



中文Bootcamp I

Reddio



✦ December 2023

🐦 [Reddio_com](https://reddio.com)

水綠

養桂

蘭培

風和

文章

氣紫

義道

暖



About Bootcamp

Onboarding devs to Starknet

Starknet Foundation Initiative

December course taught live

5 hours sessions

Chat, Q&A and Homework

Private Wechat channel

Community collaboration



Tim

Smart Contract Engineer

1 year on Cairo.

3 years on Solidity.

Specialized in DeFi, NFT and Web3 security.

Twitter: [@xyyme.eth](https://twitter.com/xyyme.eth)



Neil Han

Founder, Reddio

Envenglise StarkWare technologies for 2 years

NTU Blockchain Master Program, Guest lecturer

ex-Twilio, ex-PingCAP/TiDB

Twitter: [@NeilHANYD](https://twitter.com/NeilHANYD)



The Sessions

- 1) Starknet基本原理和生态 ← **you are here**
- 2) Starknet 与 Cairo 开发初探
- 3) Cairo 合约的编写与部署
- 4) Cairo 组件的编写与使用
- 5) Starknet前端集成





Starknet基本原理和Starknet生态

中文Bootcamp I - Session 1

✦ December 2023

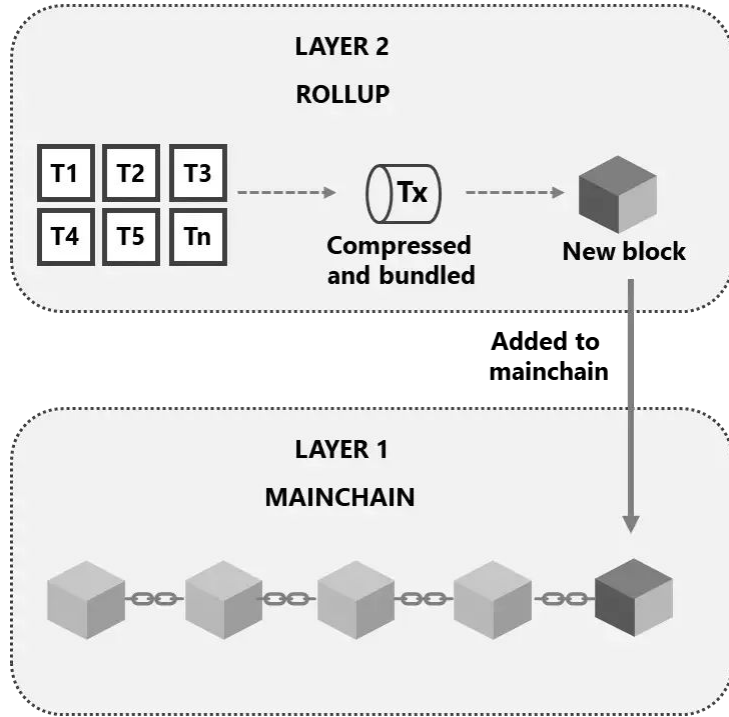
🐦 [Reddio.com](https://reddio.com)

Agenda

1. Why StarkWare/zkRollup
2. Why Starknet
3. Why Cairo
4. Starknet生态



Why StarkWare/zkRollup?



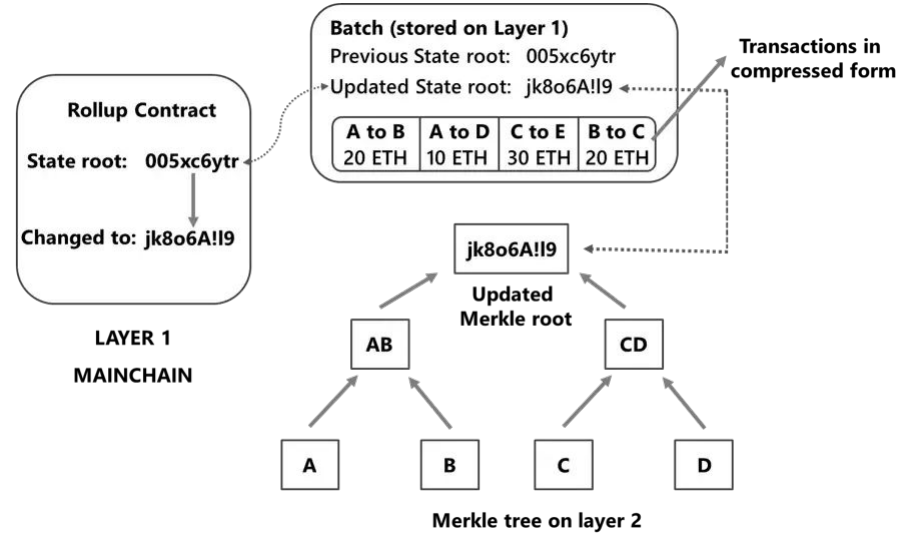
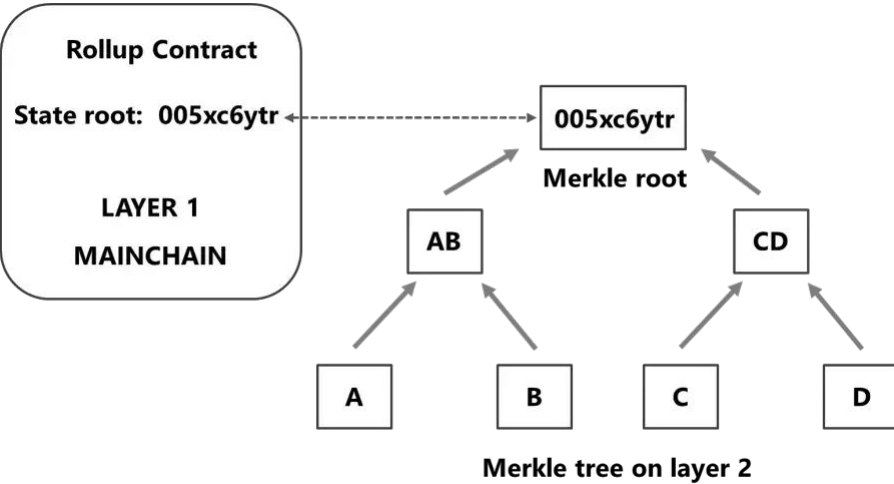
Pros.

- Data posted on the mainchain is the minimum
- Each batch of executed transactions is bundled and is posted on the mainchain
- Validate the rollups transaction required
- Much more secured

Cons.

- A major limitation of Optimistic rollups is the longer withdrawal time
- Still in the middle of maturing for zkRollup

🗣️ | How does Rollups work



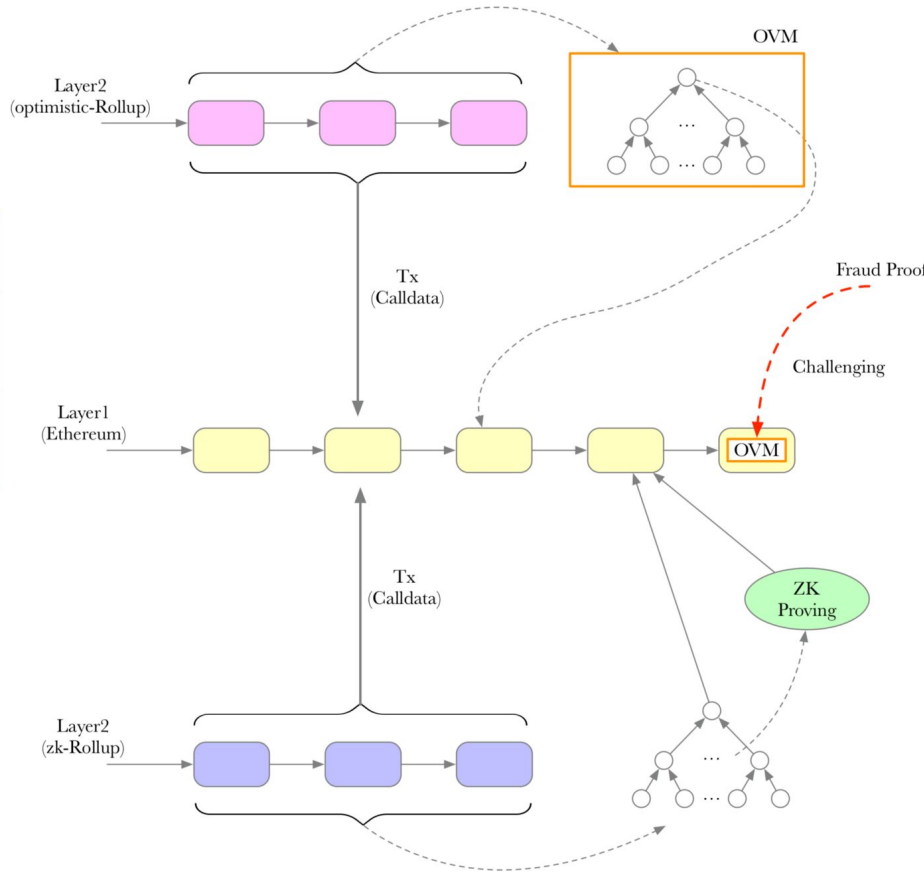
Rollup contract on the mainchain contains state root of the layer2

