

Hyperledger Besu Workshop for Financial Services

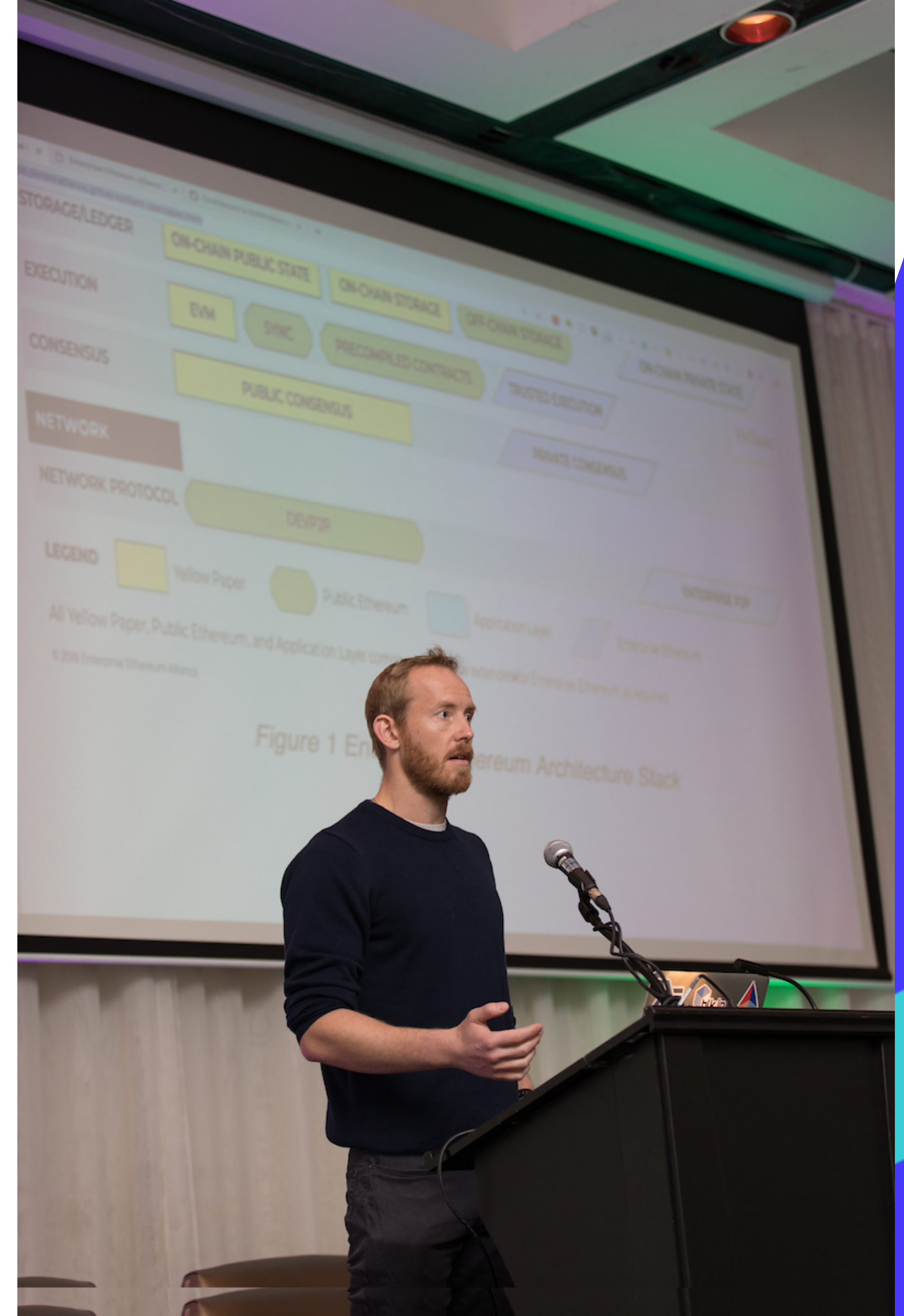
Conor Svensson

Founder/CEO @ Web3 Labs

March 2023

Perspectives on:

