**Netflix System design**

Systems design implies a systematic approach to the design of a system. System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements.

**Functional Requirements:**

* Content upload (videos, shows)
* User streaming and interaction (like, dislike, share)
* Personalized video recommendations

**Non-Functional Requirements:**

* High availability with minimal latency
* Scalability and efficiency to handle millions of users

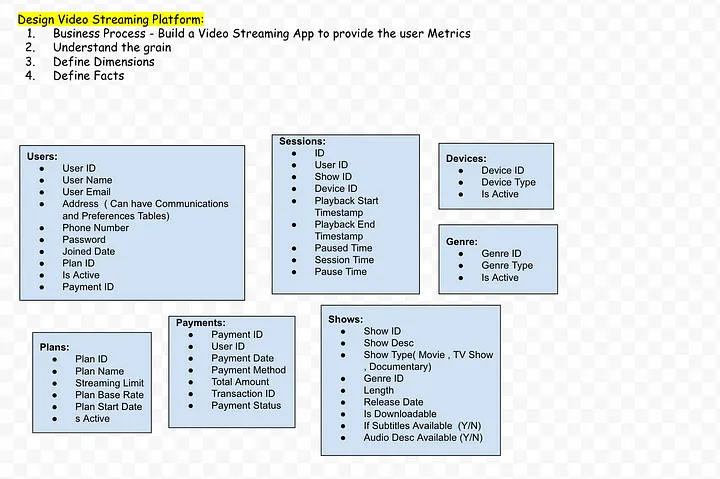
**System Design Principles:**

* Break down the problem into smaller components (services, features).
* Clearly define the system's goal and avoid overcomplication.
* Make reasonable assumptions about scale and usage (users, requests, data).
* Design data models and data flow between components.
* Employ high-level and low-level design approaches.

**Assumptions:**

* 1 billion users, 200 million daily active users
* 1 million videos, 1,000 new uploads daily
* 1 billion requests per day (12,000 per second)

**Data Storage and Management:**

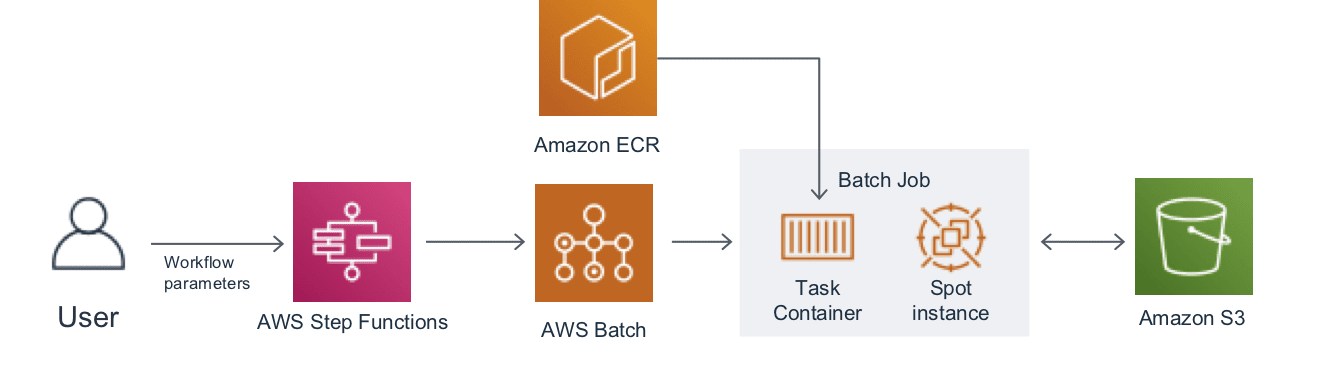


* User data (name, email, location, etc.)
* User interaction data (video history, ratings, etc.)
* Video data (ID, genre, cast, release year, stream URL, etc.)
* PostgreSQL for structured data (user, video)
* NoSQL database (Cassandra) for flexible schema (interactions)
* Data lake/warehouse (S3) for storing raw and preprocessed data

