EdgeChain System Flow - Bird's Eye View

A concise guide to how components work together for privacy-preserving federated learning on Midnight Network.

□ User Stories & Flows

Story 1: Deploy the Smart Contract

Actor: Developer **Goal**: Deploy EdgeChain contract to Midnight Testnet

```
Developer → deploy-simple.ts → Midnight Network
   ↓
1. Generate wallet seed
2. Wait for tDUST funding
3. Instantiate contract with witness
4. Deploy to blockchain
   ↓
Result: Contract address saved to deployment.json
```

Key Components: - **`deploy-simple.ts`** - Deployment orchestrator - `WalletBuilder.buildFromSeed()` - Creates wallet from hex seed - `Contract(witnesses)` - Instantiates with witness functions - `deployContract()` - Submits to Midnight Network - `farmerSecretKey()` witness - Generates ZK proof of identity

---

Story 2: Farmer Submits Model Update

Actor: Data Farmer (Lace wallet user) **Goal**: Submit locally trained model weights privately

```
Farmer → UI → Lace Wallet → Smart Contract → Public Ledger
   ↓
1. Train model locally (client-side TensorFlow.js)
2. Hash model weights
3. Sign transaction with Lace
4. Submit via submitModelUpdate() circuit
5. ZK proof generated (proof server)
   ↓
Result: Model hash recorded, identity private
```

Key Components:

UI Layer (`packages/ui/src/`): - **`App.tsx`** - Main application router - **`ContractProvider.tsx`** - Manages contract state - `useLaceWallet()` - Connects Lace wallet - `submitModelUpdate()` - Submits to contract - **`midnight.ts`** - Midnight SDK integration - `indexerPublicDataProvider()` - Query blockchain - `walletProvider()` - Sign transactions

Smart Contract (`packages/contract/src/edgechain.compact`): - **Circuit**: `submitModelUpdate()` - Input: `modelHash: Bytes<32>`, `farmerSecretKey` (witness) - Action: Increment `submissionCount`, store hash - Privacy: Uses ZK proof to hide farmer identity

Ledger State:

```
export type Ledger = {
  currentRound: Uint64,
  submissionCount: Uint64,
  currentModelVersion: Uint64,
  globalModelHash: Bytes<32>,
  isAggregating: boolean
};
```

Story 3: View Contract State

Actor: Anyone **Goal**: See current federated learning progress

```
User → view-contract.ts → Indexer → Display
   ↓
1. Query contract address
2. Fetch public ledger state
3. Decode with contract module
   ↓
Result: Shows round, submissions, aggregation status
```

Key Components: - **`view-contract.ts`** - State viewer - `indexerPublicDataProvider()` - Connect to indexer - `queryContractState()` - Fetch ledger data - `EdgeChainModule.ledger()` - Decode state - Displays: `currentRound`, `submissionCount`, `isAggregating`

□ System Architecture

```
┌──────────────────────────────────────────────────────────────┐
│                      USER INTERFACE                          │
│  packages/ui/src/                                            │
│                                                            │
│  ┌─────────────┐   ┌──────────────┐   ┌──────────────┐     │
│  │   App.tsx   │→ │ Contract     │→ │ Lace Wallet  │     │
│  │  (Router)   │  │ Provider     │  │ Integration  │     │
│  └─────────────┘   └──────────────┘   └──────────────┘     │
└──────────────────────────────────────────────────────────────┘
                         │
                         ↓
┌──────────────────────────────────────────────────────────────┐
│                   MIDNIGHT SDK LAYER                         │
│  @midnight-ntwrk/*                                           │
│                                                            │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐      │
│  │   Wallet     │  │  Indexer     │  │ Proof        │      │
│  │   Provider   │  │  Provider    │  │ Provider     │      │
│  └──────────────┘  └──────────────┘  └──────────────┘      │
└──────────────────────────────────────────────────────────────┘
                         │
                         ↓
┌──────────────────────────────────────────────────────────────┐
│                  MIDNIGHT NETWORK                            │
│                                                            │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐      │
│  │   RPC Node   │  │   Indexer    │  │ Blockchain   │      │
│  │ (Testnet)    │  │ (GraphQL)    │  │ (Ledger)     │      │
│  └──────────────┘  └──────────────┘  └──────────────┘     │
└──────────────────────────────────────────────────────────────┘
                         ↑
┌──────────────────────────────────────────────────────────────┐
│              SMART CONTRACT (On-Chain)                       │
│  dist/managed/edgechain/contract/index.cjs                   │
│  ┌────────────────────────────────────────────────────────┐ │
│  │  Ledger State: { currentRound, submissionCount, ... }│ │
│  │  Circuits: submitModelUpdate(), aggregateModels()     │ │
│  │  Privacy: ZK proofs hide farmer identities            │ │
│  └────────────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────────┘
                         ↑
┌──────────────────────────────────────────────────────────────┐
│           ZERO-KNOWLEDGE PROOF SERVER                        │
│  Docker: midnightnetwork/proof-server:latest                 │
│  - Generates ZK proofs for circuits                          │
│  - Runs locally on port 6300                                 │
│  - Required for all contract transactions                    │
└──────────────────────────────────────────────────────────────┘
```

