AYKA Platform - Complete Architecture (Non-Technical Friendly)

System Architecture Diagram

```
┌──────────────────────────────────────────────────────────────────────
│                          👥 USERS (Mobile & Web)
│
│  📱 Mobile App (React Native)         💻 Web App (Next.js)
│  - Click to record conversations      - Click to record conversations
│  - See extracted leads                 - See extracted leads
│  - Browse events                       - Browse events
│  - Contact management                  - Contact management
└──────────────────────────────────────────────────────────────────────
                                │
                                │ Internet (HTTPS - Secure)
                                │
                                ▼
┌──────────────────────────────────────────────────────────────────────
│                      🌐 AWS LOAD BALANCER
│  Purpose: Distributes traffic across multiple servers
│  Why: Handles many users at once, prevents crashes
└──────────────────────────────────────────────────────────────────────
                                │
                    ┌───────────┴────────────┐
                    │                        │
                    ▼                        ▼
            ┌─────────────────┐      ┌─────────────────┐
            │ 🖥️  SERVER 1    │      │ 🖥️  SERVER 2    │
            │ (AWS EC2)       │      │ (AWS EC2)       │
            │                 │      │                 │
            │ FastAPI (Python)│      │ FastAPI (Python)│
            │                 │      │                 │
            │ What it does:   │      │ What it does:   │
            │ ✓ User login     │      │ ✓ User login     │
            │ ✓ Upload audio   │      │ ✓ Upload audio   │
            │ ✓ Fetch leads    │      │ ✓ Fetch leads    │
            │ ✓ Get events     │      │ ✓ Get events     │
            └─────────────────┘      └─────────────────┘
                    │                        │
                    └──────────┬─────────────┘
                               │
                               │ Talks to databases & storage
                               │
             ┌─────────────────┼─────────────────┐
             │                 │                 │
             ▼                 ▼                 ▼
┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐
│ 💾 DATABASE      │  │ ⚡ REDIS         │  │ 📦 S3 STORAGE    │
│ (PostgreSQL)     │    │ (In-Memory)      │    │ (AWS S3)         │
│                  │    │                  │    │                  │
│ Stores:          │    │ Stores:          │    │ Stores:          │
│ ✓ Users           │    │ ✓ Who's logged   │    │ ✓ Audio files   │
│ ✓ Recordings      │    │   in (sessions)│    │ ✓ Exports        │
│ ✓ Transcripts     │    │ ✓ Job queue      │    │                  │
│ ✓ Leads/Contacts│    │ ✓ Cache (fast    │    │ Why separate:    │
│ ✓ Events          │    │   access)        │    │ Audio files      │
│ ✓ User profiles│    │                  │    │ are large,       │
│                  │    │ Why needed:      │    │ keep DB clean    │
│ Why needed:      │    │ Speed! Temp      │    │                  │
│ Permanent data   │    │ data, jobs       │    │                  │
│ storage          │    │                  │    │                  │
└──────────────────┘    └──────────────────┘    └──────────────────┘
                                  │
                                  │ Job Queue (tells GPU what to process)
                                  │
                                  ▼
                        ┌─────────────────────┐
                        │  🏷 CELERY WORKER   │
                        │  (Task Manager)     │
                        │                     │
                        │  What it does:      │
                        │  ✓ Manages jobs      │
                        │  ✓ Sends to GPU      │
                        │  ✓ Tracks progress   │
                        └─────────────────────┘
                                  │
                                  │ Sends audio processing jobs
                                  │
                                  ▼
┌───────────────────────────────────────────────────────────────────────
│                     🎮 GPU SERVER (E2E Cloud)
│                     Why: Heavy AI processing needs powerful GPU
│
│  ┌─────────────────────────────────────────────────────────────────┐
│  │  1❑WHISPER AI (Speech to Text)                                 │
│  │     - Converts audio → text                                     │
│  │     - Processing time: 2-5 minutes for 30-min audio             │
│  │     - Saves to: PostgreSQL (transcripts table)                  │
│  └─────────────────────────────────────────────────────────────────┘
│
│  ┌─────────────────────────────────────────────────────────────────┐
│  │  2❑PYANNOTE AI (Speaker Detection)                             │
│  │     - Identifies who said what                                  │
│  │     - Labels: Speaker 1, Speaker 2, etc.                        │
│  │     - Saves to: PostgreSQL (speakers table)                     │
│  └─────────────────────────────────────────────────────────────────┘
│
│  ┌─────────────────────────────────────────────────────────────────┐
│  │  3❑LEAD GENERATION AGENT (CrewAI + GPT-4)                      │
│  │     - Reads transcript                                          │
│  │     - Finds people mentioned                                    │
│   │     - Extracts: names, companies, roles, needs                  │
│   │     - Searches LinkedIn for profiles                            │
│   │     - Saves to: PostgreSQL (leads table)                        │
│   └─────────────────────────────────────────────────────────────────┘
│
│
│   ┌─────────────────────────────────────────────────────────────────┐
│   │  4□EVENT DISCOVERY AGENT (CrewAI + GPT-4)                       │
│   │     - Reads user profile (interests, location)                  │
│   │     - Searches Google for events                                │
│   │     - Filters by relevance                                      │
│   │     - Ranks: must attend, should attend, nice to attend         │
│   │     - Saves to: PostgreSQL (events table)                       │
│   └─────────────────────────────────────────────────────────────────┘
│
│
│   ┌─────────────────────────────────────────────────────────────────┐
│   │  5□EMAIL AGENT (CrewAI + GPT-4)                                 │
│   │     - Creates beautiful HTML email                              │
│   │     - Includes leads + events                                   │
│   │     - Sends via Gmail SMTP                                      │
│   │     - Saves log to: PostgreSQL (email_logs table)               │
│   └─────────────────────────────────────────────────────────────────┘
│
│
│  Total Processing Time: 3-8 minutes per recording
└──────────────────────────────────────────────────────────────────────
```

