E-commerce site to be used in the Project: https://www.netflix.com/tr/

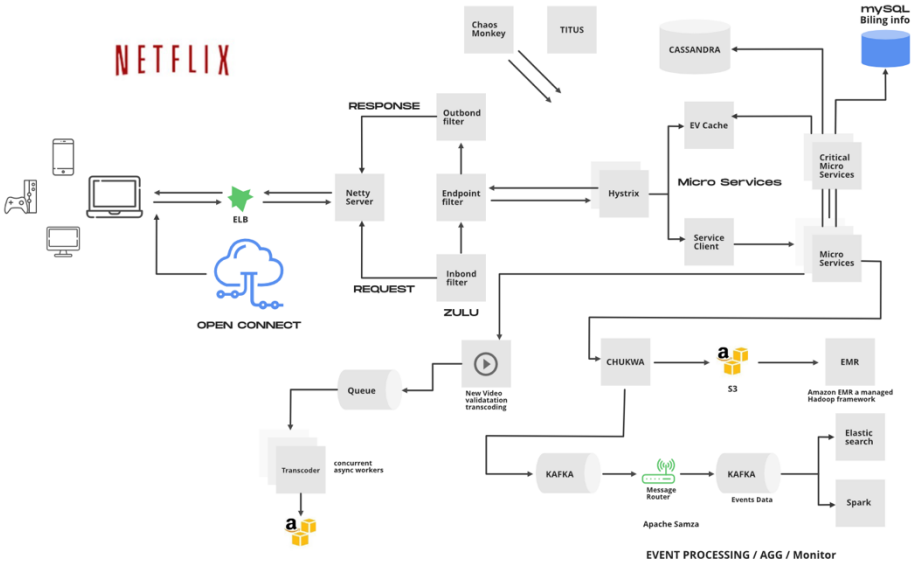
World's largest video platform : Netflix

The first of these is the Fee-For-Transaction Revenue Model. This is because Businesses using the Fee-For Transaction Type Revenue provide services and charge according to the number or size of transactions processed.

The second e-commerce income that Netflix has Web Catalog Revenue Model. This is because, Putting catalog information (or additional information) on the Web is to take the catalog model to the Web.

Netflix High-Level System Architecture

We all are familiar with Netflix services. It handles large categories of movies and television content and users pay the monthly rent to access these contents. Netflix has **180M+** subscribers in **200+** countries.

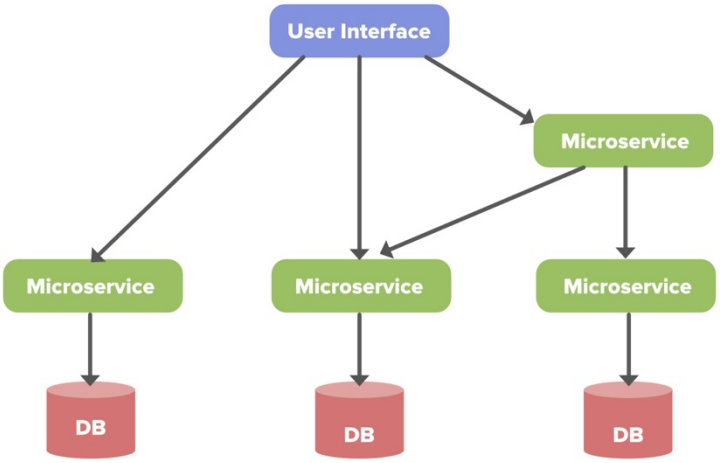


Netflix works on two clouds **AWS** and **Open Connect**. These two clouds work together as the backbone of Netflix and both are highly responsible for providing the best video to the subscribers.
The application has mainly 3 components…

- **Client:** Device (User Interface) which is used to browse and play Netflix videos. TV, XBOX, laptop or mobile phone, etc
- **OC (Open connect) or Netflix CDN:** CDN is the network of distributed servers in different geographical locations, and Open Connect is Netflix's own custom global CDN (Content delivery network). It handles everything which involves video streaming. It is distributed in different locations and once you hit the play button the video stream from this component is displayed on your device. So if you're trying to play the video sitting in North America, the video will be served from the nearest open connect (or server) instead of the original server (faster response from the nearest server).
- **Backend (Database)**: This part handles everything that doesn't involve video streaming (before you hit the play button) such as onboarding new content, processing videos, distributing them to servers located in different parts of the world, and managing the network traffic. Most of the processes are taken care of by Amazon Web Services.

Netflix frontend is written in **ReactJS** for mainly three reasons…startup speed, runtime performance, and modularity. Let's discuss the components and working of Netflix.

Microservice Architecture of Netflix



Netflix's architectural style is built as a collection of services. This is known as microservices architecture and this power all of the APIs needed for applications and Web apps. When the request arrives at the endpoint it calls the other microservices for required data and these microservices can also request the data from different microservices. After that, a complete response for the API request is sent back to the endpoint.

In a microservice architecture, services should be independent of each other, for example, the video storage service would be decoupled from the service responsible for transcoding videos. Now, let's understand how to make it reliable…

Database

Netflix uses two different databases i.e. MySQL(RDBMS) and Cassandra(NoSQL) for different purposes.

EC2 Deployed MySQL

Netflix saves data like billing information, user information, and transaction information in MySQL because it needs ACID compliance. Netflix has a master-master setup for MySQL and it is deployed on Amazon large EC2 instances using InnoDB.

The setup follows the "**Synchronous replication protocol**" where if the writer happens to be the primary master node then it will be also replicated to another master node. The acknowledgment will be sent only if both the primary and remote master nodes' write have been confirmed. This ensures the high availability of data.

Netflix has set up the read replica for each and every node (local, as well as cross-region). This ensures high availability and scalability.

Cassandra

Cassandra is a NoSQL database that can handle large amounts of data and it can also handle heavy writing and reading. When Netflix started acquiring more users, the viewing history data for each member also started increasing. This increases the total number of viewing history data and it becomes challenging for Netflix to handle this massive amount of data. Netflix scaled the storage of viewing history data-keeping two main goals in their mind…

- Smaller Storage Footprint.
- Consistent Read/Write Performance as viewing per member grows (viewing history data write to read ratio is about 9:1 in Cassandra).

Mehmet Ali Er 181550033

Mehmet Karakuş 19155062