# Instagram Live

*Instagram is already made and we are going to be building a new feature called instagram live. The data engineering team is already present and working on their own stuff.*

By:

1) Samee Haider
2) Jawad Ahmad Khan

# Functional Requirements:

1. Start a livestream
    a. The user can initiate a live video stream
    b. One active livestream per user at a time
2. Join and watch a livestream
    a. Followers of a streamer can discover and join livestream
    b. User can only join a single livestream at a time
3. Send and view comments
    a. Viewers can post comments during livestream
    b. Comments are visible to all viewers
    c. Viewers can view the viewercount as well
4. End a livestream
    a. Streamer can manually end a stream
    b. Once ended, viewers can no longer access it

# Non Functional Requirements:

1. The system must support approximately 30 million live viewers.
2. The system must support approximately 600 '000 daily streamers.
3. The system must support video latency of less than 2-5 seconds.
4. The system must support high availability.
5. Comments and viewer count can allow eventual consistency

# User growth assumptions (Slotta, 2025):

1. **Instagram app**

Monthly active users on instagram: 3 billion
Daily active users: 70% -> approx 2 billion
*Reasons for 70%*

1. *Addictive feed*
2. *Daily loop stories*
3. *Short form content (reels)*

2. **Instagram live feature (readers)**

Monthly active users on instagram live: 10% of Instagram MAU -> 0.3 billion -> 300 million
*10% due to event based feature, not everyone is free and not everyone likes every content*
Daily active users on instagram live: 10% -> 30 million readers per day

3. **Content creators (writers)**

Assuming 0.2% are creators: 0.2% of 3 billion -> 6 million creators per month
Daily active creators: 10% -> 600 000 creators per day

4. **Yearly Growth**

Approximately 15% (Thuy, 2025) compounding

5. **Peak usage**

Approximately 5 times.

# Other Approximations:

- Estimated 1 user views 1 live stream per day
- Estimated 1 creator creates 1 stream per day, so 600’000 streams per day
- Estimated 30% of the viewers comment so 9 million comments per day
- Live stream sent in 5 sec chunks

# QPS Estimation

QPS means queries per second

## Write QPS

In total we have the following operations.

a) Start Stream

In total we have 600 ’000 streams per day. That means 600’000/86400 -> 7 QPS

b) End Stream

In total we have 600 ’000 streams per day. That means 600’000/86400 -> 7 QPS

c) Join Stream (Viewer session)

In total approximately 30M viewers per day. That would mean 30M/86400 -> 347 QPS

d) Leave Stream (Viewer session)

In total approximately 30M viewers per day. That would mean 30M/86400 -> 347 QPS

e) Comments

In total 9M per day. That means 9M/86400 -> 104 QPS

- Total write QPS amounts to 7 + 7 + 347 + 347 + 104 -> 812 QPS
- Peak QPS -> 812 * 5 = approximately 5000 QPS

## Read QPS

In total we have the following type of operations.

1) DB/Backend
    a) Discover stream

    DAU is 30M and each viewer fetches one list per day. That means 30M/86400 -> 347QPS

    b) Fetch Individual livestream details

    DAU is 30M and each viewer fetches one list per day. That means 30M/86400 -> 347QPS

Peak QPS -> (347 + 347) * 5 -> 4000 QPS

2) Video/CDN Network QPS

Max concurrent viewers is 30M / 5 sec -> 6M chunks / sec. So 6M QPS.
Each chunk is 1.25 MB (calculated ahead) so 7.5 TB / sec network throughput.

