Cloud-Based File Storage System

ANNALISA PALADINO

Introduction

Task: Exploring and deploying a solution for a cloud-based file storage system

Manage user authentication/authorization and file operations

Discuss scalability, security and cost-efficiency
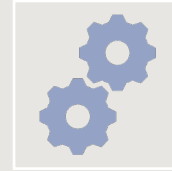
Deployment

Test the infrastructure

Nextcloud

Nextcloud

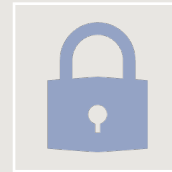|		
Open-source	Extensive documentation	Easy to implement the system
Customizable database backend solutions	Simple dockerized deployment	Production ready security system

User Authentication and File Operations

Nextcloud support by <u>default</u> user authentication, authorization and file operations:

- Users can: *sign up*, *log in* and *log out*

Admin Management:
- Create/modify/delete users
- Assign/change roles and permission

- Users have different roles: ADMIN and REGULAR USER

- Each user is provided of a **private** storage space (1GB)

- Users can upload/delete/download files in their storage space

Address Scalability

Base of a scalable system: ability to accommodate growth in number of users and data volume without compromising on performance or reliability

Goal: ensure the system remains efficient, reliable, and responsive as demand escalates

- Divide the system into a *set of small services*, each service can be scaled independently, allowing for more precise resource allocation based on demand for specific functionalities.

- Implement *stateless* application servers: any server can respond to any request

- *Horizontal scaling*: adding more servers increases capacity without additional complexity

- Cloud-based resources that automatically scale up or down based on real-time (eg. AWS Auto Scaling or Kubernetes' horizontal pod autoscaler)

- *Load balancers* to equally distribute incoming requests across the servers can prevent bottlenecks

- Implementing *caching* at various levels to reduce the load on the backend minimizes the need to compute or retrieve the same information repeatedly

- Continuously *monitor* system performance metrics (CPU usage, memory usage, response times) to identify trends and predict when to scale
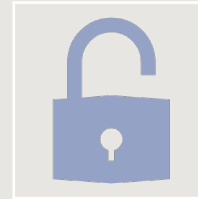
Address Security

To improve Nextcloud default configuration some additional security measures can be applied:
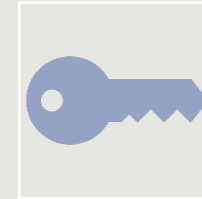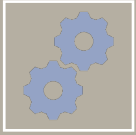
Encryption

Transport Layer Security (TLS)

Multi-Factor Authentication (MFA)
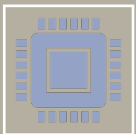
Stronger passwords requirements

Cost-Efficiency

Auto-scaling to dynamically adjust resources based on demand, this ensures you're not paying for inactive resources during low usage periods

Continuous monitoring over resource usage and cost trends

When designing the system it's crucial to have in mind cost-efficiency from the start (choosing the right algorithms, minimizing dependency on external services...)

Deployment

The deployment of Nextcloud has been executed using **Docker** and **Docker Compose**

1. Create a Dockerfile that specifies the environment, dependencies and commands needed to run the service, it puts together the Nextcloud and the MariaDB instances

2. Define a docker-compose.yml file: it describes the application's services, networks and volumes

3. Start the application with `docker-compose up -d`

4. Access to Nextcloud with http://localhost:8080 URL
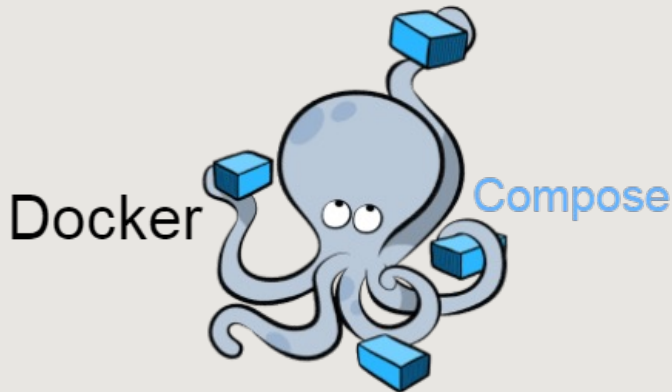
To monitor and manage the deployed system some tools can be used:
- *Portainer* provides a web-based UI for container management, including detailed statistics, log viewing, and container control
- *Prometheus* (metrics collection) and *Grafana* (metrics visualization)

