Ethereum Testing and Security

Today's Topics:

- Execution Layer Testing
  - EVM Testing: ethereum/tests, ethereum/execution-spec-tests (EEST)
- Consensus Layer Testing
  - ethereum/consensus-specs
- Cross-Layer (Interop) Testing
  - Hive
  - Devnets
  - Shadow-Forks
  - Testnets
- Security
  - Potential issues, bug bounties and public disclosures

EVM testing

Main purpose is to test that an Ethereum Execution Client adheres to the specification.

All clients must produce the same output when given the same input in the form of a single or multiple transactions, an environment, a pre-state, and hard-fork activation rules.

EVM testing - Important Characteristics of a Test

Pre-State: Entire composition of an Ethereum chain, composed of accounts with balances, nonces, code and storage, before any of the tests are executed.

Environment: Depending on the type of the test, the environment can specify things such as the timestamp, the prev-randao, the block number, the previous block hashes, the total gas limit, the base fee and hard-fork activation times.

Transaction(s): Messages to the blockchain that perform actions on the blockchain. Contains the origin and destination accounts, ether value, gas-limit and data.

Post-State: Resulting state composed of the modified or created accounts.

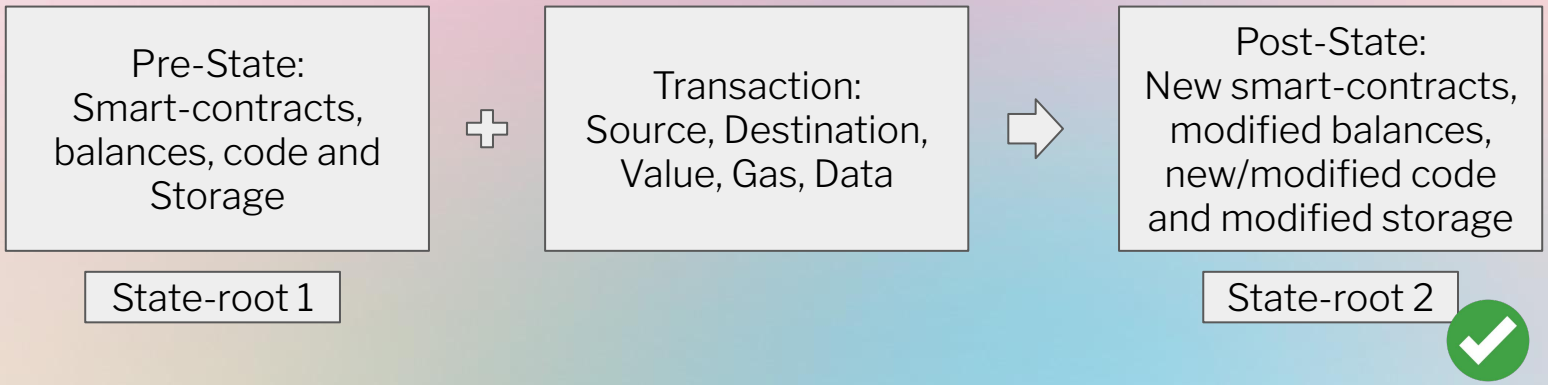
EVM testing - Tests Filling

Process of compiling a test source code into a fixture that can be consumed by any execution client.

Differs from client unit testing in the sense that the exact same test can be executed in any client implementation.

