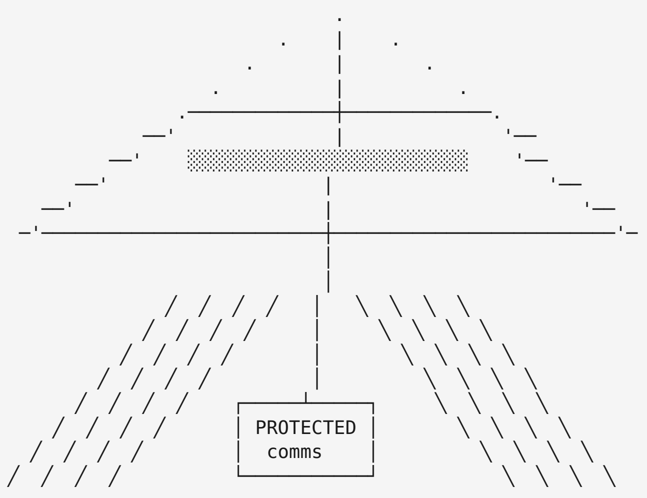
Umbrella — Regulatory Communications Monitoring Platform



1. Overview

Umbrella is a regulatory communications monitoring platform designed to capture, normalize, process, and surface electronic communications (eComm) and audio communications (aComm) for compliance review. The platform ingests data from multiple communication channels, applies NLP and transcription pipelines, and presents flagged alerts to compliance reviewers through a custom UI.

The system follows a **microservices architecture** deployed on **Kubernetes**, ensuring each component can be developed, scaled, and deployed independently.