... deploy the system in production

Amazon Web Services (AWS) is a cloud provider that could be used to deploy the system in a production environment because:

- It provides *auto-scaling* capabilities that can automatically adjust the number of compute instances based on the application's load

- It can host applications closer to the end-users, reducing latency and improving user experience

- It relies on high standards of *security*

- It has a *pay-as-you-go* pricing model, which is very useful for optimizing costs
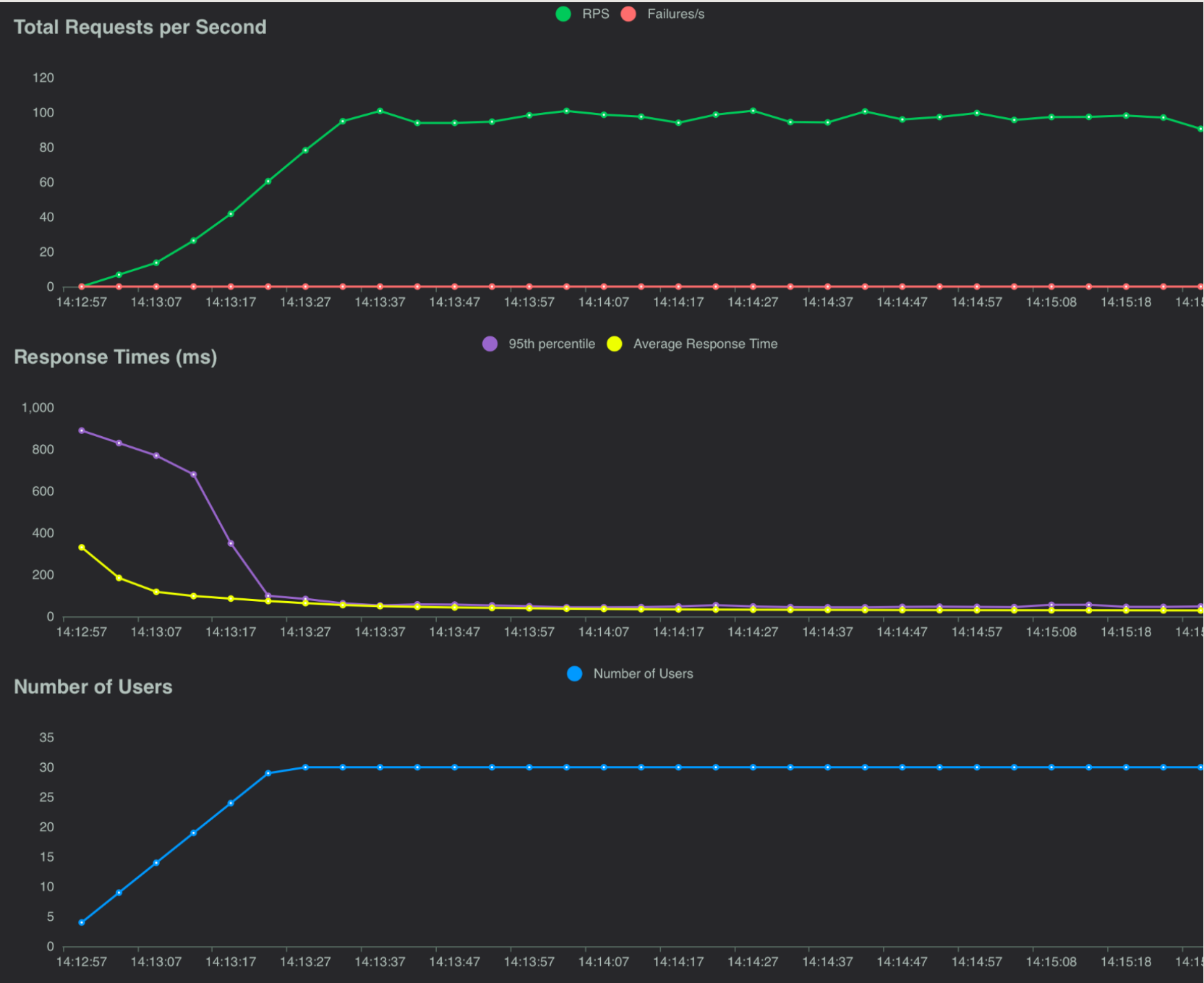
Locust

Open-source Python library and tool for performance testing and load testing of web applications

Defines test scenarios, simulates user behaviour and generates load on a web server to perform stress-test

LOCUST

Small file, 30 users

Big file, 30 users

Thank you for your attention!