Building a trustful voting system on a blockchain

Michael Heinrichs

@netopyr

- Java Champion
- Leader of JUG Freiburg
- Contractor for Swirlds Labs
- Founder of Netopyr GmbH

Hedera™ Java Champions www.netopyr.com

Michael Heinrichs

@netopyr

- I ❤️ coding
- I ❤️ my family
- I ❤️ cooking
- I ❤️ travelling

Hedera

Java Champions

www.netopyr.com

Hendrik Ebbers

@hendrikEbbers

- Java Champion
- Eclipse Adoptium WG
- Contractor for Swirlds Labs
- Founder of Open Elements
- Eclipse Board Member

Hedera OpenElements ADOPTIUM Java Champions ECLIPSE FOUNDATION

SWIRLDS LABS

Hendrik Ebbers

@hendrikEbbers

- I ❤️ Star Wars
- I ❤️ dogs
- I ❤️ boardgames
- I ❤️ open source

Hedera OpenElements ADOPTIUM Java Champions ECLIPSE FOUNDATION

SWIRLDS LABS

What you will learn today

- What is a smart contract
- How to use public ledgers

What you will **NOT** learn today

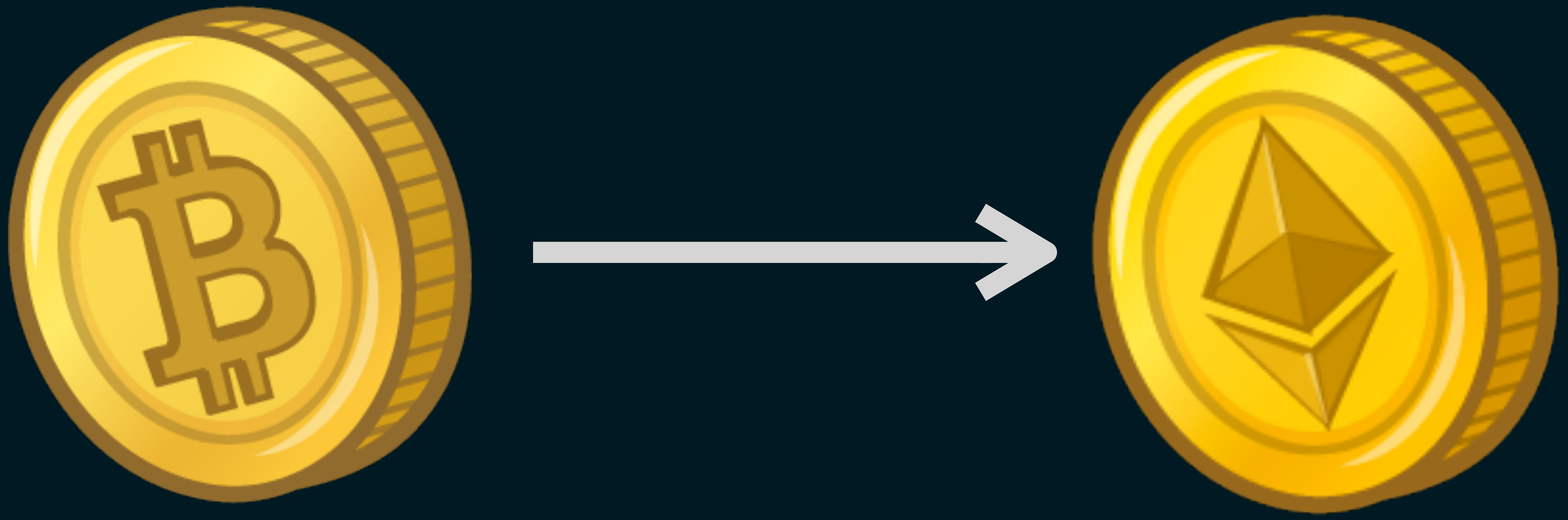
- How to trade Bitcoins
- How to get rich with NFTs

Public Ledgers

SWIRLDS LABS

The Hype started...

The next generation

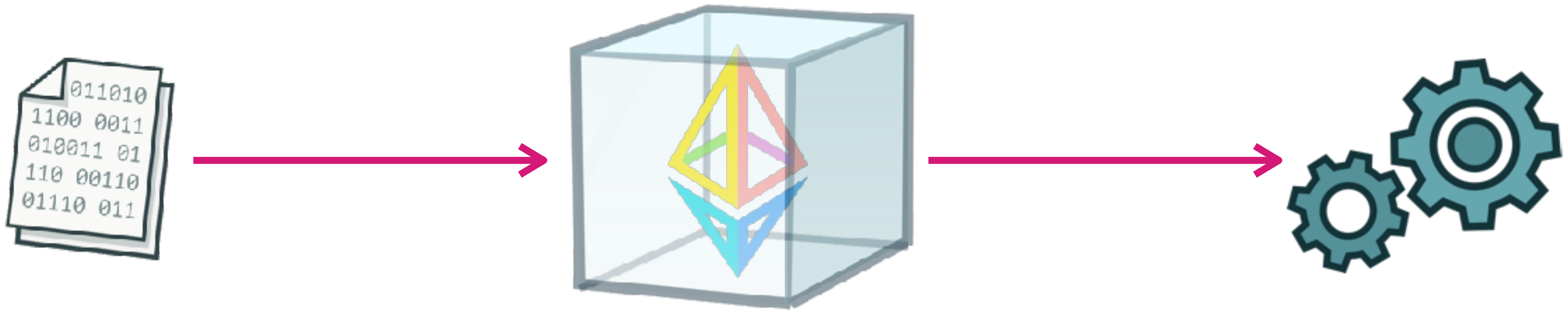


1st generation

2nd generation

Ethereum

- The big difference to Bitcoin is the Ethereum Virtual Machine (EVM)
- The EVM can be used to execute code (smart contracts) on the network



Smart Contract

EVM

Transactional Changes

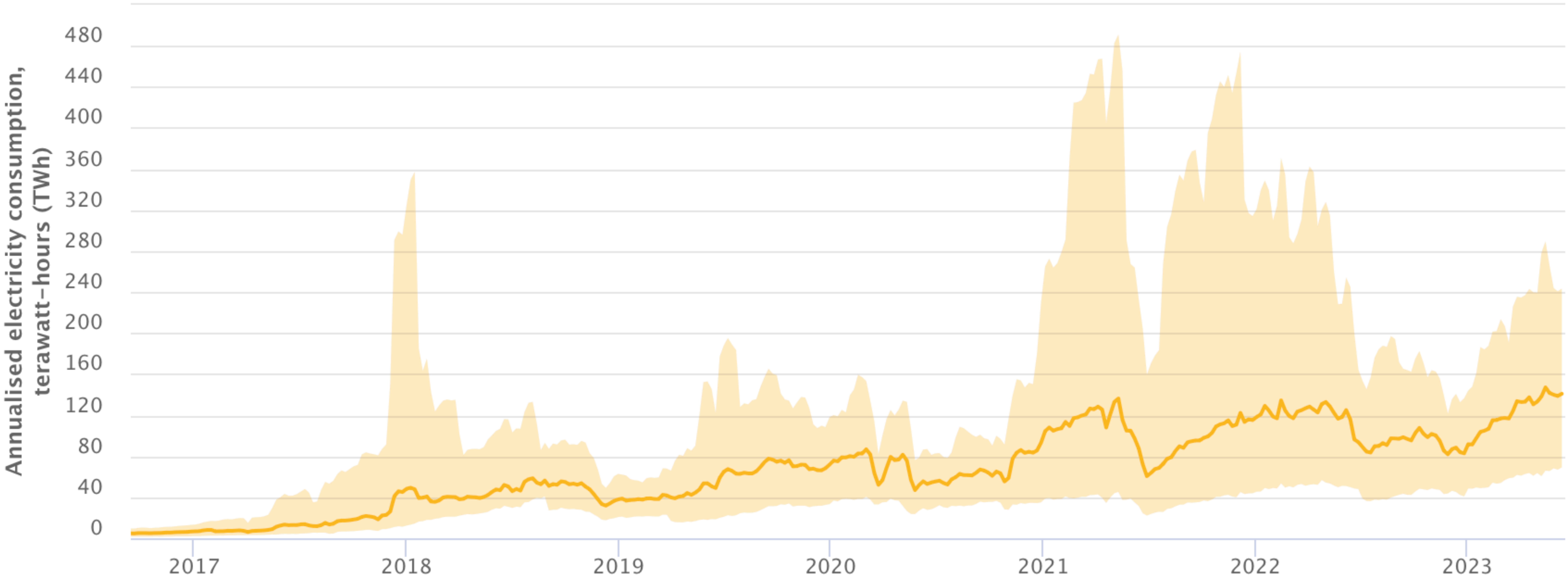
Smart Contracts and Tokens

- Ethereum (and the EVM) allows you to define any kind of token
- Non-Fungible Tokens (NFTs) are supported
- Tokens and NFTs are based on smart contracts

