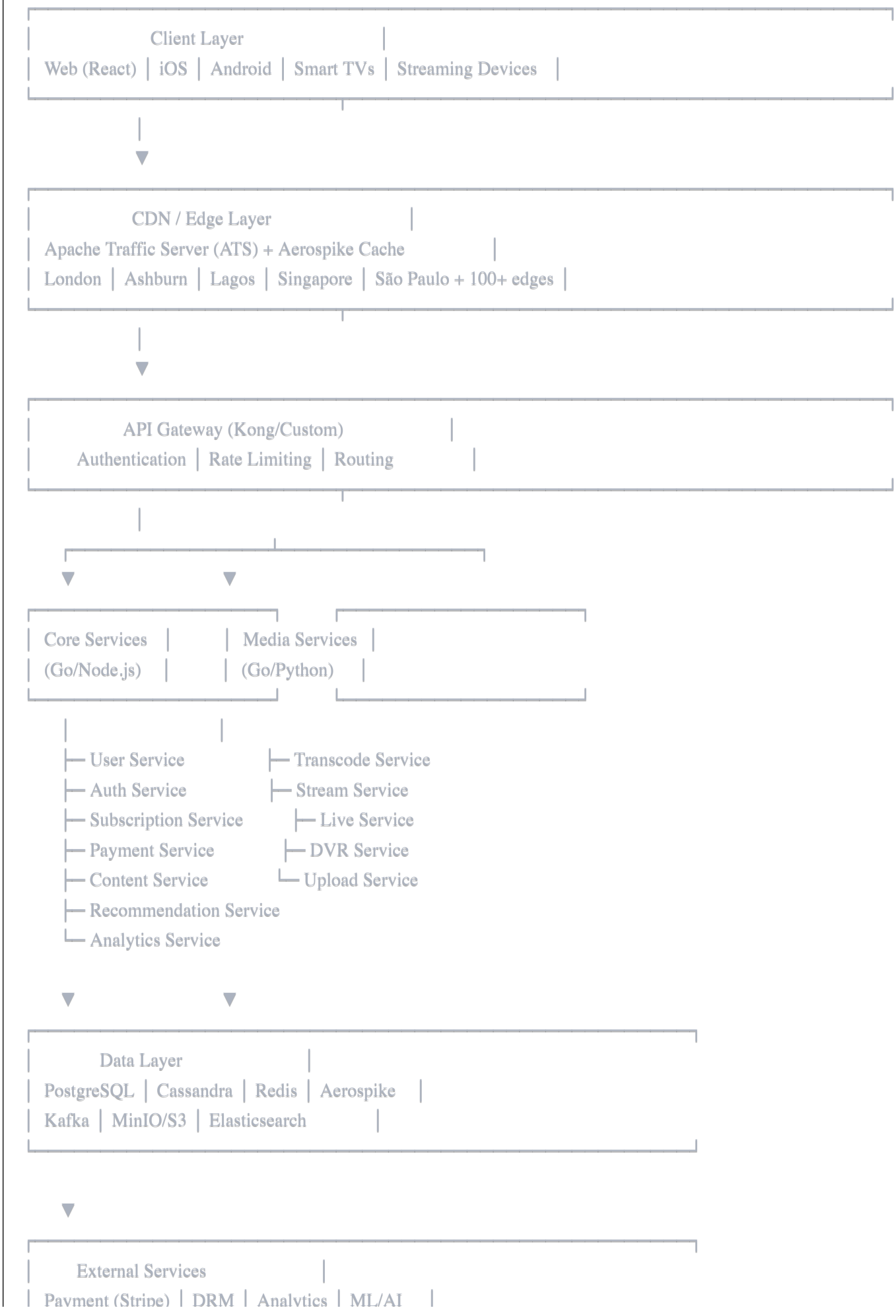
System Architecture: Global Streaming Platform

1. High-Level Architecture


Client Layer
Web (React) | iOS | Android | Smart TVs | Streaming Devices
CDN / Edge Layer
Apache Traffic Server (ATS) + Aerospike Cache
London | Ashburn | Lagos | Singapore | São Paulo + 100+ edges
API Gateway (Kong/Custom)
Authentication | Rate Limiting | Routing
Core Services
(Go/Node.js)
Media Services
(Go/Python)
User Service
Auth Service
Subscription Service
Payment Service
Content Service
Recommendation Service
Analytics Service
Transcode Service
Stream Service
Live Service
DVR Service
Upload Service
Data Layer
PostgreSQL | Cassandra | Redis | Aerospike
Kafka | MinIO/S3 | Elasticsearch
External Services
Payment (Stripe) | DRM | Analytics | ML/AI

