Hunt the Bug, Save the Chain

Uncovering Bugs in EIP Implementations

EF Testing

(Dan, Dimitry, Mario, Spencer)

Ethereum Foundation

Introduction.

Who we are

We are the Execution Specs Testing Team, within of the Ethereum Foundation, and we are responsible for developing the tests that verify the correct behavior of all Ethereum clients for all new forks.

We write extensive amounts of tests in order for clients to consume and verify that their EVM implementation does not contain any bugs.

We also help with tooling that facilitates running, debugging and monitoring of the test results.

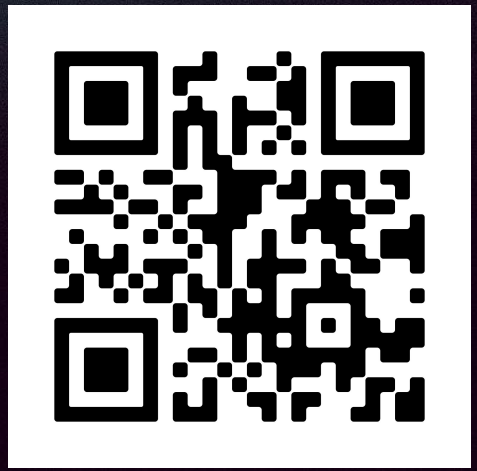
Our main repository is https://github.com/ethereum/execution-spec-tests/

What we will be doing today:

1. Get to know and setting up the testing repository ⚗️
2. Add and analyze a simple test 🔧
3. Running your tests against a live network 🚀
4. Implementing a test for a new EIP 📖
5. Finding edge-cases in the new EIP 🔍
6. Break the chain 💥

Workshop Notes

- **Please bookmark this page!**
- It contains all the information required.
- If you miss something from the slides it will be there.



https://notes.ethereum.org/@spencer-tb/eest-workshop

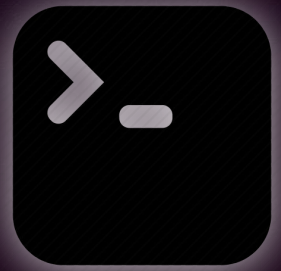
⭐ **Wifi for the room:**

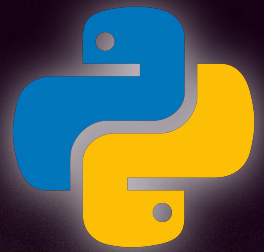
- **Name - Classroom**
- **Password - noclientisolation**

Required Tools.

What tools will we use today

- The command line:
  - Ideally you should be familiar with running commands through the command line in your computer, which might be different from machine to machine.
  - `curl`, `git`, and `python` are some examples of commands we will be using today.
- Python:
  - We'll be coding tests in python today, but there is no need to be an expert!
  - We'll to stick as much as possible to code templates provided through the presentation.
  - We'll install it through `uv` at a later stage





What tools will we use today (continued)

- Git:
  - The `git` command should already be installed in Linux and Mac
  - Verify its installation by running `git --version`
  - Otherwise it can be installed in the following links:

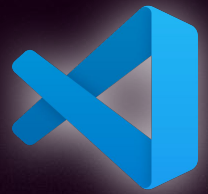
Windows

https://gitforwindows.org/

https://git-scm.com/

What tools will we use today (continued)

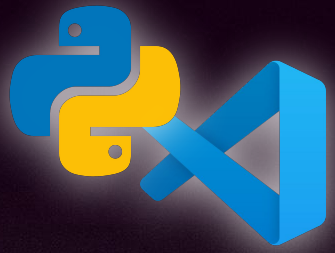
- IDE of your choice:
  - If you already have an IDE installed on your computer that can be used to develop python you can use that.
  - However we highly recommend you to use VS Code for this workshop, which can be found in the following link (or just google VS Code):



https://code.visualstudio.com/

What tools will we use today (continued)

- Extensions for your IDE:
  - If the IDE you are using today is not python native, we recommend to install a python extension to make coding easier.
  - If using VS Code, we recommend installing the official Python extension, which can be found in the following link:



https://marketplace.visualstudio.com/items?itemName=ms-python.python

What tools will we use today (continued)

- Ethereum Execution Spec Tests framework:
  - This is the main framework to develop tests for the Ethereum protocol
  - Bookmark this link since it will be used extensively!