Crypto Sustainability

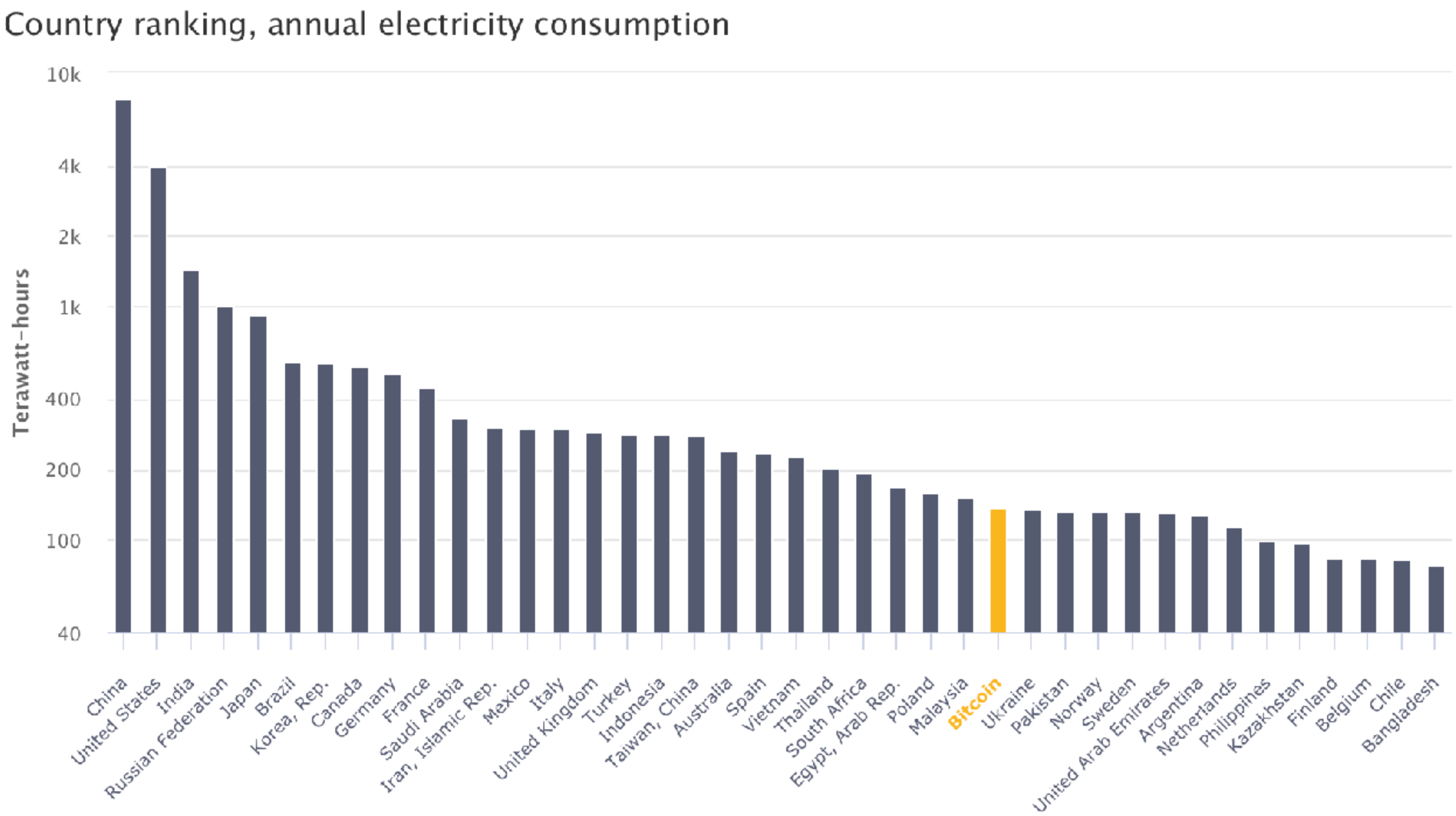
SWIRLDS LABS

Sustainability of Bitcoin



https://ccaf.io/cbnsi/cbeci

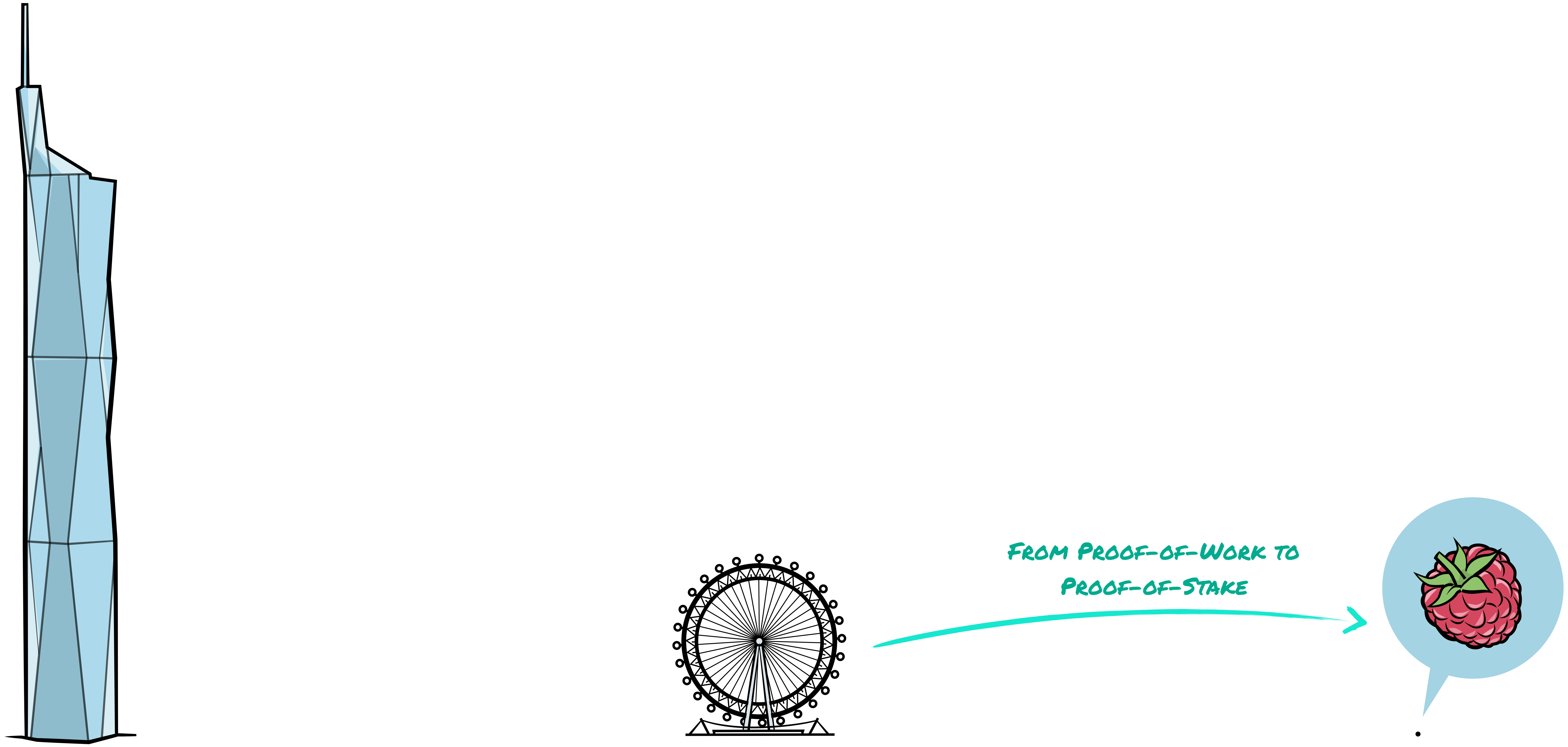
Sustainability of Bitcoin



Country ranking, annual electricity consumption

https://ccaf.io/cbnsi/cbeci

Sustainability of Ethereum



From Proof-of-Work to Proof-of-Stake

Bitcoin

Ethereum 1.0

Ethereum 2.0

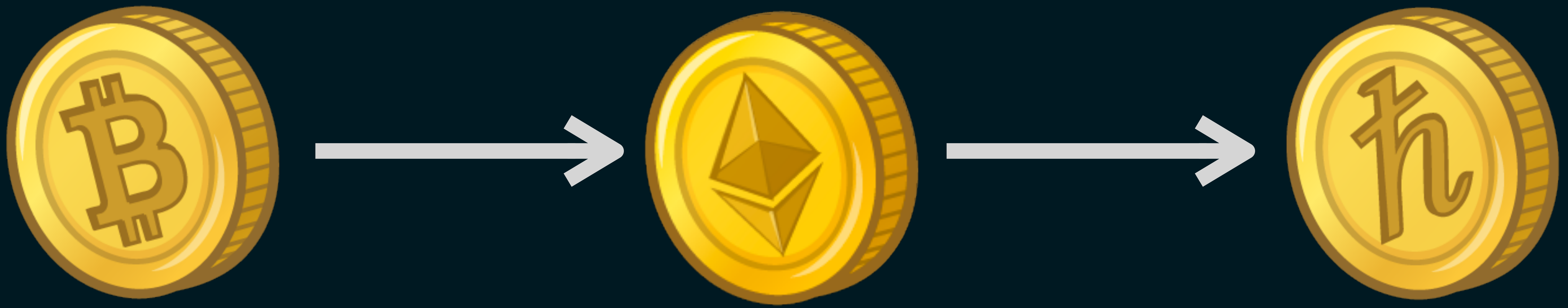
Sustainability of cloud services

- **PayPal:** yearly electricity consumption of **0.28 TWh**
- **Netflix:** yearly electricity consumption of **0.45 TWh**
- **YouTube:** yearly electricity consumption of **12 TWh**

- **Bitcoin:** yearly electricity consumption of **131 TWh**
- **Ethereum:** yearly electricity consumption of **0.0026 TWh**

https://ethereum.org/en/energy-consumption/

The next generation



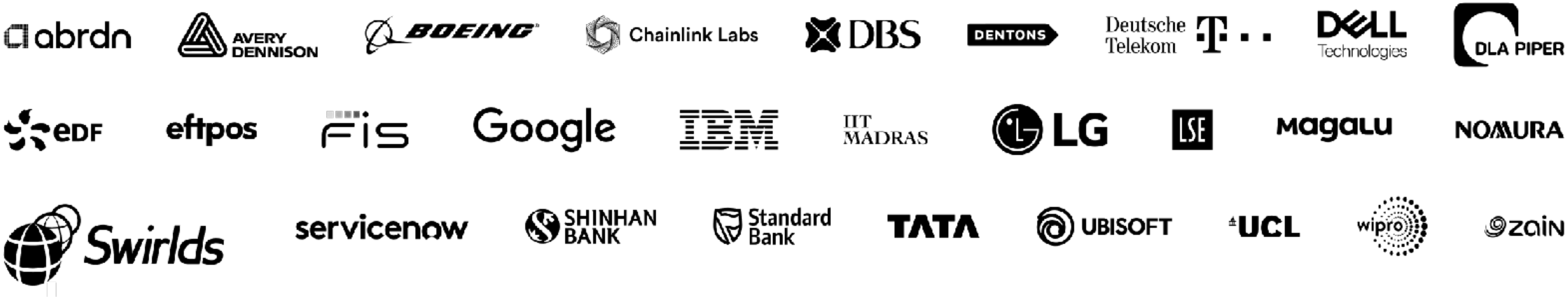
1st generation

2nd generation

3rd generation

The Hedera Network

- The Hedera Network is a network that is based on several nodes
- Nodes running on machines of the Hedera Foundation council members



Node 0 (account 0.0.3) | Hosted by LG | Seoul, South Korea	Operational
Node 1 (account 0.0.4) | Hosted by Swirlds | North Carolina, USA	Operational
Node 2 (account 0.0.5) | Hosted by FIS | Florida, USA	Operational
Node 3 (account 0.0.6) | Hosted by Wipro | Mumbai, India	Operational
Node 4 (account 0.0.7) | Hosted by Nomura | Tokyo, Japan	Operational
Node 5 (account 0.0.8) | Hosted by Google | Helsinki, Finland	Operational
Node 6 (account 0.0.9) | Hosted by Zain Group | Kuwait City, Kuwait	Operational
Node 7 (account 0.0.10) | Hosted by Magalu | São Paulo, Brazil	Operational
Node 8 (account 0.0.11) | Hosted by Boeing | Washington, USA	Operational
Node 9 (account 0.0.12) | Hosted by DLA Piper | London, UK	Operational
Node 10 (account 0.0.13) | Hosted by Tata Communications | California, USA	Operational
Node 11 (account 0.0.14) | Hosted by IBM | Washington, USA	Operational
Node 12 (account 0.0.15) | Hosted by Deutsche Telekom | Berlin, Germany	Operational
Node 13 (account 0.0.16) | Hosted by UCL | London, UK	Operational
Node 14 (account 0.0.17) | Hosted by Avery Dennison | Pennsylvania, USA	Operational
Node 15 (account 0.0.18) | Hosted by Dentons | Singapore	Operational
Node 16 (account 0.0.19) | Hosted by Standard Bank | Johannesburg, South Africa	Operational
Node 17 (account 0.0.20) | Hosted by eftpos | Sydney, Australia	Operational

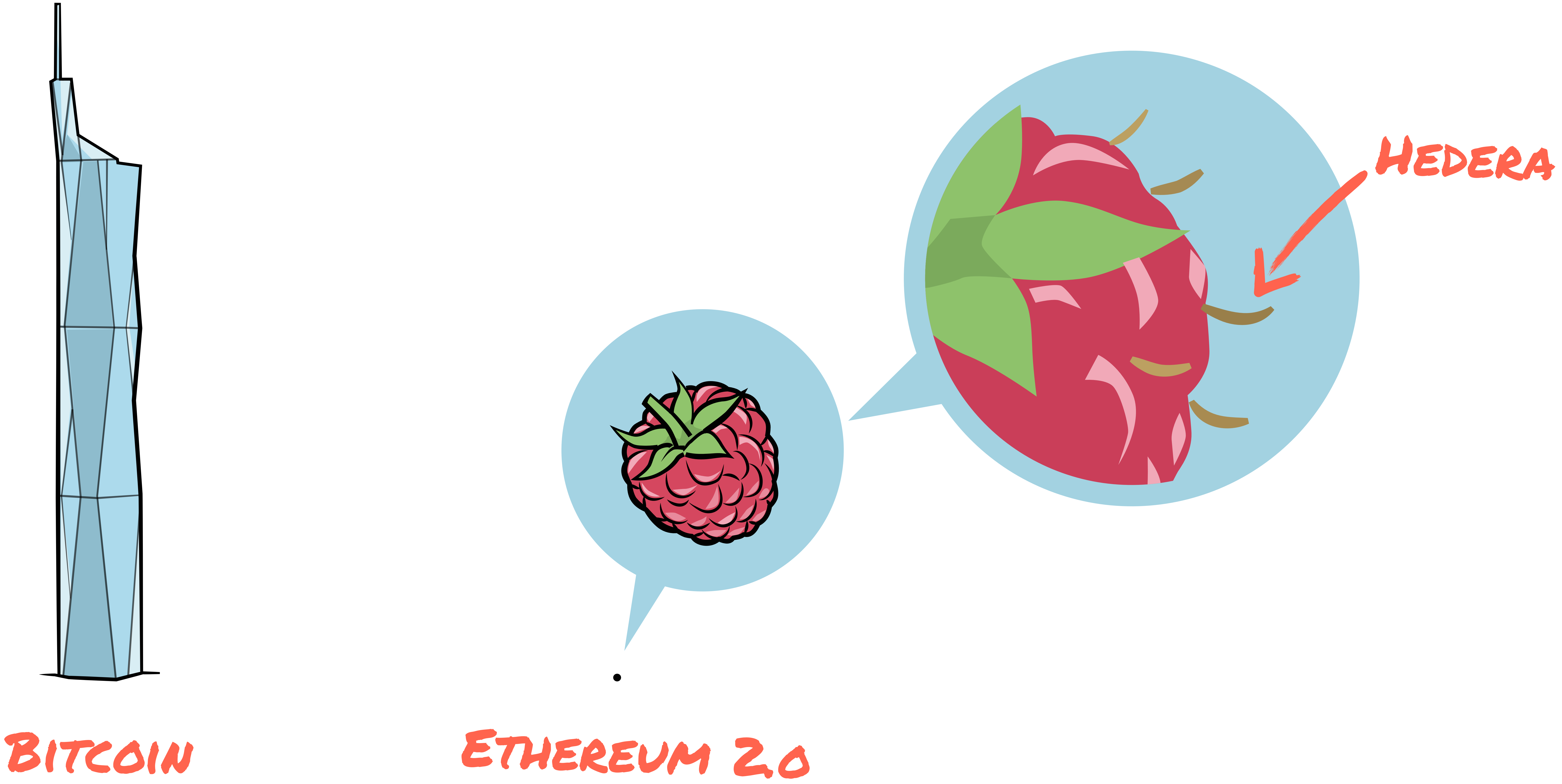
Sustainability of Hedera

- **0.000003 kWh** per transaction

0.006 KWH AT ETHEREUM

- Hedera is committed to **carbon-negative** network operations by purchasing carbon offsets quarterly

Sustainability of Ethereum



The Hedera Network

- > **40,000,000** transactions per day

1,000,000 AT ETHEREUM

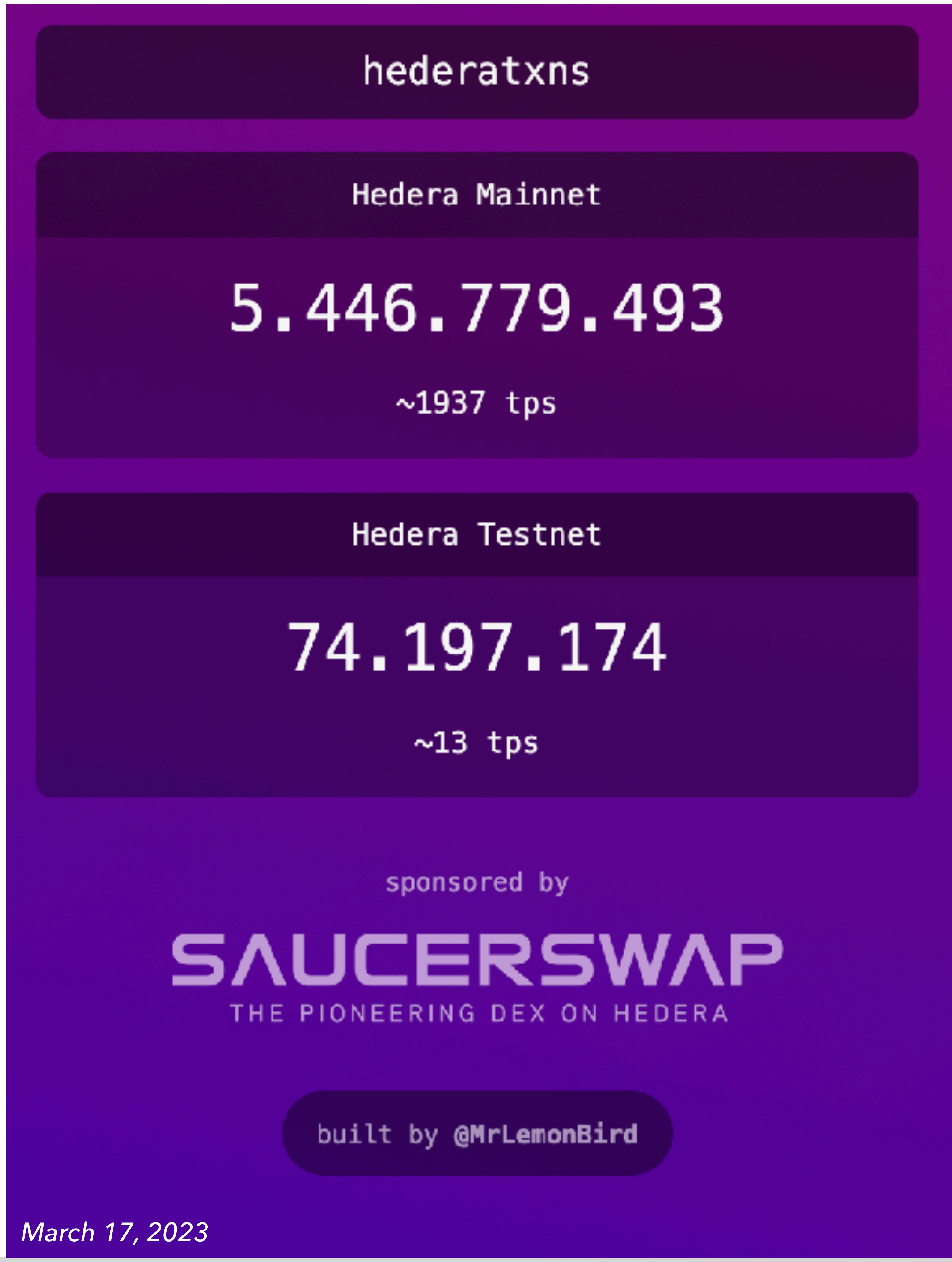
- **0.001$** cost per transaction

1.31 $ AT ETHEREUM (24.07.2023)

https://ycharts.com/indicators/ethereum_average_transaction_fee_eth
https://ycharts.com/indicators/ethereum_transactions_per_day

