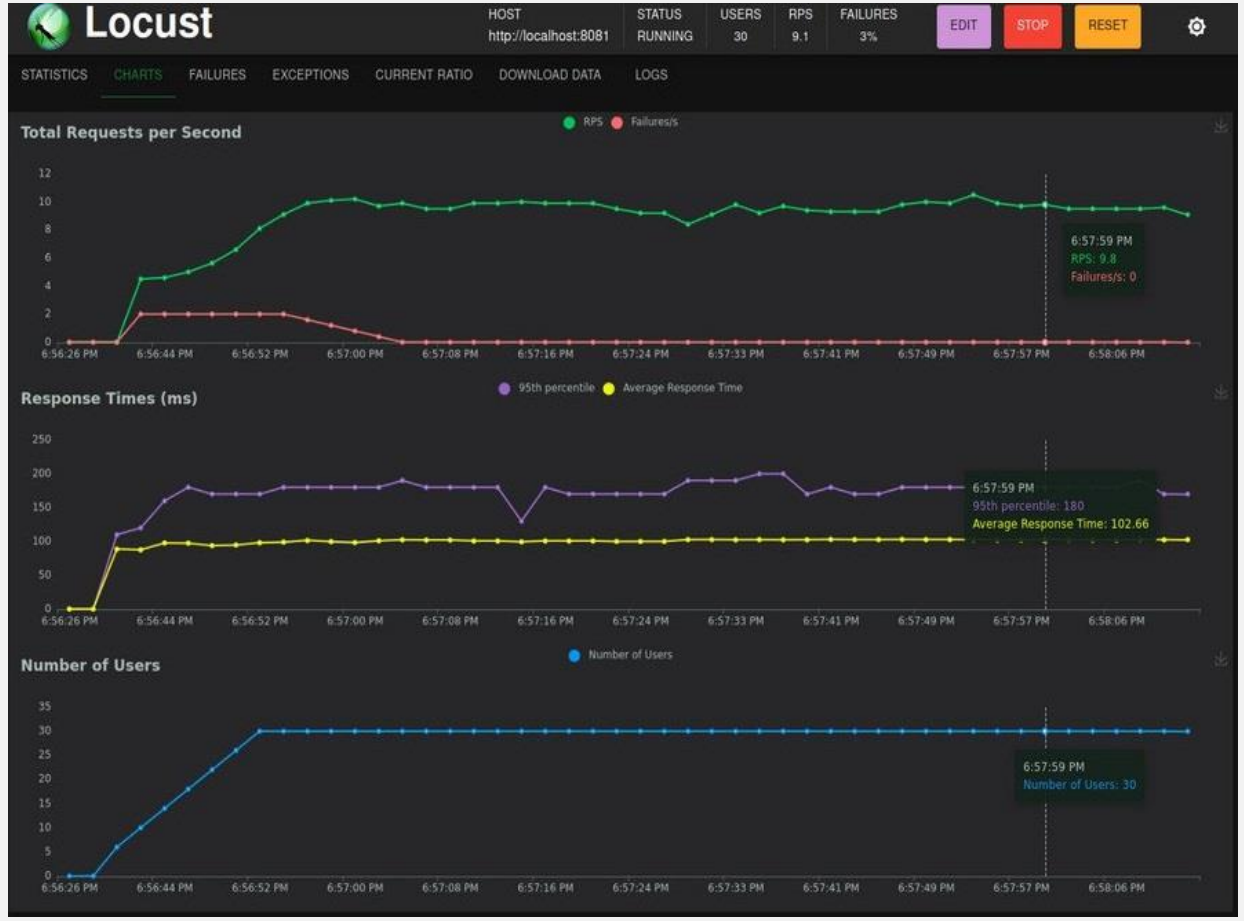
Load Testing: Locust was used to simulate user behavior and test the scalability and performance of my Nextcloud deployment. I defined user tasks in a Python script to mimic file download operations from Nextcloud

In order to see how my system behaves under heavy load, which is crucial for ensuring that my cloud-based file storage system can handle the expected user traffic without performance degradation. First, I defined 30 users with bash and saved them into a file. http://localhost:8089 address defined to use Locust.