2. Architecture Diagram

```
┌──────────────────────────────────────────────────────────────────────┐
│                             CONNECTOR LAYER                          │
│                                                                      │
│                                                                    │
│  ┌────────┐   ┌────────┐   ┌────────┐   ┌────────┐   ┌────────┐   ┌────────┐     │
│  │ Teams  │ │ │ Teams  │ │ │ Unigy  │ │ │Bloomberg │ │Bloomberg │ │ Email  │ │     │
│  │ Chat   │ │ │ Calls  │ │ │ Turret │ │ │ Chat   │ │ │ Email  │ │ │ (SMTP/ │ │     │
│  │Connector │ │Connector │ │Connector│ │Connector│ │Connector│ │ IMAP)  │ │     │
│  └────────┘   └────────┘   └────────┘   └────────┘   └────────┘   └────────┘     │
│       │          │           │            │           │           │           │
│       │          │           │            │           │           │           │
│       └──────────┴───────────┴─────┬──────┴───────────┴───────────┘           │
│                                    │                                          │
│                                    │                                          │
│                          ┌─────────────────┐                                  │
│                          │ Connector Plugin │                                 │
│                          │    Framework     │                                 │
│                          └─────────────────┘                                  │
└────────────────────────────────────┬──────────────────────────────────────────┘
                                     │
```

```
                                         │
                                         ▼
┌─────────────────────────────────────────────────────────────────────────┐
│                      INGESTION API                                      │
│                                                                         │
│  ┌───────────────┐    ┌────────────────┐    ┌────────────────┐          │
│  │  REST API     │───▶│  Parser /      │───▶│  Schema        │          │
│  │  Gateway      │    │  Normalizer    │    │  Validation    │          │
│  └───────────────┘    └────────────────┘    └───────┬────────┘          │
│                                                     │                   │
└─────────────────────────────────────────────────────┼───────────────────┘
                                                      │
                         ┌────────────────────────────┤
                         │                            │
                         ▼                            ▼
┌────────────────────────────────┐   ┌────────────────────────────────────┐
│         MESSAGE BUS            │   │          RAW STORAGE               │
│                                │   │                                    │
│  ┌──────────────────────────┐  │   │  ┌──────────────────────────────┐  │
│  │         Apache Kafka     │  │   │  │        S3                    │  │
│  │                          │  │   │  │                              │  │
│  │  topics:                 │  │   │  │  buckets:                    │  │
│  │    raw-messages          │  │   │  │    /raw       – original payloads │  │
│  │    normalized-messages   │  │   │  │    /normalized – normalized records │  │
│  │    processing-results    │  │   │  │    /audio     – audio files  │  │
│  │    alerts                │  │   │  │    /processed – enriched records │  │
│  └──────────────────────────┘  │   │  └──────────────────────────────┘  │
│                                │   │                                    │
└───────────────┬────────────────┘   └────────────────────────────────────┘
                │
                ▼
┌─────────────────────────────────────────────────────────────────────────┐
│                     PROCESSING LAYER                                    │
│                                                                         │
│  ┌───────────────┐  ┌───────────────┐  ┌───────────────┐                │
│  │ Transcription │  │ Translation   │  │ NLP           │                │
│  │ Service       │  │ Service       │  │ Service       │                │
│  │               │  │               │  │               │                │
│  │ • Speech-to-  │  │ • Language    │  │ • Entity      │                │
│  │   text (audio)│  │   detection   │  │   recognition │                │
│  │ • Speaker     │  │ • Multi-lang  │  │ • Sentiment   │                │
│  │   diarization │  │   translation │  │   analysis    │                │
│  │ • Audio format│  │ • Translated  │  │ • Keyword /   │                │
│  │   handling    │  │   text indexing │ │   lexicon match │              │
│  └───────┬───────┘  └───────┬───────┘  │ • Intent      │                │
│          │                  │          │   classification│              │
│          │                  │          │  • Alert      │                │
│          │                  │          │   generation  │                │
│          │                  │          └───────┬───────┘                │
│          └──────────────────┴──────────────────┘                        │
│                                          │                              │
└─────────────────────────────────────────────────────────────────────────┘
                                          │
                                          ▼
┌─────────────────────────────────────────────────────────────────────────┐
│                     SEARCH & STORAGE                                    │
│                                                                         │
│  ┌───────────────────────────────────────────────────────────────────┐  │
│  │                     Elasticsearch Cluster                         │  │
│  │                                                                   │  │
│  │  indices:                                                         │  │
│  │    messages-*        – full normalized + enriched messages        │  │
│  │    alerts-*          – generated alerts with scores and metadata  │  │
│  │    audit-*           – reviewer actions and audit trail           │  │
│  └───────────────────────────────────────────────────────────────────┘  │
│                                                                         │
│  ┌───────────────────────────────────────────────────────────────────┐  │
│  │                         PostgreSQL                                │  │
│  │                                                                   │  │
│  │  tables:                                                          │  │
│  │    users / roles     – reviewer accounts and RBAC                 │  │
│  │    cases             – case management records                    │  │
│  │    review_decisions  – alert dispositions and escalations         │  │
│  │    policies          – lexicons, rules, thresholds                │  │
│  └───────────────────────────────────────────────────────────────────┘  │
│                                                                         │
└─────────────────────────────────────────────────────────────────────────┘
                                              │
                                              ▼
┌─────────────────────────────────────────────────────────────────────────┐
```

```
│                             UI LAYER                                                │
│                                                                                     │
│  ┌────────────────────────┐          ┌─────────────────────────────┐                │
│  │  UI Backend (API)      │◄────────►│  Frontend (SPA)             │                │
│  │                        │          │                             │                │
│  │ • Query Elasticsearch  │          │ • Alert review dashboard    │                │
│  │ • CRUD on PostgreSQL   │          │ • Message search & replay   │                │
│  │ • Auth / RBAC          │          │ • Case management view      │                │
│  │ • Export / reporting   │          │ • Policy configuration      │                │
│  └────────────────────────┘          │ • Audit trail               │                │
│                                      └─────────────────────────────┘                │
│                                               │                                     │
│                                               ▼                                     │
│                                      ┌─────────────────┐                            │
│                                      │  Case Manager   │                            │
│                                      │  (downstream)   │                            │
│                                      └─────────────────┘                            │
└─────────────────────────────────────────────────────────────────────────────────────┘
```

3. Data Flow

```
               ┌───────────┐
               │  Channel  │   (Teams, Bloomberg, Unigy, Email, ...)
               └─────┬─────┘
                     │
                     ▼
               ┌───────────┐
               │ Connector │   Poll / webhook / stream
               └─────┬─────┘
                     │  raw payload
                     ▼
               ┌───────────┐
               │ Ingestion │   Parse, normalize, validate
               │    API    │
               └─────┬─────┘
                     │  normalized message
                ┌────┴────┐
                │         │
                ▼         ▼
          ┌─────────┐  ┌────┐
          │  Kafka  │ │ │ S3 │  Store raw + normalized
          └────┬────┘  └────┘
               │
     ┌─────────┼─────────┐
     ▼         ▼         ▼
┌─────────┐ ┌───────┐ ┌─────┐
│Transcribe│ │Translate│ │ NLP │  Enrich, detect, alert
└────┬────┘ └───┬───┘ └──┬──┘
     │          │        │
     └──────────┼────────┘
                │  enriched message + alerts
                ▼
         ┌──────────────┐
         │Elasticsearch │  Index for search & review
         └──────┬───────┘
                │
                ▼
         ┌──────────────┐
         │   UI Layer   │  Review alerts
         │  (ES + PG)   │
         └──────┬───────┘
                │  escalated alerts
                ▼
         ┌──────────────┐
         │ Case Manager │
         └──────────────┘
```

4. Component Details

4.1 Connector Layer

Each connector is a standalone microservice responsible for interfacing with a single communication channel. Connectors share a common interface defined by the **Connector Plugin Framework**, making it straightforward to add new channels.