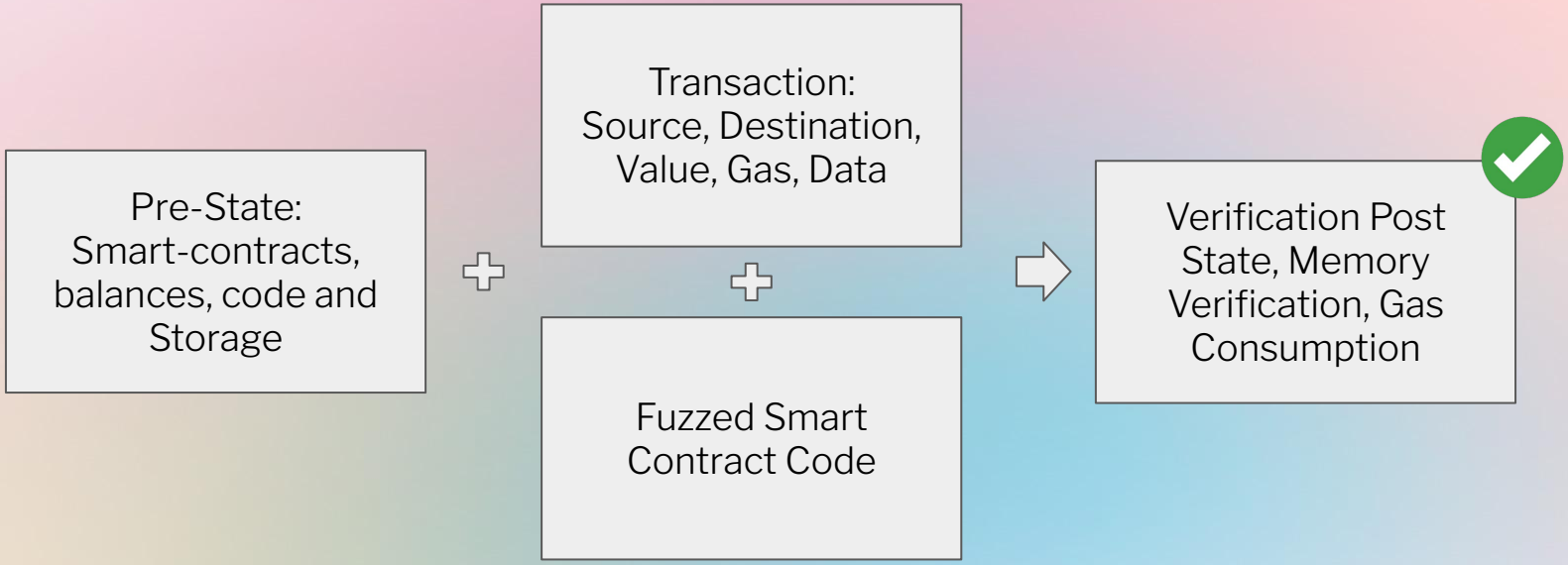
All test fixtures in their different formats are simple JSON files.