State root is updated on rollup contract when transactions are executed on the layer2

Rollup Architecture

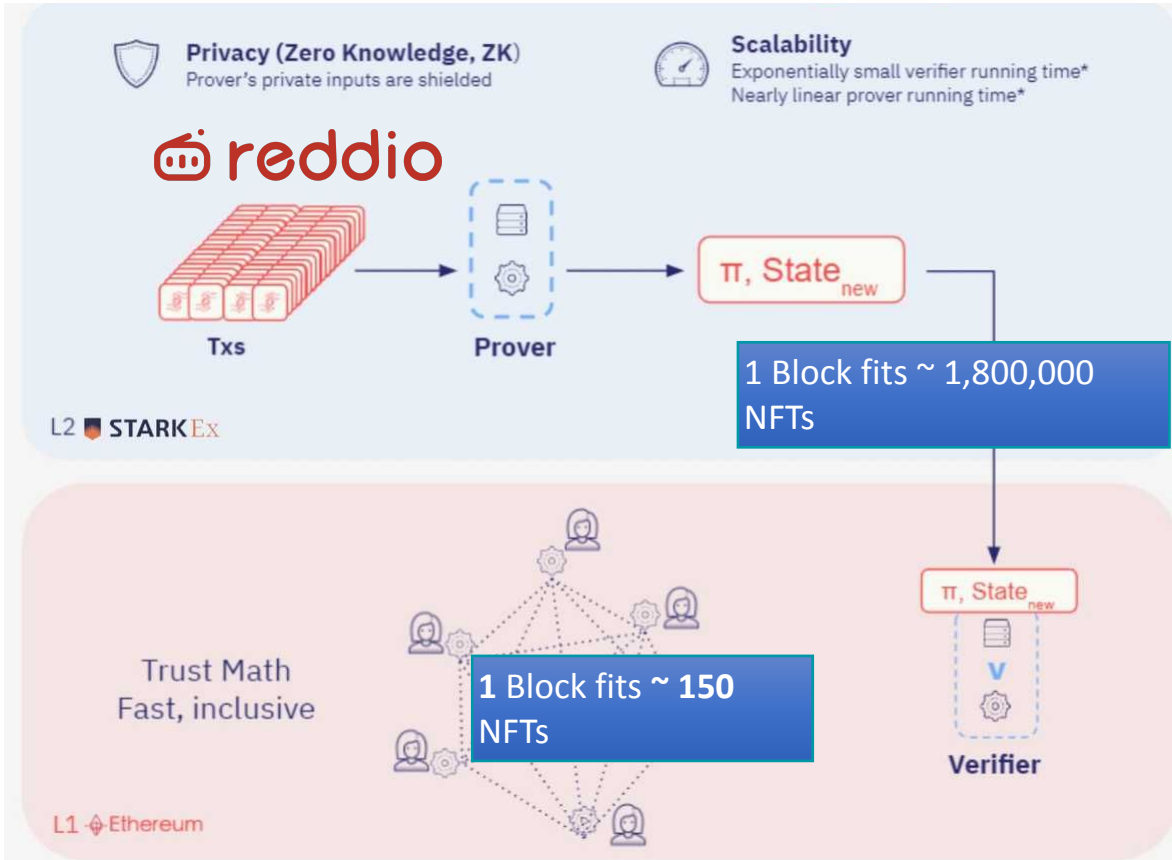
Powering Next Generation Apps

	Validity Proofs (STARKs or SNARKs)	Fraud Proofs
On-Chain Data	zK-Rollup	Optimistic Rollup
Off-Chain Data	Validium	
	Volition	
		Plasma



Volition/Validium/zkRollup: StarkEx

Powering Next Generation Apps



SALES RECEIPT

Date: May 22nd 2019

Qty.	Description	Price	Amount
1	Spinach Salad	\$8.50	\$ 8.50
1	Lamb Tagine	\$ 19.00	\$ 19.00
4	Side Rice	\$ 4.00	\$ 16.00
2	Coke	\$ 2.50	\$ 5.00
2	Beer	\$ 14.00	\$ 28.00
		Subtotal:	\$76.50
		Tax:	\$12.00
		Total:	\$89.50

Sale Made with :

- Cash
- Credit Card
- Check, No. _____
- Other _____

STATEMENT
total=\$89.50

PROVER
Party producing proof
(Restaurant owner)

VERIFIER
Party checking proof
(Customer)

STARK

“In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware”

Problem [1992]: Not scalable; Not enough atoms in solar system to write a proof

Eureka I [2005]: Scalable, barely enough atoms in solar system to write a proof

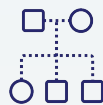
Eureka II [2018]: enough atoms on *laptop* to write a proof!

StarkWare (Est. 2018)



Mission

Integrity through Math



Pedigree

Invented ZK-STARK, FRI, Cairo, SHARP, Validium, Volition, Layer 3...



100+

Team members

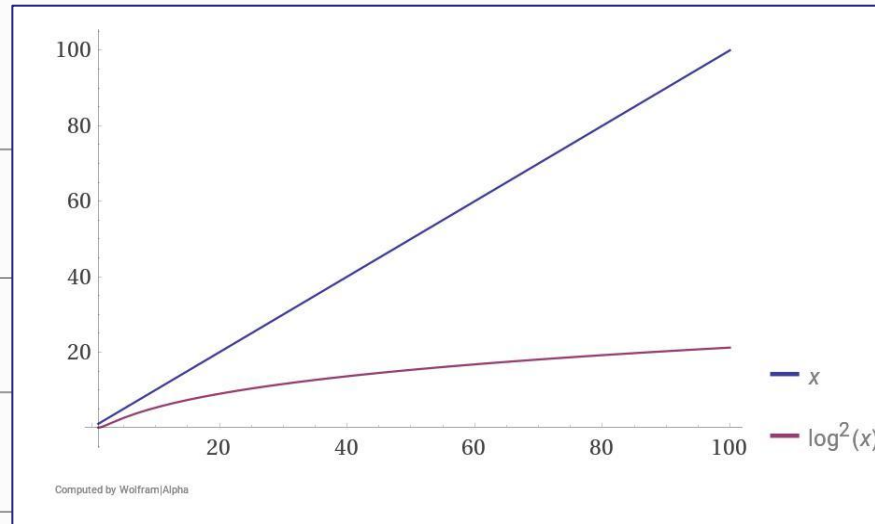


\$260+M

Funding (equity + EF grant)

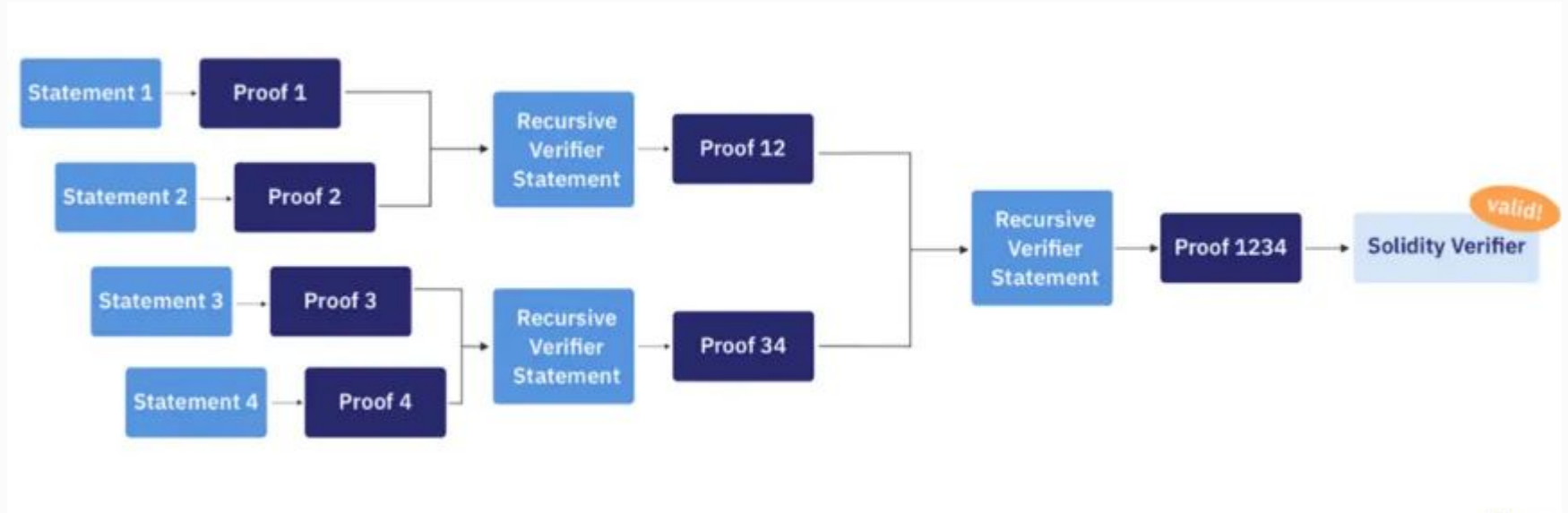
STARK vs SNARK

	STARK	SNARK
Verification	$\log^2(n)$	constant
Proof Size	~400 KB	288 b
Proving Time	1x	10x
Trusted Setup	No	Yes
Quantum Secure	Yes	No



* Improved with proof recursion

What about Proof Size?





