Cloud Basic report

Edoardo Zappia

2024

1 Cloud Basic

1.1 Introduction

This project seeks to meet the growing need for an efficient, secure, and scalable cloud-based file storage system that facilitates seamless file upload, download, and management while ensuring the privacy of individual storage spaces. To achieve these objectives without redundant efforts, Nextcloud has been selected due to its comprehensive features and the ease of deployment provided by Docker containers. The primary focus is on implementing a solution that prioritizes data privacy and security, and subsequently, evaluating the system's scalability and cost-efficiency.

1.2 NextCloud

Nextcloud is an open-source software suite that enables the storage of files, photos, videos, and other data on a private server or in the cloud. Beyond basic file storage, it offers a robust array of features, including server-side encryption, collaboration tools, and extensive customization options, making it a comprehensive platform for data management and collaboration.

The choice of Nextcloud was also influenced by its compatibility with Docker, which simplifies both deployment and scaling processes. Its strong emphasis on privacy and security aligns perfectly with our project's objectives, offering granular control over data access and user authentication. Additionally, Nextcloud's flexibility in integrating with existing infrastructure and third-party applications enhances functionality and improves the overall user experience.

1.3 Deployment

The deployment of our Nextcloud-based file storage system leverages Docker and docker-compose, facilitating a straightforward and efficient setup process on a laptop. This section details the deployment plan, utilizing the provided 'docker-compose.yml' file to establish a local containerized environment.

- **Preparation**
  Ensure Docker and docker-compose are installed and operational on the

laptop. Place the ‘docker-compose.yml‘ file in a designated project directory.

- **Configuration**
  The ‘docker-compose.yml‘ file defines two primary services: ‘db‘ (MariaDB database) and ‘app‘ (Nextcloud application). It also specifies volumes for persistent storage and a dedicated network for inter-service communication.

- **Deployment Execution**
  Open a terminal, navigate to the project directory, and run ‘docker-compose up -d‘. This command pulls the necessary images, creates volumes for data persistence, sets up the network, and starts the services in detached mode. Once the containers are running, Nextcloud can be accessed via ‘http://localhost:8080‘, where you can complete the setup through the web interface.

1.4 User Features

1.4.1 User Authentication and Authorization

User authentication and authorization are crucial to maintaining the security and functionality of our Nextcloud deployment while ensuring a seamless user experience. In Nextcloud, these aspects are managed through its web interface, providing administrators with a comprehensive set of tools to control access and user roles effectively. Here's how Nextcloud facilitates these critical functionalities:

- **User Sign-Up, Log-In, and Log-Out**
  Nextcloud allows users to create accounts, sign in to access their files, and log out when finished. The process is streamlined for ease of use, ensuring that even users with minimal technical knowledge can navigate these steps effortlessly. While public user registration is not enabled by default, plugins such as the Registration app can be installed to allow users to sign up independently, with email verification for added security.

- **Role-Based Access Control**
  Nextcloud categorizes users primarily into two groups: regular users and administrators. Each category has distinct permissions and access levels within the platform. Administrators can assign and modify these roles through the Users section of the Nextcloud admin interface.

- **Private Storage for Regular Users**
  Upon account creation, regular users are allocated private storage space for securely storing their files. This space is isolated from other users, ensuring privacy and data protection. Administrators can adjust the storage quota for each user, managing the system's capacity and ensuring fair use across all accounts.

- **Administrative User Management**
  Nextcloud administrators are responsible for managing user accounts, including creating new accounts, modifying existing ones, resetting passwords, and deleting users when necessary. Additionally, administrators can manage common settings among users by creating and maintaining user groups, ensuring consistent and efficient administration.

1.4.2 Security Measures

Nextcloud offers a variety of features and tools that administrators can implement to protect data and ensure a secure user environment. Many of these security measures can be easily managed and installed directly from Nextcloud's web interface:

- **Two-Factor Authentication (2FA)**
  To enhance the security of user accounts, Nextcloud supports two-factor authentication. Administrators can enable 2FA from the security settings in the web interface, and users can choose from several second-factor methods, including TOTP (Time-based One-Time Password) apps, SMS, or hardware tokens. The Two-Factor Admin Support application enables administrators to generate a one-time code for users to access a 2FA-protected account. This feature is particularly useful in scenarios where users have lost access to their existing 2FA methods or when 2FA is mandatory but no prior 2FA provider has been enabled.

- **File Access Control**
  The File Access Control app in Nextcloud allows administrators to create rules that restrict file access based on factors such as user groups, file type, or size. This helps prevent the sharing of sensitive data and ensures that users only have access to the files they are authorized to view or edit.