**JVM development on Ethereum
Hyperledger Besu and Quorum
Running Hyperledger Besu in
Production**

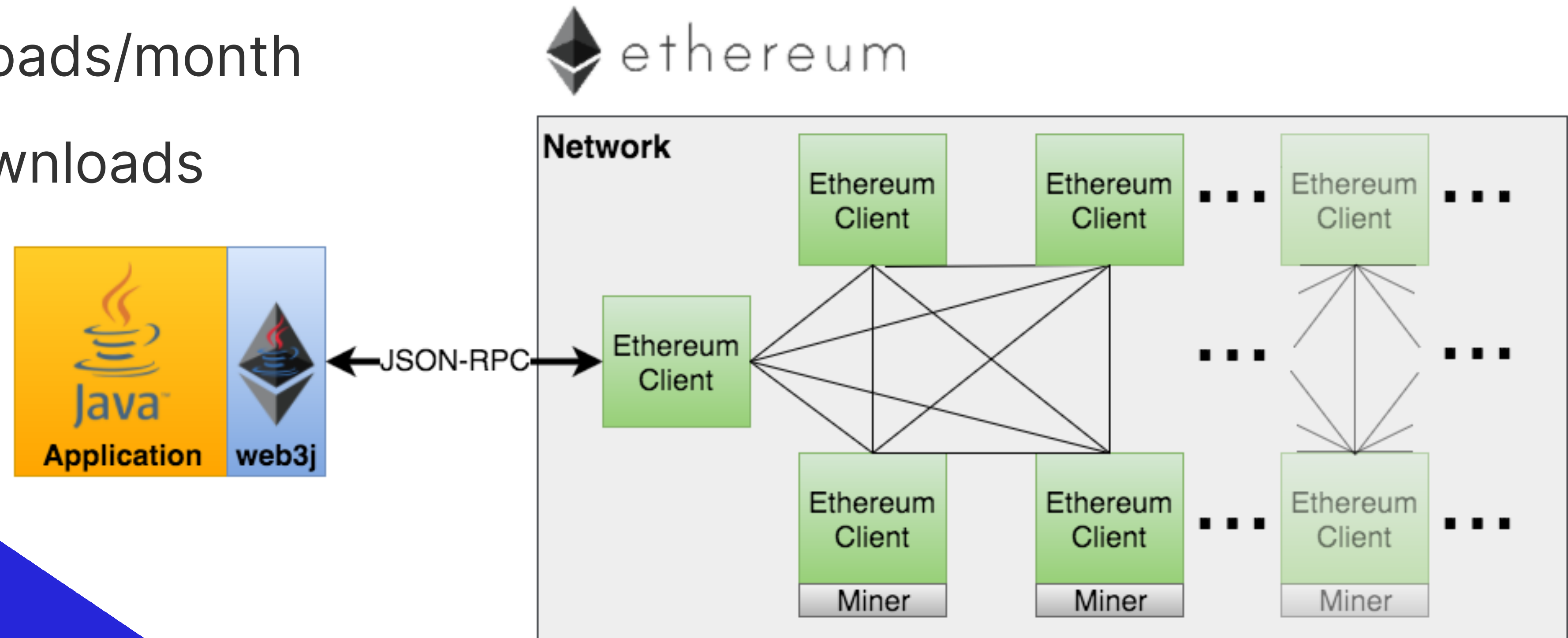


JVM development on Ethereum

Web3j (and Besu)

Web3j

- Leading open-source developer library for Ethereum
- Initial release in September 2016
- Over 100k downloads/month
- Over 2 million downloads



Components

Core library

- JSON-RPC, wallets, smart contract wrappers

Build tools

- Gradle and Maven plugins
- Embedded EVM (Hyperledger Besu)
- Embedded Solc + version manager
- Testing framework
- IntelliJ debugger

CLI

- Project scaffolding

Project stats

- Created: September 2016
- Unique contributors: 180
- Releases: 101
- GitHub stars: 4.5k

Corporate users

vmware®



vodafone



UBS

FNALITY 

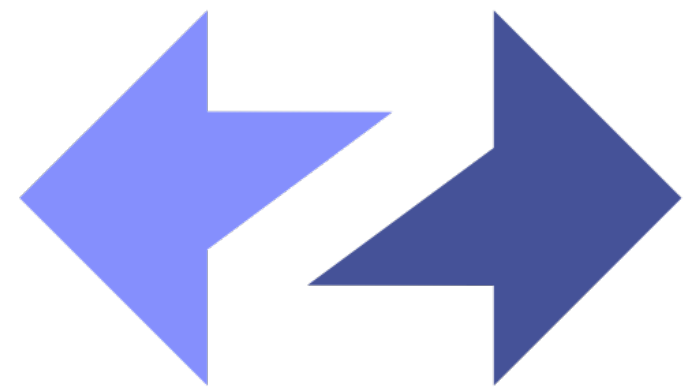
Web3 users



Magic



CONSENSYS



zkSync



AlphaWallet



SIMBA



amberdata



Harmony

Using Web3j — CLI

```
$ curl -L get.web3j.io | sh
```

```
$ web3j [OPTIONS] [COMMAND]
```

```
...
```

```
$ web3j new ...
```

```
$ web3j wallet ...
```

Web3j and Besu

Besu

- Acceptance tests

Web3j

- Besu module
- Besu EVM

The screenshot shows the Hyperledger Besu documentation website. The top navigation bar includes the Hyperledger Besu logo, a search bar, and the text 'hyperledger/besu'. Below the navigation bar, there are tabs for 'Public networks' and 'Private networks'. The main content area is titled 'Private network API methods' and includes a date of last update: 'September 27, 2022'. A sidebar on the left lists navigation options under 'Private networks', with 'Besu API' and 'Private network API objects' selected. The main content area features an 'Attention' box with a warning icon, stating that the reference contains API methods for private networks only and that JSON-RPC HTTP examples use a default host and port endpoint. Below this is a section for 'CLIQUE methods' with a 'Note' box explaining that these methods are not enabled by default for JSON-RPC and can be enabled using specific options. The page lists several API methods, including 'clique_discard', 'clique_getSigners', 'clique_getSignerMetrics', 'clique_getSignersAtHash', 'clique_proposals', and 'clique_propose'. A sidebar on the right lists 'CLIQUE methods', 'EEA methods', 'IBFT 2.0 methods', and 'PERM (Permissioning) methods' with their respective sub-methods.