3) Comments read

1000 comments/sec (calculations ahead) and 10k viewers/sec. Total deliveries 10M/sec so 10M QPS

# Storage Estimation

- Assumption: not considering the already stored data for Instagram

| **Table: USER (Re using)** | **Total: NA** |
|---|---|
| 1. User_id: uuid | - |

- Since Users would already be stored within the instagram system, we would not be involving the user table in our calculation.

| **Table: Followers (Re using)** | **Total: NA** |
|---|---|
| 1. User_id: uuid | - |
| 2. following_Id: uuid | - |

- Since followers would already be stored within the instagram system, we would not be involving the follower table in our calculation

| **Table: LIVESTREAM_STATS (Persistent)** | **Total: 44** |
|---|---|
| 1. Stream_id: uuid | 16 |
| 2. User_id: uuid | 16 |
| 3. total_comments: small int | 2 |
| 4. unique_viewers: int | 4 |
| 5. duration: big int | 8 |

- This is for the person creating a livestream and to be *used for performing analytics by the data engineering team.*
- Cost per entry: 44 Bytes
- Cost per day: 44 * 600’000 Users  = 26.4 MB

| **Table: VIEWERSESSION (Persistent)** | **Total: 64** |
|---|---|
| 1. Session_id: uuid | 16 |
| 2. user_id : uuid | 16 |
| 3. Stream_id: uuid | 16 |
| 4. Joined_at: datetime | 8 |
| 5. Left_at: datetime | 8 |

- This is for the person who joined the stream
- Cost per entry: 64 Bytes
- Cost per day: 64 * 30 million users = 1.92 GB

Therefore total costs per day of storing data: 26.4MB + 1.92 GB = approximately 2 GB per day
2 * 365 = 730 GB per year without growth for persistent growth

Applying the 15% growth

| Year 1 | Year 2 | Year 3 | Year 4 | Year 5 |
|---|---|---|---|---|
| 730 GB | 840 GB | 965 GB | 1.11 TB | 1.28 TB |

Approximately 5 TB for 5 years
Approximately 14.8 TB after 10 years

*All above calculations done for a single database with no replication of any sort*

**Maximum Ephemeral storage for videos being streamed**

- Worst case scenario is all 600 000 content creators start their livestreams at the same time
- Average Instagram live feed time: 20 minutes (Saini, 2026)

| **Table: COMMENT (Ephemeral)** | **Total: 312** |
|---|---|
| 1. Comment_id: uuid | 16 |
| 2. User_id: uuid | 16 |
| 3. Stream_id: uuid | 16 |
| 4. Content: string | 256 |
| 5. Created_at: datetime | 8 |

- This is for the person who joined the stream

**Estimations**

- Concurrent streamers: 600 ‘000
- Chunk size: 5 sec
- Buffer window: 30 sec -> 6 chunks
- Bitrate: 2Mbps (approximately 0.25 MB/sec)

**For video**

- Per chunk size: 0.25 MB * 5 sec -> 1.25 MB
- Per stream: 6 chunks * 1.25 MB -> 7.5 MB
- Total video ephemeral storage -> 600 ‘000 * 7.5 MB = 4 ‘500 ‘000 MB = 4.5 TB

**For Comments**

- 9 million comments per day, considering peak time and chances of comments on a celebrity post is much larger so estimating it to be around 1000 comments/sec in peak time.
- Assuming that comments remain in memory for like 10 seconds, that would mean 1000 * 10 -> 10 ‘000 comments in memory at a single time.
- 312 bytes for a single comment, so 312 * 10’000 -> 3 ‘120 ‘000 Bytes -> 3.12 MB

Total Ephemeral storage required is therefore 4.5 TB maximum.

