SMART CONTRACTS ON STELLAR

What they are, how they work,
and why they matter

LEIGH MCCULLOCH

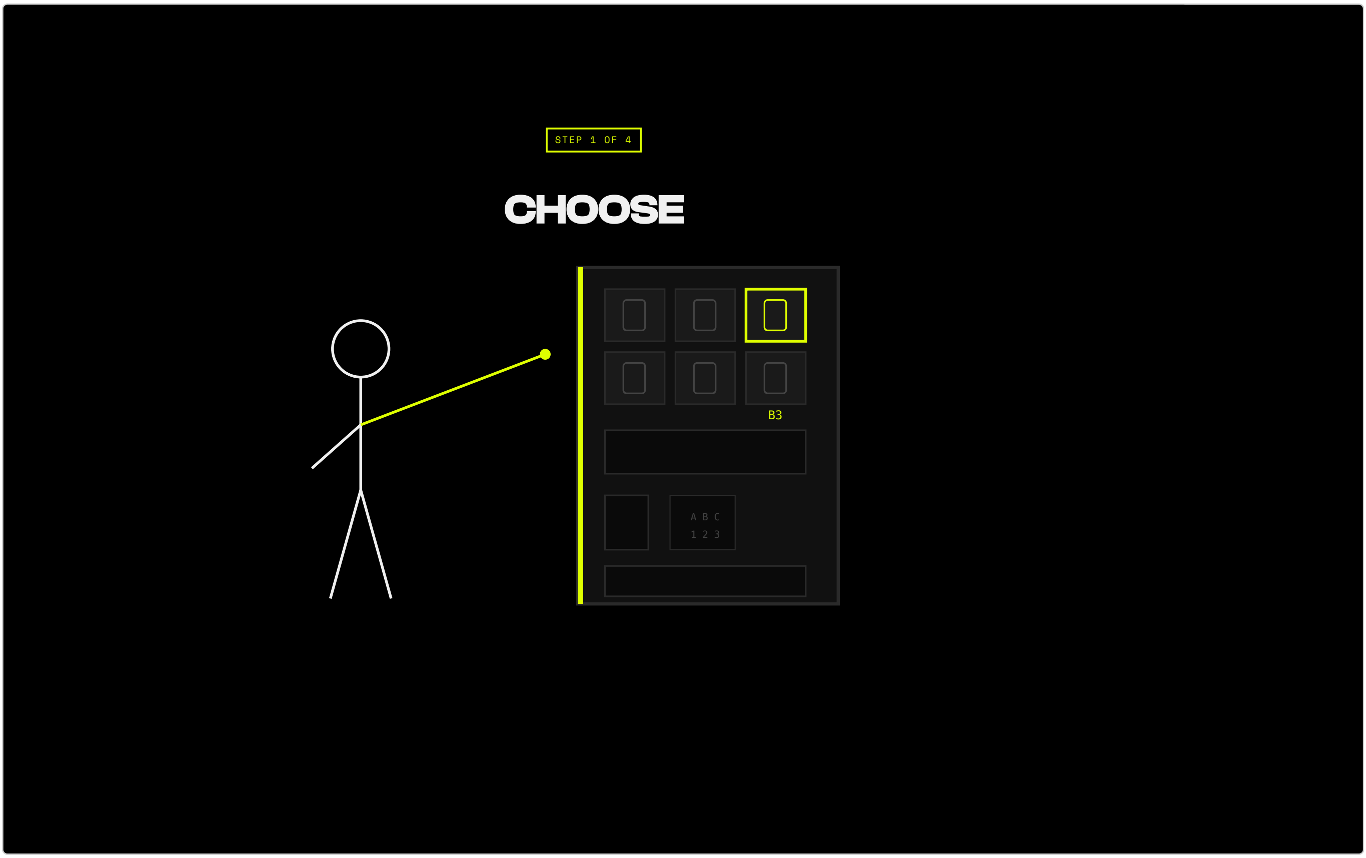
Slide Number: 1 / 37

Speaker Notes: Welcome everyone. Today we're going to talk about smart contracts on Stellar — what they are, how they work, and why they matter. To set expectations this is not a talk targeting developers. We will not be talking about how to build a smart contract or be doing any coding. This is also not a talk about a specific wallet product, so I'm not demoing anything. Rather we'll be discussing the concepts and basics of what a contract is. To help us understand what a smart contract is, we're going to start by taking a look at...

VENDING MACHINES

Slide Number: 2 / 37

Speaker Notes: ...vending machines. Can I get a show of hands, who has used a vending machine before? Is there anyone who hasn't? Okay, so when we use a vending machine, we are swapping our money for something else, like an egg sandwich. To do that, we...

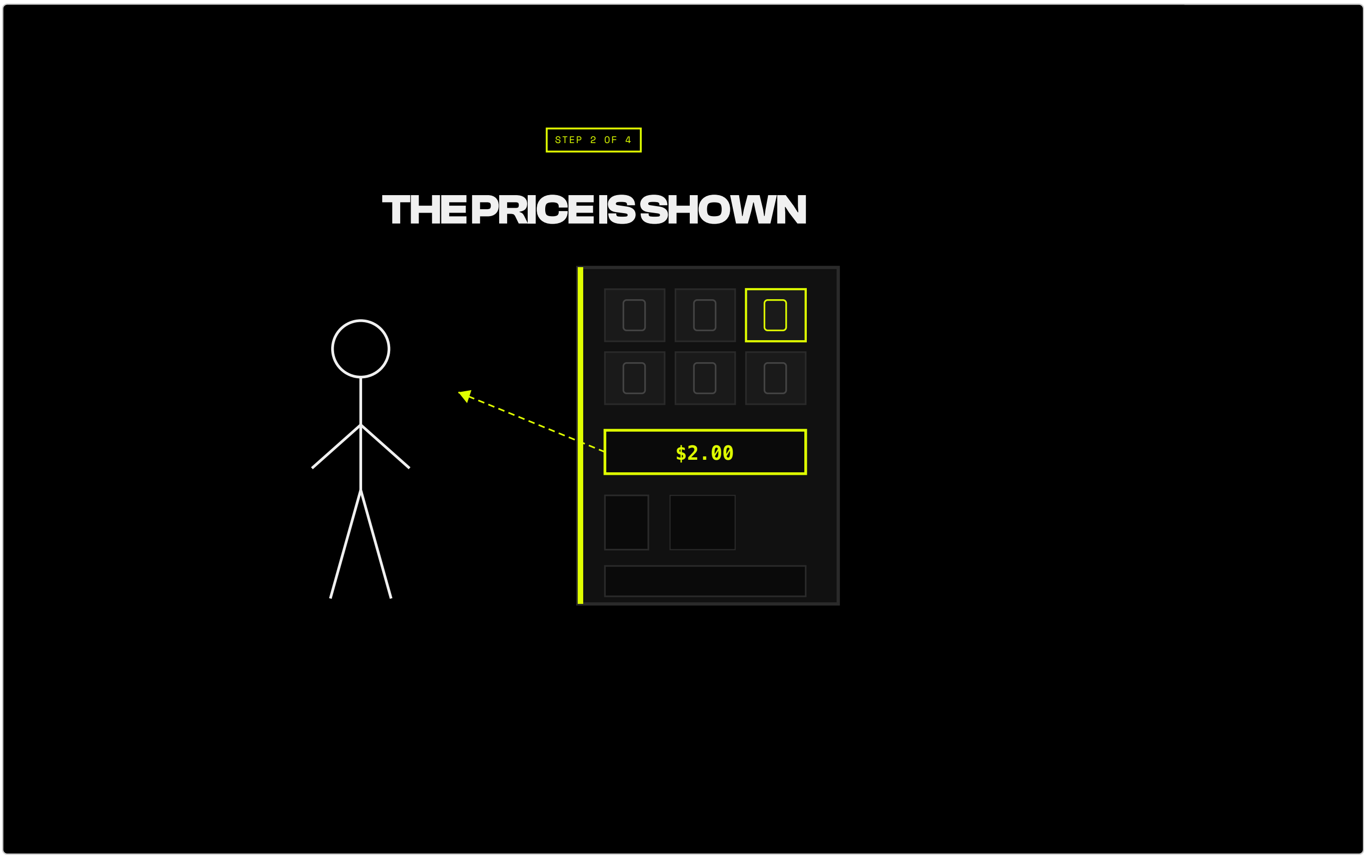


STEP 1 OF 4

CHOOSE

Slide Number: 3 / 37

Speaker Notes: ...walk up to the vending machine and look at what's available. We decide what we want — what product we'll swap our money for — when we've decided, we'll make our selection.

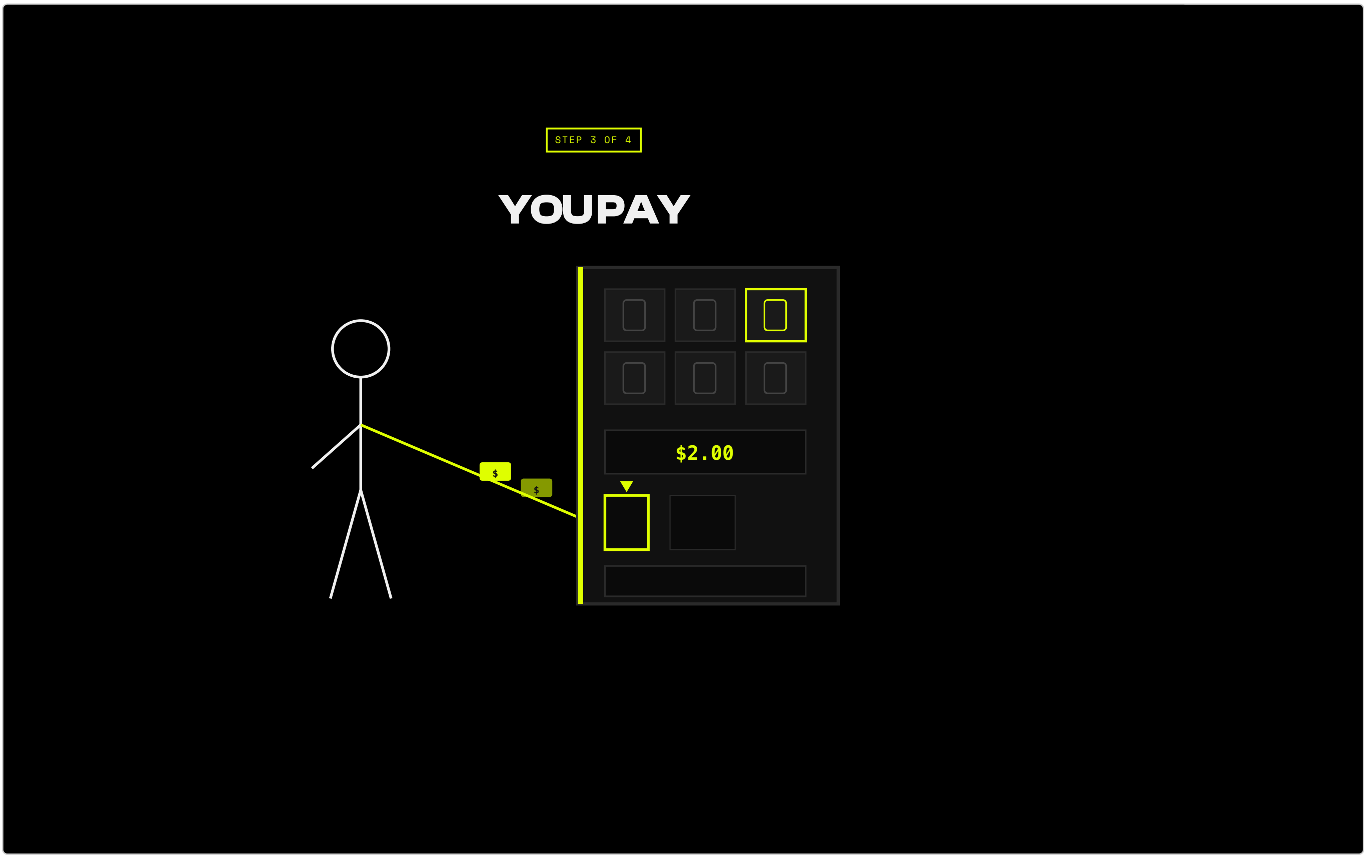


STEP 2 OF 4

THE PRICE IS SHOWN

Slide Number: 4 / 37

Speaker Notes: The machine shows us a price. Two dollars. And, it's well understood that if the machine says two dollars, it will charge us two dollars. There are rules, a mix of social conventions, laws, and our own experience, that tell us we can expect the machine to give us the product we paid for. We don't truly know that it will, but we assume it will, and we go on.

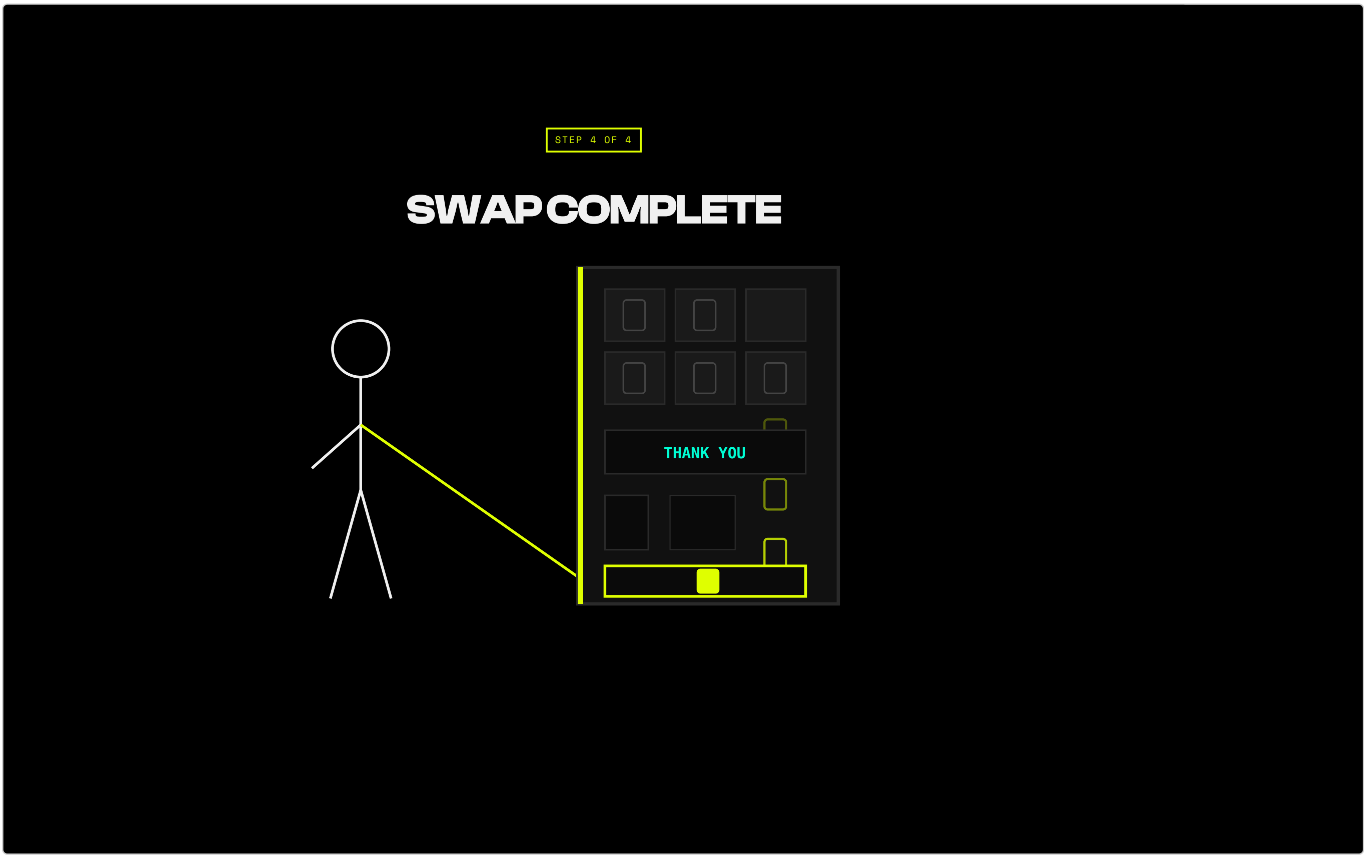


STEP 3 OF 4

YOUPAY

Slide Number: 5 / 37

Speaker Notes: We pay. We tap a card, or insert the cash. And we wait.

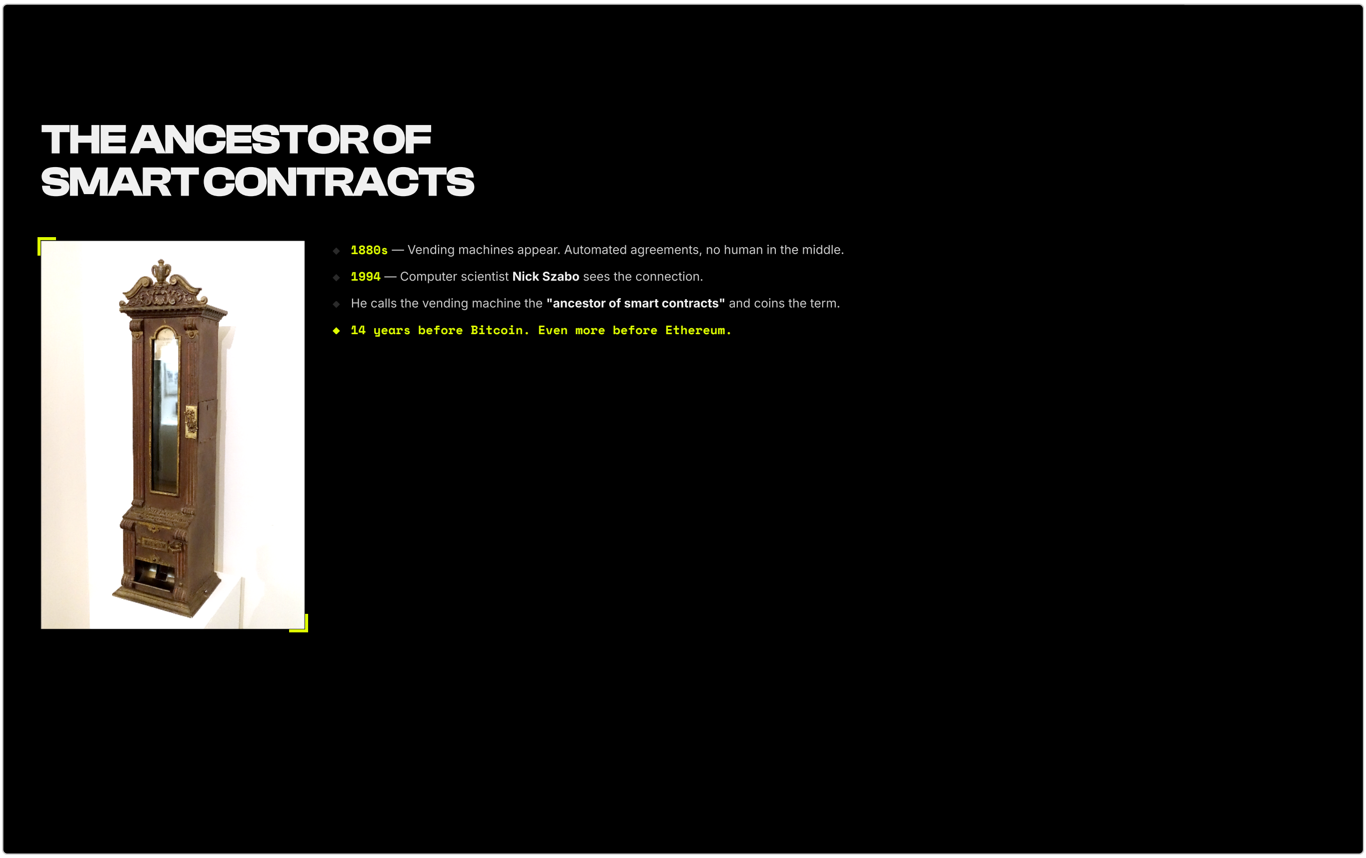


STEP 4 OF 4

SWAP COMPLETE

Slide Number: 6 / 37

Speaker Notes: The machine starts to execute our request. It whirrs and moves, and we watch as it delivers the product to us. Using the machine, we swapped something of value, money, for something else of value, an egg sandwich. No cashier, no negotiation — just us with our expectations, and the machine with its rules, executing them. So what does this have to do with smart contracts? Well this is very similar to how smart contracts work.

