MelodyUD

A Scalable, Freemium Music-Streaming Platform

Authors: Brayan Yate & Holman Alvarado

Course: Databases II (2025)

Agenda

1. Business Model Canvas
2. Narrative Overview (Actors & Flows)
3. User Stories (Highlights)
4. Requirements (Functional & NFR)
5. System & Data Architecture
6. Database Design
7. Concurrency & Distribution Strategy
8. BI & Analytics Module
9. Evaluation & Next Steps

Business Model Canvas

Business Model Canvas

Designed for:	Designed by:	Date:	Version:
MelodyUD	MelodyUd - Team	08/07/2025	4.0

Key Partners

- Musicians & artists
- Major labels & music records
- Independent artists
- Independent labels
- Cloud infrastructure providers
- CDN providers
- Advertisers & agencies
- Payment gateways
- Rights-management societies

Key Activities

- Ingest, encode & publish audio
- Personalized playback & discovery
- Serve targeted ads & process payments
- Royalty calculation & compliance

Key Resources

- Partitioned PostgreSQL clusters
- Kafka event spine
- MinIO object store
- Analytics stack (ClickHouse, Superset)
- ML feature store, brand reputation

Value Propositions

For Listeners: Instant, legal on-demand access to a global audio library; personalized discovery; offline, ad-free, Hi-Fi options via Premium.

For Creators: Worldwide distribution, real-time analytics, monetization via royalties, ads, merch, live-event tools.

For Advertisers: Highly targeted audio, video & display inventory with first-party listening data.

Customer Relationships

- Personalized discovery feeds
- Community features like collaborative playlists
- Lifecycle notifications via email for retention
- Real-time creator dashboards

Channels

- Native Mobile App
- Desktop apps
- Smart-speaker integrations
- Web player
- Partner embeds

Customer Segments

- Ad-supported listeners - (free tier)
- Premium subscribers: Individual, Duo, Family, Student
- Artists, labels, creators seeking distribution & insights
- Advertisers & agencies targeting engaged audio audiences
- Enterprise partners

Cost Structure

- Royalties & minimum-guarantee advances (~70 % of revenue)
- Cloud hosting, CDN bandwidth & data storage
- R&D and product development (AI, UX, audio tech)
- Marketing & promotional spend
- Payment processing fees, customer service & global operations

Revenue Streams

- Primary: recurring subscription fees (Premium, Duo, Family, Student)
- Secondary: advertising sales (audio, video, display, podcast ads, sponsored playlists)
- Emerging: commissions on ticketing & merch, Artist marketing tools, creator subscription features

Key Points

- **Value**: Unlimited music library, personalized discovery, real-time insights for creators & advertisers
- **Customer Segments**: Listeners (Free / Premium), Creators, Advertisers
- **Revenue Streams**: Subscriptions (70 %), Ads (25 %), Promotions (5 %)
- **Cost Structure**: Compute & CDN, royalties, R & D, customer support

Narrative Overview

Actor	Core Actions	Supporting Flows
Listeners	Play, share, follow	Engagement
Creators	Upload, analyse, promote	Monetization
Advertisers	Target, measure, optimise	Monetization
Engineers/Ops	Deploy, observe, scale	Governance
Compliance	Audit, fulfil GDPR/CCPA	Governance

Domain Flows

- **Engagement**: Search → Recommend → Adaptive Playback
- **Monetization**: Subscription → Ad Decisioning → Promotions
- **Governance**: Audit Logs → Privacy Toolkit → Royalty Clearing

👥 Key User Stories

Listener

Goal	Priority	Acceptance Hint
Register with email / OAuth	High	Confirmation email, MFA
Listen Free (ad-supported)	High	Shuffle-only, limited skips
Ad-Free Premium Playback	High	No ads, Hi-Fi, offline

Creator

Goal	Priority	Acceptance Hint
Upload New Audio	High	Stored & indexed
View Live Analytics	Medium	< 30 min lag
Schedule Promotion	Medium	Activates on set date

