EVERNODE

Seriously smart contracts for the XRPL Ecosystem

by

Evernode Labs Pty Ltd

20 Nov 2023

Whitepaper 2.0

Table of Contents

1. EXECUTIVE SUMMARY

1.1 Evernode is a global, permissionless, decentralised "layer 2" network tailored to hosting hyper-flexible, hyper-scalable dApps that perform as bespoke mini-blockchains ("AppChains"), empowering developers to build with their choice of language, functionality, geography, and scale, but without needing to invent their own consensus mechanism.

Evernode's Five Components

1.2 Evernode has five main components:

(a) **Consensus:** A consensus-as-an-operating-system environment in which code runs to enable multiple instances to function as a bespoke chain.

(b) **Hosting:** A way to deploy multiple instances of an executable program to a global network of independently owned and operated hosts.

(c) **Decentralised Network:** A way of coordinating the network via a layer 1 chain so the network is as secure and decentralised as possible.

(d) **Native Token**: A new digital asset for the network to incentivise hosts and facilitate automated payment for network registration and hosting fees.

(e) **Decentralised Governance:** A way for hosts to vote to update the rules for registration, rewards, and governance, including deregistering bad hosts.

Hosts

1.3 In concert, these five components create a decentralised network where anybody can become a Host by downloading the Evernode software and paying the registration fee in Evers. Reliable hosts earn network rewards for being connected to the network and earn income in the network's currency from hosting dApps.

Developers

1.4 For Developers, these five components result in a global network of independently owned and operated Hosts on which they can build and deploy dApps with their choice of language, behaviours, geography, and scale. In conjunction with Hooks, Xahau's lite smart contract solution, a vast new range of use cases for decentralised applications arises.

Initiator

1.5 Evernode is an initiative of Evernode Labs Pty Ltd, a private Australian company spun out of the Australian National University to commercialise IP arising from research funded by grants from Ripple's University Blockchain Research Initiative.

Proposed Launch

1.6 Evernode plans to launch fully functional on 18th December, or as soon as possible thereafter, depending on the existence of a wallet that fully supports the Xahau Network and its XRPL account-cloning feature.

2. HOTPOCKET – CONSENSUS-AS-AN-OPERATING-SYSTEM

2.1 The inspiration for Evernode was this: the XRP Ledger is a payment App inside a consensus engine, so what if you could take the payment App out and insert any other type of App as desired into the consensus engine? Thus, was born the consensus engine we call HotPocket.

2.2 HotPocket uses a cooperative UNL-based consensus algorithm akin to the Ripple Consensus Protocol, abstracted for any transaction or input type. It transforms standard Apps into dApps.

2.3 We call them dApps, but really, they are bespoke mini-blockchains (or "AppChains") that behave as the developer wants. Its UNL, the files it reads and writes, its inputs and outputs etc are all subject to consensus within the dApp's UNL. HotPocket provides all this as a sort of "consensus-as-an-operating-system" solution as summarised in Figure 1 below.

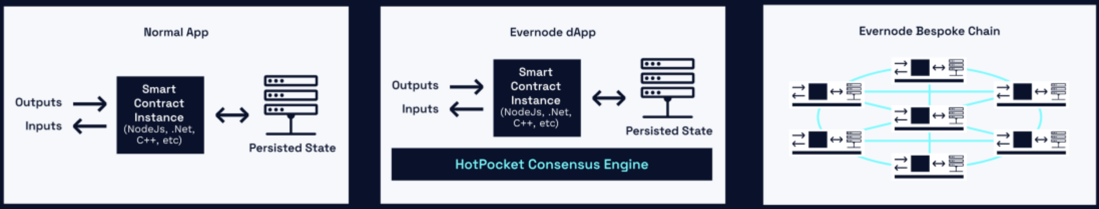

Normal App
Outputs
Inputs
Smart Contract Instance (NodeJs, .Net, C++, etc)
Persisted State
Evernode dApp
Outputs
Inputs
Smart Contract Instance (NodeJs, .Net, C++, etc)
Persisted State
HotPocket Consensus Engine
Evernode Bespoke Chain

Figure 1: Evolution of a HotPocket Cluster

2.4 HotPocket has four main components:

(a) **UNL Consensus Protocol:** HotPocket is a Unique Node List (UNL) based consensus protocol that allows multiple Linux machines to become a mini-blockchain by enforcing consensus rules on inputs and outputs and maintaining a shared, canonical state across multiple instances.

(b) **Rapid State Sync:** HotPocket contains a bundle of additional features designed to make it as easy as possible to spin up a HotPocket node and sync it to the existing network.

(c) **Contract Lifecycle Management:** HotPocket ensures all instances of a contract have the same configuration and allows contracts to be upgraded automatically at consensus.

(d) **Minimal Setup:** HotPocket enables new nodes to join an existing contract with minimal known information to sync the new node with the cluster.

UNL Consensus Protocol

2.5 The HotPocket consensus engine works as follows:

(a) **Set Up:** The programmer configures a set of servers and builds a Unique Node List of these servers' public keys and a peer list of IPs and ports. This configuration is copied to all servers in the contract's network.

(b) **Synchronisation:** During execution, each HotPocket instance connects to its peers and synchronizes the current contract state across all UNL peers according to a configurable consensus threshold of between 50% and 100% of nodes, depending on the desired trade-off between liveness and security. Once state transfer and synchronization are achieved, the contract's network collects inputs from the contract's users.

(c) **Inputs from Users:** Users can connect to any node not explicitly configured to reject their connections. Users identify themselves by proving ownership of a public key and their inputs are then circulated into the consensus mechanism, akin to transactions being circulated into the Bitcoin, XRPL, or Ethereum networks.

(d) **Consensus on Inputs:** The contract's nodes then execute a consensus round deciding which user inputs made it into this block and which will be held off for next block. Other essential consensus information, like the time of the round, the current contract state, and the identity of the last closed ledger (the canonical state of the network) also enter into consensus. After three rounds, consensus is reached by the dev-configured majority of UNL peers. Non-UNL peers can also observe consensus if the contract network is configured as a public network.

(e) **Execution**: Upon consensus, each HotPocket node simultaneously executes the smart contract binary and provides the same set of user inputs in the same canonical order to the binary. The smart contract processes the batch of user inputs and produces a set of contract outputs. These outputs are of two forms: updates to the contract's state and user outputs (to be sent back to users).

(f) **Node Party Line & Sub-Consensus Feature:** During smart contract execution, the UNL nodes may communicate with each other over a broadcast service provided by HotPocket. This is called the Node Party Line. This feature allows nodes to run sub-consensus agreement and information sharing before exiting and alleviates the need for each node to behave completely deterministically. For example, the nodes may wish to pass a multi-sig transaction between themselves and each sign with a key only that individual node possesses, before agreeing on a canonical final signed multi-sig transaction and exiting.

(g) **User Outputs:** Once the smart contract has executed, another consensus round takes place to ensure all nodes produced the same output and the same changes to their state. Once the result of the execution is agreed upon, the user outputs are passed to the users for whom they were destined according to the contract's internal programming or dropped if the user is no longer connected anywhere on the contract's network. In practice, this new consensus round also gathers up a new batch of user inputs to feed into the contract's next execution.

2.6 The above features provide a robust and proven byzantine-fault-tolerant consensus mechanism that is agnostic as to the nature of the App inside it.