Connector	Channel Type	Integration Method	Data Type
Teams Chat	eComm	Microsoft Graph API (webhooks + polling)	Text, attachments
Teams Calls	aComm	Microsoft Graph Call Records API	Audio (WAV/OGG)
Unigy Turret	aComm	Unigy REST/SFTP export	Audio (WAV)
Bloomberg Chat	eComm	Bloomberg SAPI / B-Pipe	Text, attachments
Bloomberg Email	eComm	Bloomberg MSG export / SFTP	Text, attachments
Email (SMTP/IMAP)	eComm	IMAP polling / Journaling endpoint	Text, attachments (EML)

Adding a new connector: Implement the `ConnectorInterface` (ingest, health-check, backfill) and deploy as a new Kubernetes deployment. The framework handles registration, retry logic, and dead-letter routing.

4.2 Ingestion API

A centralized REST API gateway that all connectors push data into. Responsibilities:

- **Authentication** — mTLS between connectors and the API
- **Parsing** — extract structured fields from each channel's raw format
- **Normalization** — map every message to a unified schema: `{ "message_id": string, "channel": enum, "direction": "inbound" | "outbound" | "internal", "timestamp": ISO-8601, "participants": [{ "id", "name", "role" }], "body_text": string | null, "audio_ref": S3 URI | null, "attachments": [{ "name", "type", "s3_uri" }], "metadata": { channel-specific fields } }`
- **Dual write** — publish to Kafka topic `normalized-messages` and persist to S3

4.3 Message Bus (Kafka)

Apache Kafka acts as the central nervous system. Key topics:

Topic	**Producer**	**Consumer(s)**
`raw-messages`	Connectors	Ingestion API
`normalized-messages`	Ingestion API	Processing services, S3 archiver
`processing-results`	Processing services	Elasticsearch indexer
`alerts`	NLP service	Elasticsearch indexer, UI backend

Kafka ensures at-least-once delivery and allows each processing service to consume independently at its own pace.

4.4 Raw Storage (S3)

All data is persisted to S3 for long-term retention and regulatory audit requirements:

- `/raw/` — original payloads as received from connectors
- `/normalized/` — normalized JSON records
- `/audio/` — audio files (calls) referenced by `audio_ref`
- `/processed/` — enriched records post-processing

Lifecycle policies enforce retention periods per regulatory requirements (e.g., 7 years for FINRA/MiFID II).

4.5 Processing Layer

Three independent microservices consume from Kafka and publish enriched results back.

Transcription Service

- Consumes messages with `audio_ref` set
- Downloads audio from S3, runs speech-to-text (e.g., Whisper, Azure Speech)
- Performs speaker diarization to attribute utterances to participants
- Publishes transcript back to `processing-results`

Translation Service

- Detects language of `body_text` or transcript
- Translates non-primary-language content to English (configurable)

- Publishes translated text alongside original to `processing-results`

NLP Service

- **Lexicon / keyword matching** — configurable watchlists (e.g., "guarantee", "off the record", insider terms)
- **Named entity recognition** — people, organizations, securities, monetary values
- **Sentiment analysis** — flag aggressive or unusual tone
- **Intent classification** — detect potential policy violations
- **Alert generation** — score each message against configured policies, generate alerts above threshold
- Publishes enriched records to `processing-results` and alerts to `alerts`

4.6 Search & Storage Layer

Elasticsearch Cluster

- Indexes enriched messages for full-text and structured search
- Stores generated alerts with risk scores, matched policies, and highlighted excerpts
- Maintains audit trail of reviewer actions

PostgreSQL

- Stores application state: user accounts, RBAC, review decisions
- Manages case records and escalation workflows
- Holds policy/lexicon configuration used by the NLP service

4.7 UI Layer

A custom single-page application backed by an API server.

