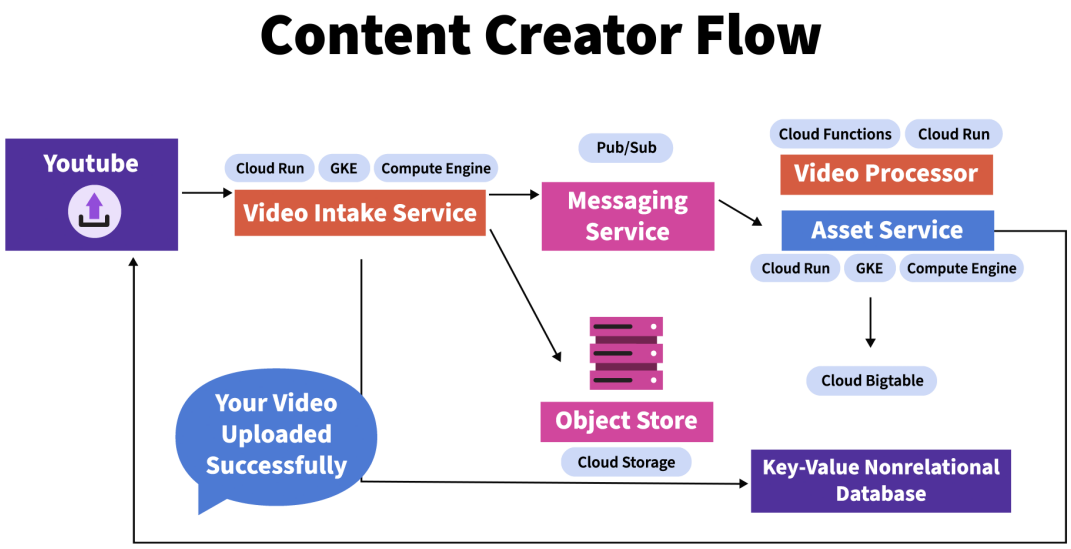
Video Streaming Service Architecture

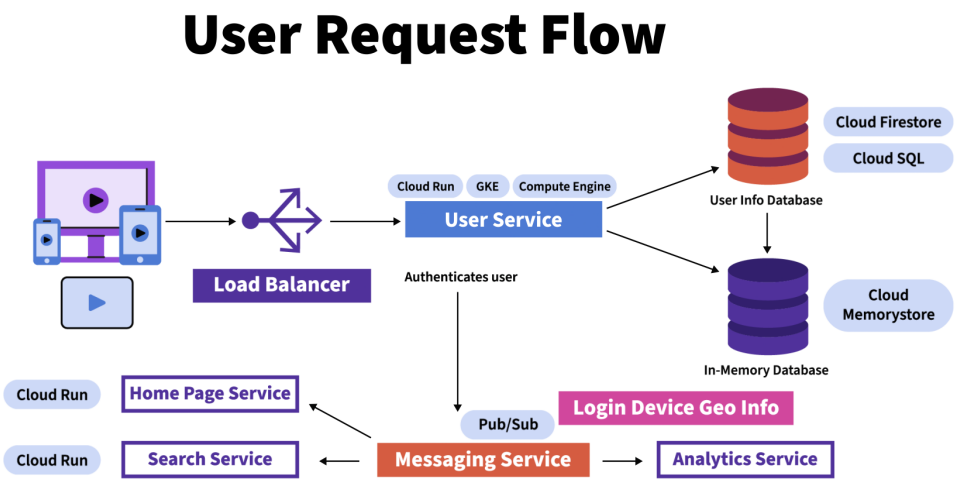
Content Creator Flow



- In Google Cloud, the “video intake service” can be on Compute Engine, Kubernetes engine, or Cloud Run.
- I recommend using containers for this service because it’s a modern architecture and gives more flexibility. This means our choices are Cloud Run if we want serverless and Google Kubernetes Engine (GKE) for more options to control.
- For videos and static files, we will use Cloud Storage (an object store in Google Cloud).
- For a metadata store or nonrelational key-value database, we will use Cloud Bigtable. It’s a NoSQL database for very low latency reads and writes.
- All the messages are sent to the messaging service: Pub/Sub in Google Cloud. Pub/Sub is an asynchronous and scalable messaging service that decouples the producer and subscriber’s messages with low latency.
- It ingests the messages, and a subscriber subscribes to those messages.
- The video processor is a service that subscribes to the Pub/Sub with the ingested file. The video processing service has multiple functions:
  - A file chunker service to break the file into small chunks
  - A content filter service filters unwanted content
  - A content tagging service adds relevant tags to the content

  - A thumbnail service creates thumbnails
  - A transcoder service creates different formats for bandwidth and devices
- Each of these services are single-purpose. They do the one-and-only task which is perfect to deploy on Cloud Functions, which is a function-as-a-service for event-driven serverless functions.