As of September 12, 2022



Launched - June 2020

\$ 711 Billion

Cumulative Trading

230M

Tx Settled

11.6M

Max txs/week*
33% more than Ethereum

66M

NFTs Minted

600K

NFT Mints/Proof

<500

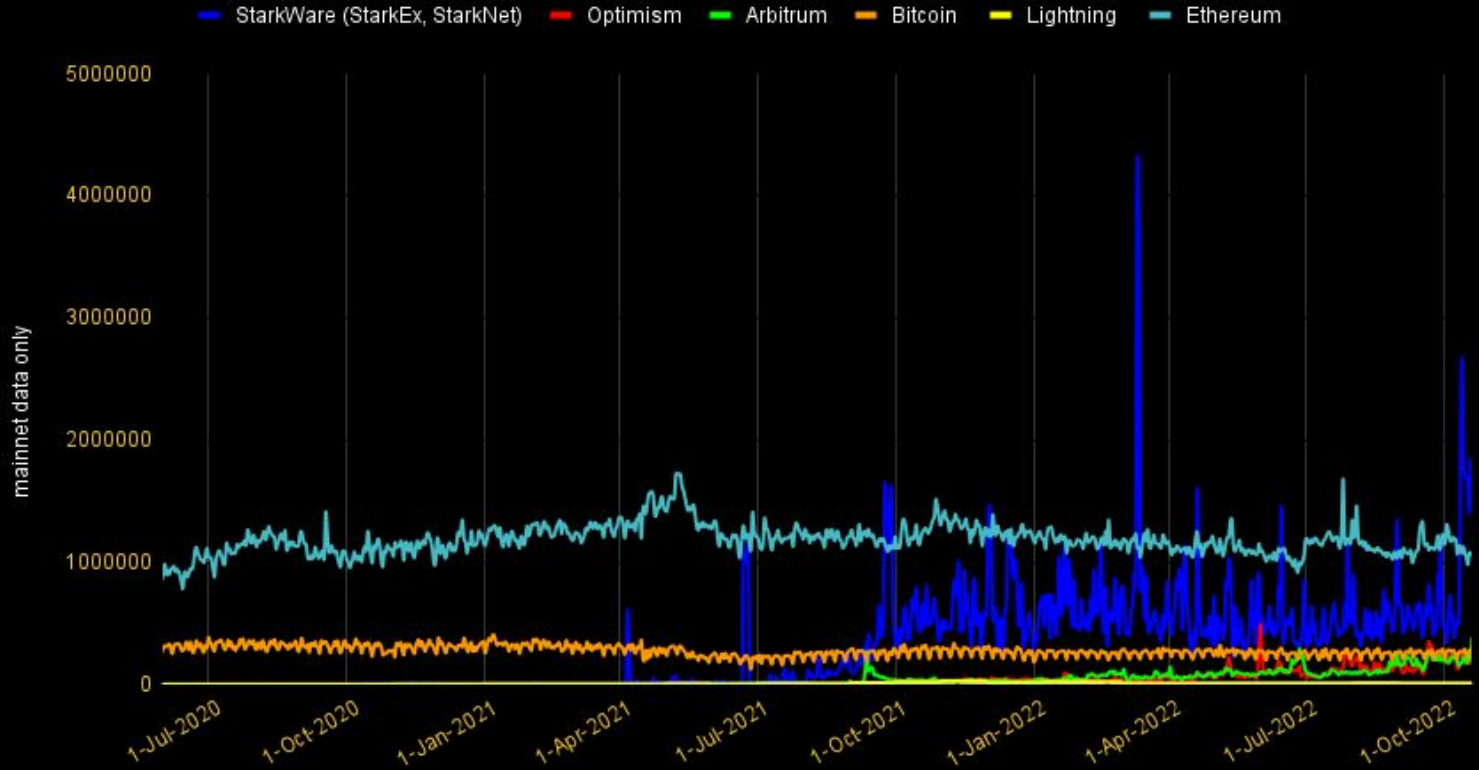
Gas/tx



* March 6-12, 2022

@elibensasson | @starkwareltd 30

Reddit | Daily Transactions: Ethereum & L2s vs Bitcoin & Lightning



source: starkware, arbitrum, optimism, for btc nasdaq, lightning arcane estimates

What is StarkNet?

A Decentralized Permissionless STARK-based Validity-Rollup,
offering scalable & secure Ethereum-like state