The Hedera Network

- MainNet can handle **> 1.000 tps** (transaction per second)
- Over **5.000.000.000** transactions have been handled in production
- In near future it will be **> 10 Billion** transactions



hederatxns

Hedera Mainnet

5.446.779.493

~1937 tps

Hedera Testnet

74.197.174

~13 tps

sponsored by

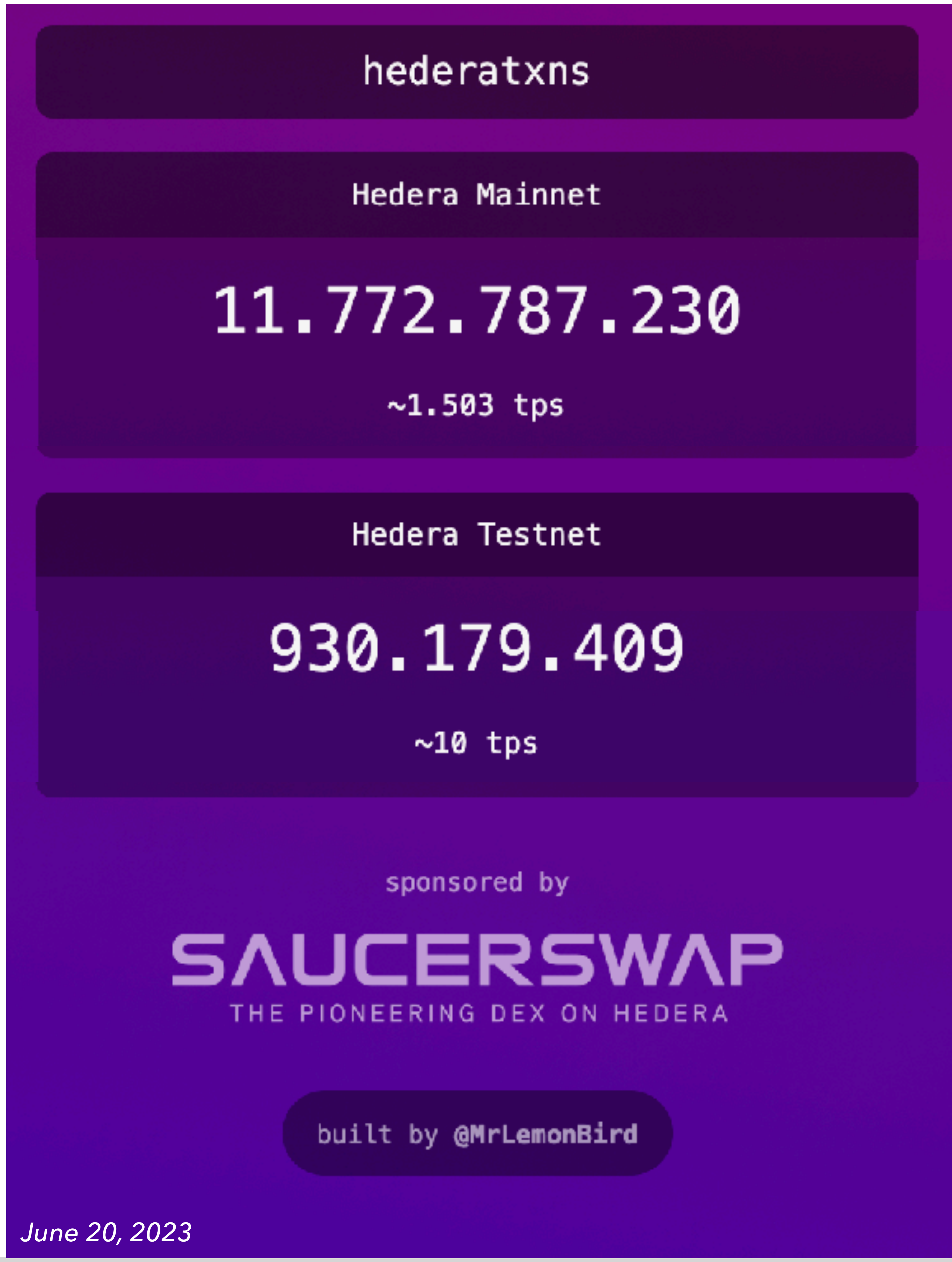
built by @MrLemonBird

March 17, 2023

@net0pyr | @hendrikEbbers

The Hedera Network

- MainNet can handle **> 1.000 tps** (transaction per second)
- Over ~~**5.000.000.000**~~ 10.000.000.000 transactions have been handled in production
- In near future it will be **> ~~10~~ 20 Billion** transactions



hederatxns

Hedera Mainnet

11.772.787.230

~1.503 tps

Hedera Testnet

930.179.409

~10 tps

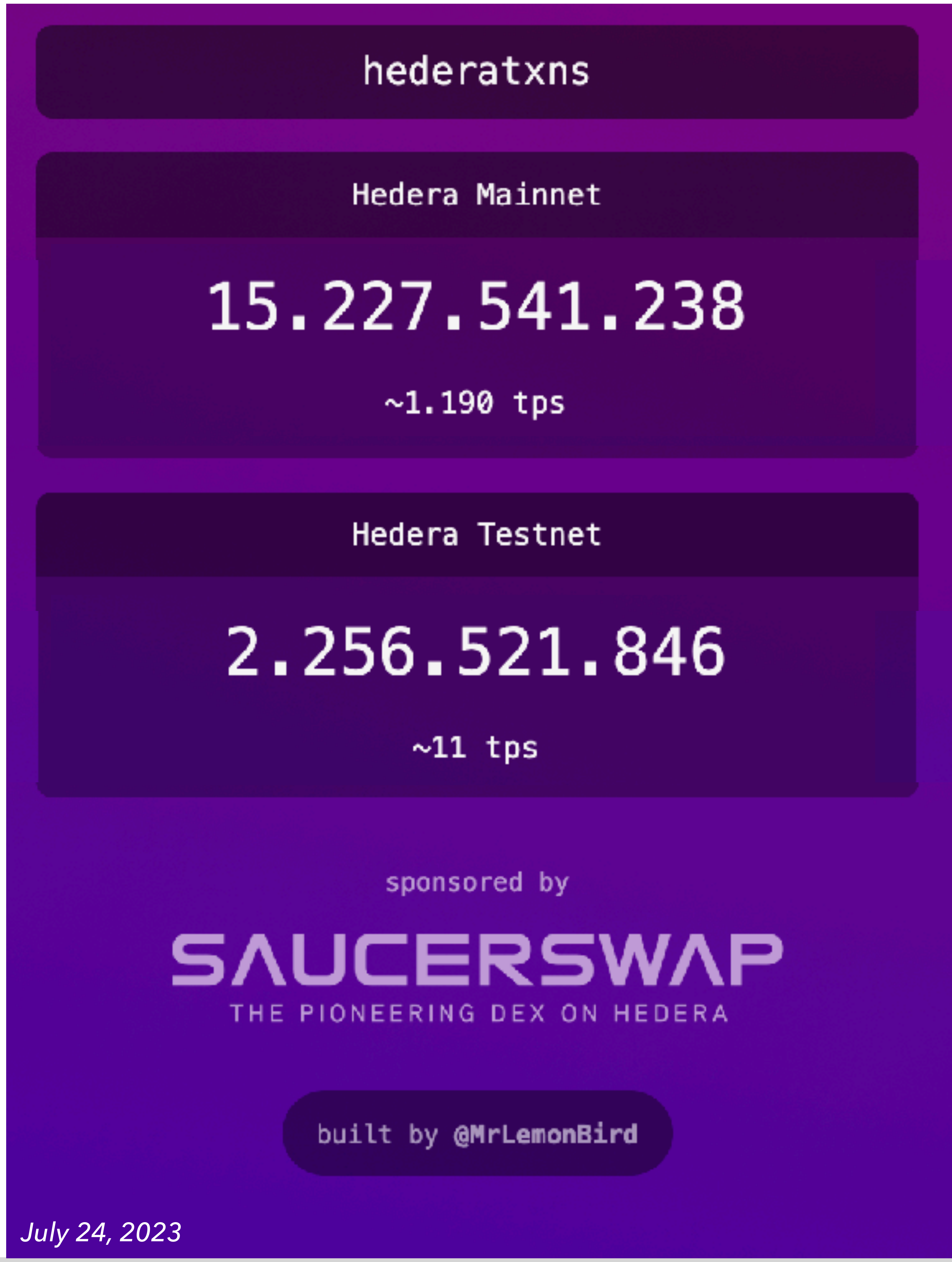
sponsored by

built by @MrLemonBird

June 20, 2023

The Hedera Network

- MainNet can handle **> 1.000 tps** (transaction per second)
- Over 15.000.000.000 ~~**5.000.000.000**~~ transactions have been handled in production
- In near future it will be **> 20 ~~10~~ Billion** transactions



hederatxns

Hedera Mainnet

15.227.541.238

~1.190 tps

Hedera Testnet

2.256.521.846

~11 tps

sponsored by

built by @MrLemonBird

July 24, 2023

UCL CENTRE
FOR BLOCKCHAIN
TECHNOLOGIES

Maximum **Transactions Per Second**

10,000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1,000 TX/SEC
Algorand
40 TX/SEC
Tezos
1,000 TX/SEC
Polkadot.
257 TX/SEC
CARDANO
3,000 TX/SEC
ethereum 2.0
10,000 TX/SEC
Hedera

UCL Centre for Blockchain Technologies Report from Q32021 "Energy Footprint of Blockchain Consensus Mechanisms Beyond Proof-of-Work"

AVERY DENNISON

The world's leading connected product cloud

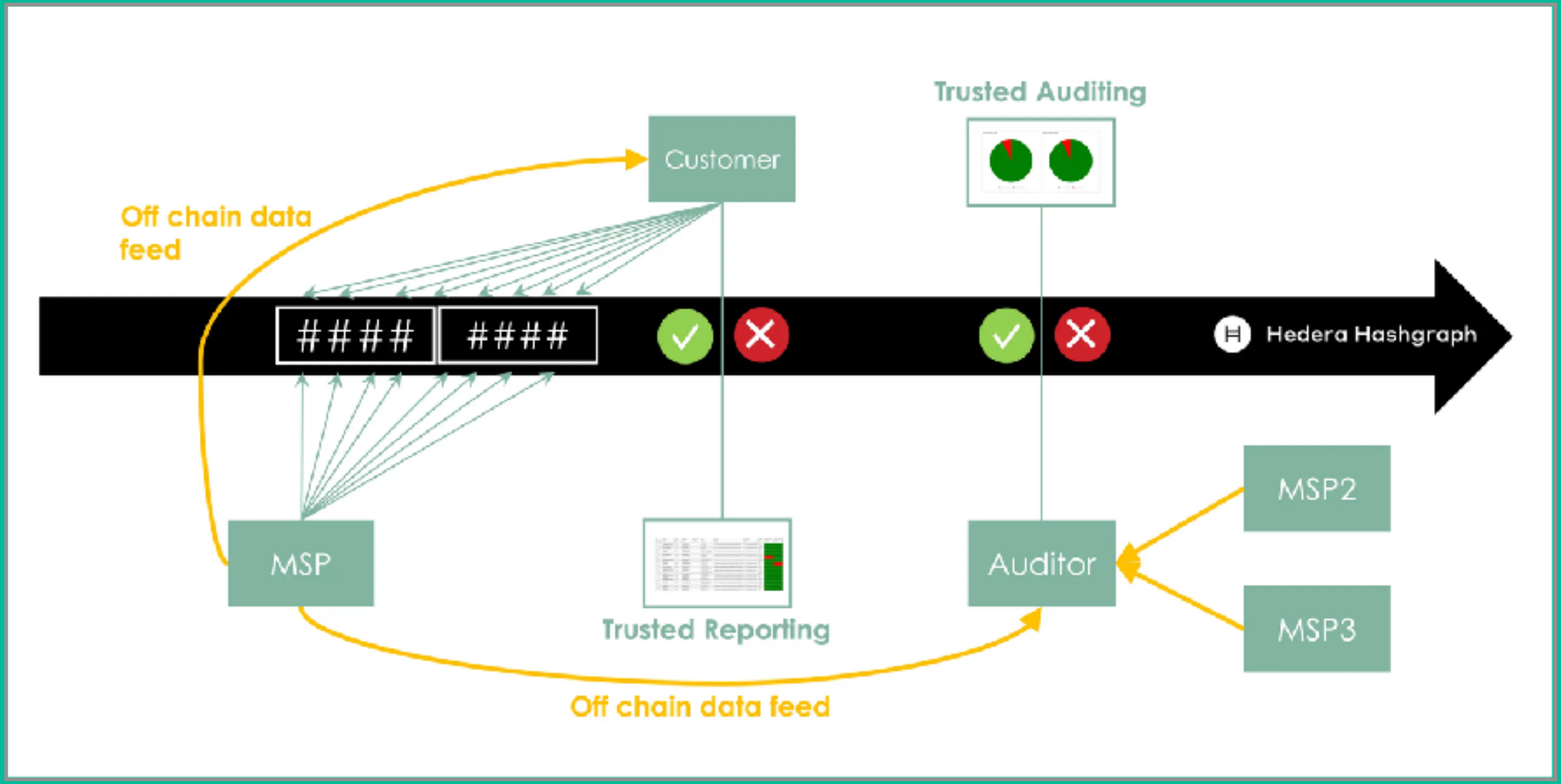
A platform that unlocks the power of connected products by assigning unique digital IDs to everyday items, providing unparalleled end-to-end transparency by tracking, storing and managing all the events associated with each individual product — from source to consumer and beyond to enable circularity.

EUR 29.90
GBP 24.90
SLO-MO
VIDEO
PHOTO
PORTRAIT
PANO
Request a demo

servicenow

Trusted SLAs

"Service Level Agreements (SLAs) are a classic friction point in the relationship, where the MSP has a conflict of interest between transparency and the need to meet contractual obligations. Moreover, in the long-term, MSPs need to avoid relationship degradation with their customers and, at the same time, differentiate their offerings against the competition."