□ End-to-End Flow: Submit Model Update

Step-by-Step Component Interaction

```
┌──────────────┐
│ 1. UI Layer  │ Farmer clicks "Submit Model"
└──────┬───────┘
       ↓
  App.tsx:submitModel()
       ↓
  ContractProvider.tsx:submitModelUpdate()
       ↓
  midnight.ts:prepareTransaction()
       ↓
┌──────────────┐
│ 2. Wallet    │ Lace Wallet prompts for signature
└──────┬───────┘
       ↓
  User approves in Lace extension
       ↓
  walletProvider.sign(transaction)
       ↓
┌──────────────┐
│ 3. ZK Proof  │ Generate proof for submitModelUpdate circuit
└──────┬───────┘
       ↓
  proofProvider.prove({
    circuit: "submitModelUpdate",
    witness: { farmerSecretKey: <hidden> },
    public: { modelHash: <hash> }
  })
       ↓
  [Proof Server: 30-60 seconds]
       ↓
  Proof generated □
       ↓
┌──────────────┐
```

```
│ 4. Network    │ Submit transaction to Midnight
└───────┬───────┘
        ↓
   rpcProvider.submitTransaction({
     proof: <zkproof>,
     transaction: <signed_tx>
   })
        ↓
   Transaction included in block
        ↓
┌───────────────┐
│ 5. Contract   │ Execute submitModelUpdate circuit
└───────┬───────┘
        ↓
   edgechain.compact:submitModelUpdate()
   - Verify ZK proof □
   - Update ledger.submissionCount++
   - Store modelHash
   - Keep farmer identity private
        ↓
   State updated on blockchain
        ↓
┌───────────────┐
│ 6. Indexer    │ Index new state
└───────┬───────┘
        ↓
   Indexer detects state change
   Updates GraphQL API
        ↓
┌───────────────┐
│ 7. UI Update  │ Display new state
└───────┬───────┘
        ↓
   indexerProvider.queryContractState()
   ContractProvider updates React state
   UI shows: "Submission count: 1"
        ↓
   □ COMPLETE
```

□ Key Code Components Reference

Deployment (`packages/contract/src/deploy-simple.ts`)

```typescript
// 1. Create wallet
const wallet = await WalletBuilder.buildFromSeed(
  indexerURL, indexerWSURL, proofServerURL, nodeURL,
  walletSeed, networkId, "info"
);

// 2. Instantiate contract with witness
const witnesses = {
  farmerSecretKey: () => {
    const secretKey = new Uint8Array(32);
    crypto.getRandomValues(secretKey);
    return secretKey;
  }
};
const contractInstance = new EdgeChainModule.Contract(witnesses);
```

```
// 3. Deploy
const deployedContract = await deployContract(
  providers,
  contractInstance
);
```