- All these services are in the video processor. You could also just kick off these processes from Google Cloud Storage.
- After all this processing, finally an uploader service uploads the large number of chunks to the object store (cloud storage) with CDN enabled.
- Once the processing happens, the Pub/Sub messaging service receives all the events for chunks and the asset service keeps track of final aggregation of the entire movie.

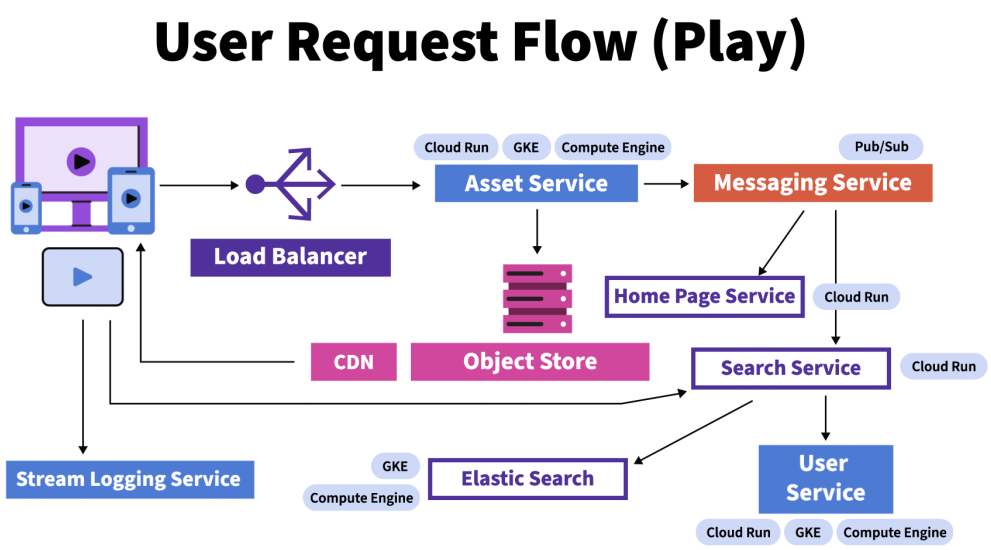
User Request Flow



- The user logs in from their device and that login request goes through Google Cloud external HTTPs load balancing service. Then it ends up on the user service, which authenticates the user and collects relevant information. This user service can be on Compute Engine, GKE or Cloud Run. I would keep this service pretty lightweight and use serverless Cloud Run here; it gives you the flexibility to scale as needed, so you don't need to think about infrastructure.
- User information can be stored in a nonrelational document database. I would use Firestore here. Firestore, a database-as-a-back-end service, is serverless, scales as needed, and offers flexibility so you can add more information about the user later without changing the schema.

- Although, you could also just use Cloud SQL (a managed relational database) or AlloyDB for managed PostgreSQL .
- In order to make things faster, I recommend enabling Memorystore for Redis. This in-memory database allows you to can cache requests, so you can provide users with a latency-free experience.
- All these requests from the user service generate an event in Pub/Sub.
- The analytics service, which can be deployed on Cloud Run, subscribes to Pub/Sub and collects information from our other services.

