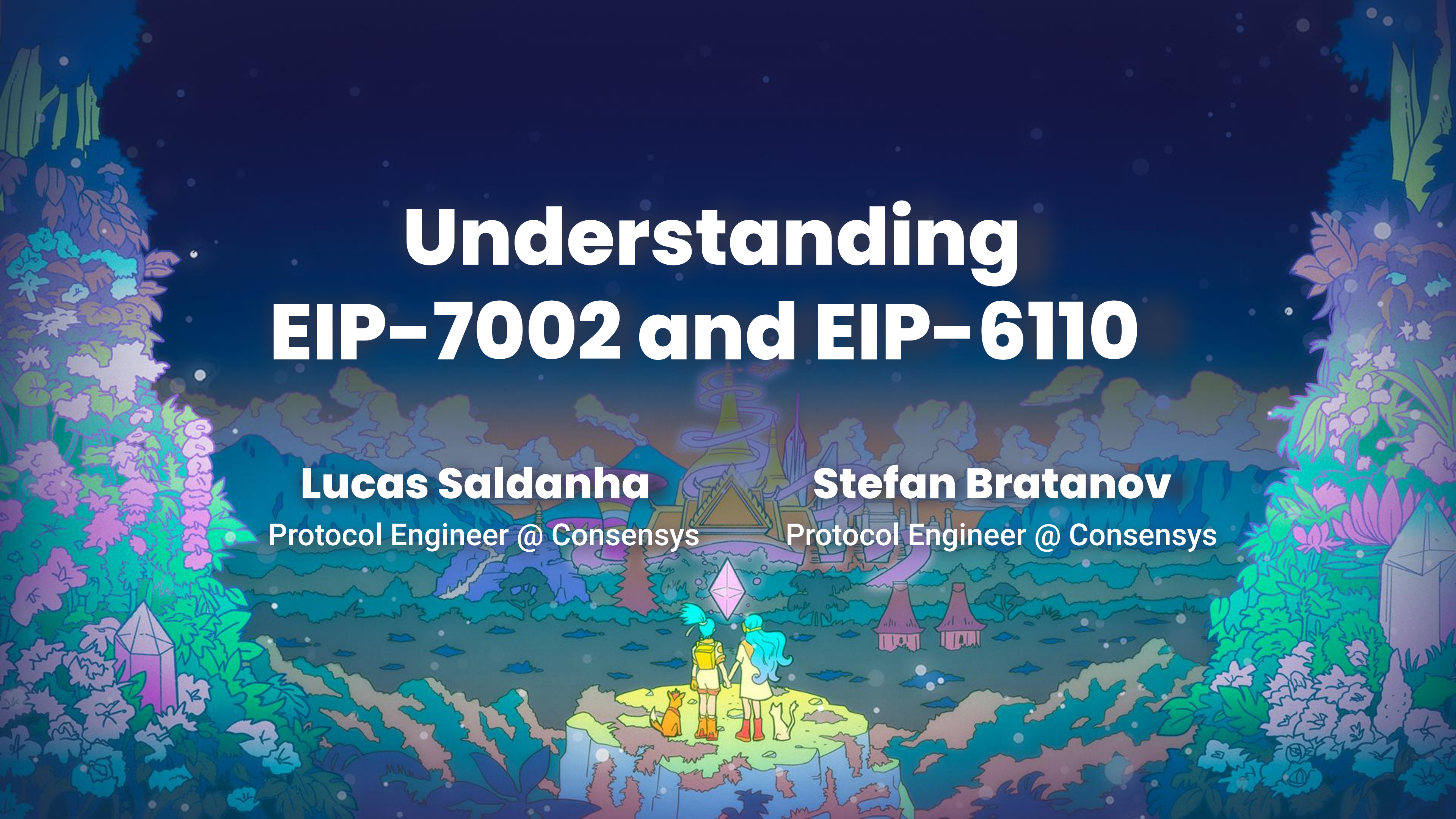


Understanding EIP-7002 and EIP-6110

Lucas Saldanha
Protocol Engineer @ Consensys

Stefan Bratanov
Protocol Engineer @ Consensys

EIP-7002
Execution Layer
Triggerable Withdrawals

A little bit of context

- Ethereum Validators
  - Attest and propose blocks on the network
  - Stake Ether to join (32 Ether minimum)

