NETFLIX ARCHITECTURE

PREPARED BY

Syed Ghazi Raza
FA22-BSE-082
M. Aalyan Mughal
FA22-BSE-094
Hassan Ali Mashwani
FA22-BSE-067

TO
(Sir) Mukhtiar Zamin

November 28, 2024

•

Report:

Netflix Architecture

Table of Contents

1. **Introduction**
2. **Evolution of Netflix's Architecture**
    - **Early Architecture (Monolithic)**
    - **Transition to Cloud**
    - **Current Architecture (Microservices)**
3. **Major Features and Release Highlights**
4. **Architectural Diagrams**
    - **Monolithic Architecture**
    - **Microservices Architecture**
    - **Open Connect Architecture**
5. **Frameworks and Technologies Used**
6. **Object Relationships**
7. **Implementation of Software Design Principles**
    - **GRASP Principles**
    - **SOLID Principles**
    - **GoF Patterns**
8. **Conclusion**
9. **References**
10. **Contributions by Group Members**

1. Introduction

- Netflix began as a DVD rental service and transitioned into a global leader in video streaming.
- This report analyzes the architectural evolution of Netflix, highlighting its design principles and technical choices.

2. Evolution of Netflix's Architecture

- **Early Architecture:**
  - Monolithic system hosted on on-premises servers.
  - Challenges: Scalability limitations and maintenance complexity**.**
- **Transition to Cloud:**
  - Migration to AWS in 2011 enabled scalability, reliability, and global reach.
  - Adopted service-oriented architecture (SOA) for better modularity.
- **Current Architecture:**
  - Fully microservices-based architecture with hundreds of loosely coupled services.
  - Use of Open Connect CDN to optimize video delivery and reduce latency.

3. Major Features and Release Highlights

- 2007: Launch of online streaming alongside DVD rentals. (Source)
- 2011: Migration to AWS and adoption of microservices. (AWS Case Study)
- 2016: Expansion to 190 countries. (Global Expansion)
- 2020: Introduction of AV1 codec for video compression. (Tech Blog)

4. Architectural Diagrams

- **Diagram 1: Monolithic Architecture (Pre-Cloud).**
  - Description: Centralized system with single points of failure.

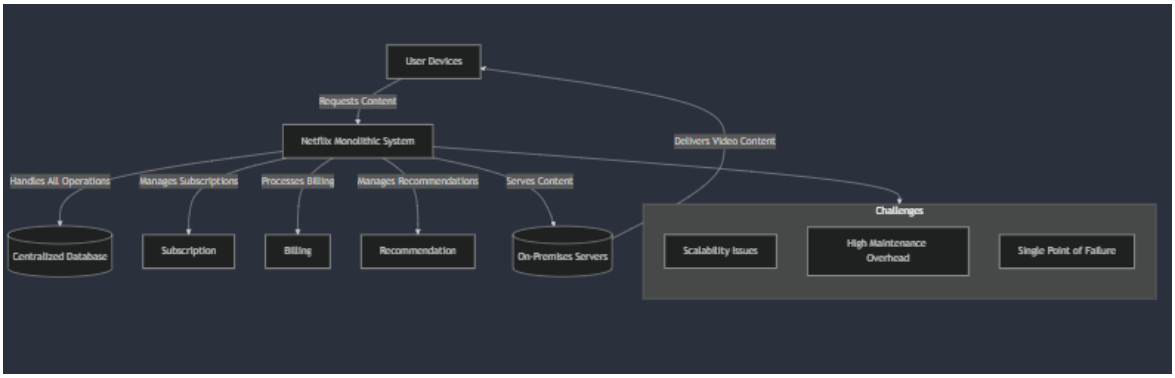


Diagram Link

https://shorturl.at/tOCtI

- **Diagram 2: Microservices Architecture (Post-Cloud).**
    - Description: Independent services managing specific functionalities (e.g., recommendations, billing).