HYPERLEDGER
BESU

Search

hyperledger/besu

Public networks Private networks

Private networks

Get started >

How to >

Concepts >

Tutorials >

Reference >

Besu command line >

Besu API >

Private network API objects

Accounts for testing

Plugin API interfaces

Date of last update: September 27, 2022

private networks

Private network API methods

Attention

- This reference contains API methods that apply to only private networks. For API methods that apply to both private and public networks, see the [public network API reference](#).
- All JSON-RPC HTTP examples use the default host and port endpoint `http://127.0.0.1:8545`. If using the `-rpc-http-host` or `-rpc-http-port` options, update the endpoint.

CLIQUE methods

The `CLIQUE` API methods provide access to the [Clique](#) consensus engine.

Note

The `CLIQUE` API methods are not enabled by default for JSON-RPC. To enable the `CLIQUE` API methods use the `--rpc-http-api` or `--rpc-ws-api` options.

`clique_discard`

CLIQUE methods

- `clique_discard`
- `clique_getSigners`
- `clique_getSignerMetrics`
- `clique_getSignersAtHash`
- `clique_proposals`
- `clique_propose`

EEA methods

- `eea_sendRawTransaction`

IBFT 2.0 methods

- `ibft_discardValidatorVote`
- `ibft_getPendingVotes`
- `ibft_getSignerMetrics`
- `ibft_getValidatorsByBlockHash`
- `ibft_getValidatorsByBlockNu...`
- `ibft_proposeValidatorVote`

PERM (Permissioning) methods

- `perm_addAccountsToAllowlist`
- `perm_addNodesToAllowlist`
- `perm_getAccountsAllowlist`
- `perm_getNodesAllowlist`
- `perm_reloadPermissionsFro...`
- `perm_removeAccountsFrom...`
- `perm_removeNodesFromAllo...`

Web3j and Besu

```
@EVMTest
class GreeterTest {

    @Test
    fun greeterDeploys(
        web3j: Web3j,
        transactionManager: TransactionManager,
        gasProvider: ContractGasProvider
    ) {
        val greeter = Greeter.deploy(web3j, transactionManager, gasProvider, "Hello EVM").send()
        val greeting = greeter.greet().send()
        assertEquals("Hello EVM", greeting)
    }
}
```