SWIRLDS LABS

@net0pyr | @hendrikEbbers

SwirldsLabs.com

Liithos

The Last of Us, Uncharted and Days Gone Veterans from PlayStation Re-unite to form AAA Gaming and Entertainment Studio Liithos, Announce Ashfall for PC, Console, and Web 3

ASHFALL

SWIRLDS LABS

@net0pyr | @hendrikEbbers

SwirldsLabs.com

Hedera EVM Compatibility

- The same smart contracts can be deployed and executed on Ethereum and Hedera
- Hedera uses an EVM implementation internally

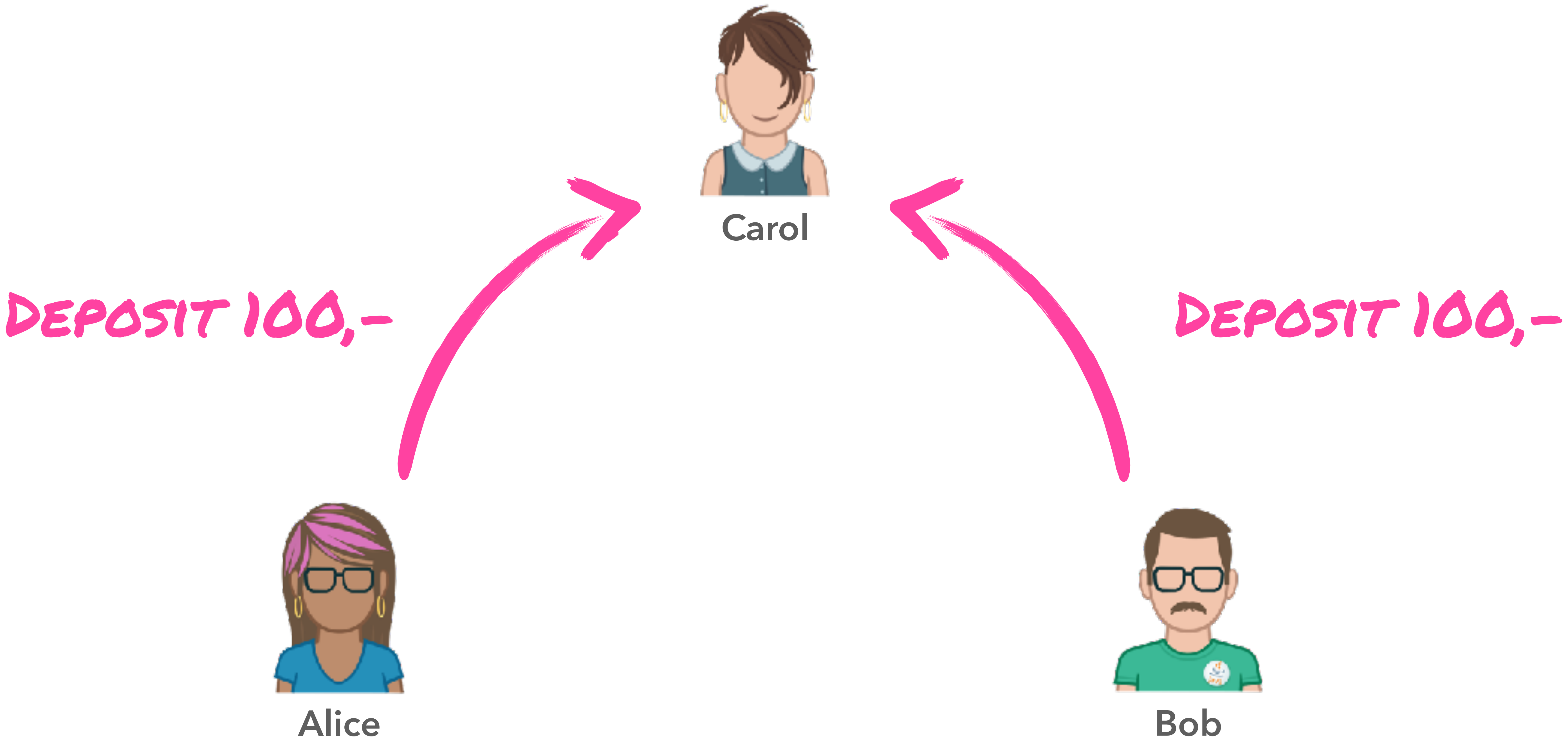
HYPERLEDGER BESU

Ethereum Version	Hedera Version
Shanghai	0.38
London	0.19
Berlin	0.19

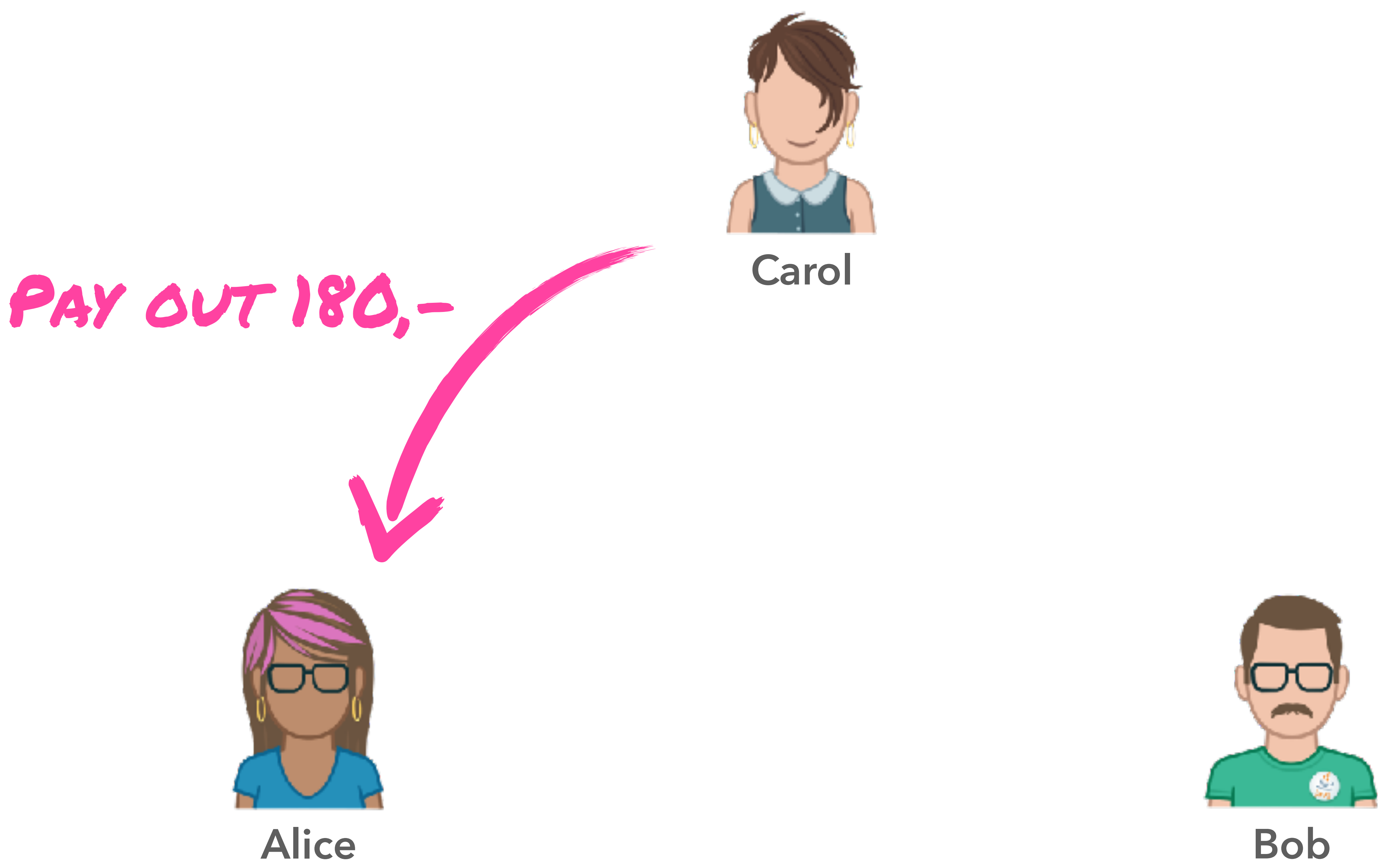
Smart Contracts

SWIRLDS LABS

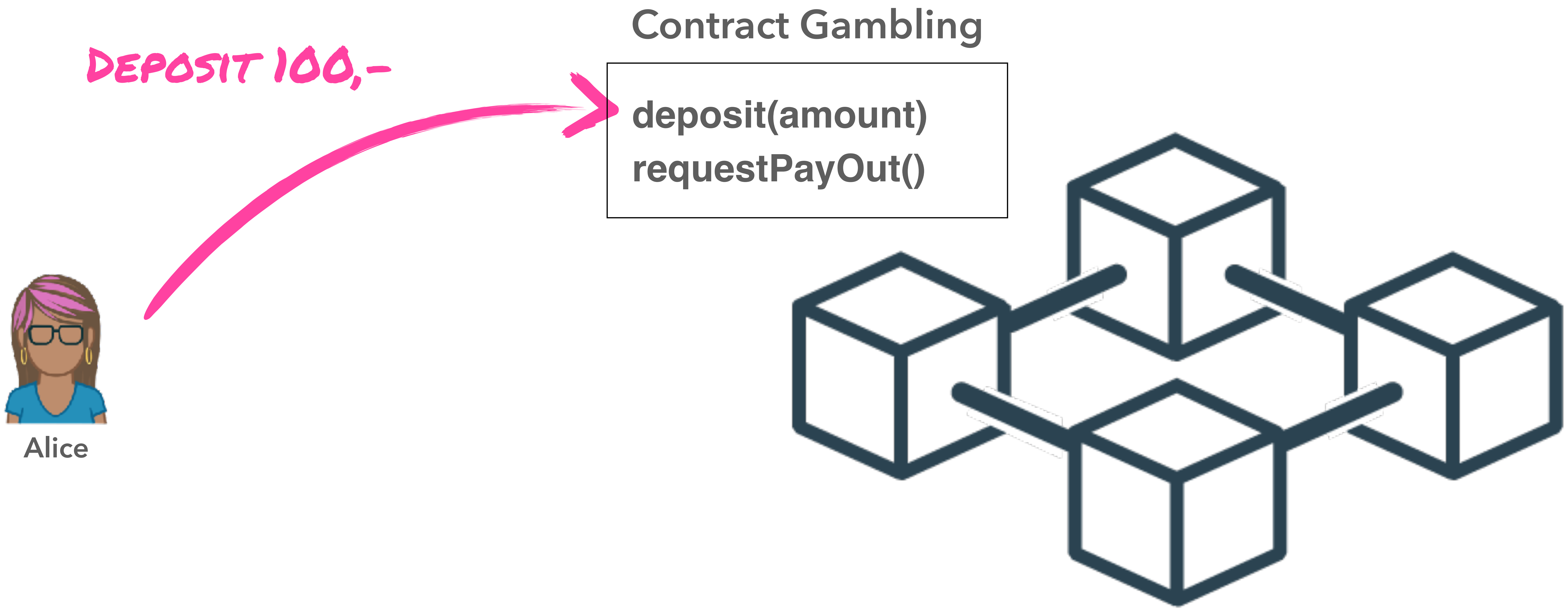
Gambling Example



Gambling Example



Gambling Example



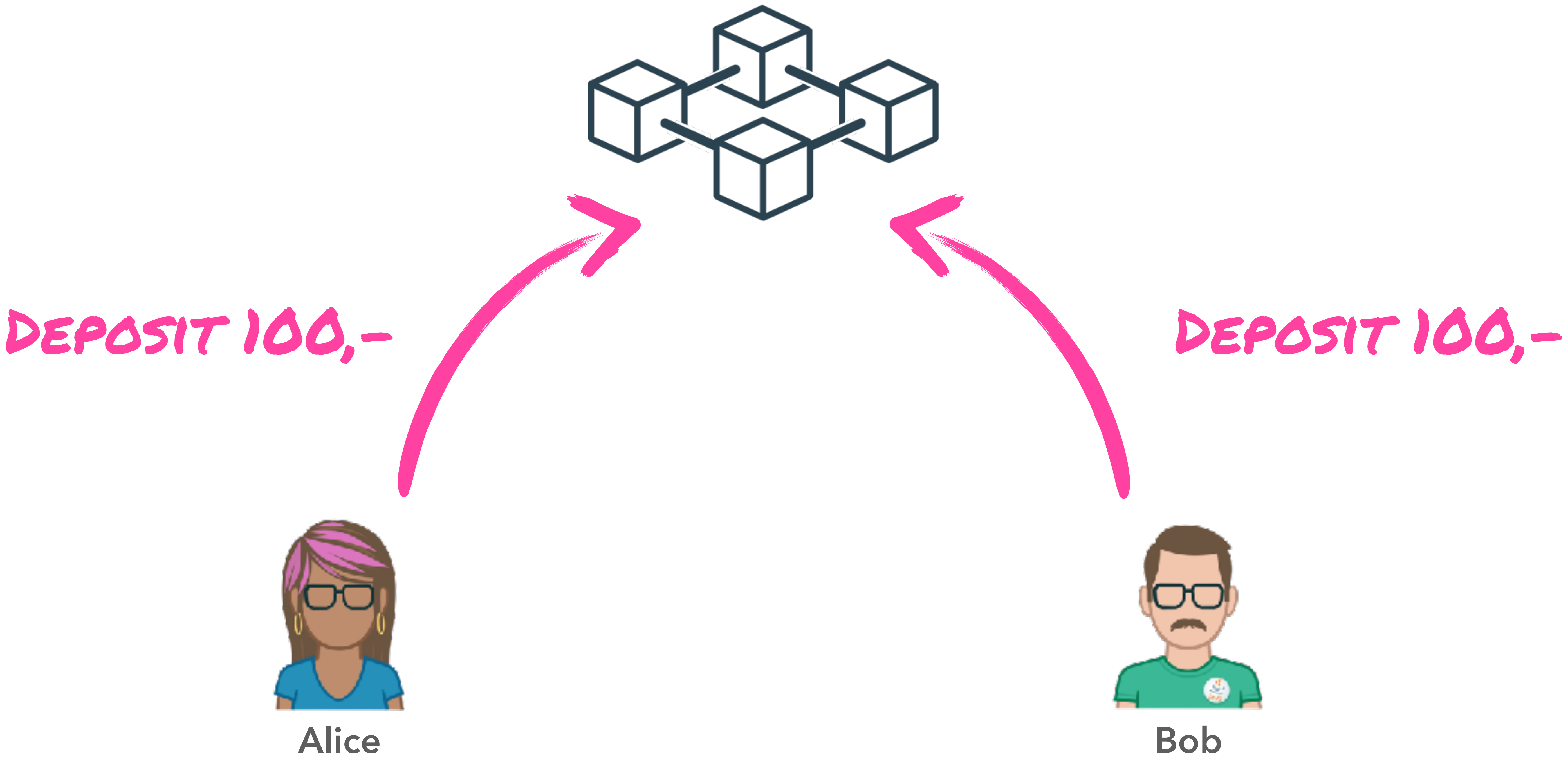
DEPOSIT 100,-

Contract Gambling

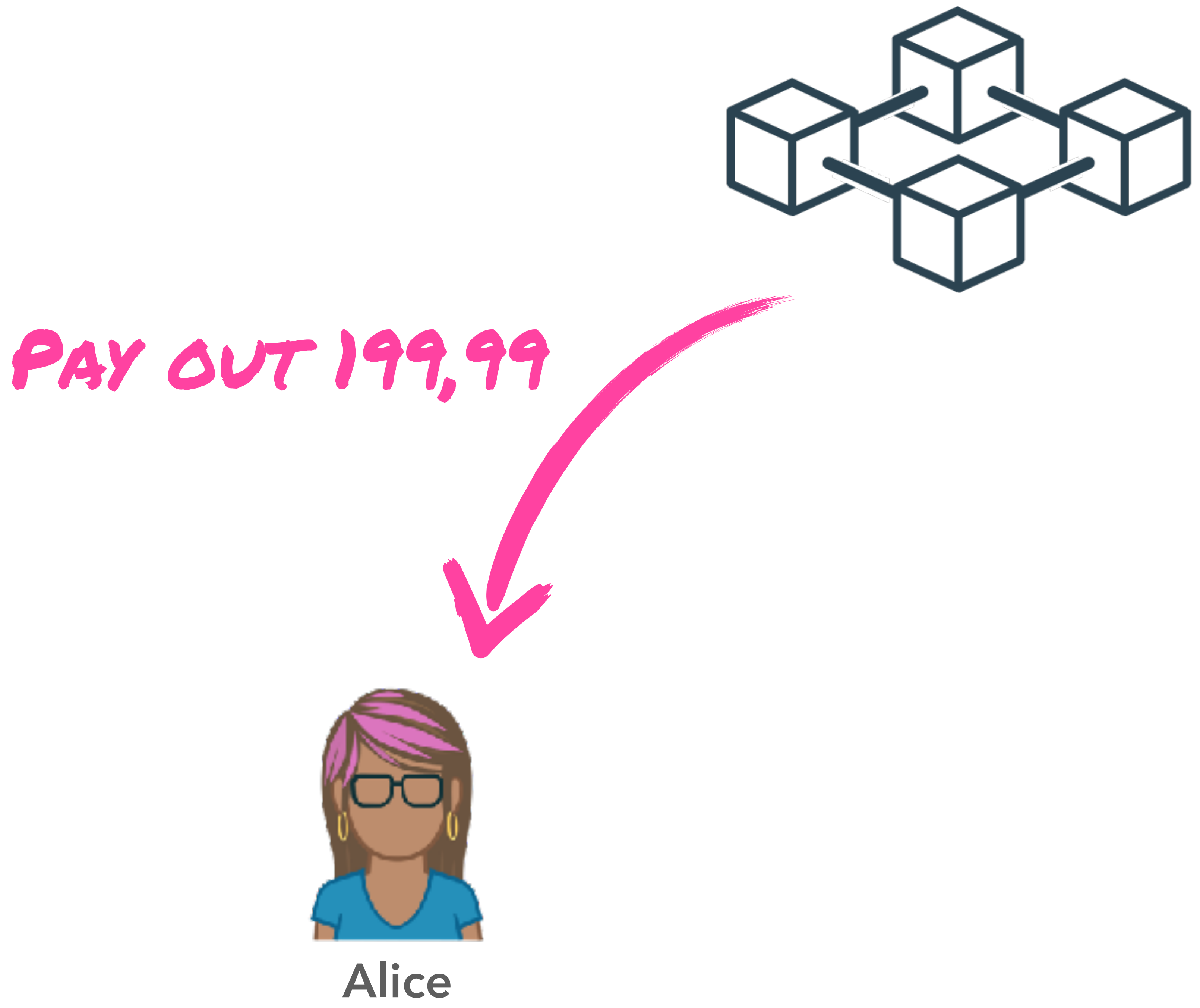
deposit(amount)
requestPayOut()

Alice

Gambling Example



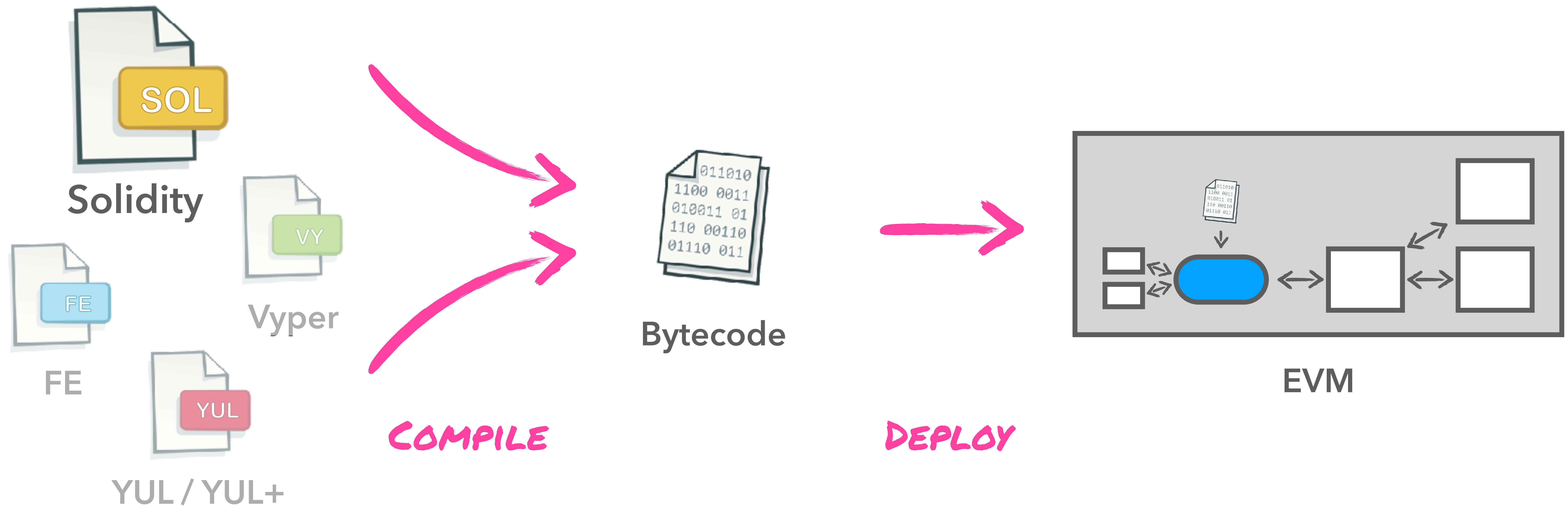
Gambling Example



Etherum Virtual Machine

SWIRLDS LABS

EVM Code



SWIRLDS LABS

@net0pyr | @hendrikEbbers

SwirldsLabs.com

OpCodes

Stack	Name	Gas	Initial Stack	Resulting Stack	Notes
00	STOP	0			halt execution
01	ADD	3	a, b	a + b	(u)int256 addition modulo 2**256
02	MUL	5	a, b	a * b	(u)int256 multiplication modulo 2**256
03	SUB	3	a, b	a - b	(u)int256 addition modulo 2**256
04	DIV	5	a, b	a // b	uint256 division
05	SDIV	5	a, b	a // b	int256 division
06	MOD	5	a, b	a % b	uint256 modulus
07	SMOD	5	a, b	a % b	int256 modulus

OpCodes

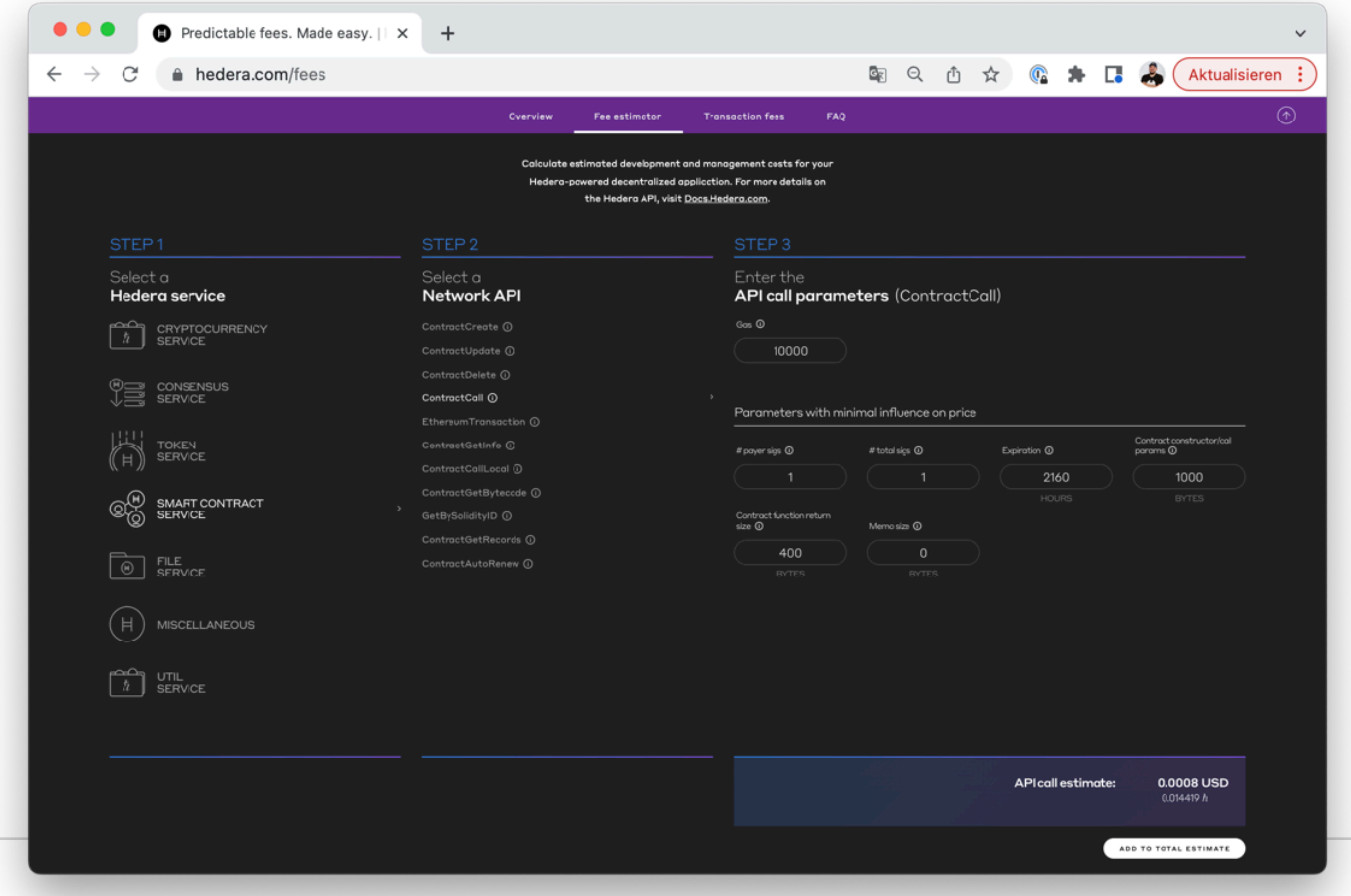
Stack	Name	Gas	Initial Stack	Resulting Stack	Notes
00	STOP	0			halt execution