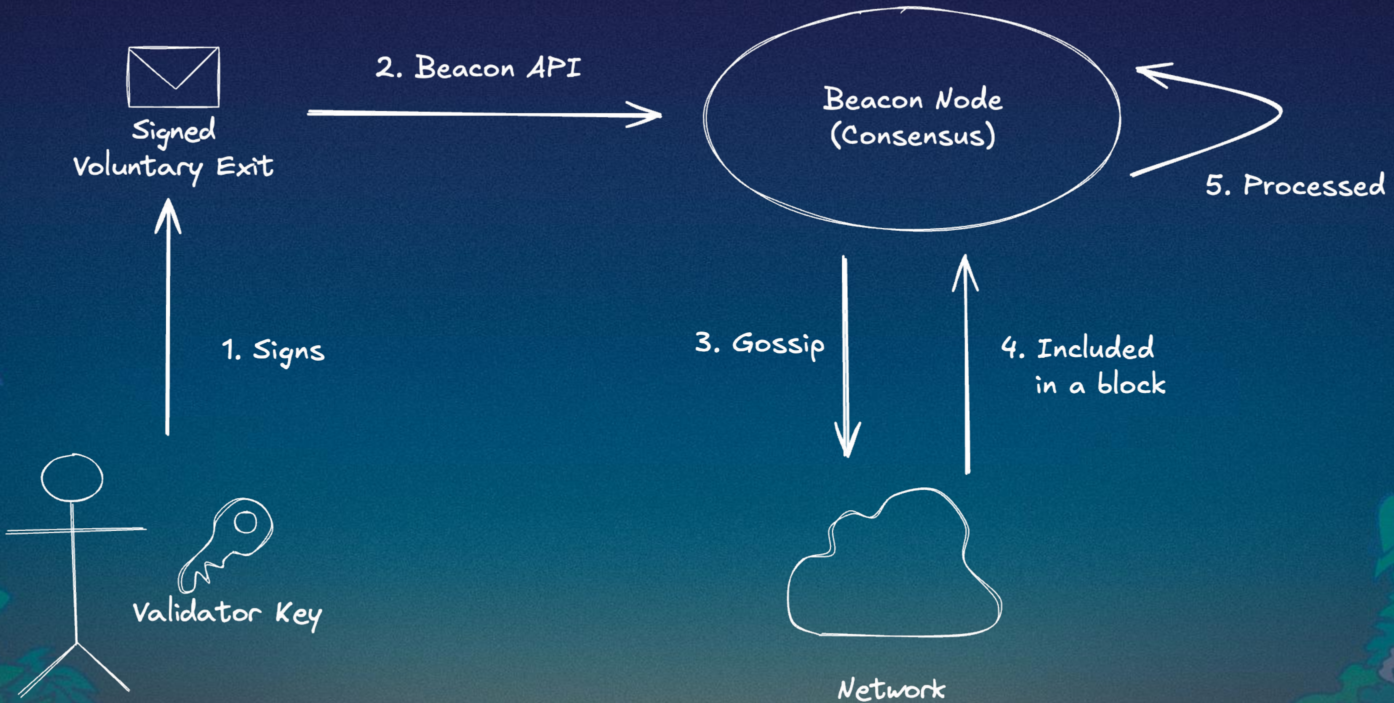
A little bit of context

- Ethereum Validators
  - Attest and propose blocks on the network
  - Stake Ether to join (32 Ether minimum)
- Validator Credentials
  - Validator Key: used to perform validator duties
  - Withdrawal Key: owner of staked amount

A little bit of context

- Ethereum Validators
    - Attest and propose blocks on the network
    - Stake Ether to join (32 Ether minimum)
- Validator Credentials
    - Validator Key: used to perform validator duties
    - Withdrawal Key: owner of staked amount
- Solo Staking vs Delegated Staking
    - Different kinds of arrangement: custodial, non-custodial, etc.

Voluntary Exit (Phase 0) - Recap



Problem:

Only the Validator Key can be used to exit a validator

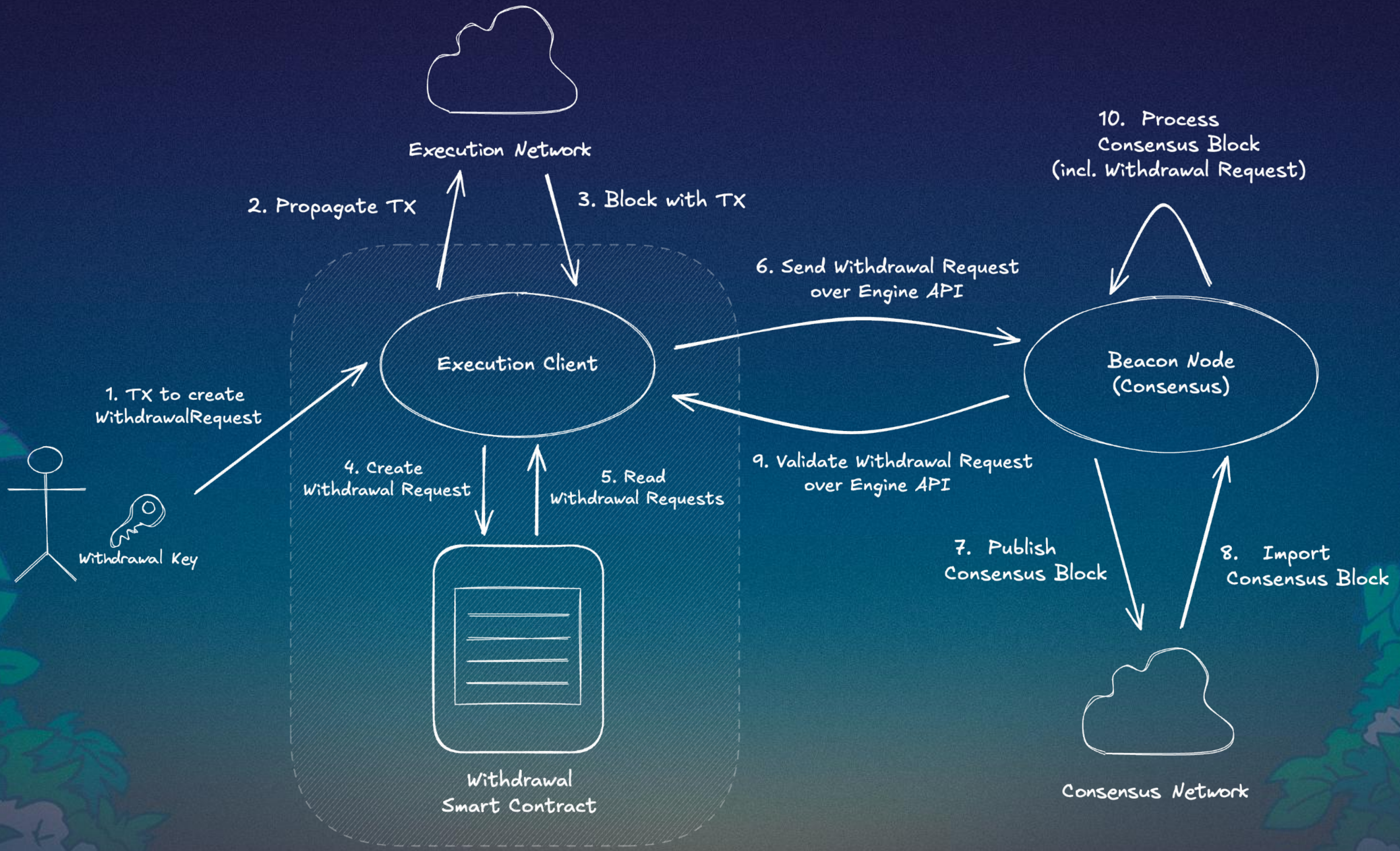
Solution:

Allow both Validator Key and Withdrawal Key to exit a validator

EIP-7002: Execution Layer Triggerable Withdrawals

- Execution Layer and Consensus Layer have different world views
- Design a mechanism to create a message on the Execution side and send it to the Consensus side to be processed
    - Authentication happens on EL side (msg.caller)
    - Authorization happens on CL side (processing request)
- Withdrawal Request (source_address, validator_pubkey, amount)
    - Full Withdrawal (amount = 0) ⟶ add validator to exit queue
    - Partial Withdrawals (amount > 0) ⟶ create a pending withdrawal for the amount