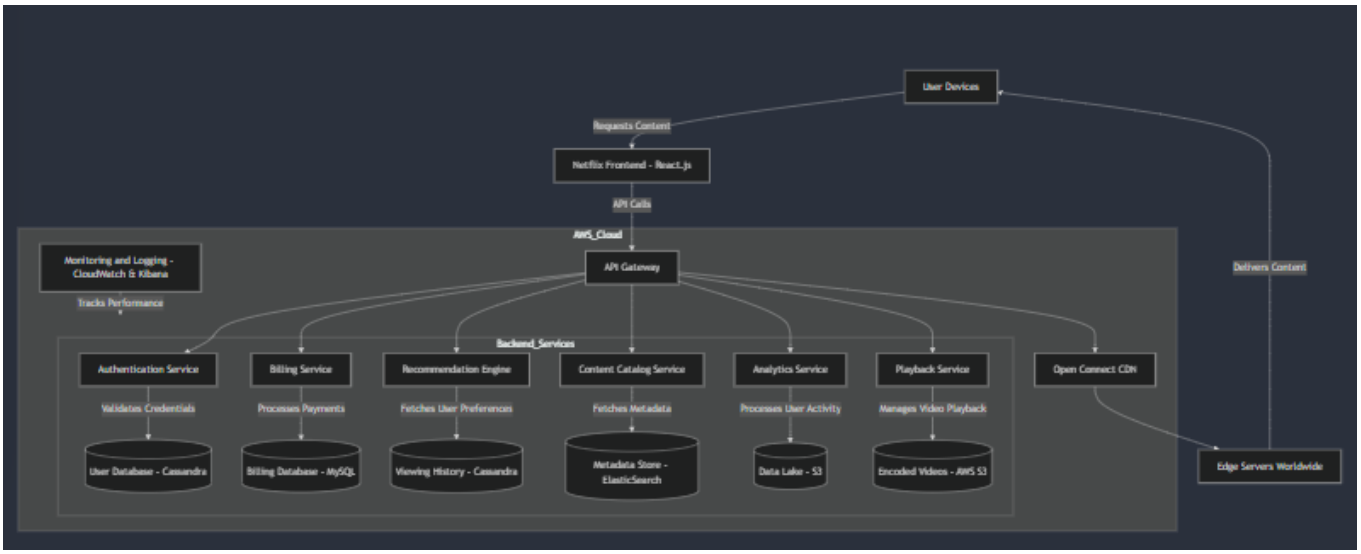


Diagram Link:

https://shorturl.at/LAM2z

- **Diagram 3: Open Connect Architecture.**
    - Description: Netflix's custom CDN with edge servers deployed globally.

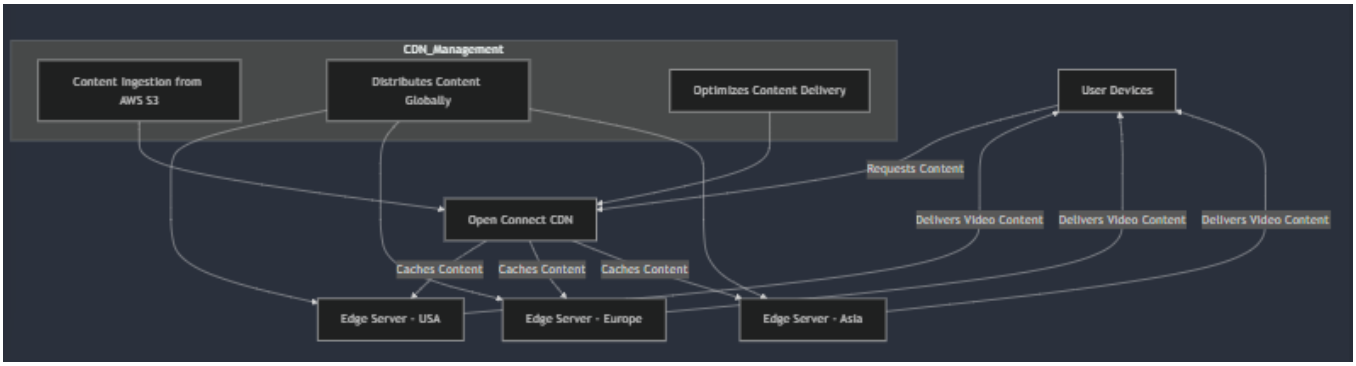


Diagram Link:

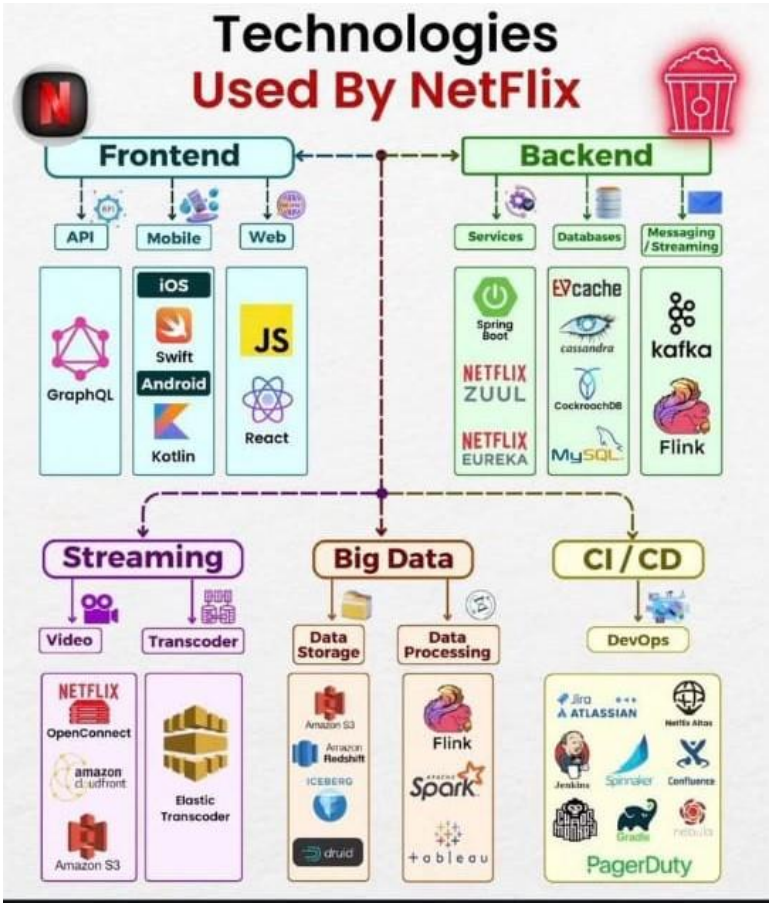
https://shorturl.at/vbOmY

Overall Diagram link:

https://shorturl.at/Ozd3d

5. Frameworks and Technologies Used

- Cloud Platform: Amazon Web Services (AWS) for global scalability.
- Backend Framework: Spring Boot for microservices.
- Database: Cassandra and MySQL for distributed data storage.
- Messaging System: Apache Kafka for real-time event streaming.
- Frontend Framework: React.js for the user interface.
- Streaming Protocol: DASH (Dynamic Adaptive Streaming over HTTP).
- Video Encoding: AV1 codec for efficient compression.



6. Object Relationships

- Dependency: Microservices like the recommendation engine rely on analytics services.
- Association: User profiles link to subscription services.
- Aggregation/Composition: Various microservices combine to form Netflix's functionality.
- Inheritance: Shared base functionality among services.
- Realization: Interfaces ensure consistent service behavior**.**

7. Implementation of Software Design Principles

- **GRASP Principles:**
    - Controller: Microservices coordinate workflows.
    - Information Expert: Recommendation engine encapsulates domain knowledge.
- **SOLID Principles:**
    - Single Responsibility: Services have narrowly focused responsibilities.
    - Open-Closed: New features added without altering existing code.
- **GoF Patterns:**
    - Singleton: Used for shared configurations.
    - Factory: Dynamically creates service instances.

8. Conclusion

- Netflix's architectural evolution demonstrates the importance of adopting modern, scalable technologies and principles.
- It highlights the need for adaptability and resilience in large-scale systems.

9. References

- **Official Netflix Tech Blog:** Netflix Tech Blog
- **AWS Case Study on Netflix:** AWS Case Study
- **Netflix's Global Expansion:** Global Expansion
- **Research papers on Netflix's architecture.**
    - **Here are the research papers and articles on Netflix's architecture**
    - "Netflix System Design and Software Architecture"
    - "Netflix and Its System Architecture"
    - "Analysis of Netflix Architecture and Business Model"
    - "System Design: Netflix – A Complete Architecture"

- "Microservices Architecture Using Netflix Tech Stack – Conceptual View"
- "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery"
- "Netflix's System Architecture Represents A Rare Competitive Edge"
- "Open Connect Everywhere: A Glimpse at the Internet Ecosystem through the Lens of the Netflix CDN"
- "Using the Buffer to Avoid Rebuffers: Evidence from a Large Video Streaming Service"
- "Engineering for a Science-Centric Experimentation Platform"
- "Learning to Predict Streaming Video QoE: Distortions, Rebuffering, and Memory"
- **These resources should provide valuable insights into Netflix's system and architecture**

- **Articles from authoritative sources (e.g., Medium, GitHub).**
  - **Here are the articles from authoritative sources with in-line links:**
  - "Cloud Architecture" – Netflix TechBlog
  - "System Design: Netflix – A Complete Architecture" – GeeksforGeeks
  - "Netflix System Design- Backend Architecture" – DEV Community
  - "Let's Decode Netflix System Design and Backend Architecture" – TechAhead
  - "10 Things You Can Learn from Netflix's Architecture" – DEV Community

10. Contributions by Group Members

1. **Syed Ghazi Raza :**
   - **Introduction**
   - **Evolution of Netflix's Architecture**
   - **Major Features and Release Highlights**
2. **Hassan Ali Mashwani :**
   - Architectural Diagrams
   - Frameworks and Technologies Used
   - Object Relationships
3. **M. Aalyan Mughal :**
   - Implementation of Software Design Principles
   - Conclusion
   - References