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

What tools will we use today (continued)

- A running Ethereum Devnet!
  - We will be running our tests in a live ethereum network deployed only for the purposes this workshop, as we need the tools to explore and monitor what happens in the chain.
  - Bookmark this link as we will be using it extensively

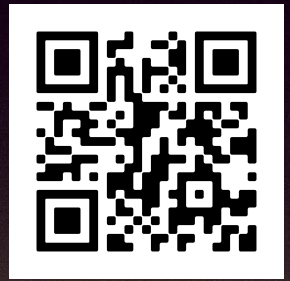
http://eest-devnet.ethpandaops.io/

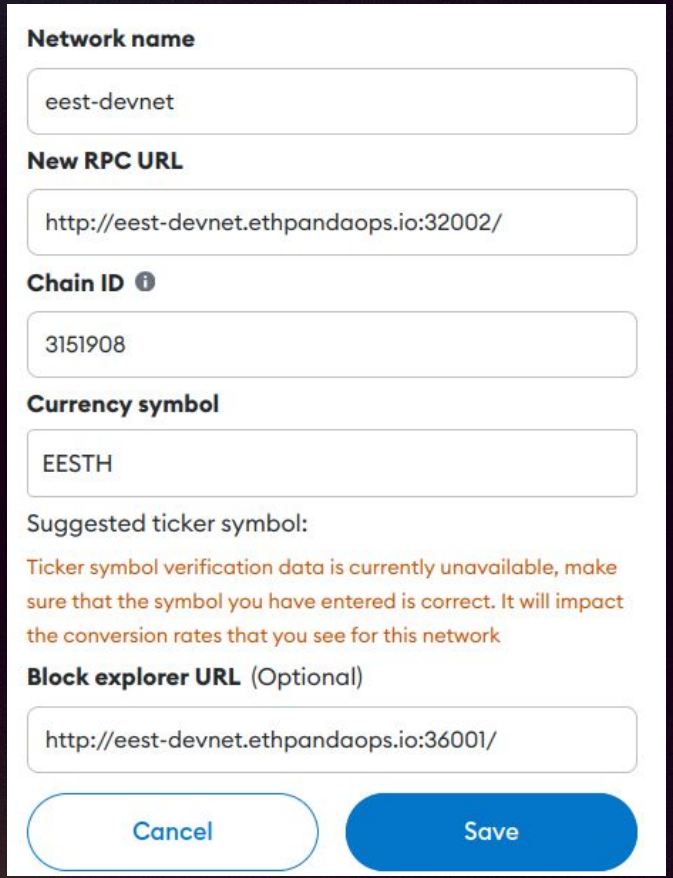
Special thanks to the ethPandaOps team for their help with this deployment!

What tools will we use today (continued)