L2

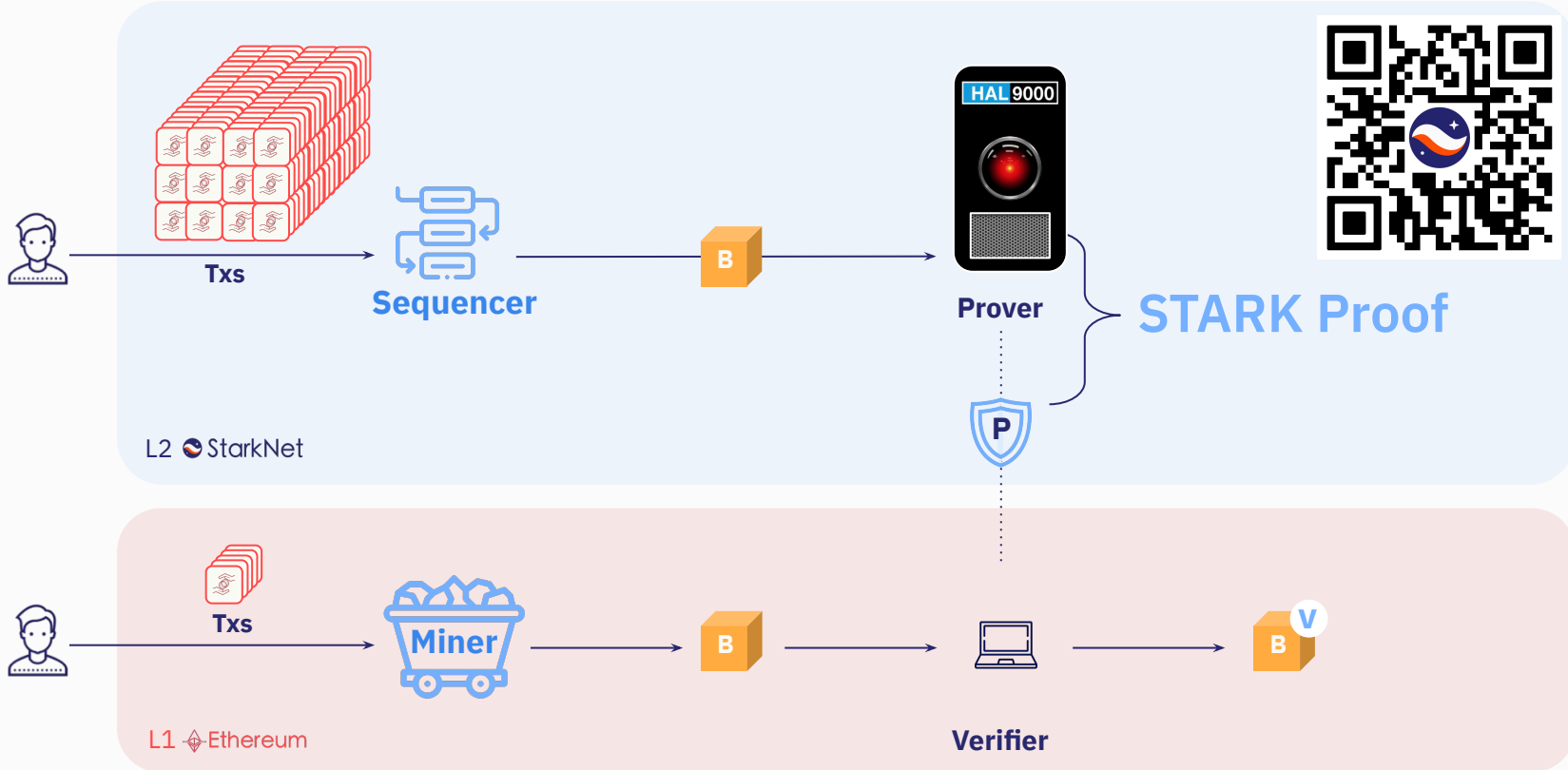
SMART CONTRACTS

GENERAL
COMPUTATION

COMPOSABILITY



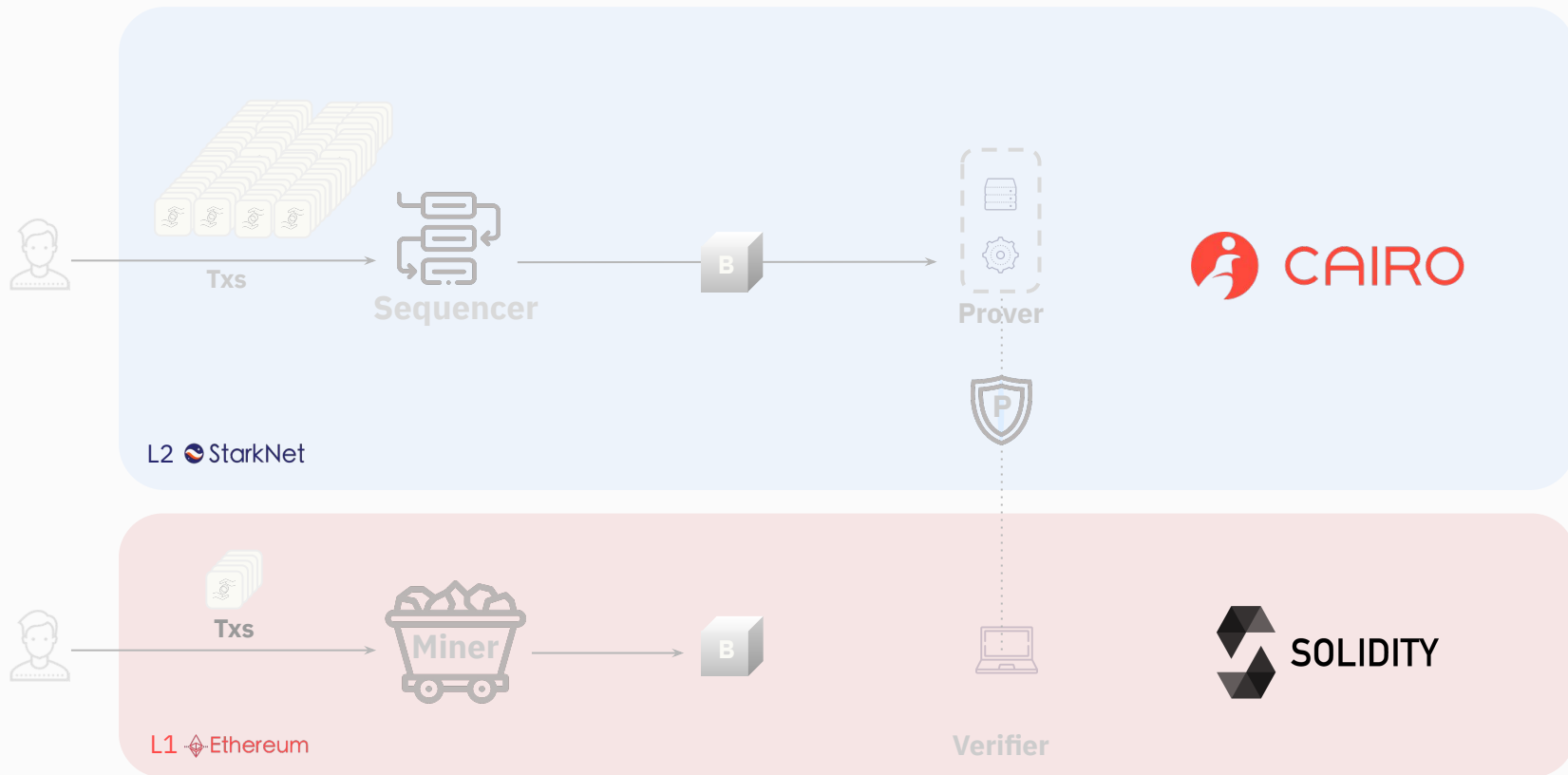
StarkNet Enhances Ethereum




Cairo

The Language of StarkNet

New Technology \Rightarrow New Language



New Technology \Rightarrow New Language

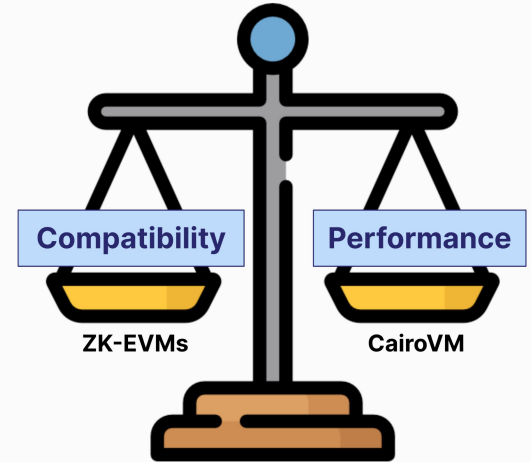
 **SOLIDITY** is a new language, compared to C, Python, Rust, ...

But it's the right language for Ethereum contracts

Likewise,  **CAIRO** is the right language for StarkNet contracts

Why? STARKs have different constraints

- Algebraic steps mod p are cheap
- Bitwise ops, Keccak, etc. are expensive



Vitalik's zk-EVM Classification

	EVM Changes	Compatibility*	Performance**	Projects
Type 1	Nothing	Full	Very Slow	-
Type 2	Storage data structure	High	Slow	-
Type 3	Storage, hashes, precompiles	Partial	Fast	Kakarot , zkSync, Scroll, Polygon zkEVM
Type 4	Completely different VM	None	Very Fast	Starknet , Polygon Miden

*how feasible it is to execute an Ethereum smart contract without any change

**how much time and resources does it take to create a validity proof

```
struct Rectangle {  
    height: u64,  
    width: u64,  
}
```

Custom data types

Scalar types

```
trait ShapeGeometry {  
    fn boundary(self: Rectangle) -> u64;  
    fn area(self: Rectangle) -> u64;  
}
```

Functionality blueprint

Method signature

```
impl RectangleGeometry of ShapeGeometry {  
    fn boundary(self: Rectangle) -> u64 {  
        2 * (self.height + self.width)  
    }  
    fn area(self: Rectangle) -> u64 {  
        self.height * self.width  
    }  
}
```

Implementations have names

Method implementation

```
#[starknet::interface] ←----- Contract's interface
trait ISimpleStorage<TContractState> {
  fn set(ref self: TContractState, x: u128); ←----- Reference ⇒ Modifies state
  fn get(self: @TContractState) -> u128; ←----- Snapshot ⇒ Read-only
}

#[starknet::contract] ←----- Metaprogramming
mod SimpleStorage {
  #[storage]
  struct Storage {
    stored_data: u128 ←----- Smart contract state
  }

  #[external(v0)] ←----- Public methods
  impl SimpleStorage of super::ISimpleStorage<ContractState> { ←----- Interface implementation
    fn set(ref self: ContractState, x: u128) {
      self.stored_data.write(x); ←----- Writes to storage
    }
    fn get(self: @ContractState) -> u128 {
      self.stored_data.read() ←----- Reads from storage
    }
  }
}
```


Cairo's Features



Creates **provable** programs

Runs on top of CairoVM

Syntax inspired by Rust

Similar ownership model

Strongly typed

Can be used outside of Starknet

No need to know ZK!



Summary

Why Cairo?