UI Backend (API) - Queries Elasticsearch for alert search, message retrieval, and analytics - CRUD operations on PostgreSQL for cases, decisions, and policies - Handles authentication (SSO/OIDC) and role-based access control - Provides export/reporting endpoints

Frontend (SPA) - **Alert Review Dashboard** — queue of alerts sorted by risk score, filterable by channel, date, policy, reviewer assignment - **Message Search & Replay** — full-text search across all indexed communications, audio playback for calls with synchronized transcript - **Case Management** — group related alerts into cases, add notes, escalate or close - **Policy Configuration** — manage lexicons, rules, and thresholds - **Audit Trail** — full log of who reviewed what and when

Case Manager Integration — alerts escalated from the review UI are forwarded to an external case management system via API or webhook.

---

5. Kubernetes Deployment Model

```
┌──────────────────────────────────────────────────────────┐
│                   Kubernetes Cluster                      │
│                                                          │
│  namespace: umbrella-connectors                         │
│  ┌──────────┐ ┌──────────┐ ┌──────────┐ ┌──────────┐   │
│  │teams-chat│ │teams-call│ │  unigy   │ │bloomberg │   │
│  │ deploy   │ │ deploy   │ │  deploy  │ │  deploy  │   │
│  └──────────┘ └──────────┘ └──────────┘ └──────────┘   │
│                                                          │
│  ┌──────────┐ ┌──────────┐                               │
│  │bb-email  │ │  email   │                               │
│  │ deploy   │ │  deploy  │                               │
│  └──────────┘ └──────────┘                               │
│                                                          │
│  namespace: umbrella-ingestion                          │
│  ┌──────────────────────┐                                │
│  │  ingestion-api       │  (HPA: 2–10 replicas)         │
│  └──────────────────────┘                                │
│                                                          │
│  namespace: umbrella-streaming                          │
│  ┌──────────────────────┐                                │
│  │  kafka (StatefulSet) │  (3 brokers + ZK or KRaft)    │
│  └──────────────────────┘                                │
│                                                          │
│  namespace: umbrella-processing                         │
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐     │
│  │ transcription│ │ translation  │ │      nlp     │     │
│  │  deploy (GPU)│ │   deploy     │ │    deploy    │     │
│  └──────────────┘ └──────────────┘ └──────────────┘     │
│                                                          │
│  namespace: umbrella-storage                            │
│  ┌──────────────────────┐ ┌──────────────────────┐      │
│  │  elasticsearch       │ │  postgresql          │      │
│  │  (StatefulSet, 3+)   │ │  (StatefulSet, HA)   │      │
│  └──────────────────────┘ └──────────────────────┘      │
│                                                          │
│  namespace: umbrella-ui                                 │
│  ┌──────────────────────┐ ┌──────────────────────┐      │
│  │  ui-backend          │ │  ui-frontend (nginx) │      │
│  │  deploy (HPA)        │ │  deploy              │      │
│  └──────────────────────┘ └──────────────────────┘      │
│                                                          │
│  namespace: umbrella-infra                              │
│  ┌──────────┐ ┌──────────┐ ┌──────────┐                │
│  │ ingress  │ │ cert-mgr │ │monitoring│                │
│  │controller│ │          │ │(prom/graf)│               │
│  └──────────┘ └──────────┘ └──────────┘                │
│                                                          │
└──────────────────────────────────────────────────────────┘

        External:  S3 (object storage)
```

---

6. Technology Summary

Layer	Technology
Connectors	Python / Go microservices, channel-specific SDKs
Ingestion API	Go or Python (FastAPI), OpenAPI spec
Message Bus	Apache Kafka (KRaft mode)
Object Storage	S3 (or MinIO for on-prem)
Transcription	OpenAI Whisper / Azure Speech Services
Translation	Azure Translator / Cloud Translate
NLP	spaCy, custom models, lexicon engine
Search	Elasticsearch 8.x
Application DB	PostgreSQL 16
Frontend	React + TypeScript
UI Backend	Node.js (Express/Fastify) or Python (FastAPI)
Orchestration	Kubernetes (EKS / AKS / GKE)
CI/CD	GitHub Actions, Helm, ArgoCD
Observability	Prometheus, Grafana, OpenTelemetry

---

7. Next Steps

1. **Define the normalized message schema** — finalize the canonical data model shared across all services
2. **Scaffold the monorepo** — set up project structure, shared libraries, CI/CD pipelines
3. **Build the connector plugin framework** — define the interface, implement the first connector (e.g., Email)
4. **Stand up Kafka + S3** — deploy the message bus and object storage
5. **Implement the ingestion API** — parser, normalizer, dual-write to Kafka and S3
6. **Build the processing services** — transcription, translation, NLP (can be parallelized)
7. **Deploy Elasticsearch + PostgreSQL** — configure indices and database schema

8. **Build the UI** — alert review dashboard, search, case management
9. **Integrate with case manager** — define the escalation API contract