SecuWeb

Food Use Case

Gertjan De Mulder – Ben De Meester

Food Supply Chains

Issues within existing food supply chains (FSCs)

- Waste
- Fraud & counterfeit
- Delayed recall and interventions

Because existing FSC rely on manual processes, which are

- error-prone
- not tamper proof
- make authenticity hard to achieve

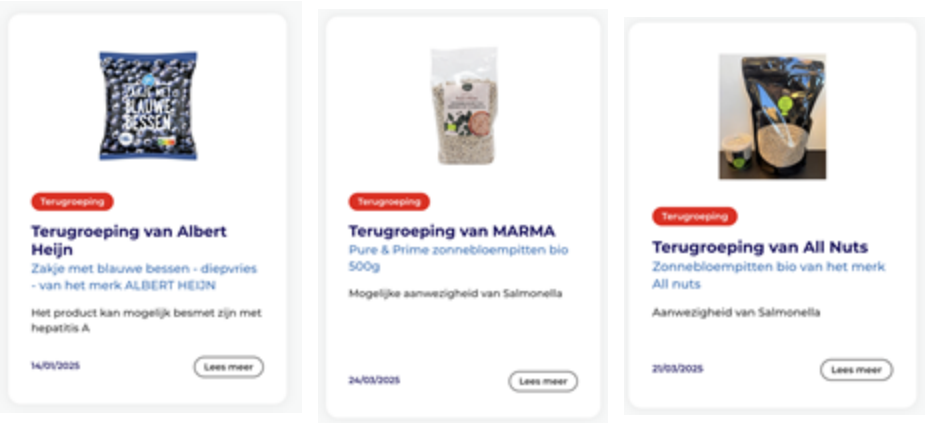
- Food Supply Chains
- **How can blockchain improve Food Supply Chains?**
- Open Challenges for BC-based solutions
- A hybrid approach is needed
- Solution
- Use Case: Food Recall
- Architecture
- Demo

How can the of blockchain improve FSCs?

Blockchains (BCs) can enable a **transparent** end-to-end FSC providing **traceability** of products, which increases **sustainability** and **operational efficiency** [1], because of:

- The ability to automatize (otherwise manual) processes (how: smart contracts)
- More efficient food recall processes
- The ability to optimize SCs to reduce food waste

Moreover, BC-based SCs can have a substantial impact on **health**, due to **improved quality assurance**, facilitating the prevention of distributing bad/unsafe products [4].



https://favv-afsca.be/nl/producten

- Food Supply Chains
- How can blockchain improve Food Supply Chains
- **Open Challenges for BC-based solutions**
- A hybrid approach is needed
- Solution
- Use Case: Food Recall
- Architecture
- Demo

Open challenges for BC-based solutions

Public/permissionless BCs are slow (e.g., ETH transactions typically take ± 23sec)

Generally, the more data on-chain, the slower the system becomes and, the higher the risk of privacy threats.

BCs shouldn’t store all data, so where to should it be stored?

- Food Supply Chains
- How can blockchain improve Food Supply Chains
- Open Challenges for BC-based solutions
- **A hybrid approach is needed**
- Solution
- Use Case: Food Recall
- Architecture
- Demo

A hybrid approach is needed

A hybrid approach that allows to balance the tradeoff between on and- off-chain data in a way that not only harnesses the features provided by a blockchain, but also extends them with additional features such as **data sovereignty** and **interoperability**.

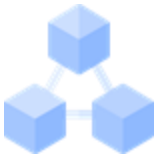
- Food Supply Chains
- How can blockchain improve Food Supply Chains
- Open Challenges for BC-based solutions
- A hybrid approach is needed
- **Solution**
- Use Case: Food Recall
- Architecture
- Demo

Solution

A hybrid approach that allows to balance the tradeoff between on and- off-chain data in a way that not only harnesses the features provided by a blockchain, but also extends them with additional features such as **data sovereignty** and **interoperability**.

