PG3402 - Project Description

A generic streaming service using microservices

Joar Heimonen
contact@joar.me

October 14, 2024

Contents

1 Introdcution

This document will describe the implementation of a simple scalable streaming service built using multiple microservices [3]. We will describe how the system handles user access control, content managment and load balancing. The most important element of this project will be JSON web tokens[2] that allow data to pass between the different microservices. The service will also feature load balancing using the GoodDNS server[1].

2 Overview

In this project, we aim to develop a robust streaming service that leverages a microservices architecture to offer scalable video streaming. The service is designed to cater to two main types of users: administrators and registered viewers. Administrators have the ability to upload and manage content, while registered users can browse and stream content. The core functionalities include user registration and login, content management, and dynamic load balancing to handle varying loads. The system will utilize JSON web

tokens for secure data transmission between microservices utilizing users as intermediates. it incorporate a custom DNS solution, GoodDNS [1], for load management.

3 User Stories

3.1 Story 1: User Registration

As a user,
I want to be able to register an account using a username and a password,
So that I can access the streaming service.

Acceptance Criteria:

1. The registration screen is accessible from the home page.
2. Users can enter a username and password.
3. The system validates that the username is valid.
4. An error message is displayed if the username is not valid.

3.2 Story 2: Content Browser

As a user,
I want to be able to browse content,
So that I can access and stream the content.

Acceptance Criteria:

1. The homepage consists of a list of content.
2. Users can select the content they want to watch.
3. The system checks if the user is logged inn.
4. The user is able to stream content when logged inn.
5. A login prompt is shown if the user is not logged inn.

3.3 Story 3: Add New Content

As an Administrator,
I want to add new contet to the service,
So that users can watch the content.

Acceptance Criteria:

1. There is an "Admin" button on the home page.
2. Clicking the button takes the user to an admin page.
3. The content can be submitted on the adming page.
4. An error message is shown if the content submission fails.

4 Requirements

4.1 Users

The system must be able to handle user registration and login. This includes the ability to securely store user credentials and personal information in a database.

4.2 Content Management

The system must be able to handle content uploading and streaming efficiently. This involves accepting new content from administrators, storing it in the content database and the file system, and serving it to authenticated users upon request. The system should support various media formats and ensure smooth streaming experiences even under high load. Additionally, content metadata management (like titles, descriptions, and tags) should be implemented to enhance content discoverability.

4.3 Load Balancing

The system should efficiently distribute incoming network traffic across multiple servers to ensure reliability and scalability. This involves using a DNS server that can dynamically allocate requests to the least loaded server. The load balancing DNS server must have a second failover DNS server. If neceserry an infinite ammount of sub-sub DNS servers can be implemented. The

time to live is used to control how often users poll the DNS servers for new server allocations.

ttl	**ttl**
0	3600

- `userserver1.dns1.mydomain.com`
- `userserver1.dns2.mydomain.com`
- `contentserver2.dns2.mydomain.com`

5 Architecture

Figure 1 shows a service consisting of the following microservices:

- **DNS Server (Load Balancer)**: Directs incoming network traffic to different servers, balancing the load and serving as a DNS resolver. All servers recieve their own sub-domains with short times to live.
- **Clients**: End-users who interact with the application through a web browser.
- **User Management Service**: Manages user authentication, authorization, and profiles, interfacing with the **User Database**.
- **Content Management Service**: Handles content storage and retrieval through the **Content Database** and manages file-based content with the **File System**.
- **Web Server**: Serves a static single page application used by the client to interact with the microservices.
- **Databases and File System**: Separate databases for user and content data ensure efficient management and scalability. The file system stores non-database content.

Communication between clients and services is secured using HTTPS, with REST API requests facilitating interactions between clients and microservices. This architecture ensures security, and scalability under varying load.

