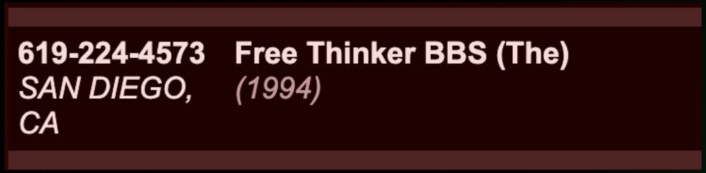


619-224-4573 *SAN DIEGO, CA*	**Free Thinker BBS (The)** *(1994)*

@HummusOnRails

Layer What??

Explaining the Architecture of Blockchain

Hi 👋, I'm Ben

Former BBS Sysop and current Head of Developer Relations at Fuel Network and Principal Consultant at Yalla, DevRel LLC.

→ @HummusOnRails

HUMMUS ON RAILS

FUEL

Our Journey Together

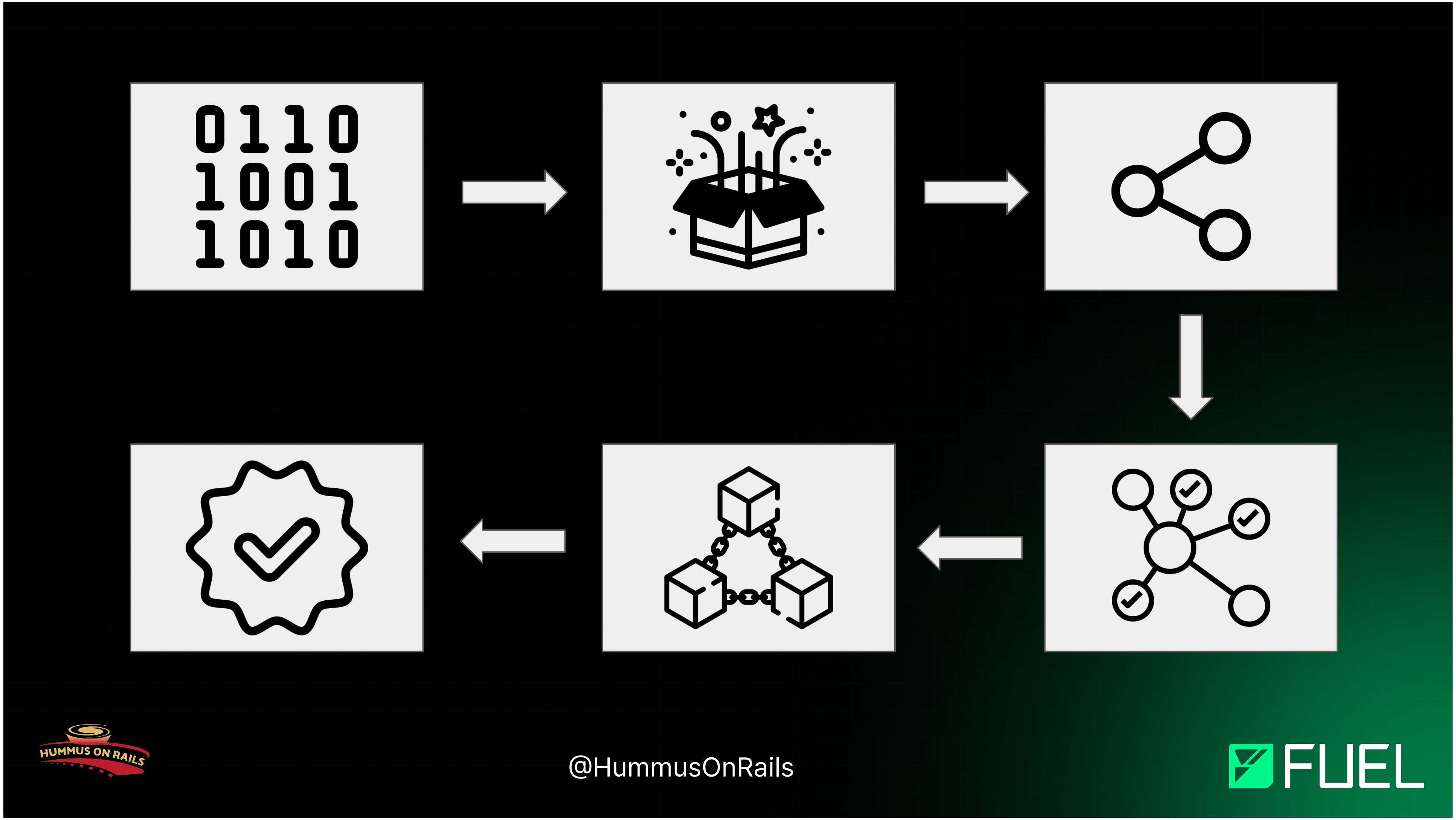
★ Blockchain 101
★ Network Structure
★ Consensus Mechanisms
★ Smart Contracts
★ dApps
★ Blockchain Layers

Blockchain 101

@HummusOnRails

What is a blockchain?

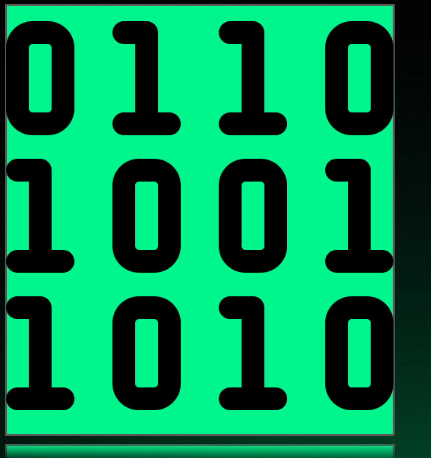
It’s a chain of blocks



@HummusOnRails

What can be sent?

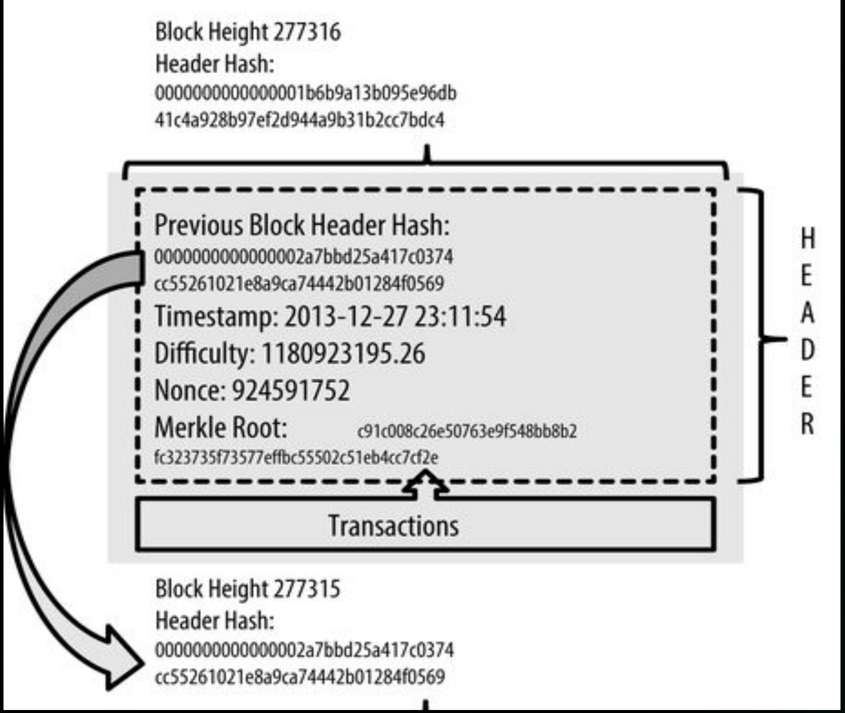
A blockchain can transmit any form of digital data, including but not limited to monetary transactions, smart contracts, images, documents and more.



What’s in a block?

A block contains a block header (with metadata such as the previous block's hash, a timestamp, and a nonce), a Merkle tree root hash representing the transactions included in the block, and a set of individual transaction records.

What’s in a block?



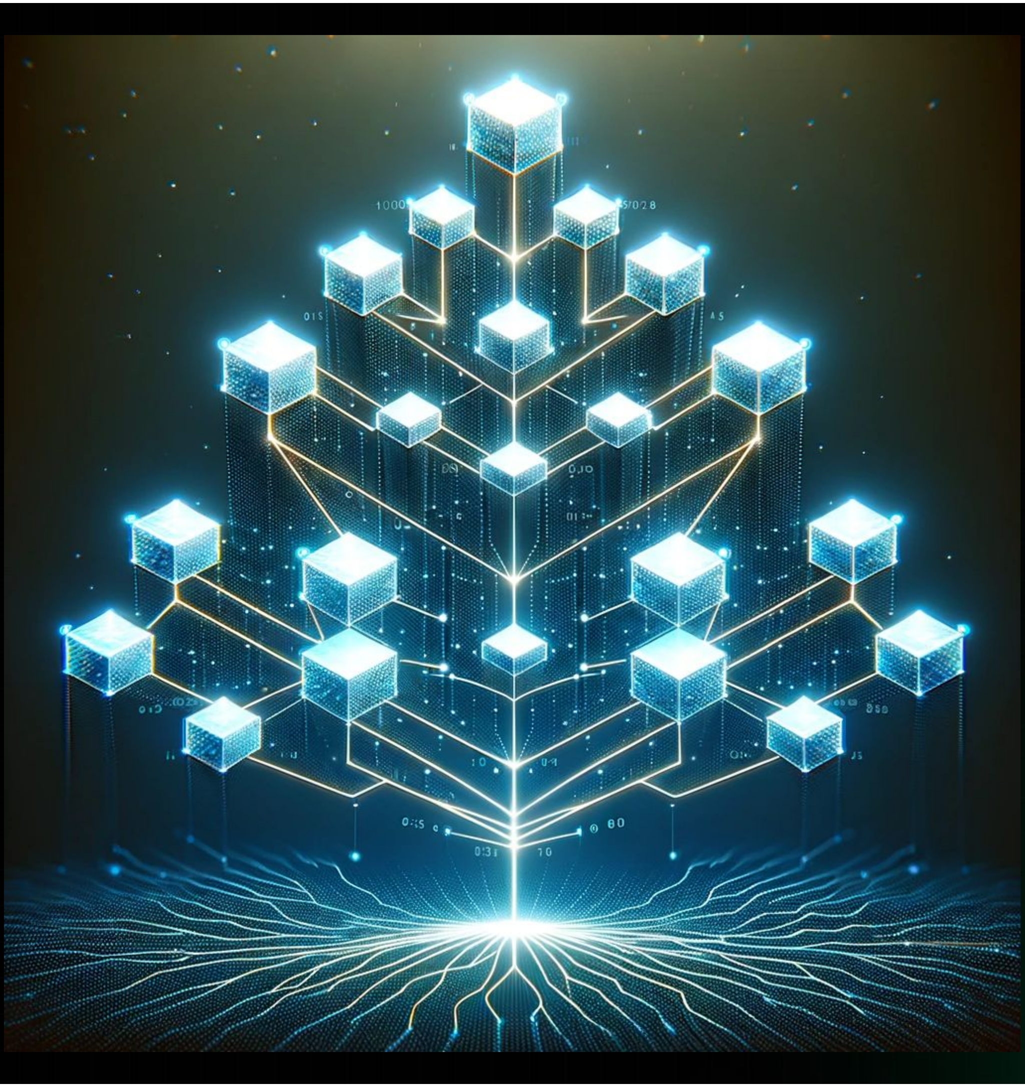
Block Height 277316
Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4

Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e

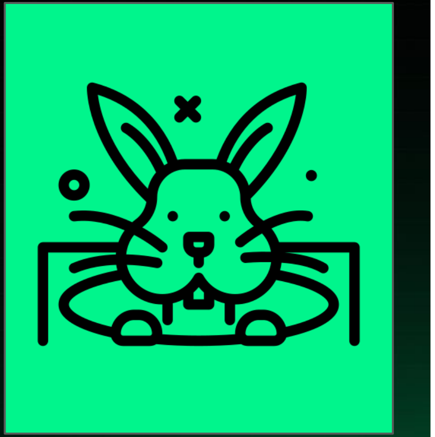
Transactions

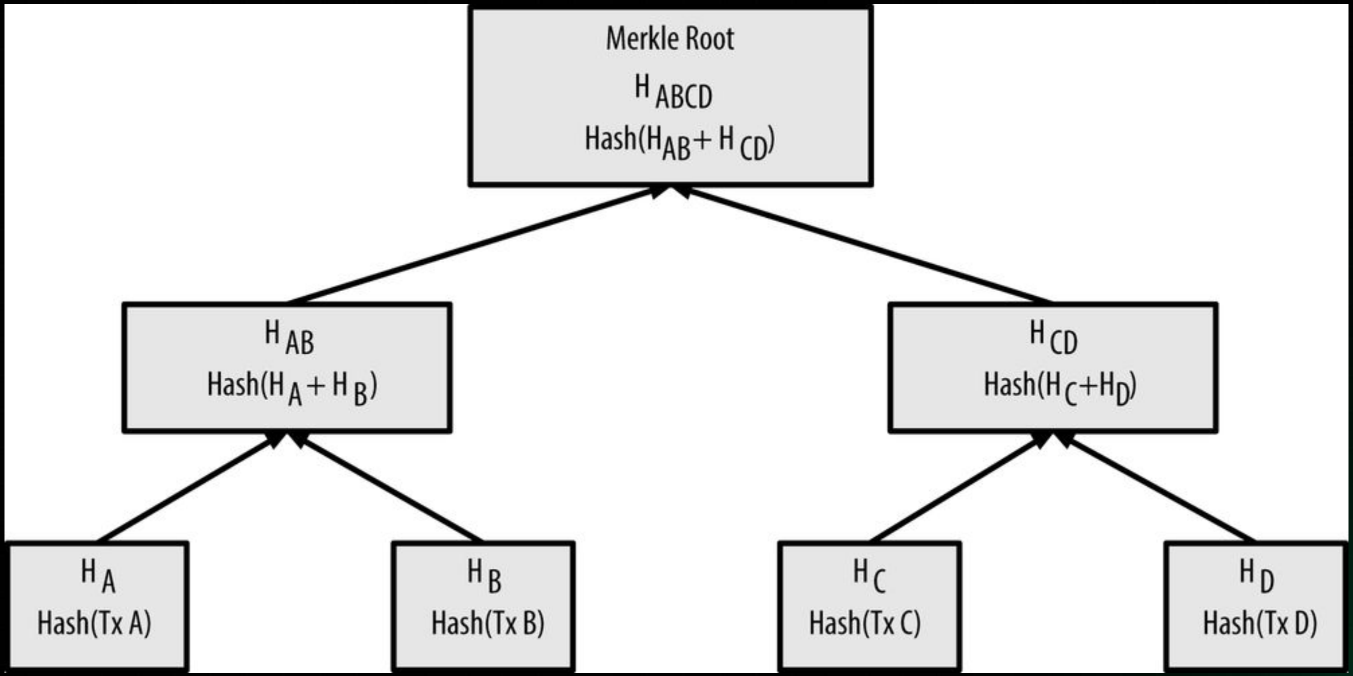
HEADER

Block Height 277315
Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569

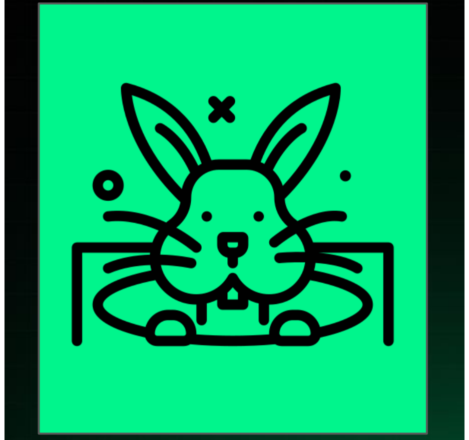


Down the rabbit hole...





Down the rabbit hole...



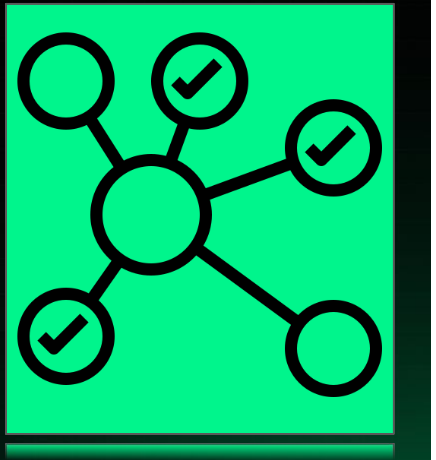
@HummusOnRails

Who validates?

Blockchain nodes are individual computers that validate and store a copy of the entire ledger, ensuring the integrity and consistency of the distributed network.

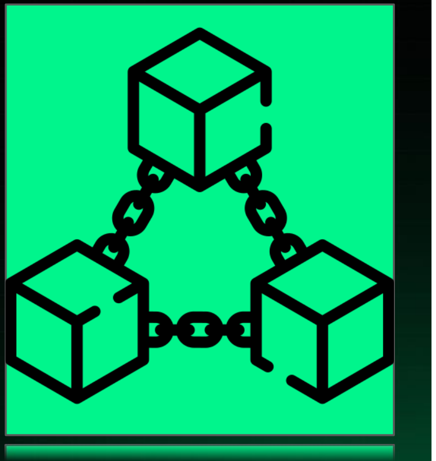
How are blocks validated?

Blocks are validated through a consensus mechanism, where nodes verify transaction integrity and collectively agree on the block's legitimacy.



What happens next?

Once validated, a block is added to the blockchain as a permanent record and then propagated to all network nodes to ensure a consistent and updated ledger across the system.



Finally…

After a block is added to the blockchain, the transactions within it are executed, updating user accounts and data, and triggering any relevant events or actions defined in smart contracts.

Network Structure

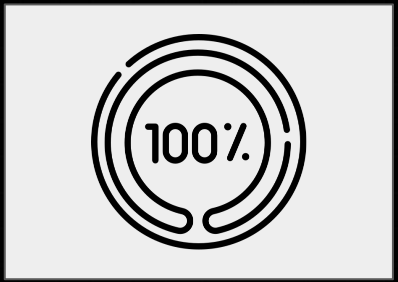
What is a node?

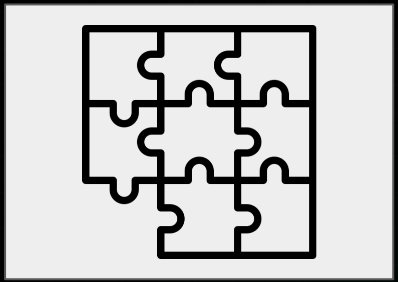
@HummusOnRails

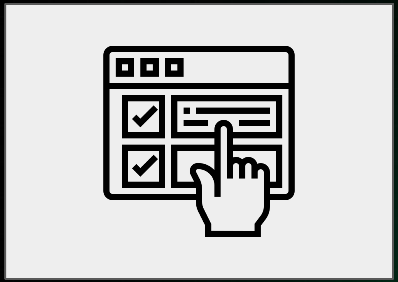
Your computer.

Your phone.

Your toaster.


100%



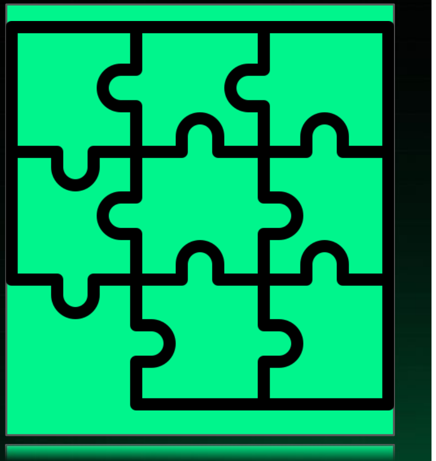


Full Nodes

These nodes maintain a complete copy of the blockchain ledger and validate each block and transaction according to the consensus rules of the network.

Light Nodes

These nodes do not store the entire blockchain but instead hold only essential information. They rely on full nodes for more detailed information, making them more suitable for devices with limited resources.