Rapid State Syncing

2.7 To enable new nodes to quickly catch-up state, HotPocket uses a state management system akin to BitTorrent.

2.8 Each contract execution allows the contract binary to read and write into its state folder which is in fact a mounted FUSE device managed by a HotPocket sub-process. Each time the contract writes to, or alters, its on-disk state the FUSE sub-process updates its Merkel tree representation of the contract's state. The deltas between Merkel trees are then computed to allow efficient transfer of only the changed data between two arbitrarily distant states in the ledger chain.

2.9 This means nodes do not need to replay old ledgers to catch up state, which would likely be otherwise impossible for high throughput contracts.

Contract Config Sync

2.10 HotPocket has two features to help elegantly manage contract lifecycles.

(a) **Online Configuration:** First, the contract config is subjected to consensus ensuring all contract nodes use the same configuration which effects deterministic execution. Contracts can also update their own configuration at runtime and rely on consensus to ensure all the nodes retain the same configuration.

(b) **Self-Editable Contracts:** Second, HotPocket smart contracts can be configured (if desired) to "live" among the contract data, subjecting the contract binaries and upgrade activities to consensus. HotPocket offers a handoff mechanism to perform contract upgrades in between consensus rounds by means of an installation shell script provided by the contract. The results of the upgrade are automatically validated in subsequent consensus rounds.

2.11 Combined, these two contract management features mean HotPocket offers a rich administrative environment in which the contract can self-manage its own life cycle.

3. SASHIMONO – DECENTRALISED HOSTING

3.1 Despite HotPocket's utility, it would be a relatively centralised solution if developers had to spin up each of the machines on which their dApps run. Instead, there should be a global marketplace of independently owned and operated Hosts running software capable of hosting HotPocket dApps. This network of dApp Hosts is the problem Sashimono solves.

3.2 Sashimono is a hosting daemon. It provides the ability to deploy an instance of a HotPocket dApp to a shared Linux hosting environment and have the dApp instance run without interfering with any other dApp instances running on the same host. This makes Sashimono a multi-tenant cloud hosting environment specialized in HotPocket dApp deployment.

Anatomy Of A DApp Instance

3.3 Sashimono packages the HotPocket and dApp executables into a container image and runs them with the help of a security-hardened container manager.

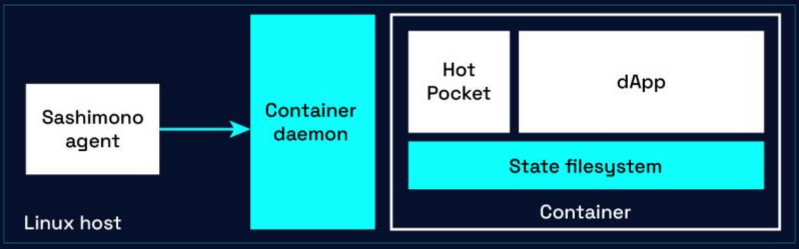


Figure 2: Sashimono 1st Design Stage

3.4 In Figure 2 above, the dApp box essentially contains untrusted code. Bundling the dApp and dependencies into a secure container helps protect the rest of the system from malicious dApps.

State Filesystem

3.5 HotPocket uses a **FUSE** filesystem layer to help manage the dApp state as shown in Figure 3. This provides for necessary HotPocket-specific features such as checkpointing and rapid state syncing.

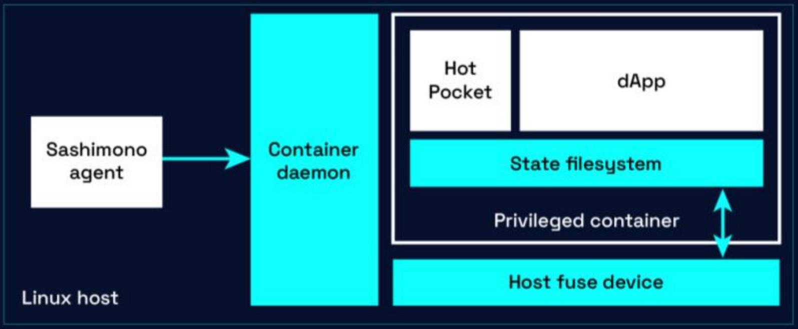


Figure 3: Sashimono 2nd Design Stage

3.6 Granting containers direct access to the Host's native FUSE device may become a security risk, so Sashimono maintains the state filesystem in an outer sandbox alongside the container s shown in figure 4.

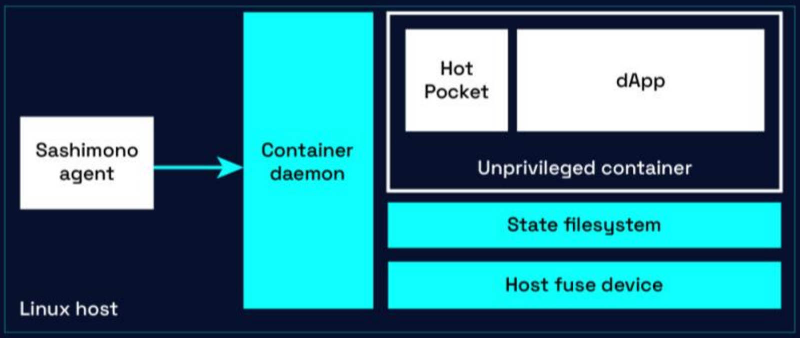


Figure 4: Sashimono 3rd Design Stage

3.7 In this manner the container still has access to the dApp sate filesystem without requiring access to the FUSE device on the host. However, privileged containers are a security risk.

3.8 Since the state filesystem is outside the container, container privileged mode can now be removed, and the state filesystem can directly access the FUSE device on the host.

HotPocket vs dApp

3.9 Within the container, HotPocket and the dApp are running at the same capability level. So, a malicious dApp can affect the execution of HotPocket and perform unintended operations. We solve this problem by having the dApp execute under an unprivileged user account within the container as depicted in Figure 5 below.

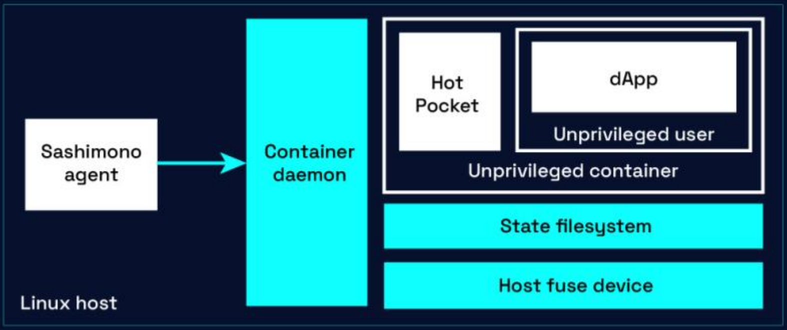


Figure 5: Sashimono 4th Design Stage

3.10 Since the dApp is executing under a least-privileged user account within the container, it cannot affect HotPocket execution or perform anything outside its user permission boundary. We now have two barriers to defend against a malicious dApp. To perform an attack on the host system, the dApp must break out of its unprivileged user account inside the container, and then also break out of the unprivileged container itself.

Multi-Tenant Instance/Container Management

3.11 However, for Evernode to work as a network each Host must be capable of hosting separate instances of multiple dApps without those dApps interfering with the Host or each other. To achieve this, we implement the following set-up, where multiple dApp instances run as containers managed by a container daemon as shown in Figure 6 below.