- **Brute Force Protection**
  Nextcloud includes a brute force protection feature that detects and slows down repeated failed login attempts, making it more difficult for attackers to gain unauthorized access through password guessing.

- **Server-Side Encryption**
  Nextcloud allows administrators to enable server-side encryption via its web interface, ensuring that data stored on the server is encrypted and protected from unauthorized access. This will reduce the performance of the system but will prevent an intruder that gains access to the data to read it. When file encryption is enabled, all files can still be shared by a user using the Nextcloud interface but won't be sharable directly from the remote server. Note that enabling encryption also increases the amount of storage space required by each file.

- **Password Policy**
  The system will not be secure unless users have robust, non-trivial pass-

words. From the configuration page, administrators can enhance the password policy by requiring the inclusion of numbers and symbols. Additionally, we can mandate that passwords be changed periodically.

1.5 Locust Testing

To evaluate the performance and scalability of our Nextcloud deployment under realistic user load conditions, we utilize Locust, a powerful open-source load testing tool. Our Locust test script is designed to simulate various user actions that mimic typical interactions with the Nextcloud platform. Specifically, the script executes a 'PROPFIND' request to list files, a 'PUT' request to upload files of different sizes (1KB, 1MB, and 1GB), and a 'GET' request to read a shared Readme.md file. Each task uses HTTPBasicAuth for authentication, ensuring that each request is performed as a unique user with proper credentials.

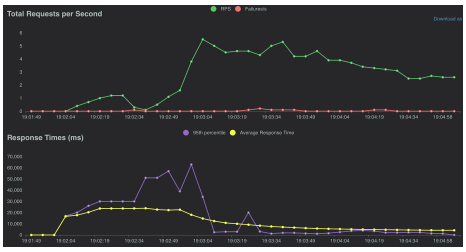
To simulate user actions on Nextcloud, 70 users were created using a bash script (‘create_users.sh‘ in the 'scripts' directory), which automatically generates 70 users with 3GB of private storage each. Before performing each Locust test, we ensure that all users' private storage is empty using another bash script (‘clear_user_storage.sh‘ in the 'scripts' directory).

1.5.1 Results

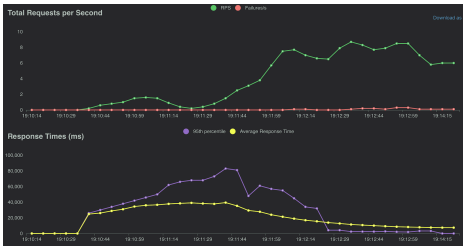
We begin testing with a modest number of users to establish baseline performance metrics under normal operating conditions. This initial testing phase provides insight into the system's default handling of user requests. To stress-test the system, we simulate a high number of concurrent users performing tasks as defined in the script. By examining the Locust test reports for simulations with 10, 30 and 50 users, we identify patterns that highlight the scalability and performance characteristics of the Nextcloud deployment. As the number of simulated users increases, so does the strain on the system, which is reflected in various metrics including response times, failure rates, and system throughput.

- **10-user test**
  The system under test demonstrated commendable performance and stability with 10 users. Average response times were consistently low, and the system efficiently handled requests with minimal errors.

- **30-user test**
  Increasing the user count to 30 resulted in a slight increase in response times, particularly for resource-intensive operations such as uploading 1GB files. While the system continued to maintain a low failure rate, the increased load began to expose limitations in handling concurrent large file uploads.



- **50-user test**
  With 50 users, the strain on the system became more pronounced. The average response times for operations, particularly file uploads, showed a significant increase. The occurrence of 'RemoteDisconnected' errors suggests that the server was overwhelmed, likely due to limitations in handling concurrent HTTP connections or issues with the web server configuration.



1.6 Scalability

Taking into account the performance results from the Locust tests, we outline suitable solutions to optimize scalability for the Nextcloud deployment. Given the increase in response times and failure rates observed with higher user loads, particularly when dealing with file uploads and downloads, the following solutions are proposed:

- **Caching Layers**
  Introducing a caching mechanism can reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally.

- **Horizontal Scaling**
  To handle an increasing load and traffic, we can create additional instances

of Nextcloud and distribute the load among them, adopting a horizontal scaling approach.

- **Object Storage Service**
  Many cloud providers offer auto-scaling solutions that dynamically adjust computing resources based on the current workload. Utilizing such object storage services can enhance scalability and efficiency.

1.6.1 On-Premise Cluster Deployment

If an organization already owns a cluster of nodes or prefers an on-premise solution, deploying Nextcloud on multiple nodes with a shared production-ready database can be advantageous. In this setup, each node handles user requests to balance the overall load, while data is stored on a shared NFS file system. The database, such as PostgreSQL or MariaDB, must be robust and capable of managing concurrent connections from multiple nodes. Additionally, implementing a redundant replication and backup system is essential to ensure data integrity and availability in case of node failures.