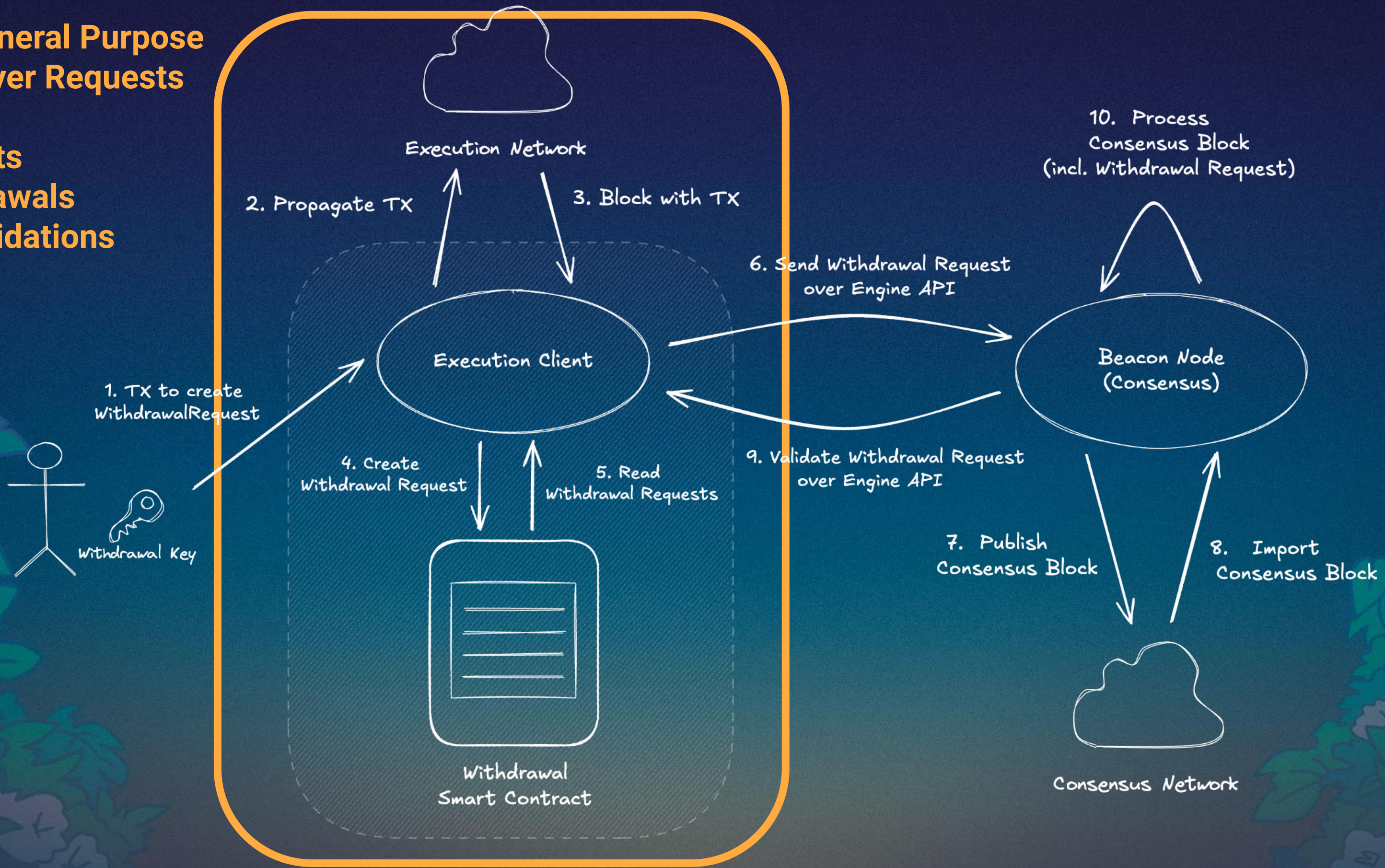
Withdrawal Requests Creation/Processing (EIP-7002)


Execution Network
2. Propagate TX
3. Block with TX
10. Process Consensus Block (incl. Withdrawal Request)
6. Send Withdrawal Request over Engine API
Execution Client
Beacon Node (Consensus)
1. TX to create WithdrawalRequest
9. Validate Withdrawal Request over Engine API
4. Create Withdrawal Request
5. Read Withdrawal Requests
Withdrawal Key
7. Publish Consensus Block
8. Import Consensus Block
Withdrawal Smart Contract
Consensus Network

Withdrawal Requests Creation/Processing (EIP-7002)

EIP-7685: General Purpose Execution Layer Requests

- Deposits
- Withdrawals
- Consolidations



Execution Network
2. Propagate TX
3. Block with TX
Execution Client
1. TX to create WithdrawalRequest
Withdrawal Key
4. Create Withdrawal Request
5. Read Withdrawal Requests
Withdrawal Smart Contract
6. Send Withdrawal Request over Engine API
9. Validate Withdrawal Request over Engine API
Beacon Node (Consensus)
10. Process Consensus Block (incl. Withdrawal Request)
7. Publish Consensus Block
8. Import Consensus Block
Consensus Network