Creates **provable** programs

Proof of computational integrity

Verification without re-execution

Powerful & flexible language

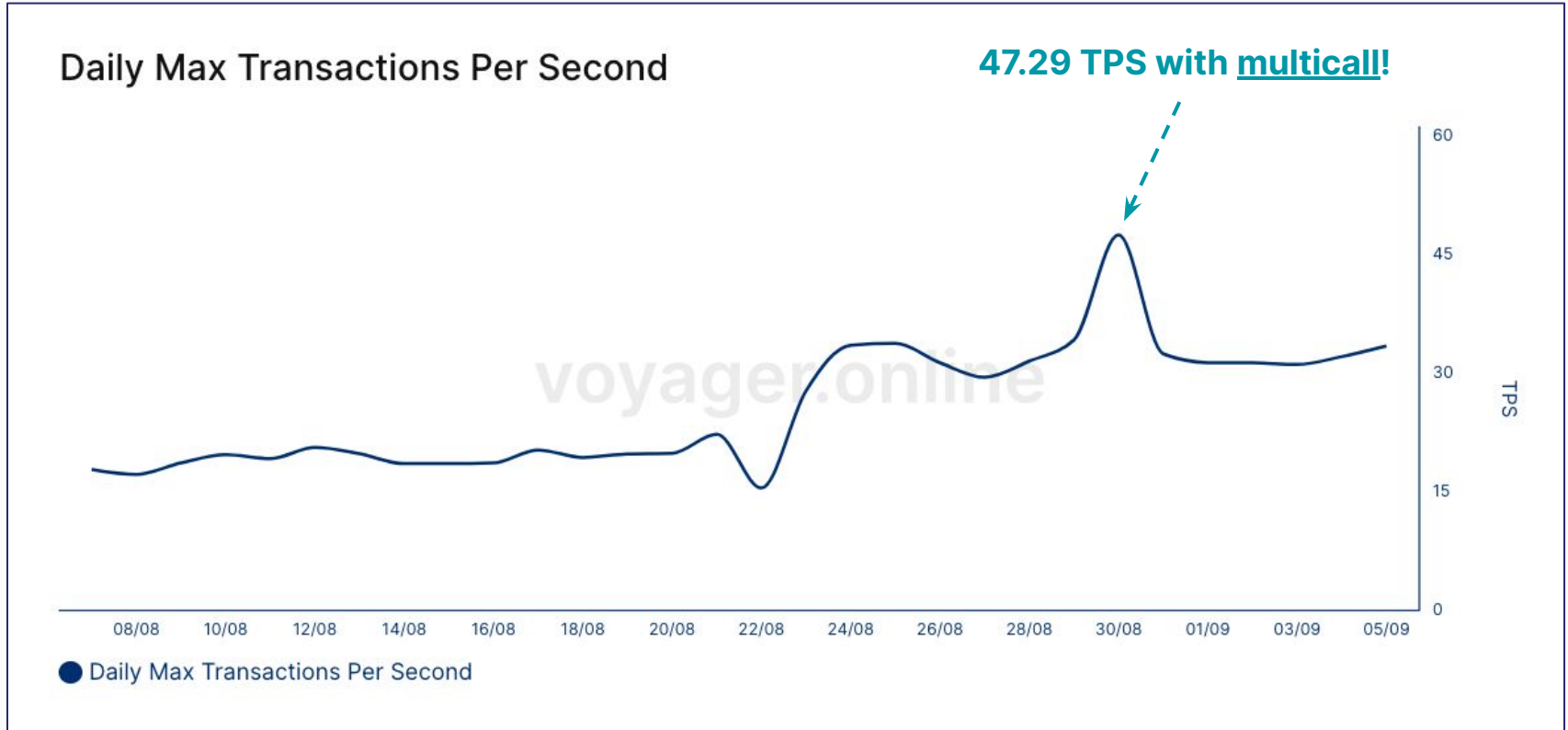
Prevents cheating and malfunction

Keeps a supercomputer **honest**



Mainnet Performance

Stats from [Voyager](#) block explorer



Summary

Why Starknet?

Optimized for ZK tech

↑ computing power ↓ gas fees

Secured by STARKs + Ethereum

Powerful programming language

Battled tested tech stack (2y | \$1T)

Withdraw assets to L1 in ~10h

No trusted setup





Voting



Virtual Identity



Defi



Gaming

Company



STARKWARE



STARKEx

*L2 Validium
(permissioned)*



STARKNET

*L2 Rollup
(permissionless)*



**STARKNET
Foundation**

*Non-profit
organization*



Starknet Ecosystem

✦ October 2023

🐦 [Reddio_com](#)



STARKNET Tech Stack

Cairo VMs:

- Rust
- Python
- Go
- Zig
- Typescript

Devtools:

- Starknet-foundry
- Devnet
- Starknet.{js, py, rs, dart, go}
- Starkli



Sequencers



Call for Contributors

The entity in charge of:

- Aggregating transactions
- Processing transactions
- Producing blocks

Similar to validators in Ethereum

Currently there's only one

High liveness requirements



Provers

The entity in charge of:

- Receiving blocks
- Processing them
- Generating a proof for their correct processing
- Sending it to Ethereum

High machine requirements, but

- Can be split into smaller proofs
- Can be done asynchronously



Nodes



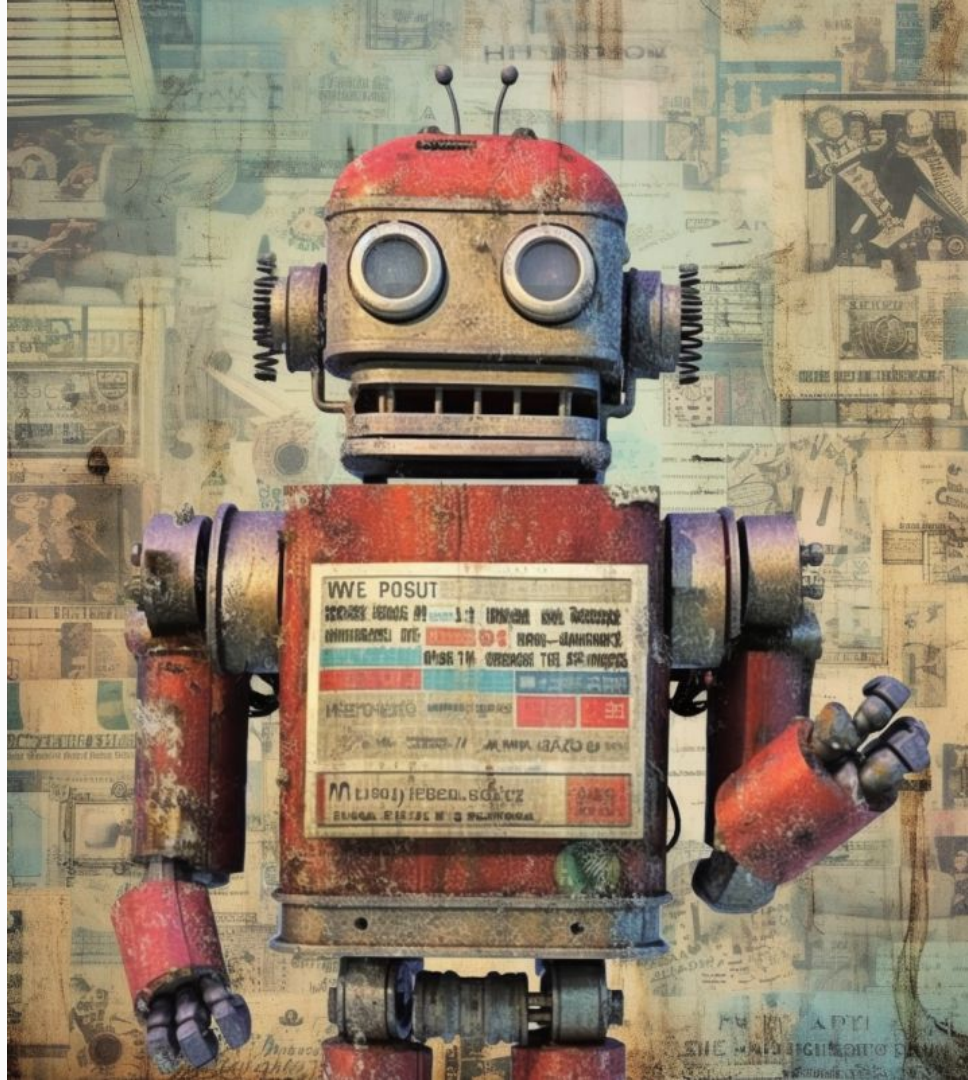
Entities that keep track of the latest state of Starknet

They can do so by:

- Replaying transactions
- Relying on L2 consensus
- Checking proof validations on L1

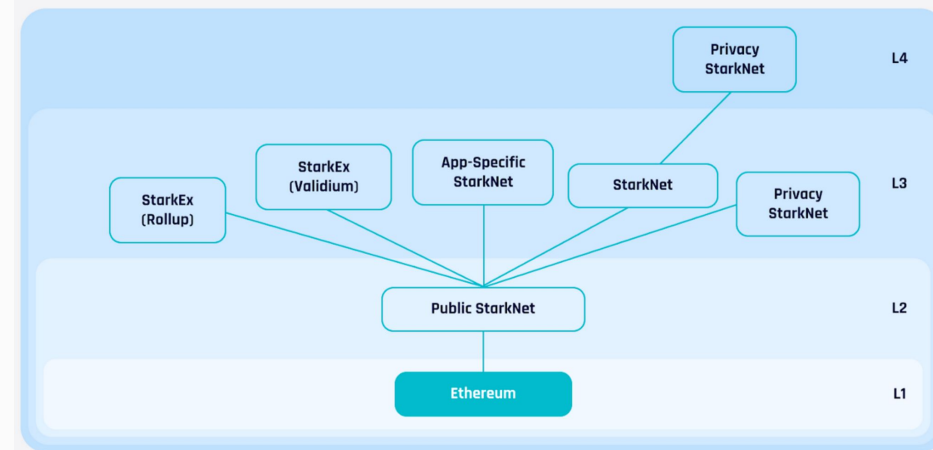
Each setup has pros and cons

- Hardware requirements
- Trust assumptions
- Latency



L2 and L3

- Recursive proofs open up surprising and novel design options
- Introducing L3, the application-specific layer, built recursively over L2
- L3 serves the bespoke needs of apps, such as hyper-scalability, better control of the tech stack, and privacy
- StarkEx, currently serving customers as an L2 solution, will be ported to L3
- Standalone instances of StarkNet will also be available as L3



L3s - Madara

Madara is an open source project aiming to create an easy to use Starknet compatible sequencer

It will allow anyone to spin up their own Starknet network

These will be proven on L1, or L2, or elsewhere



L2s/L3s - Kakarot

Kakarot is an open source ZKevm built in Cairo

It is a set of Cairo smart contracts that operate by interpreting Solidity smart contracts

It will operate as an L3 on top of Starknet



PERMISSIONLESS
v1.54

DYOR!!