EVM testing - State Testing

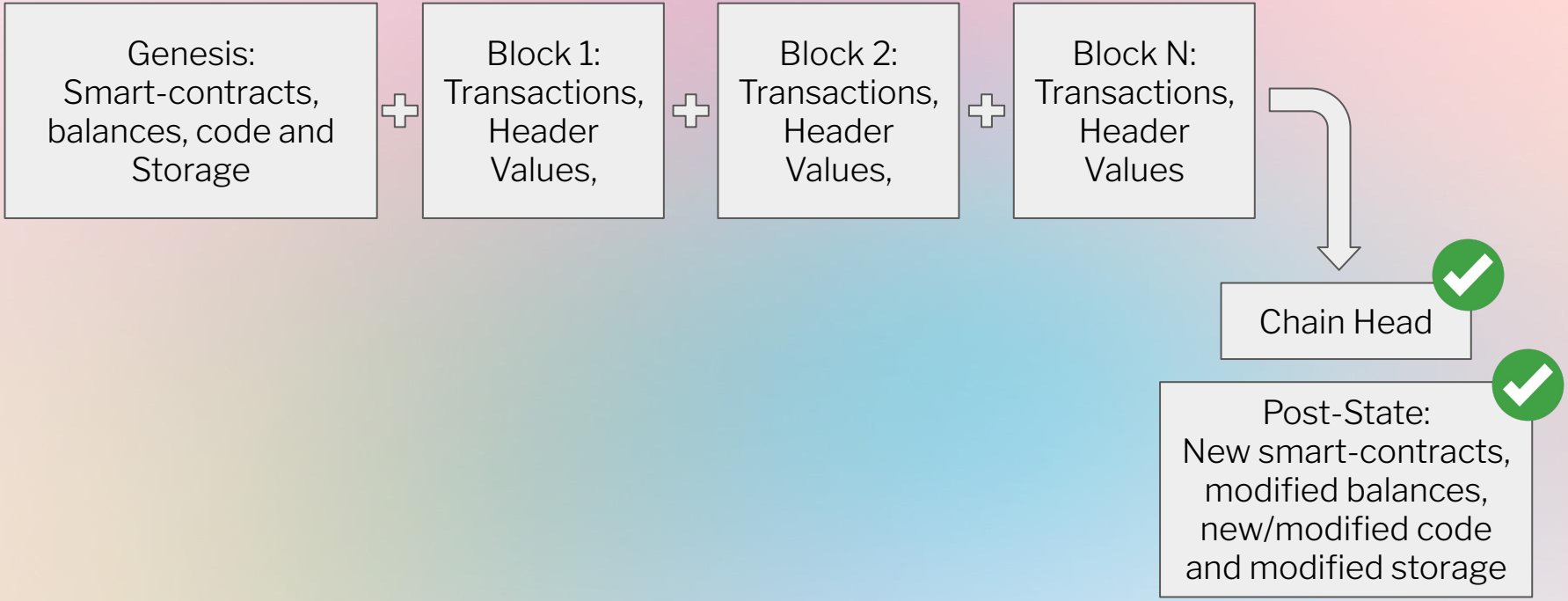


State-root: Cryptographic computation that securely commits all the contents of the state

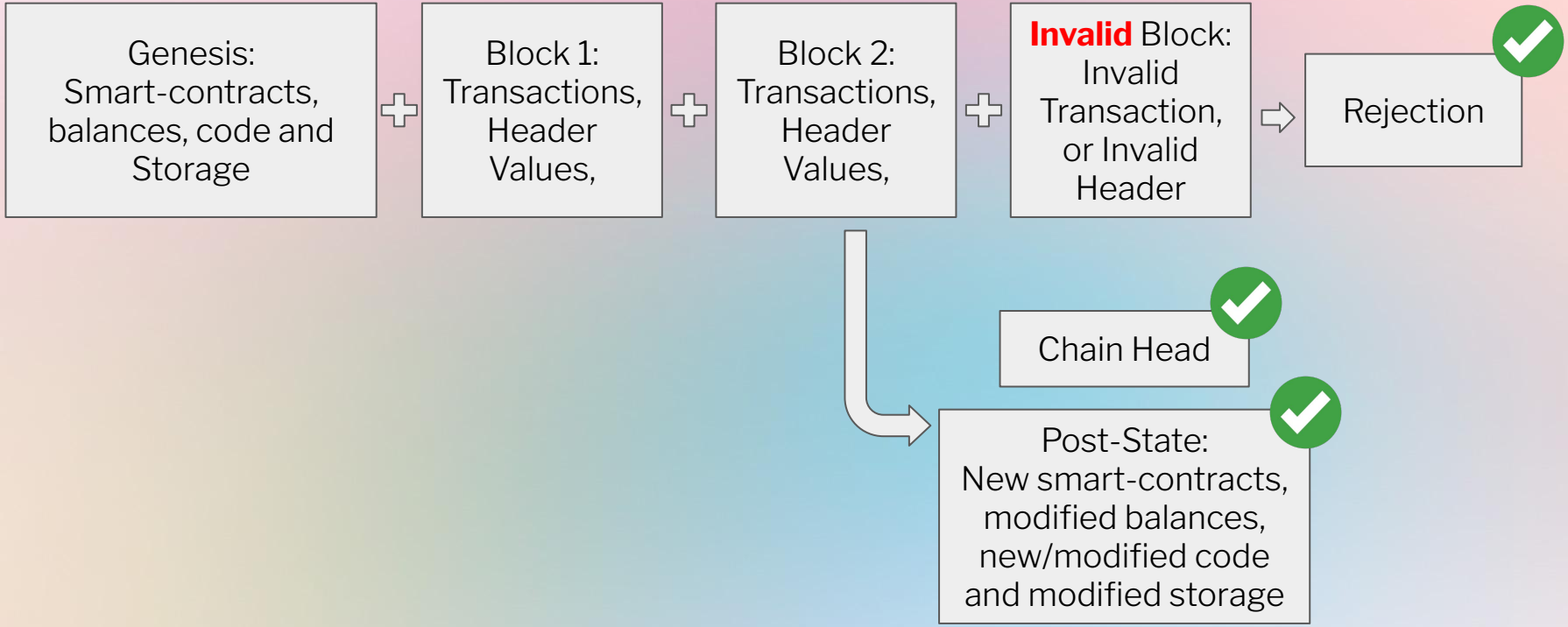
EVM testing - Fuzzy Differential State Testing