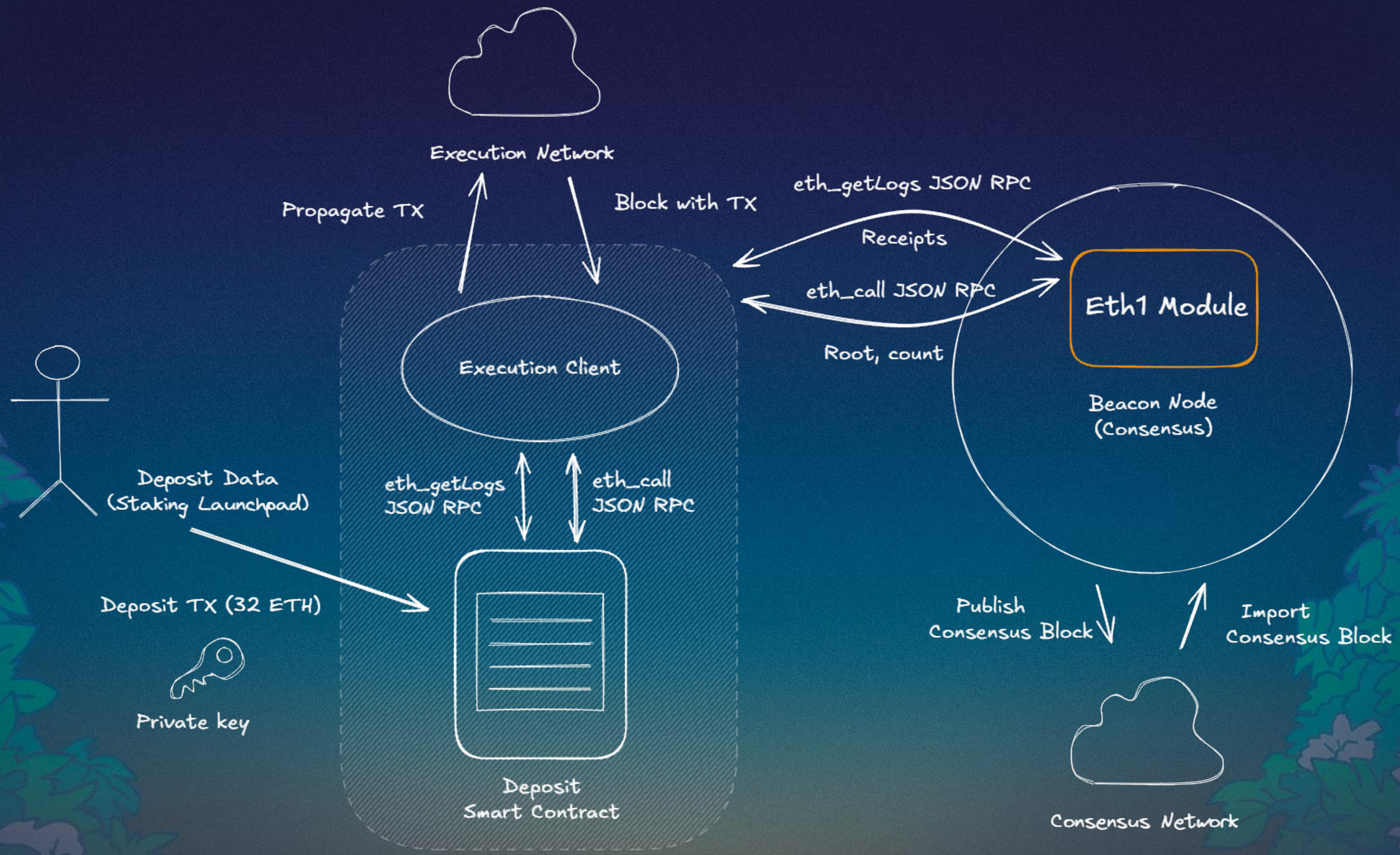
EIP-7002: Execution layer triggerable withdrawals (Cont.)

- Caveats
  - Successful request creation does not mean successful withdrawal
  - Validator withdrawal credentials using contract addresses
  - Withdrawal credentials cannot be changed (for now)
- Does not replace Voluntary Exits
- Withdrawal Requests != Withdrawals (Capella)