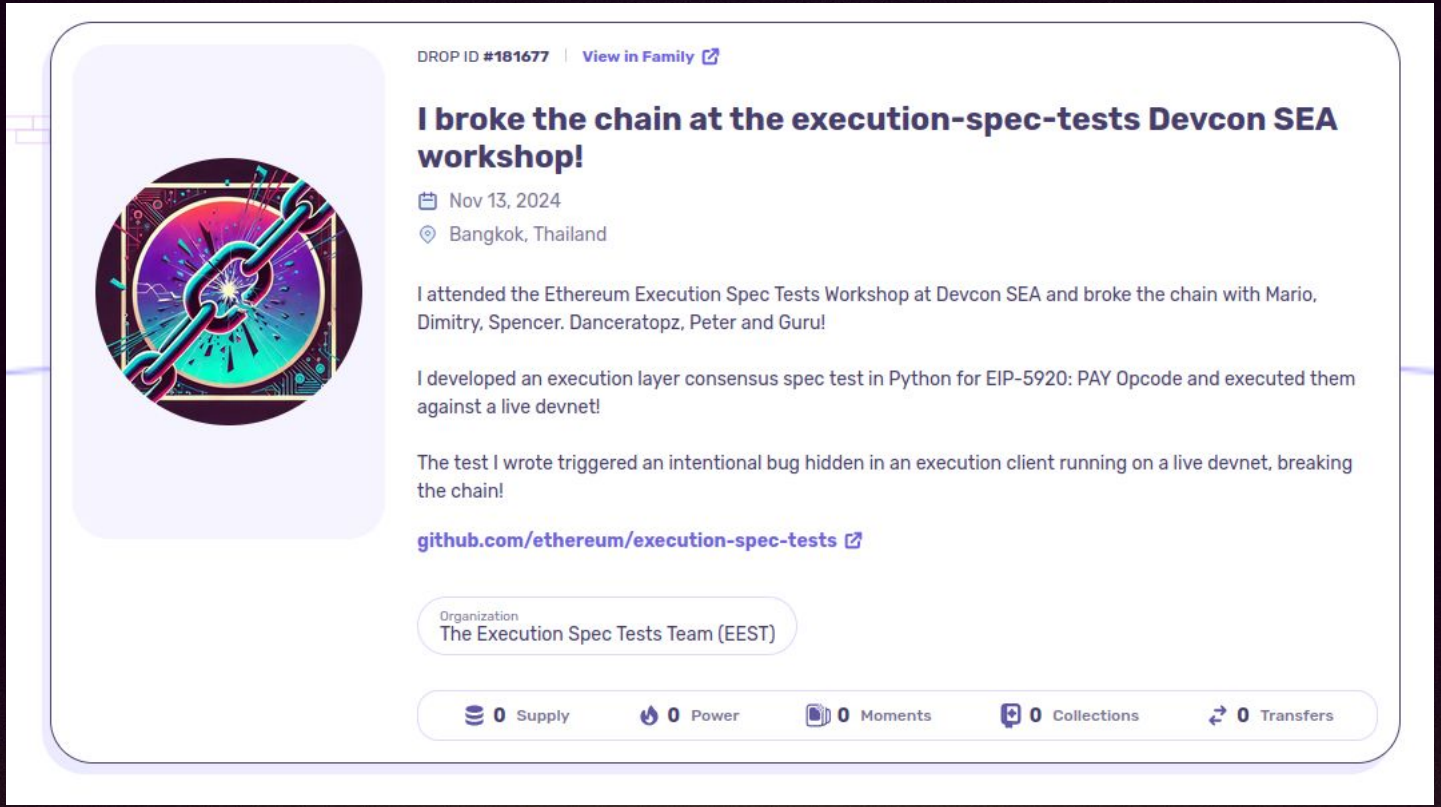
- [Optional] Add the devnet to metamask:
  - It's possible to add this network to metamask in order to use it to send transactions to the network

https://metamask.io/download





POAPs for “hunting the bug” and “breaking the chain” :)



The theory.

Execution Spec Tests - Ethereum Virtual Machine tests repository

- Python source code
- Powered by pytest and provides simple to complex parametrization
- Requires execution-specs (EELS) to fill tests
- Capable of filling and verifying tests, and running tests against clients or even live networks.

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

Parts of a Test

- Setup (pre-state)
  - Smart contracts with code to execute
  - Pre-funded accounts to send transactions
- Action (transaction)
  - Transaction(s) that execute an action over the pre-state
  - Can be a single transaction or many
- Verification (post-state)
  - Checks that the virtual machine state mutated into the expected form
  - Usually by checking balances and storage of accounts

Let’s get started.

Initial Setup

- Install uv on your machine by running the following command:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

- (Might not be necessary) Reload bash or run the following command to be able to run `uv`

```
source $HOME/.cargo/env
```

Initial Setup (continued 1)

- Fetch the `execution-spec-tests` repository:

```
git clone --depth 1 https://github.com/ethereum/execution-spec-tests
```

- Move into the `execution-spec-tests` folder:

```
cd execution-spec-tests
```

Initial Setup (continued 2)

- Create the python virtual environment

```
uv venv --python 3.12
```

- Perform the initial `uv` sync

```
uv sync --all-extras
```

- Verify you can run EEST commands

```
uv run fill --collect-only
```

Initial Setup (continued 3)