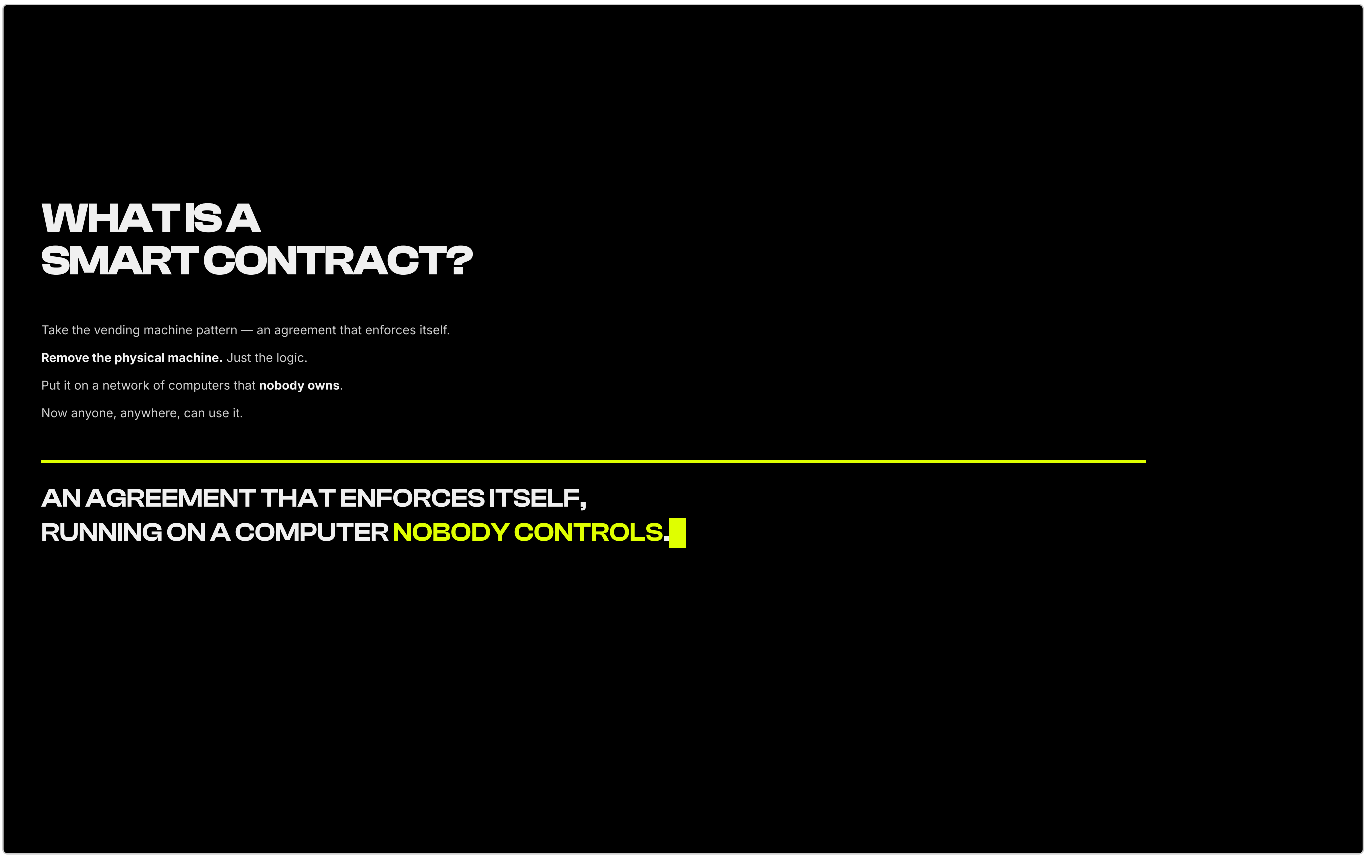


THE ANCESTOR OF SMART CONTRACTS

- **1880s** — Vending machines appear. Automated agreements, no human in the middle.
- **1994** — Computer scientist **Nick Szabo** sees the connection.
- He calls the vending machine the **"ancestor of smart contracts"** and coins the term.
- **14 years before Bitcoin. Even more before Ethereum.**

Slide Number: 7 / 37

Speaker Notes: In 1994, a computer scientist named Nick Szabo coined the term smart contracts, and when describing them used the vending machine as an example of a primitive one. He imagined that smart contracts would come about through embedding self-enforcing logic into technology, and that it would eliminate the need for trust between parties. He coined the term more than a decade before blockchains like Bitcoin, Stellar, or Ethereum existed. He wasn't talking about blockchains specifically, but today blockchains have really owned the term. When you hear people say smart contracts today, they mean smart contracts on blockchains.



WHAT IS A SMART CONTRACT?

Take the vending machine pattern — an agreement that enforces itself.

Remove the physical machine. Just the logic.

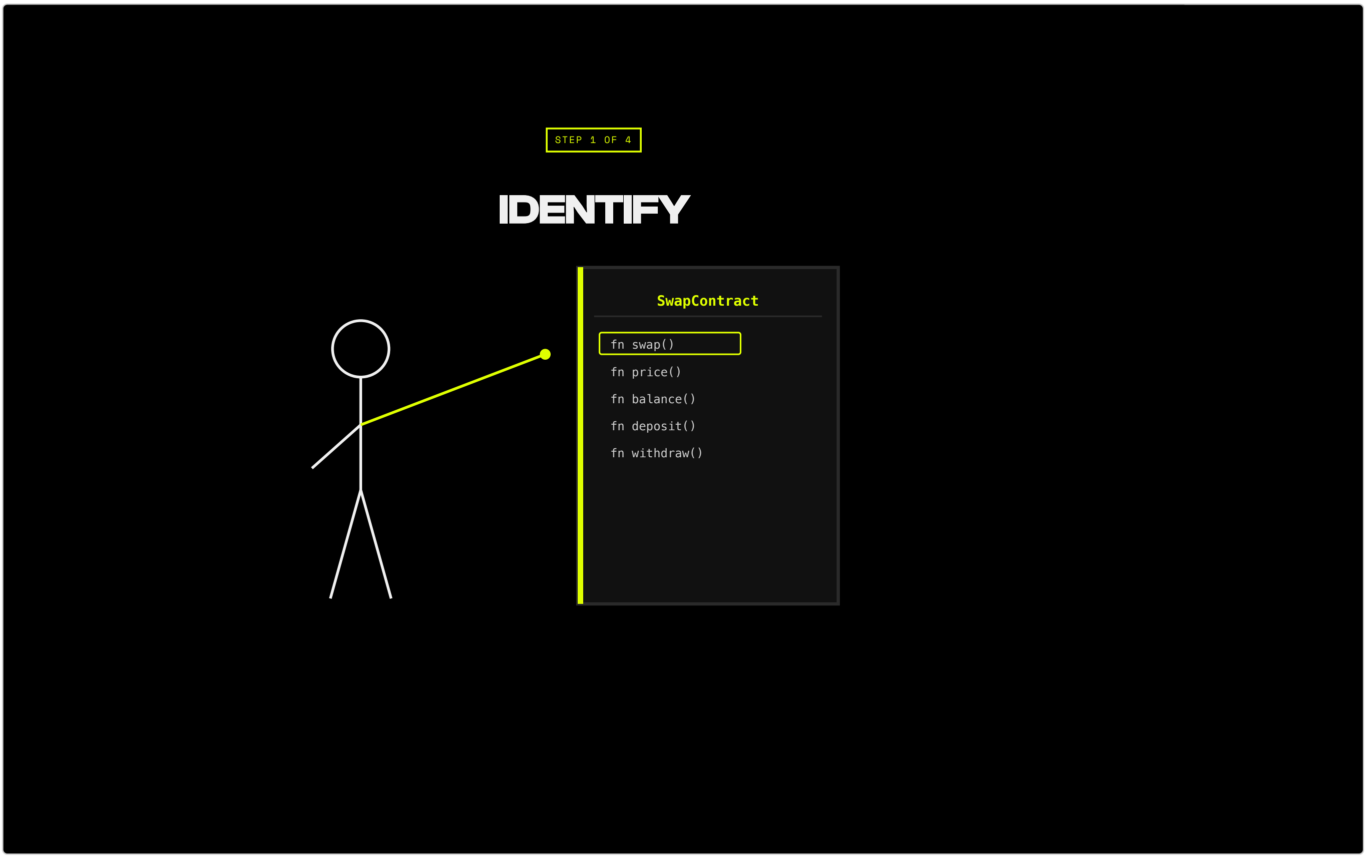
Put it on a network of computers that **nobody owns**.

Now anyone, anywhere, can use it.

AN AGREEMENT THAT ENFORCES ITSELF, RUNNING ON A COMPUTER NOBODY CONTROLS.

Slide Number: 8 / 37

Speaker Notes: So what is a smart contract? It's an agreement that enforces itself. Take the vending machine, remove the physical hardware, keep only the logic, and put it on a decentralized network. Now anyone, anywhere can use it. That's a smart contract. And the experience of using one follows a similar flow.



STEP 1 OF 4

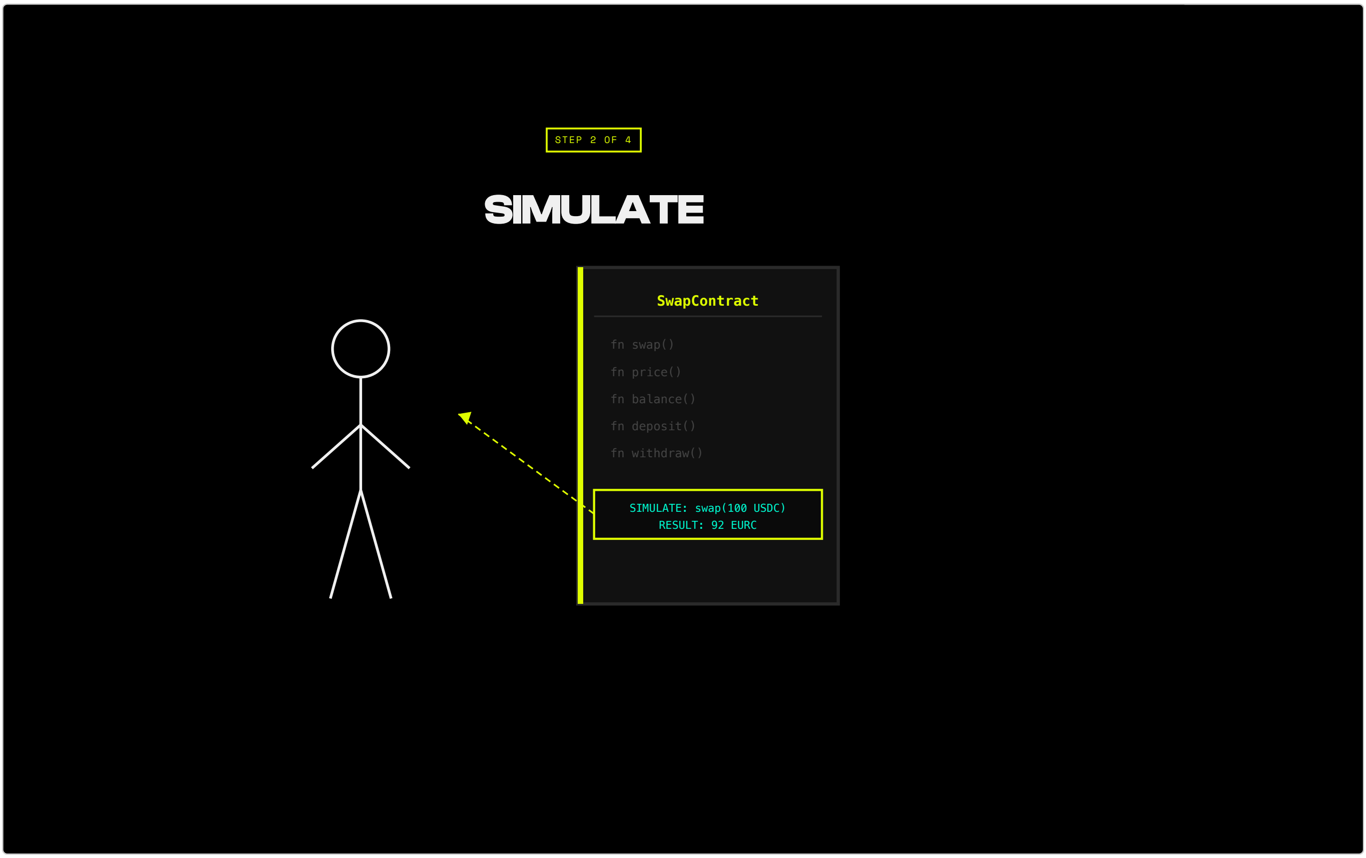
IDENTIFY

SwapContract

```
fn swap()
fn price()
fn balance()
fn deposit()
fn withdraw()
```

Slide Number: 9 / 37

Speaker Notes: Just like with a vending machine, the first step is choosing what we want. We find a contract — a swap, a pool, a token — and look at what it can do. In practice this usually happens through a web app, but here we're looking one layer beneath.



STEP 2 OF 4

SIMULATE

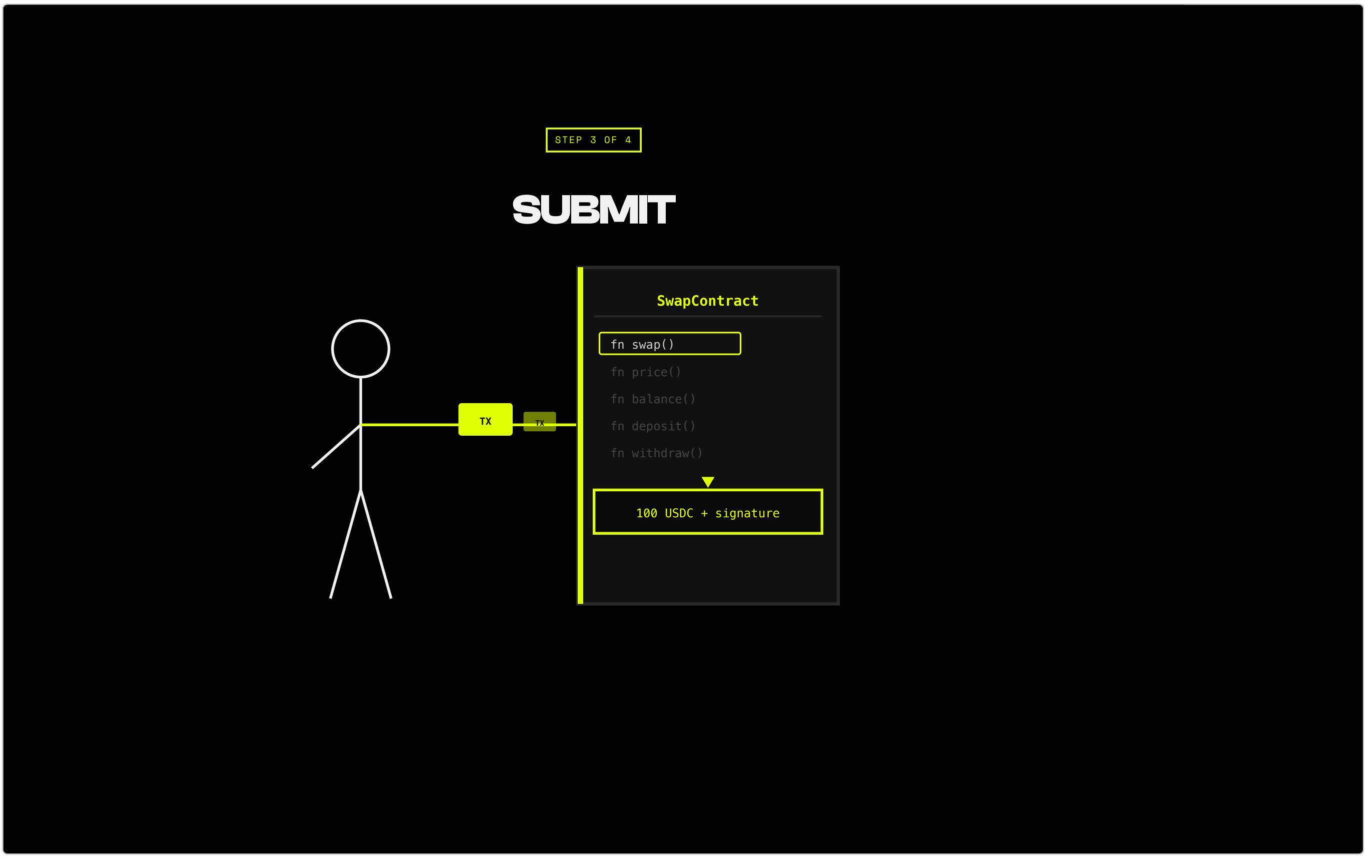
SwapContract

```
fn swap()
fn price()
fn balance()
fn deposit()
fn withdraw()
```

```
SIMULATE: swap(100 USDC)
RESULT: 92 EURC
```

Slide Number: 10 / 37

Speaker Notes: Next, we simulate. We ask the contract: if I swap 100 USDC, what do I get back? It tells us the outcome before we spend anything. Much like reading the price on the display of a vending machine.