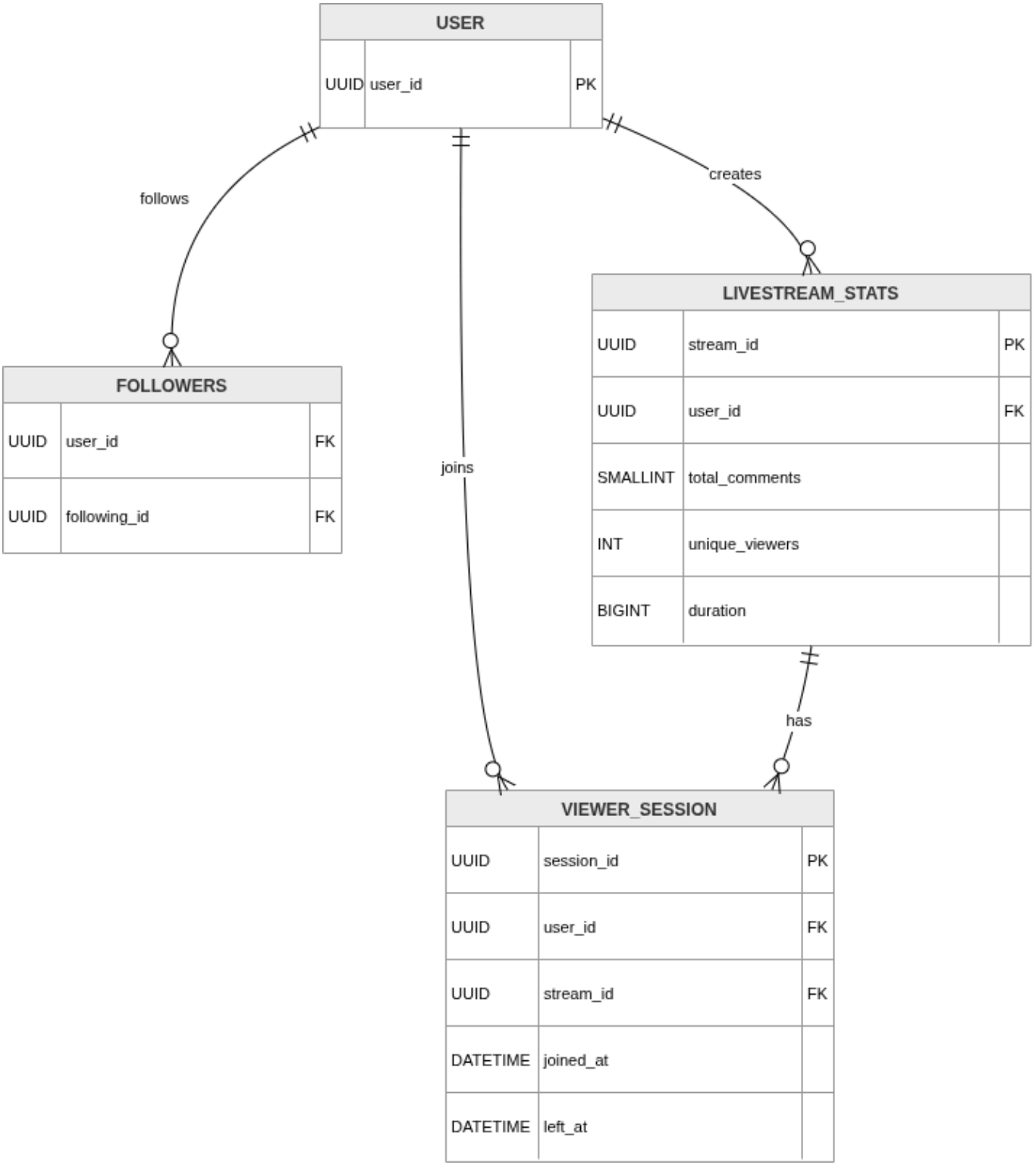
*The above estimation is not for a database but for object storage for video and in memory storage for comments without replication.*