- (Optional) Open the created folder in your IDE

```
code .
```

The simplest test

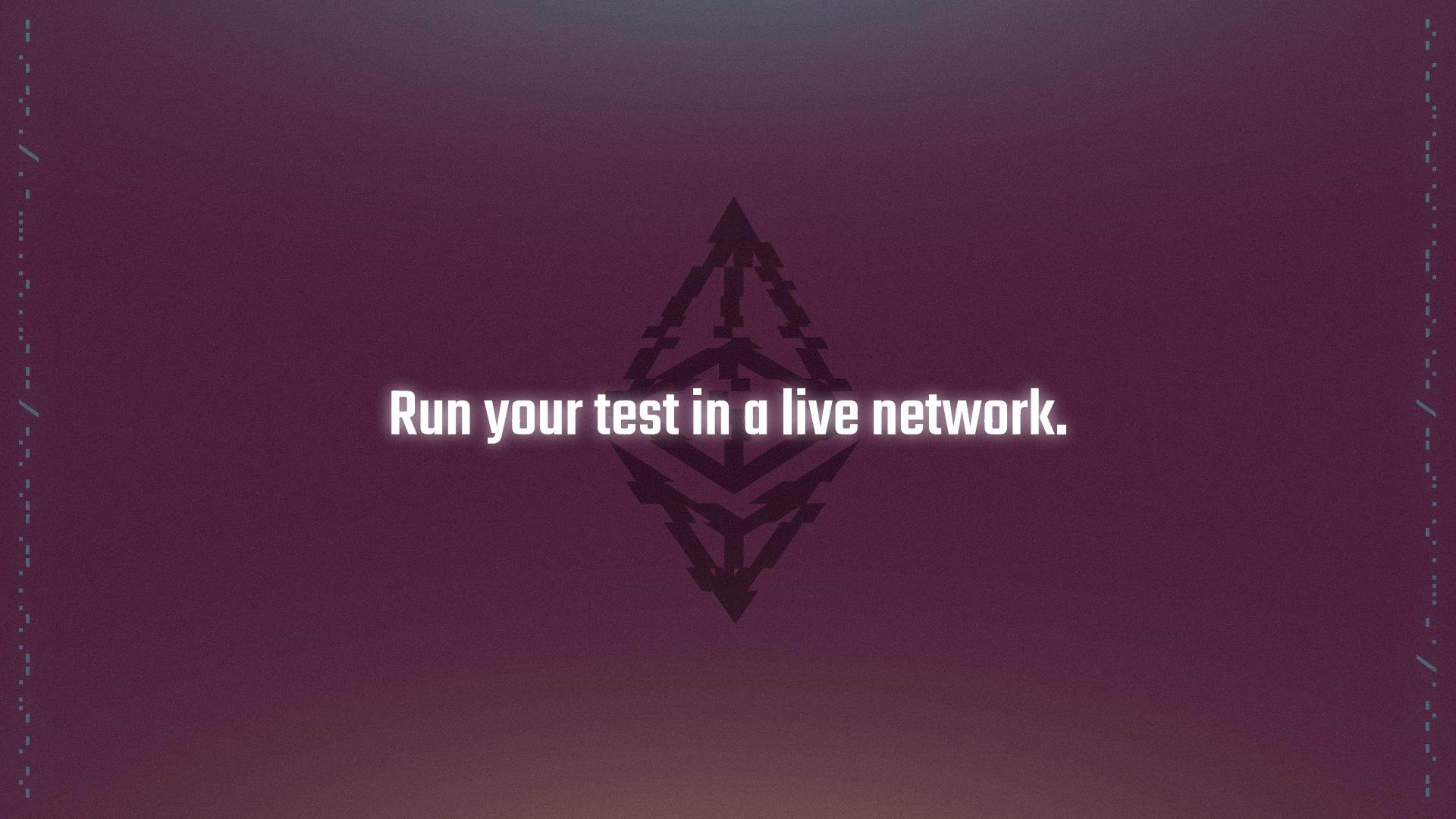
https://gist.github.com/marioevz/fa297d5da592b52d7890a2266e539e73

Create this file in the `tests/cancun` subfolder

Filling a simple test

- To fill a test, the `fill` command is used.
- Fill the test using the following command:

```
uv run fill --fork=Cancun ./tests/cancun/test_simplest.py
```


Run your test in a live network.