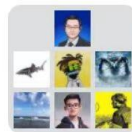


非常感谢!

Reddio

We cannot wait to see what you build!

✦ December 2023



群聊: Reddio用户讨论组



该二维码7天内(12月15日前)有效, 重新进入将更新

Deep Dive

Cairo 0

Released in 2020

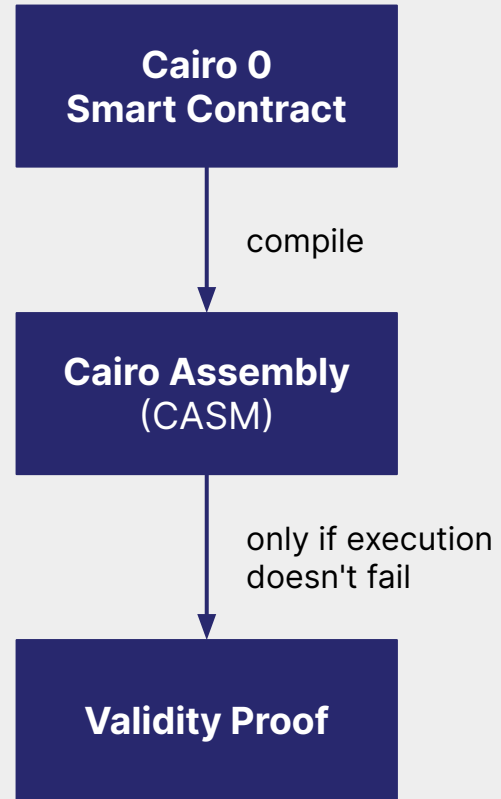
Low level language

Steep learning curve

Failed txs not added to blocks

Sequencer not compensated

DoS vector



Cairo

High level language

Compiles to Sierra

Safe Intermediate Representation

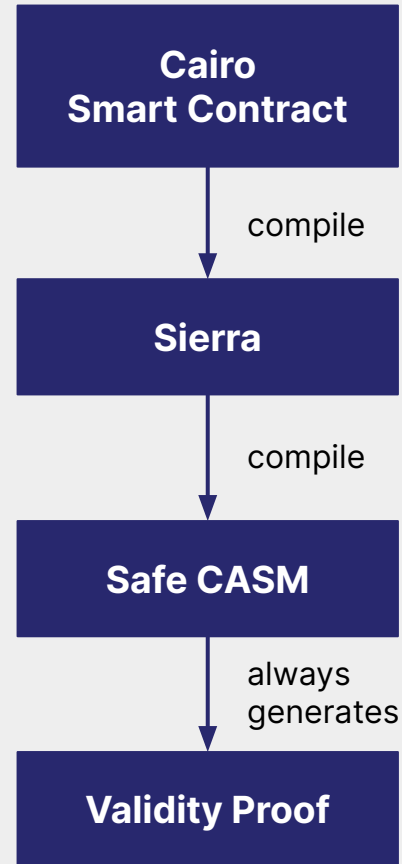
Decoupling of Cairo to CairoVM

Generates “Safe CASM”

Allows for failed txs to be “reverted”

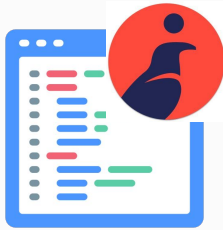
Sequencer ALWAYS compensated

No more DoS vector

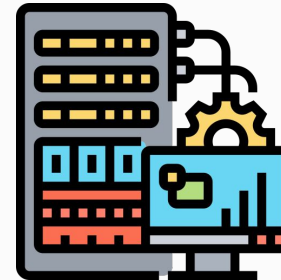


Back to our story...

