# Architecture Discussion: Technical Deep Dive

**Dev 1:**

If we go hybrid, we need to define clear boundaries. Node.js handles blockchain RPC calls, transaction signing, and event subscriptions. FastAPI handles AI-related endpoints like prompt embeddings and lightweight inference.

**Dev 2:**

Right. For communication, I'd suggest REST initially, but we should abstract that behind an internal client. That way, we can switch to gRPC later without breaking contracts.

**Dev 1:**

Agreed. For Node, I'll set up an Express server with TypeScript. We'll modularize the blockchain logic: one service for wallet management, one for contract interactions, and one for listening to event streams via WebSockets.

**Dev 2:**

On the FastAPI side, I'll create a microservice with routes for `/preprocess`, `/embed`, and `/inference`. For embeddings, we can hook into sentence-transformers initially, but the design should allow swapping in OpenAI or HuggingFace models.

**Dev 1:**

How do we handle state? We'll need a shared database for persistent data like user profiles and transaction history. Postgres makes sense, but Redis can help with caching blockchain call results to reduce latency.

**Dev 2:**

Yes, Redis for short-lived data like mempool lookups or inference cache. For inter-service communication, we could also use Redis Pub/Sub or RabbitMQ if async queues become necessary.

**Dev 1:**

Good call. Another thing—security. For Node, all wallet interactions must use environment-stored private keys with strict ACLs. We should isolate signing operations in a dedicated module, maybe even an HSM later.

**Dev 2:**

And for FastAPI, we'll enforce authentication with JWTs. All inference and preprocessing calls need to be gated by user tokens, verified against the Node service as the source of truth.

**Dev 1:**

Deployment plan? Each service containerized with Docker. We can orchestrate with Docker Compose at first. If we need to scale horizontally, migrate to Kubernetes with separate pods for Node and FastAPI.

**Dev 2:**

Exactly. Monitoring is another piece—we should instrument both with Prometheus metrics. Node service can track RPC latency and failed transactions, FastAPI can track request throughput and inference latency.

**Dev 1:**

For CI/CD, GitHub Actions pipelines: linting, unit tests, Docker build, and push to registry. Then deploy with versioned tags. Rollbacks are as simple as re-tagging a previous image.

### Dev 2:

Finally, documentation—we'll maintain an OpenAPI spec for both services. That way devs know exactly how to interact with each service regardless of language stack.

### Dev 1:

Perfect. That gives us modularity, scalability, and clear ownership between blockchain and AI logic.