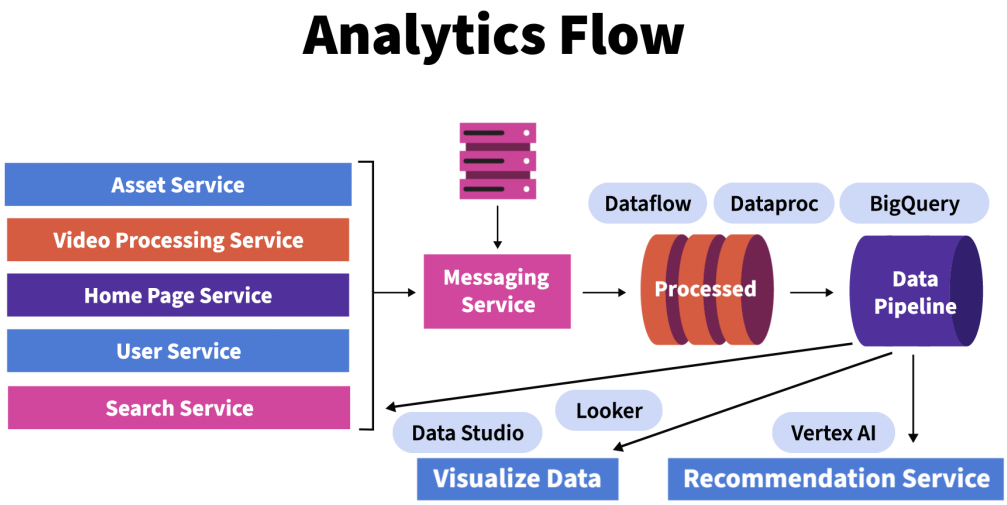
User Request Flow (Play)



- The request goes through the Google Cloud external load balancer.
- The asset service on Cloud Run gets information about the Cloud Storage URL for the video, and streams it via the CDN server closest to the user. In Google Cloud, the CDN is directly enabled at the load balancer.
- The Pub/Sub messaging service is ingesting all these messages asynchronously, and other services are subscribing to them!
- The users accesses the videos from the Home page service and the search service, both of which can be deployed on Cloud Run for fast and easy container-based development and scaling.
- The Home page service displays the videos with the desired thumbnail, genres, categories, etc.
- When a user searches for a movie, the load balancer directs the search query to the Search Service, which connects to Elasticsearch deployed on GKE or Compute Engine for fuzzy searching and type-ahead suggestions.

- The stream logging service deployed on Cloud Run logs stream information such as user interaction, including the length of time watched, likes, and dislikes—which can be used to determine a rating for the video.

Analytics Flow



- All the analytics-related data that these services were collecting in the analytics service can be streamed or batch uploaded in a data pipeline via Pub/Sub messaging.
- The data then gets processed in a processing service called Dataflow. Dataflow is a serverless, fully managed data processing service based on the open-source Apache Beam SDK.
- If your company uses Hadoop for data processing, then you can also use Dataproc. Dataproc is a managed hadoop service that makes it easy to deploy a Hadoop cluster in minutes, so you can start processing faster.
- Once the data is processed, it is stored in BigQuery—a serverless data warehouse you can use to manage and analyze your data using simple SQL commands. It lets you query terabytes in seconds and petabytes in minutes because it separates the compute to analyze your data from your storage. This architecture is pretty unique!
- As far as the visualization of the data for business users, you can do that in the Looker Studio to show off the trends, charts, and dashboards.
- Many the features in a video service are recommendations for what the user is interested in. If I watch *Iron Man*, then I might like *Mission Impossible* because they are both action movies. Or if I watch a Shah Rukh Khan movie, then I might like another romantic movie. All those are generated through data in a recommendation service. Vertex AI in Google Cloud lets you build those machine learning recommendation models.