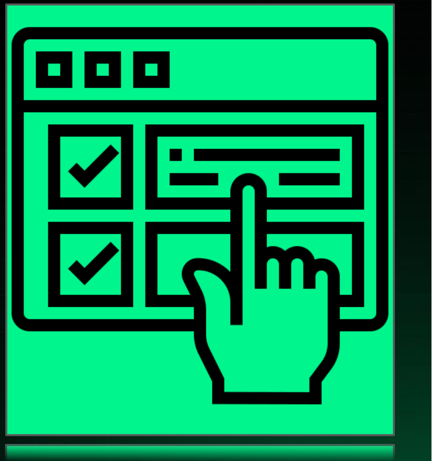


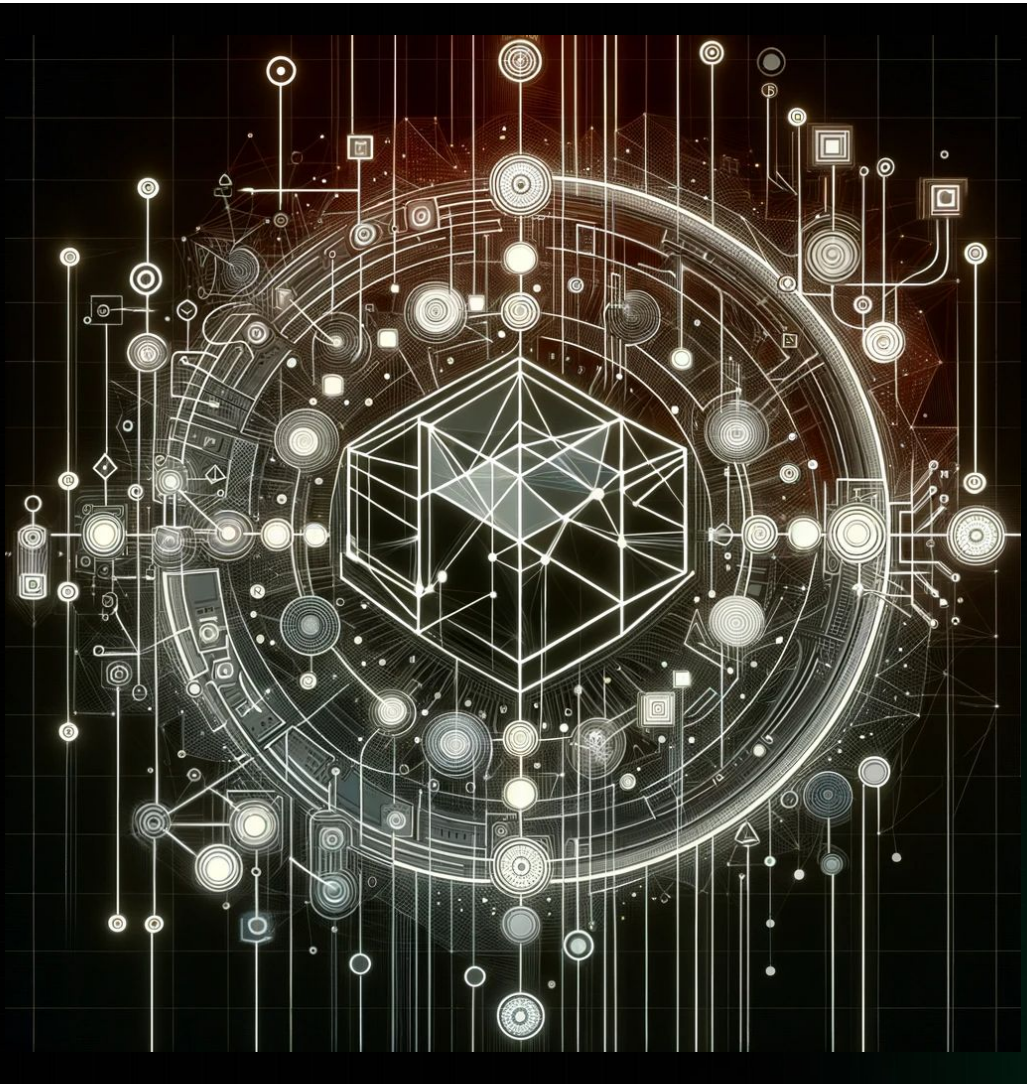
Archive Nodes

These nodes store the entire blockchain ledger, including historical states, which is beyond the requirements of a full node. They are useful for retrieving historical blockchain data.

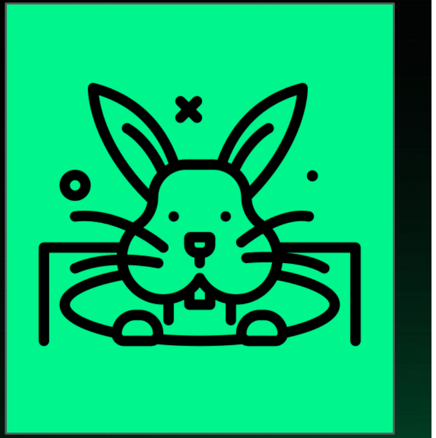
RPC Nodes

Nodes that provides an interface for interacting with the blockchain to send commands or requests, such as querying the current state of the blockchain, sending transactions, or invoking smart contract functions.



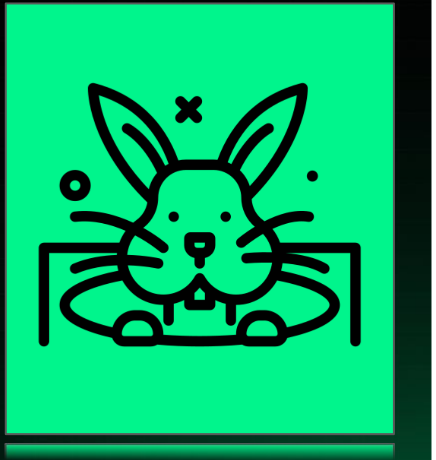


Down the rabbit hole...



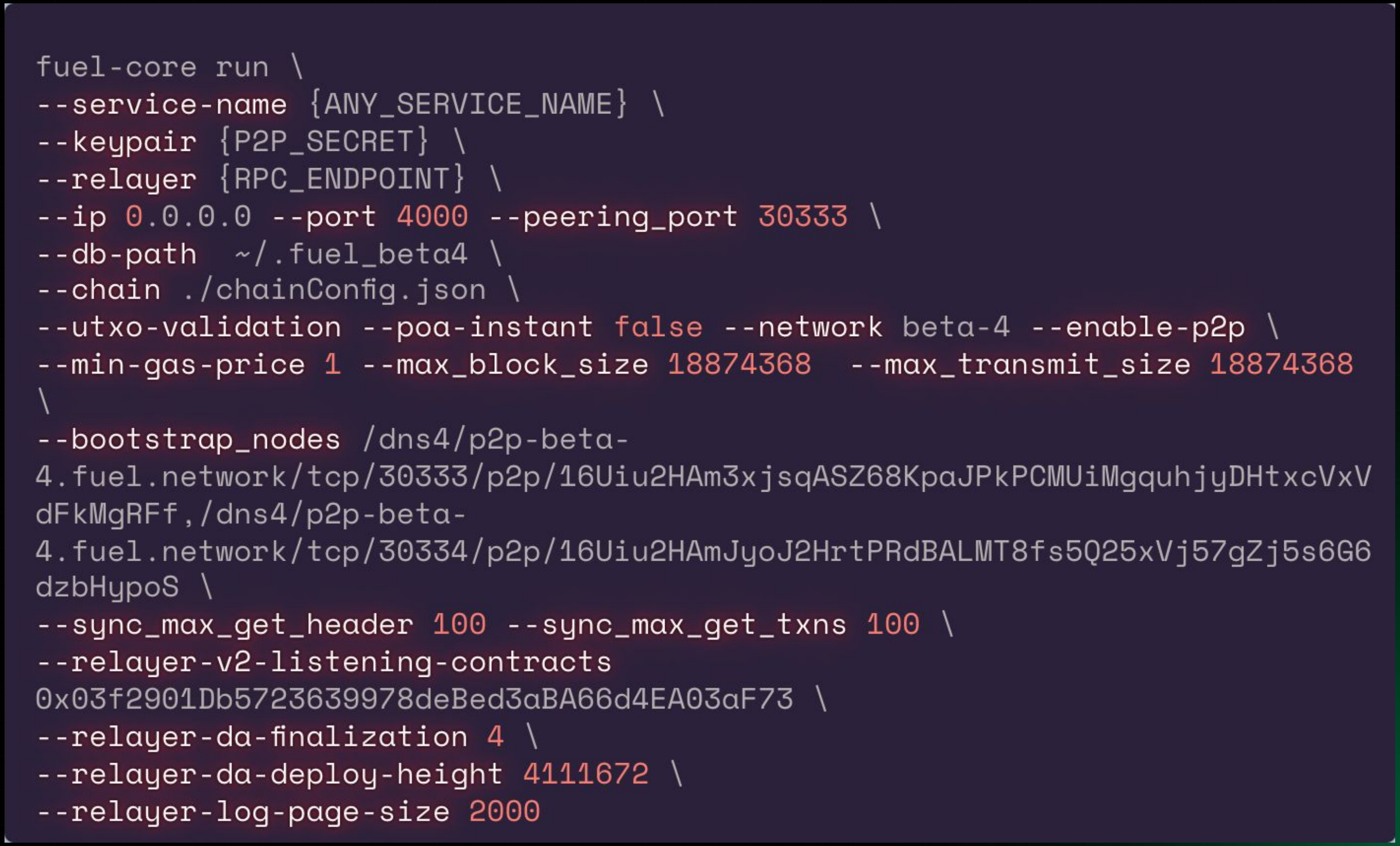
Remote Procedure Call (RPC)

Down the rabbit hole...



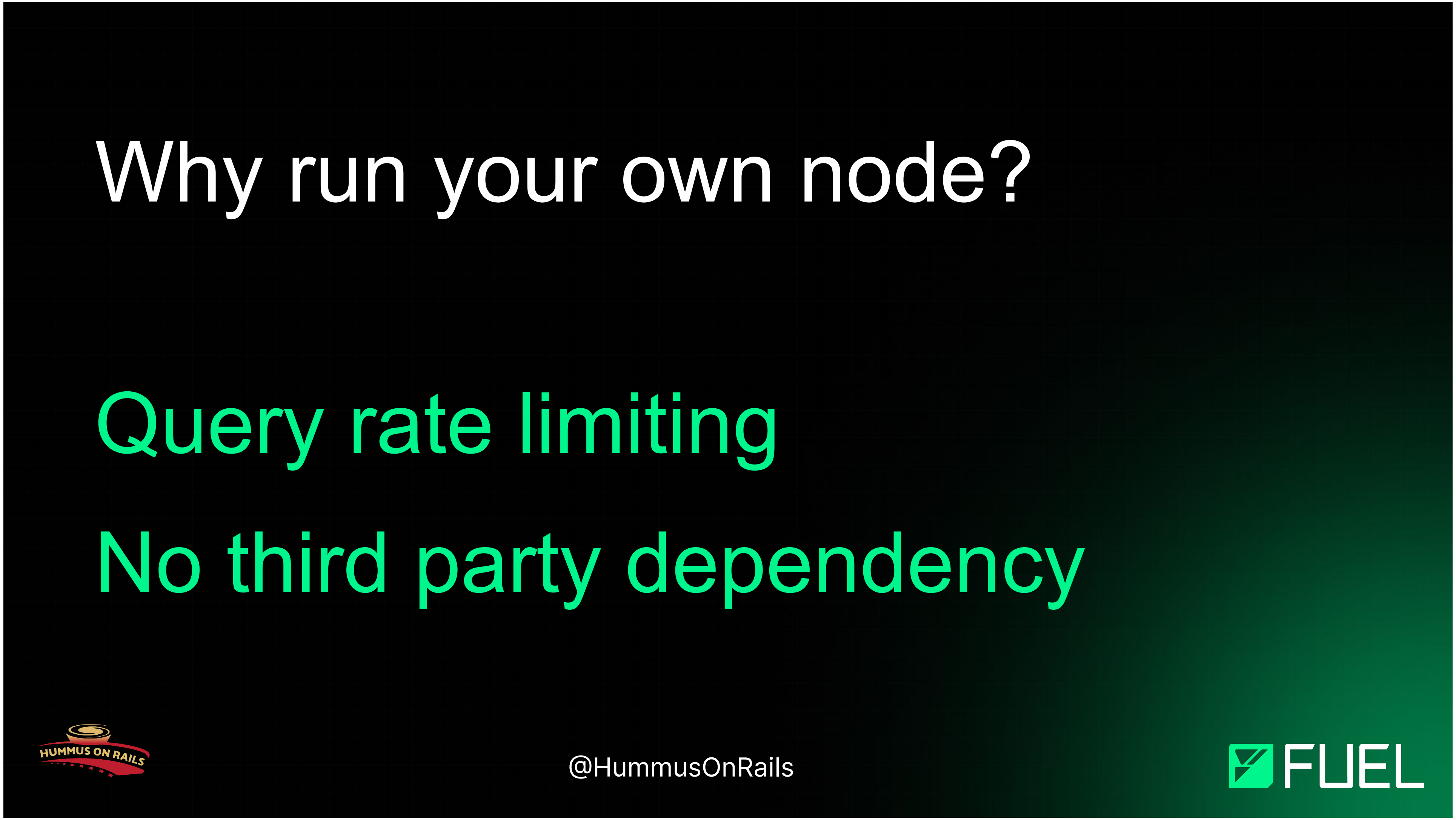
RPC enables remote execution of functions and procedures in a distributed network, allowing clients to call server-side procedures with local-like simplicity, streamlining data exchange and system integration for developers.