EVM testing - Blockchain Testing



EVM testing - Blockchain Negative Testing



EVM testing - Tests Filling - ethereum/tests

- Simple JSON and YAML source codes
- Provides simple parametrization
- Filled by Retesteth (Written in C++)

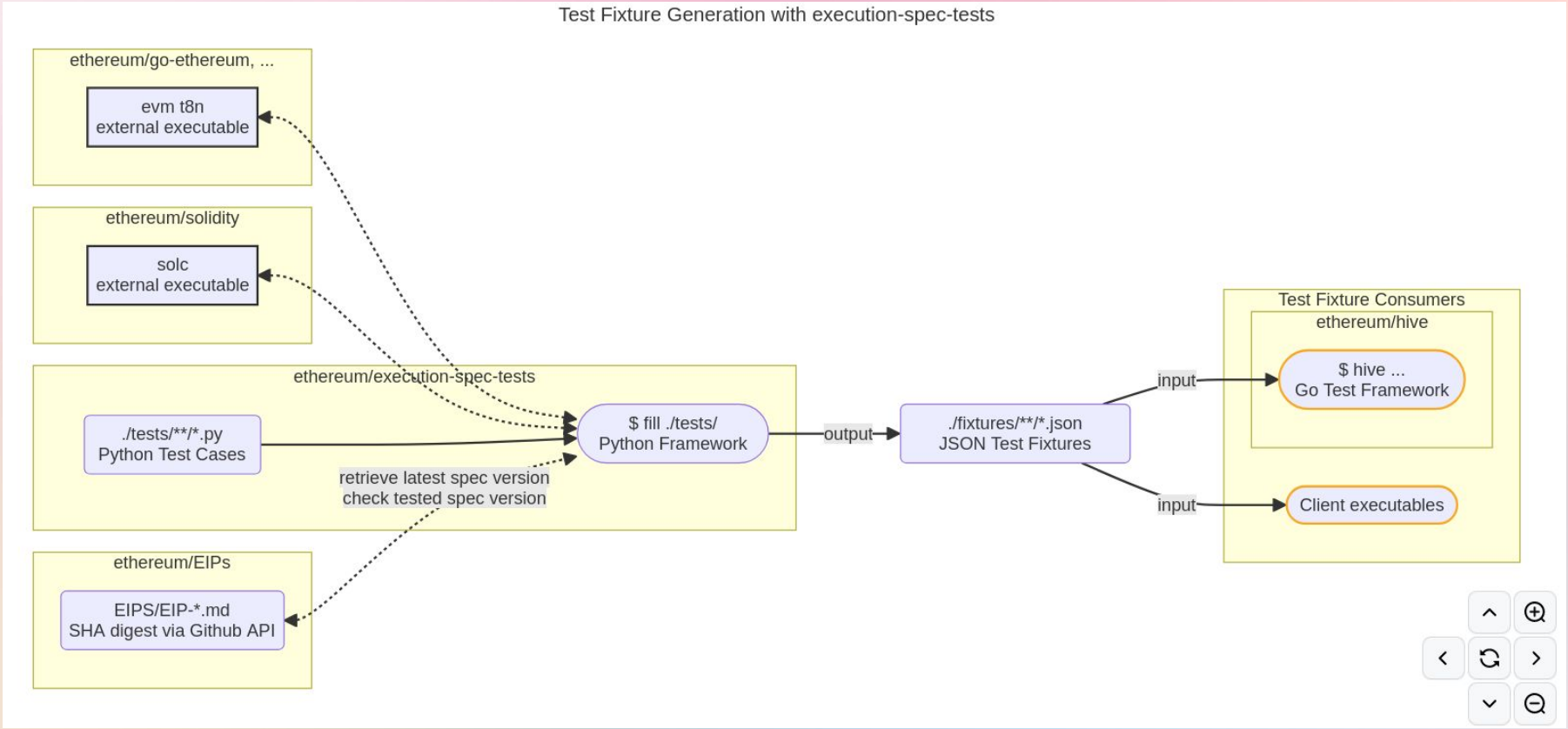
https://github.com/ethereum/tests

EVM testing - Tests Filling - Execution Spec Tests

- Python source code
- Powered by pytest and provides simple to complex parametrization
- Still requires an actual client implementation to fill because of the transition function

https://github.com/ethereum/execution-spec-tests/

EVM testing - Tests Filling - Execution Spec Tests



EVM testing - Tests Filling - Execution Spec Tests

Demo →

Execution Layer Testing - EVM testing - Fuzzy Differential Testing

https://github.com/holiman/goevmlab

FuzzyVM [fuzzεvm]