The Live Network

http://eest-devnet.ethpandaops.io/

Executing a simple test on a live network: Generate a key

- First generate a test account to be able to send transactions

```python
"""
Script to generate an Ethereum private key along with its address.

This key is NOT secure, please do NOT use it for means other than testing.
"""

import random

from ethereum_test_tools import EOA

eoa = EOA(key=random.randint(0, 2**256))
print(eoa.key)
print(eoa)
```

https://gist.github.com/marioevz/2fd79e986befbd711d4a7da7913d42af

Note: First printed hexadecimal number is the private key, second one is the address

Executing a simple test on a live network: Get funds

- The `execute` command uses the private key to generate the transactions that run on the devnet, but it needs funds to be able to do so
- Use the faucet to obtain funds
- http://eest-devnet.ethpandaops.io:8080/
- If not possible give us your address and we will send funds to it

Note: You can check the balance of your account using the devnet block explorer: http://eest-devnet.ethpandaops.io:36001/

Executing a simple test on a live network: Execute the test

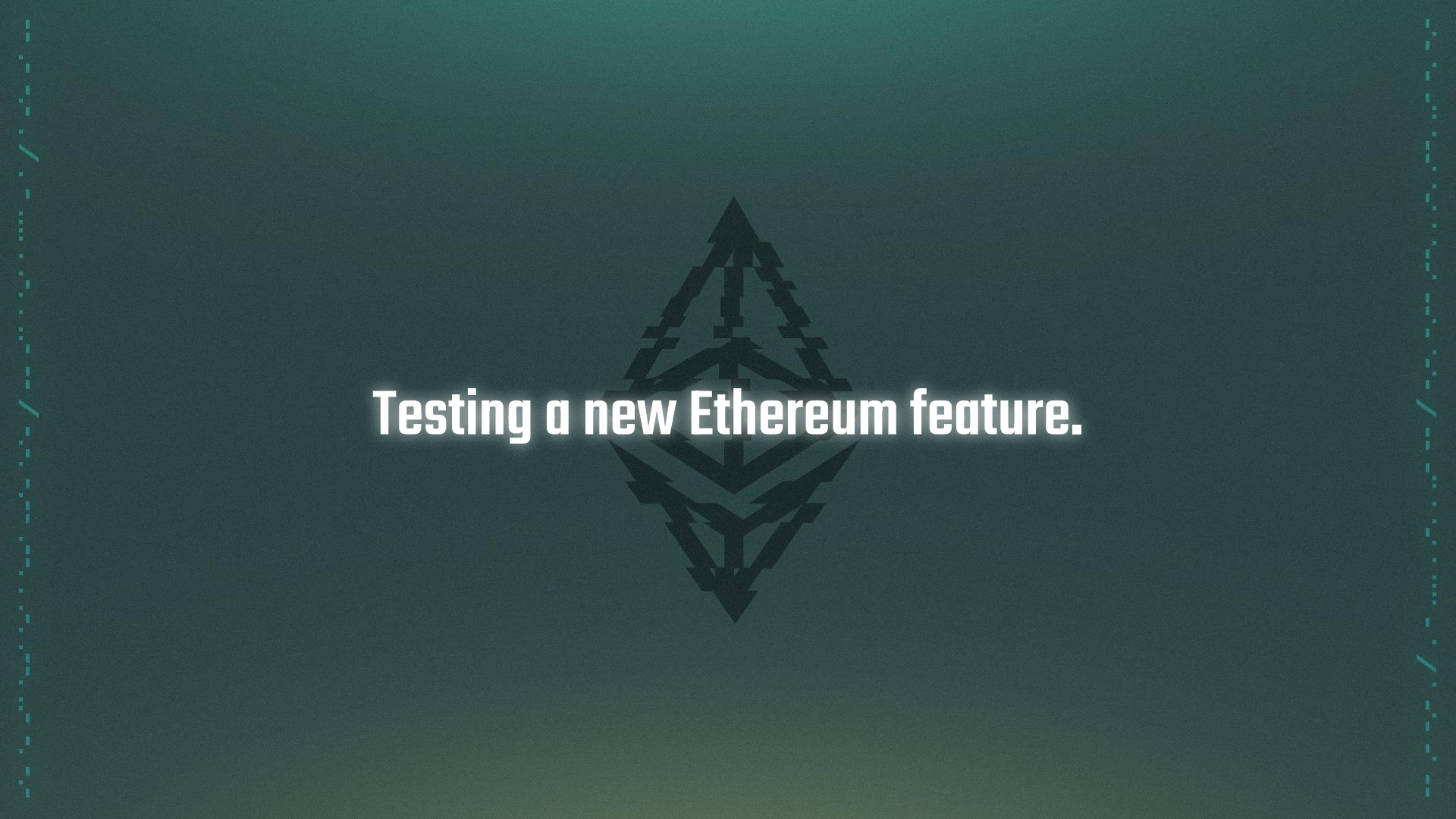
- Run the `execute` command to send the transactions that comprise the test to the live testnet

```
uv run execute remote --rpc-seed-key <PRIVATE_KEY> --rpc-endpoint
http://eest-devnet.ethpandaops.io:32002 --seed-account-sweep-amount
"10 eth" --rpc-chain-id 3151908 --fork Cancun
./tests/cancun/test_simplest.py
```