Learn More

 <https://github.com/web3j/>

 web3labs.com

 @ConorSvensson / @web3labs

Besu and Quorum

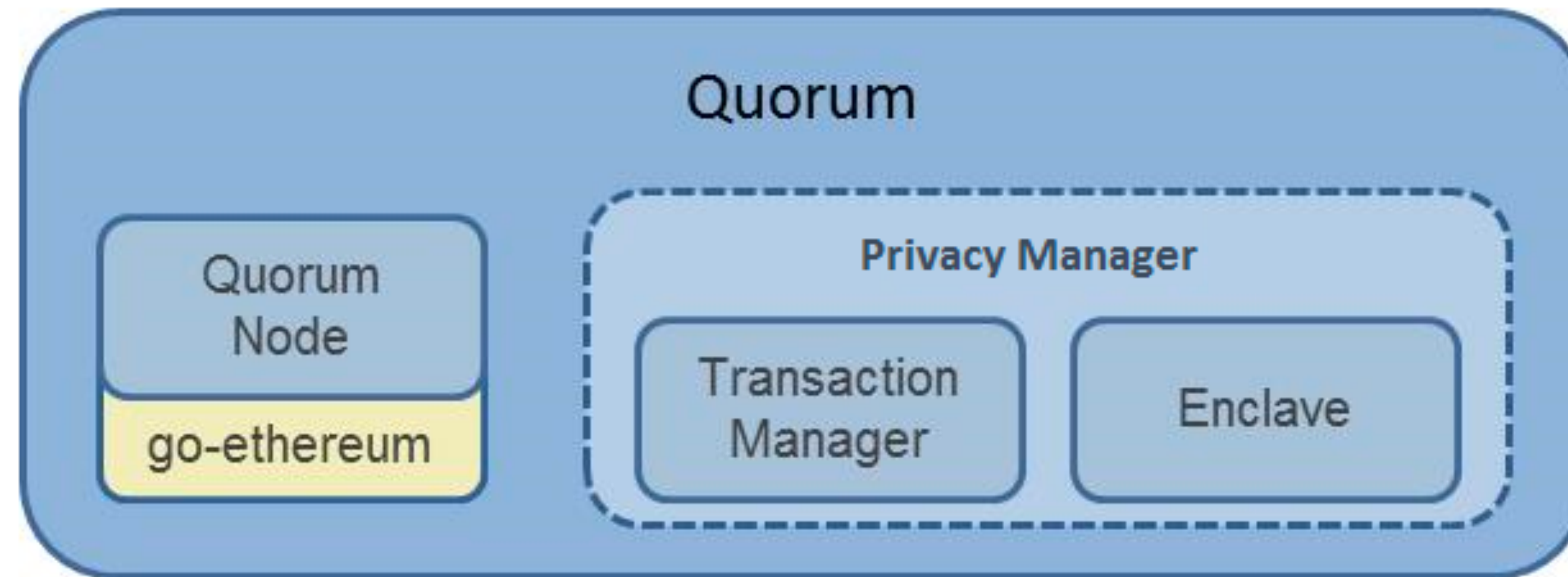
A historical perspective

The dawn of Enterprise Ethereum

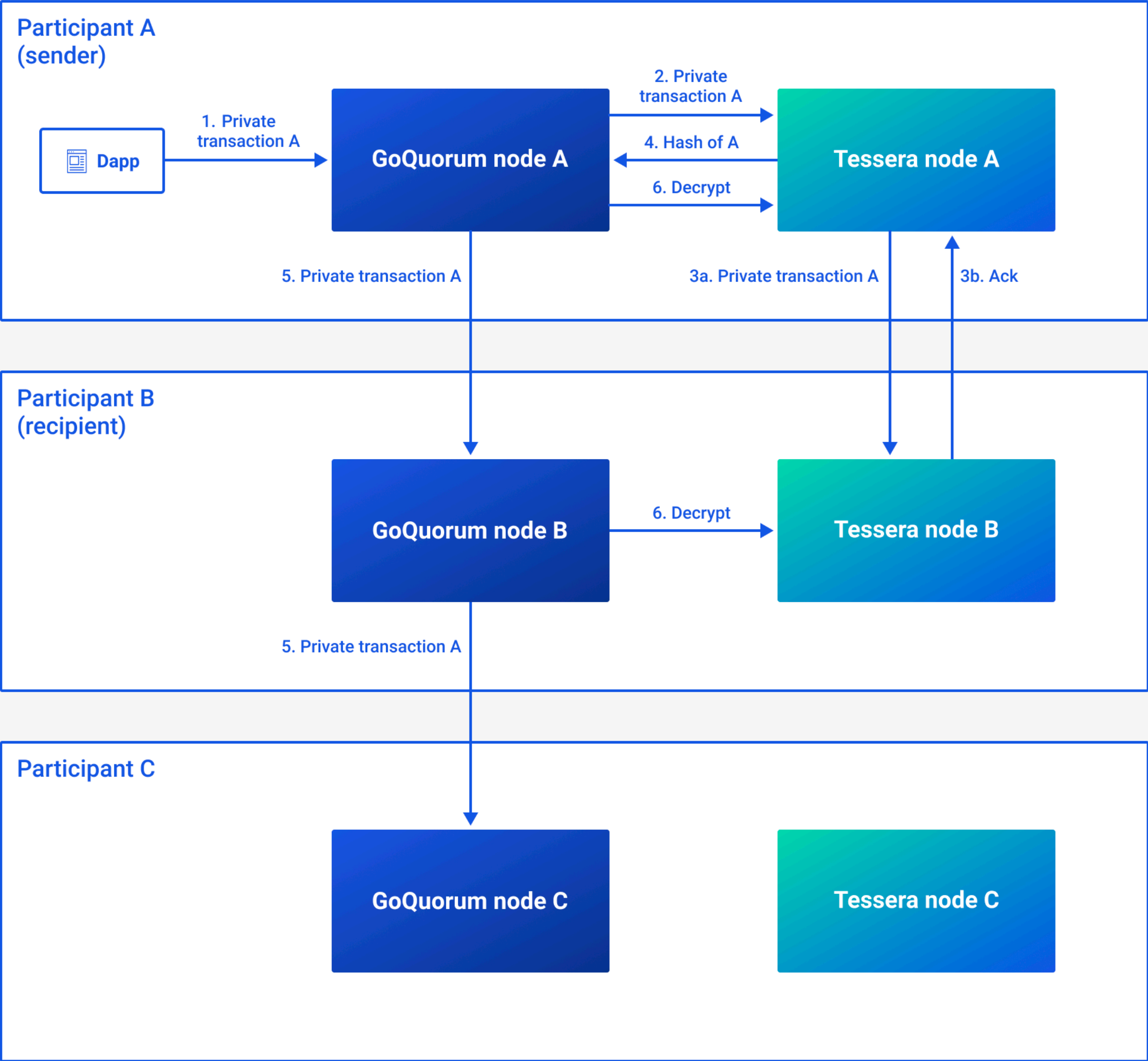
Quorum (November 2016) + Enterprise Ethereum Alliance (March 2017)