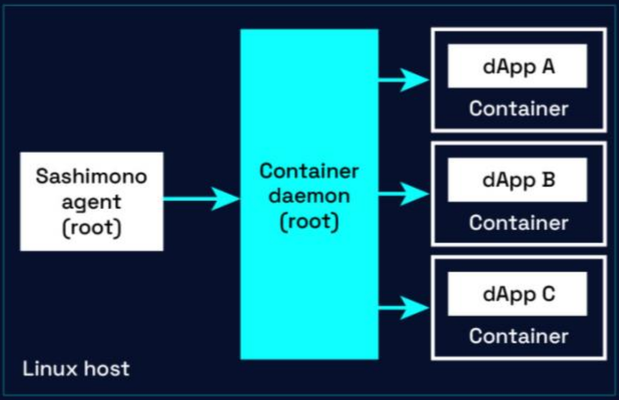


Figure 6: Sashimono 5th Design Stage

Sashimono – Container Daemon Separation

3.12 However, if a malicious dApp escapes the container barriers, it could affect the operation of the Sashimono agent, because the container daemon and the Sashimono agent run in the same privilege level (root). To solve this, we move all container management activities to an unprivileged user account as shown in Figure 7 below.

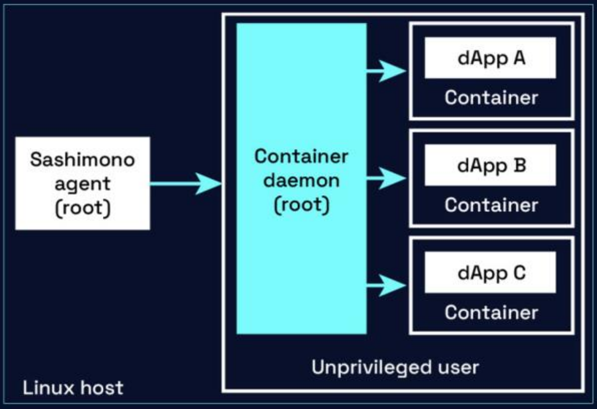


Figure 7: Sashimono 6th Design Stage

3.13 Since all dApp containers and the container daemon are running under an unprivileged user account, they cannot affect Sashimono or the host itself.

Multi-Tenant Separation

3.14 A weakness of the setup in Figure 7 is that even though all dApps are isolated from the critical components of the system, they can still interfere with each other. A malicious dApp which breaks the container defences will be able to compromise ALL the other dApp instances (tenants) running on the same host. Therefore, we allocate dedicated Linux user accounts for each dApp tenant as shown below in Figure 8.

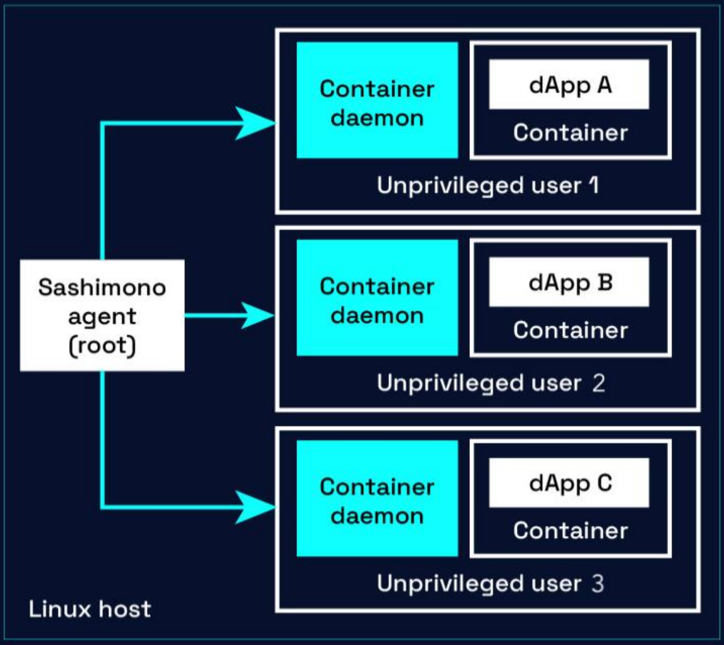


Figure 8: Sashimono Finalised Design

3.15 With this final setup, each dApp and its container management environment gets its own unprivileged user account, significantly increasing per-tenant isolation and system security. The final multi-tenant separation design has 3 barriers to prevent malicious code breaking out, is able to rely on strong user account security provided by the Linux operating system, can restrict resource allocation (CPU, RAM, Disk space) with Linux user quotas, and allows for the clean creation and destruction of tenants.

3.16 This set-up is not without trade-offs. It involves a higher per-tenant container daemon overhead, container images must be duplicated as common container images cannot be shared among tenants, and container management upgrades are more cumbersome as they need to be performed on a per-tenant basis.

4. XAHAU REGISTRY HOOK - DECENTRALISED NETWORK

4.1 Sashimono enables a decentralised network of hosts specialising in hosting HotPocket dApps. A Sashimono cluster thus looks something as shown below in Figure 9.

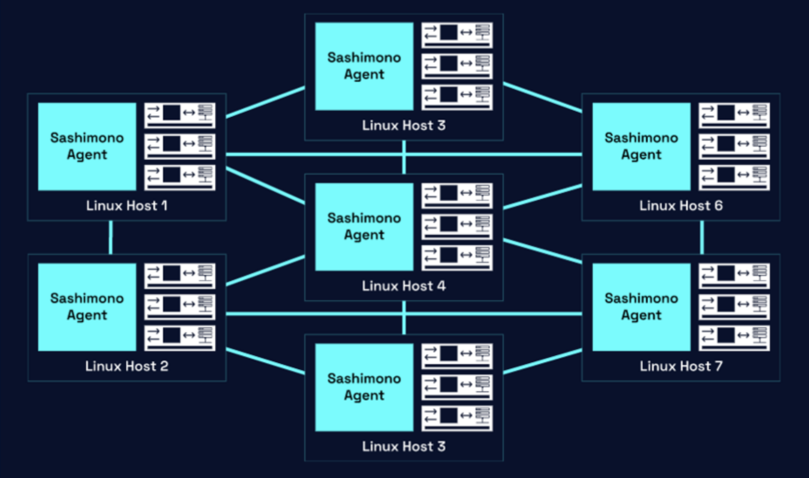


Figure 9: A Sashimono Network

4.2 But for the network to be useable, there must exist a register of Hosts so people can know which machines comprise the network. If this register is centralised, then Evernode risks being centralised with a single source of failure. The problem of making Evernode's registry of Hosts decentralised is solved via a Registry Hook on the Xahau Network.

The Xahau Network & Hooks

4.3 The Xahau Network (Xahau) is the new smart contract sidechain for the XRPL ecosystem. It's whitepaper describes it as follows:

> *"...a code-fork of the XRP Ledger's (XRPL's) open-source rippled codebase. It embodies all the useful and innovative features of the XRPL, including its speed and low transaction costs, but tweaks and upgrades them to support smart contracts."*

4.4 Xahau implements Hooks, a smart contract technology developed specifically for the XRPL ecosystem. The Xahau whitepaper describes Hooks as follows:

> *"Hooks are small, efficient pieces of code defined on a Xahau account that execute logic on transactions sent to or received by the account before those transactions are finalised in the ledger. Hooks are thus a way for developers to create and deploy smart contracts on Xahau, opening a wide range of possibilities for decentralized applications (dApps) and automated transactions."*

Making Evernode Decentralised with Xahau Hooks

4.5 So, Hooks are a way of automating transactions received by and emitted from Xahau Accounts. Evernode uses a Hook set on a Xahau Account to automate its canonical registry of Evernode Hosts.