You can run your own node.



```
fuel-core run \
--service-name {ANY_SERVICE_NAME} \
--keypair {P2P_SECRET} \
--relayer {RPC_ENDPOINT} \
--ip 0.0.0.0 --port 4000 --peering_port 30333 \
--db-path  ~/.fuel_beta4 \
--chain ./chainConfig.json \
--utxo-validation --poa-instant false --network beta-4 --enable-p2p \
--min-gas-price 1 --max_block_size 18874368  --max_transmit_size 18874368
\
--bootstrap_nodes /dns4/p2p-beta-
4.fuel.network/tcp/30333/p2p/16Uiu2HAm3xjsqASZ68KpaJPkPCMUiMgquhjyDHtxcVxV
dFkMgRFf,/dns4/p2p-beta-
4.fuel.network/tcp/30334/p2p/16Uiu2HAmJyoJ2HrtPRdBALMT8fs5Q25xVj57gZj5s6G6
dzbHypoS \
--sync_max_get_header 100 --sync_max_get_txns 100 \
--relayer-v2-listening-contracts
0x03f2901Db5723639978deBed3aBA66d4EA03aF73 \
--relayer-da-finalization 4 \
--relayer-da-deploy-height 4111672 \
--relayer-log-page-size 2000
```

@HummusOnRails



Why run your own node?

Query rate limiting

No third party dependency

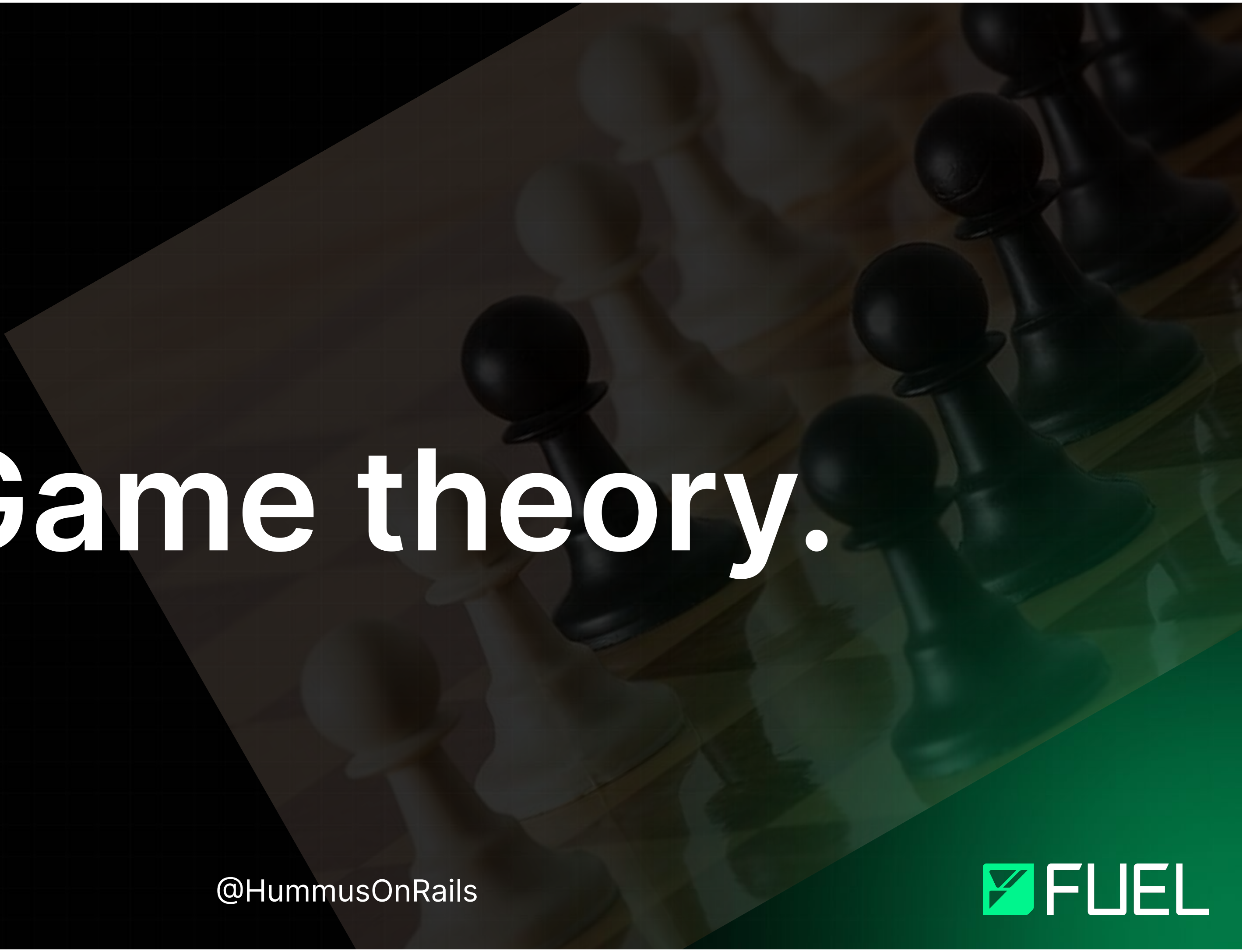
HUMMUS ON RAILS

@HummusOnRails

FUEL

Consensus Mechanisms

Game theory.



@HummusOnRails

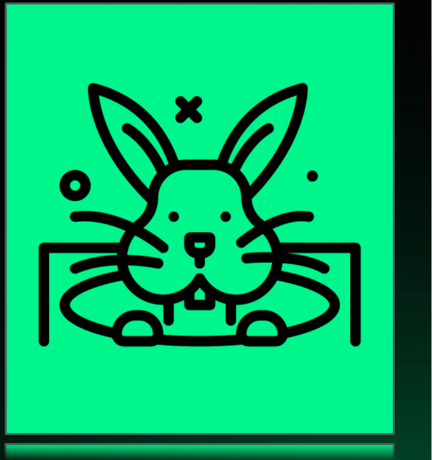
Two players choose to be either aggressive "hawks" or passive "doves".

Both choose hawk, they both face high costs due to conflict.

Hawk against dove, the hawk wins big, the dove gets nothing.

Both as doves, they share a moderate reward.

Down the rabbit hole...



Nash Equilibrium occurs when participants cannot benefit by changing their decision unilaterally, creating a state of stable, mutual strategy.

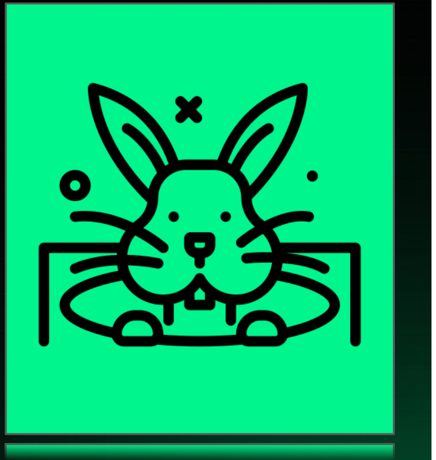
Players in “The Game”

- Validators
- Delegators
- Transactors
- Developers
- Node Operators
- Governance Participants
- Miners
- Investors

Down the rabbit hole...

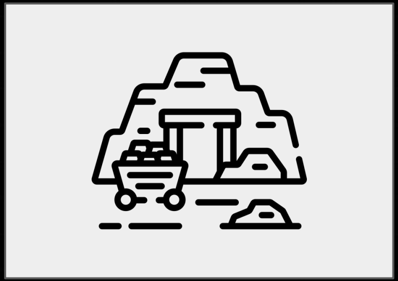
Byzantine Fault Tolerance

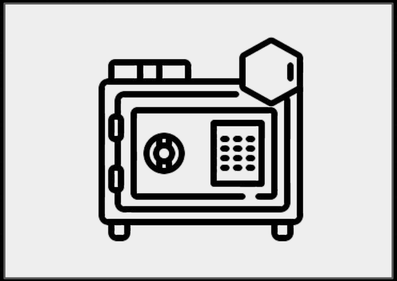
Down the rabbit hole...

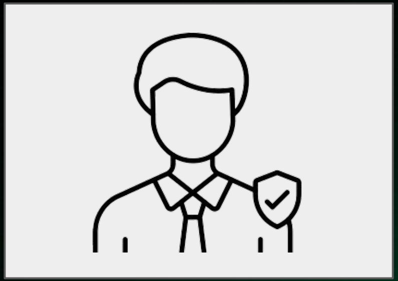


Byzantine Fault Tolerance is a system property that allows a distributed network to reach consensus and maintain functionality, even when some nodes fail or act maliciously, ensuring consistent and reliable protocol execution among participants.

@HummusOnRails

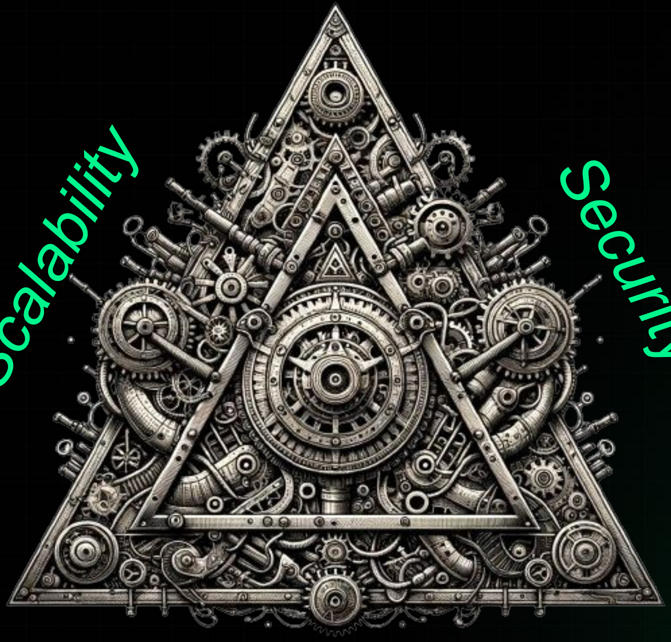






@HummusOnRails

Blockchain Trilemma



Decentralization

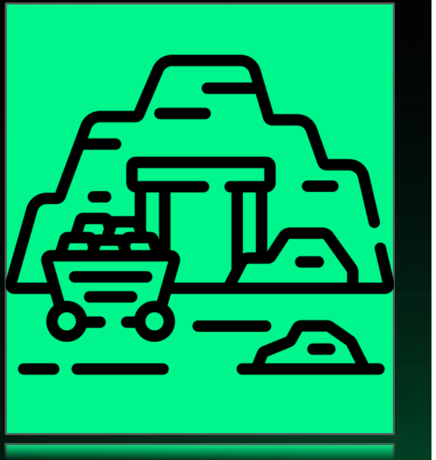
Scalability

Security

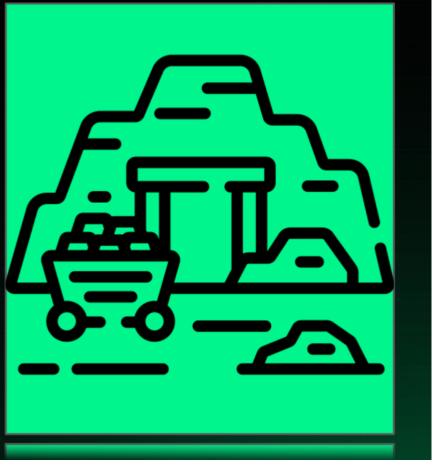
@HummusOnRails

Proof of Work

Proof of Work (PoW) necessitates solving cryptographic hash puzzles through brute-force computation, ensuring blockchain security and consensus by tethering block validation to CPU-intensive mining efforts.