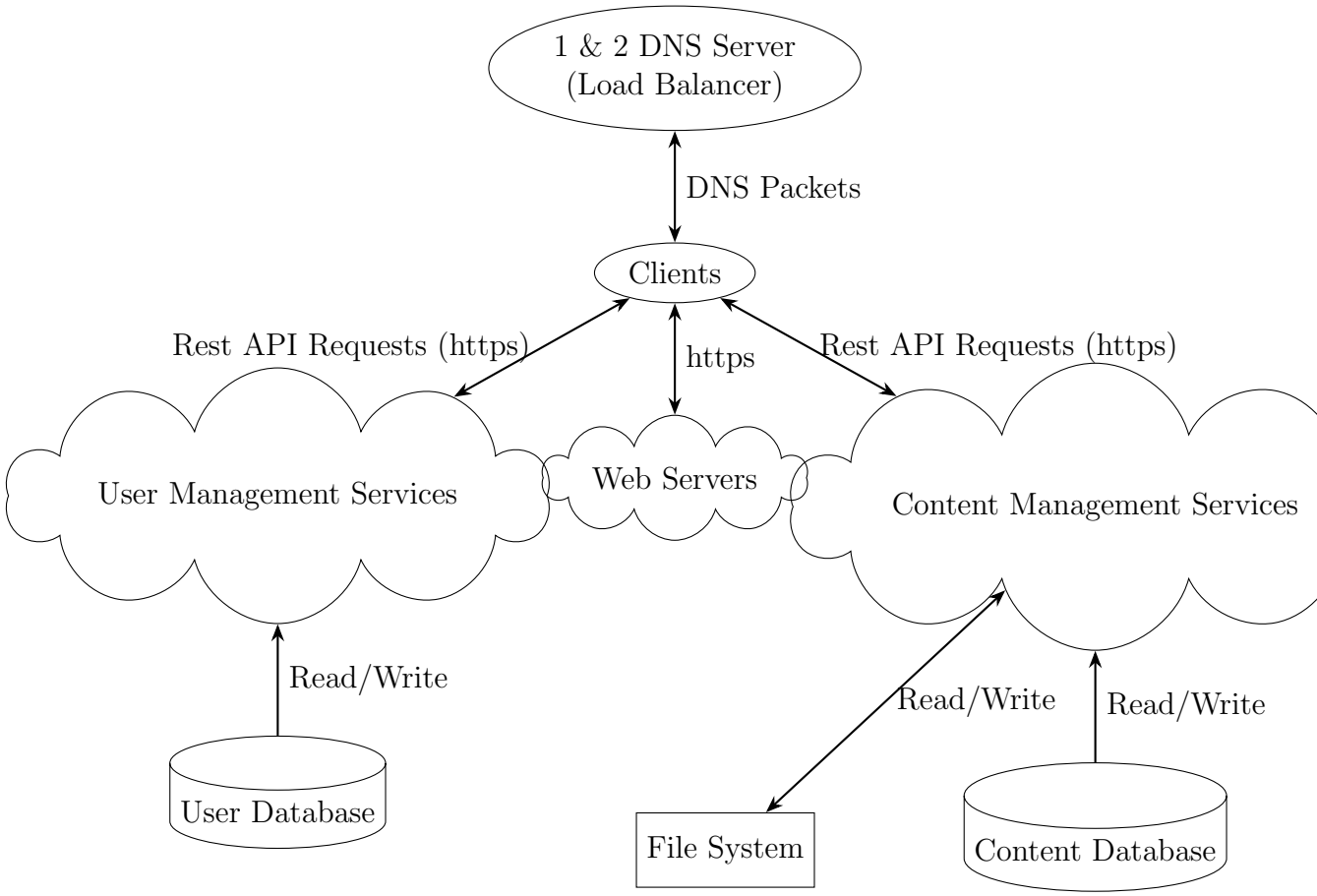

Figure 1: A graph of the service architecture. The clouds represents clusters of multiple microservices.

6 Notes

The architecture does not utilize a central API gateway, Instead it relies on the clients sending requests to the relevant services. This is made possible by the GoodDNS [1] load balancer. This architecture was selected as a central API gateway requires proxying all traffic.

References

[1] Joar Heimonen. *Slenderman00/GoodDns*. Dec. 2023. (Visited on 10/13/2024).

[2] Michael B. Jones, John Bradley, and Nat Sakimura. *JSON Web Token (JWT)*. Request for Comments RFC 7519. Internet Engineering Task Force, May 2015. DOI: 10.17487/RFC7519. (Visited on 10/14/2024).

[3] "Microservices". In: *Wikipedia* (Oct. 2024). (Visited on 10/14/2024).