A framework to fuzz Ethereum Virtual Machine implementations. FuzzyVM creates state tests that can be used to differential fuzz EVM implementations against each other. It only focus on the test generation part, the test execution is handled by goevmlab.

https://github.com/MariusVanDerWijden/FuzzyVM

Go evmlab

This project is inspired by EVMlab, which was written in Python. EVMlab featured a minimal "compiler", along with some tooling to view traces in a UI, and execute scripts against EVMs (parity and geth).

This is a golang rewrite from scratch of that same project, this time in go-lang to be more stable and nice to use.

Execution Layer Testing - Execution APIs

README CC0-1.0 license

Execution API Specification

JSON-RPC

View the spec

The Ethereum JSON-RPC is a standard collection of methods that all execution clients implement. It is the canonical interface between users and the network. This interface allows downstream tooling and infrastructure to treat different Ethereum clients as modules that can be swapped at will.

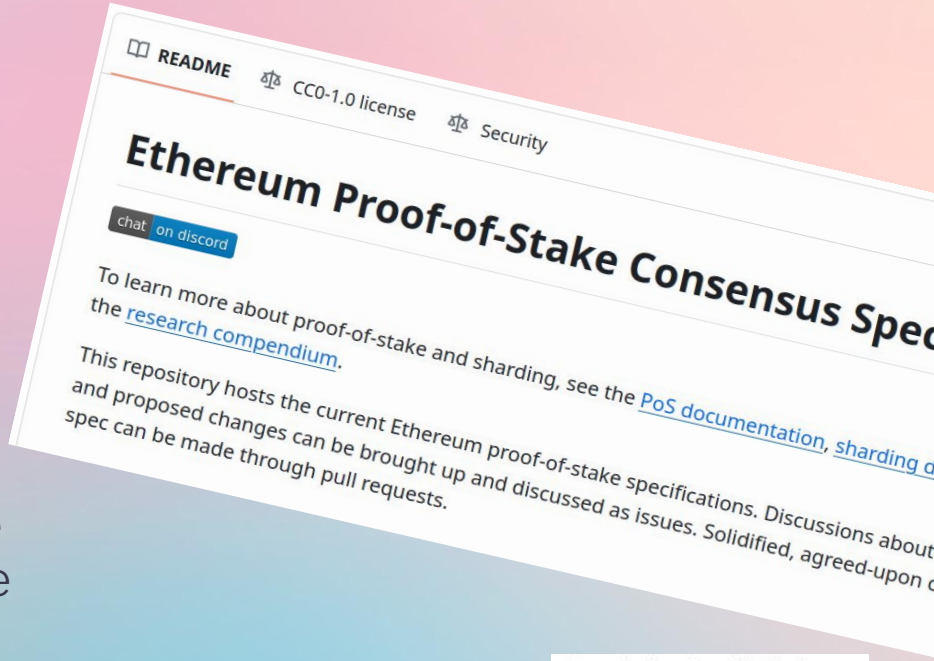
https://github.com/ethereum/execution-apis

Consensus Layer Testing

Similar idea in terms of generating test fixtures in many different formats that all clients can consume.

Self contained within the spec, hence the tests can be written and filled in the same repository.

Written in Python, same as execution-spec-tests



https://github.com/ethereum/consensus-specs/tree/dev/tests

Cross-Layer Testing - Execution + Consensus

Involve testing a fully instantiated client, feeding information to it and verifying the correctness of its behavior.

Tools:

- https://github.com/ethereum/hive
- https://github.com/ethpandaops/assertoor
- https://github.com/kurtosis-tech/ethereum-package

Cross-Layer Testing -Hive

hive - Ethereum end-to-end test harness

Hive is a system for running integration tests against Ethereum clients.