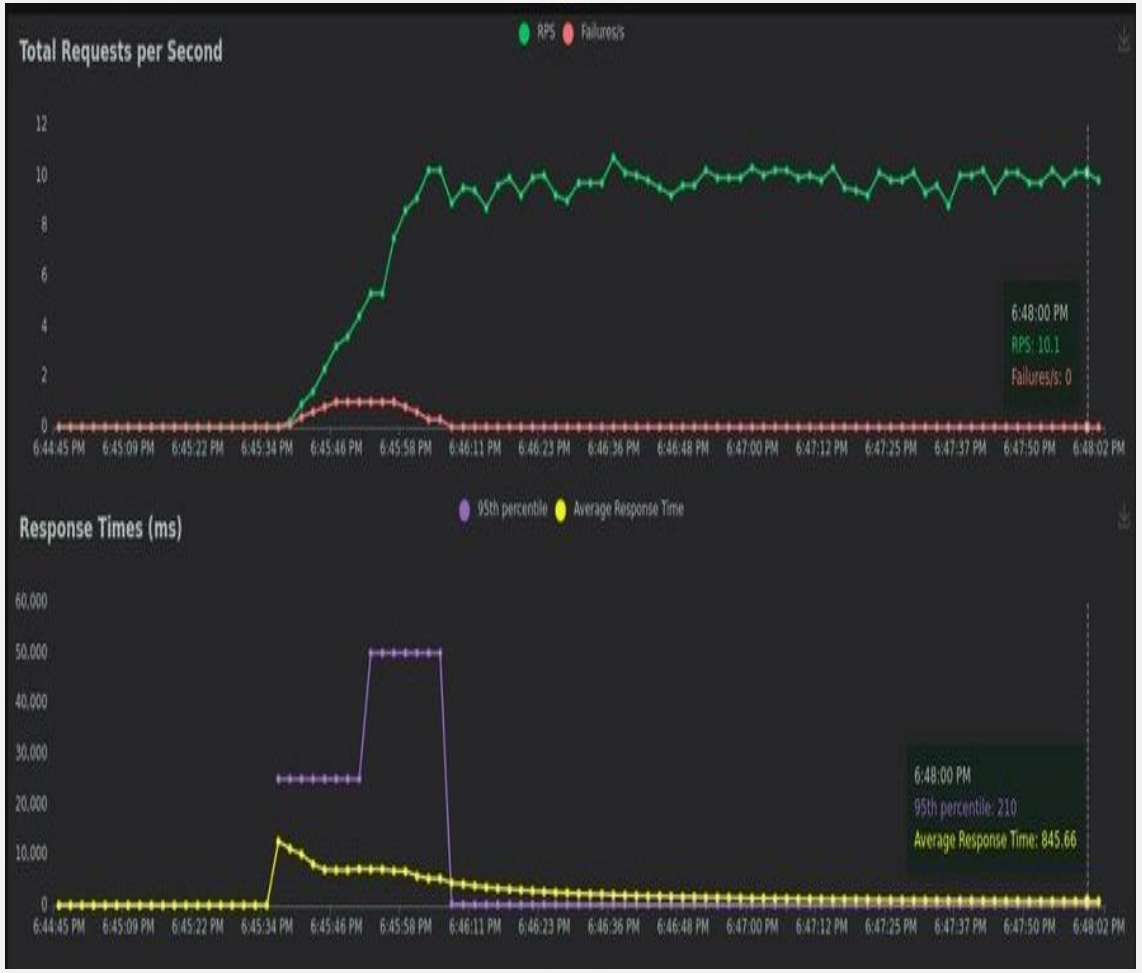
RESULTS

10 KB

1MB



1GB



Increasing number of users

CHALLENGES

- Database connection
- Container identification
- Storage issues
- Nginx configuration issues
- Locust testing challenges

Database Connection Issues: I experienced database connection errors, particularly "Failed to connect to the database" and "Temporary failure in name resolution," indicating issues with the database container's network configuration or the Nextcloud application's database access settings. While I was writing docker compose file I decided to use MySQL 5.7 and successfully run Nextcloud on localhost but when I was checking Security & setup warnings, I noticed that I should use MySQL 8.0 or MariaDB so I changed it but then I started getting errors.

Solution: After getting lots of errors I uninstalled docker-compose and install it again and start the project from scratch to solve the issues.

Container Identification Issues: Errors like "No such container" were encountered when I tried to interact with containers that were either not running or incorrectly referenced in my commands.

Solution: To address the "No such container" errors, I ensured that my Docker containers were up and running using docker-compose up and verified their status with docker ps. I corrected wrong references in my commands.

```
docker ps
```

Storage Issues: I experienced storage issues while I try to run docker compose up after making changes to the docker-compose.

Solution: When I faced challenges related to storage from previous unsuccessful attempts, using docker system prune helped by cleaning up the environment. This action could have resolved issues like: Running out of disk space due to accumulated unused Docker images and containers.