Contract State Query (`packages/contract/src/view-contract.ts`)

```
// 1. Connect to indexer
const publicDataProvider = indexerPublicDataProvider(
  indexerURL, indexerWS
);

// 2. Query state
const state = await publicDataProvider.queryContractState(
  contractAddress
);

// 3. Decode
const ledger = EdgeChainModule.ledger(state.data);
console.log(`Current Round: ${ledger.currentRound}`);
console.log(`Submissions: ${ledger.submissionCount}`);
```

UI Contract Integration (`packages/ui/src/providers/ContractProvider.tsx`)

```
// 1. Connect wallet
const { connect, address } = useLaceWallet();

// 2. Create providers
const walletProvider = {
  coinPublicKey: wallet.coinPublicKey,
  encryptionPublicKey: wallet.encryptionPublicKey,
  // ... sign function
};

// 3. Submit to contract
const submitModelUpdate = async (modelHash: Uint8Array) => {
  const witnesses = { farmerSecretKey: generateKey() };
  const tx = await contract.submitModelUpdate(
    modelHash,
    witnesses
  );
  await walletProvider.sign(tx);
};
```

Smart Contract (`packages/contract/src/edgechain.compact`)

```
// Ledger definition
export type Ledger = {
  currentRound: Uint64,
  submissionCount: Uint64,
  currentModelVersion: Uint64,
  globalModelHash: Bytes<32>,
  isAggregating: boolean
};
```

```
// Witness (private input)
witness farmerSecretKey(): Bytes<32>;

// Circuit (public transaction)
export circuit submitModelUpdate(
  modelHash: Bytes<32>
): Void {
  // Update state
  state.submissionCount = state.submissionCount + 1;

  // Privacy: farmerSecretKey is never revealed
  // Only ZK proof that farmer knows the key
}
```

□ Data Flow Summary

Public Data (Everyone can see)

- Contract address
- Current round number
- Total submission count
- Global model hash
- Aggregation status

Private Data (Hidden by ZK proofs)

- Farmer identity
- Individual model weights
- Farmer secret keys
- Transaction details

Witnessed Data (Known only to submitter)

- `farmerSecretKey` - Proves identity without revealing it
- Model training data
- Local model parameters

□ Critical Configuration Files

Network Config (All scripts use these):

```
const TESTNET_CONFIG = {
  indexer: "https://indexer.testnet-02.midnight.network/api/v1/graphql",
  indexerWS: "wss://indexer.testnet-02.midnight.network/api/v1/graphql/ws",
  node: "https://rpc.testnet-02.midnight.network",
  proofServer: "http://127.0.0.1:6300"
};
```

Fly.io Config (`packages/ui/fly.toml`):

```
app = 'edgechain-midnight-ui'
primary_region = 'iad'
```

```
[http_service]
  internal_port = 8080

[[http_service.headers]]
  for = "/*"
  [http_service.headers.values]
    Cross-Origin-Embedder-Policy = "require-corp"  # For WASM
    Cross-Origin-Opener-Policy = "same-origin"     # For SharedArrayBuffer
```

Quick Reference: Where Things Happen

Function	File	Line/Section
Deploy contract	`deploy-simple.ts`	Lines 90-250
View contract state	`view-contract.ts`	Lines 70-95
Submit model update	`edgechain.compact`	Lines 30-50
Connect Lace wallet	`ContractProvider.tsx`	Lines 40-80
ZK proof generation	Proof server (Docker)	Port 6300
Query blockchain	`midnight.ts`	indexerPublicDataProvider
Sign transactions	Lace Wallet Extension	Browser

Key Takeaways

3 Main Components: 1. **Smart Contract** (Compact) - Business logic, privacy via ZK proofs 2. **UI** (React + Midnight SDK) - User interface, wallet integration 3. **Infrastructure** (Proof Server + Midnight Network) - ZK proof generation, blockchain

Flow Pattern:

```
User Action → UI → Wallet Signature → Proof Generation →
Network Submission → Contract Execution → State Update →
Indexer → UI Refresh
```

Privacy Mechanism: - Witness functions hide sensitive data - ZK proofs verify without revealing - Only public state visible on-chain - Farmer identities never exposed

---

Last Updated: November 8, 2025 **Checkpoint**: stable-v1.0 **Complete Code**: Available in repository