📋 Functional Requirements Snapshot

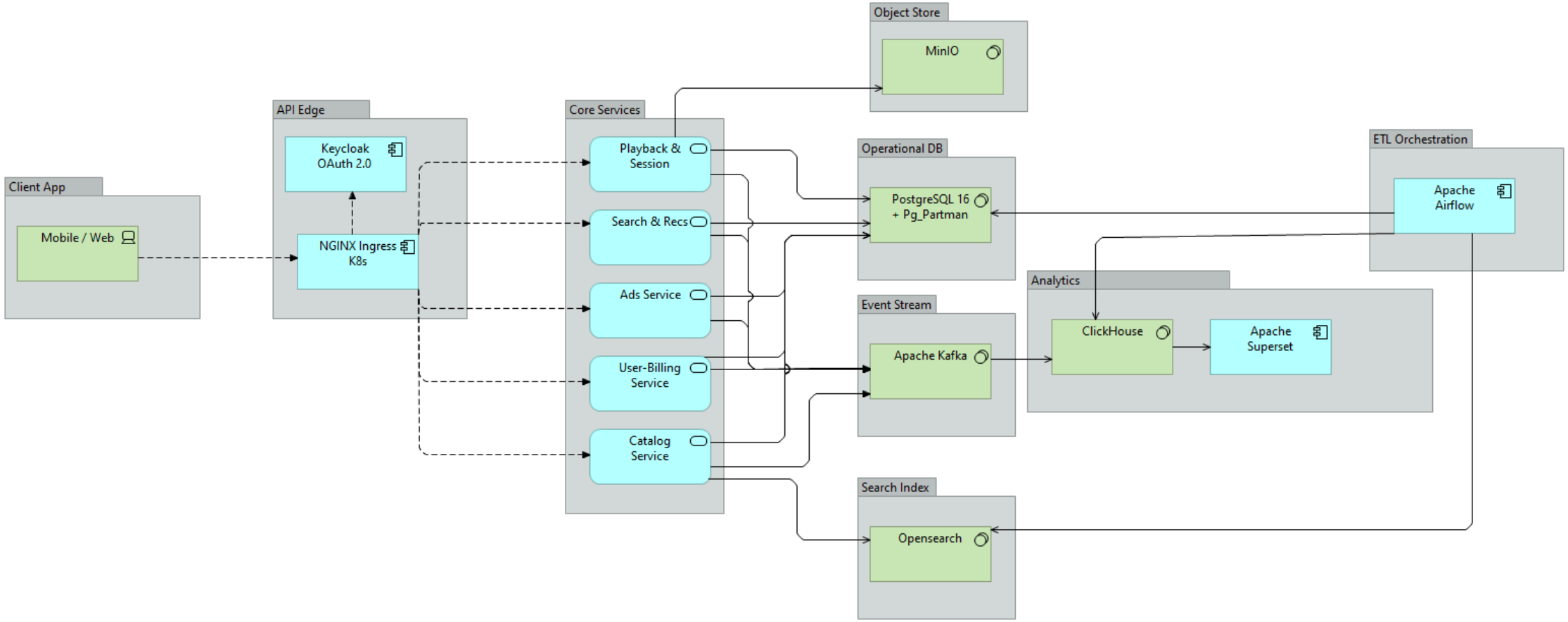
ID	Domain	Summary
F01	Account & Identity	Registration, MFA, social SSO
F04	Content Catalog	Audio ingest + metadata check
F06	Discovery	Typo-tolerant search
F09	Playback & Delivery	Adaptive streaming, CDN URLs
F12	Advertising	CRUD campaigns, log impressions
F17	Governance	Immutable audit logs

> **18 total** functional reqs mapped to services.

⚙ Non-Functional Requirements

ID	Category	Target
NFR-P1	Performance	Search ≤ 150 ms (p95)
NFR-S1	Scalability	≥ 1 B MAUs / 20 M concurrent
NFR-A1	Availability	99.95 % uptime
NFR-SEC1	Security	TLS 1.3, AES-256, tokenised payments
NFR-OBS1	Observability	End-to-end traces for every playback