Your Country



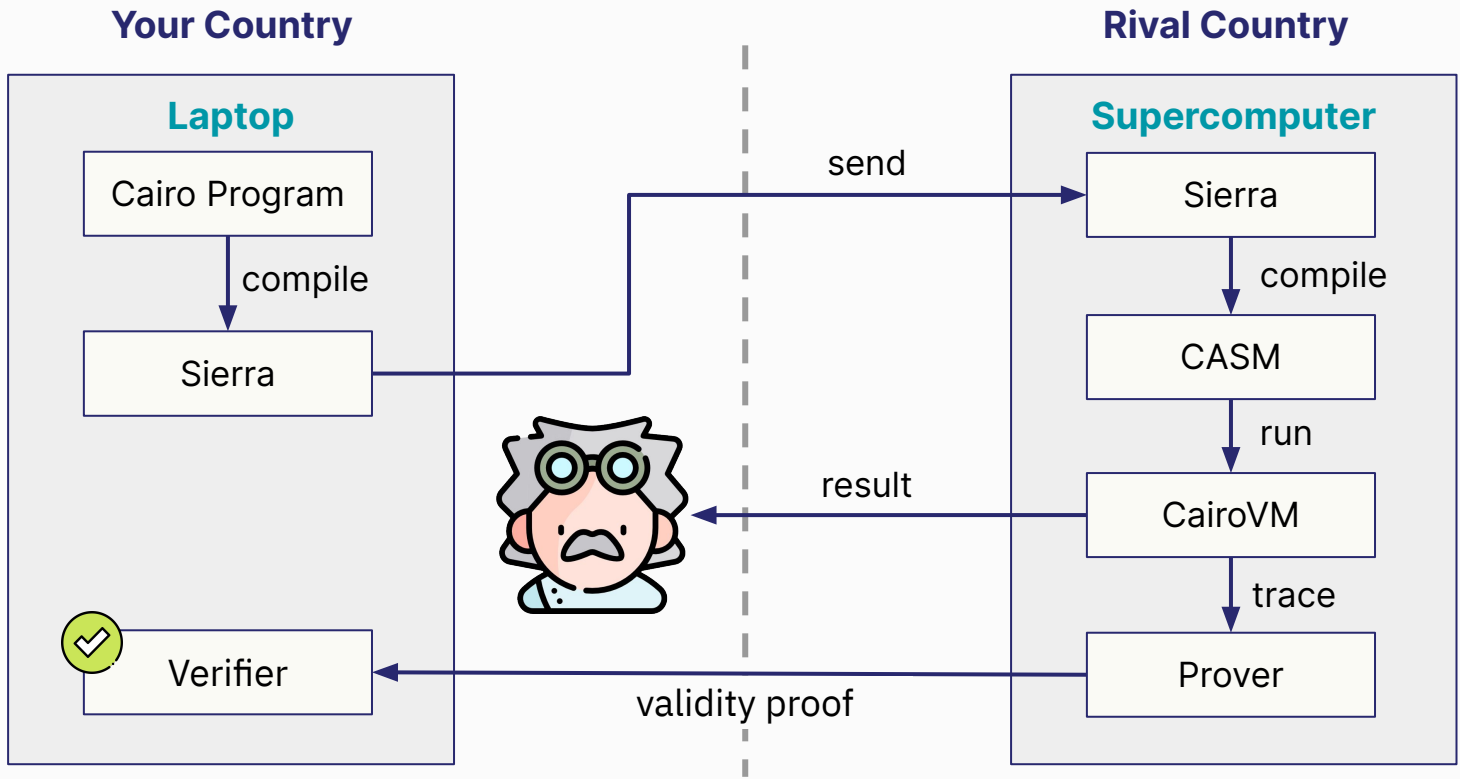
Rival Country



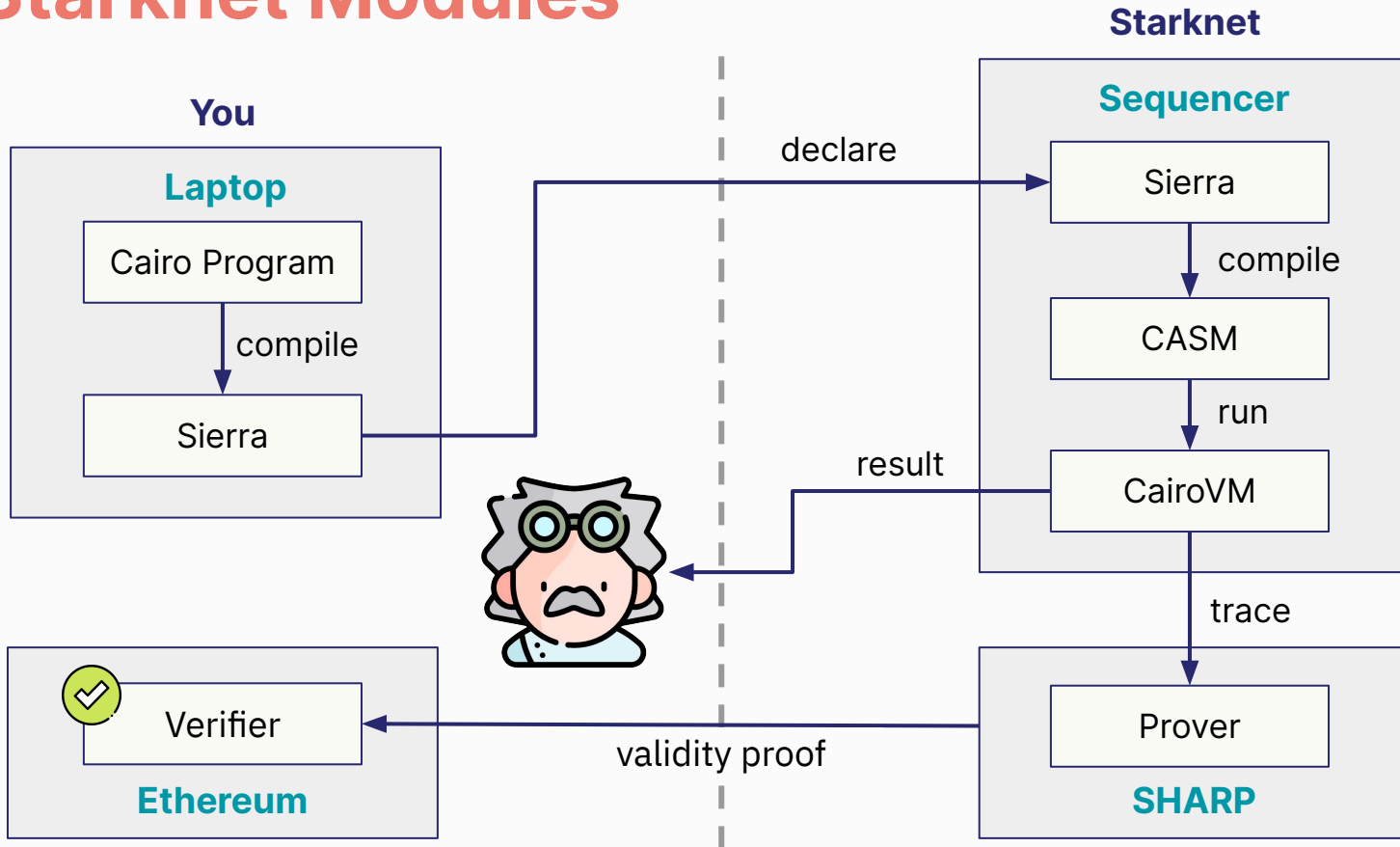
execute

result
+
proof

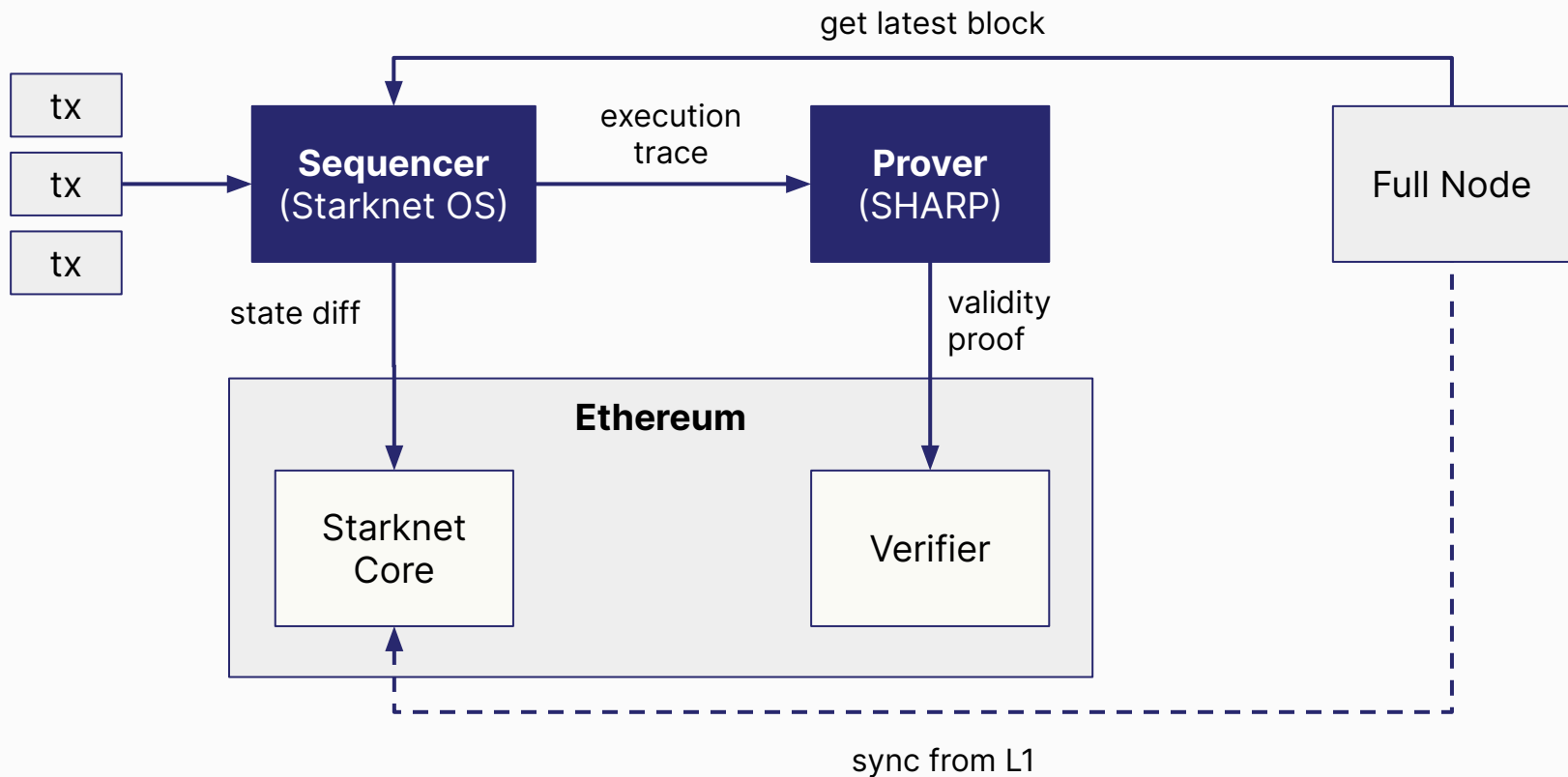
The Internal Modules



Starknet Modules



Zooming Out



Starknet Components

Sequencer: Validates, executes and bundle txs into blocks

SHARP: Creates validity proofs for Starknet and StarkEx

Verifier: L1 smart contract that verifies validity proofs from SHARP

Starknet Core: L1 smart contract that store changes to L2 global state (DA)

Full Node: Provide data to L2 dapps



Declaring vs Deploying

Declaring registers Sierra code on L2

Declared code is aka “contract class”

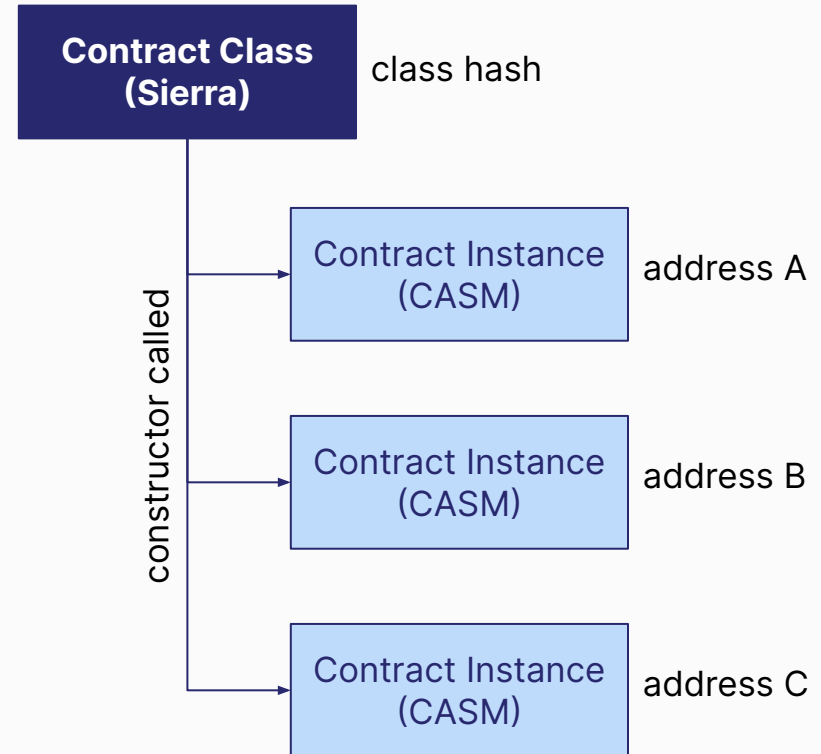
Contract classes don't have internal storage