STEP 3 OF 4

SUBMIT

TX

TX

SwapContract

fn swap()

fn price()

fn balance()

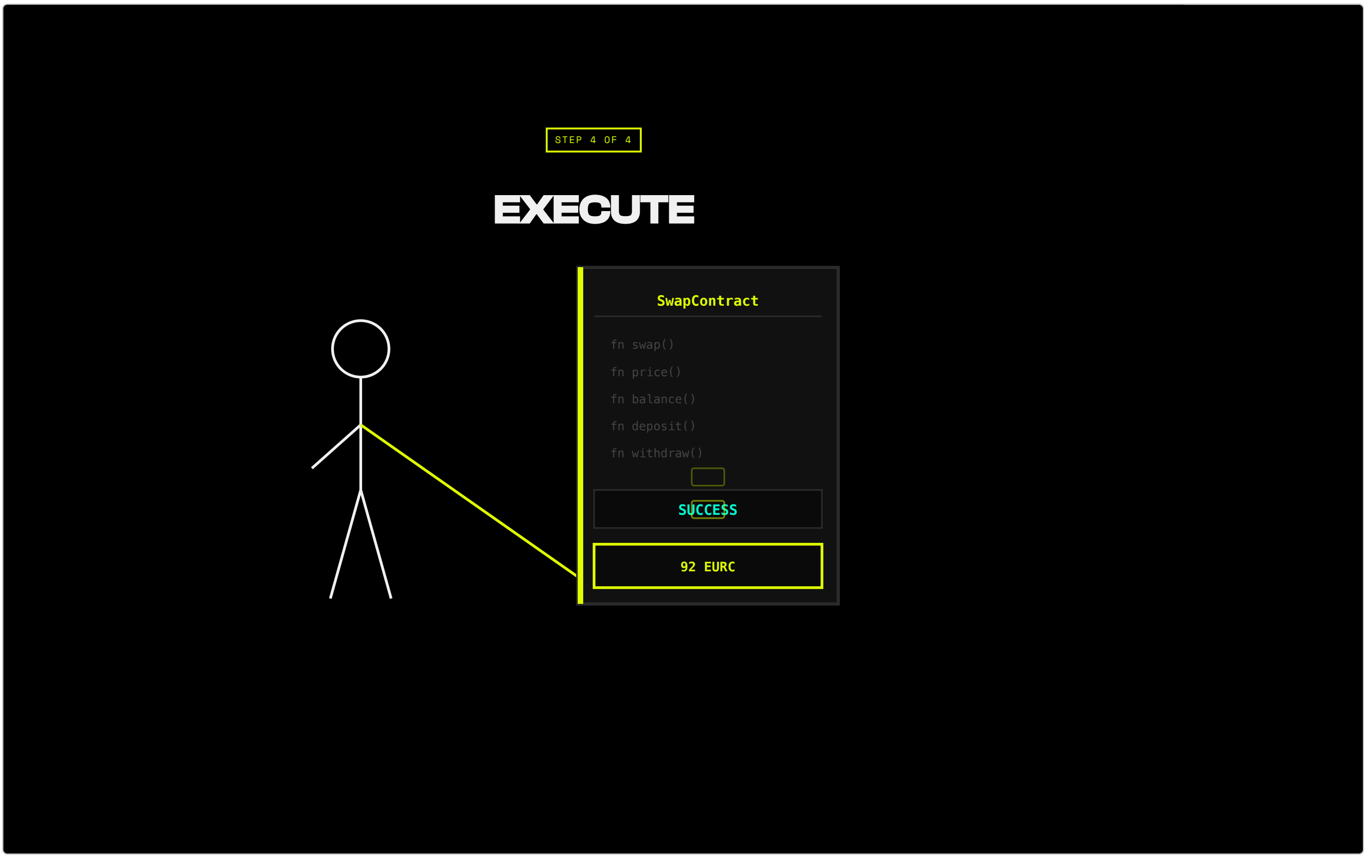
fn deposit()

fn withdraw()

100 USDC + signature

Slide Number: 11 / 37

Speaker Notes: Then we authorise and submit the transaction using our wallet. This sends it to the network. This is the moment we commit — our USDC is on the line, in much the same way our cash was when we paid the vending machine. And then we wait for the network to confirm the transaction.



STEP 4 OF 4

EXECUTE

SwapContract

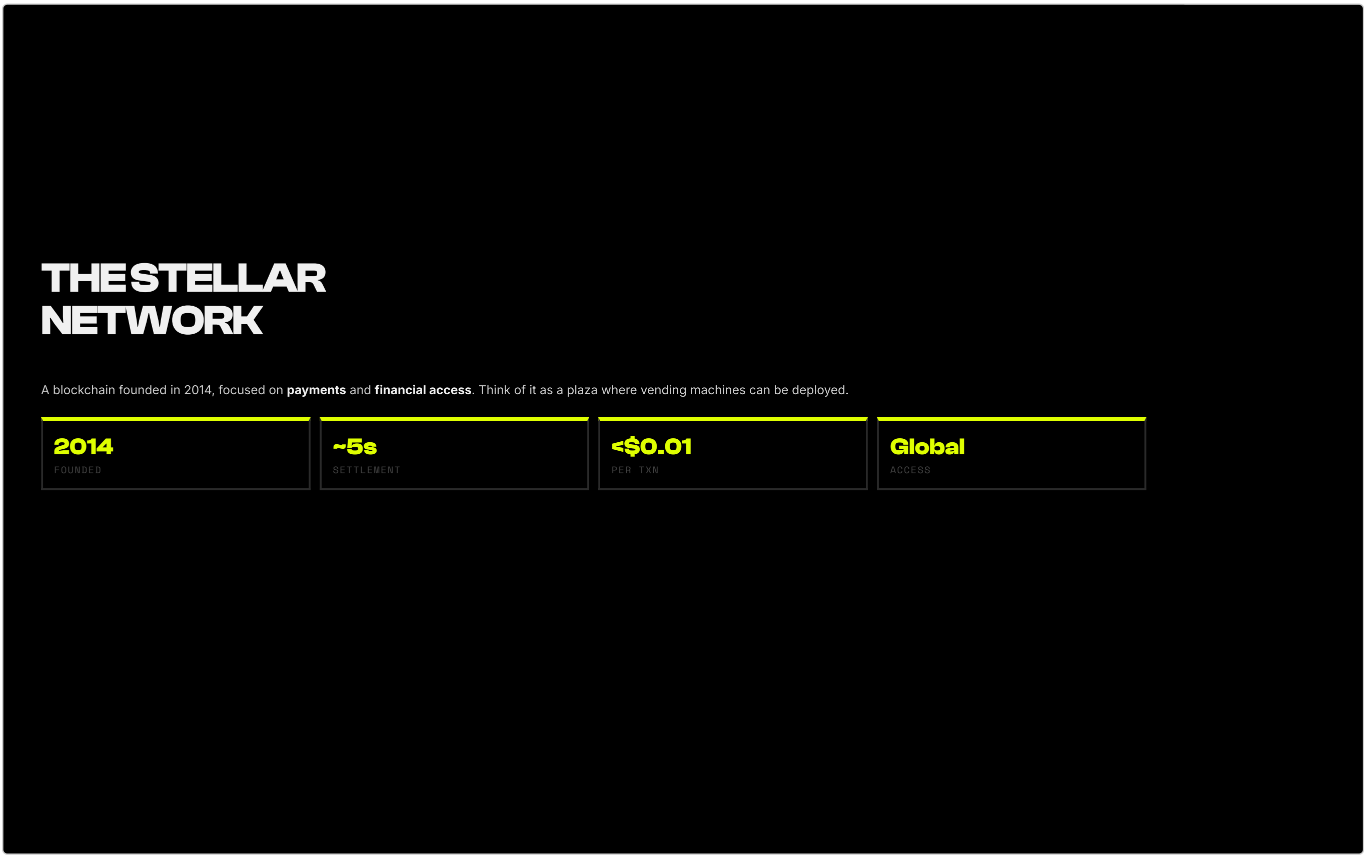
```
fn swap()
fn price()
fn balance()
fn deposit()
fn withdraw()
```

SUCCESS

92 EURC

Slide Number: 12 / 37

Speaker Notes: Eventually, the contract executes. The network runs the code, swaps the USDC for the egg sandwich token, and delivers the result. No person decided whether to approve it. The code ran, the rules were followed, and the outcome is final.



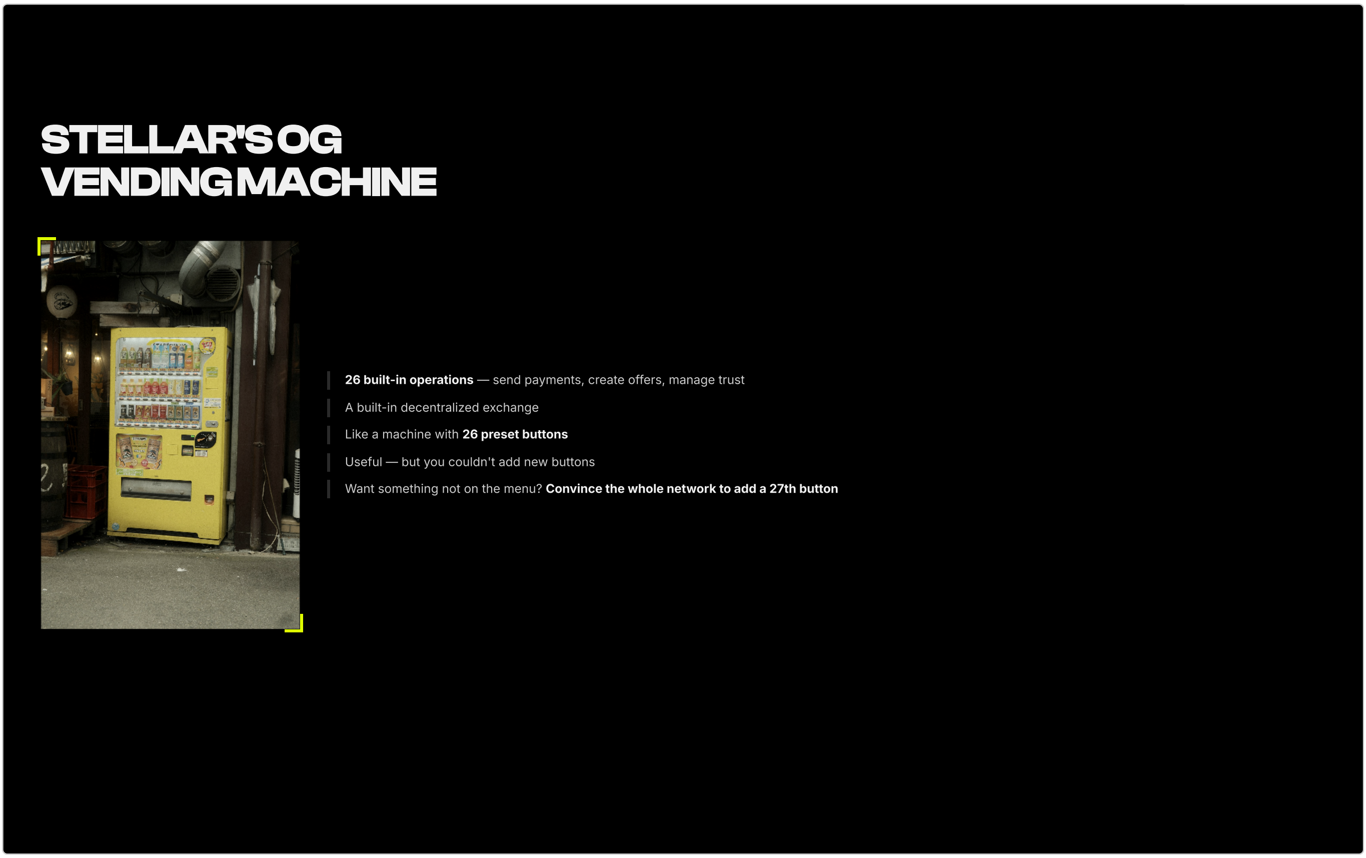
THE STELLAR NETWORK

A blockchain founded in 2014, focused on **payments** and **financial access**. Think of it as a plaza where vending machines can be deployed.

2014 FOUNDED	**~5s** SETTLEMENT	**<$0.01** PER TXN	**Global** ACCESS

Slide Number: 13 / 37

Speaker Notes: So where do these contracts run? On Stellar. Founded in 2014, Stellar was built for payments and financial access — fast settlement, fraction-of-a-penny fees. We all work at the Stellar Development Foundation, so I hope none of this is a surprise, that Stellar is a blockchain, and that's where smart contracts execute. Continuing our analogy of vending machines in the physical world, we can think of Stellar as a plaza where vending machines get installed, and people can go to use them.

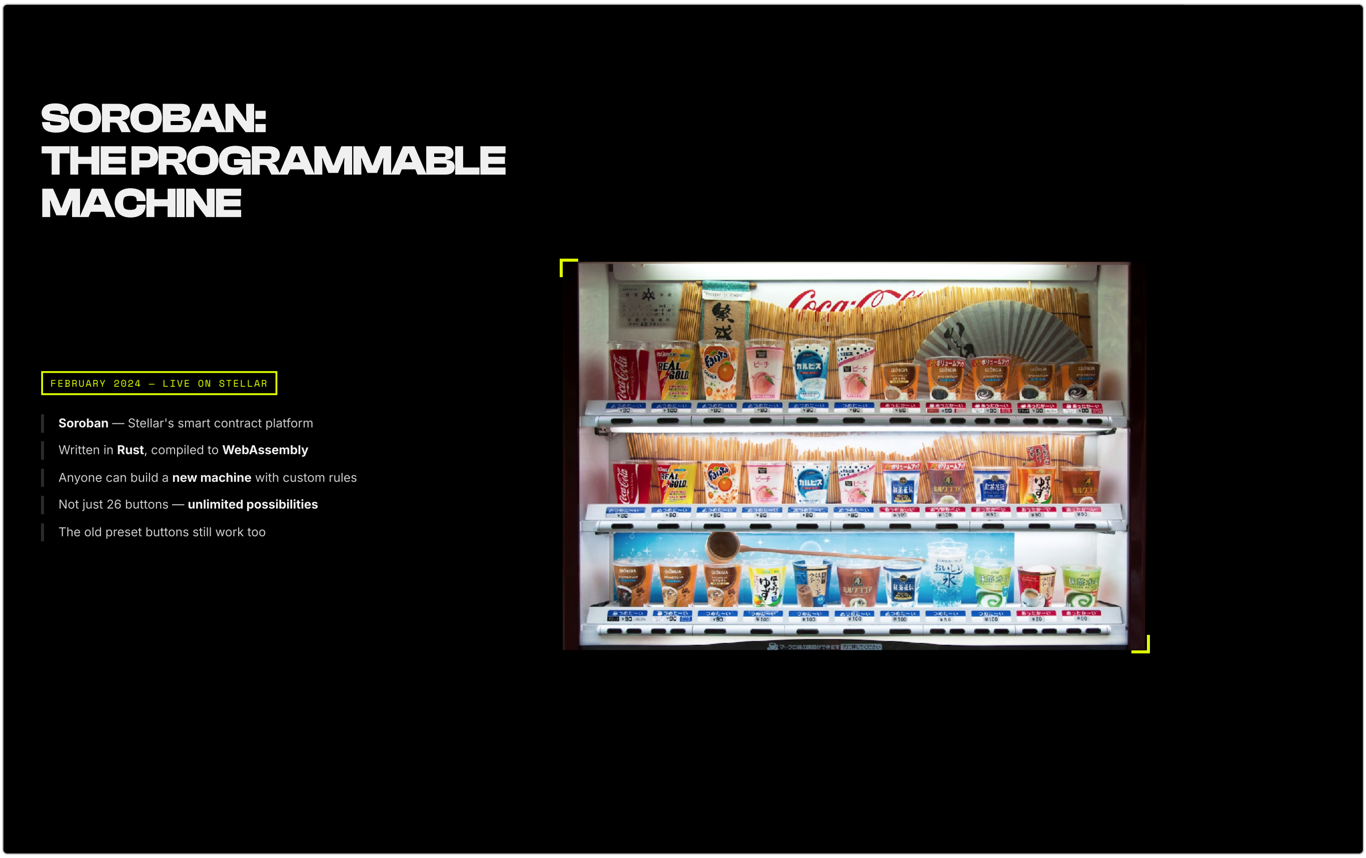


STELLAR'S OG VENDING MACHINE

- **26 built-in operations** — send payments, create offers, manage trust
- A built-in decentralized exchange
- Like a machine with **26 preset buttons**
- Useful — but you couldn't add new buttons
- Want something not on the menu? **Convince the whole network to add a 27th button**

Slide Number: 14 / 37

Speaker Notes: For its first decade, Stellar had 26 built-in operations spanning payments, a decentralised exchange, and account management. You could think of Stellar having a few vending machines and across them were 26 preset buttons that did different things. It was very useful, but if you wanted something not on the menu, you had to convince the Stellar Development Foundation, and the network to add a new button. That technically fits Szabo's definition of a smart contract, but in blockchain terms, smart contracts mean programs anyone can deploy.