2. Microservices Architecture

2.1 Core Services

User Service

- **Responsibility**: User profiles, preferences, watch history
- **Tech**: Go + PostgreSQL + Redis
- **API**: gRPC internal, REST external
- **Scale**: 10K RPS per instance

Auth Service

- **Responsibility**: Authentication, JWT tokens, sessions
- **Tech**: Go + Redis + PostgreSQL
- **Features**: OAuth 2.0, JWT, refresh tokens, MFA
- **Scale**: 50K RPS per instance

Subscription Service

- **Responsibility**: Plans, billing cycles, entitlements
- **Tech**: Go + PostgreSQL + Kafka
- **Integration**: Stripe webhooks, payment processing
- **Scale**: 5K RPS per instance

Content Service

- **Responsibility**: Metadata, catalogs, search
- **Tech**: Go + PostgreSQL + Elasticsearch
- **Features**: Multi-language metadata, rights management
- **Scale**: 100K RPS per instance (read-heavy)

Recommendation Service

- **Responsibility**: Personalized recommendations
- **Tech**: Python + TensorFlow Serving + Redis
- **Models**: Collaborative filtering, deep learning
- **Scale**: 20K RPS, <100ms P95 latency

2.2 Media Services

Transcode Service

- **Responsibility**: VOD transcoding pipelines
- **Tech**: Go orchestration + GStreamer workers
- **Profiles**: 6-8 ABR profiles (240p-4K)
- **Scale**: 1000+ concurrent jobs

Stream Service

- **Responsibility**: Manifest generation, packaging
- **Tech**: Go + Nginx/ATS
- **Protocols**: HLS, DASH, LL-HLS
- **Scale**: 1M+ concurrent streams

Live Service

- **Responsibility**: Live ingest, transcoding, distribution
- **Tech**: OvenMediaEngine + Go orchestration
- **Features**: WebRTC, LL-HLS, DVR buffer
- **Scale**: 10K+ concurrent live streams

DVR Service

- **Responsibility**: Cloud DVR recording, playback
- **Tech**: Go + GStreamer + MinIO
- **Storage**: Time-shifted content (7-30 days)
- **Scale**: 100K concurrent recordings

2.3 Analytics & ML Services

Analytics Service

- **Responsibility**: Event ingestion, real-time metrics
- **Tech**: Go + Kafka + Cassandra + Druid
- **Events**: Play, pause, seek, buffering, errors
- **Scale**: 1M events/sec

ML Service

- **Responsibility**: Model training, inference
- **Tech**: Python + PyTorch + Kubeflow
- **Models**:
  - Recommendation (collaborative + deep learning)
  - CDN selection (gradient boosting)
  - Churn prediction (XGBoost)
  - Content understanding (computer vision)

3. Data Architecture

3.1 Databases

PostgreSQL (Transactional)

- **Usage**: User accounts, subscriptions, content metadata
- **Setup**: Primary-replica with read replicas per region
- **Backup**: Daily full, hourly incremental (30-day retention)
- **HA**: Patroni + pgBouncer connection pooling

Cassandra (Time-Series & High Write)

- **Usage**: Analytics events, playback history, logs
- **Setup**: 3 nodes per datacenter, RF=3
- **Compaction**: Time-window compaction strategy
- **TTL**: 90 days for raw events, 2 years for aggregates

Redis (Cache & Session)

- **Usage**: Session storage, hot data cache, rate limiting
- **Setup**: Redis Cluster (6 shards, 3 replicas each)
- **Eviction**: LRU for cache, no eviction for sessions
- **Persistence**: AOF for sessions, snapshots for cache

Aerospike (CDN Cache Metadata)

- **Usage**: Content hotness scores, TTL hints, edge cache coordination
- **Setup**: 5 nodes per Tier 1 PoP (rack-aware)
- **Replication**: Cross-region replication for hotness scores
- **Performance**: <1ms P99 read latency

3.2 Message Queue

Apache Kafka

- **Topics**:
  - `user.events` (clicks, views)
  - `playback.events` (play, pause, seek)
  - `billing.events` (subscriptions, payments)
  - `content.events` (uploads, transcode jobs)
- **Retention**: 7 days (events), 30 days (billing)
- **Partitions**: 50-100 per topic (based on throughput)
- **Consumers**: Analytics, ML training, real-time dashboards

3.3 Object Storage

MinIO (Warm/Cold Tier)

- **Usage**: Transcoded video segments, images, backups
- **Setup**: Distributed mode, erasure coding (EC:4+2)
- **Lifecycle**: Auto-tier to cold storage after 60 days
- **CDN**: Origin for ATS cache pulls