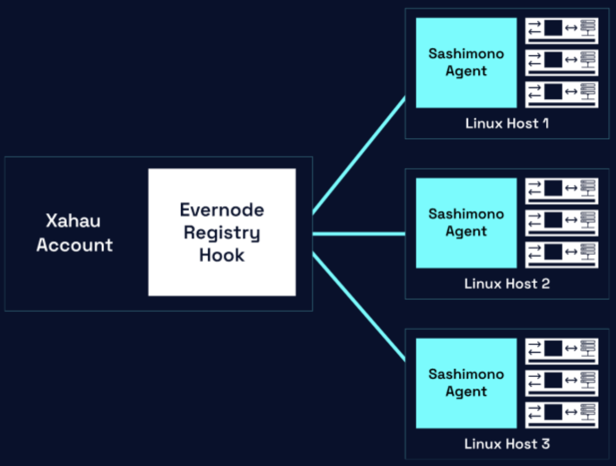
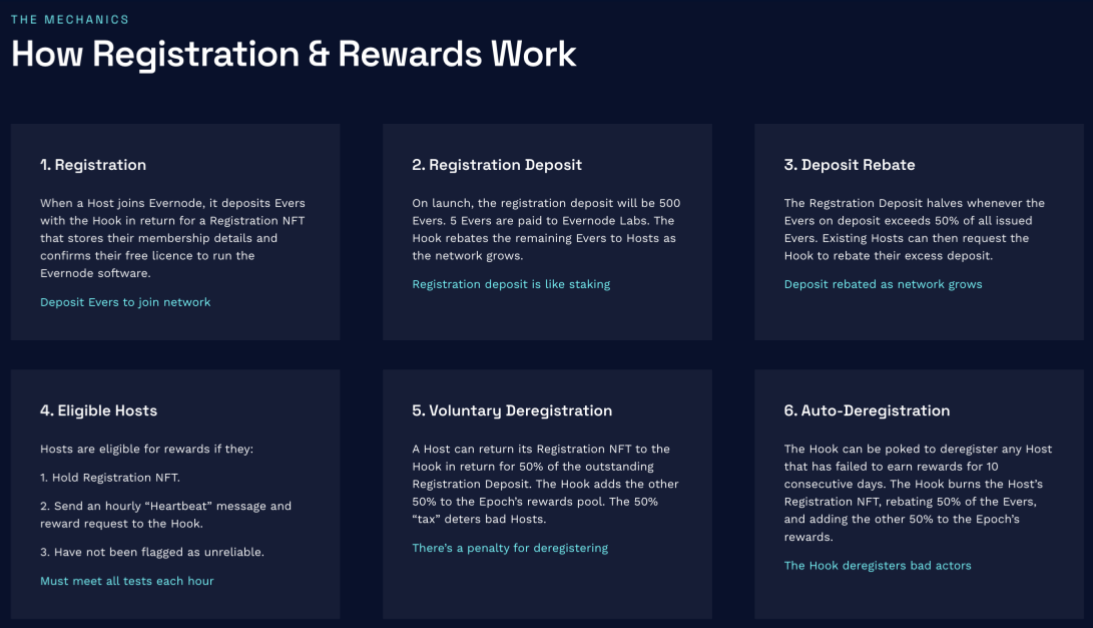


Figure 10: Xahau Registry Hook

4.6 The mechanics for registration and deregistration of Hosts on the network relies on the Hook auto-issuing and redeeming Registration NFTs on the Xahau Network in exchange for deposits of Evers. It works as outlined in Figure 11:



THE MECHANICS

How Registration & Rewards Work

1. Registration

When a Host joins Evernode, it deposits Evers with the Hook in return for a Registration NFT that stores their membership details and confirms their free licence to run the Evernode software.

Deposit Evers to join network

2. Registration Deposit

On launch, the registration deposit will be 500 Evers. 5 Evers are paid to Evernode Labs. The Hook rebates the remaining Evers to Hosts as the network grows.

Registration deposit is like staking

3. Deposit Rebate

The Regstration Deposit halves whenever the Evers on deposit exceeds 50% of all issued Evers. Existing Hosts can then request the Hook to rebate their excess deposit.

Deposit rebated as network grows

4. Eligible Hosts

Hosts are eligible for rewards if they:

1. Hold Registration NFT.

2. Send an hourly "Heartbeat" message and reward request to the Hook.

3. Have not been flagged as unreliable.

Must meet all tests each hour

5. Voluntary Deregistration

A Host can return its Registration NFT to the Hook in return for 50% of the outstanding Registration Deposit. The Hook adds the other 50% to the Epoch's rewards pool. The 50% "tax" deters bad Hosts.

There's a penalty for deregistering

6. Auto-Deregistration

The Hook can be poked to deregister any Host that has failed to earn rewards for 10 consecutive days. The Hook burns the Host's Registration NFT, rebating 50% of the Evers, and adding the other 50% to the Epoch's rewards.

The Hook deregisters bad actors

Figure 11: Registration & Rewards Mechanics

Ensuring Quality Hosts

4.7 It is important to maintain Host quality in a decentralised way. At launch, this will be achieved in several ways:

(a) **The Carrot:** To earn network rewards, Hosts must send a "heartbeat" message to the Hook every hour. A Host that fails to send a heartbeat is classified "Inactive". Inactive Hosts are ineligible to earn rewards.

(b) **The Stick:** 80% of Hosts can vote to deregister bad actors by forcibly redeeming their Registration NFT. The penalty for unilateral deregistration is that the Hook will rebate only 50% of their Registration Deposit and the other 50% to that Epoch's rewards.

(c) **The Bigger Stick:** Anybody can send a "prune" message to the Registration Hook. In response, the Registration Hook will unilaterally redeem/burn the Registration NFT of any Host that has not sent a heartbeat in the last 10 consecutive days.

(d) **The Auditors:** We will use bounties to incentivise the emergence of third-party "Audit" services that use our Host Audit Tool – a tool for confirming that a Host is properly configured to accept new dApp instances onto its "Slots" – to provide Host ranking services to dApps, developers, and the Evernode community. Immediately after launch, while the network is in Piloted mode, Evernode Labs will perform this role.

4.8 Further improvements to these measures will be explored and implemented after the network is live and its real behaviour known.

Why Not Be a "Layer 1" Network?

4.9 The way Evernode is designed it could have been constructed as a stand-alone chain to manage its network registry and native currency.

4.10 We chose to build Evernode as a layer 2 solution composed via another chain because of benefits that come from being within an existing ecosystem. Those benefits include:

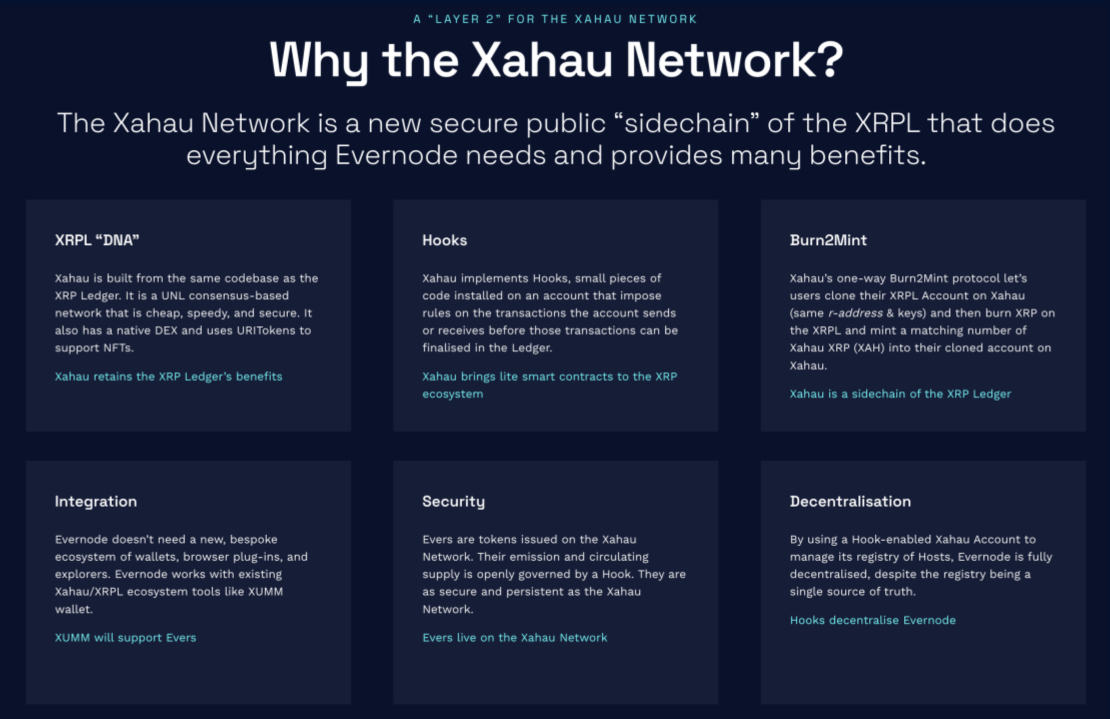
(a) **Integration:** By issuing Evers as a token on a layer 1 chain we avoid the need to build an independent ecosystem of wallets, browser plug-ins, and explorers. Evernode will interoperate with existing Xahau tools, like Xumm. Exchanges that support Xahau's native token (XAH) can easily support Evers.

(b) **No dUNL:** By issuing Evers on a layer 1 chain, we avoid the need for a separate Evernode network with a separate dUNL and all the complications that come with specifying, identifying, and incentivising a decentralised dUNL.

Why Choose the Xahau Network?

4.11 The way Evernode is designed it could have been "nailed" to any layer 1 chain with sufficient smart contract capabilities to issue tokens and function as a host registry.

4.12 We chose to develop Evernode on the Xahau Network because Xahau is fundamentally the XRP Ledger but with the Hooks lite-smart-contract amendment added as summarised in Figure 12 below.



A "LAYER 2" FOR THE XAHAU NETWORK

Why the Xahau Network?

The Xahau Network is a new secure public "sidechain" of the XRPL that does everything Evernode needs and provides many benefits.

XRPL "DNA"

Xahau is built from the same codebase as the XRP Ledger. It is a UNL consensus-based network that is cheap, speedy, and secure. It also has a native DEX and uses URITokens to support NFTs.

Xahau retains the XRP Ledger's benefits

Hooks

Xahau implements Hooks, small pieces of code installed on an account that impose rules on the transactions the account sends or receives before those transactions can be finalised in the Ledger.

Xahau brings lite smart contracts to the XRP ecosystem

Burn2Mint

Xahau's one-way Burn2Mint protocol let's users clone their XRPL Account on Xahau (same *r-address* & keys) and then burn XRP on the XRPL and mint a matching number of Xahau XRP (XAH) into their cloned account on Xahau.

Xahau is a sidechain of the XRP Ledger

Integration

Evernode doesn't need a new, bespoke ecosystem of wallets, browser plug-ins, and explorers. Evernode works with existing Xahau/XRPL ecosystem tools like XUMM wallet.

XUMM will support Evers

Security

Evers are tokens issued on the Xahau Network. Their emission and circulating supply is openly governed by a Hook. They are as secure and persistent as the Xahau Network.

Evers live on the Xahau Network

Decentralisation

By using a Hook-enabled Xahau Account to manage its registry of Hosts, Evernode is fully decentralised, despite the registry being a single source of truth.

Hooks decentralise Evernode

Figure 12: Benefits of Xahau Network

Why Not the XRP Ledger?

4.13 In initial conception and development, Evernode was intended to be launched on the XRP Ledger. At the time, it was expected Hooks would be adopted as an amendment to the XRP Ledger. In our initial Whitepaper, we noted that without Hooks we would have to pivot to a new chain.

4.14 As it became increasingly apparent that Hooks would not be adopted by the XRP Ledger in the foreseeable future, we experimented in development with a version of Evernode that did not rely on Hooks, where a multi-sig XRPL Account acted as the registry of Hosts and treasury of Evers.

4.15 This option proved unviable for two reasons:

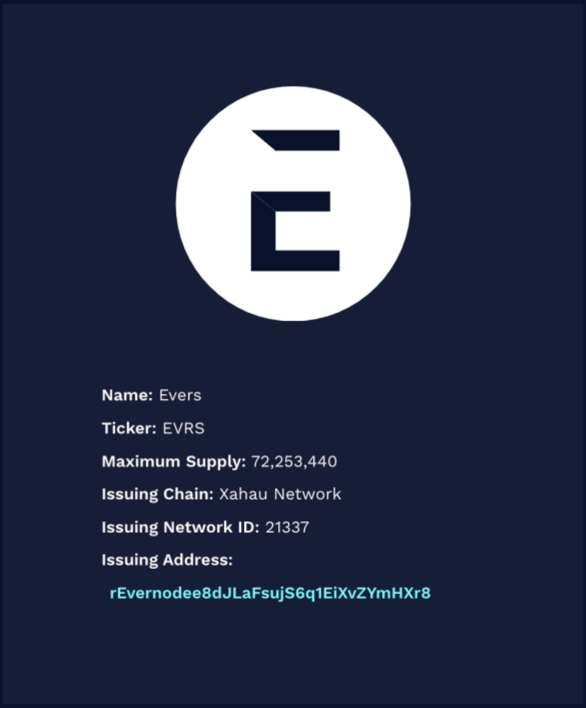
(a) **Finalisation of Transactions:** First, badly formed registration requests (for example, the wrong number of Evers provided as a deposit) are still valid XRPL transactions and would still enter the ledger. They would have to be reversed by a subsequent transaction, leading to reputational risks and potential attack vectors. Instead, with Hooks, the Hook can reject a badly formed transaction and that failed transaction simply never enters the ledger: there's nothing to reverse and Evernode can't be accused of wrongly taking people's Evers and giving nothing in return.

(b) **Centralised Treasury:** The second problem is that Evers could not be auto-distributed as rewards. Those rewards would be controlled by a multi-sig account, and that account would be controlled by whoever held the keys. Significant, and ultimately unacceptable, security and regulatory complications arise from a limited number of people controlling the treasury of a blockchain project. Nobody wanted to be the custodian of the project's Evers.

4.16 In the end, it was deemed infeasible to launch on the XRPL as intended. Thankfully, Xahau Network was launched which from Evernode's perspective was the XRPL codebase but with Hooks implemented.

5. EVERS - EVERNODE'S NATIVE DIGITAL CURRENCY

5.1 Evernode uses a new digital currency, called Evers (see Figure 13 for details).



Name: Evers

Ticker: EVRS

Maximum Supply: 72,253,440

Issuing Chain: Xahau Network

Issuing Network ID: 21337

Issuing Address:

rEvernodee8dJLaFsujS6q1EiXvZYmHXr8

Figure 13: Evers Details

Holding Evers

5.2 Evers will be issued on the Xahau Network. Any Xahau-compatible wallet will hold Evers. Since Xahau is a code fork of the XRP Ledger, most wallets that support XRP will find it easy enough to also support Xahau (and therefore Evers).