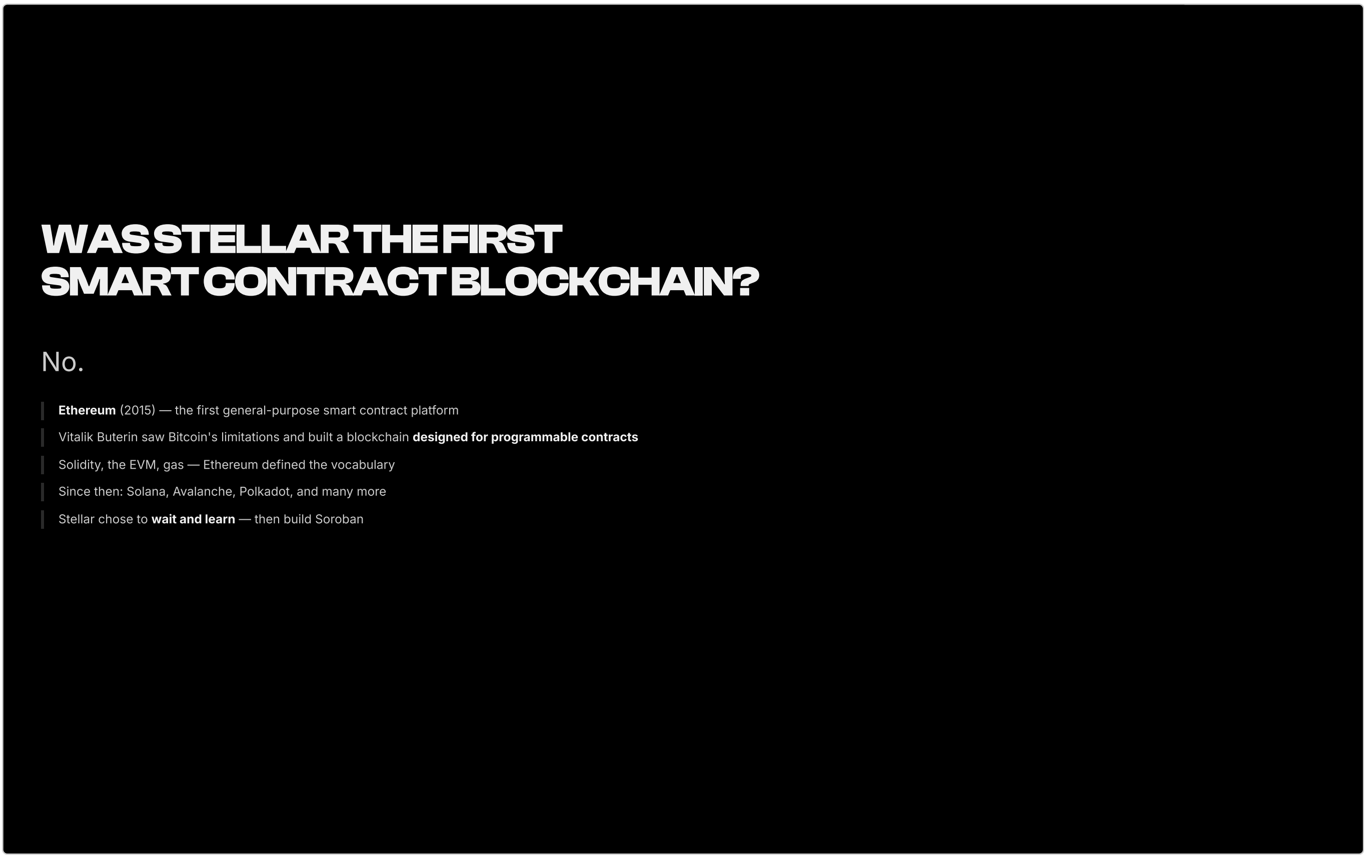
SOROBAN: THE PROGRAMMABLE MACHINE

FEBRUARY 2024 — LIVE ON STELLAR

- **Soroban** — Stellar's smart contract platform
- Written in **Rust**, compiled to **WebAssembly**
- Anyone can build a **new machine** with custom rules
- Not just 26 buttons — **unlimited possibilities**
- The old preset buttons still work too

Slide Number: 15 / 37

Speaker Notes: So in 2024, Stellar launched Soroban — the ability for anyone to deploy their own smart contracts. Contracts are written in Rust and compiled to WebAssembly. Instead of 26 fixed buttons, anyone can now build a vending machine that sits in the Stellar plaza with whatever buttons they want.



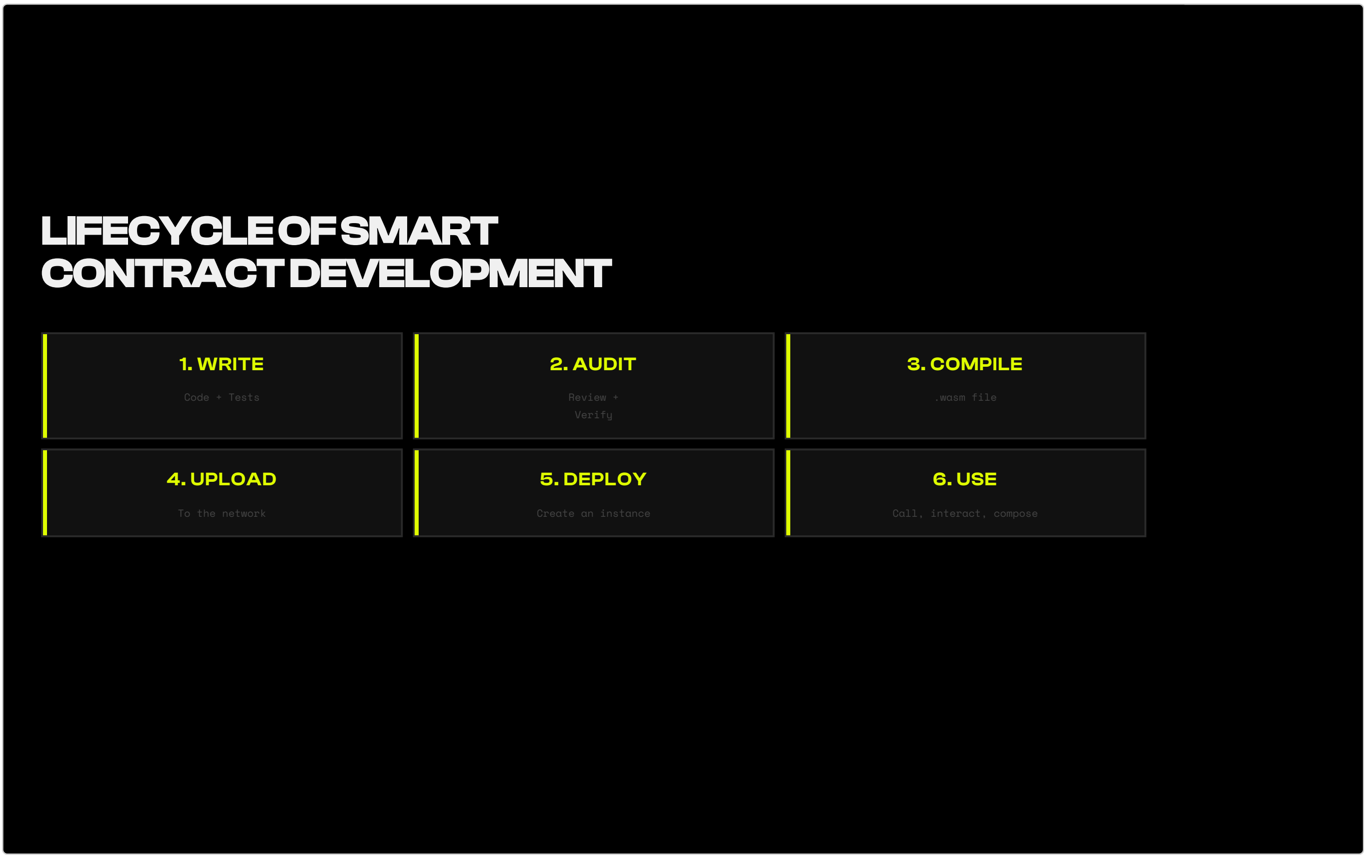
WAS STELLAR THE FIRST SMART CONTRACT BLOCKCHAIN?

No.

- **Ethereum** (2015) — the first general-purpose smart contract platform
- Vitalik Buterin saw Bitcoin's limitations and built a blockchain **designed for programmable contracts**
- Solidity, the EVM, gas — Ethereum defined the vocabulary
- Since then: Solana, Avalanche, Polkadot, and many more
- Stellar chose to **wait and learn** — then build Soroban

Slide Number: 16 / 37

Speaker Notes: Was Stellar first? No. Ethereum launched in 2015, the year after Stellar first launched and 9 years before Stellar launched smart contracts. Ethereum really defined what a smart contract on a blockchain would be. It proved that programmable contracts on a blockchain could work and now most blockchains offer a smart contract experience that is similar.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE

Code + Tests

2. AUDIT

Review +
Verify

3. COMPILE

.wasm file

4. UPLOAD

To the network

5. DEPLOY

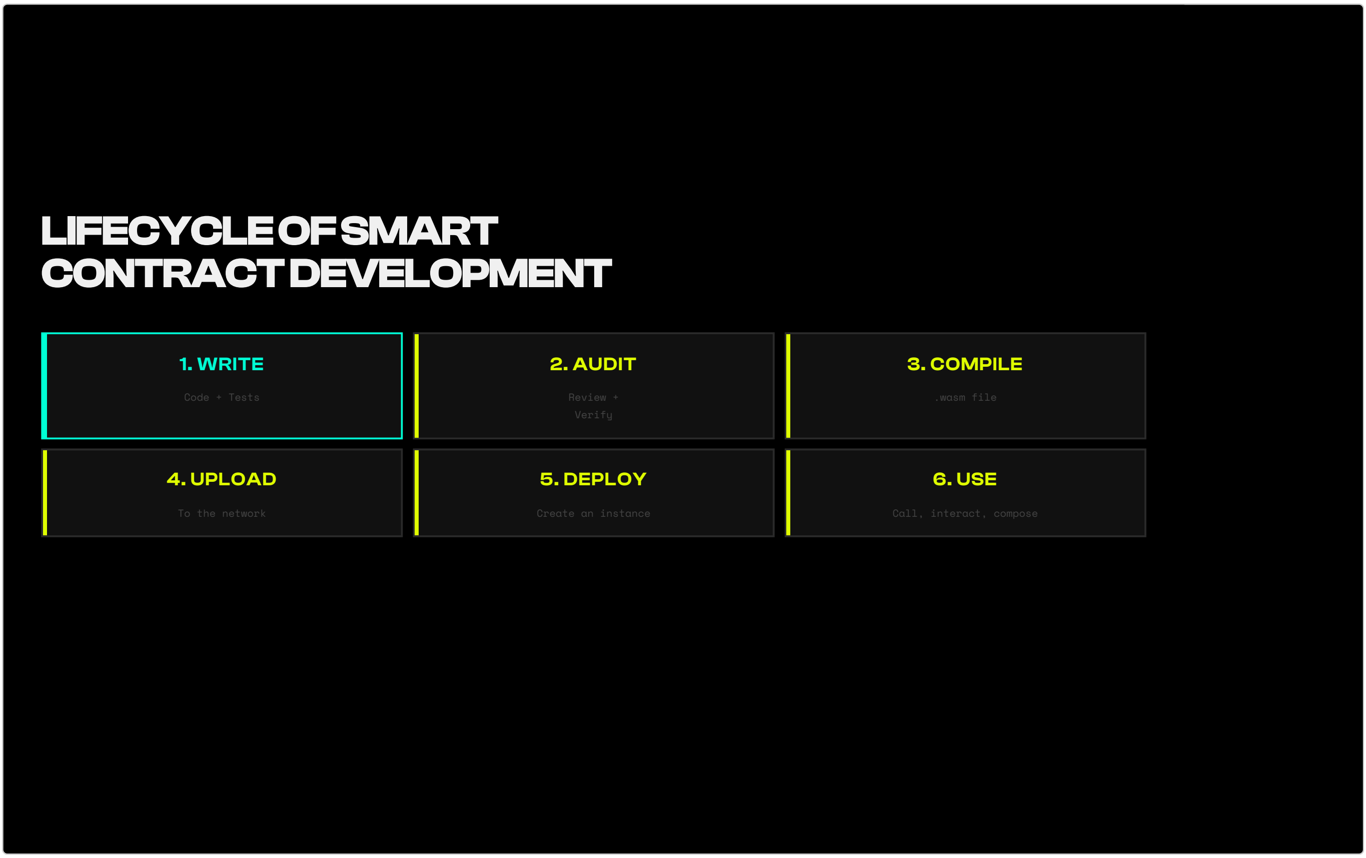
Create an instance

6. USE

Call, interact, compose

Slide Number: 17 / 37

Speaker Notes: Here's how a smart contract goes from idea to production. There are six steps.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE

Code + Tests

2. AUDIT

Review + Verify

3. COMPILE

.wasm file

4. UPLOAD

To the network

5. DEPLOY

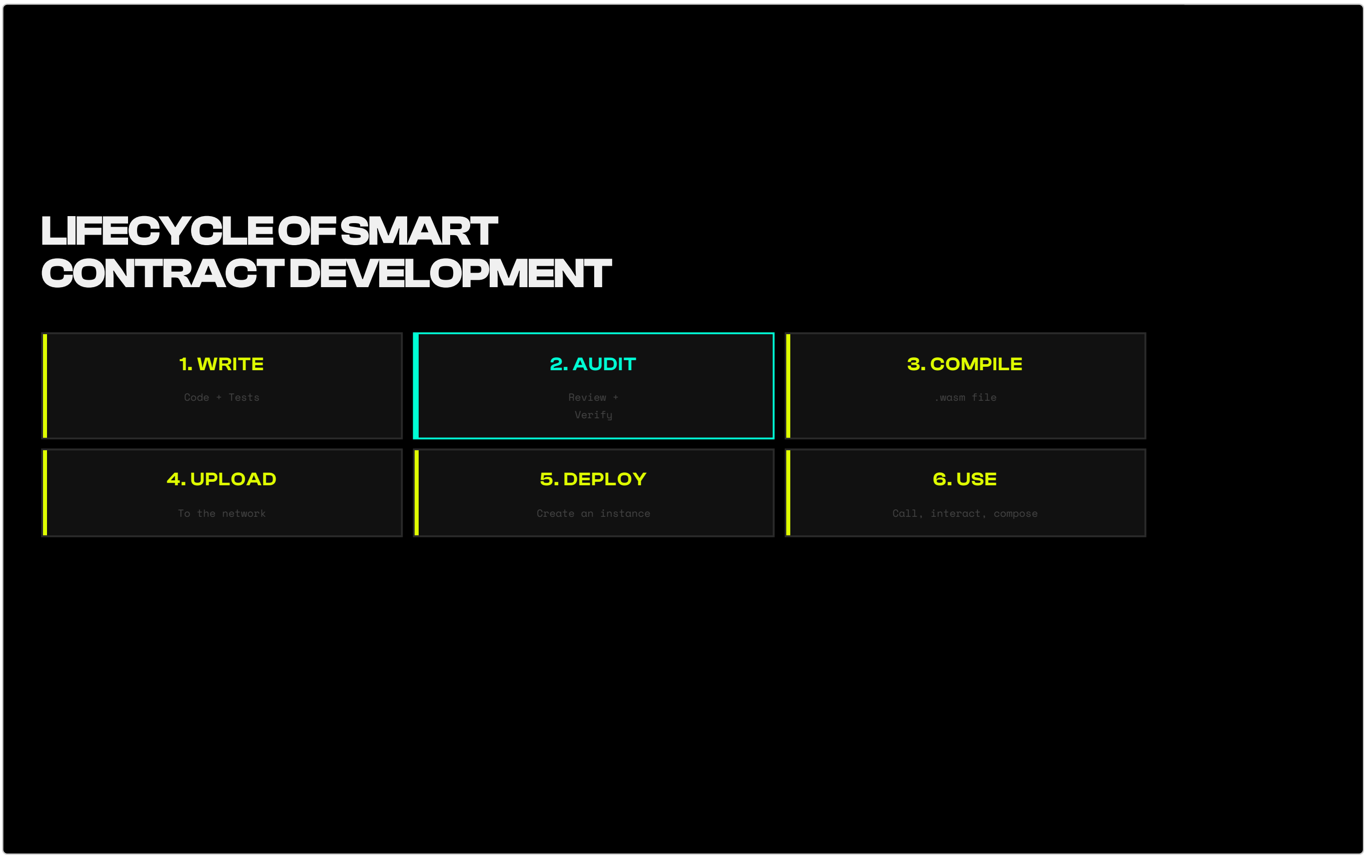
Create an instance

6. USE

Call, interact, compose

Slide Number: 18 / 37

Speaker Notes: First, it gets written in Rust using the soroban-sdk. Ideally what gets built is a mix of the on-chain logic and tests that give the developer confidence it does what they expect.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE

Code + Tests

2. AUDIT

Review +
Verify

3. COMPILE

.wasm file

4. UPLOAD

To the network

5. DEPLOY

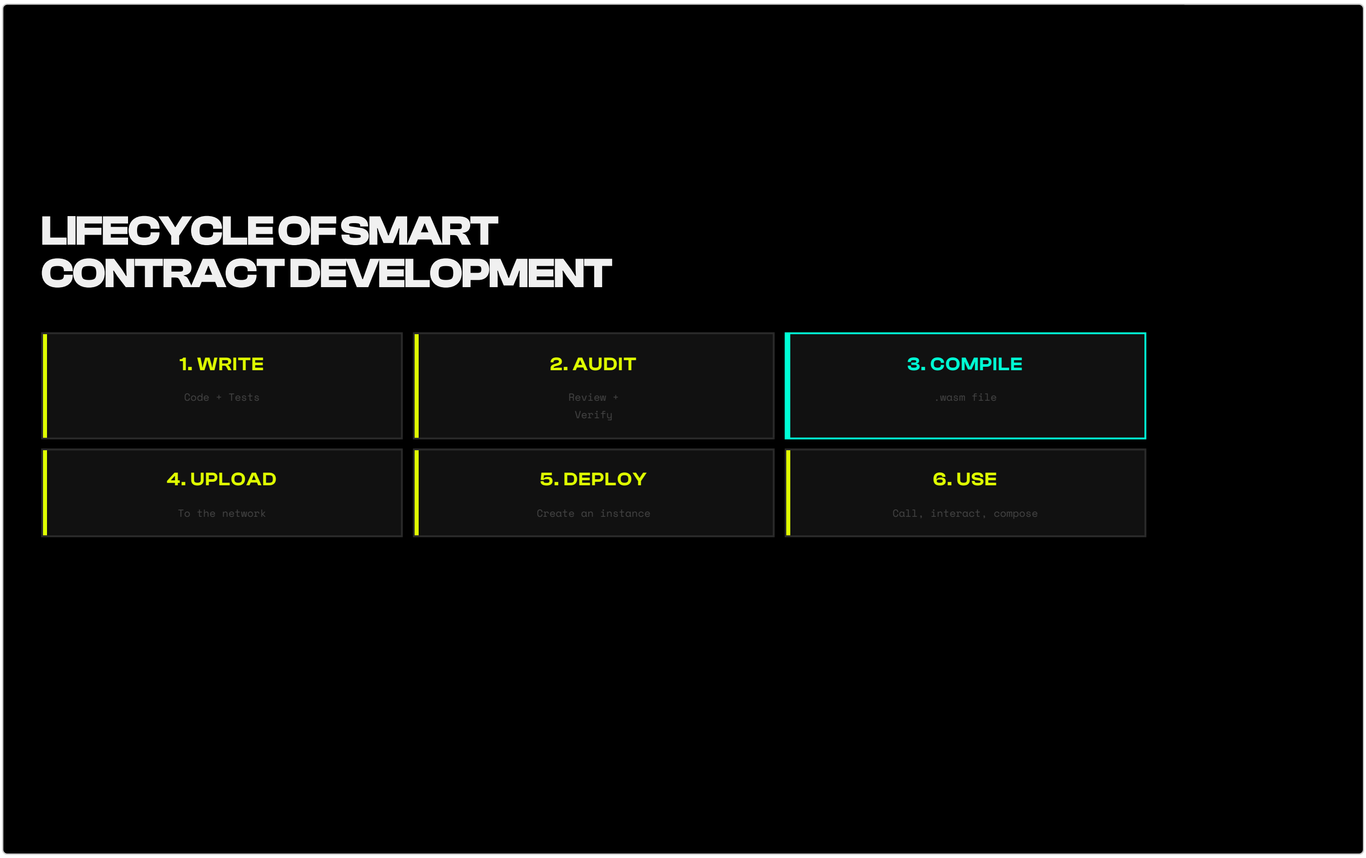
Create an instance

6. USE

Call, interact, compose

Slide Number: 19 / 37

Speaker Notes: Once the developer thinks it's ready, it gets audited — and this step is critical. Smart contracts can have bugs, just like vending machines sometimes get jammed. But unlike a $2 item getting stuck, a contract bug could put millions of dollars at risk. Audits serve two purposes: they give the creator a chance to find and fix bugs, and they build reputation. Most people can't read smart contract code and verify it does what it claims, so we rely on auditors, evidence, and reputation to close that gap.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE
Code + Tests