01	ADD	3	a, b	a + b	(u)int256 addition modulo 2**256
02	MUL	5	a, b	a * b	(u)int256 multiplication modulo 2**256
03	SUB	3	a, b	a - b	(u)int256 addition modulo 2**256
04	DIV	5	a, b	a // b	uint256 division
05	SDIV	5	a, b	a // b	int256 division
06	MOD	5	a, b	a % b	uint256 modulus
07	SMOD	5	a, b	a % b	int256 modulus

Gas definition

- When creating a transaction a gas value needs to be defined
- The value defines the maximum of gas that the transaction can cost
- Transaction will be aborted if the cost is too high

```
transaction failed pre-check with
the status `INSUFFICIENT_GAS`
```

HAPI - Gas definition



Calculate estimated development and management costs for your Hedera-powered decentralized applicction. For more details on the Hedera API, visit Docs.Hedera.com.

STEP 1

Select a
Hedera service

- CRYPTOCURRENCY SERVICE
- CONSENSUS SERVICE
- TOKEN SERVICE
- SMART CONTRACT SERVICE
- FILE SERVICE
- MISCELLANEOUS
- UTIL SERVICE

STEP 2

Select a
Network API

- ContractCreate
- ContractUpdate
- ContractDelete
- **ContractCall**
- EthereumTransaction
- ContractGetInfo
- ContractCallLocal
- ContractGetBytecode
- GetBySolidityID
- ContractGetRecords
- ContractAutoRenew

STEP 3

Enter the
API call parameters (ContractCall)

Gas: 10000

Parameters with minimal influence on price

payer sigs: 1

total sigs: 1

Expiration: 2160 HOURS

Contract constructor/call params: 1000 BYTES

Contract function return size: 400 BYTES

Memo size: 0 BYTES

API call estimate: 0.0008 USD
0.014419 ℏ

CALCULATE ONLINE

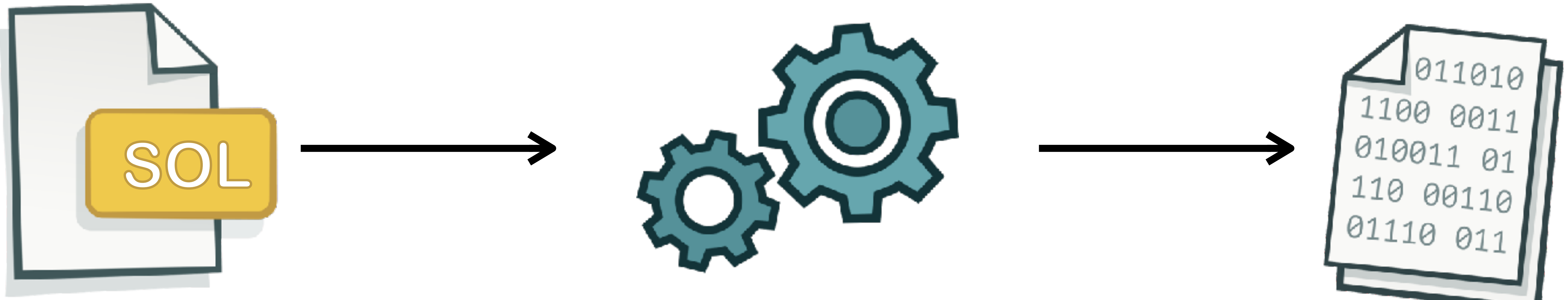
Compiling the first Smart Contracts

Hello World Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HelloWorldContract {