Advantages

- **Control and Ownership**
  The organization maintains full control over the hardware and infrastructure.
- **Customization**
  On-premise solutions allow for the customization of hardware configurations, optimizing the system for specific performance requirements.
- **Predictable Costs**
  While there may be an initial investment in hardware, an organization can benefit from more predictable and potentially lower operational costs over time.
- **Security Policies**
  The organization can implement and enforce its own security policies without relying on third-party providers.
- **Data Residency**
  On-premise deployment ensures that data remains within the organization's physical or virtual infrastructure, which is crucial for sensitive data.

Cost Considerations

- **Infrastructure Costs**
  The organization bears the cost of owning and maintaining the physical hardware or virtual machines for the cluster nodes. Periodic hardware upgrades may be necessary as technology becomes outdated or more prone to failures.

- **Backup System Costs**
  Implementing a reliable backup and data replication system incurs additional expenses.

1.6.2 Cloud Providers

Another option is to utilize a cloud provider. Utilizing a cloud provider offers the advantage of scalability and flexibility. In this section of the report, we focus on evaluating two prominent cloud providers, Amazon S3 and Google Cloud Storage (GCS), for hosting our Nextcloud deployment. By comparing these platforms, we aim to determine the most suitable option based on a detailed assessment of their features, cost structures, and overall compatibility with our requirements.

Cost-Efficiency

- **Amazon S3:** Amazon S3 offers a detailed pricing model that includes costs for storage per GB, requests, and data transfer. It features various storage classes such as S3 Standard, S3 Intelligent-Tiering, S3 One Zone-IA, and Glacier, each designed for different use cases and cost savings based on access frequency and data durability requirements.
- **Google Cloud Storage:** GCS also offers competitive pricing with a simple structure based on data storage, network usage, and operations. It includes storage classes like Standard, Nearline, Coldline, and Archive, each priced according to data access patterns and storage duration.

Scalability

- **Amazon S3:** Designed to offer great scalability and handle vast amounts of data and traffic, S3 supports extensive data operations with strong consistency and built-in redundancy.
- **Google Cloud Storage:** GCS benefits from Google's global infrastructure, which is optimized for high performance and rapid data access. Google's private network provides fast data transfer rates, crucial for applications requiring quick access to large amounts of data.

Security

- **Amazon S3:** Provides detailed access controls, bucket policies, and integrates with AWS Identity and Access Management (IAM) for fine-grained security management.
- **Google Cloud Storage:** Offers robust security measures, similar encryption capabilities, and easy integration with Google Cloud's Identity and Access Management for managing permissions.

1.7 Cost Optimization

Comparing the two solutions, an on-premise deployment may require higher upfront infrastructure costs but could lead to lower operational expenses over time. On the other hand, deploying with a cloud provider offers flexibility and scalability, though costs can increase based on usage. The optimal choice depends on factors such as hardware availability and workload variability.

To optimize costs for the cloud provider solution, assuming the stored data isn't sensitive or the cloud service provides robust security measures, consider disabling server-side encryption in Nextcloud. This reduces file sizes and improves performance. Similarly, in on-premise solution, if there are constraints on computing power or storage capacity, disabling encryption could also be beneficial.

1.8 Conclusion

In conclusion, this project explored the implementation of a cloud-based file storage system using Nextcloud, chosen for its comprehensive feature set that closely matches the project requirements. Utilizing Nextcloud's intuitive interface, scalable architecture, and Docker container deployment greatly streamlined both development and deployment processes.

The selected features of Nextcloud, including robust user authentication, role-based access control, private storage allocation, and comprehensive admin management capabilities, establish a robust foundation for constructing a secure and user-centric file storage platform. By integrating security measures such as server-side file encryption and multi-factor authentication, sensitive user data is effectively safeguarded.

Nextcloud's versatility in supporting different database backends and storage solutions facilitates adaptation to diverse deployment scenarios. This report explores two scalable deployment options: on-premise deployment utilizing a shared cluster and cloud provider deployment with autoscaling capabilities. These solutions offer flexible production environments tailored to meet varied operational needs.

This project underscores the importance of intelligent decision-making in balancing features, security, scalability, and cost-effectiveness to achieve a comprehensive and efficient solution.

2 Cloud Advanced

2.1 Introduction

The assignment requires to redeploy the cloud-based file storage system using the Kubernetes container orchestration platform alongside the Helm package manager. Specifically, the system needs to be deployed on a single-node Kubernetes cluster, which must be created and properly configured.