Cloudflare R2 (Global Hot Tier)

- **Usage**: Most-watched content globally
- **Setup**: R2 + Cache Reserve for ultra-low latency
- **Sync**: Selective replication based on Aerospike hotness scores

4. CDN Architecture

4.1 Apache Traffic Control (ATC) Setup

Traffic Ops (Control Plane)

- **Function**: CDN configuration, delivery service management
- **API**: RESTful API for automation
- **Database**: PostgreSQL for CDN config
- **UI**: Web console for operators

Traffic Router (DNS/HTTP Routing)

- **Function**: Geo-based user → optimal edge routing
- **Method**: GeoDNS + HTTP 302 redirects
- **Metrics**: Real-time latency, capacity monitoring
- **Failover**: Automatic unhealthy edge exclusion

Traffic Monitor

- **Function**: Edge server health checks, metrics collection
- **Interval**: 10s health checks, 60s stats aggregation
- **Data**: CPU, disk, network, cache hit ratio per edge
- **Alerting**: PagerDuty integration for failures

4.2 Apache Traffic Server (ATS) Configuration

L2 Parent Servers (Tier 1 PoPs)

- **Hardware**: 128GB RAM, 20TB NVMe SSD, 40Gbps NIC
- **Cache Size**: 15TB hot cache per server
- **Config**:
  - Strict parenting: `go_direct=false`
  - Origin: MinIO, R2
  - TTL: Manifest 15s, segments 3600s (from Aerospike)

L1 Edge Servers (100+ Locations)

- **Hardware**: 64GB RAM, 4TB NVMe SSD, 10Gbps NIC
- **Cache Size**: 3TB hot cache per server
- **Config**:
  - Parent: Nearest L2
  - Cachekey plugin: Remove `?token=` query params
  - Header rewrite: Add `X-Region`, remove `Pragma`

4.3 Aerospike Integration

Hotness Tracking

```
Key: content_id (video/channel ID)
Record: {
  hotness_score: float (0-1, computed from recent views),
  ttl_hint: int (seconds, for ATS cache),
  last_accessed: timestamp,
  region_scores: map<region, float>
}
```

Replication Strategy

- **S1 (Tier 0)**: Top 1% hottest content → replicate globally
- **S2 (Tier 1)**: Top 10% → replicate to relevant regions
- **S3 (Long Tail)**: Pull from origin on-demand, cache locally

TTL Management (Lua Plugin in ATS)

```lua
function do_remap()
  local content_id = extract_content_id(ts.client_request.get_uri())
  local aerospike_record = query_aerospike(content_id)

  if aerospike_record then
    local ttl = aerospike_record.ttl_hint
    ts.client_response.header['Cache-Control'] = 'max-age=' .. ttl
  else
    -- Default TTL
    ts.client_response.header['Cache-Control'] = 'max-age=3600'
  end

  return 0
end
```

5. Streaming Architecture

5.1 VOD Workflow

1. Content Upload → MinIO
2. Transcode Job → Kafka → Transcode Service
3. GStreamer: Generate ABR ladder (6 profiles)
4. Package: HLS/DASH manifests + segments → MinIO
5. Manifest URL → Content Service → User request
6. Playback:
   - User → DNS → Traffic Router
   - Traffic Router → Optimal ATS edge
   - ATS edge → (cache hit?) → User
   - ATS edge → (cache miss?) → L2 parent → MinIO → cache → User

5.2 Live Streaming Workflow

1. Ingest Source (RTMP/SRT) → OvenMediaEngine cluster
2. OME: Transcode to ABR profiles (real-time)
3. OME: Generate LL-HLS (2s segments, 0.5s parts)
4. OME: Publish to ATS edge (origin-push or edge-pull)
5. Playback:
   - User → ATS edge (cached live segments)
   - <3s glass-to-glass latency

5.3 FAST Channel Architecture

1. Scheduler Service: Pre-planned content grid (24hrs+)
2. Playout Service:
   - Read schedule → fetch VOD segments from cache
   - Generate continuous HLS stream (server-side stitching)
   - Insert ads (SSAI) based on ad decision service
3. Output: HLS/DASH endpoint per channel
4. Distribution: Via ATS CDN (same as VOD)

6. Security Architecture

6.1 DRM Flow

1. User requests protected content
2. Auth Service: Verify entitlement (subscription/purchase)
3. License Server:
   - Generate license (Widevine/PlayReady/FairPlay)
   - Include playback restrictions (HDR, 4K based on tier)
4. Client: DRM decryption → playback
5. Content: AES-128 or AES-256 encrypted segments