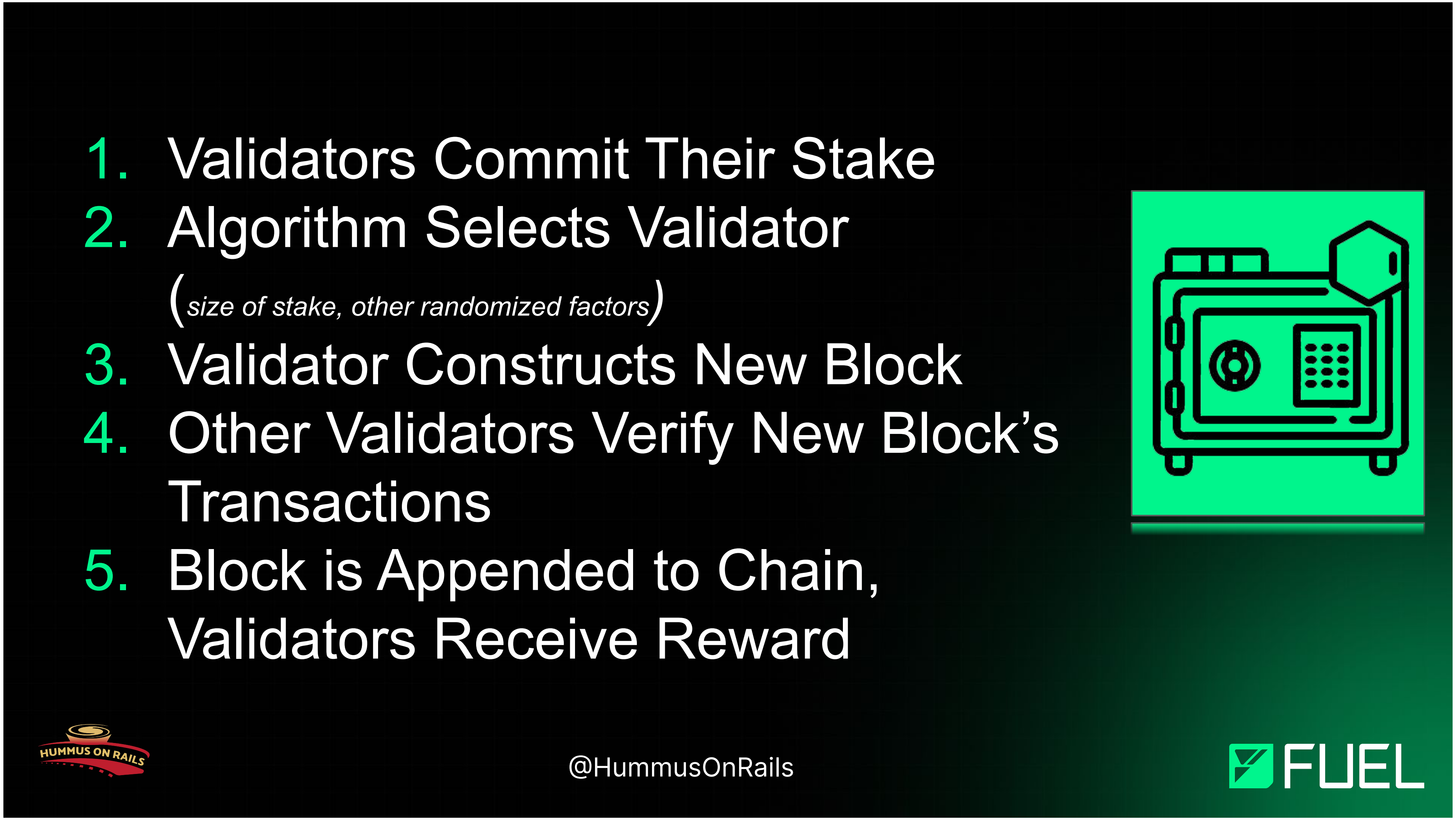


1. Complete Hash Function
2. Find Hash Value < Difficulty Target
3. Broadcast Solution to All Nodes
4. Nodes Verify Validity
    a. Correct Transaction Set
    b. Hash Below the Target
5. Block is Appended to Chain



Proof of Stake

A stake-weighted selection process for block validation, where validators are chosen based on the amount they commit to stake. Validators risk losing their stake for dishonest behavior. *(In Delegated Proof of Stake, stakeholders vote to elect a limited number of delegates to perform validation, optimizing for faster consensus and improved network throughput.)*



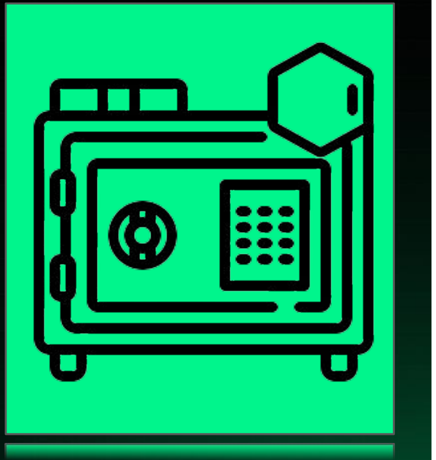
1. Validators Commit Their Stake
2. Algorithm Selects Validator
   (*size of stake, other randomized factors*)
3. Validator Constructs New Block
4. Other Validators Verify New Block’s Transactions
5. Block is Appended to Chain, Validators Receive Reward

HUMMUS ON RAILS

@HummusOnRails

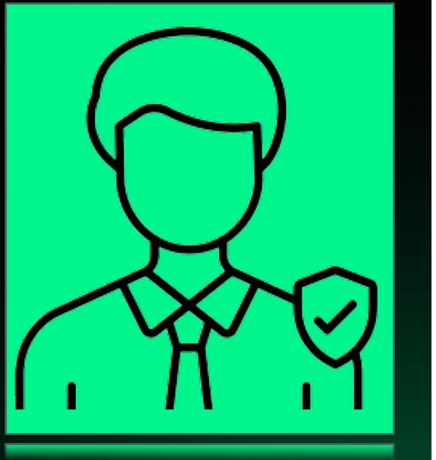
FUEL

1. Token Holders Commit Stake for Validators
2. Network Elects Validators
3. Validator Constructs New Block
4. Other Validators Verify New Block’s Transactions
5. Block is Appended to Chain, Validators and Token Holders Receive Reward



Proof of Authority

A consensus model where a limited number of pre-approved validators, identified through rigorous verification, create and validate blocks, offering a low-latency, energy-efficient mechanism



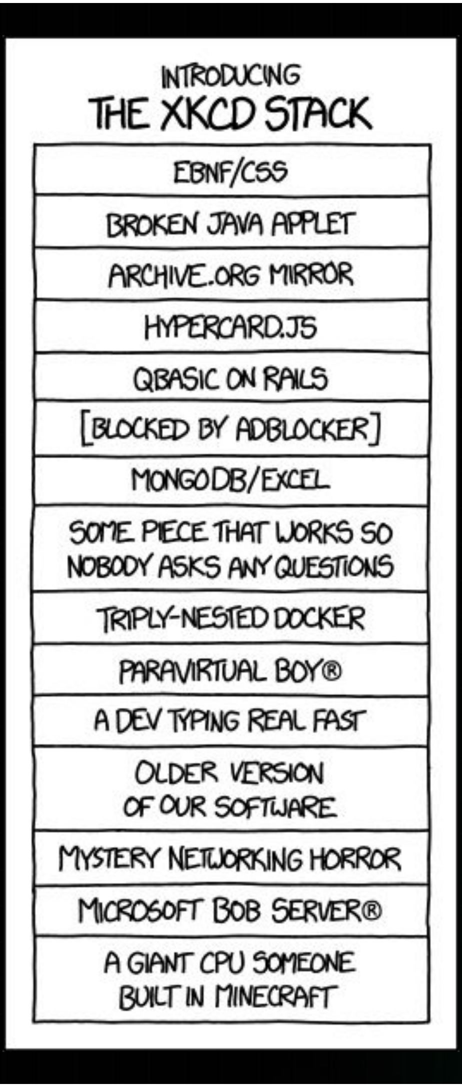
How do the different mechanisms address the Blockchain Trilemma?

@HummusOnRails

Smart Contracts

Are you a Full Stack Developer?

@HummusOnRails



HUMMUS ON RAILS

FUEL

The smart contract is the backend of your application

… with *some* differences.