BC

- Decentralized
- **Immutability**
- **Verifiability**



+

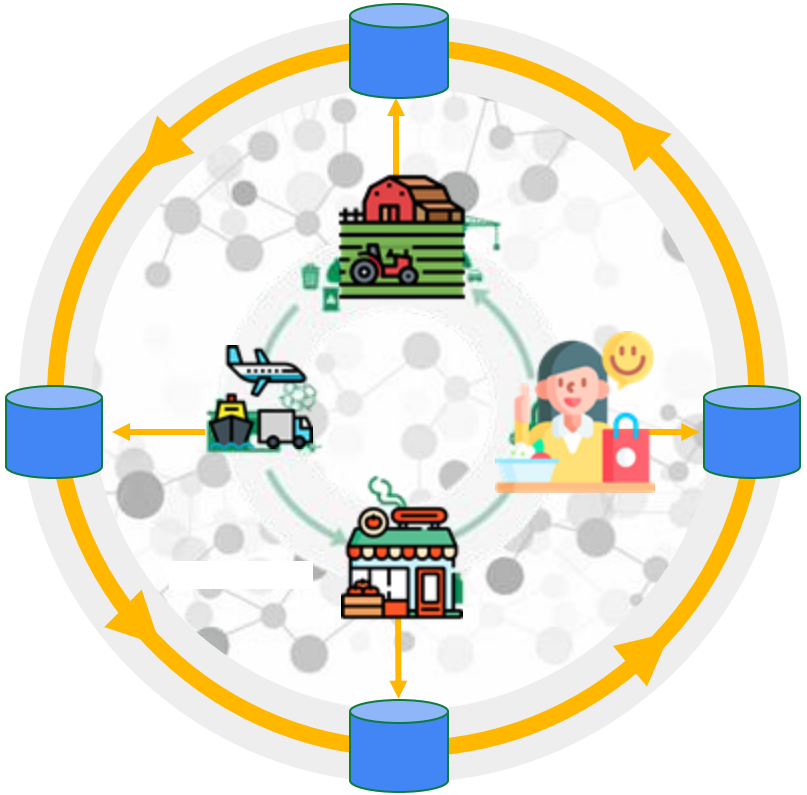
Solid

- Decentralized
- **Data sovereignty**
- **Interoperability**

Food Supply Chains (FSCs) need data flows

Information needs to flow between actors in a way that that increases **traceability**, **transparency**, and **efficiency** in FSCs.

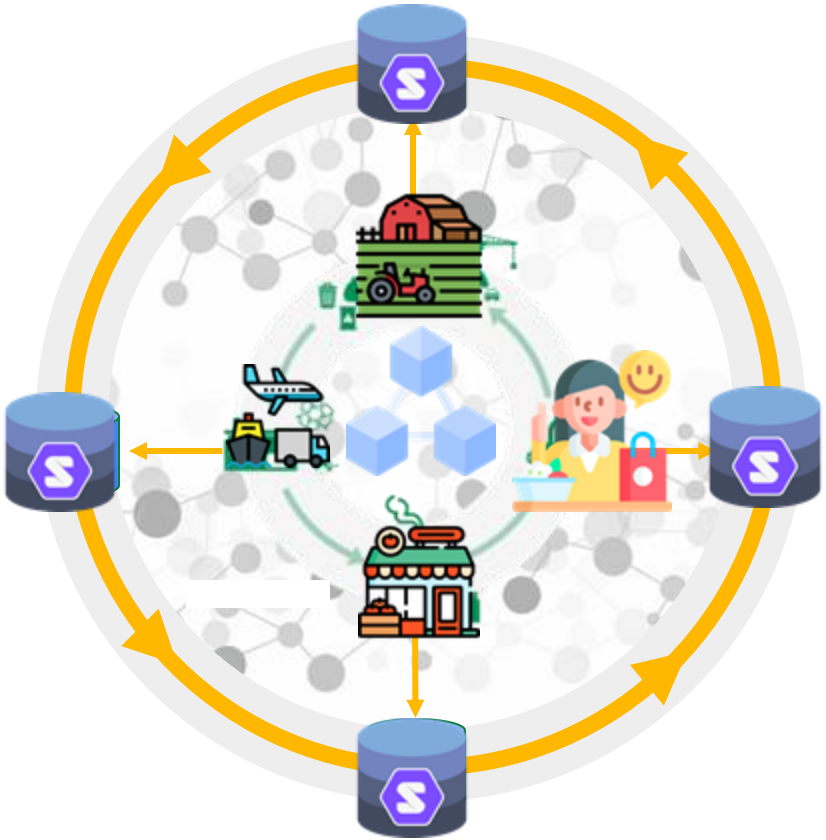
While the actors can **control** their data, and **trust** the data of others.



Food Supply Chains (FSCs) need data flows

Information needs to flow between actors in a way that that increases **traceability**, **transparency**, and **efficiency** in FSCs.

While the actors can **control** their data and **trust** the data of others.