Why Each Component? (Simple Explanations)

1. Why 2 Servers (FastAPI on EC2)?

- **Reason**: If one server crashes, the other keeps working
- **Benefit**: Website stays online 24/7
- **Analogy**: Like having 2 cashiers at a store - if one is busy, the other serves customers

2. Why PostgreSQL Database?

- **Reason**: Store all user data permanently
- **What's stored**: Users, recordings, leads, events, everything
- **Analogy**: Like a filing cabinet that never loses data

3. Why Redis?

- **Reason 1**: Super fast temporary storage (100x faster than PostgreSQL)
- **Reason 2**: Manages job queue (who's next in line for processing)
- **Reason 3**: Remembers who's logged in
- **Analogy**: Like sticky notes for quick reminders vs. filing cabinet for permanent records

4. Why S3 Storage?

- **Reason**: Audio files are HUGE (30-min recording = 50-100MB)
- **Benefit**: Don't clog the database with large files

- **Analogy**: Like a warehouse for bulky items vs. office desk for documents

5. Why Separate GPU Server?

- **Reason**: AI processing needs powerful hardware (expensive)
- **Benefit**: Only pay for GPU when processing, not 24/7
- **Cost**: GPU = $200/month, Regular server = $30/month
- **Analogy**: Like renting a bulldozer only when you need it vs. owning a car

6. Why Load Balancer?

- **Reason**: Distributes users across servers evenly
- **Benefit**: No single server gets overloaded
- **Analogy**: Like a traffic cop directing cars to less busy lanes

