TRAILS TECH · OPEN SOURCE

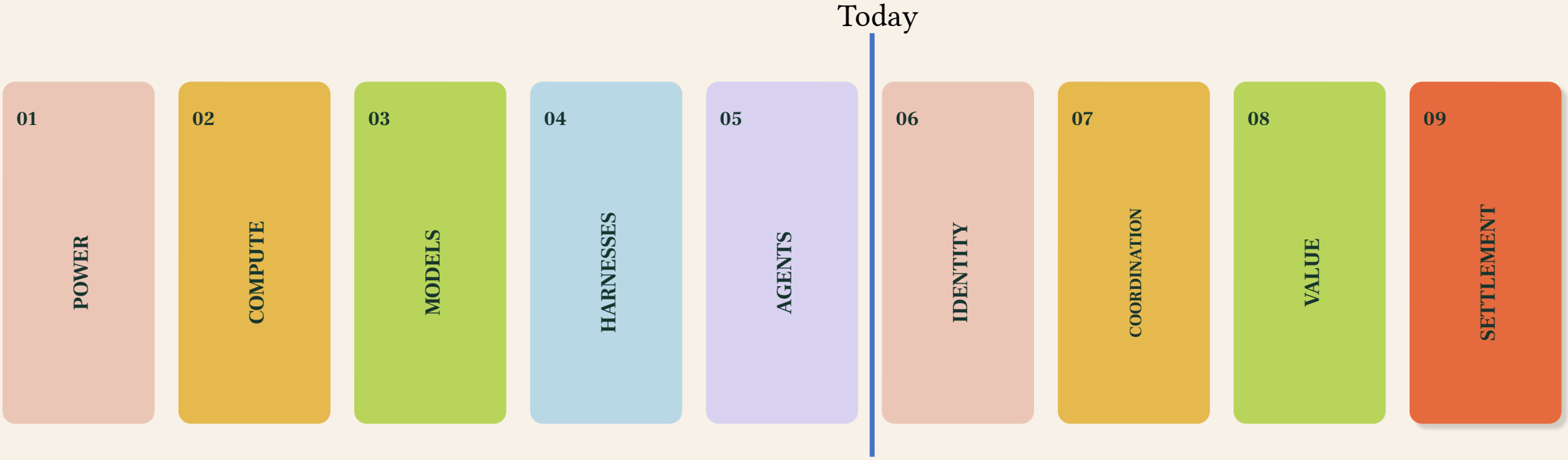
Open networks for an agentic world

Open source gives us inspectable tools.
Open protocols let systems connect.
Open networks let participants exchange value.

25-minute talk · then demos + Q&A

The stack extends beyond intelligence

As intelligence becomes abundant, coordination, identity, and value become the scarce layers.



If agents become economic actors, the stack cannot end with "answer the prompt."

They need to identify, coordinate, pay, and settle.

Open is not one thing

Three layers—and each solves a different dependency.

01

OPEN-SOURCE SOFTWARE

The code can be inspected, modified, and redistributed.

Linux · Goose · Buzz

02

OPEN PROTOCOL

Independent implementations can speak the same language.

HTTP · Nostr · MCP · A2A

03

OPEN NETWORK

Many parties can participate without one owner controlling every account or transaction.

The web · Nostr · permissionless money

Open code can still live inside a closed platform. The real unlock is interoperability plus permissionless participation.

An open agent stack is replaceable by design

No single vendor has to own the model, the tool connection, the identity, the room, and the money.

INTELLIGENCE	**Models**	Open-weight or commercial—swappable
RUNTIME	**Goose / other harnesses**	Context, tools, memory, policy
TOOLS	**MCP**	Common connection to data and capabilities
AGENT ↔ AGENT	**A2A**	Interoperable delegation and collaboration
IDENTITY	**Nostr keys**	Portable identity and signed activity
COORDINATION	**Buzz**	Shared rooms for humans and agents
VALUE	**Payment protocol / HTTP 402**	Fast, programmable exchange
SETTLEMENT	**Neutral monetary network**	**A shared final record nobody controls**

Replace one layer without surrendering the whole system.

[4] [5] [7] [8]

Linux became the shared floor.

Companies stopped needing to own the operating system to build valuable businesses above it.

1991

Linux kernel begins

2000

IBM commits $1B

TODAY

Linux runs most cloud infrastructure

1. One shared foundation
2. Many competing implementations
3. **Value moves to service, integration, and experience**

[1] [2]

An agent needs resources/data.