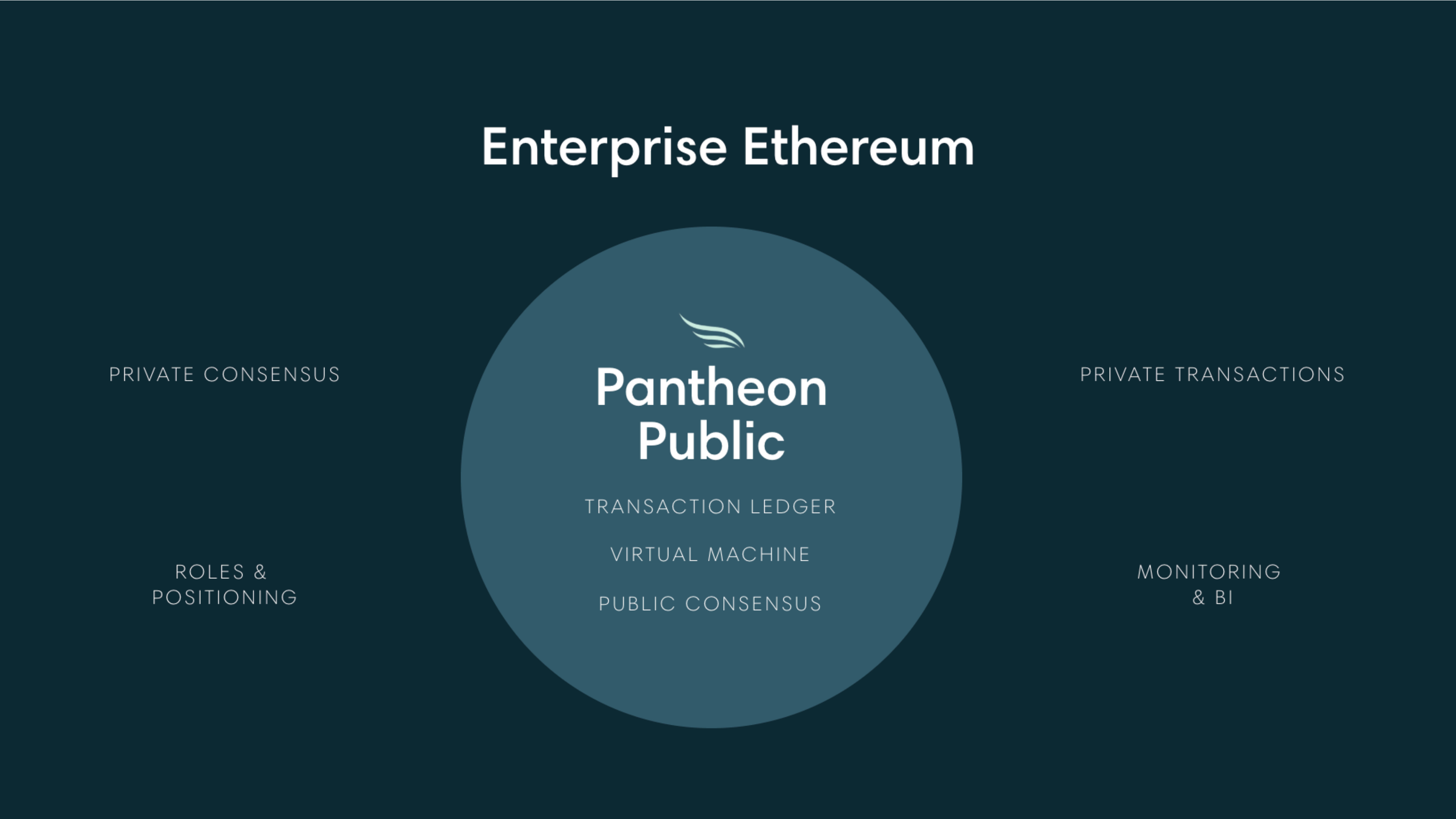
The Quorum Innovation



Quorum Privacy



Enter Pantheon (2019)