2. AUDIT
Review +
Verify

3. COMPILE
.wasm file

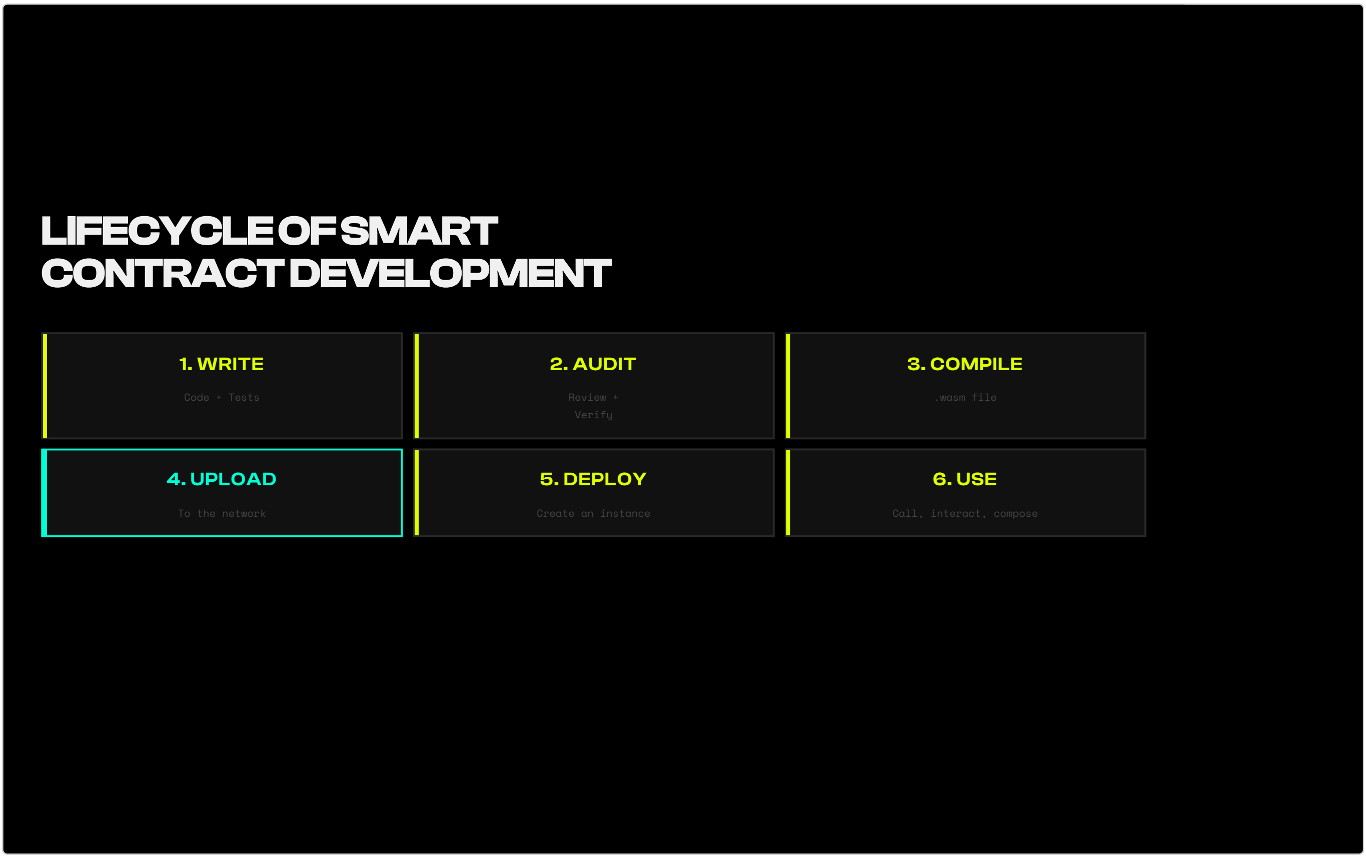
4. UPLOAD
To the network

5. DEPLOY
Create an instance

6. USE
Call, interact, compose

Slide Number: 20 / 37

Speaker Notes: After audit, the contract gets compiled into a WebAssembly file — a .wasm. This is portable, deterministic bytecode that the network can execute.

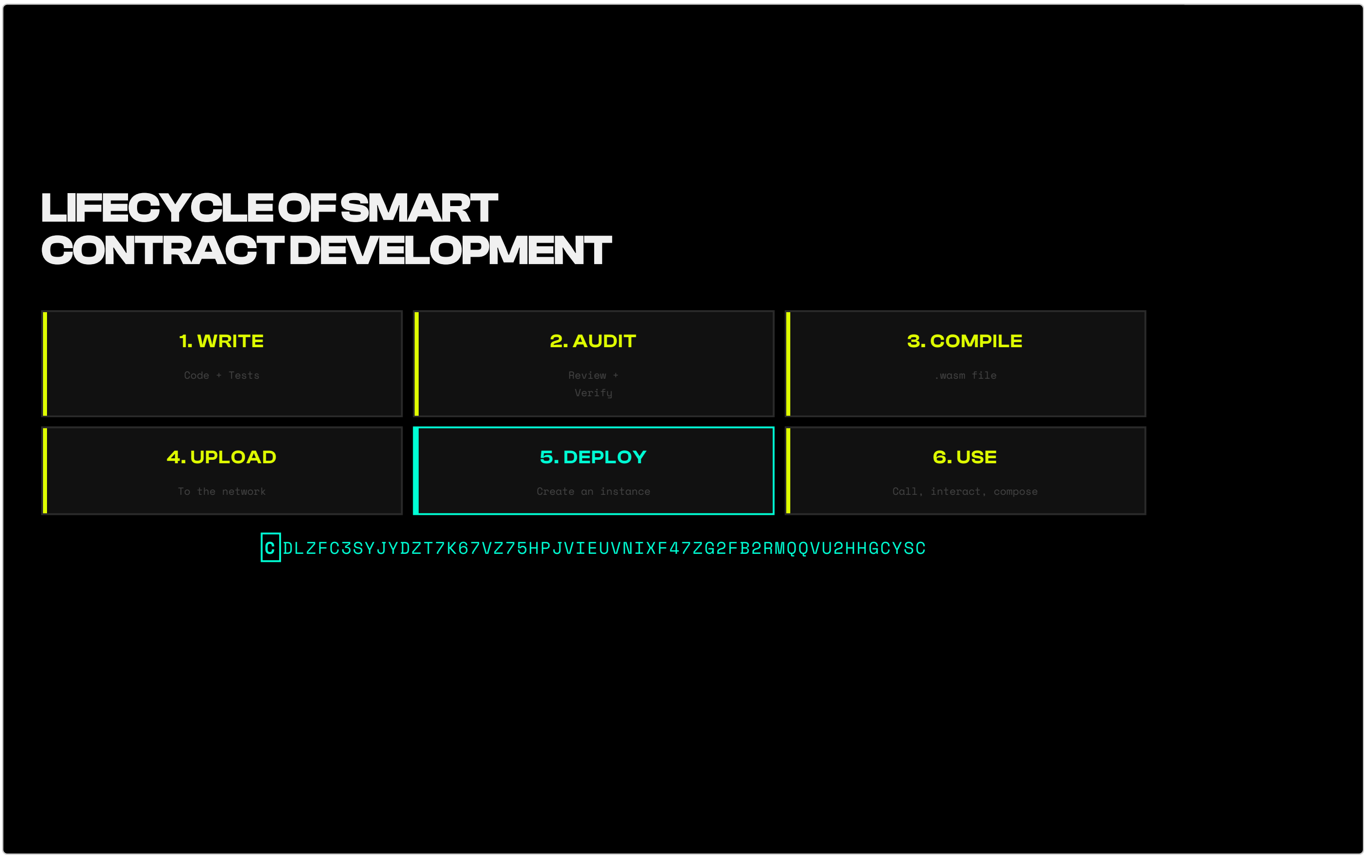


LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE Code + Tests	2. AUDIT Review + Verify	3. COMPILE .wasm file
4. UPLOAD To the network	5. DEPLOY Create an instance	6. USE Call, interact, compose

Slide Number: 21 / 37

Speaker Notes: The .wasm file gets uploaded to the Stellar network. At this point the code is on-chain, but it's not running yet — it's just stored, ready to be instantiated.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE
Code + Tests

2. AUDIT
Review + Verify

3. COMPILE
.wasm file

4. UPLOAD
To the network

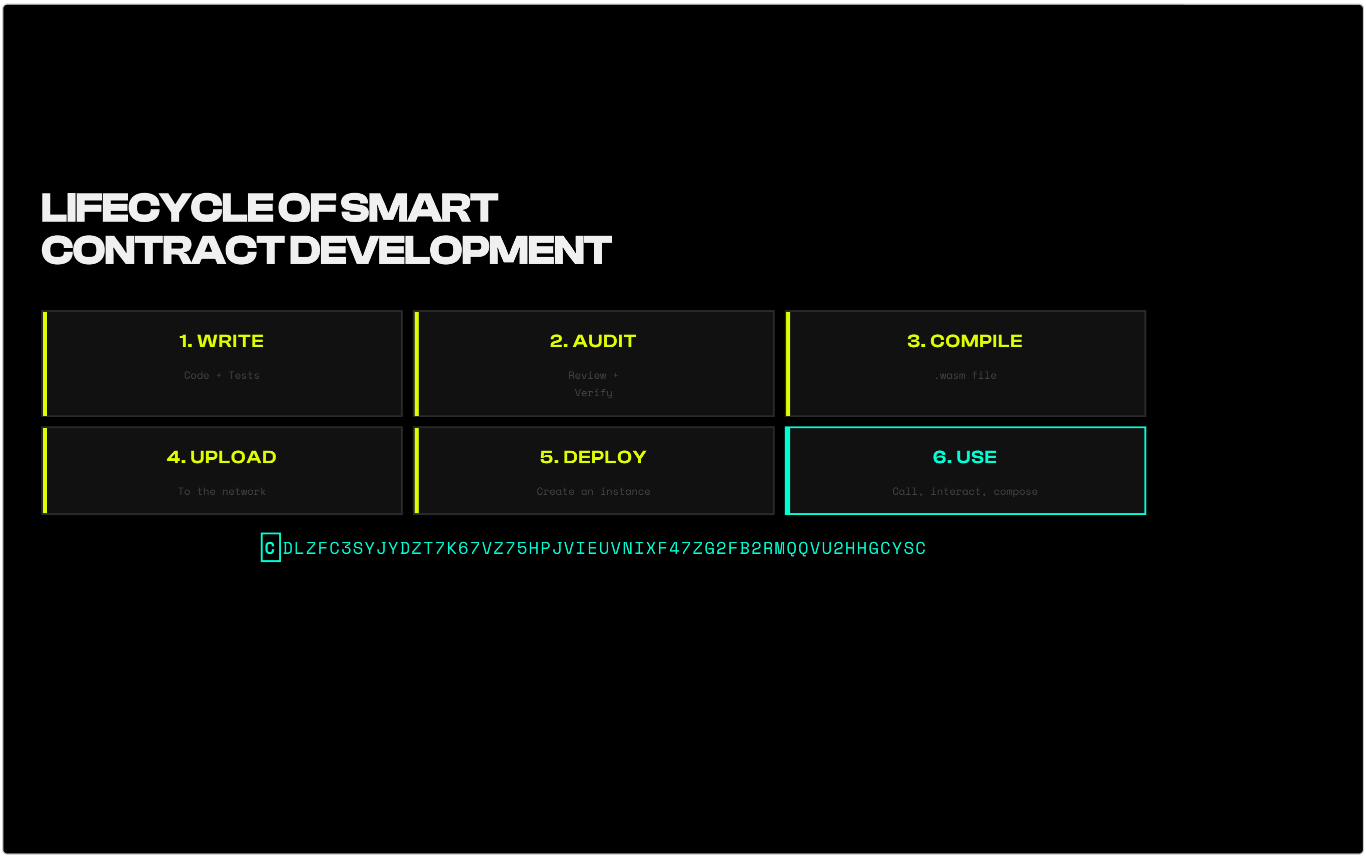
5. DEPLOY
Create an instance

6. USE
Call, interact, compose

CDLZFC3SYJYDZT7K67VZ75HPJVIEUVNIXF47ZG2FB2RMQQVU2HHGCYSC

Slide Number: 22 / 37

Speaker Notes: Anyone can deploy an instance of that uploaded code. Each instance gets its own contract address and its own storage. Think of it as an isolated copy of the machine, running independently.



LIFECYCLE OF SMART CONTRACT DEVELOPMENT

1. WRITE

Code + Tests

2. AUDIT

Review + Verify

3. COMPILE

.wasm file

4. UPLOAD

To the network

5. DEPLOY

Create an instance

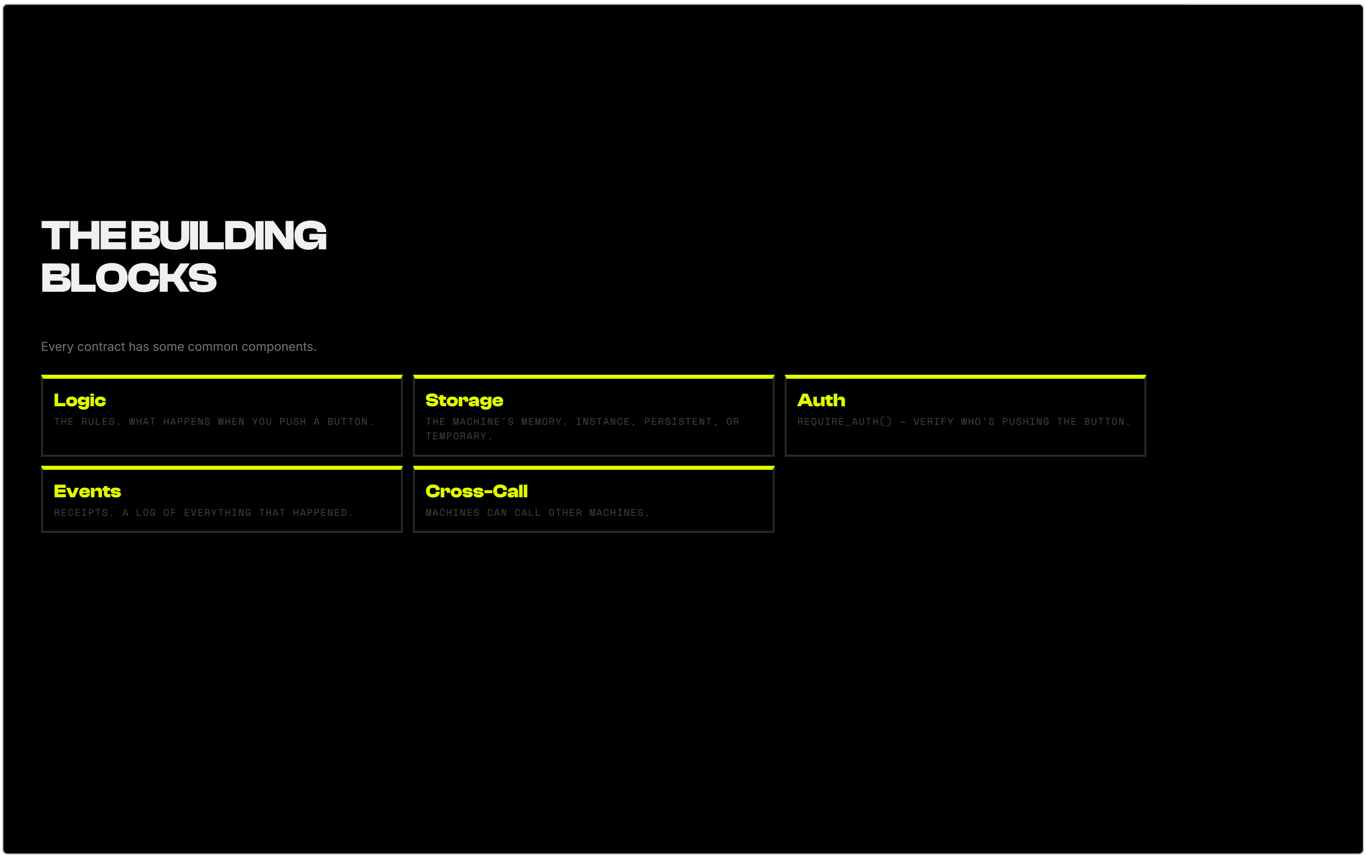
6. USE

Call, interact, compose

CDLZFC3SYJYDZT7K67VZ75HPJVIEUVNIXF47ZG2FB2RMQQVU2HHGCYSC

Slide Number: 23 / 37

Speaker Notes: And then people use it — calling its functions, interacting with it, composing it with other contracts. This is where the contract starts doing real work.



THE BUILDING BLOCKS

Every contract has some common components.