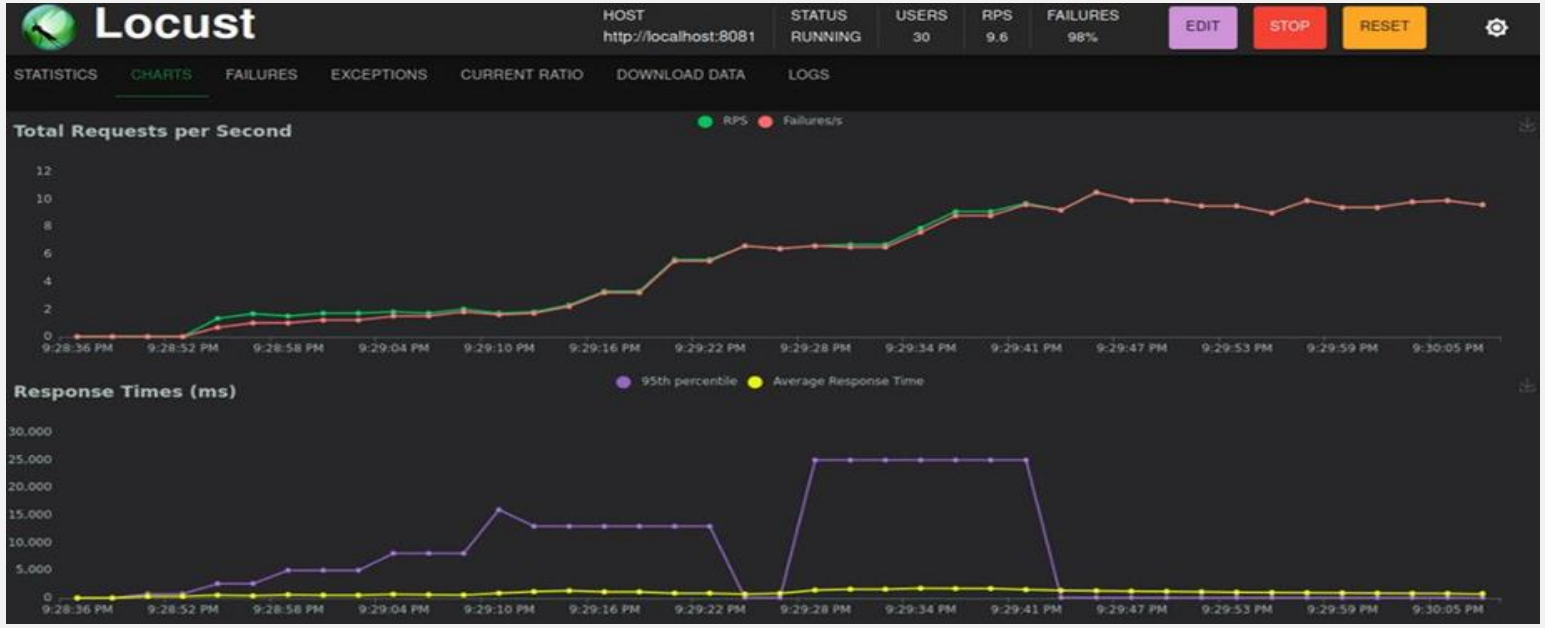
```
docker system prune
```

Nginx configuration issues: I encountered errors related to the upstream directive placement and binding Nginx to port 80 when the port was already in use. Additionally, there were configuration challenges related to setting up load balancing correctly.

Solution: I resolved the Nginx configuration issues by editing the nginx.conf file to correct syntax errors and ensure proper upstream configuration for load balancing. I also made sure that Nginx was not attempting to bind to a port already in use, typically by changing or ensuring the port was available. I changed the ports to 8081 and 8082 to access my Nextcloud instances.

Locust Testing Challenges: During load testing with Locust, I encountered script errors related to user credential unpacking and incorrect file path references. Additionally, there were performance issues with a high failure rate for file downloads.

Solution: I addressed the script errors in Locust by correcting the user credentials file format and ensuring it was properly parsed. I also fixed file path references to ensure the Locust tasks could accurately request the target files. At first, I get errors on unsuccessful login attempts as I didn't create users to test. Then I write a bash file to create users then remember that in order to be able to download a file from Nextcloud first I need to upload some file there. I uploaded the files I created with in admin account and then I gave permission of access to users and fixed the problem. (First, I test only with one user and I get failures all the time, then I understand that I need to make users).



THANK YOU