Using Evers

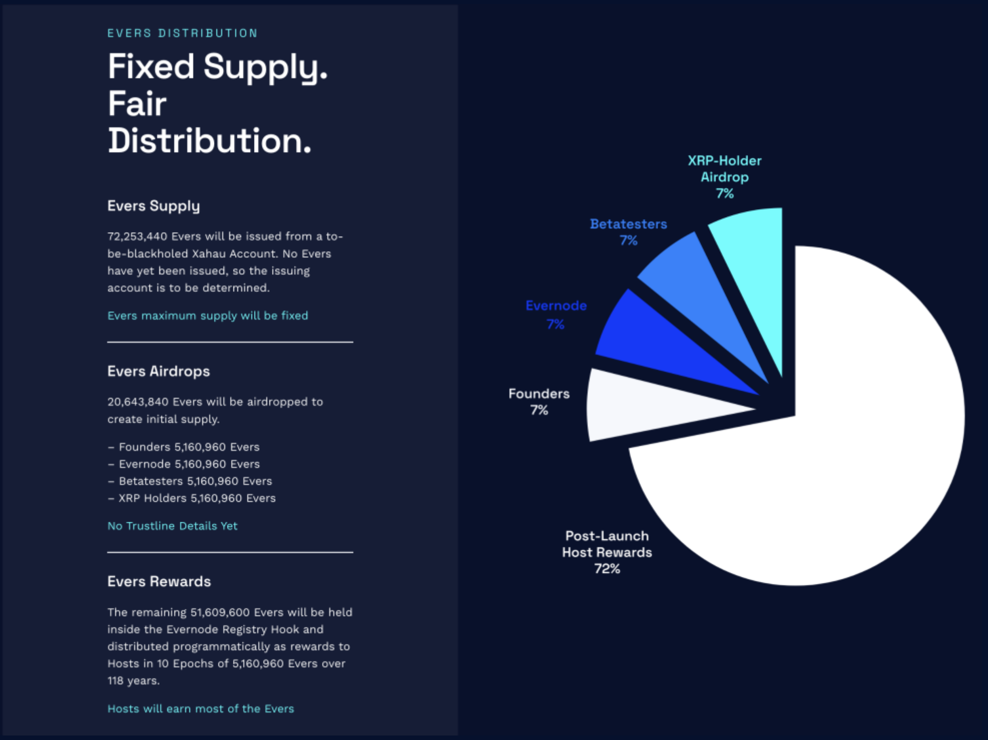
5.3 All services on Evernode will be priced and paid for in Evers. Hosts will need Evers to pay for registration on the network and Tenants (dApps) will need Evers to pay for hosting.

Acquiring Evers

5.4 At launch Evers will trade on Xahau's native DEX. Other exchanges may subsequently choose to list Evers, but since Xahau has a DEX, the protocol doesn't need exchanges to list the token for people to acquire it.

Evers Distribution – Fixed Supply, Fairly Distributed

5.5 No pre-sale or ICO is planned. The protocol will be launched fully-functioning without pooling of any external funds for development, other than grant funds. Evers will be either gifted via discretionary airdrops or distributed by a Hook as rewards for running a reliable Host. This is summarised in Figure 14 below.



EVERS DISTRIBUTION

Fixed Supply. Fair Distribution.

Evers Supply

72,253,440 Evers will be issued from a to-be-blackholed Xahau Account. No Evers have yet been issued, so the issuing account is to be determined.

Evers maximum supply will be fixed

Evers Airdrops

20,643,840 Evers will be airdropped to create initial supply.

- Founders 5,160,960 Evers
- Evernode 5,160,960 Evers
- Betatesters 5,160,960 Evers
- XRP Holders 5,160,960 Evers

No Trustline Details Yet

Evers Rewards

The remaining 51,609,600 Evers will be held inside the Evernode Registry Hook and distributed programmatically as rewards to Hosts in 10 Epochs of 5,160,960 Evers over 118 years.

Hosts will earn most of the Evers

Figure 14: Evers Distribution Model

5.6 Details of the eligibility and process for claiming airdrops will be announced prior to launch.

Why Have a Native Currency?

5.7 Pricing network services in Evers allows developers to automate their dApps' buying and selling of hosting services from Evernode Hosts. Without a native digital currency (and the related features of Lease NFTs) this process would be prohibitively cumbersome or hopelessly centralised.

5.8 Evers will be instantly useful as a means of exchange for hosting services on Evernode, regardless of whether Evers have any value outside the network. In the long run, the price (if any) of Evers should reflect the perceived value (if any) of the decentralised dApp hosting services Evernode provides.

Why Not Use XRP or XAH?

5.9 During development, we experimented with building Evernode without a native currency. We explored the option of network registration and fees being paid in the layer 1's native currency (XRP or XAH). This did not work for several reasons.

(a) **No Host Rewards:** First, it would mean we had no mechanism for incentivising Hosts. There could be no rewards for being a member of the network because the protocol had no rewards to give, and acquiring those rewards to give away for free would be prohibitively expensive. Further, the potential tax implications would be complicated, existentially so in some scenarios.

(b) **Constant Re-Pricing:** Second, it would have made the problem of pricing network services almost impossible. It would be possible-but-impractical for Hosts to adjust what they charge in rent based upon current value of XRP, but it would be much harder for the protocol itself to adjust the registration fee. That would require a range of oracles and other complicated mechanisms all of which can be gamed and lead to undesirable outcomes.

5.10 By having a native currency that is distributed in a fixed and decentralised manner by a smart contract/Hook we avoid all these problems. In particular, we can set the registration fee at a specific number of Evers and let the market work out what that fee should cost. Having a native asset became a no brainer because it solved so many problems.

6. EVERNODE'S ON-CHAIN GOVERNANCE GAME

6.1 Since Evernode uses Hooks on the Xahau Network, there must exist a mechanism for those Hooks to be updated in the future. This function is handled by a third Hook that implements Evernode's Governance Game.

6.2 The Governance Game allows eligible participants in the Evernode network to propose and vote on new Hooks. These proposals will get accepted or purged according to a predetermined rule-set on received votes.

Participants

6.3 There are two classes of participants in the Governance Game.

(a) **Evernode Labs**: Evernode Labs always has 1 vote and has special rights when the Hooks are in Piloted and Co-Piloted modes, but no special rights when the Hooks are in Auto-Piloted mode.

(b) **Valid Hosts**: Accounts that hold a Registration NFT for the previous 3 continuous months and are not eligible to be pruned due to unreliability have 1 vote each.

Types of Proposals

6.4 Participants can submit three types of proposals:

(a) **Proposal for a New Hook Candidate:** A proposal for a new hash for Evernode's three Hooks.

(b) **Proposal for removing a Dud Host:** A proposal for the Registration Hook to forcibly redeem the Registration NFT of an allegedly dud Host.

(c) **Proposal for changing the governance mode:** A proposal to change the governing mode of the Hooks between Piloted, Co-Piloted, and Auto-Piloted modes.

6.5 To propose a change to any Hook, the Proposer must present the hash of all three Hooks that would apply if the proposal were successful. Any Participant can submit a Proposal for a new Hook. The Proposer must collateralize their Proposal with an amount of Evers equivalent to the current Moment's reward quota, according to the Evernode Reward Schedule. The Hooks which bear the proposed hashes must be deployed to some existing Xahau account.