# API Design and Structure

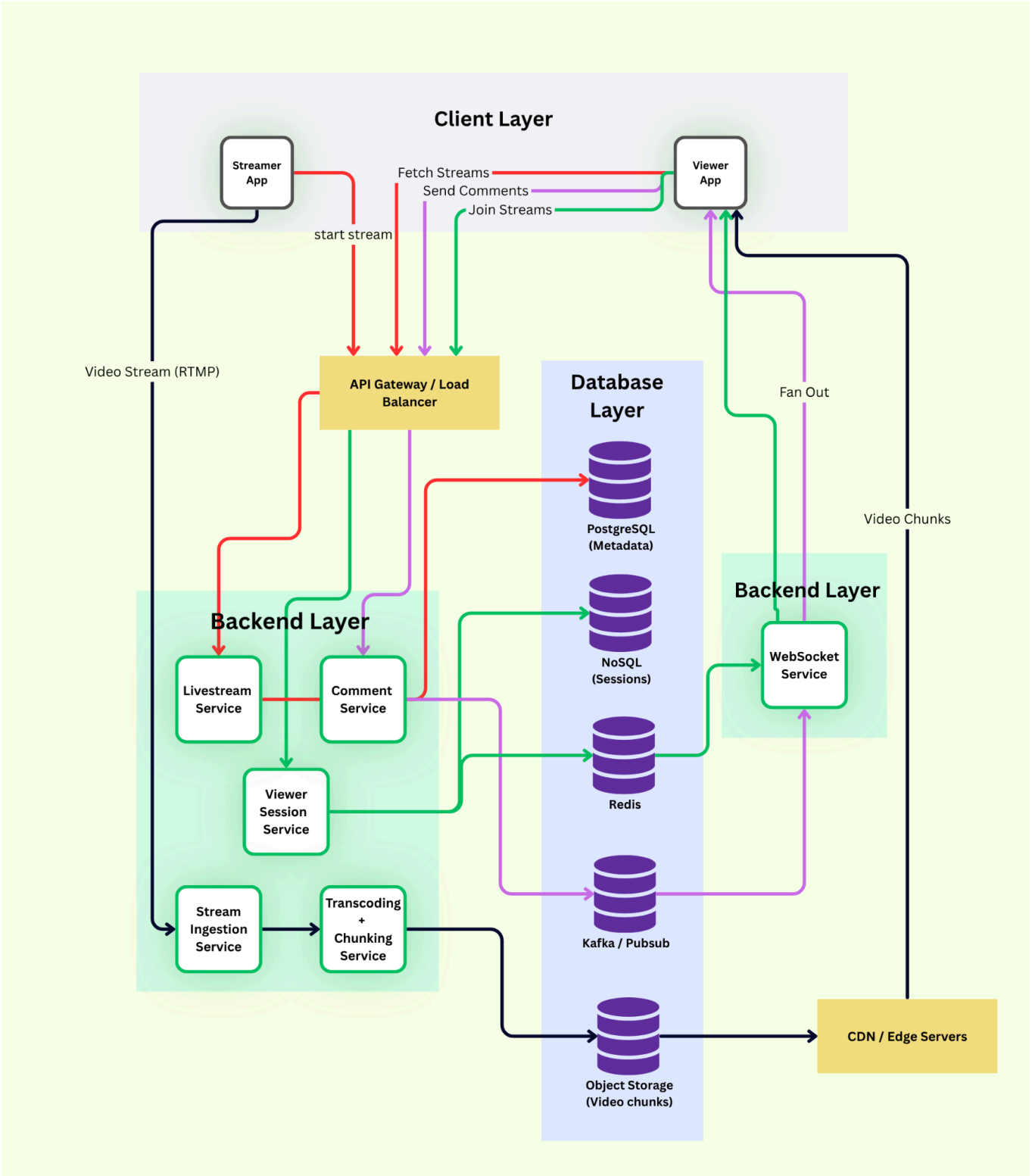
| Endpoint | Method | Description | Payload / Query Params |
|---|---|---|---|
| /v1/streams | POST | Initiates a new livestream. | **Req:** { "title": "string" }<br><br>**Res:** { "stream_id": "uuid", "ingest_url": "rtmp://..." } |
| /v1/streams/{stream_id}/end | PUT | Ends the livestream. | **Req:** {}<br><br>**Res:** 200 OK |
| /v1/streams/discover | GET | Fetches active streams of followed users. | **Query:** ?limit=20&offset=0<br><br>**Res:** [ { "stream_id", "user_id", "title" } ] |
| /v1/streams/{stream_id} | GET | Fetches details to join a stream. | **Res:** { "playback_url": "https://cdn...", "ws_url": "wss://..." } |
| /v1/streams/{stream_id}/join | POST | Records a VIEWERSESSION start. | **Req:** {}<br><br>**Res:** 201 Created |