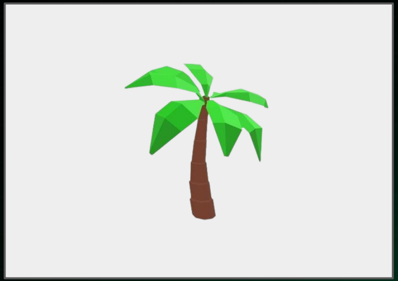
1. Immutability

2. Decentralization

3. Trustless

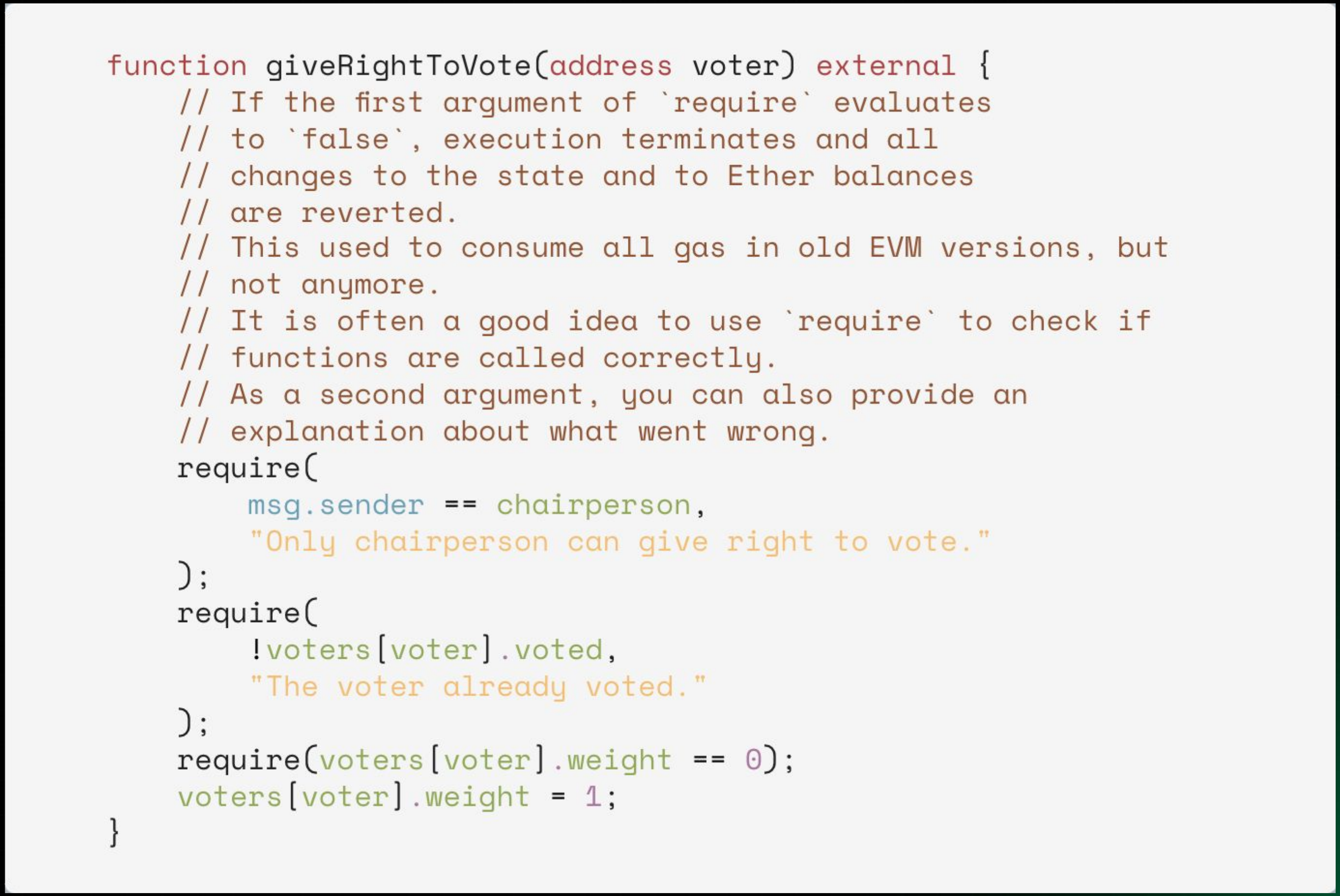
How do you write a Smart Contract?

@HummusOnRails



Solidity

Solidity is a statically-typed, contract-oriented programming language for writing smart contracts on the Ethereum blockchain, featuring syntax similar to JavaScript.



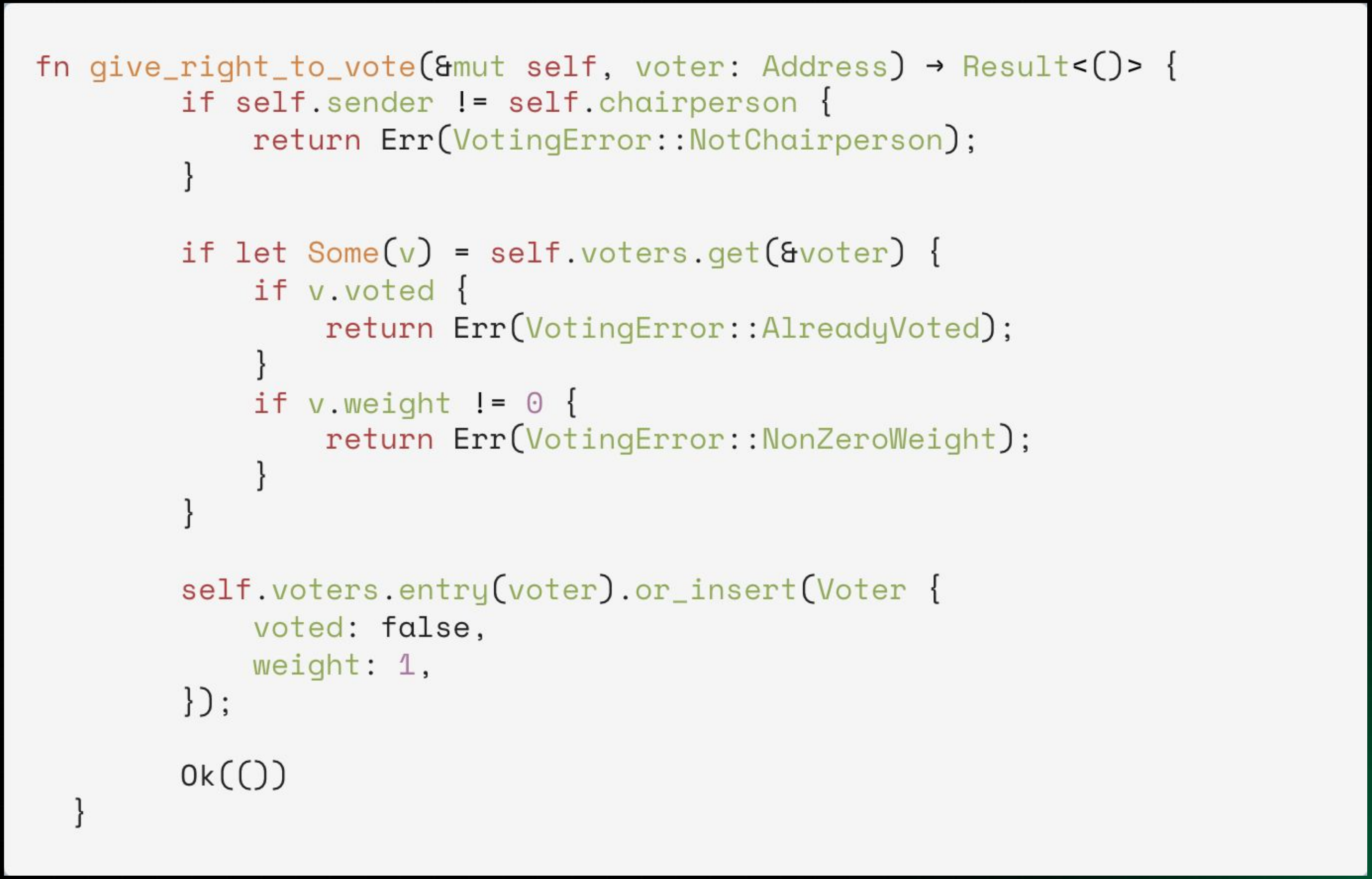
```
function giveRightToVote(address voter) external {
    // If the first argument of `require` evaluates
    // to `false`, execution terminates and all
    // changes to the state and to Ether balances
    // are reverted.
    // This used to consume all gas in old EVM versions, but
    // not anymore.
    // It is often a good idea to use `require` to check if
    // functions are called correctly.
    // As a second argument, you can also provide an
    // explanation about what went wrong.
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].voted,
        "The voter already voted."
    );
    require(voters[voter].weight == 0);
    voters[voter].weight = 1;
}
```

1. Solidity by Example

2. Crypto Zombies

Rust

Rust is a systems programming language gaining popularity in smart contract development for its emphasis on safety, speed, and concurrent processing capabilities.



```
fn give_right_to_vote(&mut self, voter: Address) → Result<()> {
    if self.sender != self.chairperson {
      return Err(VotingError::NotChairperson);
    }

    if let Some(v) = self.voters.get(&voter) {
      if v.voted {
        return Err(VotingError::AlreadyVoted);
      }
      if v.weight != 0 {
        return Err(VotingError::NonZeroWeight);
      }
    }

    self.voters.entry(voter).or_insert(Voter {
      voted: false,
      weight: 1,
    });

    Ok(())
 }
```