6.6 A Dud host removal Proposal nominates the Xahau Address of the malfunctioning host to be removed from the platform. Any Participant can submit this kind of proposal. Proposer must collateralize their Proposal with Evers rewards worth 25% of the current Moment's reward quota.

6.7 The rules for submitting a proposal to change the Governance Mode are set out below.

Withdrawing a Proposal

6.8 The Proposer can withdraw their Proposal at any time before it Succeeds or Purges. If the Proposal is withdrawn, the proposer gets half their Evers back. Lost Evers are added to that Epoch's reward pool.

Purging a Proposal

6.9 If a Proposal has not Succeeded three months after being proposed, it will be purged. If a Proposal expires, the Proposer loses all their staked Evers. Lost Evers are added to that Epoch's reward pool.

Voting

6.10 Hosts can make their choice of voting via Evernode-CLI. The Participant's vote is captured via their heartbeat, which is managed by the Evernode software installed on the host. They either Support or Reject a Proposal, with Reject being the default. Support is a positive vote for the Proposal.

Electing a Proposal

6.11 A Proposal succeeds if it is continuously Supported by at least 80% of possible Participants for 2 weeks. If a Proposal for a new Hook Candidate succeeds all other existing Proposals for that Hook are Purged and their staked Evers are added to the Epoch's reward pool. For any proposal, the Proposer gets all their staked Evers back.

Evernode Labs Special Rights

6.12 As the licensor of the Evernode software and the developer of the three Hooks, Evernode Labs has agreed to provide a limited, ongoing role to guard against catastrophic failure, reflected in some special privileges for its Xahau Account within the Governance Game.

6.13 First, a Xahau Account controlled by Evernode Labs will be always eligible to vote. This means even if there are no other eligible Participants for any reason (such as the Registration Hook becomes corrupted and all Registration NFTs become invalid), Evernode Labs will at least be able to vote to recover and restart the Hooks.

6.14 Second, Evernode Lab's vote carries special weight depending upon the mode of the Governance Game.

Governance Mode – 3 Levels of Decentralisation

6.15 The Governance Game has three modes:

(a) **Piloted**: Evernode Labs vote determines the outcome of all Proposals.

(b) **Co-Piloted**: No Proposal can succeed unless Evernode Labs Supports it.

(c) **Auto-Piloted**: The standard voting rules apply with Evernode Labs being treated equally with any other Participant.

6.16 This provides a transparent and flexible framework for monitoring the performance of the Hooks and ensuring everything is running smoothly before transitioning to a protocol entirely controlled by participants. In particular:

(a) **Launch In Piloted Mode:** At launch, governance will be in Piloted mode so Evernode Labs can quickly address any catastrophic Hook failures.

(b) **Moving to Co-Pilot Mode:** The governance mode moves from Piloted to Co-piloted at the election of Evernode Labs. It is intended this step would be taken as soon as it becomes clear the Hooks are functioning properly and some catastrophic bug is unlikely to emerge immediately post-launch.

(c) **Moving to Auto-Pilot Mode**: If the governance mode is Co-Piloted, it becomes Auto-Piloted, again, at the election of Evernode Labs. At this point Evernode Labs becomes just another participant with no special rights except the persistence of its 1 vote.

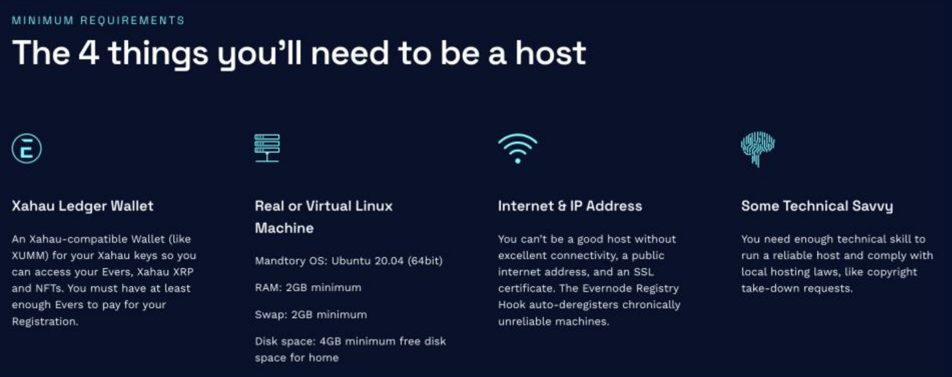
(d) **Moving Back to Co-Piloted/Piloted Mode**: Finally, if the game is Auto-Piloted, Participants can vote under standard rules to return the game to Piloted or Co-Piloted mode. This is another safety measure that enables Participants to give a single entity the ability to make quick changes to the Hooks. It is a measure to guard against catastrophe and enable a quick response where Participants feel it is warranted.

6.17 Taken as a whole, Evernode's governance arrangements provide a transparent, on-chain mechanism for Participants to govern the three Hooks into the long term, tempered by some temporary emergency powers available to Evernode Labs to quickly address catastrophic failure.

7. ANYONE CAN BE A GOOD HOST & EARN EVERS

7.1 The Evernode Network will be comprised of a global network of independently owned and operated Hosts. Thanks to the decentralised registry, anyone can become an Evernode host by downloading and running the software and registering with the network.

7.2 The minimum requirements for an Evernode host are quite modest as summarised in Figure 15 below.



MINIMUM REQUIREMENTS

The 4 things you'll need to be a host

Xahau Ledger Wallet

An Xahau-compatible Wallet (like XUMM) for your Xahau keys so you can access your Evers, Xahau XRP and NFTs. You must have at least enough Evers to pay for your Registration.

Real or Virtual Linux Machine

Mandtory OS: Ubuntu 20.04 (64bit)

RAM: 2GB minimum

Swap: 2GB minimum

Disk space: 4GB minimum free disk space for home

Internet & IP Address

You can't be a good host without excellent connectivity, a public internet address, and an SSL certificate. The Evernode Registry Hook auto-deregisters chronically unreliable machines.

Some Technical Savvy

You need enough technical skill to run a reliable host and comply with local hosting laws, like copyright take-down requests.

Figure 15: Minimum Host Requirements

Being a Good Evernode Host

7.3 To facilitate their role as a host, Evernode requires Hosts to have a domain, email address, and SSL certificate before it's Registration Hook will issue a Registration NFT. These are all necessary to ensure a smooth user/dApp experience. Without the email and domain, you cannot get an SSL certificate and without the SSL certificate the dApp user's experience will be degraded or even impossible as the dApp will be flagged as having potential security or privacy concerns.

7.4 Further, Evernode supports IPv6, meaning each "Slot" (see para 7.8) can be given a unique IP address. This is highly advised. Otherwise, a Host with many instances risks being treated as malicious by nodes on the Xahau Network and other third-party. Too many simultaneous requests coming from the same IP address will be regarded as an attack. The Host will be temporarily blocked from accessing the node, meaning all dApps it hosts might fail because they will be unable to submit transactions to the Xahau Network to pay their hosting fees.

Hosting Rewards – Incentivising Hosts to Exist

7.5 At launch, Evernode will face a chicken-and-egg problem arising from the symbiotic relationship between Hosts and dApps: Hosts will only exist if there's sufficient demand for hosting from dApps, while developers will only build on Evernode if a vibrant and reliable network of Hosts exists.

7.6 To solve this problem, the protocol will reward Hosts simply for being a reputable host on the network, similar to block rewards in Bitcoin and Ethereum. The emission schedule for Evers programmed into the Hook are summarised below in Table 1.