EIP-6110
Supply Validator Deposits on Chain

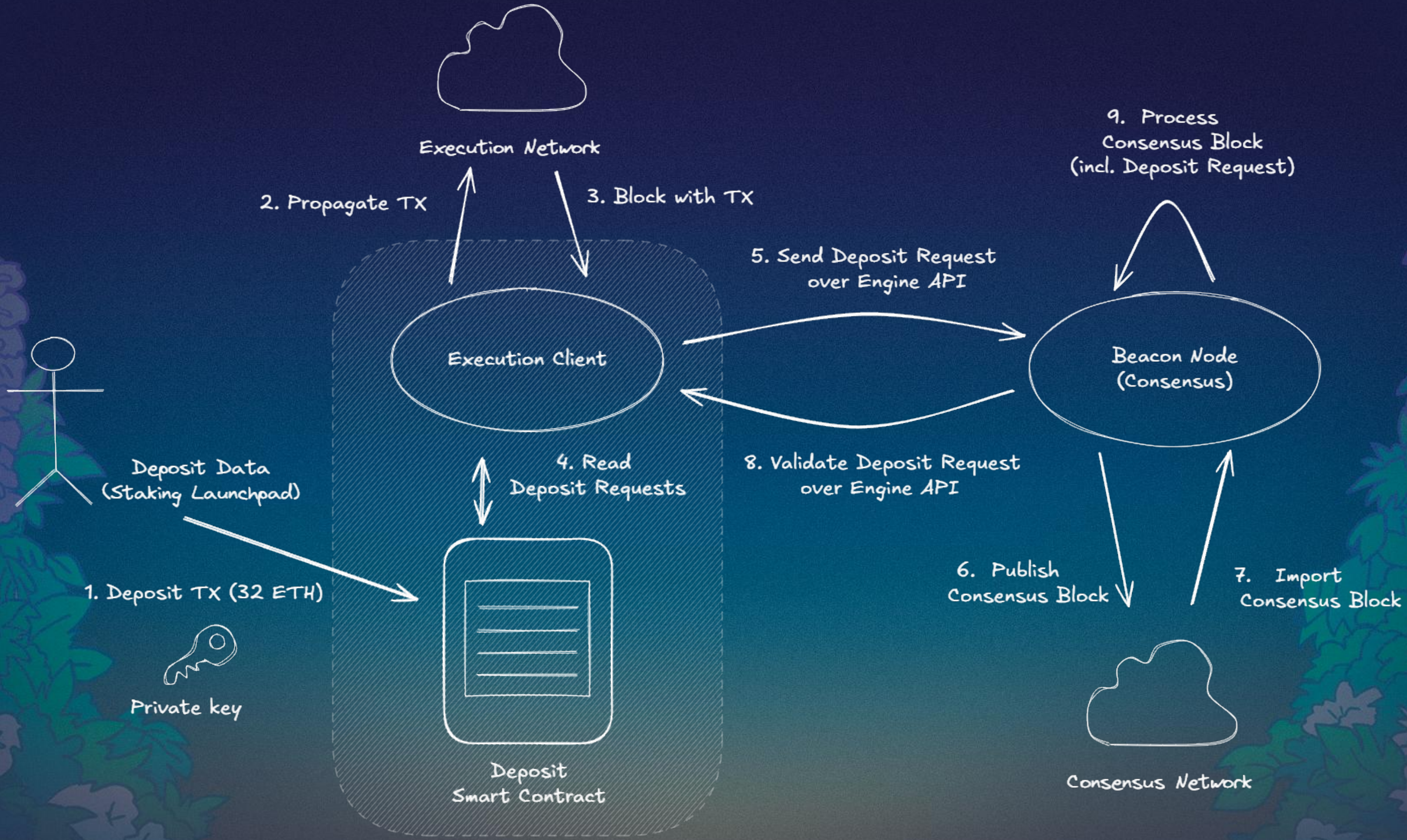
Current Deposit Processing

- A user submits a transaction (32 ETH) containing deposit data to the deposit contract. (e.g. via Staking Launchpad)
- The consensus layer considers the state of the deposit contract after an ~8 hour delay (2048 PoW blocks).
- The voting (64 epochs) is an off-protocol consensus mechanism designed before the Merge to ensure validators agree on the same view of the deposit contract.
- When voting is finished (half number of votes are the same), proposers include deposits alongside inclusion proofs in blocks that all nodes can verify.
- A (valid) deposit will either create a new validator on the consensus layer, or top up the balance of an existing validator.