Hyperledger Besu vs Quorum



EEA Core Specifications Working Group

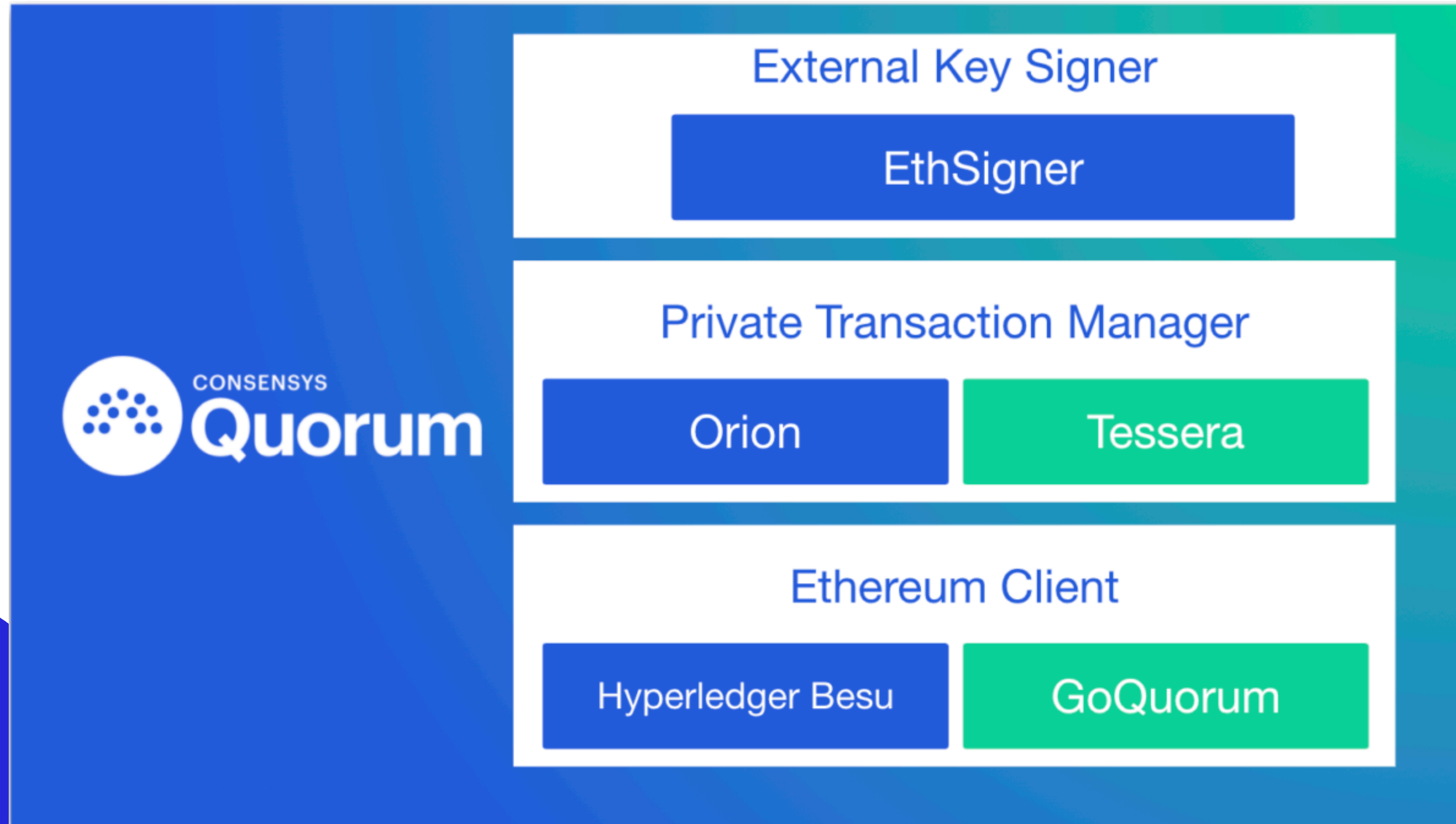
Convergence (2020)

ConsenSys Acquires J.P. Morgan's Quorum to Advance Enterprise Blockchain Adoption

Just announced, the JPM Quorum open source codebase will now be maintained by ConsenSys and made compatible with other enterprise products.

by **ConsenSys** – August 25, 2020

The present — Besu and Quorum



2023 and beyond

Hyperledger Besu

- ✓ Influence ⇒ governance via Hyperledger Foundation
- ✓ Actively contributed to by multiple firms
- ✓ Important to ConsenSys' strategy
- ✓ Apache 2.0 licensed

ConsenSys GoQuorum

- ✗ Little influence on upstream development (Ethereum Foundation Geth)
- ✗ GPL licensed
- ✗ Few contributing firms
- ✗ Not core to ConsenSys' strategy
- ✗ Quorum Blockchain Service EOL'd
- ✓ Client diversity

Running Besu in production

Some considerations

Key considerations

- Security
- Monitoring
- Support



Security

- Node type segregation — validators, RPC nodes (read-only), service connectors (validators → RPC)
- Key management — EthSigner, vendor or bespoke?
- Adhering to infosec best practices
- AI ...

ChatGPT-4



Conor
@jconorgrogan · [Follow](#)



I dumped a live Ethereum contract into GPT-4.

In an instant, it highlighted a number of security vulnerabilities and pointed out surface areas where the contract could be exploited. It then verified a specific way I could exploit the contract

```
This is a solidity contract. Can you help me review it and let me know if there are any security vulnerabilities? **
*Submitted for verification at Etherscan.io on 2018-01-28
*/

pragma solidity ^0.4.18;

// If you wanna escape this contract REALLY FAST
// 1. open MEW/METAMASK
// 2. Put this as data: 0xb1e35242
// 3. send 150000+ gas
// That calls the getMeOutOfHere() method

// Wacky version, 0-1 tokens takes 10eth (should be avg 200% gains), 1-2 takes a
// 30eth (avg 100% gains), and beyond that who the fuck knows but it's 50% gains
// 10% fees, price goes up crazy fast
contract PonziTokenV3 {
    uint256 constant PRECISION = 0x10000000000000000; // 2^64
    // CRR = 80 %
    int constant CRRN = 1;
    int constant CRRD = 2;
    // The price coefficient. Chosen such that at 1 token total supply
    // the reserve is 0.8 ether and price 1 ether/token.
    int constant LOGC = -0x296ABF784A358468C;

    string constant public name = "ProofOfWeakHands";
    string constant public symbol = "POWH";
    uint8 constant public decimals = 18;
}
```

from later investors. Such schemes are inherently fraudulent and illegal in most countries.

- The contract is not secure against reentrancy attacks. In particular, the `withdraw()` function can be called recursively by a malicious contract, leading to a drain of the contract's funds. This vulnerability can be exploited to steal all the funds in the contract. The `sellMyTokensDaddy()` function also contains a reentrancy vulnerability.
- The `fund()` function accepts payments of any value greater than 0.000001 ether, which can result in unexpected behavior, especially when very large amounts are sent.
- The `sell()` function can lead to integer underflow if the input value `amount` is greater than `totalSupply`. This can be exploited to steal funds from the contract.
- The `getEtherForTokens()` function contains a potential division-by-zero error if the input value `tokens` is equal to `totalSupply`.
- The contract relies on fixed-point arithmetic for mathematical calculations, which is prone to precision errors and can result in unexpected behavior.
- The contract does not include sufficient comments and explanations of the code.