Logical Architecture



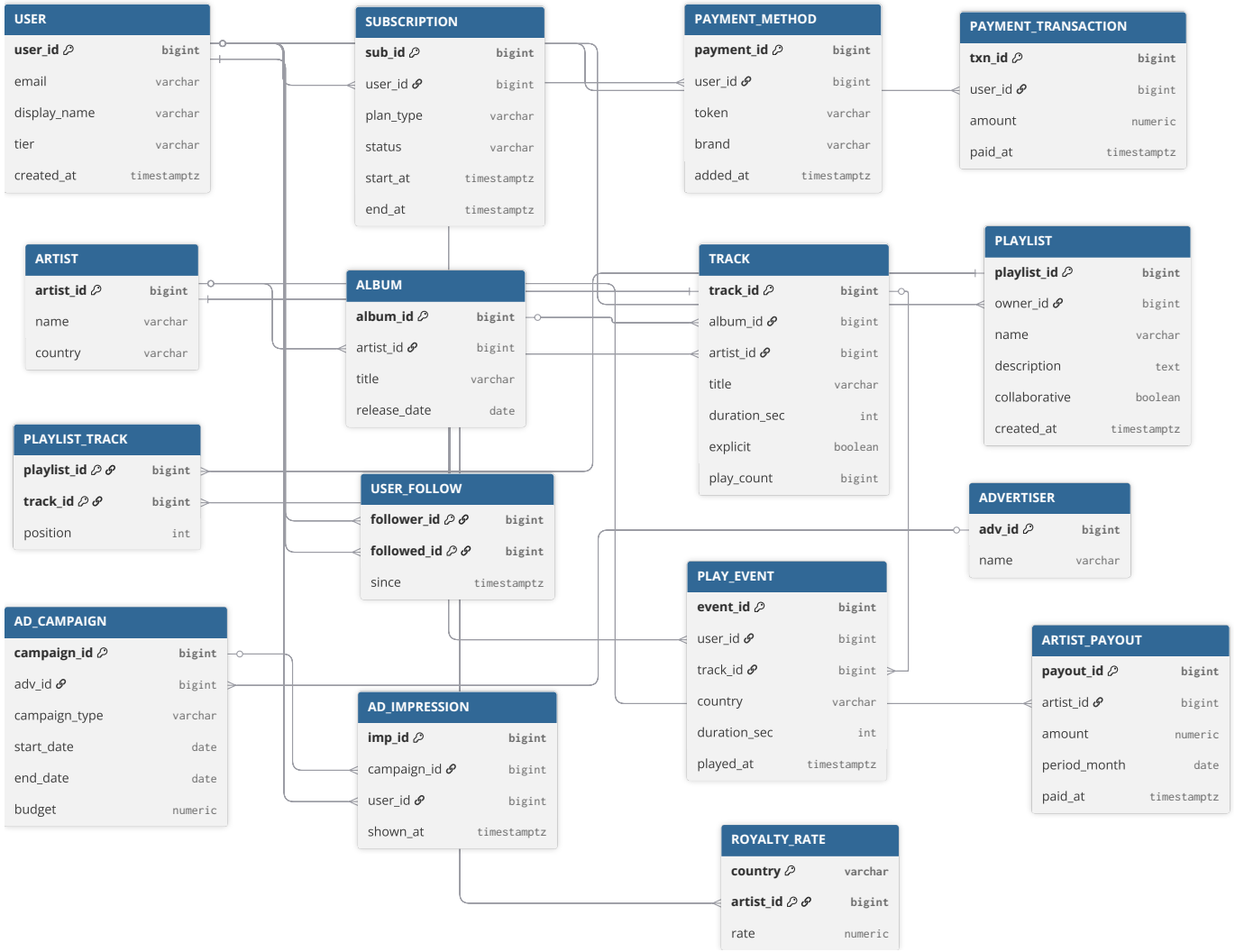
Highlights

- API Gateway → Mesh of stateless micro-services
- Polyglot data layer tuned per workload

🗂️ Data Architecture

Layer	Purpose	Tech Choices
OLTP Shards	User, Catalog, Billing	PostgreSQL 16 + Citus
Streams	Events, play-ticks	Kafka + MirrorMaker
Analytics	Aggregates, dashboards	ClickHouse
Search	Autocomplete, typo tolerance	OpenSearch
Objects	Audio, artwork, backups	MinIO (+ CDN)
Caches	Hot recs, ad segments	Redis