Execution Network
Propagate TX
Block with TX
eth_getLogs JSON RPC
Receipts
eth_call JSON RPC
Root, count
Eth1 Module
Beacon Node
(Consensus)
Execution Client
eth_getLogs
JSON RPC
eth_call
JSON RPC
Deposit Data
(Staking Launchpad)
Deposit TX (32 ETH)
Private key
Deposit
Smart Contract
Publish
Consensus Block
Import
Consensus Block
Consensus Network

EIP-6110 Deposit Processing

- A user submits a transaction (32 ETH) containing deposit data to the deposit contract. (same as before)
- Deposits come directly from the EL via the Engine API, are included in blocks and processed on finalization of the chain (~13 minutes)
- A (valid) deposit will either create a new validator on the consensus layer, or top up the balance of an existing validator. (same as before)



Execution Network

2. Propagate TX

3. Block with TX

Execution Client

4. Read
Deposit Requests

Deposit
Smart Contract

Deposit Data
(Staking Launchpad)

1. Deposit TX (32 ETH)

Private key

5. Send Deposit Request
over Engine API

8. Validate Deposit Request
over Engine API

Beacon Node
(Consensus)

9. Process
Consensus Block
(incl. Deposit Request)

6. Publish
Consensus Block

7. Import
Consensus Block

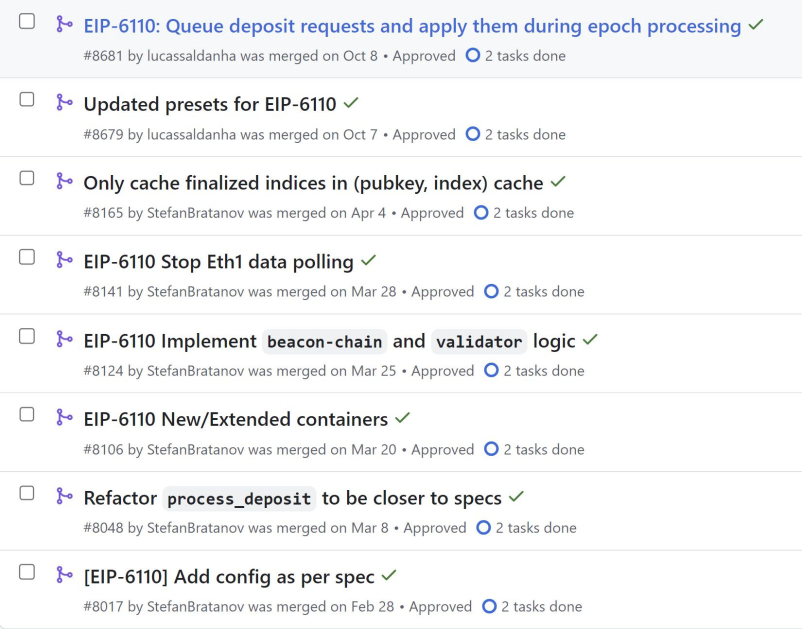
Consensus Network

Advantages of EIP-6110

- Delay of around ~11.4 hours (follow distance ~ 8h + voting ~ 3.4 h) reduced to around 2 epochs (~13 minutes)
- Increased security by inheriting the security of the chain rather than depending on off-protocol mechanism
- Eliminate requirement to maintain and distribute deposit contract snapshots (EIP-4881)
- No dependency on the JSON-RPC API (only the Engine API)
- Deprecate legacy and brittle part of the CL clients` code

Teku Implementation

https://github.com/Consensys/teku/pulls?q=EIP-6110



□ EIP-6110: Queue deposit requests and apply them during epoch processing ✓
#8681 by lucassaldanha was merged on Oct 8 • Approved ○ 2 tasks done

□ Updated presets for EIP-6110 ✓
#8679 by lucassaldanha was merged on Oct 7 • Approved ○ 2 tasks done

□ Only cache finalized indices in (pubkey, index) cache ✓
#8165 by StefanBratanov was merged on Apr 4 • Approved ○ 2 tasks done

□ EIP-6110 Stop Eth1 data polling ✓
#8141 by StefanBratanov was merged on Mar 28 • Approved ○ 2 tasks done