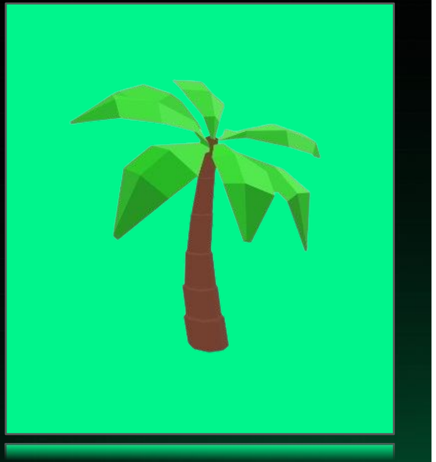
1. Rust Book

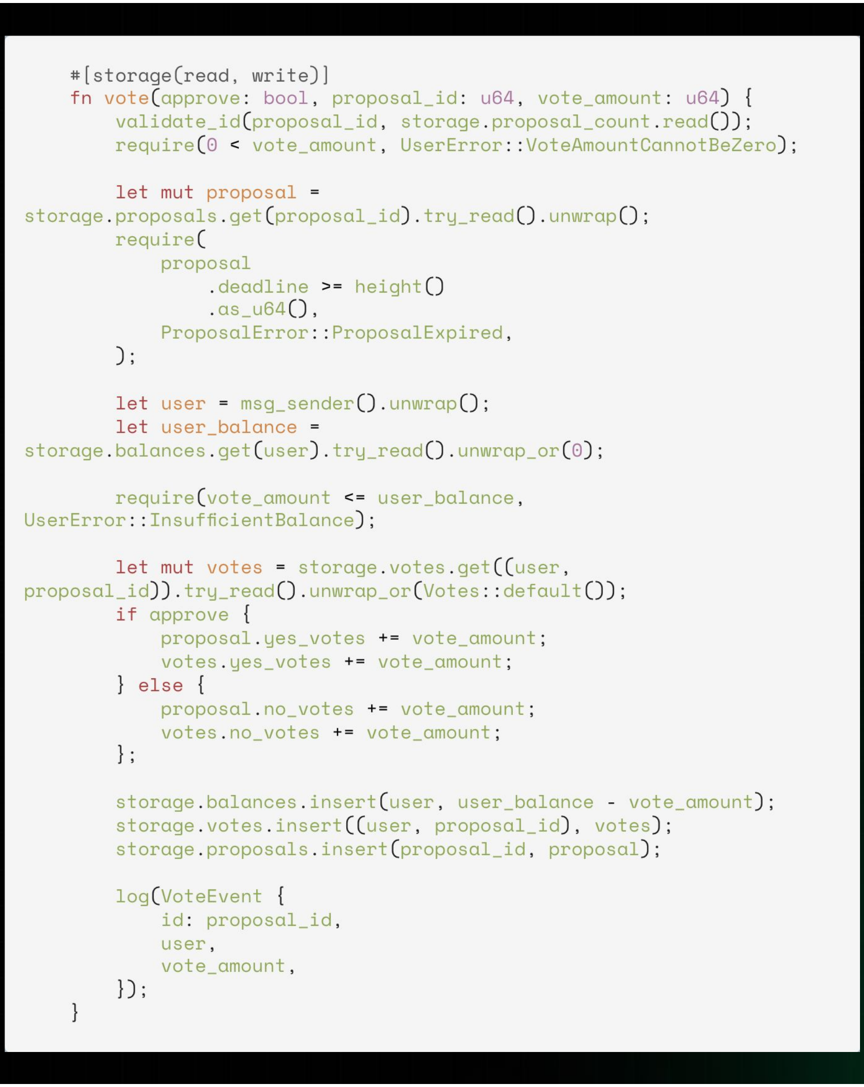
2. Rustlings

@HummusOnRails

Sway

Sway is a statically-typed, Rust-inspired domain-specific language designed for smart contract development, focusing on safe, parallelizable operations and optimized for high-throughput and efficient execution.





```
    #[storage(read, write)]
    fn vote(approve: bool, proposal_id: u64, vote_amount: u64) {
        validate_id(proposal_id, storage.proposal_count.read());
        require(0 < vote_amount, UserError::VoteAmountCannotBeZero);

        let mut proposal =
storage.proposals.get(proposal_id).try_read().unwrap();
        require(
            proposal
                .deadline >= height()
                .as_u64(),
            ProposalError::ProposalExpired,
        );

        let user = msg_sender().unwrap();
        let user_balance =
storage.balances.get(user).try_read().unwrap_or(0);

        require(vote_amount <= user_balance,
UserError::InsufficientBalance);

        let mut votes = storage.votes.get((user,
proposal_id)).try_read().unwrap_or(Votes::default());
        if approve {
            proposal.yes_votes += vote_amount;
            votes.yes_votes += vote_amount;
        } else {
            proposal.no_votes += vote_amount;
            votes.no_votes += vote_amount;
        };

        storage.balances.insert(user, user_balance - vote_amount);
        storage.votes.insert((user, proposal_id), votes);
        storage.proposals.insert(proposal_id, proposal);

        log(VoteEvent {
            id: proposal_id,
            user,
            vote_amount,
        });
    }
```

1. Sway Docs

2. Sway Playground

3. Sway Example Apps

Decentralized Applications

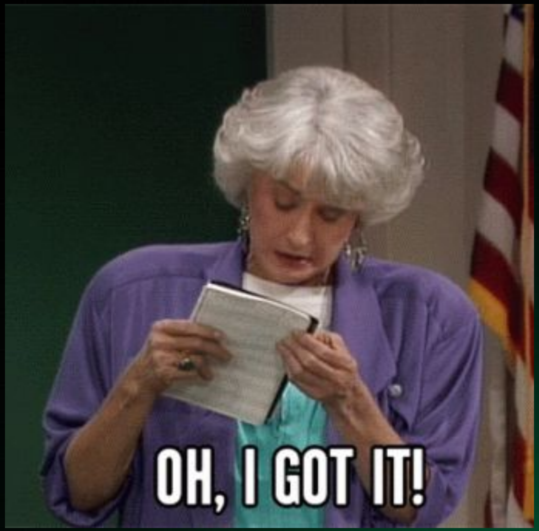
(dApps)

@HummusOnRails

Have you heard of React?
… or Vue?

Then, you can build your own dApp.

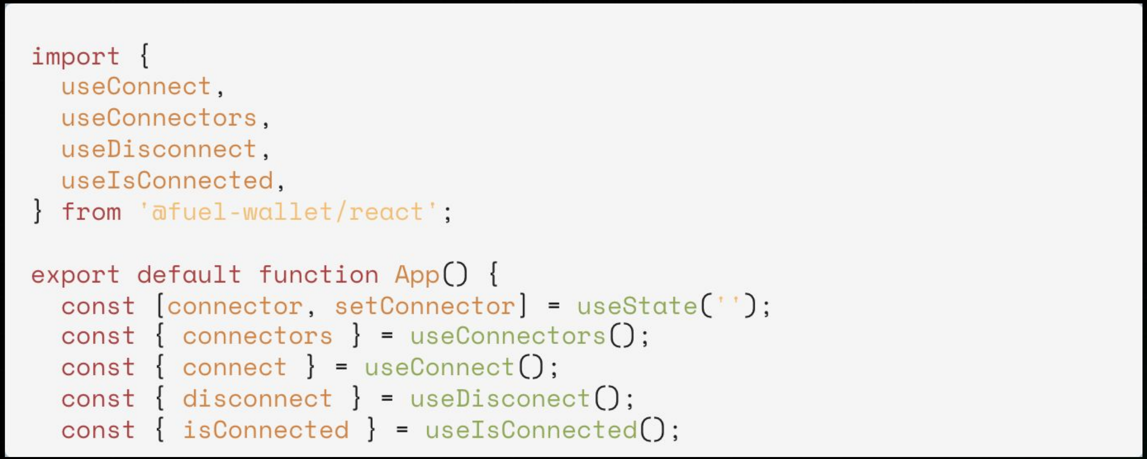
A decentralized application is a smart contract + a frontend



HUMMUS ON RAILS

@HummusOnRails

FUEL

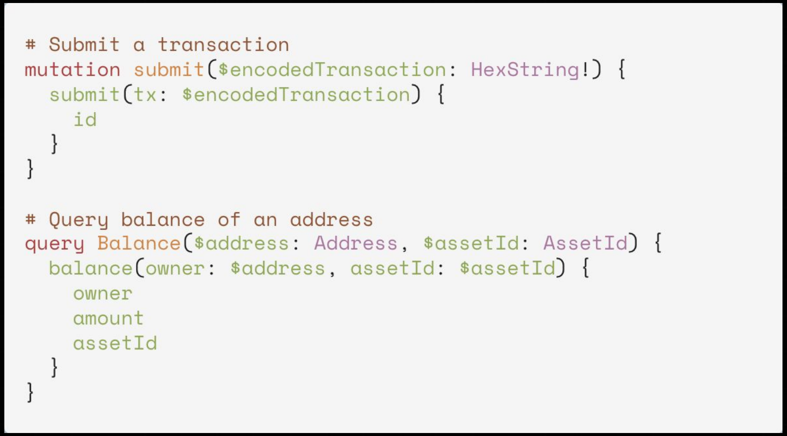


```
import {
  useConnect,
  useConnectors,
  useDisconnect,
  useIsConnected,
} from '@fuel-wallet/react';

export default function App() {
  const [connector, setConnector] = useState('');
  const { connectors } = useConnectors();
  const { connect } = useConnect();
  const { disconnect } = useDisconnect();
  const { isConnected } = useIsConnected();
```

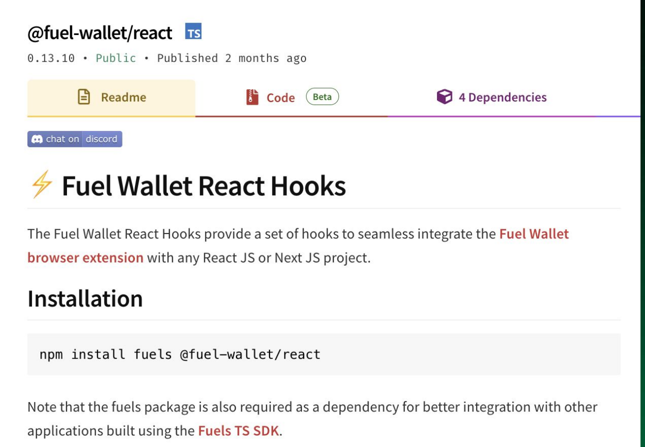
React Hooks Docs

@HummusOnRails



```
# Submit a transaction
mutation submit($encodedTransaction: HexString!) {
  submit(tx: $encodedTransaction) {
    id
  }
}

# Query balance of an address
query Balance($address: Address, $assetId: AssetId) {
  balance(owner: $address, assetId: $assetId) {
    owner
    amount
    assetId
  }
}
```



@fuel-wallet/react

0.13.10 • Public • Published 2 months ago

⚡ Fuel Wallet React Hooks

The Fuel Wallet React Hooks provide a set of hooks to seamless integrate the Fuel Wallet browser extension with any React JS or Next JS project.

Installation

```
npm install fuels @fuel-wallet/react
```

Note that the fuels package is also required as a dependency for better integration with other applications built using the Fuels TS SDK.