Logic

THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage

THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

Auth

REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events

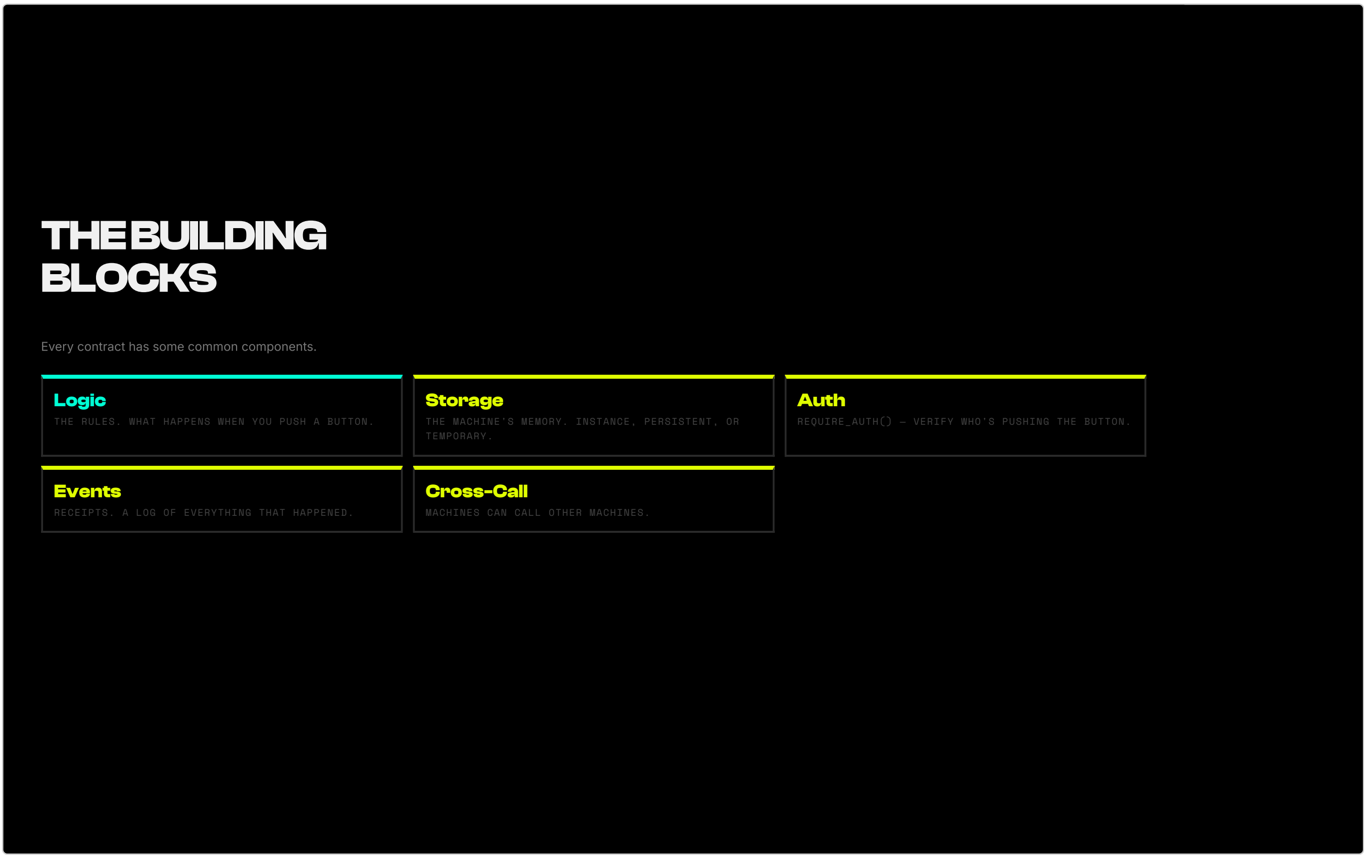
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call

MACHINES CAN CALL OTHER MACHINES.

Slide Number: 24 / 37

Speaker Notes: Every contract is built from the same primitives. Let's walk through each one.



THE BUILDING BLOCKS

Every contract has some common components.

Logic

THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage

THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

Auth

REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events

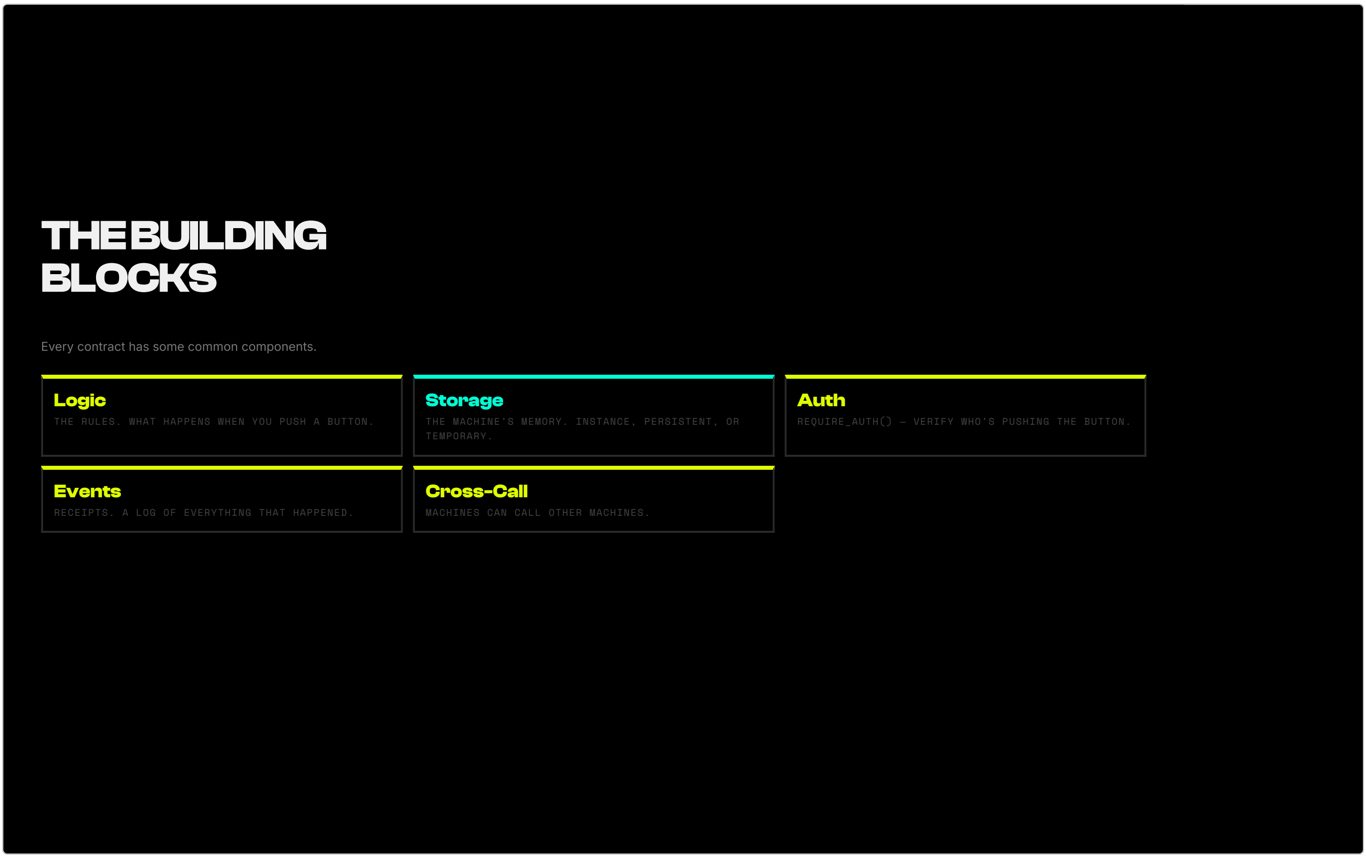
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call

MACHINES CAN CALL OTHER MACHINES.

Slide Number: 25 / 37

Speaker Notes: First: logic. This is the code that defines the rules — what happens when someone calls a function on the contract.



THE BUILDING BLOCKS

Every contract has some common components.

Logic
THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage
THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

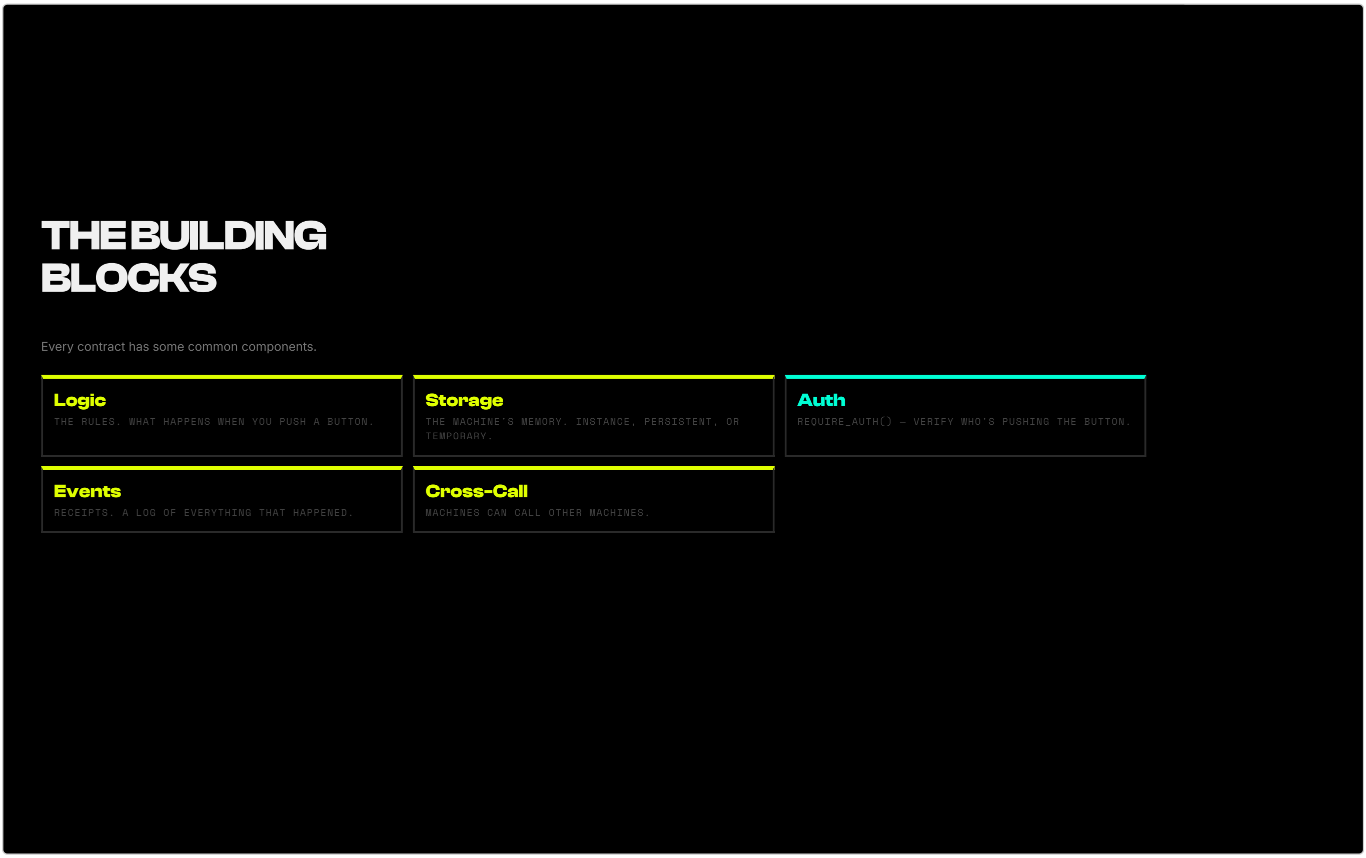
Auth
REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call
MACHINES CAN CALL OTHER MACHINES.

Slide Number: 26 / 37

Speaker Notes: Storage is the contract's memory. Data that persists between calls. On Stellar there are three types: instance, persistent, and temporary — each with different lifetimes and costs.



THE BUILDING BLOCKS

Every contract has some common components.

Logic
THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage
THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

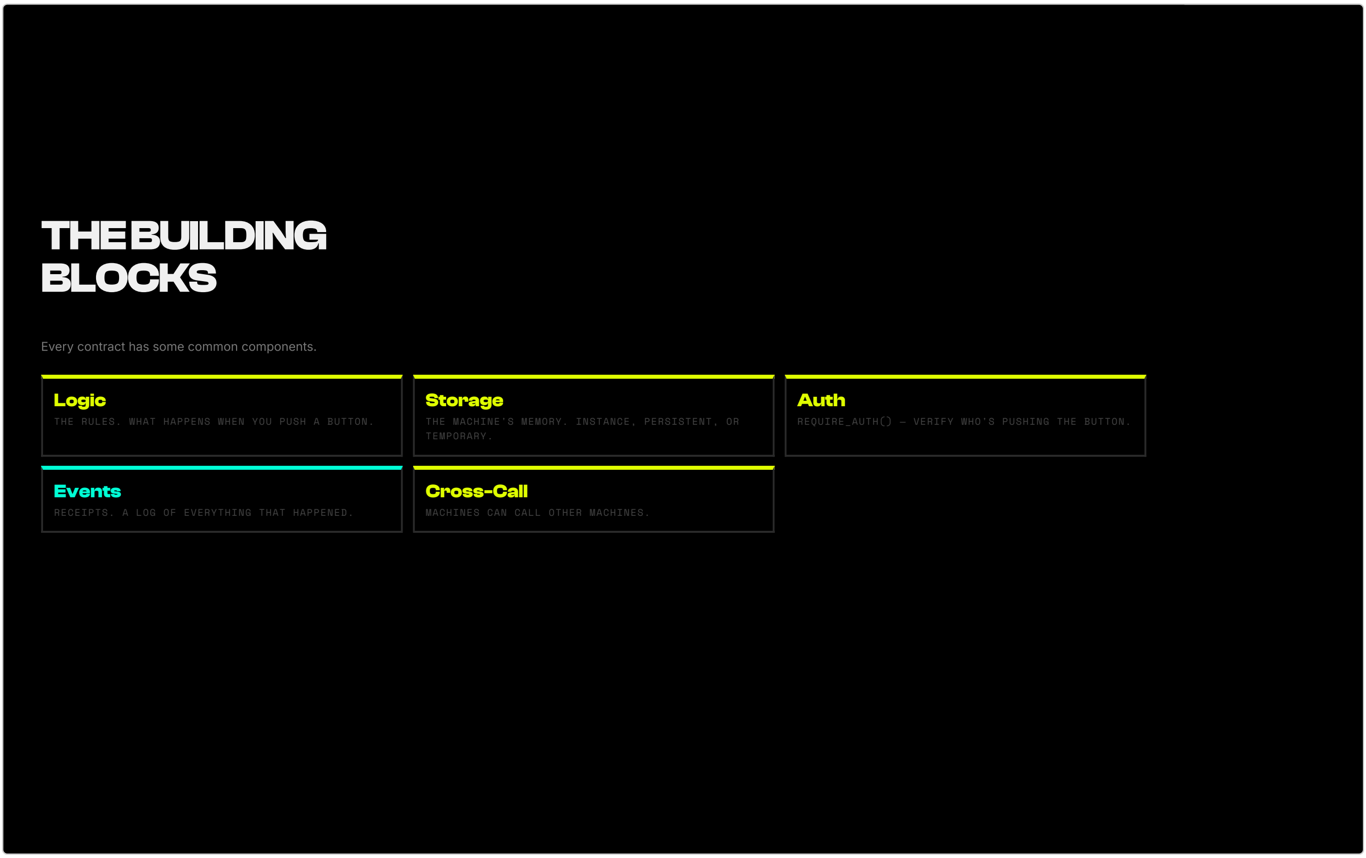
Auth
REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call
MACHINES CAN CALL OTHER MACHINES.

Slide Number: 27 / 37

Speaker Notes: Auth — verifying who's calling. The require_auth function checks that the caller has signed the transaction. This is how contracts enforce permissions.



THE BUILDING BLOCKS

Every contract has some common components.

Logic
THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage
THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

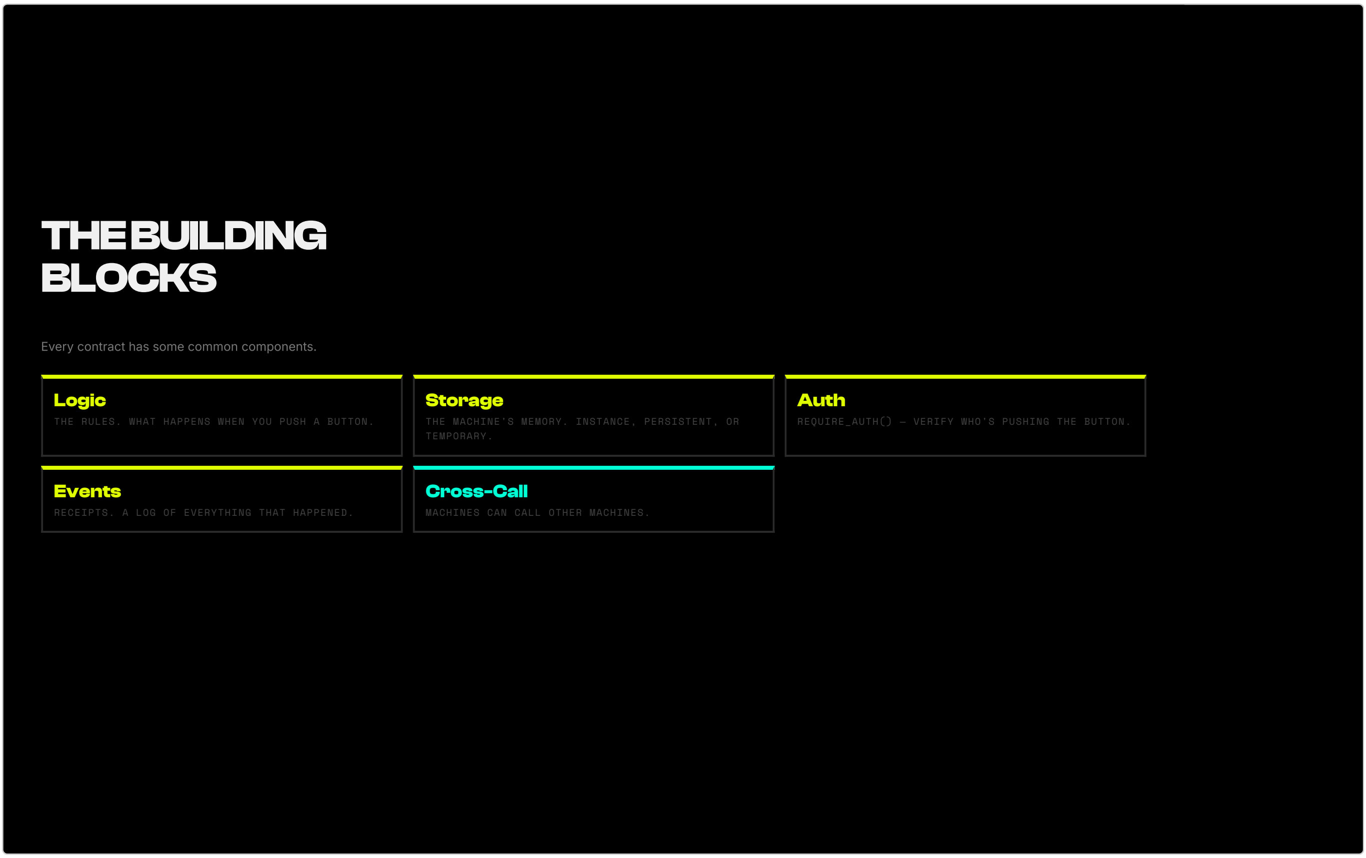
Auth
REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call
MACHINES CAN CALL OTHER MACHINES.