---

Complete Data Flow (Step by Step)

```
1. 👤 User opens app/website
   ↓
2. 🔐 User logs in
   → Server checks PostgreSQL (users table)
   → Redis stores session (you're logged in)
   ↓
3. 🎙️ User clicks "Record" → Audio captured
   ↓
4. 📤 Audio uploaded
   → Sent to S3 (audio storage)
   → Metadata saved in PostgreSQL (recordings table)
   → Job created in Redis queue
   ↓
5. 🎮 GPU server picks up job
   ↓
6. 🗣️ Whisper converts speech → text
   → Saved to PostgreSQL (transcripts table)
   ↓
7. 👥 Pyannote identifies speakers
   → Saved to PostgreSQL (speakers table)
   ↓
8.   Lead Agent analyzes transcript
   → Finds people, companies, opportunities
   → Searches LinkedIn
   → Saved to PostgreSQL (leads table)
   ↓
9. 📅 Event Agent searches for events
   → Finds relevant conferences/meetups
   → Ranks by relevance
   → Saved to PostgreSQL (events table)
   ↓
10. 📧 Email Agent sends summary
    → HTML email with leads + events
    → Saved to PostgreSQL (email_logs table)
    ↓
11. ✓ User sees results
    → API fetches from PostgreSQL
    → Redis caches for fast access
    → Displayed on app/website
```

Optional: Graph Database (Phase 2)

Neo4j - Why Later?

What it does: Connects people by shared interests

Example:

- You like: AI, Crypto, Startups
- John likes: AI, Crypto
- Sarah likes: AI, Startups
- **Neo4j finds**: "You should meet John and Sarah - you have common interests!"

Why not now: Adds complexity, not needed for MVP

When to add: When you have 1000+ users and want smart matching

---

Technology Choices - Why These?

Backend: FastAPI (Python)

- **Why**: Fast, modern, easy to add AI
- **Alternative**: Node.js/Express (slower for AI tasks)

Frontend: Next.js

- **Why**: Works on all devices, SEO-friendly
- **Alternative**: Plain React (no SEO)

Mobile: React Native

- **Why**: Write once, works on iOS + Android
- **Alternative**: Native apps (need 2 codebases)

Database: PostgreSQL

- **Why**: Reliable, handles millions of records, free
- **Alternative**: MongoDB (less structure, not ideal for our use case)

GPU: E2E Cloud

- **Why**: Cheaper than AWS GPU (50% less)
- **Alternative**: AWS GPU (2x cost)

Cost Breakdown (Monthly)

Component	Why Needed	Cost
2× API Servers (EC2)	Handle user requests, always online	$60
PostgreSQL Database	Store all data permanently	$20
Redis Cache	Fast access, job queue	$15
S3 Storage	Store audio files	$10
GPU Server (E2E)	AI processing (Whisper, Agents)	$200
Load Balancer	Distribute traffic	$20
TOTAL		**$325/month**

For comparison:

- Hiring 1 person to manually process recordings: $3000+/month
- Our AI does it automatically: $325/month

Scalability - What Happens When You Grow?

10 Users

- 1 API server
- Small database
- **Cost**: $150/month

100 Users

- 2 API servers
- Medium database
- **Cost**: $325/month (current plan)

1,000 Users

- 5 API servers (auto-scaling)
- 2 GPU servers
- Larger database
- **Cost**: $800/month

10,000 Users

- 20+ API servers
- 5+ GPU servers
- Database replicas
- **Cost**: $3,000-5,000/month
- **Revenue needed**: $10,000+/month (break even at $10/user)

Security & Privacy

How Data is Protected

1. **Encryption in Transit**: All data encrypted during transfer (HTTPS)
2. **Encryption at Rest**: Database encrypted on disk
3. **Access Control**: Only you see your data
4. **Backups**: Daily automatic backups (7 days)
5. **Audio Storage**: Can be auto-deleted after processing (GDPR compliant)

Who Can Access What?

- **Users**: Only their own data
- **Admins**: Only with permission (audit logged)
- **AI Agents**: No permanent storage, process and delete

Summary - Why This Architecture?

✓ **Reliable**: If one server fails, others keep working ✓ **Fast**: Redis caching, optimized queries ✓ **Scalable**: Add more servers as users grow ✓ **Cost-effective**: Only pay for GPU when processing ✓ **Secure**: Industry-standard encryption and access control ✓ **Maintainable**: Modular design, easy to update

Bottom Line: Professional, production-ready system that can handle growth from 10 to 10,000 users.