| | | | |
|---|---|---|---|
| /v1/streams/{stream_id}/leave | POST | Ends the VIEWERSESSION. | **Req:** {}<br><br>**Res:** 200 OK |

# Entity and Relationship



# System Architecture



# Choice of Databases

1) PostgreSQL: for the stream metadata
2) NoSQL (Casandra): for the viewer sessions _ allow high write scalability and approximations aren't a problem, non critical data, no issue of eventual consistency
3) Redis: for viewer count _ very fast
4) Kafka: comments _ high throughput streaming
5) Object Storage (S3): video chunks
6) CDN: cached video _ massive read scaling

- ) We could have used RabbitMQ instead of Kafka if it wasn't streaming and the traffic was smaller.
- ) We could have chosen MongoDB instead of Cassandra, but MongoDB has a SPOF and Cassandra offers better scalability and availability.
- ) We could have used Block storage instead of S3 but for static videos S3 is cheaper.

# Other Architectural Decisions

- We are using both Kafka + Websockets to ensure scalability. Using only websockets will work but only at a very small scale. Kafka acts as a buffer or shock observer.
- Partitioning/Sharding across **stream_id** in table **ViewerSession**.
- Issue of hotspots if a famous celebrity starts a live stream with millions of followers.
    - Will use sub partitioning on the specific hotspot
- Since viewers can join and leave quickly and post comments, eventual consistency will be achieved which is not a big problem.
- There can be a delay in stream which is also not a big issue since it is not a critical system.
- The Comment table has a timestamp of which it was created to ensure that the comment matches the livestream chunk as well and can be as close as possible.
- Unique viewers and total comments will be added after the stream ends, so eventual consistency is achieved.