Used as **libraries** and “**blueprints**”

From blueprint an instance can be **deployed**

Contract instances have internal storage

Every instance has a different address



Transaction Types

deploy_account

- Deploys an account contract

declare

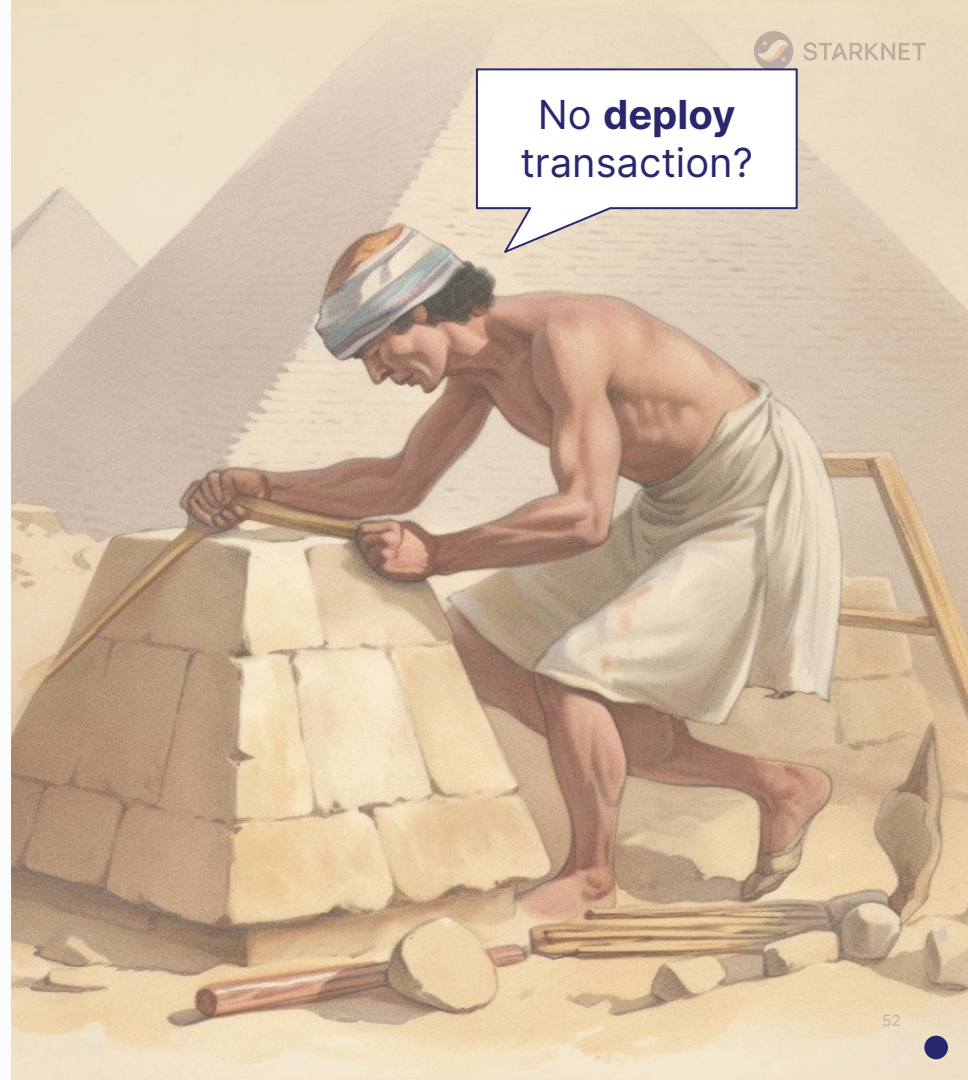
- Registers the Sierra code of a SC

invoke

- Executes “write” functions
- Modifies the global state
- Requires paying gas fees

Calling a **read-only** functions is not a transaction as it doesn't modify the global state (no gas fees)

No **deploy** transaction?



Universal Deployer

Smart contract that deploys other smart contracts

Only one selector: **deployContract**

Selector must be invoked with:

- Blueprint's class hash
- Constructor arguments

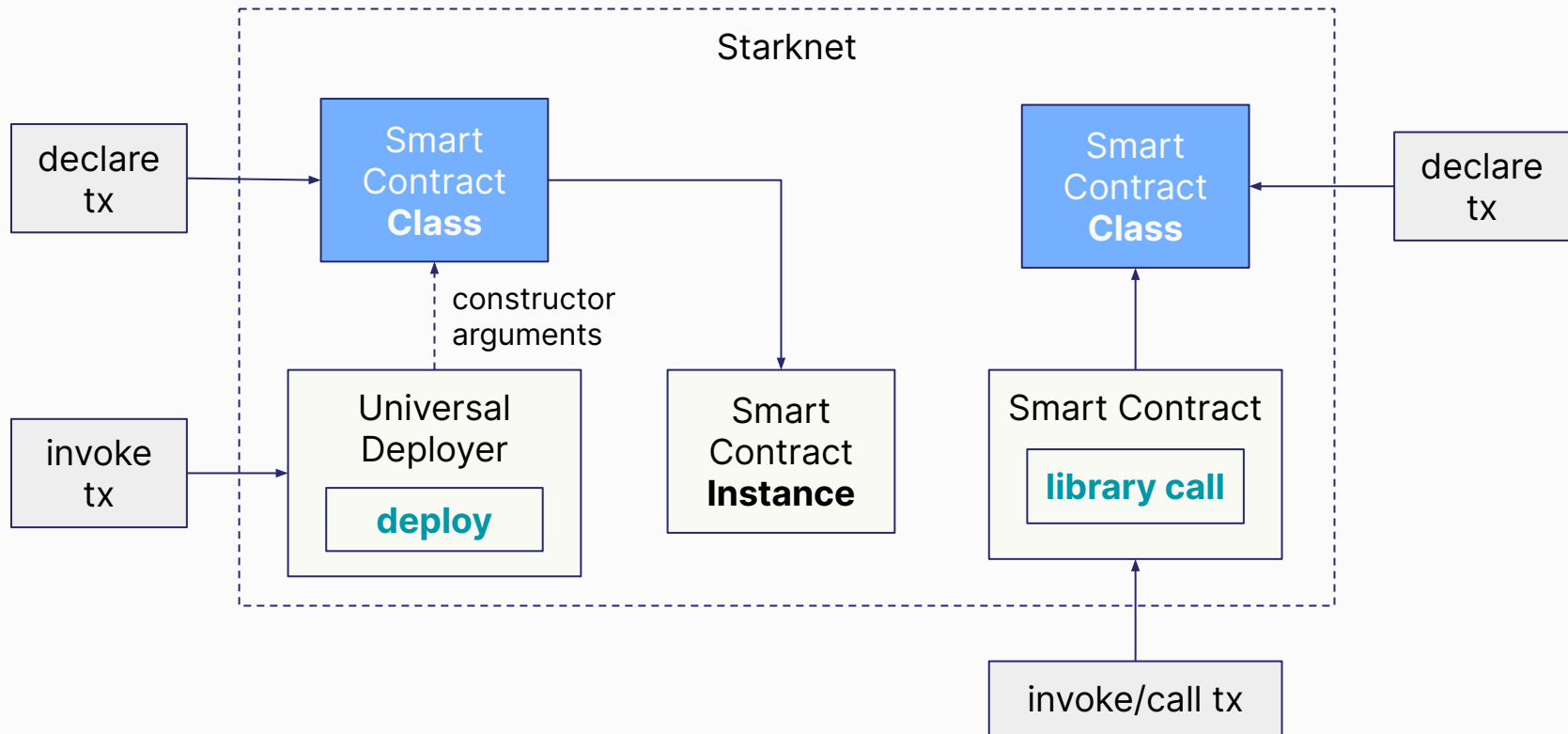
Internally uses the **deploy** syscall

Created by OZ as a public good

Written in Cairo0



Smart Contract Classes vs Instances



Summary

Deep Dive

Cairo compiles to Sierra

Sierra allows the creation of safe CASM

Starknet nodes:

- L2: Sequencer, Prover, Full Nodes
- L1: Verifier, Starknet Core

Declare vs Deploy \Rightarrow Class vs Instance

Tx types: `deploy_account`, `declare` and `invoke` (call is free)

Deploy \Rightarrow Invoke Universal Deployer