DISCOVER

Find a compatible supplier or API

IDENTIFY

Sign with a portable key

REQUEST

Ask for price, stock, or delivery data

PAY

Approve a bounded micropayment

VERIFY

Receive data and a signed receipt

No new corporate account for every relationship. Policy still matters—but permission can be encoded, bounded, and audited.

EVERYONE KNOWS

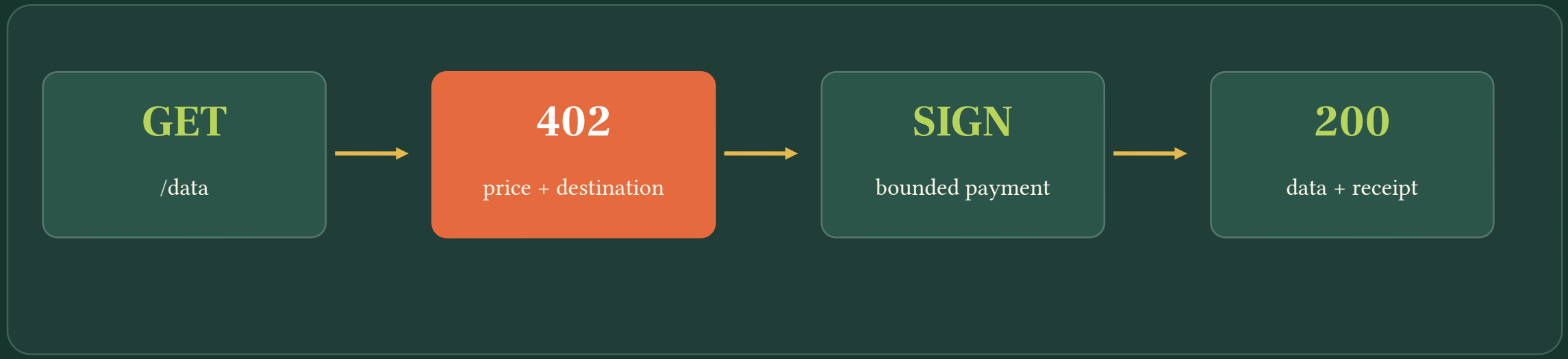
404

NOT FOUND

FEWER PEOPLE KNOW

402

PAYMENT REQUIRED



HTTP reserved 402 for future use. A machine-readable price can replace signup friction—and make mass abuse economically costly.

[13] [14]

Agents break the account-based internet

Human friction becomes an architectural bottleneck at machine scale.

THE CLOSED PATH

1. Create an account
2. Verify email / phone
3. **Wait for approval**
4. Store API credentials
5. Add a billing method
6. Accept changing platform rules

THE OPEN PATH

- ✓ **Generate cryptographic identity**
- ✓ Discover a compatible service
- ✓ Read terms in a standard response
- ✓ Sign a request or payment
- ✓ Receive result + receipt
- ✓ Switch implementation without starting over

A person can tolerate ten logins. A swarm of agents cannot tolerate ten thousand permission gates.

Agents need identity without an account desk

Opening a Facebook or X account can be impractical—or impossible—for autonomous software.

ACCOUNT-BASED IDENTITY

FACEBOOK
Email · phone · CAPTCHA · platform approval

X
Account credentials · automation rules · rate limits

ANOTHER SERVICE
Another signup · another identity · another dependency

The platform creates the account, controls access, and can remove the identity.

KEY-BASED IDENTITY · NOSTR

1. **GENERATE** Create a keypair locally—no signup form.
2. **KEEP** The private key stays secret.
3. **SHARE** The public key becomes the portable identity.
4. **SIGN** Messages are signed; anyone can verify the author.

Permission may govern a room. It does not have to own the identity.

[4] [11] [12] [20] [21]

Open Source options

Goose opens the agent runtime. Buzz opens the shared room where humans and agents work.

GOOSE / HERMES / OPENCLAW

OPEN AGENT / HARNESS

- Model-agnostic runtime
- Connects to tools through MCP
- Community can extend and inspect it
- The agent can outlive one vendor model

BUZZ

OPEN COORDINATION

- Humans and agents share rooms
- Nostr identities and signed activity
- Agent- and model-agnostic
- Self-hosted or managed deployment

Goose opened the worker. Buzz opened the workplace.

[9] [10] [11] [12]

What properties would neutral settlement for open agents actually require?

01

NO ISSUER

No company or foundation can create units outside the public rules.

02

OPEN ACCESS

A participant can create keys and transact without network membership approval.

03

VERIFY, DON'T TRUST

Ordinary users can run software that checks the same rules for themselves.

04

CREDIBLE SCARCITY

Supply is independently enforced—not merely promised in a document.