    function say_hello() public pure returns (string memory) {
        return "Hello, World!";
    }
}
```

Compile Solidity

- To execute the smart contract we need to compile it
- The compilation is normally stored in a binary BIN file



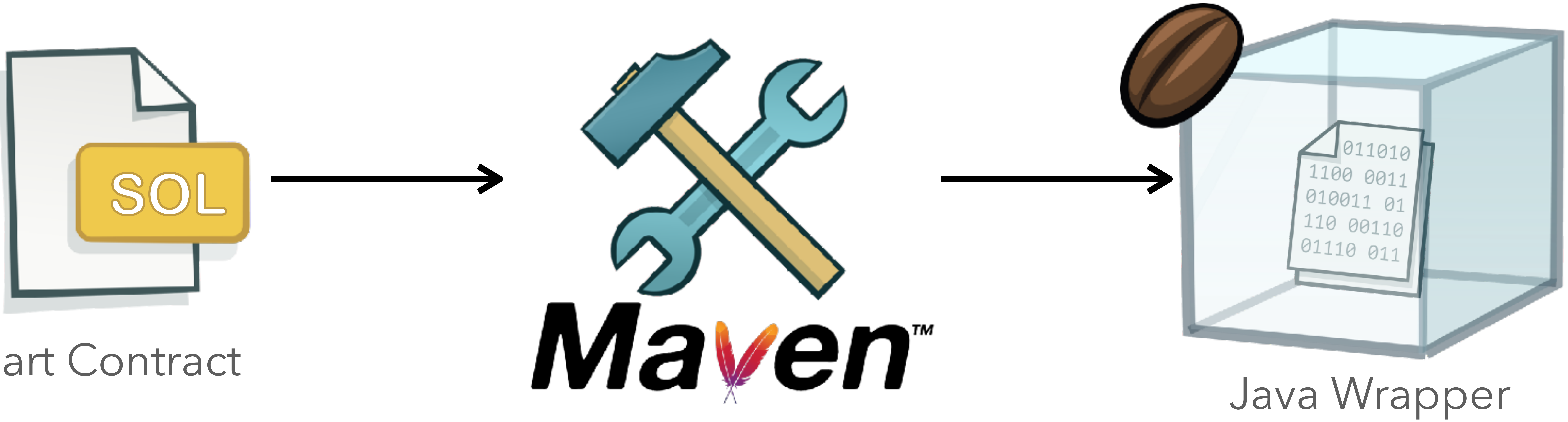
Smart Contract

solc
Solidity Compiler

compiled contract

Compile Solidity with Maven

- As a Java developer I want to integrate the compilation in my build

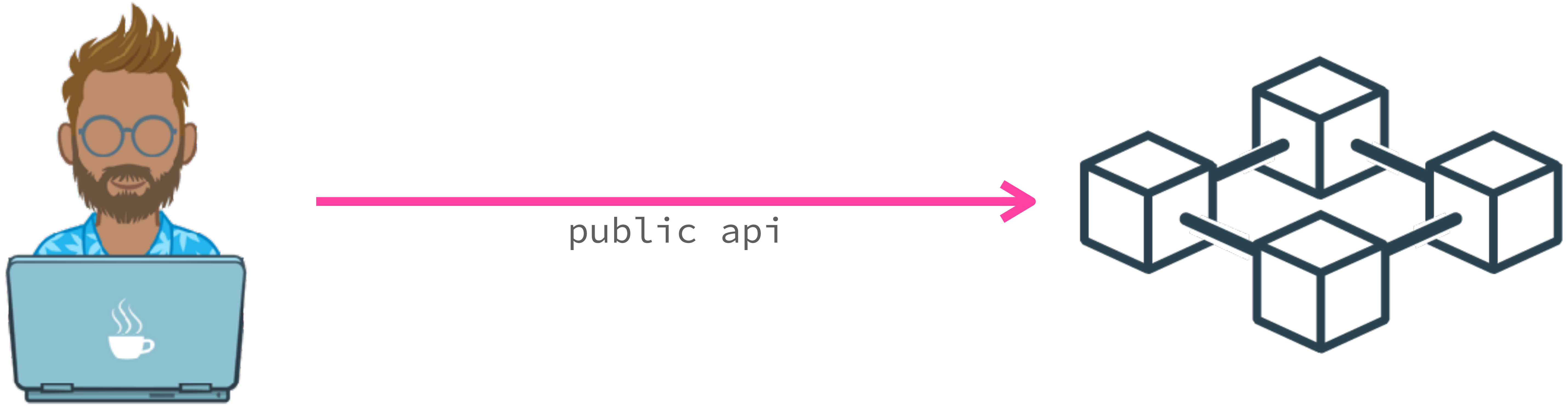


Smart Contract

Java Wrapper

Deploying a smart contract

- To execute a smart contract we need to deploy it on a ledger
- Public ledgers like Ethereum or Hedera provide public APIs to interact with the ledger



Accessing Hedera Hashgraph

SWIRLDS LABS

HAPI - Hedera API

- Rich documentation available online

  `https://docs.hedera.com/guides/docs/hedera-api`

- API libraries available for several languages

Java JS GO Swift .NET

[The Rust Programming Language C++] ← IN DEVELOPMENT

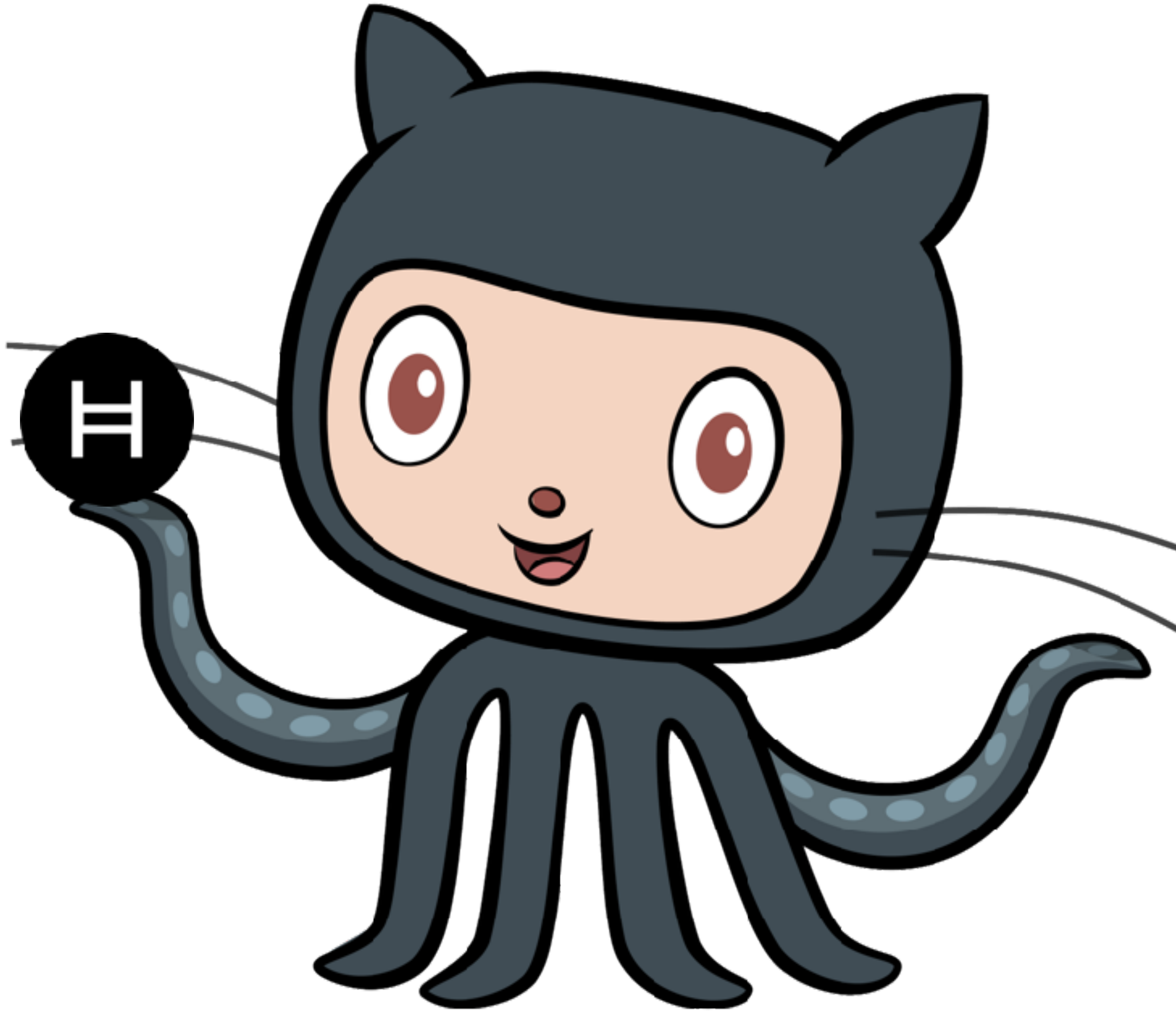
HAPI - Hedera API

- We will concentrate on Java

```
<dependency>
    <groupId>com.hedera.hashgraph</groupId>
    <artifactId>sdk</artifactId>
    <version>2.17.0</version>
</dependency>
```

- All Hedera sources can be found at GitHub

```
https://github.com/hashgraph/hedera-sdk-java
```

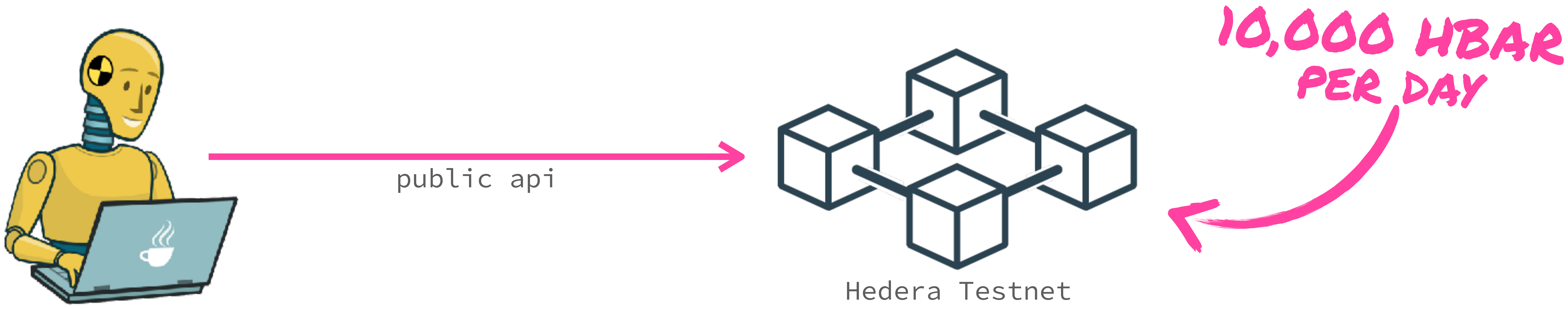


Hedera Testnet

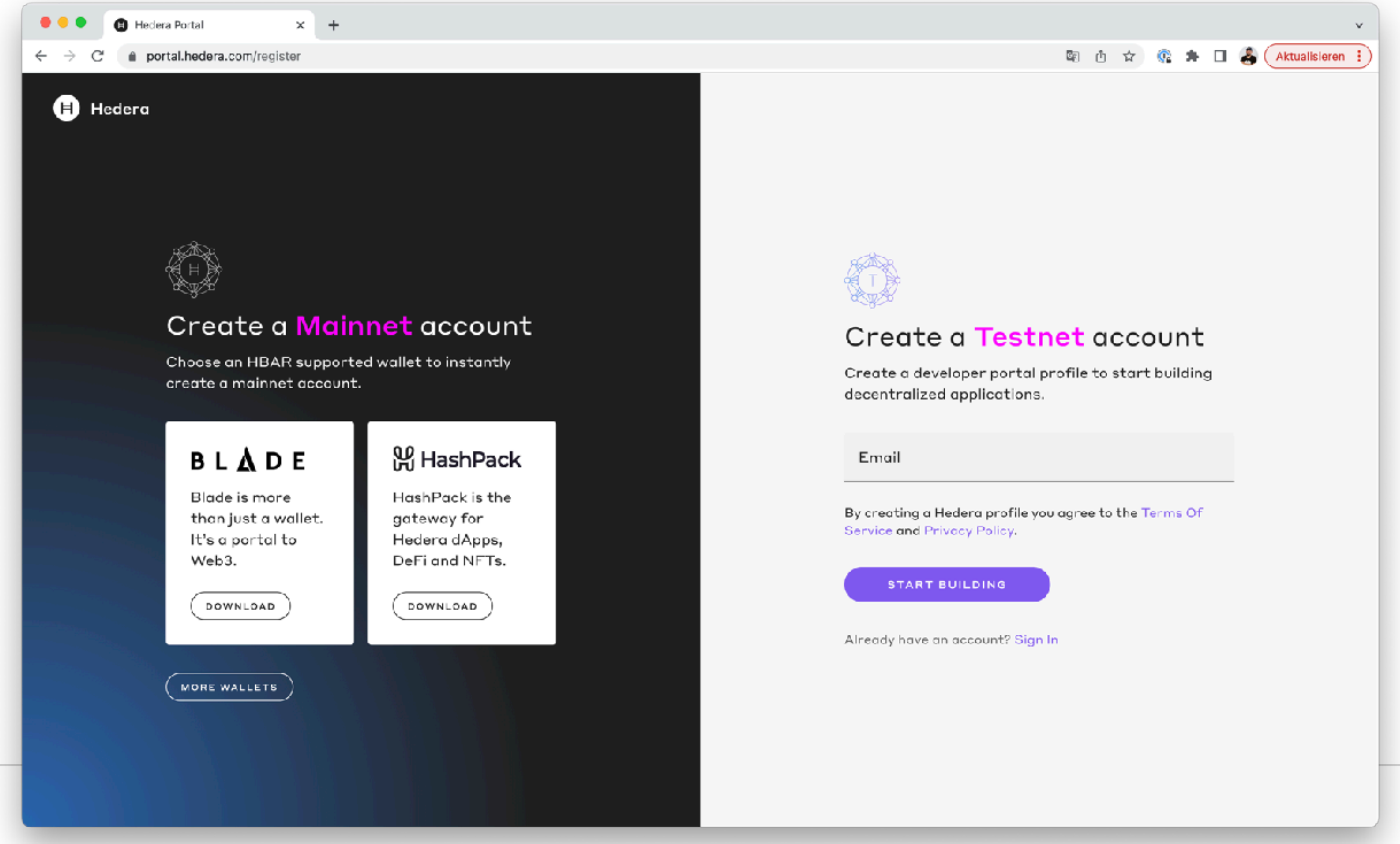
- We do not want to execute our contracts on the real Hedera ledger at devolpment time

- Hedera provides a test instance - Hedera Testnet

`https://docs.hedera.com/guides/testnet/testnet-access`



Hedera Testnet



Create a Mainnet account

Choose an HBAR supported wallet to instantly create a mainnet account.

BLADE

Blade is more than just a wallet. It's a portal to Web3.

HashPack

HashPack is the gateway for Hedera dApps, DeFi and NFTs.

Create a Testnet account

Create a developer portal profile to start building decentralized applications.

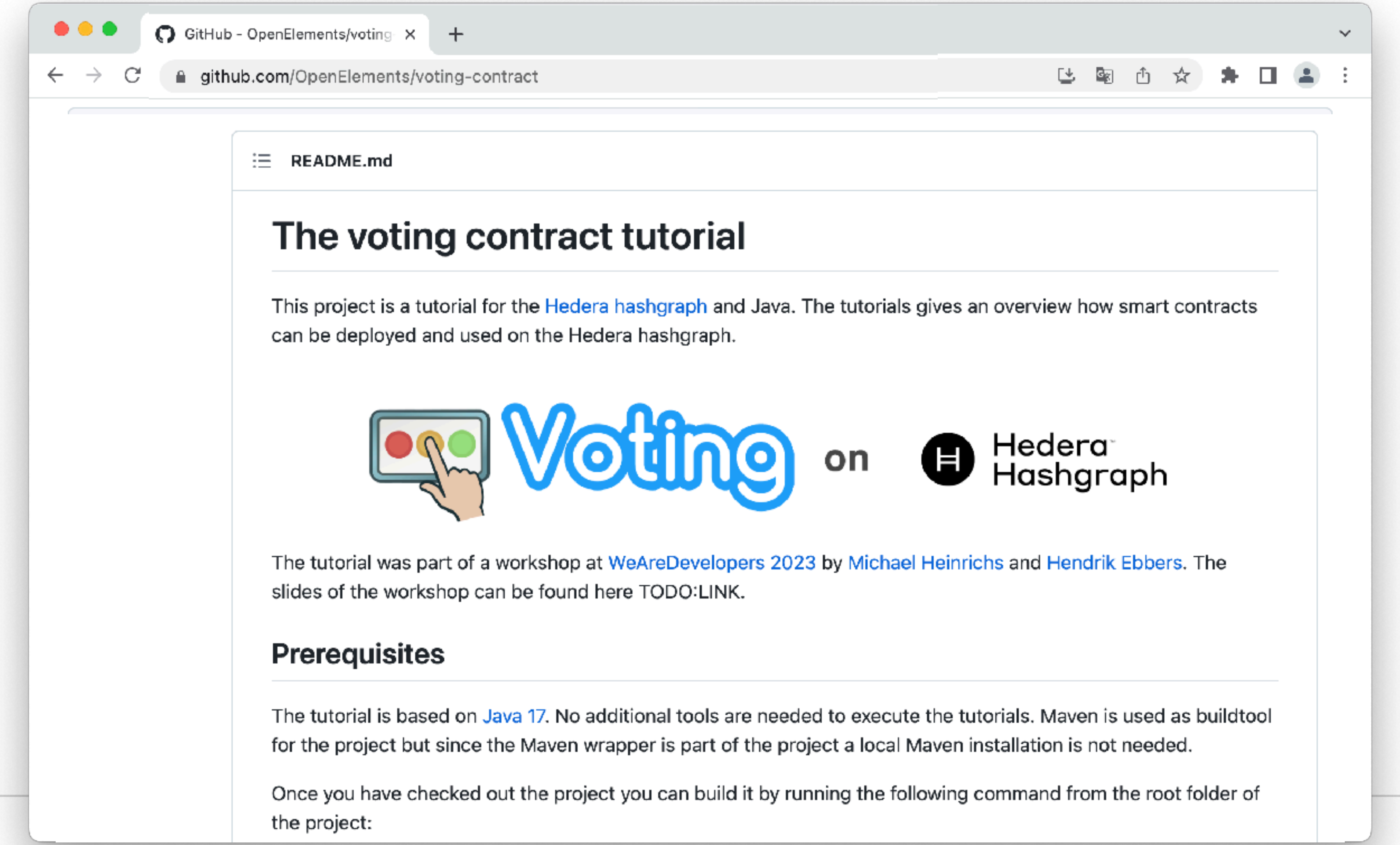
Email

By creating a Hedera profile you agree to the Terms Of Service and Privacy Policy.

Already have an account? Sign In

SCAN FOR URL

Our GitHub Repository


GitHub - OpenElements/voting
github.com/OpenElements/voting-contract
README.md
Voting
on
Hedera
Hashgraph

The voting contract tutorial

This project is a tutorial for the Hedera hashgraph and Java. The tutorials gives an overview how smart contracts can be deployed and used on the Hedera hashgraph.

The tutorial was part of a workshop at WeAreDevelopers 2023 by Michael Heinrichs and Hendrik Ebbers. The slides of the workshop can be found here TODO:LINK.

Prerequisites

The tutorial is based on Java 17. No additional tools are needed to execute the tutorials. Maven is used as buildtool for the project but since the Maven wrapper is part of the project a local Maven installation is not needed.

Once you have checked out the project you can build it by running the following command from the root folder of the project:

SCAN FOR URL

Let's code

SWIRLDS LABS

Contracts with Solidity

SWIRLDS LABS

Solidity

- The Solidity language documentation can be found at

`https://docs.soliditylang.org/`

Solidity

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Building Blocks

VERSION PRAGMA

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Building Blocks

CONTRACT

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Building Blocks

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

STATE VARIABLES

Building Blocks

EVENTS