| Horizontal Scaling | Vertical Scaling |
| --- | --- |
| When additional machines are added to the existing system to meet the higher expectation, it is known as horizontal scaling. | When new resources are added in the existing system (increasing RAM, CPU) to meet the expectation, it is known as vertical scaling |
| It is easier to upgrade. | It is harder to upgrade and may involve downtime. |
| It is difficult to implement | It is easy to implement |
| It is costlier, as new server racks comprise a lot of resources | It is cheaper as we need to just add new resources |
| Cassandra, MongoDB, Google Cloud Spanner | MySQL and Amazon RDS |

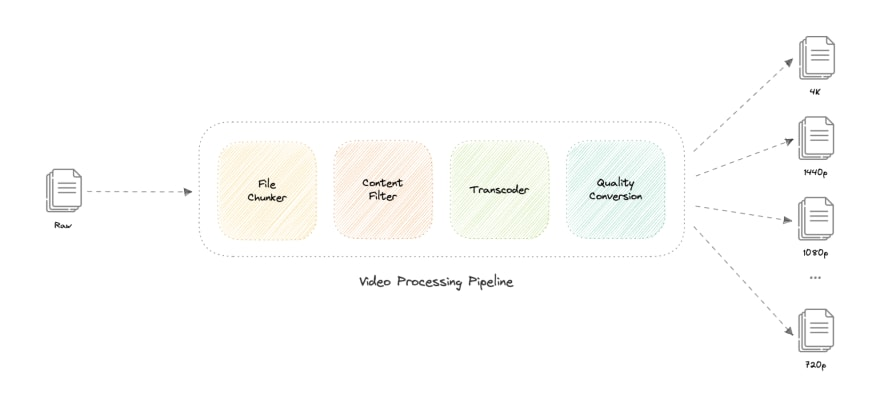
**Compute and Orchestration:**

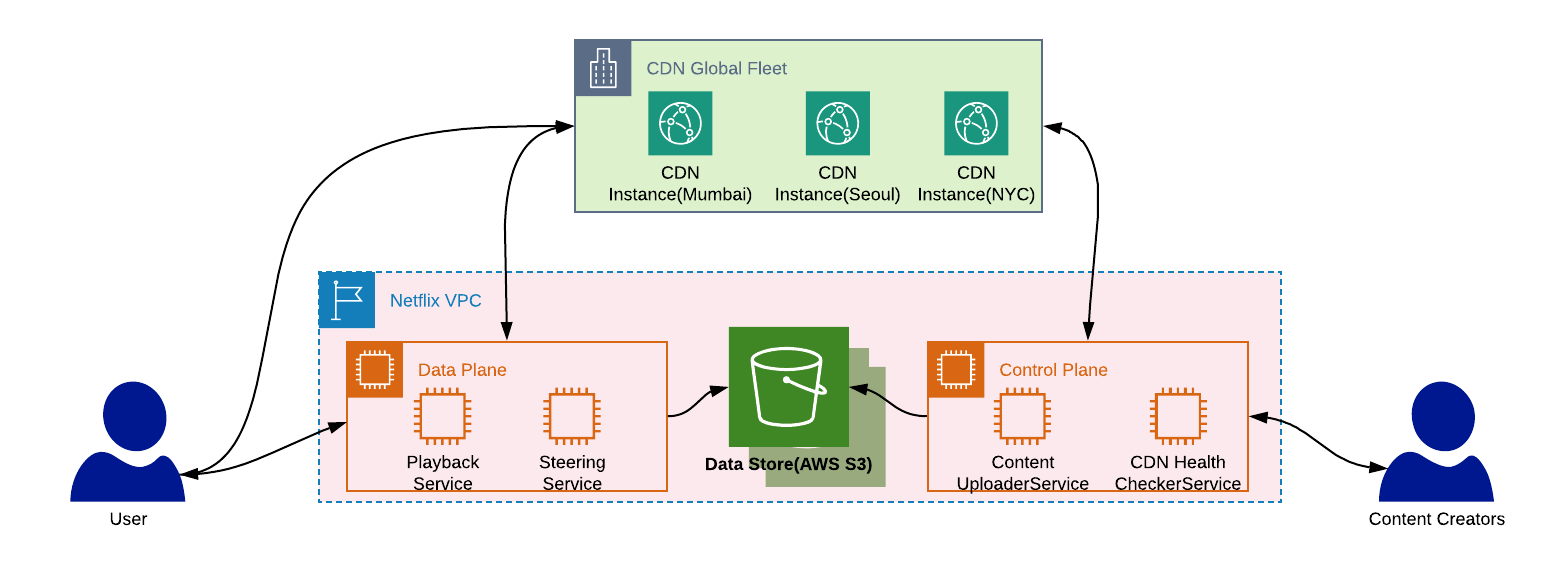
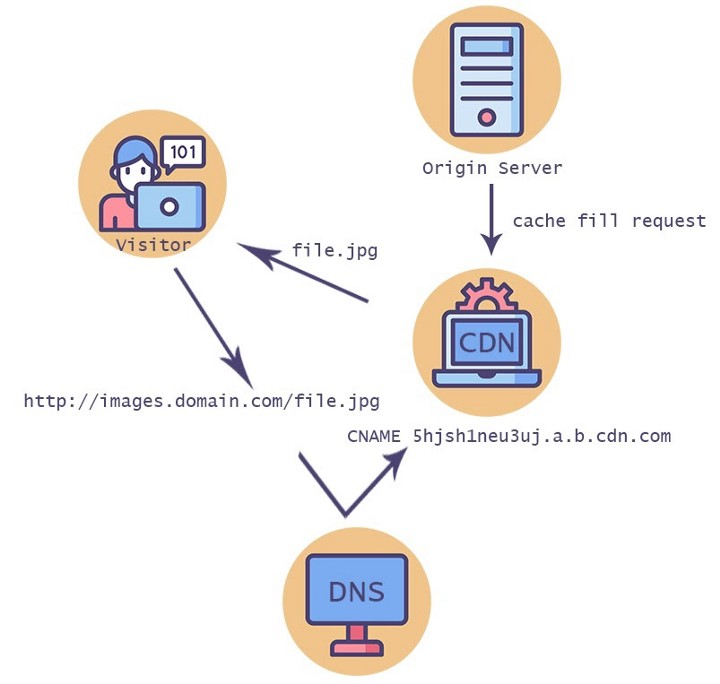
Today almost all of Netflix runs on VMs (virtual machines) in AWS. A customer's catalog browsing experience, content recommendation calculations, and payments are all served from AWS.



* Cloud-based infrastructure (AWS)
* EC2 virtual machines for core services
* AWS Step Functions and Batch for workflow automation and batch processing
* SageMaker notebooks for experimentation

**Video Content Storage and Delivery:**



* Amazon S3 for scalable and highly available storage
* Pre-encoded videos in multiple formats and resolutions for different devices
* Content Delivery Network (CDN) for geographically distributed content caching (Open Connect program)
* Low latency and high availability through geographically dispersed content replicas
* 
* 

**User Interactions and Recommendations:**

* User interactions stored in NoSQL database for scalability and flexible schema
* Recommendation systems based on machine learning algorithms (collaborative filtering, content-based filtering, etc.)
* Personalized recommendations based on viewing history, ratings, device, time of day, etc.

### **To make our system more resilient we can do the following:**

* Running multiple instances of each of our services.
* Introducing load balancers between clients, servers, databases, and cache servers.
* Using multiple read replicas for our databases.
* Multiple instances and replicas for our distributed cache.