05

COSTLY TO REWRITE

Changing history requires real resources; checking the proof stays inexpensive.

06

VOLUNTARY CHANGE

Developers can propose rules, but participants choose what software to run.

[18]

A blockchain can be public—and still behave like a company

Many crypto networks are open-source products with a managed centre, not neutral monetary infrastructure.

MANAGED TOKEN NETWORK

- Foundation, issuer, or founding company matters
- Insiders, administrators, or large validators can have outsized influence
- Issuance or rules may change through governance or upgrades
- Bridges, stablecoin issuers, or custodians add trusted counterparties
- The network may depend on a product roadmap and leadership team

OPEN MONETARY NETWORK

- No central issuer or membership desk
- Users can hold keys and verify rules directly
- Invalid issuance is rejected by independent nodes
- History is anchored by an external resource cost
- Changing the monetary rules requires broad voluntary adoption

Ask: is this public infrastructure—or a company with a token?

[18]

The conclusion is Bitcoin.

Because its design matches the properties.

✓ **NO ISSUER**

New units follow public rules.

✓ **FINITE SUPPLY**

21 million is enforced, not promised.

✓ **INDEPENDENT NODES**

Anyone can reject invalid blocks.

✓ **VOLUNTARY ADOPTION**

Developers propose; participants decide.

✓ **PROOF OF WORK**

Costly to rewrite; cheap to verify.

✓ **OPEN ACCESS**

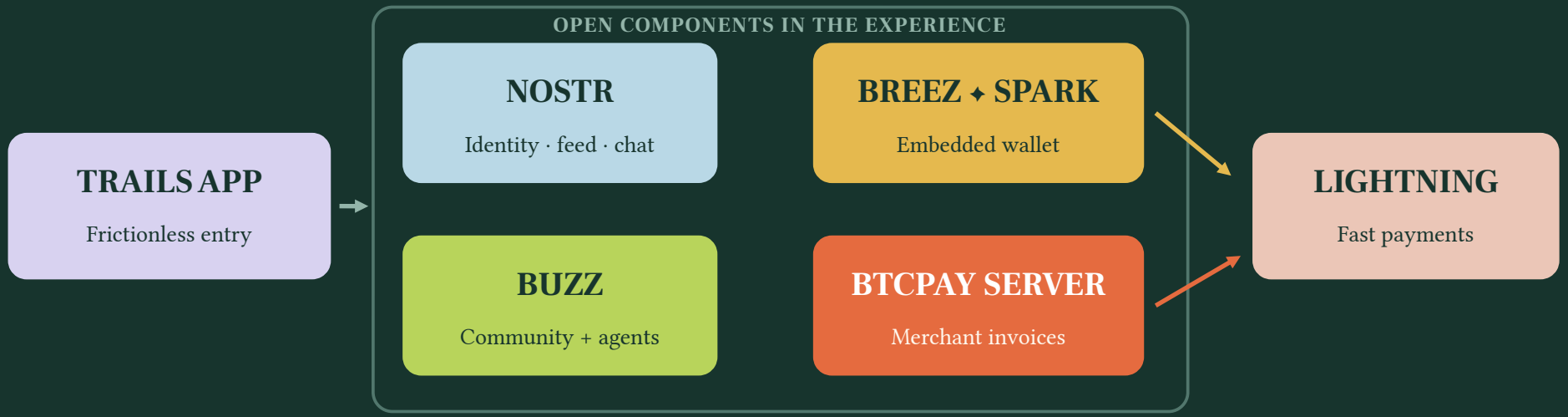
Keys first—no network account required.

THE TRADE-OFFS ARE REAL **Base-layer settlement is slower. Self-custody adds responsibility. Price is volatile. Proof of work uses energy.**

Lightning adds the fast micropayment layer; the base network remains the neutral settlement layer.

[5] [18]

We are not asking permission.
We are assembling on open rails.



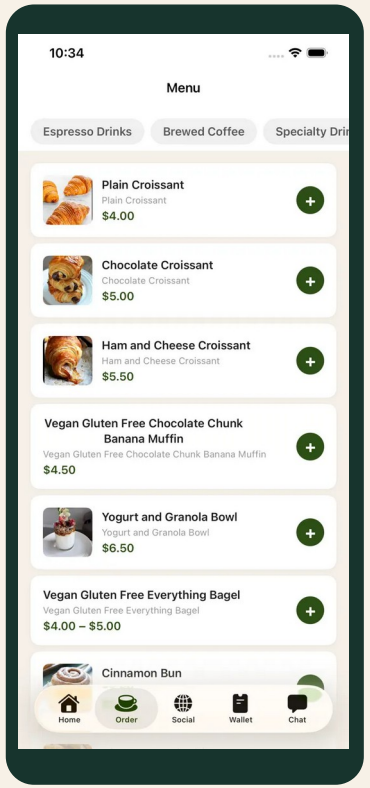
Open web ✦ Linux infrastructure underneath

Every layer is replaceable. Users can leave with their identity and value.