Yes, submitting a large amount in the `uint256 _value` field can exploit a vulnerability in this contract. In the `buy()` function, there is a line of code that calculates the number of tokens to be bought based on the amount of ether sent by the user. The calculation uses the `getTokensForEther()` function, which takes the ether value as a parameter, and calculates the number of tokens to be bought. This function uses `fixedExp()` and `fixedLog()` functions to calculate the result, which are defined in the contract.

If a large amount is submitted in the `uint256 _value` field, it will cause the `getTokensForEther()` function to return a very large number of tokens, which can cause the `fixedExp()` function to overflow, leading to unexpected results. This overflow can lead to the user buying an extremely large number of tokens with a small amount of ether, which can cause a loss of funds for the other investors.

Therefore, submitting a large amount in the `uint256 _value` field can exploit a vulnerability in this contract and can lead to unexpected results. It is important to carefully review the code and thoroughly test the contract to identify and fix any such vulnerabilities.

5:31 PM · Mar 14, 2023



[Read the full conversation on Twitter](#)



11.5K





Reply

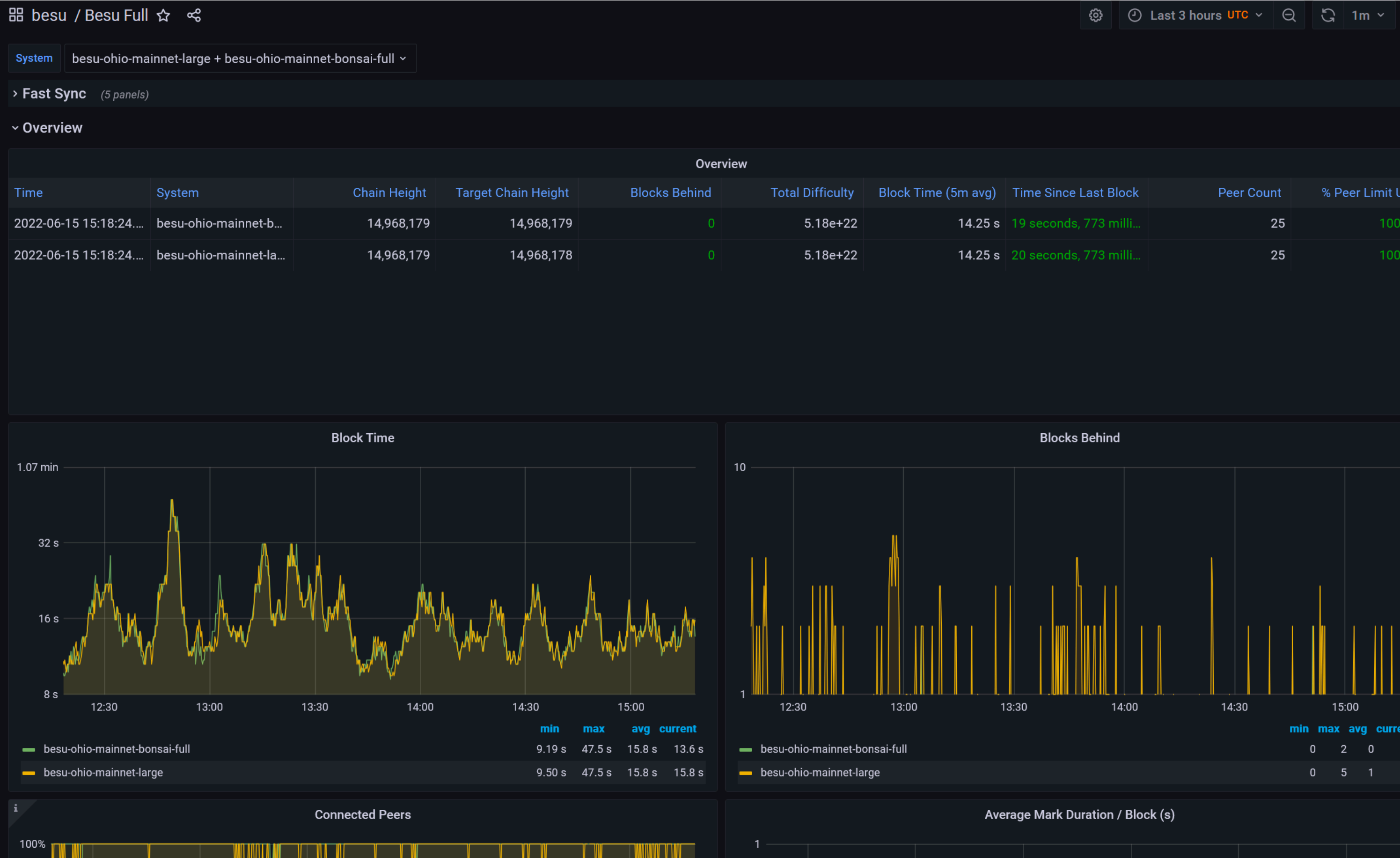


Copy link

Monitoring

Operational metrics

- Prometheus/Grafana
-  Node health scenarios
-  Network health scenarios



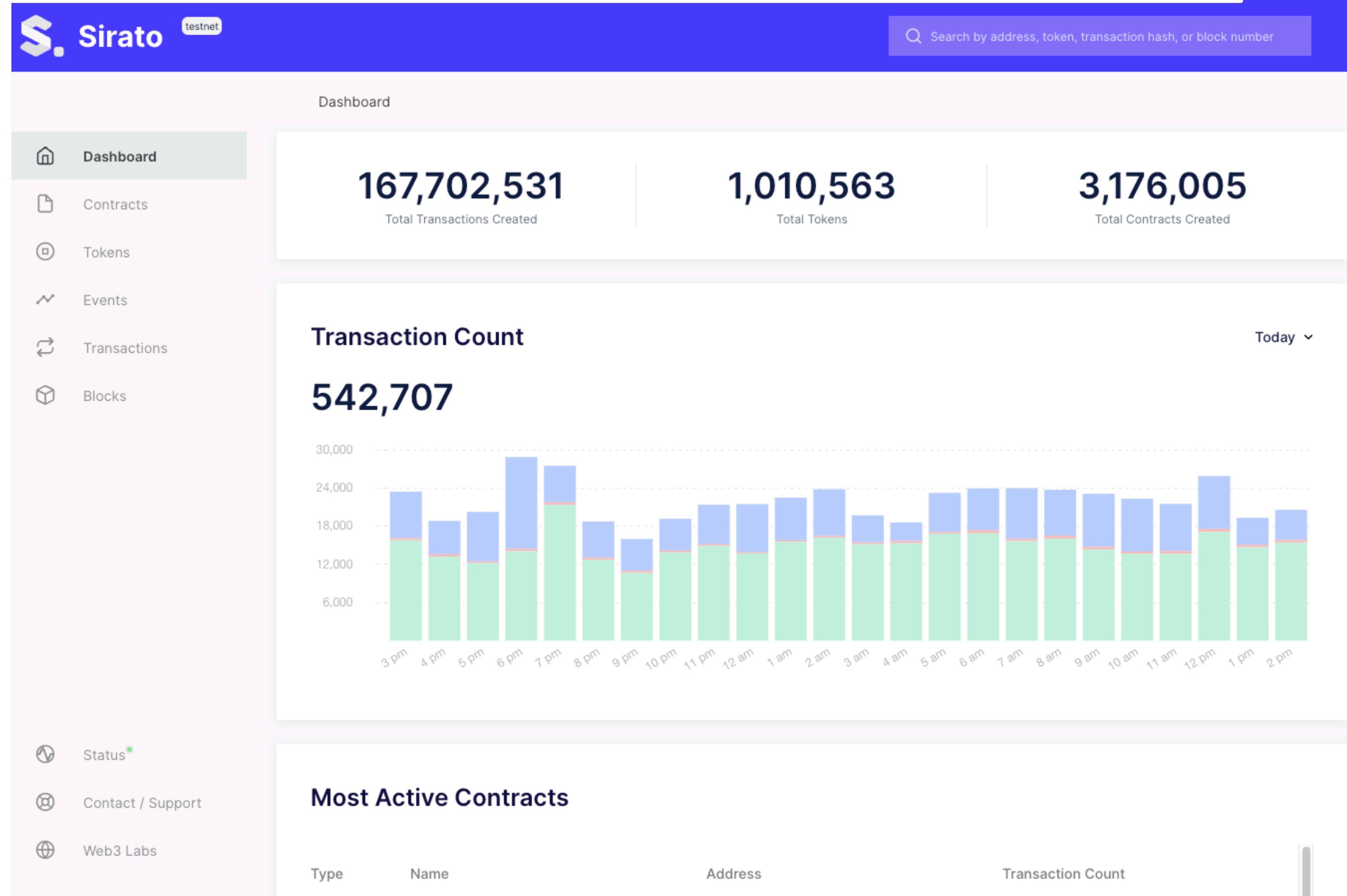
Monitoring

On-chain metrics

- Token or asset activity
- Contract events

Reporting

- Ad-hoc
- Business intelligence



Support

Production support

- Metric based SLAs
- Knowledge bases

Custom releases

- Hotfixes
- Verified and/or long term support builds

Takeaways

Web3j 🤝 Besu

✅ **Hyperledger Besu** or GoQuorum?

Production — security, monitoring and support

- RPC versus validator nodes, key management, AI
- Challenges with network health + on-chain activity
- SLAs + hotfixes or LTS

Web3 Labs

Web3 Labs is a leading blockchain strategy and technology company specialising in decentralised infrastructure solutions.

Their Sirato platform provides data and analytics for EVM and Substrate based blockchains.

They also provide commercial support for Hyperledger Besu and Quorum.



web3labs.com



hi@web3labs.com



youtube.com/c/Web3Labs



[@web3labs](https://twitter.com/web3labs)



linkedin.com/company/web3labs



github.com/web3labs