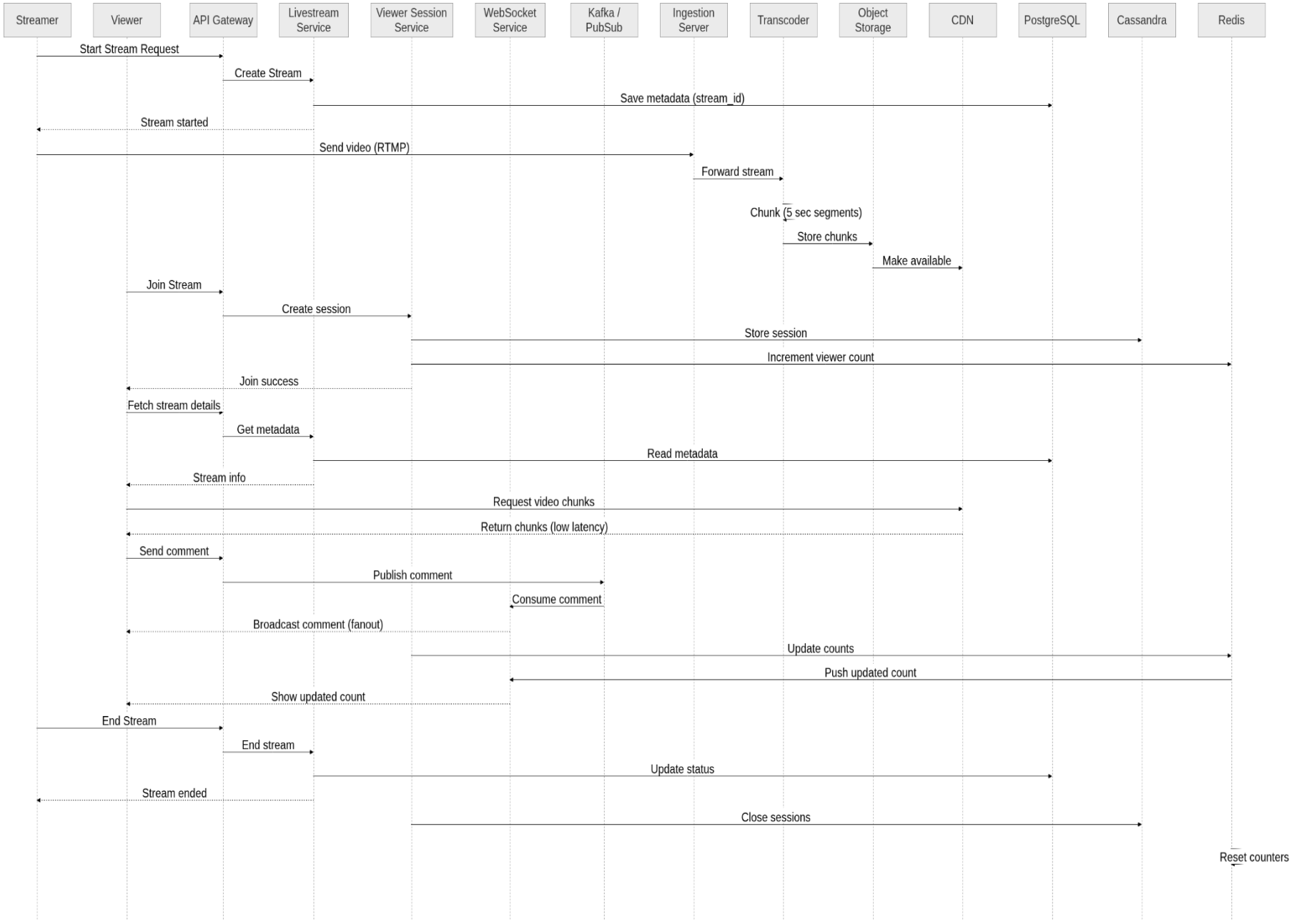
## Vertical vs Horizontal scaling

- **Vertical Scaling:** Transcoding machines would be working in parallel and are GPU intensive.
- **Horizontal Scaling:** web servers, websockets, redis clusters, kafka

## Fault tolerance and Recovery

- S3 offers 99.999999999 (11 9's) of data durability so we do not need a backup of our own and since the data being stored here is ephemeral, not a major issue of data loss, will not affect in a long run
- PostgreSQL has multi-AZ deployment with 1 synchronous replica at standby at all times.
- Cassandra has a masterless architecture with RF=3 across different zones for high availability, if for instance a node does go down, it will increase latency but not compromise availability.
- Redis has a primary replica structure setup, with each shard having a replica ready to take over in case of failure.
- Web servers, web sockets and the GPU intensive transcoding machines are in auto-scaling groups with health checks so will automatically spin up another server in case of failure or overload.
- If a websocket crashes, it will result in the *thundering herd* scenario; thousands of viewers will disconnect when the server crashes so we'll use exponential backoff with jitter, this will make the client wait some random time before attempting to reconnect to the healthy websocket server.

# Sequence Diagram



# References


Saini, M. (2026, 2 12). *30+ Instagram Live Statistics: How Live Streaming is Changing Engagement [2026]*. Cropink. https://cropink.com/instagram-live-statistics

Slotta, D. (2025, 12 17). *Instagram - statistics & facts*. Statista. https://www.statista.com/topics/1882/instagram/?srsltid=AfmBOor3GB0oA5BKVVD4xDgnOx6m-0fSvK7eZAF8l_RHfcr5saZcVolt#topicOverview

Thuy. (2025, 11). *Year-on-year audience growth of selected social media platforms worldwide as of October 2025*. Statista. https://www.statista.com/statistics/1294062/social-media-year-on-year-growth/