```solidity
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Building Blocks

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Constructor

Building Blocks

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

FUNCTIONS

Solidity

```
pragma solidity >= 0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Value Types

- Boolean
- Integers
  (int, int8, int16, ..., int256, uint, uint8, uint16, ..., uint256)
- Fixed Point Numbers (👷)
- Address
- Byte Arrays (fixed and dynamically-sized)
- Enums

Reference Types

- Array
- Map
- Struct

Control Structures

- if, else
- while, do
- for
- break, continue
- return

Error Handling

- state-reverting
- try-catch
- require
- revert

Compile Solidity

- The Solidity compiler **`solc`** can easy be installed locally

`https://docs.soliditylang.org/`

- The compiler provides different ways how it can be installed locally (brew, npm, …)

Compile Solidity

- We can easily compile our smart contract by using solc from the commandline:

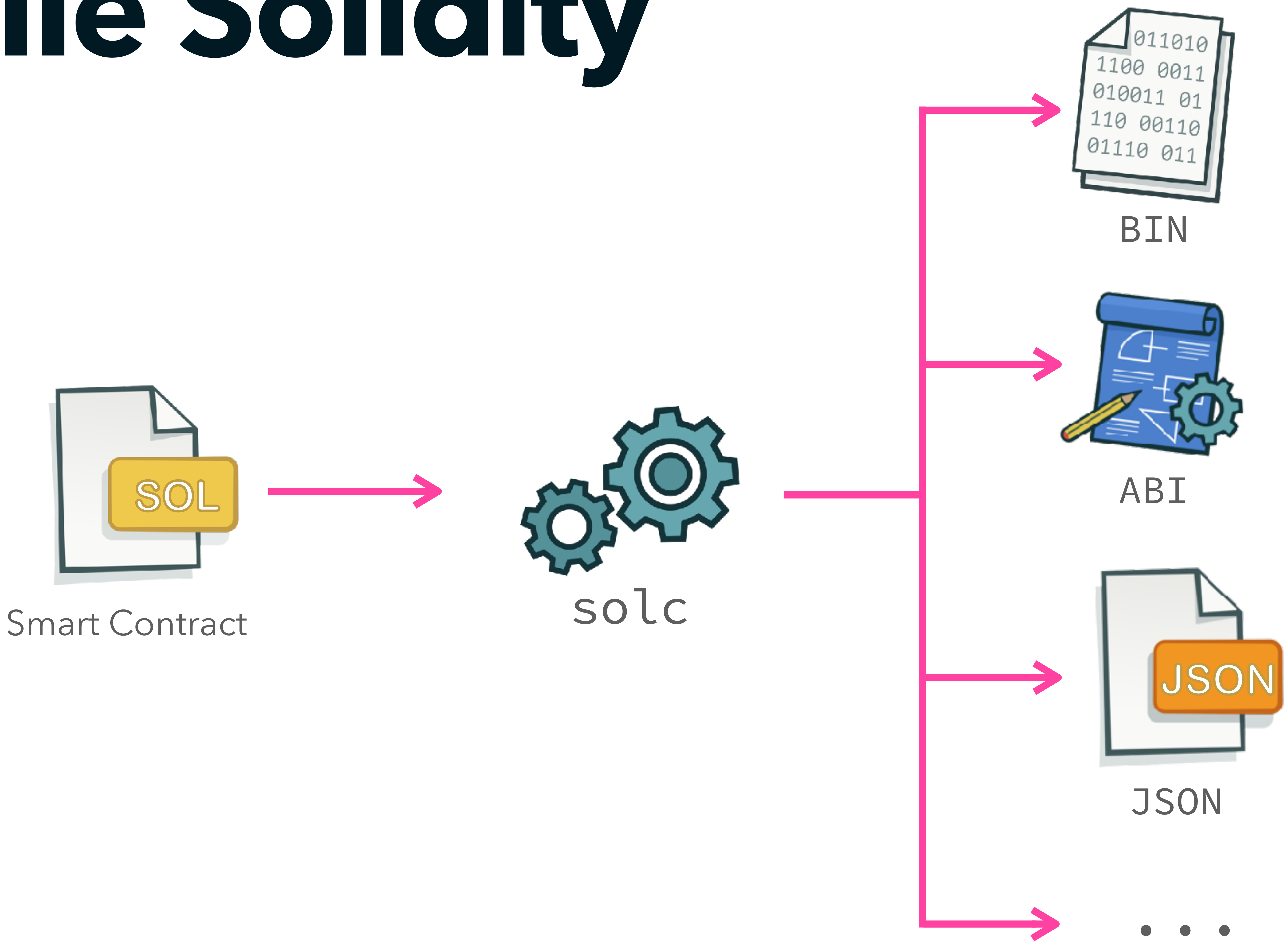
```
solc --bin -o build/contracts contracts/hello_world.sol
```

WE WANT TO
CREATE THE BIN FILE

OUTPUT FOLDER

INPUT

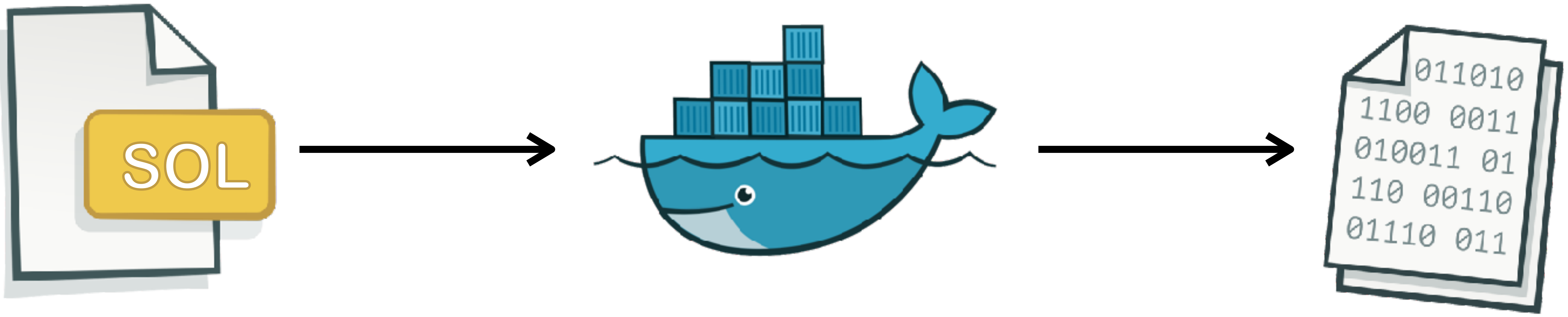
Compile Solidity



Compile Solidity

- Instead of installing the compiler locally you can use it wrapped in a docker container

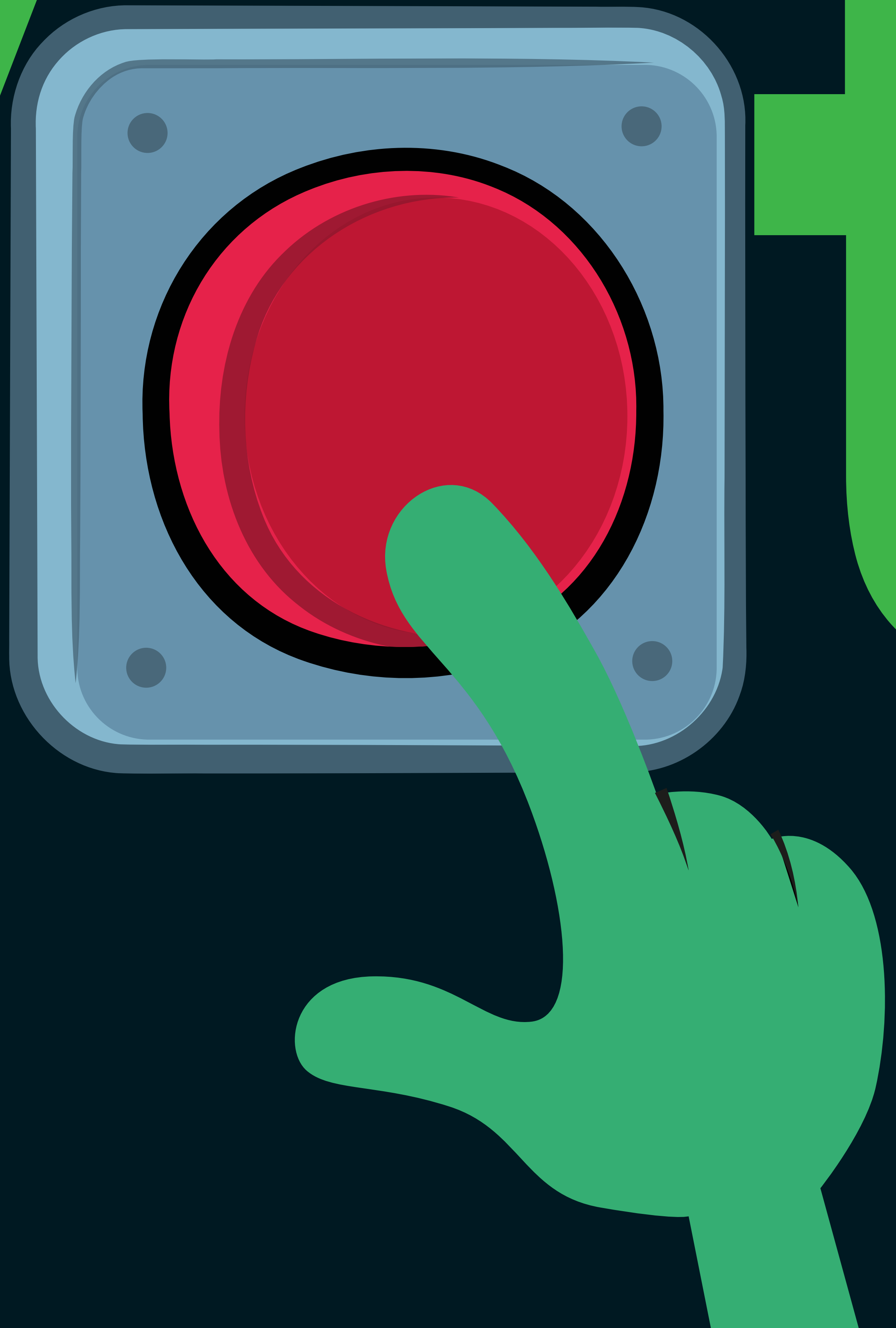
```
docker run -v $(pwd)/contracts:/contracts ethereum/solc:stable -o /contracts/output --abi --bin
```



Smart Contract

solc container

compiled contract

A contract for Vting

SWIRLDS LABS

Voting Contract V1

- Setup proposals
- Vote for a proposal
- Show winner

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    constructor(bytes32[] memory proposalNames) {

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}
}
```

Data Structure

- Proposal has name and counter
- Several proposals stored in array
- Proposals are publicly readable

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    struct Proposal {
        bytes32 name;
        uint count;
    }

    Proposal[] public proposals;

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}
}
```

Setup proposals

- Iterate over all proposal names
- Create a Proposal
- Add the proposal to the proposal-array

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {
        for (uint i = 0; i < proposalNames.length; i++) {
            Proposal memory proposal = Proposal({
                name: proposalNames[i],
                count: 0
            });
            proposals.push(proposal);
        }
    }

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}
}
```

Vote for a proposal

- Increase the counter

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {
        proposals[proposal].count++;
    }

    function winner() public view returns (bytes32 result) {}
}
```

Show winner

- Iterate over all proposals
- Find proposal with most votes
- Check condition:

  ```
  if (condition) {}
  ```

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {
        result = "";
        ...

                    ?

    }
}
```

Show winner

- Iterate over all proposals
- Find proposal with most votes
- New elements:
  `if (condition) {}`