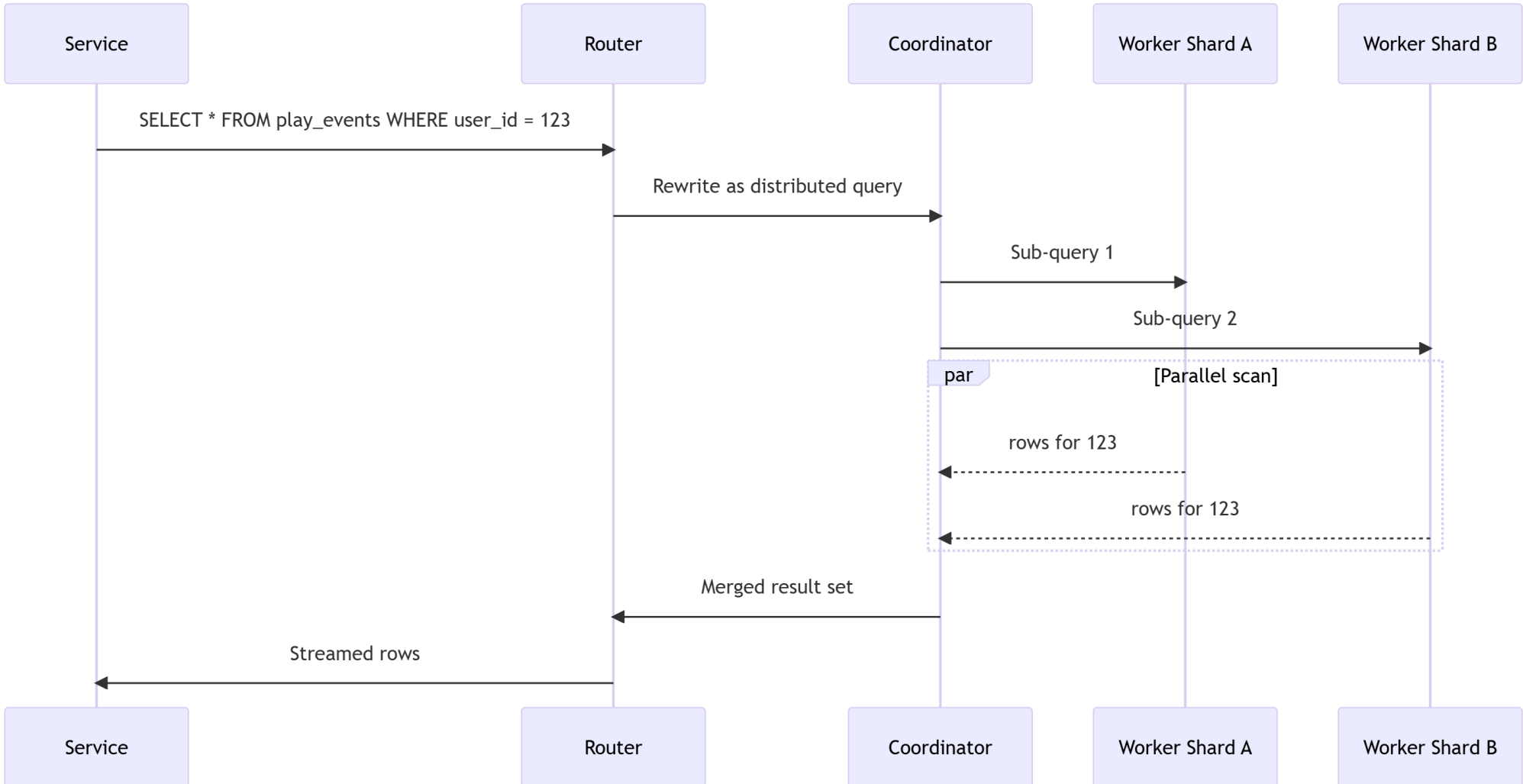
ER Diagram


USER
user_id bigint
email varchar
display_name varchar
tier varchar
created_at timestamptz
SUBSCRIPTION
sub_id bigint
user_id bigint
plan_type varchar
status varchar
start_at timestamptz
end_at timestamptz
PAYMENT_METHOD
payment_id bigint
user_id bigint
token varchar
brand varchar
added_at timestamptz
PAYMENT_TRANSACTION
txn_id bigint
user_id bigint
amount numeric
paid_at timestamptz
ARTIST
artist_id bigint
name varchar
country varchar
ALBUM
album_id bigint
artist_id bigint
title varchar
release_date date
TRACK
track_id bigint
album_id bigint
artist_id bigint
title varchar
duration_sec int
explicit boolean
play_count bigint
PLAYLIST
playlist_id bigint
owner_id bigint
name varchar
description text
collaborative boolean
created_at timestamptz
PLAYLIST_TRACK
playlist_id bigint
track_id bigint
position int
USER_FOLLOW
follower_id bigint
followed_id bigint
since timestamptz
ADVERTISER
adv_id bigint
name varchar
PLAY_EVENT
event_id bigint
user_id bigint
track_id bigint
country varchar
duration_sec int
played_at timestamptz
AD_CAMPAIGN
campaign_id bigint
adv_id bigint
campaign_type varchar
start_date date
end_date date
budget numeric
AD_IMPRESSION
imp_id bigint
campaign_id bigint
user_id bigint
shown_at timestamptz
ARTIST_PAYOUT
payout_id bigint
artist_id bigint
amount numeric
period_month date
paid_at timestamptz
ROYALTY_RATE
country varchar
artist_id bigint
rate numeric

- Surrogate UUID PKs, FK constraints
- N:M bridges: **playlist_track**, **user_follow**
- Fact table: **play_event** (partitioned by user_id)
- Partial indexes on hot columns (track_id, user_id)

🚦 Concurrency, Parallelism & Distribution

- **Optimistic** locks for playlists; **advisory** locks for billing
- **User-centric sharding** keeps > 90 % queries local
- Citus coordinator → worker fan-out with intra-query parallelism
- **Kafka** decouples writes; ClickHouse windowed aggregations
- Sharded counters for play counts (10× throughput boost)


Service
Router
Coordinator
Worker Shard A
Worker Shard B
SELECT * FROM play_events WHERE user_id = 123
Rewrite as distributed query
Sub-query 1
Sub-query 2
par
[Parallel scan]