6.2 API Security

- **Authentication**: JWT (access token 15min, refresh token 30 days)
- **Authorization**: RBAC (roles: admin, editor, viewer, guest)
- **Rate Limiting**: Token bucket (100 req/min per user, 10K/min per API key)
- **WAF**: Cloudflare/AWS WAF for DDoS, injection attacks
- **Encryption**: TLS 1.3 only, HSTS enabled

7. Scalability & Performance

7.1 Auto-Scaling Strategy

Service	Metric	Scale Up	Scale Down
API Gateway	CPU > 70%	+20% instances	-10% instances
Stream Service	Concurrent streams > 80% capacity	+50% instances	-25% instances
Transcode Workers	Queue depth > 100 jobs	+10 workers	-5 workers
Database Read Replicas	Replication lag > 5s	+1 replica	Manual

7.2 Performance Targets

Metric	Target	Measurement
API Response Time (P95)	<200ms	Prometheus
Video Startup Time (P95)	<2s	Client telemetry
Buffering Ratio	<0.5%	Playback events
CDN Hit Ratio	≥90%	ATS logs → Aerospike
Concurrent Streams	5M+	Stream Service metrics
Transcode Time	1x realtime (VOD)	Transcode Service

7.3 High Availability

- **Multi-AZ Deployment**: All services in 3 AZs per region
- **Database**: Primary + sync replica + 2 async replicas
- **CDN**: N+1 redundancy (if 1 PoP fails, reroute to next nearest)
- **Kafka**: 3 brokers per cluster, RF=3, min ISR=2
- **Load Balancers**: Active-active ALB/NLB with health checks

8. Monitoring & Observability

8.1 Metrics (Prometheus)

Infrastructure:

- CPU, memory, disk, network per service
- Request rate, latency, error rate (RED method)

Business:

- Concurrent streams, new subscribers, churn rate
- Revenue (subscriptions, ads, transactions)

Quality:

- Video startup time, buffering events, bitrate distribution
- CDN hit ratio, origin traffic

8.2 Logs (ELK Stack)

- **Structured Logging**: JSON format with trace IDs
- **Centralized**: Filebeat → Logstash → Elasticsearch
- **Retention**: 30 days (hot), 90 days (warm), 1 year (cold)
- **Alerting**: ElastAlert for error rate spikes

8.3 Tracing (Jaeger/OpenTelemetry)

- **Instrumentation**: All Go services with OpenTelemetry SDK
- **Sampling**: 100% errors, 1% successful requests
- **Storage**: Elasticsearch backend
- **UI**: Jaeger UI for distributed tracing

8.4 Dashboards (Grafana)

- **CDN Ops**: Hit ratio, bandwidth, top content
- **Platform Health**: Service status, error rates, latencies
- **User Engagement**: MAU, watch time, top genres
- **Revenue**: Subscriptions, ads, PPV by region

9. Deployment Architecture

9.1 Kubernetes Setup

- **Clusters**: 1 per Tier 1 PoP (5 total) + 1 management cluster
- **Namespaces**: `production`, `staging`, `monitoring`
- **Ingress**: Nginx Ingress Controller with TLS termination
- **Service Mesh**: Istio for traffic management, observability

9.2 CI/CD Pipeline

```
Code Push (GitHub) →
  ↓
GitLab CI/CD:
  - Lint, Test, Build
  - Docker Image → Container Registry
  - Helm Chart Update
  ↓
ArgoCD:
  - GitOps Sync
  - Deployment to K8s (blue-green or canary)
  ↓
Smoke Tests → Rollback (if failed)
```

9.3 Infrastructure as Code

- **Terraform**: AWS/GCP resources (VPC, databases, load balancers)
- **Ansible**: Server configuration, ATS setup
- **Helm**: Kubernetes app deployments
- **GitOps**: All config in Git, ArgoCD for sync

10. Disaster Recovery

10.1 RTO & RPO Targets

Service	RTO	RPO	Strategy
API Services	15 min	0	Multi-AZ, instant failover
Database	1 hour	5 min	Point-in-time recovery
Content Library	4 hours	24 hours	S3 cross-region replication
CDN	5 min	0	Multi-PoP, auto-rerouting

10.2 Backup Strategy

- **Databases**: Daily full + 15-min incremental (30-day retention)
- **Content**: 3-2-1 rule (3 copies, 2 media types, 1 offsite)
- **Config**: Git repositories (immutable history)

Document Version: 1.0
Last Updated: 2025-01-08
Architect: Platform Engineering Team