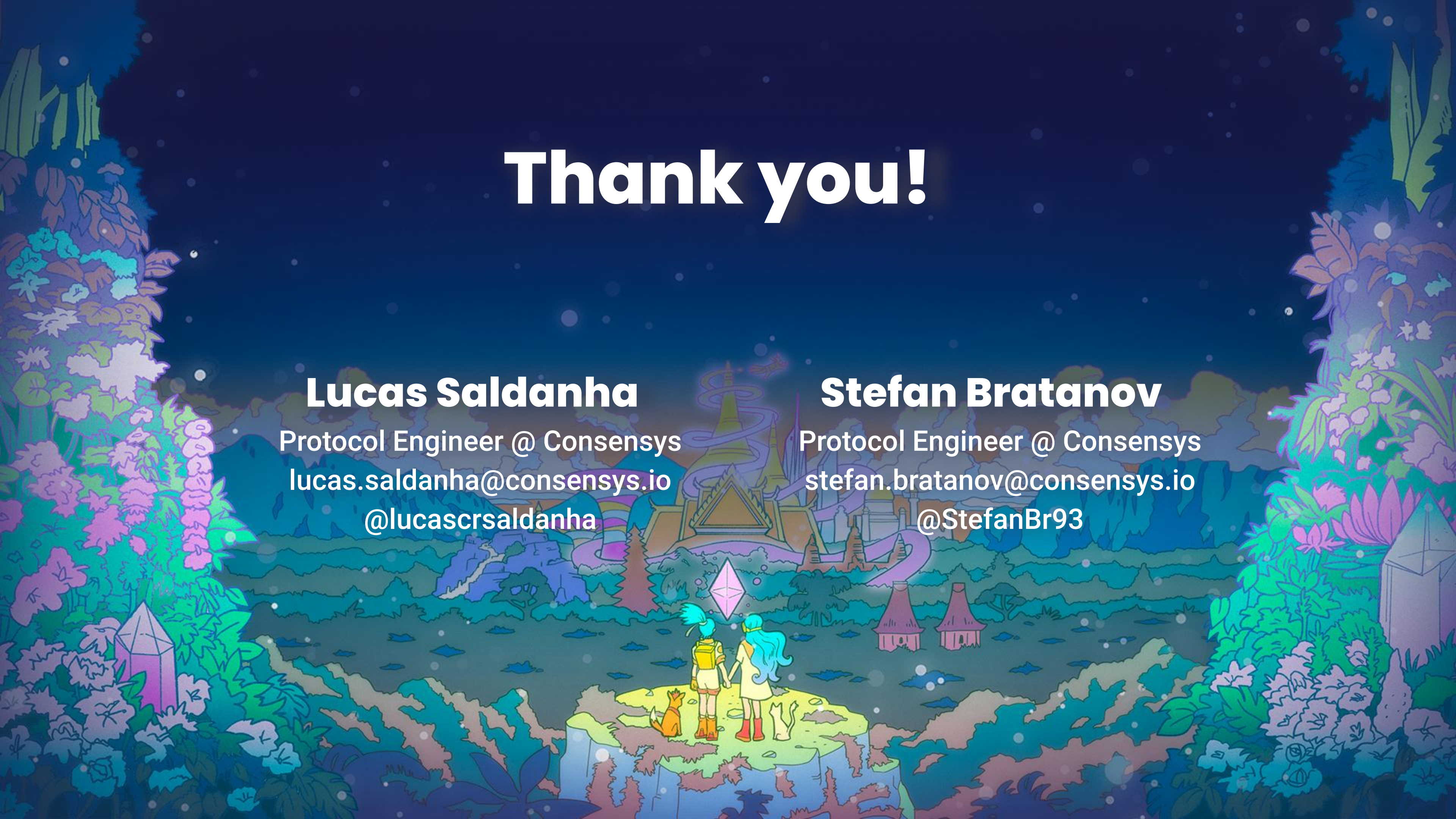
□ EIP-6110 Implement `beacon-chain` and `validator` logic ✓
#8124 by StefanBratanov was merged on Mar 25 • Approved ○ 2 tasks done

□ EIP-6110 New/Extended containers ✓
#8106 by StefanBratanov was merged on Mar 20 • Approved ○ 2 tasks done

□ Refactor `process_deposit` to be closer to specs ✓
#8048 by StefanBratanov was merged on Mar 8 • Approved ○ 2 tasks done

□ [EIP-6110] Add config as per spec ✓
#8017 by StefanBratanov was merged on Feb 28 • Approved ○ 2 tasks done

Questions?



Thank you!

Lucas Saldanha

Protocol Engineer @ Consensys
lucas.saldanha@consensys.io
@lucascrsaldanha

Stefan Bratanov

Protocol Engineer @ Consensys
stefan.bratanov@consensys.io
@StefanBr93

References

- https://eips.ethereum.org/EIPS/eip-7002
- https://kb.beaconcha.in/ethereum-staking/ethereum-2-keys
- https://ethereum2077.substack.com/p/eip-7002-execution-layer-exits
- https://dune.com/hildobby/eth2-staking
- https://university.cex.io/custodial-and-non-custodial-staking-explained
- https://eth2book.info/capella/part2/deposits-withdrawals/deposit-processing/
- https://eips.ethereum.org/EIPS/eip-6110