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {
        result = "";
        uint maxCount = 0;
        for (uint i = 0; i < proposals.length; i++) {
            if (proposals[i].count > maxCount) {
                maxCount = proposals[i].count;
                result = proposals[i].name;
            }
        }
    }
}
```

Voting Contract V2

- Authorize voters
- One vote per user

Setup admin

- Creator of contract becomes admin
- Users identified by address
- `msg` contains metadata of message

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    address public admin;

    ...

    constructor(bytes32[] memory proposalNames) {
        admin = msg.sender;
        ...
    }

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}
}
```

Safeguard vote()

- Voter can have one of three states
- State for all voters is stored in mapping
- `require` checks condition

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    enum VoterState { NotAuthorized, Authorized, Voted }

    mapping(address => VoterState) public voters;

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {
        require(
            voters[msg.sender] == VoterState.Authorized,
            "Not authorized“
        );

        voters[msg.sender] = VoterState.Voted;
        proposals[proposal].count++;
    }

    function winner() public view returns (bytes32 result) {}
}
```

Authorize users

- Only admin can authorize voters
- Only unauthorized users can be authorized

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}

    function authorize(address voter) public {

                ?

    }
}
```

Authorize users

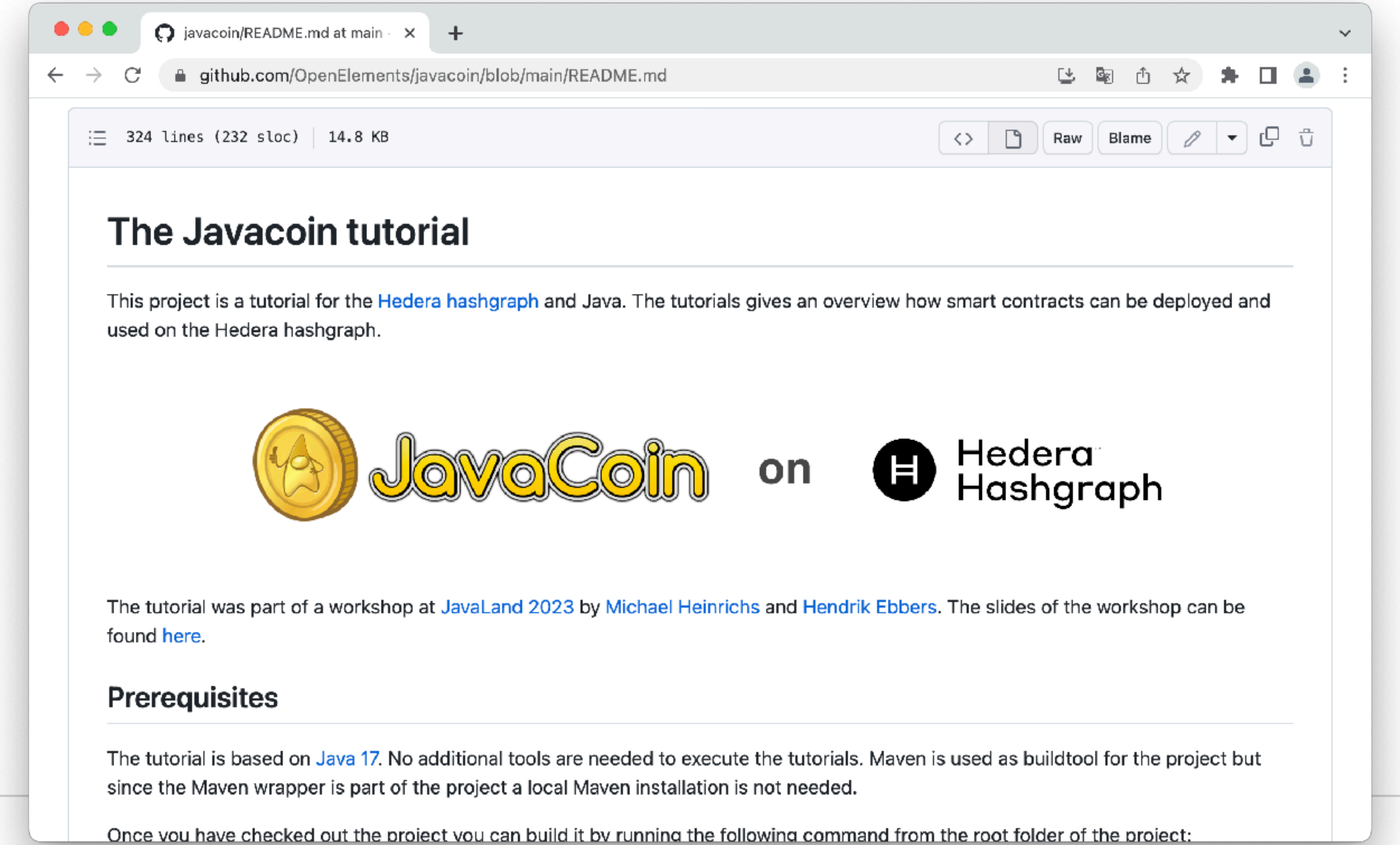
- Only admin can authorize voters
- Only unauthorized users can be authorized

```
pragma solidity >= 0.7.0 <0.9.0;

contract Poll {

    ...

    constructor(bytes32[] memory proposalNames) {}

    function vote(uint proposal) public {}

    function winner() public view returns (bytes32 result) {}

    function authorize(address voter) public {
        require(
            msg.sender == admin,
            "Only admin can authorize"
        );
        require(
            voters[voter] == VoterState.NotAuthorized,
            "Already authorized"
        );

        voters[voter] = VoterState.Authorized;
    }
}
```

Additional Ressources

SWIRLDS LABS

Our Token Workshop


javacoin/README.md at main ·
github.com/OpenElements/javacoin/blob/main/README.md
324 lines (232 sloc) | 14.8 KB
Raw
Blame
JavaCoin
on
Hedera Hashgraph

The Javacoin tutorial

This project is a tutorial for the Hedera hashgraph and Java. The tutorials gives an overview how smart contracts can be deployed and used on the Hedera hashgraph.

The tutorial was part of a workshop at JavaLand 2023 by Michael Heinrichs and Hendrik Ebbers. The slides of the workshop can be found here.

Prerequisites

The tutorial is based on Java 17. No additional tools are needed to execute the tutorials. Maven is used as buildtool for the project but since the Maven wrapper is part of the project a local Maven installation is not needed.

Once you have checked out the project you can build it by running the following command from the root folder of the project:

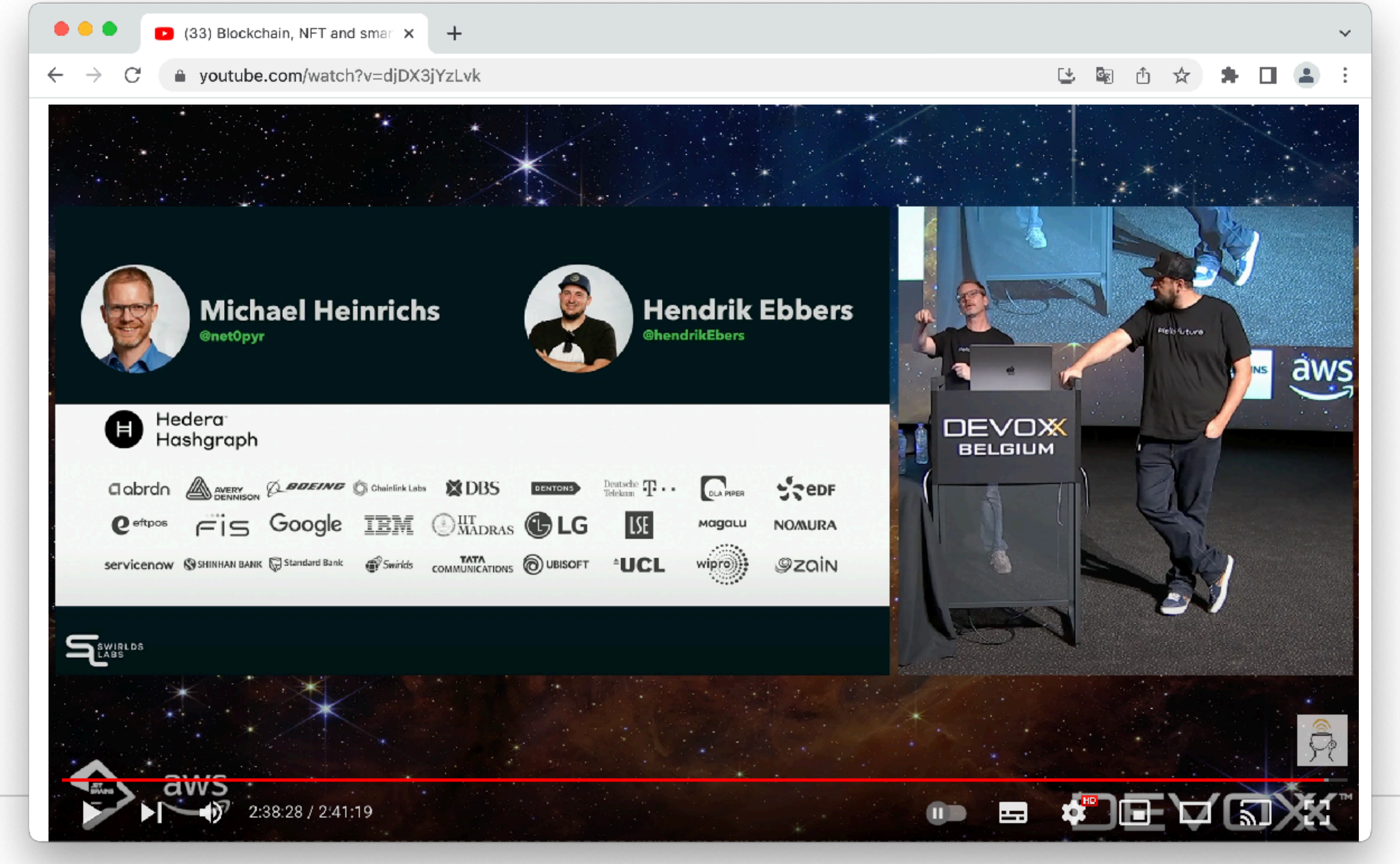
SCAN FOR URL

SWIRLDS LABS

@net0pyr | @hendrikEbbers

SwirldsLabs.com

Our Crypto Deep Dive



(33) Blockchain, NFT and smar

youtube.com/watch?v=djDX3jYzLvk

Michael Heinrichs
@net0pyr

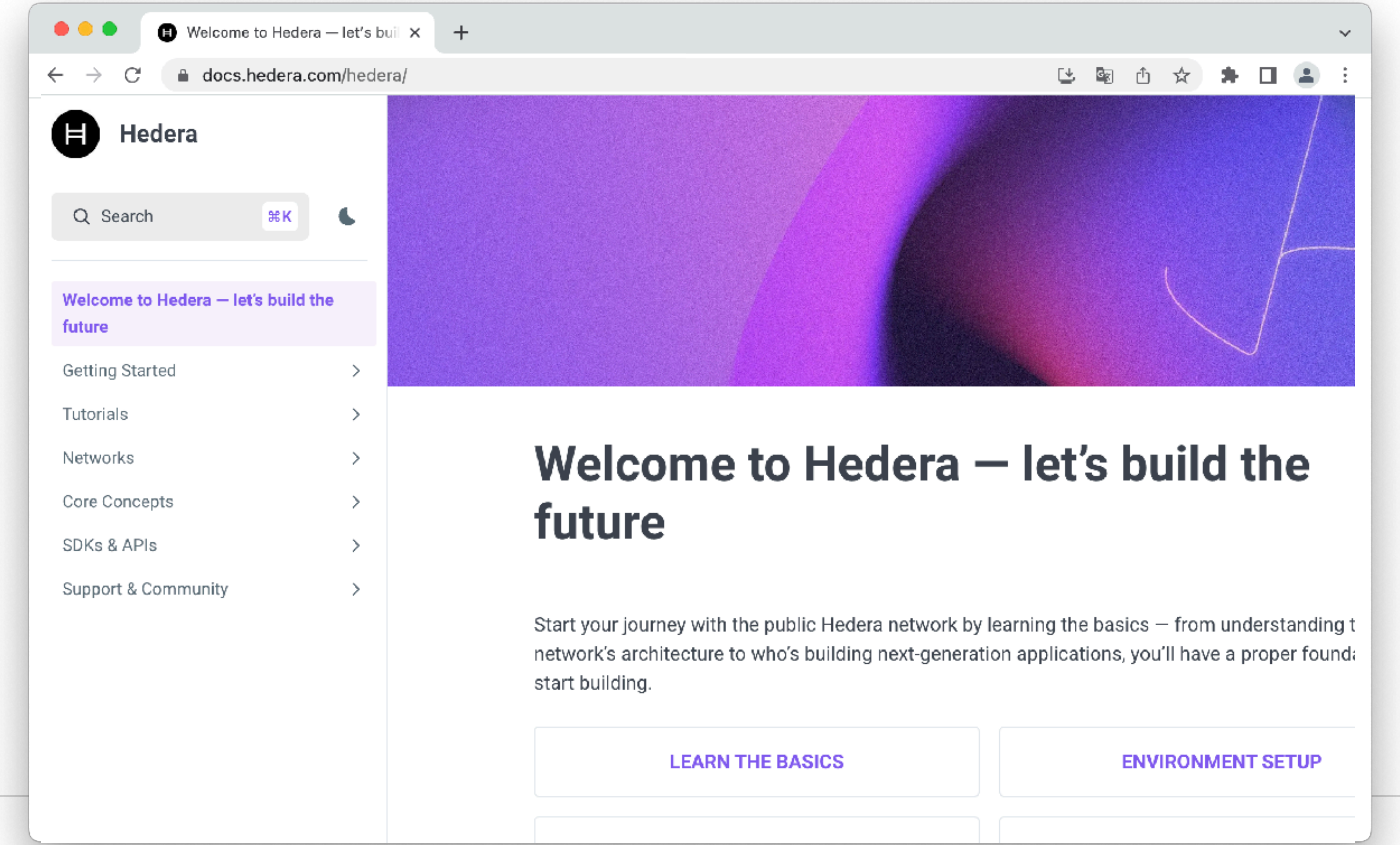
Hendrik Ebbers
@hendrikEbers

Hedera Hashgraph

2:38:28 / 2:41:19

SCAN FOR URL

The Hedera Documentation



Welcome to Hedera — let's build the future

Start your journey with the public Hedera network by learning the basics — from understanding t network's architecture to who's building next-generation applications, you'll have a proper founda start building.

LEARN THE BASICS

ENVIRONMENT SETUP

SCAN FOR URL

Michael Heinrichs

@net0pyr

Hendrik Ebbers

@hendrikEbers

Hedera Hashgraph

abrdn · AVERY DENNISON · BOEING · Chainlink Labs · DBS · DENTONS · Deutsche Telekom · DLA PIPER · eDF

eftpos · FIS · Google · IBM · IIT MADRAS · LG · LSE · Magalu · NOMURA

servicenow · SHINHAN BANK · Standard Bank · Swirlds · TATA COMMUNICATIONS · UBISOFT · UCL · wipro · zain

SWIRLDS LABS