Slide Number: 28 / 37

Speaker Notes: Events are like receipts. A log of what happened during the call. Off-chain systems — wallets, explorers, analytics — consume these to track activity.



THE BUILDING BLOCKS

Every contract has some common components.

Logic
THE RULES. WHAT HAPPENS WHEN YOU PUSH A BUTTON.

Storage
THE MACHINE'S MEMORY. INSTANCE, PERSISTENT, OR TEMPORARY.

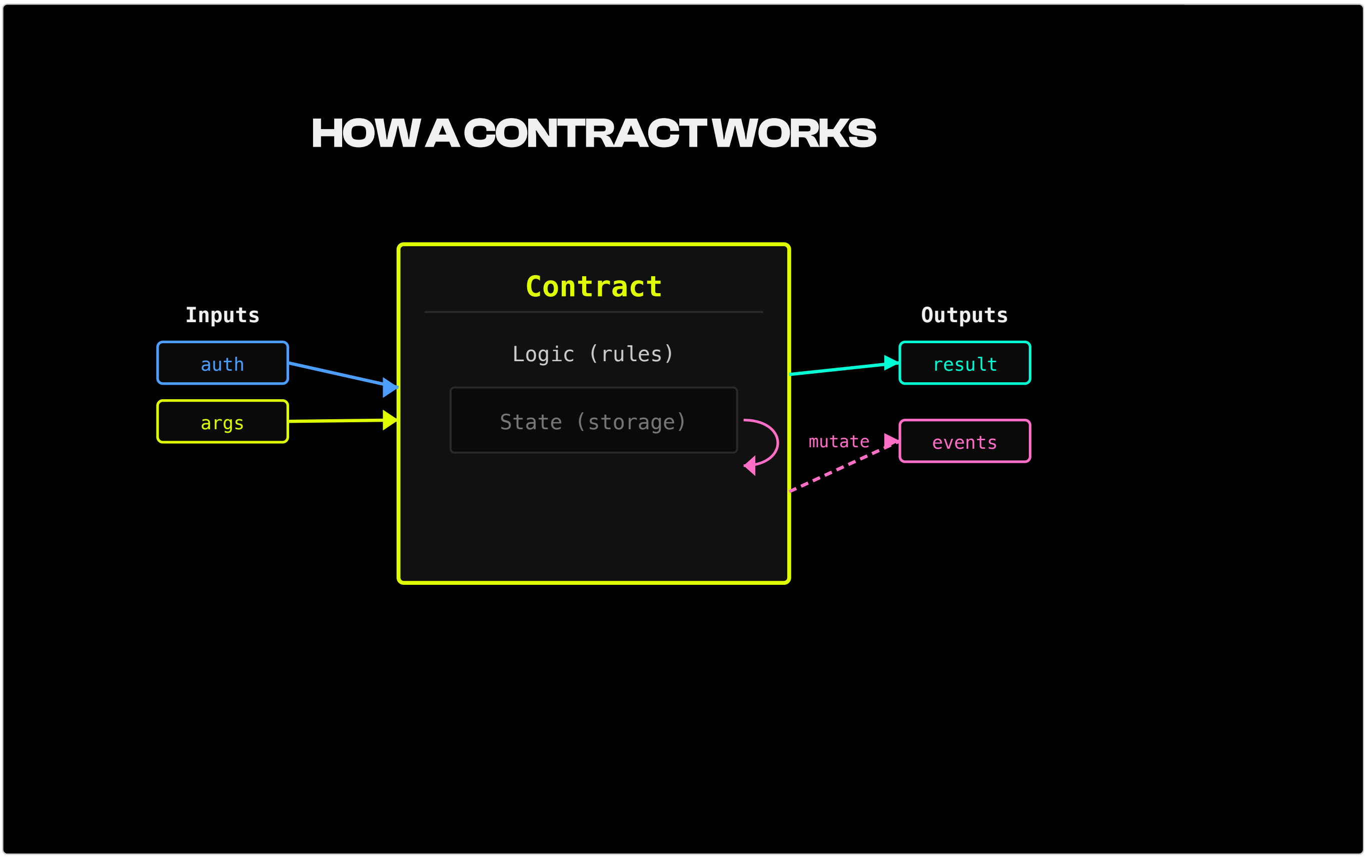
Auth
REQUIRE_AUTH() — VERIFY WHO'S PUSHING THE BUTTON.

Events
RECEIPTS. A LOG OF EVERYTHING THAT HAPPENED.

Cross-Call
MACHINES CAN CALL OTHER MACHINES.

Slide Number: 29 / 37

Speaker Notes: And cross-contract calls. Contracts can invoke other contracts. A swap contract calls token contracts. A lending contract calls price oracles. This composability is what makes the ecosystem powerful. These building blocks come together in a flow that looks like this...



HOW A CONTRACT WORKS

Inputs

auth

args

Contract

Logic (rules)

State (storage)

mutate

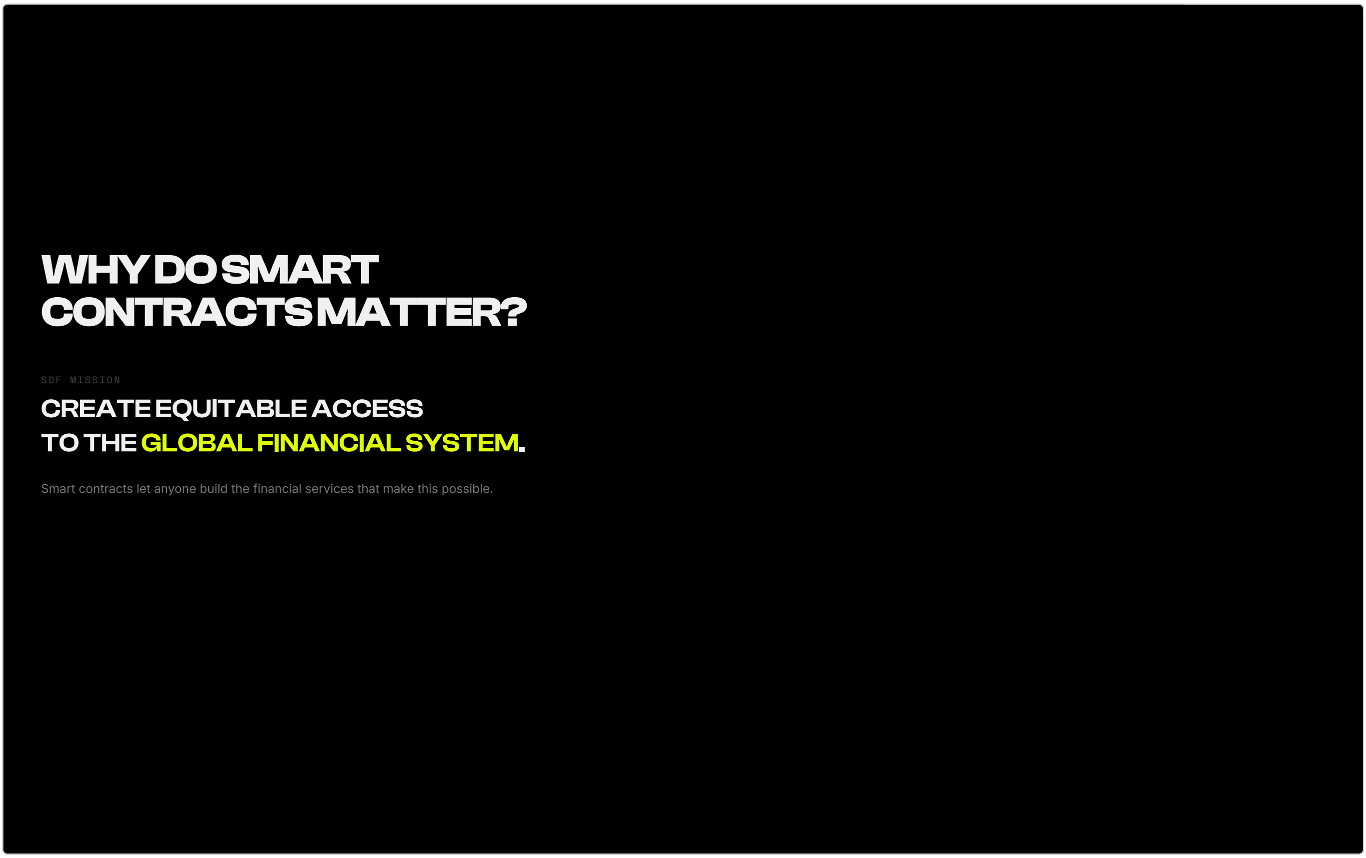
Outputs

result

events

Slide Number: 30 / 37

Speaker Notes: These building blocks come together in a flow that looks like this... When calling a contract, inputs come in. In a payment using a token contract this might be a sender address, a destination address, and an amount. Auth to verify who's authorising the action to take place. It's super important that for the payment that the contract checks that the sender authorised the payment. The logic runs. State, like account balances, gets loaded, read, then updated. A result comes back and is returned to the caller. And events are published for off-chain systems to consume.



WHY DO SMART CONTRACTS MATTER?

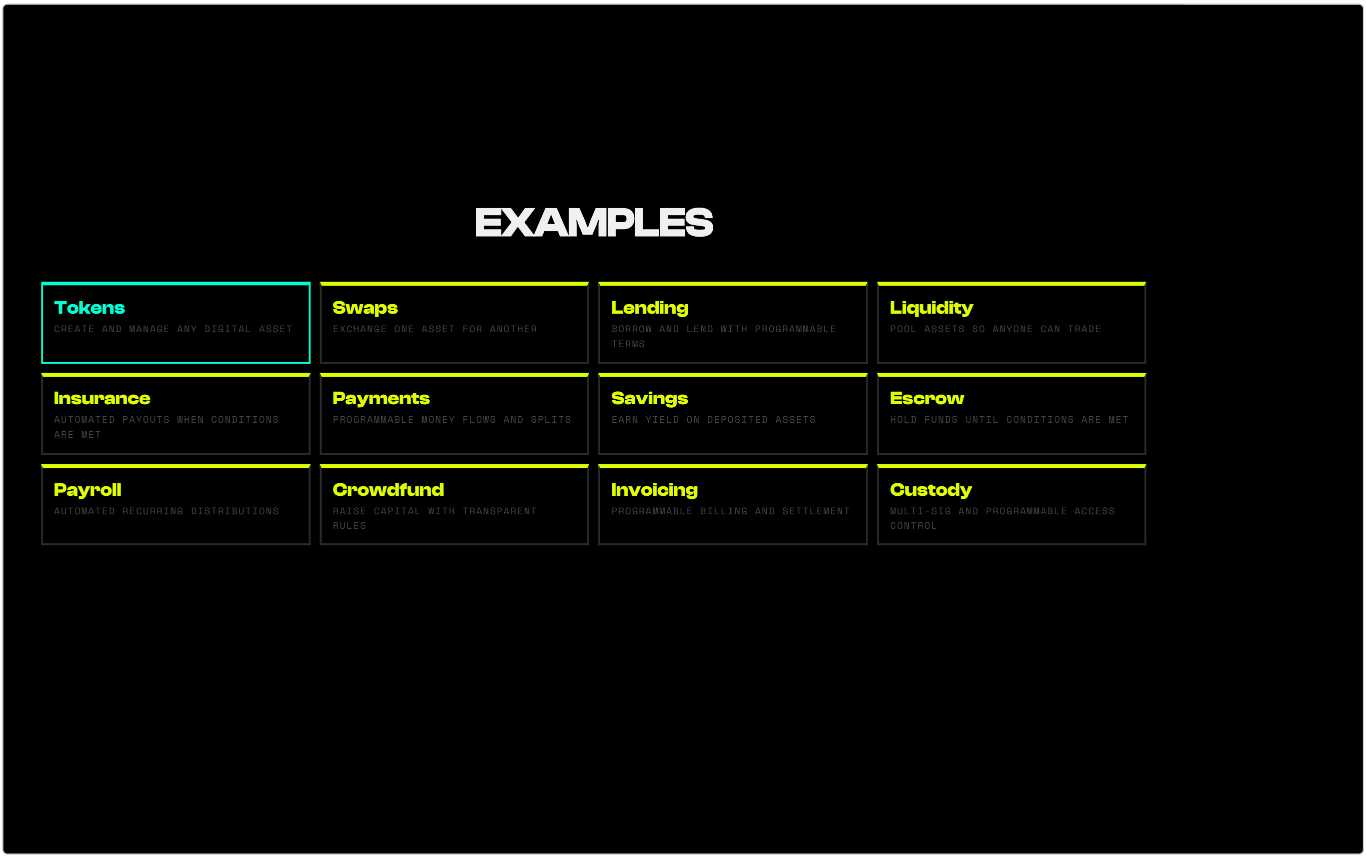
SDF MISSION

CREATE EQUITABLE ACCESS TO THE GLOBAL FINANCIAL SYSTEM.

Smart contracts let anyone build the financial services that make this possible.

Slide Number: 31 / 37

Speaker Notes: So why does all this matter? The Stellar Development Foundation's mission is to create equitable access to the global financial system. Smart contracts are a key tool for that — they let anyone build financial services that are open, transparent, and accessible.



EXAMPLES

Tokens
CREATE AND MANAGE ANY DIGITAL ASSET

Swaps
EXCHANGE ONE ASSET FOR ANOTHER

Lending
BORROW AND LEND WITH PROGRAMMABLE TERMS

Liquidity
POOL ASSETS SO ANYONE CAN TRADE

Insurance
AUTOMATED PAYOUTS WHEN CONDITIONS ARE MET

Payments
PROGRAMMABLE MONEY FLOWS AND SPLITS

Savings
EARN YIELD ON DEPOSITED ASSETS

Escrow
HOLD FUNDS UNTIL CONDITIONS ARE MET

Payroll
AUTOMATED RECURRING DISTRIBUTIONS

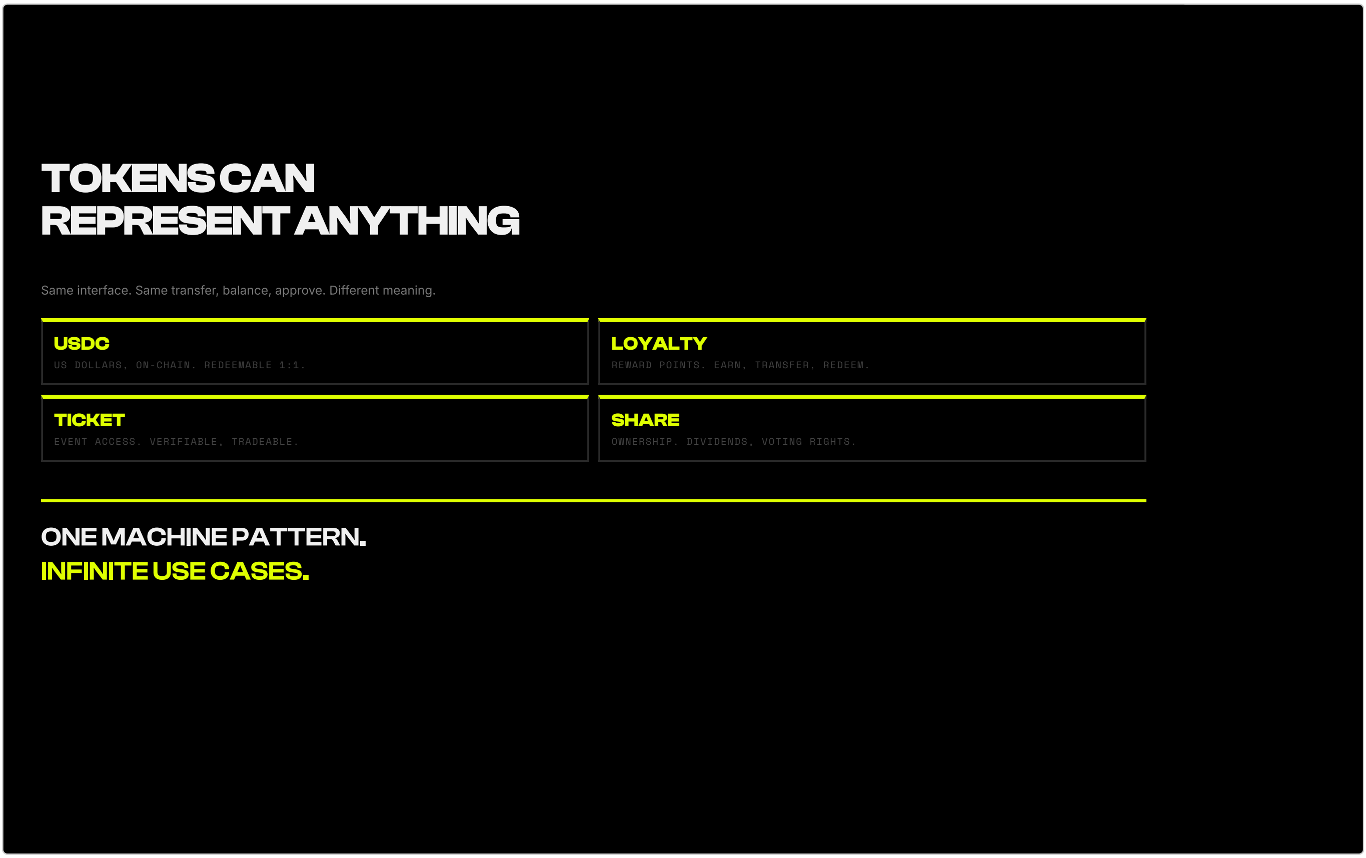
Crowdfund
RAISE CAPITAL WITH TRANSPARENT RULES

Invoicing
PROGRAMMABLE BILLING AND SETTLEMENT

Custody
MULTI-SIG AND PROGRAMMABLE ACCESS CONTROL

Slide Number: 32 / 37

Speaker Notes: Smart contracts on Stellar are designed to power any financial service — tokens, swaps, lending, liquidity pools, insurance, payments, savings, escrow, and more. Anything you can describe as a set of rules, you can build as a contract. Of all those things, tokens are really the core building block and a concept you'll hear of frequently.



TOKENS CAN REPRESENT ANYTHING

Same interface. Same transfer, balance, approve. Different meaning.

USDC
US DOLLARS, ON-CHAIN. REDEEMABLE 1:1.

LOYALTY
REWARD POINTS. EARN, TRANSFER, REDEEM.