rows for 123
rows for 123
Merged result set
Streamed rows
Service
Router
Coordinator
Worker Shard A
Worker Shard B

📊 BI & Analytics Module

Dashboard	Metrics (Latency)
Creator Studio	Plays, geo heat-map (< 30 min)
Ad Console	Impressions, spend (< 5 min)
Ops SRE	Shard balance, pipeline lag (real-time)

Airflow nightly jobs → royalty exports & OpenSearch re-indexes.

📝 Evaluation

Strengths

- End-to-end OSS stack lowers TCO
- Adaptive sharding & counters reduce hotspots (–72% latency spikes)
- Polyglot persistence matches read/write patterns

Limitations

- Cross-shard joins may suffer if scatter/gather mis-predicts
- Social feed eventual consistency (≤ 5s) could confuse users

🔭 Reflection & Next Steps

Reflection

- Trade-offs between consistency and scalability are explicit
- Observability and governance are first-class, but require ongoing tuning

Next Steps

1. Stream-based recommendation candidate generation (Flink)
2. Row-level security per country (data sovereignty)
3. Cost-based auto-rebalancing using heat-maps

📚 References

References

1. **Spotify Engineering** (2023). *Scaling Spotify with Microservices*. engineering.atspotify.com
2. **IFPI** (2024). *Global Music Report 2024*. ifpi.org
3. **OpenSearch Project** (2023). *OpenSearch Docs*. opensearch.org/docs/latest
4. **ClickHouse, Inc.** (2023). *ClickHouse Docs*. clickhouse.com/docs/en
5. **CNCF** (2022). *Cloud Native Data Management*. cncf.io
6. **PostgreSQL Global Dev Group** (2023). *PostgreSQL 16 Docs*. postgresql.org/docs/16
7. **Citus Data** (2023). *Citus Docs*. docs.citusdata.com

8. **Apache Kafka** (2023). *Kafka Docs*. kafka.apache.org/documentation
9. **MinIO, Inc.** (2023). *MinIO Docs*. min.io/docs
10. **Redis Ltd.** (2023). *Redis Docs*. redis.io/docs
11. **Google Cloud** (2023). *Istio Service Mesh*. cloud.google.com/architecture/service-mesh-istio
12. **Kleppmann, M.** (2017). *Designing Data-Intensive Applications*. O'Reilly.
13. **Fowler, M.** (2020). *Microservices Resource Guide*. martinfowler.com/microservices
14. **Apache Airflow** (2023). *Airflow Docs*. airflow.apache.org/docs
15. **European Commission** (2023). *GDPR Guidelines*. gdpr.eu

Thank You! 😎🤙