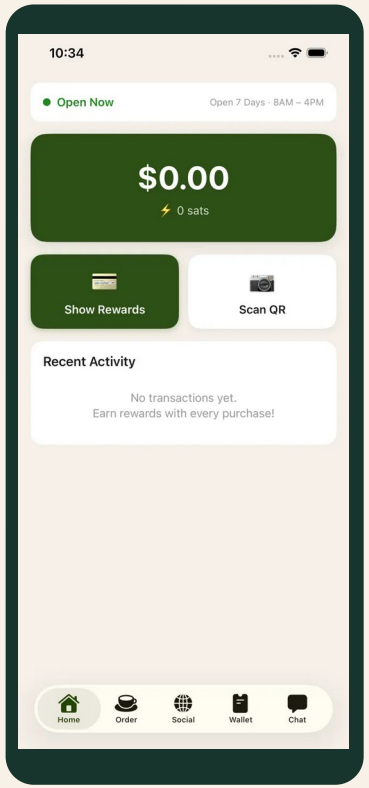
[4] [5] [11] [15] [16] [17] [19]

The product still has to earn the user

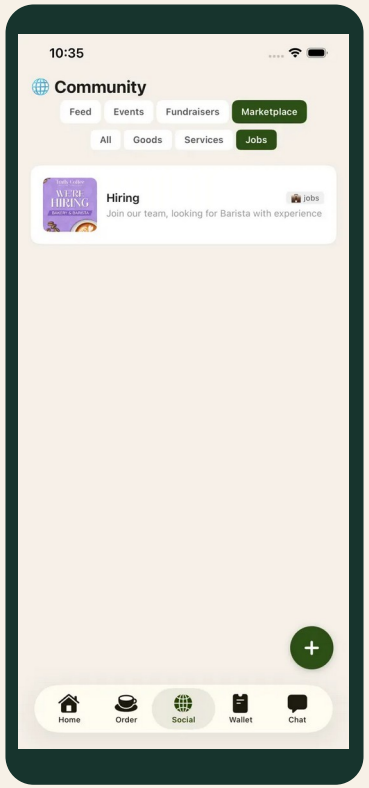
Open infrastructure is not an excuse for difficult UX. It lets us compete on experience without trapping the customer.



ORDER



EARN SATS



COMMUNITY

NOT LOYALTY POINTS

Rewards are portable sats.

- Usable with compatible wallets and merchants
- Spendable beyond Trails Coffee
- Interoperable with Lightning and Bitcoin tools
- Introduces open technology through a familiar coffee experience

Our moat must be the best experience—not captive value.

[17] [19]

THE CONCLUSION

Open source lets us inspect. Open protocols let us connect. Open networks remove permission.

No issuer. Independent verification.
Credible scarcity. Costly to rewrite.

Q&A ✦ DEMOS

Possible demos: wallet creation · sats reward · Lightning payment · Nostr identity · Buzz room

Sources

Open-source history, protocols, agent infrastructure, and payments.

[1] IBM — The story of open source at IBM

[2] Microsoft Azure — Microsoft's open-source journey

[3] Linux Foundation — State of Open Source Security

[4] nostr-protocol/nostr — protocol repository

[5] Lightning Network technical summary

[6] Coinbase Developer Platform — x402 overview

[7] Anthropic — Model Context Protocol

[8] Google Developers Blog — Agent2Agent protocol

[9] Block — Codename Goose

[10] aaif-goose/goose — repository

Sources continued

First-party project documentation and implementation evidence.

[11] Block — Introducing Buzz

[12] block/buzz — repository

[13] RFC 9110 — HTTP Semantics, 402 Payment Required

[14] x402.org — open payment standard documentation

[15] BTCPay Server — open-source, self-hosted payment processing

[16] Breez SDK on Spark — embedded Bitcoin and Lightning

[17] Trails Coffee — Why Bitcoin

[18] My First Bitcoin — open-source Bitcoin Diploma curriculum

[19] Trails Coffee first-party app and site source reviewed for implementation details

[20] Meta — Account Integrity and Authentic Identity

[21] X Help Center — Automation rules

Implementation claims were checked against the current Trails iOS and website codebases, including Nostr, Breez/Spark, Lightning, BTCPay Server, and wallet/rewards flows.