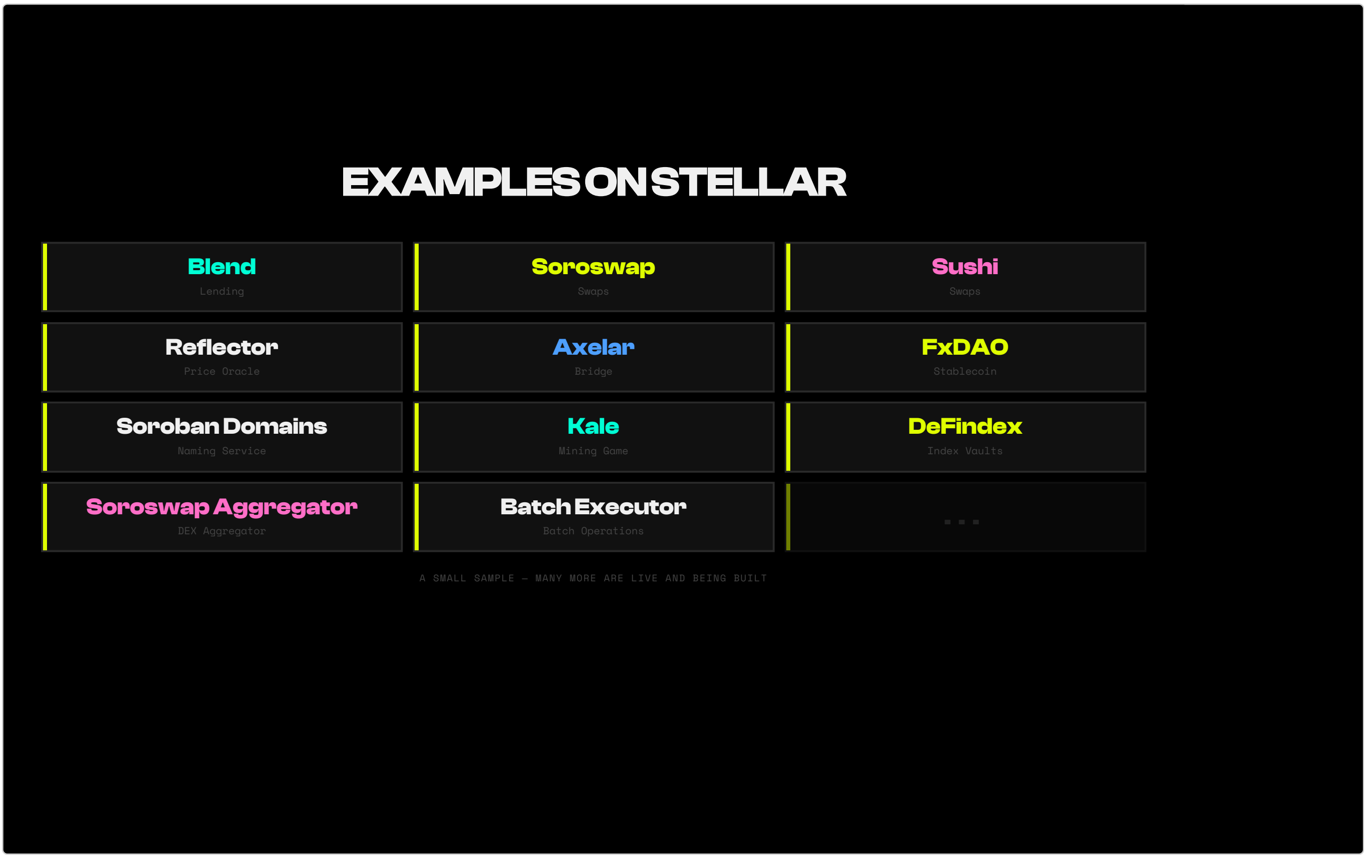
TICKET
EVENT ACCESS. VERIFIABLE, TRADEABLE.

SHARE
OWNERSHIP. DIVIDENDS, VOTING RIGHTS.

ONE MACHINE PATTERN.
INFINITE USE CASES.

Slide Number: 33 / 37

Speaker Notes: A token is a digital unit that represents some form of value, a right, or an asset. And a token contract tracks ownership of that token. The storage of a token contract is typically a big table listing each account holder and what balance they hold. Operations on the token contract like a transfer update the rows in that table. If I send $10 to you, my row will be reduce by $10, and yours will be increased by $10. Tokens get used in all of the other services listed on the previous screen. When you hear terms like USDC, or PYUSD, both of these things are tokens.



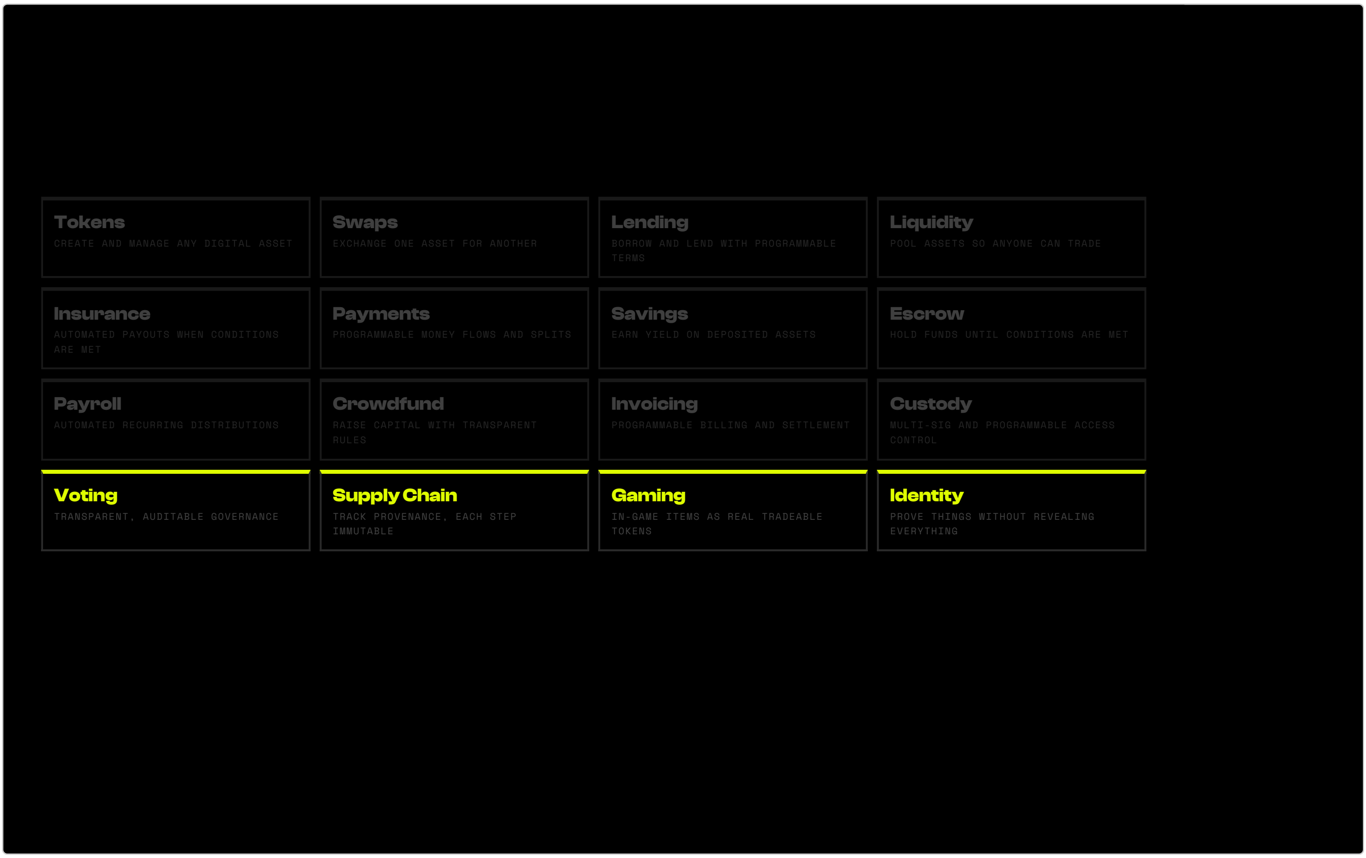
EXAMPLES ON STELLAR

Blend Lending	**Soroswap** Swaps	**Sushi** Swaps
Reflector Price Oracle	**Axelar** Bridge	**FxDAO** Stablecoin
Soroban Domains Naming Service	**Kale** Mining Game	**DeFindex** Index Vaults
Soroswap Aggregator DEX Aggregator	**Batch Executor** Batch Operations	...

A SMALL SAMPLE — MANY MORE ARE LIVE AND BEING BUILT

Slide Number: 34 / 37

Speaker Notes: These aren't hypotheticals — there are real smart contracts live on Stellar right now. Blend is a lending protocol. Soroswap and Sushi handle swaps. Reflector provides price data as an oracle. Axelar bridges assets from other chains. FxDAO creates stablecoins. Soroban Domains is a naming service. Kale is a game built entirely on-chain. And DeFindex provides index vaults. This ecosystem is growing fast.



Tokens
CREATE AND MANAGE ANY DIGITAL ASSET

Swaps
EXCHANGE ONE ASSET FOR ANOTHER

Lending
BORROW AND LEND WITH PROGRAMMABLE TERMS

Liquidity
POOL ASSETS SO ANYONE CAN TRADE

Insurance
AUTOMATED PAYOUTS WHEN CONDITIONS ARE MET

Payments
PROGRAMMABLE MONEY FLOWS AND SPLITS

Savings
EARN YIELD ON DEPOSITED ASSETS

Escrow
HOLD FUNDS UNTIL CONDITIONS ARE MET

Payroll
AUTOMATED RECURRING DISTRIBUTIONS

Crowdfund
RAISE CAPITAL WITH TRANSPARENT RULES

Invoicing
PROGRAMMABLE BILLING AND SETTLEMENT

Custody
MULTI-SIG AND PROGRAMMABLE ACCESS CONTROL

Voting
TRANSPARENT, AUDITABLE GOVERNANCE

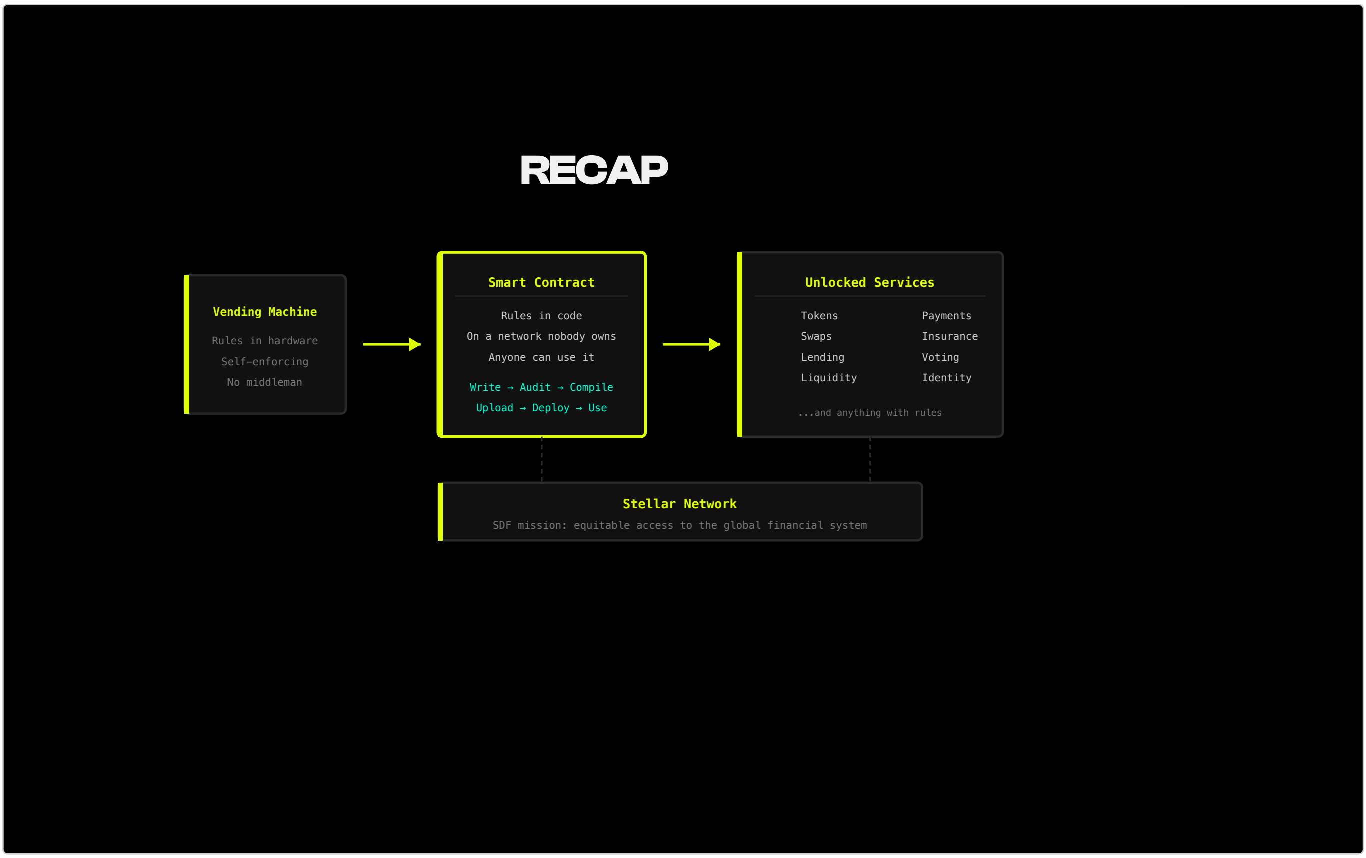
Supply Chain
TRACK PROVENANCE, EACH STEP IMMUTABLE

Gaming
IN-GAME ITEMS AS REAL TRADEABLE TOKENS

Identity
PROVE THINGS WITHOUT REVEALING EVERYTHING

Slide Number: 35 / 37

Speaker Notes: The focus for us at SDF is financial services, because our mission is about access to the global financial system. But what's possible with smart contracts goes beyond finance. There are many other things that people are using blockchain for: voting, supply chain tracking, gaming items, digital identity — anywhere you need rules that enforce themselves, a smart contract can do the job.



RECAP

Vending Machine

Rules in hardware
Self-enforcing
No middleman

Smart Contract

Rules in code
On a network nobody owns
Anyone can use it

Write → Audit → Compile
Upload → Deploy → Use

Unlocked Services

Tokens
Swaps
Lending
Liquidity

Payments
Insurance
Voting
Identity

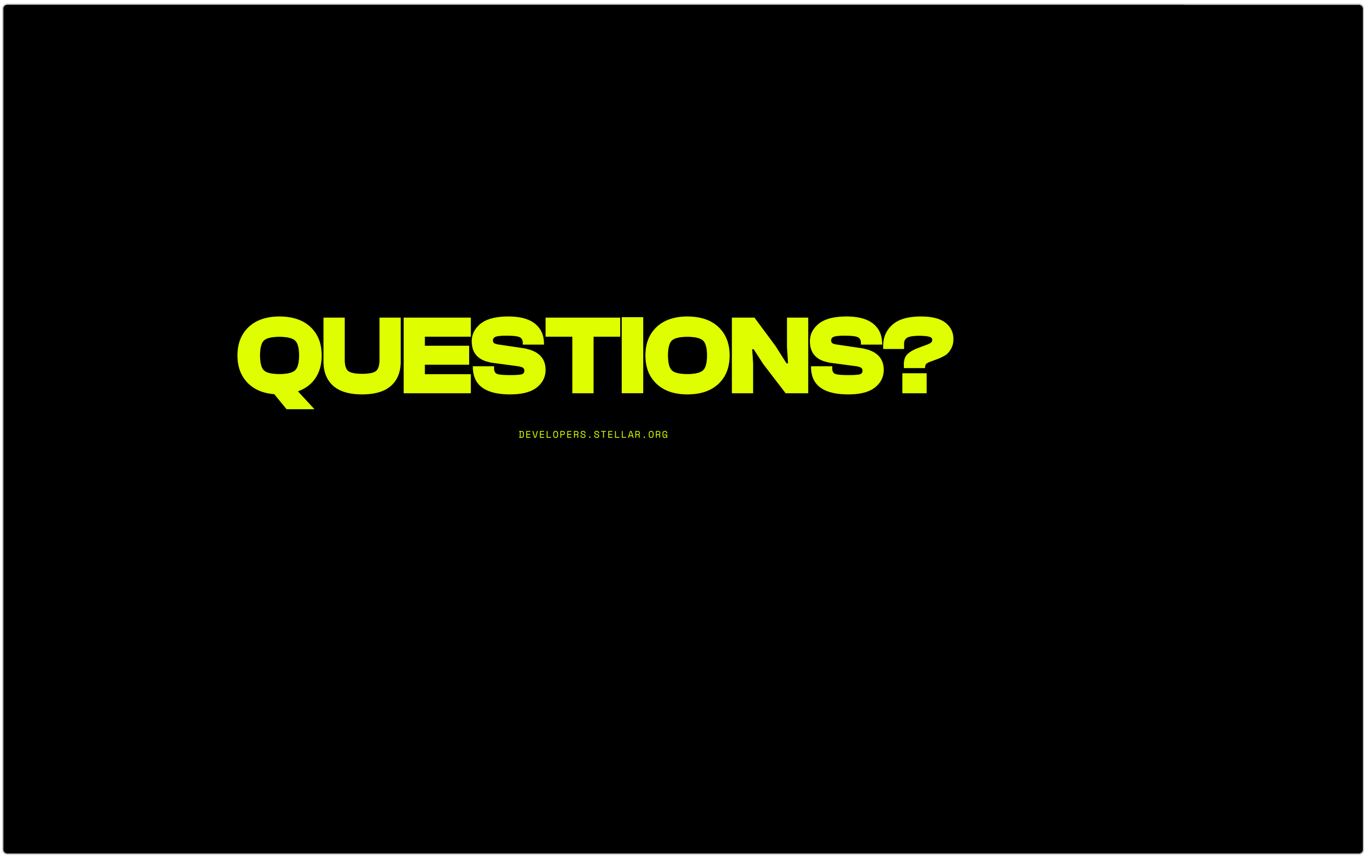
...and anything with rules

Stellar Network

SDF mission: equitable access to the global financial system

Slide Number: 36 / 37

Speaker Notes: To bring it all together: we talked about vending machines — how they are a self-enforcing system, of rules defined in hardware. A smart contract on a blockchain is the same idea in software, running on a decentralized network. On Stellar, smart contracts unlock every kind of financial service, they enable anyone to build financial services and anyone to integrate other financial systems. And this is all in service of the mission — to create equitable access to the global financial system.



QUESTIONS?

DEVELOPERS.STELLAR.ORG

Slide Number: 37 / 37

Speaker Notes: I said at the beginning this talk wasn't for developers, but if you want to get hands on and build your own contract and deploy it to testnet, developers.stellar.org has everything you need. Questions?