2.2 Deployment

The deployment of the Nextcloud instance is based on the use of Helm. Specifically, the official Nextcloud Helm chart is employed, with appropriate custom values configured in the 'values.yaml' file. The chart deploys several Kubernetes components:

- **Nextcloud Pod**: This pod contains the Nextcloud application. The Nextcloud pod is configured to connect to a backend PostgreSQL database and Redis for caching. Both the database and Redis are deployed in their own pods.
- **PostgreSQL Pod**: A separate PostgreSQL pod is deployed for managing the database, and it is linked to a Persistent Volume Claim (PVC) to ensure data persistence even if the pod is restarted or deleted.
- **Redis Pod**: Redis is deployed in a separate pod for caching purposes and to manage session data, enhancing the performance of the Nextcloud instance.
- **Persistent Volumes (PV) and Persistent Volume Claims (PVC)**: The Nextcloud and PostgreSQL pods are both linked to PVCs, ensuring that their data is persisted across pod restarts or failures.
- **MetalLB Load Balancer**: The deployment uses MetalLB to manage external IP addresses and expose the Nextcloud service to the outside world. This allows the Nextcloud instance to be accessible via an external IP address.
- **Ingress Controller**: The ingress controller routes incoming traffic to the Nextcloud service. It is configured with an external IP address assigned by MetalLB, allowing users to access the service via a domain name or IP.
- **Kubernetes Secrets**: Secrets are used to securely manage sensitive information such as database passwords and Nextcloud admin credentials.

2.3 Back-end Storage

To ensure data persistence in the event of pod crashes or accidental deletions, Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) were utilized to provide storage for both the PostgreSQL database and Nextcloud pods. This setup guarantees that data remains intact even when individual pods are terminated or recreated. In this deployment, a local path provisioner was used to map a directory on the host machine to a persistent volume in the Kubernetes cluster. While effective for basic data persistence, this back-end storage method has several limitations:

- **Single Point of Failure**: Since the storage is bound to the host machine, any failure of the host will result in the loss of all data stored locally, as the storage is not replicated across multiple nodes.

- **Manual Volume Creation**: The local path provisioner does not support dynamic volume provisioning, meaning that Persistent Volumes (PVs) must be manually defined before the service can use them.
- **Node Affinity Restrictions**: Since PVs are tied to a specific node, the corresponding pods must always be scheduled on the same node, limiting the ability to distribute workloads across multiple nodes.
- **Limited Scalability**: The storage capacity is constrained by the physical storage available on the host machine, making it difficult to scale the service as demand increases.

These limitations make local path storage suitable for development, testing, or small-scale production environments where data persistence is important, but high availability and scalability are not critical requirements. For larger-scale, production-grade environments that demand higher resilience and scalability, distributed storage solutions like Ceph or GlusterFS would be more suitable.

2.4 High Availability Considerations

To achieve true high availability (HA), several improvements could be made to the current deployment:

- **Database Replication**: Configuring PostgreSQL with replication, either in an active-active or active-passive setup, would ensure that the database remains available even if one instance fails.
- **Redis Replication**: Implementing Redis replication and enabling Redis Sentinel for automatic failover would prevent Redis from becoming a single point of failure.
- **Multiple Nodes**: While the current setup runs on a single node, adding additional nodes to the Kubernetes cluster would distribute the workload and provide resilience in case one node fails.
- **Persistent Volumes in a Distributed Storage System**: Using a distributed storage system like Ceph or GlusterFS for persistent volumes would allow for data replication across multiple nodes, ensuring that storage is not a single point of failure.

By implementing these enhancements, the system would be more resilient to failures and better suited for production environments with strict uptime requirements.

2.5 Comparison with the Docker Solution

In the Docker Compose deployment, services are deployed as standalone containers, each running independently and directly managed by Docker. In contrast, the Kubernetes-based solution deploys these services within pods, where

containers are managed by a powerful container orchestration platform. The Kubernetes deployment offers a more scalable, flexible, and robust environment. It enables seamless scaling, either by adding more nodes to the cluster or increasing the number of replicas. Additionally, Kubernetes automates critical tasks such as scaling, load balancing, and self-healing, ensuring the service is managed more efficiently.

As discussed earlier, Kubernetes can be configured for high availability, making it ideal for large-scale production environments. However, Kubernetes requires a more complex setup and higher resource allocation. For instance, while Docker Compose has no strict minimum requirements, machines running Kubernetes typically need at least 2 CPUs and 2 GB of RAM.

Despite its complexity, Kubernetes is well-suited for large deployments requiring advanced orchestration features. On the other hand, Docker Compose remains a valid option for development, testing, and small-scale production environments due to its simplicity in setup and management.