Evernode Reward Schedule

The Rules: *Rewards are distributed every Moment (=1 hour). Rewards are shared equally among eligible Hosts. Rewards start at 5120 Evers per Moment and halve every Epoch. The first Epoch lasts 6 weeks. Each subsequent Epoch doubles in duration. Rewards cease at the end of the tenth Epoch, after roughly 118.04 years.*

Epoch	Epoch Duration (est. Weeks)	Reward Trigger (1hr/Moment)	Hosting Rewards				Elapsed Years
			Each Hour	Each Day	Each Week	Each Epoch	
1st	6	Every hour	5120	122,880	860,160	5,160,690	0.11
2nd	12	Every hour	2560	61,440	430,080	5,160,690	0.23
3rd	24	Every hour	1280	30,720	215,040	5,160,690	0.46
4th	48	Every hour	640	15,360	107,520	5,160,690	0.92
5th	96	Every hour	320	7,680	53,760	5,160,690	1.85
6th	192	Every hour	160	3,840	26,880	5,160,690	3.69
7th	384	Every hour	80	1,920	13,440	5,160,690	7.38
8th	768	Every hour	40	960	6,720	5,160,690	14.77
9th	1536	Every hour	20	480	3,360	5,160,690	29.54
10th	3072	Every hour	10	240	1,680	5,160,690	59.08
TOTALS	**6138**					**51,606,900**	**118.04**

Table 1- Evernode Reward Schedule

7.7 Distribution is skewed to favour early-adopters, since each new Host is relatively more valuable to the network when it is young and small.

Hosting Fees – Paying Hosts for Hosting

7.8 In addition to Hosting Rewards, Hosts earn hosting fees in Evers. In the long-term, this will be the primary source of revenue for Hosts. Sashimono divides the Host machine into equal-sized hosting "Slots". It then mints Lease NFTs for each Slot, specifying the hourly rent in Evers, and then offers them for sale on the Xahau Network. Sashimono automatically handles all this for the Host. The model is summarised in Figure 16 below.

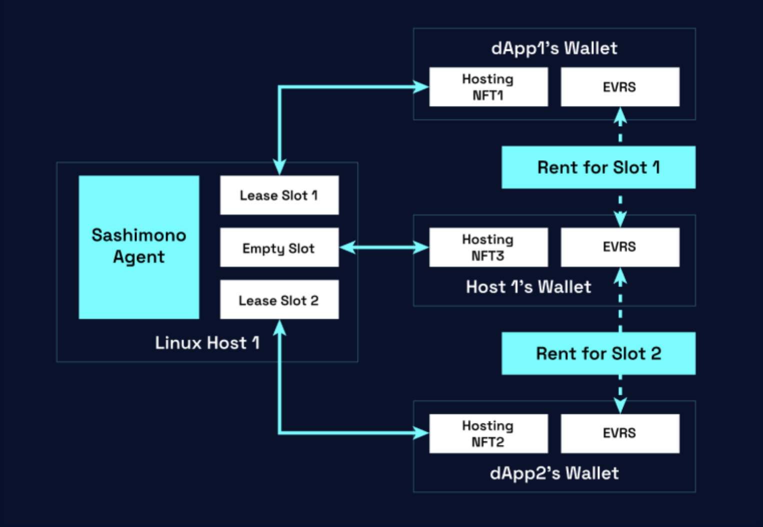


Figure 16: Leasing "Slots"

7.9 The Lease NFT represents a "best-endeavours" promise to provide exclusive use of that Slot to the holder of the Lease NFT for so long as the hourly rent is paid. At any time, a Host can revoke a Lease NFT and cancel the tenant dApp's instance, or a tenant can just stop paying rent. This right is necessary because Hosts have no relationship with the dApp but may have real-world legal obligations to meet, like copyright takedown requests, depending on their home jurisdiction.

8. ANYONE CAN BE AN EVERNODE DAPP DEVELOPER

8.1 HotPocket dApps don't **run on** blockchains, they **are** blockchains. Each dApp is its own mini-blockchain (sometimes called "AppChain") with its own chain history and dedicated nodes. This unique architecture allows for hyper-flexible, hyper-powerful dApps.

Benefits of Evernode dApps

8.2 Evernode dApps may be public or private. They may call external services, read and write data directly to disk and the web, and generally perform any task a regular program can, without centralisation or trusted third parties and without requiring the programmer to implement their own consensus mechanisms.

8.3 This flexibility solves many problems that limit mass adoption of dApps including:

(a) **Any Language**: Because Evernode dApps are just normal Apps with a consensus engine, Evernode dApps can be programmed in any POSIX-compliant language, including NodeJS, C++, and rust.

(b) **Any Functionality**: Evernode dApps can do in concert anything normal Apps can do solo, including read/write data, perform complex computations, and connect to external services.

(c) **Any Jurisdiction**: Evernode dApps can be programmed to run only on servers from certain jurisdictions, giving developers the flexibility to comply with problematic laws like privacy/GDPR and trade embargoes.

(d) **Any Scale**: Evernode dApps can be deployed to as many or as few Hosts as you desire to meet competing needs on cost, security, performance, and censorship resistance.

Sample Use Cases

8.4 In the course of development, we piloted and demonstrated a number of "working toy" versions of various use-cases, including:

(a) **Nomadic Contract**: A simple contract that spawns itself on a target number of Hosts, then randomly shuts down an instance and spins up on a new Host, making it harder to compromise.

(b) **Membership Contract:** A demo of a contract concept whereby people join the contract by running (and funding) an instance.

(c) **iXRPL – Self-KYC: A** self-sovereign identity solution where the verified identity documents are encrypted and stored on-chain and shared via single-use keys.

(d) **EVM Cluster**: A tool that lets you cut-and-paste your solidity contracts and have them run as Evernode dApps on the Evernode network.

(e) **Decentralised Hotel Booking:** A decentralised hotel booking site running on Evernode.

(f) **Digital Cows:** A working toy of a project for tokenising and trading interests in Australian cattle.

(g) **"Everdog" NFT Project:** A sample NFT project where all the data, including the generated jpegs, were stored on-chain.

(h) **On-Demand Oracles:** HotPocket dApps can elect a sub-set or jury of their own nodes to get data from off-chain, agree on the truth, and report to the rest of the chain as a bespoke, on-demand oracle.

8.5 This is a small sample of the full range of applications and use-cases Evernode can support. The scope for Evernode dApps is so broad because they are normal Apps that function as mini-blockchains and maintain a shared canonical state across multiple instances via an out-of-the-box consensus mechanism. The capacity for valuable and profitable businesses to be built on Evernode is at least as deep as the existing market for Apps.

9. INITIATOR - EVERNODE LABS PTY LTD

9.1 The Evernode Network is an initiative of Evernode Labs Pty Ltd. Evernode Labs Pty Ltd is the owner of the Evernode IP generated inside the Australian National University through UBRI-funded research and with the help of XRPL Grants.

9.2 Evernode Labs will make the Evernode IP available to the world for bona-fide purposes associated with the Evernode Network through a mixture of free open-source software (the 3 Hooks) and closed-source licensing arrangements (HotPocket/Sashimono binaries).

10. PROPOSED LAUNCH

10.1 Evernode is targeting a launch before the end of 2023. Launch of the network is dependent on full wallet support for the Xahau Network. Until there is a reliable wallet that supports users cloning their XRPL Account on Xahau, we will be unable to finalise our airdrop and unable to launch because prospective Hosts will not be able to access the Evers they need to become a Host.