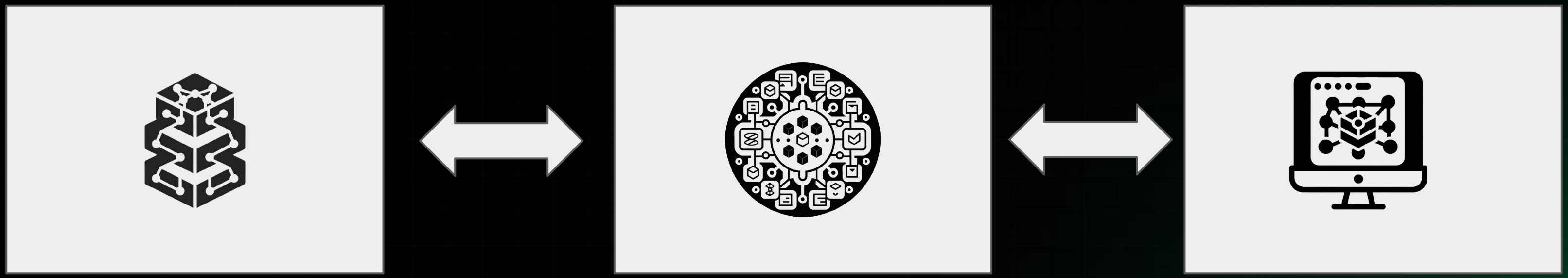
HUMMUS ON RAILS

@HummusOnRails

FUEL

Smart Contract && Frontend == dApp

Blockchain Layers

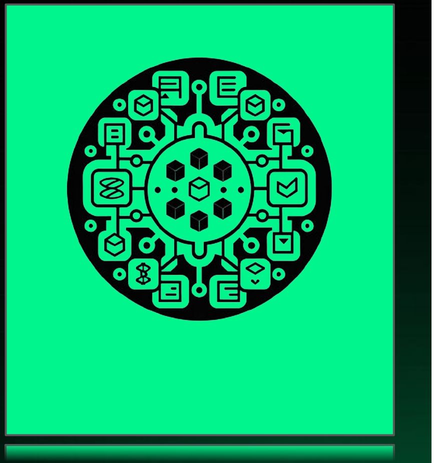


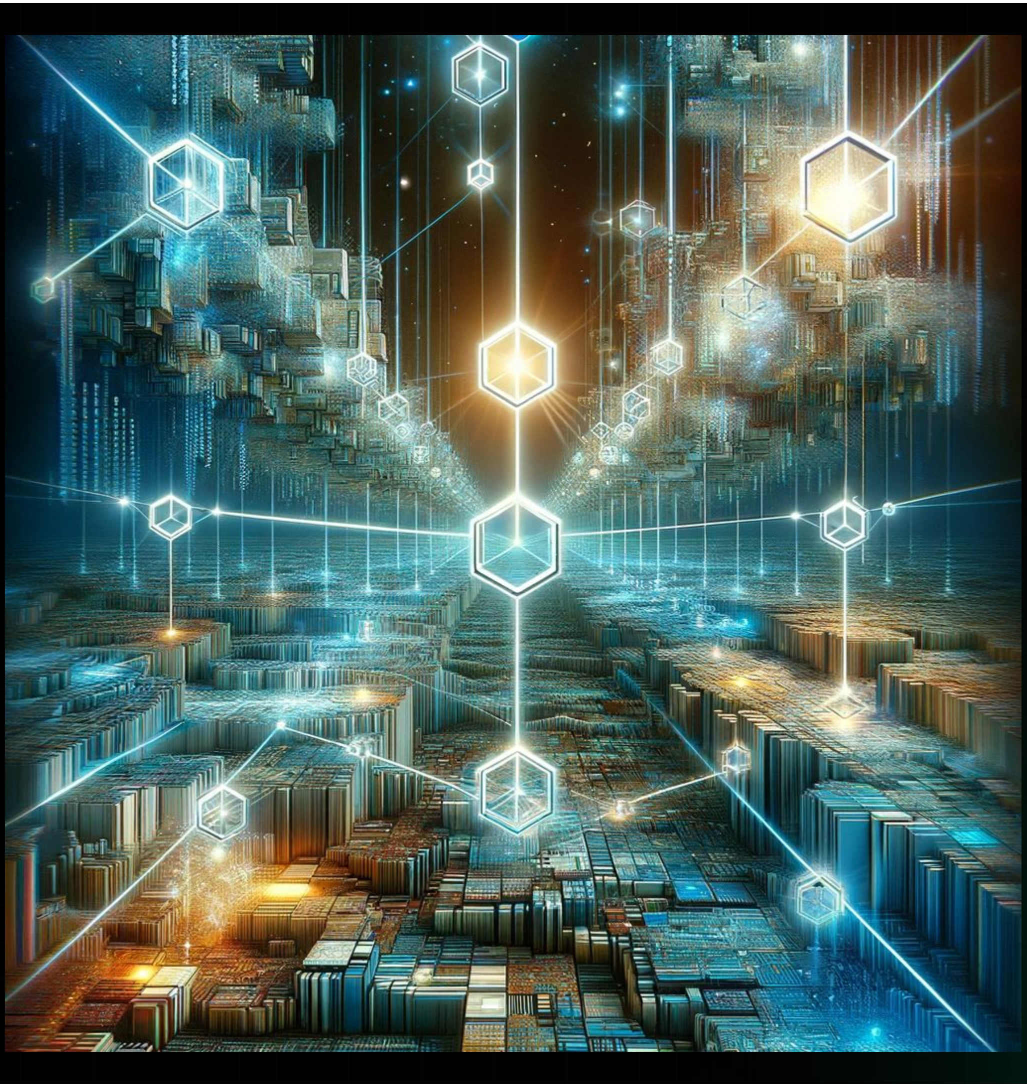
Layer 1

Layer 1 is the blockchain protocol itself, encompassing consensus mechanisms like Proof of Work or Proof of Stake, and is responsible for the network's primary transaction processing, security, and data integrity.

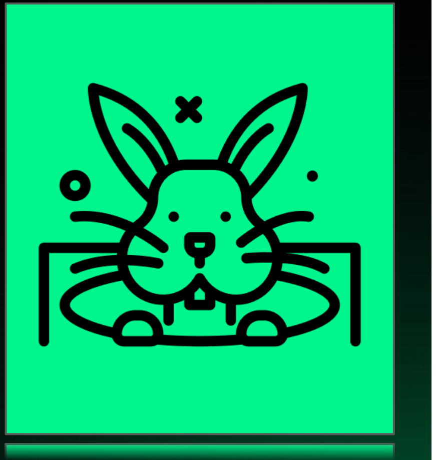
Layer 2

A secondary framework or protocol that is built on an underlying blockchain to enhance scalability and transaction throughput, utilizing solutions like rollups.



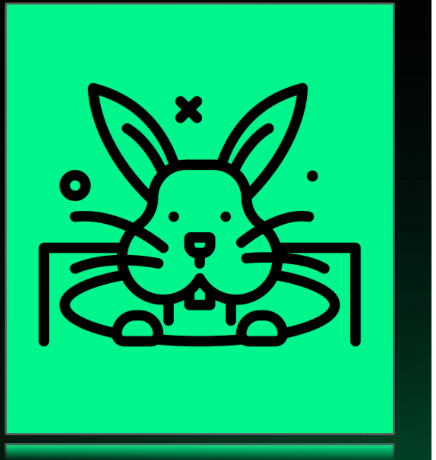


Down the rabbit hole...



What is a rollup?

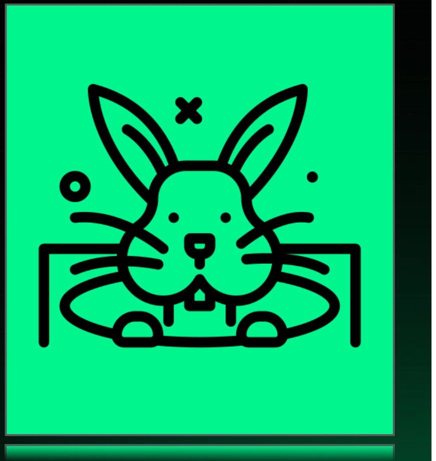
Down the rabbit hole...



A solution that aggregates and processes multiple transactions into a single batch, significantly enhancing throughput and efficiency while maintaining the underlying blockchain's security guarantees.

Optimistic Rollups

Down the rabbit hole...

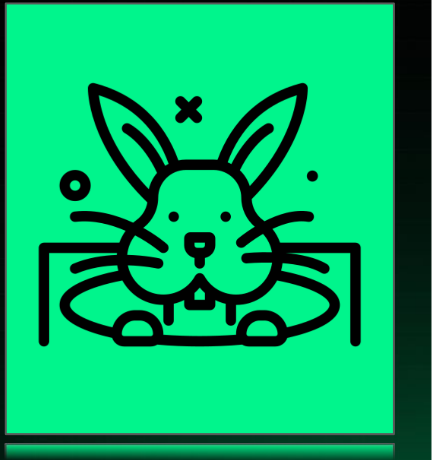


Execute transactions off-chain with presumed validity, batching them for on-chain verification, enhancing throughput with a fraud-proof system for retrospective dispute resolution, which allows network participants to challenge and rectify invalid transactions after they are processed.

@HummusOnRails

Down the rabbit hole...

ZK-Proof Rollups



Leverage zero-knowledge proofs to batch and validate multiple off-chain transactions in a single on-chain proof, ensuring data integrity and privacy.

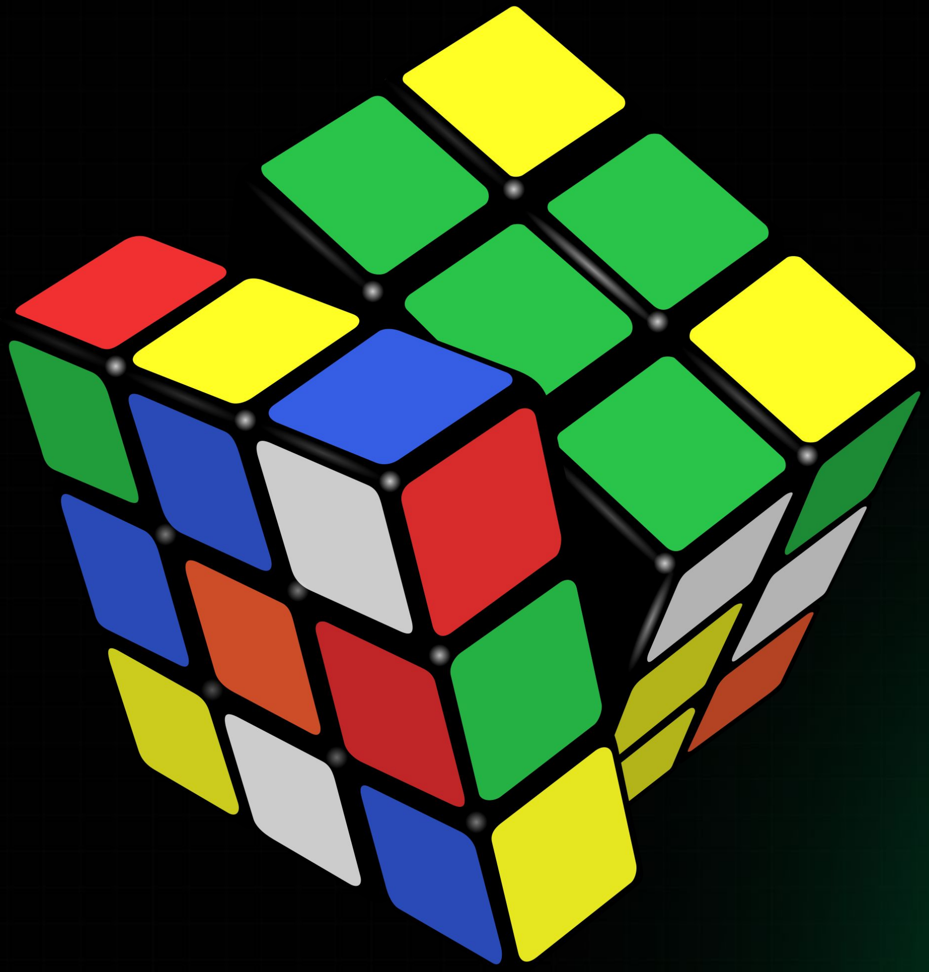
@HummusOnRails

But, wait…

*What is a **zero-knowledge** proof?*

Privacy-preserving cryptographic proof technique

@HummusOnRails

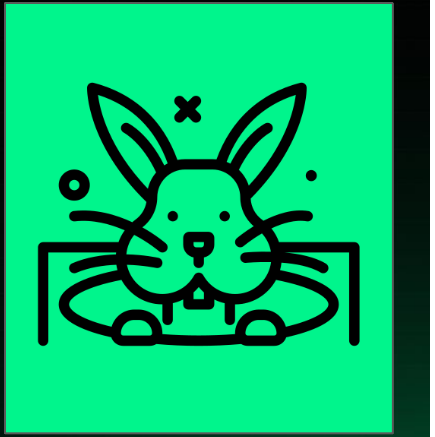


HUMMUS ON RAILS

FUEL

Hybrid Proving

Down the rabbit hole...



Bridging the gap between Optimistic and ZK Rollups. This approach blends single-round fraud-proving models with ZK proofs.

@HummusOnRails

Layer 3

The application layer, utilizing the underlying layers for secure, trustless transaction processing and application logic execution.

Conclusion

After our journey together, you are now officially a...

Blockchain Expert

@HummusOnRails

ok, not really

... but you do have a good foundation! 👏

Continue your learning journey

HUMMUS ON RAILS

@HummusOnRails

FUEL

DEVELOPER
CHAMPIONS PROGRAM

- Mentorship programs
- Dedicated learning opportunities
- Networking events
- Exclusive swag

@HummusOnRails

THANK YOU

Questions? Comments? Hummus Recipes?

@HummusOnRails

Credits

- http://bbslist.textfiles.com/619/ - Free Thinker BBS listing
- https://bytecellar.com/wp-content/uploads/2019/12/BBSing_image.jpg - Picture of BBS on monitor
- https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch07.html - Block contents
- https://www.flaticon.com - Icons used in slides
- https://www.explainxkcd.com/wiki/images/d/d7/xkcd_stack.png - XKCD cartoon