CoreChain Federated Learning System: Complete Architecture

Tech Stack Overview

Backend & Infrastructure

Component	Technology	Version	Purpose
Aggregator	Python	3.10	Coordinates FL training & blockchain management
FL Framework	Flower	1.6.0	Orchestration of the federated learning rounds
Blockchain	Ganache	-	Immutable storage for model hashes & metadata
Web3	Web3.py	-	Blockchain interaction layer
Communication	gRPC	-	High-performance client-server communication
Containerization	Docker	-	Service isolation and deployment

Machine Learning & Data

Component	Technology	Details
Framework	TensorFlow 2.x	Model training and inference engine
Model	Custom CNN	Specialized architecture for X-ray TB detection
Dataset	Shenzhen TB	662 chest X-ray images (80/20 train/test split)
Preprocessing	OpenCV, NumPy	Image normalization and resizing (224x224)

System Architecture

1. Cloud Infrastructure (AWS EC2 Aggregator)

The central hub hosted on AWS handles the heavy lifting of aggregation and blockchain verification.

- **CoreChain-Aggregator Container:** Houses the Aggregator gRPC service (Port 50051), Flower Server (Port 8080), and Ganache Blockchain.
- **Process Management:** Handled via **Supervisord** for high availability.

2. Edge Nodes (Hospital Nodes)

Local hospital environments where the actual data resides.

- **Hospital-FL-Test Container:** Runs the Hospital Node (gRPC), Flower Client, and a **FastAPI** Dashboard (Port 3000).
- **Security:** Patient data remains inside this container; only model weights are exported.

Core Workflow & Data Flow

Phase 1: Initialization

1. **Hospital:** Starts the FastAPI dashboard, loads the Shenzhen dataset from `/data`, and initializes the local CNN.
2. **Aggregator:** Starts the Ganache blockchain and the Flower server, waiting for `min_available_clients` (default: 1) to connect.

Phase 2: Training Round (10 Rounds Total)

1. **Distribution:** The Aggregator selects clients and sends the global model weights via gRPC.
2. **Local Training:** The Hospital trains the model for 5 epochs on local X-rays.
3. **Aggregation:** The Aggregator collects updates and uses the **FedAvg (Federated Averaging)** algorithm to update the global model.
   $$W_{next} = \frac{\sum (w_{i} \times n_{i})}{\sum n_{i}}$$
   Where $w$ represents client weights and $n$ represents the number of samples.
4. **Blockchain Logging:** A SHA-256 hash of the new global parameters, the round number, and accuracy metrics are stored in the **ModelRegistry** smart contract.

Data Storage & Persistence

Entity	Storage Type	Data Stored
Hospital	In-memory / PNG	Raw X-ray images, local training metrics, current weights

Entity	**Storage Type**	**Data Stored**
Aggregator	In-memory / Log	Global model weights, Flower training logs
Blockchain	Ganache DB	Immutable transaction hashes, Model metadata, Audit trails

Model Architecture: Custom CNN

The system utilizes a deep Convolutional Neural Network optimized for grayscale X-ray analysis:

- **Input Layer:** (224, 224, 1)
- **Feature Extraction:** 5 blocks of `Conv2D` + `BatchNorm` + `ReLU` + `MaxPool`.
- **Classification:** `GlobalAvgPool → Dense(256) → Dense(128) → Dense(1, Sigmoid)`.
- **Total Size:** ~56.7 MB (56 weight arrays).

Deployment Command Summary

Aggregator (AWS)

Bash
```
docker run -d --name corechain-aggregator \
  -p 50051:50051 -p 8080:8080 \
  -e MIN_CLIENTS=1 -e FL_ROUNDS=10 \
  corechain-aggregator:latest
```

Hospital Node (Local)

Bash
```
docker run -d --name hospital-fl-test \
  -p 3000:3000 -v /path/to/data:/data \
  -e AGGREGATOR_ADDRESS=54.173.119.88:50051 \
  corechain-hospital:dashboard
```

Security & Privacy Highlights

- **Privacy:** Raw data never leaves the hospital; only non-reversible weight gradients are shared.
- **Integrity:** Blockchain provides a tamper-proof audit trail of the model's evolution.
- **Efficiency:** gRPC keepalives (10s pings) ensure stable connections over unstable hospital networks.