LFDT and Hiero

An overview of the Linux Foundation Decentralized Trust and the Hiero Ledger project

The Linux Foundation Decentralized Trust (LFDT)

- A trust operated by the Linux Foundation
- Focus is on decentralized technology
- https://www.lfdecentralizedtrust.org/
  - 50+ projects, 169M+ lines of code
  - Ledger, Cryptography, Identity, Interoperability
  - Linux Foundation Events
  - LFDT Events

Hiero Ledger

- A Decentralized Ledger
- Joined the LFDT in September 2024, Graduated August 2025
- Web links
  - https://www.lfdecentralizedtrust.org/projects/hiero
  - https://github.com/hiero-ledger
  - https://www.lfdecentralizedtrust.org/blog/tag/hiero
- Started with donation (code and financial) from Hedera
- Most active contributors and maintainers employed by Hashgraph and Limechain
- Hundreds of individual contributors
- Over 30 active repositories on Github

Hiero Improvement Proposals (HIPs)

- Most Hiero Ledger changes require a HIP
- Markdown files managed in github, approved by the TSC
  - Most are also adopted by Hedera, but this is up to the Hedera Council
- Anyone with a github account can
  - Fork the repo and send a Pull Request for a new HIP
  - Participate in discussions and comment on PRs.
- HIPs are a good place to start learning and contributing

Hiero Consensus Node

- The software that operates the ledger
  - Runs the Hashgraph consensus algorithm
  - Executes transactions
  - Accepts new transactions and adds them to events
  - Maintains the state of the ledger
  - Produces the Block Stream
- Written entirely in Java, with minimal dependencies
  - Every individual node independently executes and verifies all transactions
  - One library written in Rust (HiNTS/TSS Cryptography)
- Continuously tested
  - 24 hour performance test baseline exceeds 10,000 TPS
  - 24 hour quality/acceptance test
  - 168 hour extended quality/acceptance test
  - CI testing on every PR runs in roughly 30 minutes
- Monthly release cadence
  - Releases are pipelined
  - 4 weeks development, roughly 8-10 weeks testing per release
  - Testing includes a public testnet
  - Some features are released in preview several months in advance

Hiero Block Node

- A new project under development
- Stores Block Streams, which will replace Record Streams
  - Fully decentralized, each block carries a full ZK proof of validity
    - Full State Proof support
  - Much more efficient than record streams
- Accepts the Block Stream from the Consensus Node
  - Passes the low-latency stream to subscribers
  - Verifies the proof on every block
  - Stores all blocks "forever"
  - Maintains a copy of network state
    - Validates state
    - Generates state snapshots
    - Offers State Proofs
- Plugin-based, written in Java
  - Anyone can run a block node, and may choose which subset of functionality to support
  - Block Nodes may offer value-added services

Hiero Mirror Node

- Indexes the network history
- Provides multiple query APIs
  - gRPC
  - GraphQL
  - ReST
- Written in Java/SpringBoot
- Uses a sharded form of PostrgreSQL
- Hedera runs 2 public instances
- Reads Record Streams from cloud buckets
  - Not decentralized enough, but have worked since OA in 2019
  - Inefficient, and does not support clean state proofs
- Will read Block Streams from Block Nodes "soon"

Hiero SDKs and TCK

- SDKs are available in most common (and some uncommon) languages
- The core API (called "HAPI") is defined in Protocol Buffers and gRPC
  - Highly efficient
  - Clear and easy to understand
  - Wide language support
- JSON-RPC Relay and GRPC-Web servers offer gateways for clients that cannot easily use gRPC and Protocol Buffers.
- The TCK is written mostly in Javascript
  - Tests substantially the entire HAPI
  - Part of the continuous testing for the Consensus Node
- Consensus node endpoints and GRPC-Web endpoints are managed on-chain.
  - There is a HIP to start to enable adding Mirror Nodes, Block Nodes, and JSON-RPC Relay nodes on-chain (HIP-1137).

Solo

- A New project to make deploying Hiero nodes (Consensus, Mirror, Block, JSON-RPC Relay, etc...) nodes simple and easy
- Supports development, testing, and production operations
- Based on single-node Kubernetes and many related technologies
- Written mostly in Go