- Rpc-endpoint can be either:
  - http://eest-devnet.ethpandaops.io:32002
  - http://eest-devnet.ethpandaops.io:32007

Modifying the simple test

- We can do some modifications to the simple test:
  - Change the smart contract
  - Add smart contracts
  - Add parametrization
  - Change the expected outcomes


Testing a new Ethereum feature.

Ethereum Improvement Proposals (EIPs)

- The EIPs are the updates to the Ethereum protocol that are included in each fork
- They can be found in the following repository: https://github.com/ethereum/EIPs

EIP-5920: PAY Opcode

- Today we are focusing on a change that is not yet included in any fork: EIP-5920
- Includes a new opcode that sends transfers Ether to another account without executing any of its code
- Normally we use `CALL` or `DELEGATECALL` opcodes to send value, but this also triggers code execution.

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-5920.md

Writing tests for a new fork: Osaka

- When testing a new fork, it needs to be added to `src/ethereum_test_forks/forks/forks.py` in the EEST repository
- In the case of Osaka, the fork already exists.

Writing tests for a new Opcode

- If an EIP adds a new opcode, we need to create it in `src/ethereum_test_vm/opcode.py`
- For EIP-5920 the opcode PAY has to be added with binary value of 0xf9

https://gist.github.com/marioevz/8a6ba200a220a89ddaae0379347f5754

Writing a new test for EIP-5920

- Simplest test for EIP-5920

https://gist.github.com/marioevz/53454ebd39c4fdcec624eef180245edf

Executing the new test for EIP-5920

- Using the `execute` command to send the transactions that comprise the test to the live testnet

```
uv run execute remote --rpc-seed-key <PRIVATE_KEY> --rpc-endpoint
http://eest-devnet.ethpandaops.io:32002 --seed-account-sweep-amount
"10 eth" --rpc-chain-id 3151908 --fork Osaka
./tests/osaka/eip5920_pay/test_pay.py::test_pay
```

- Rpc-endpoint can be either:
  - http://eest-devnet.ethpandaops.io:32002
  - http://eest-devnet.ethpandaops.io:32007

Expanding the EIP-5920 test

- We can expand testing of the EIP-5920 with the following tests:
  - Test sending more balance than what the account has
  - Test sending balance to a precompile
  - Test running without enough gas
  - Test sending balance to self
  - Try doing an opcode stack underflow

Finding the EIP-5920 potential consensus issues

- What should the error be when the opcode is called without balance?

What can we learn from this.

Mitigating consensus issues

- All clients must pass the consensus tests before joining a devnet
- If a client passes the tests, but a bug was found in their code, the tests must be updated.
- Specs can sometimes be underspecificed, if such an issue is found during testing, the EIP must be updated to be more descriptive

Writing tests saves the chain

- We invite everyone to verify the tests that secure Ethereum
- If you can write a test that finds a potential consensus issue, you might be eligible for a bounty: https://ethereum.org/en/bug-bounty/

Thank you!

EF Testing

Dan: @danceratopz

Dimitry: @winsvega

Mario: @marioevz

Spencer: @spencer-tb

Ethereum Foundation