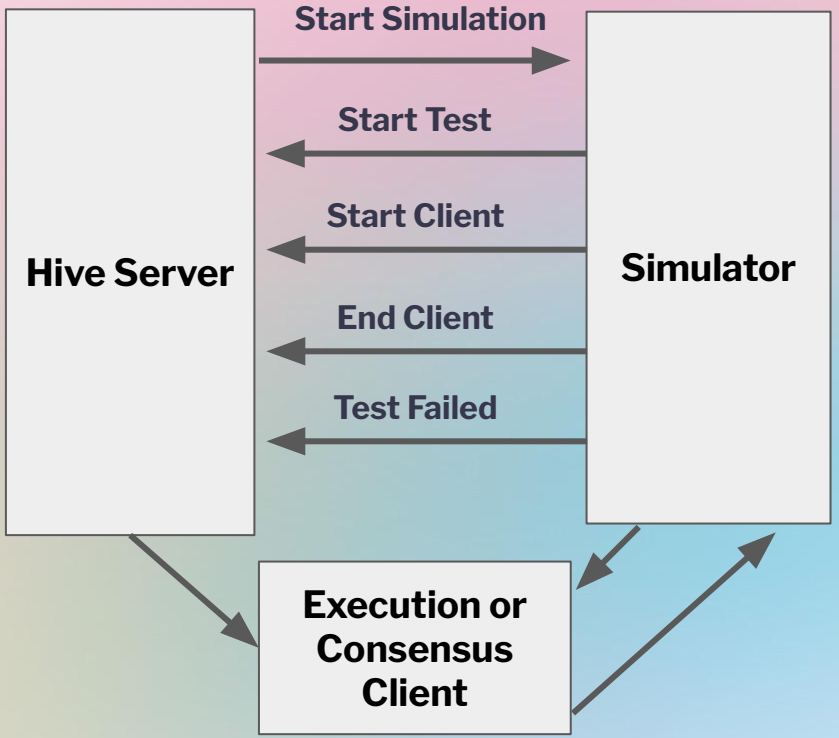
Ethereum Foundation maintains two public Hive instances to check for consensus, p2p and blockchain compatibility:

- eth1 consensus, graphql and p2p tests are on https://hivetests.ethdevops.io
- Engine API integration and rpc tests are on https://hivetests2.ethdevops.io

To read more about hive, please check the documentation.

https://github.com/ethereum/hive/

Cross-Layer Testing -Hive



Cross-Layer Testing -Hive -Simulators

https://github.com/ethereum/hive/tree/master/simulators

Cross-Layer Testing -Hive -Demo

Demo →

Cross-Layer Testing -Devnets, Shadow-forks, Testnets

- **Devnets**
  - Limited node count chains that are used to verify proof of concept or early stages of hard-forks
- **Shadow-Forks**
  - Limited node count forks that are configured to follow Ethereum mainnet, but have an early hard-fork configuration time in order to test real network activity
- **Public Testnets**
  - https://goerli.etherscan.io/ (RIP)
  - https://sepolia.etherscan.io/ (Launched Oct-23-2021)
  - https://holesky.etherscan.io/ (Launched Sep-28-2023)

Ethereum Security

- Potential Issues
- Bug Bounties
- Public Disclosures

Ethereum Security - Potential Issues

- **Execution Layer Side:**
  - Valid Invalidation: Execution client invalidates a block that fully complies with the Ethereum specification
  - Invalid validation: Execution client validates a block that does _not_ comply with the Ethereum specification
  - DoS during block execution: A client takes too much time to process a block due to a transaction

Ethereum Security - Potential Issues

- **Consensus Layer Side:**
  - **Faulty clients and finalization:**
  - <33% faulty node majority: Can cause missed slots but chain will still finalize
  - 33%+ faulty node majority: Can cause delayed finality
  - 50%+ faulty node majority: Can disrupt forkchoice
  - 66%+ faulty node majority: Can finalize an incorrect chain

Ethereum Security - Bug Bounty and Disclosure

https://bounty.ethereum.org/

https://github.com/ethereum/public-disclosures

Thank you!

Mario Vega
EF Testing Team

Twitter: @elbuenmayini, Github: marioevz

Appendix

Some extra assets

ethereum foundation

ecosystem support program

fellowship program

Appendix

Some extra assets