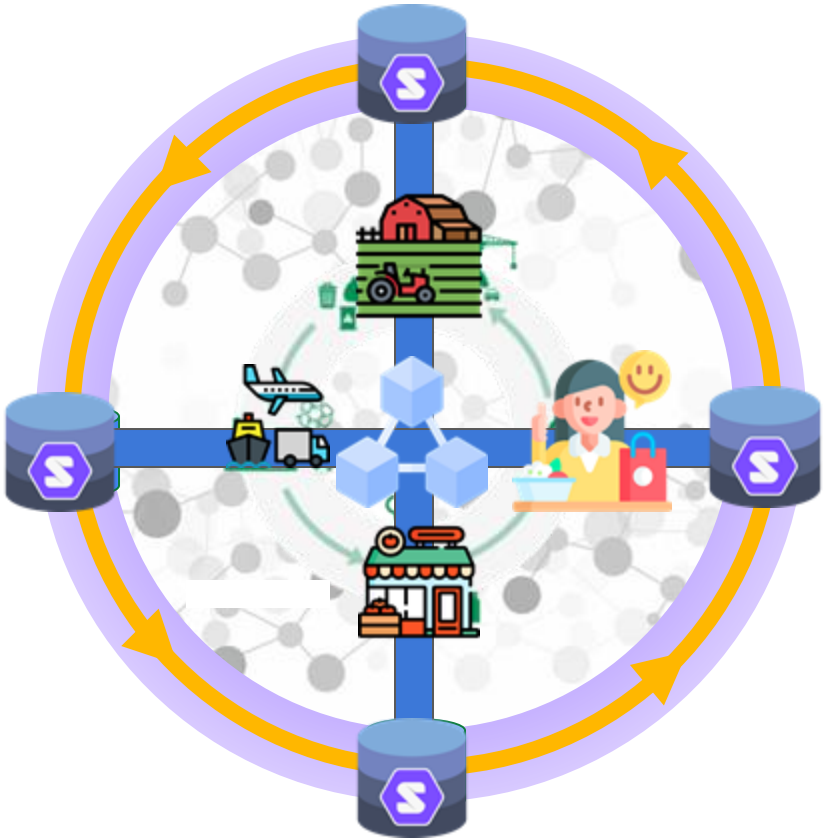
Food Supply Chains (FSCs) need decentralized data flows

Information needs to flow between actors in a way that that increases **traceability**, **transparency**, and **efficiency** in FSCs.

While the actors can **control** their data and **trust** the data of others.

By storing their data on a personal data store, or Solid Pod.

By verifying the **authenticity** and **integrity** of that data.

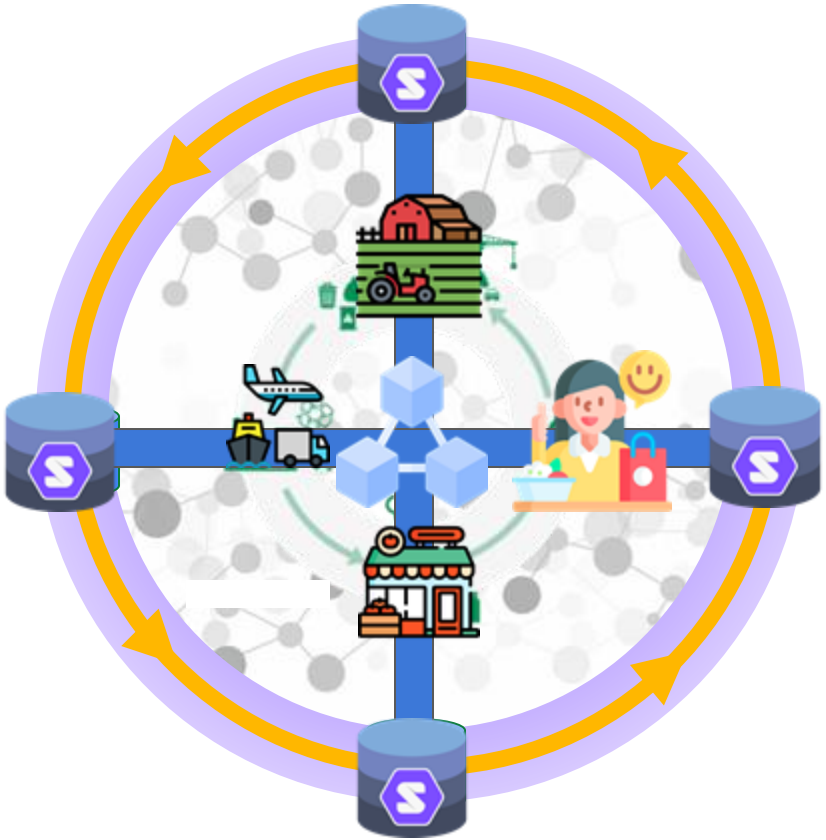


- Food Supply Chains
- How can blockchain improve Food Supply Chains
- Open Challenges for BC-based solutions
- A hybrid approach is needed
- Solution
- **Use Case: Food Recall**
- Architecture
- Demo

Use case: Solid Food Recall

Prelims: Alice & Store Z are “Solid-enabled”.

1. Alice buys products X and Y at store Z. At check-out, Alice’s Solid Pod is updated with the newly bought groceries.
2. An actor somewhere along the SC warns about product Y being contaminated with Salmonella.
3. Alice immediately receives a warning that her recently bought product Y is likely contaminated.



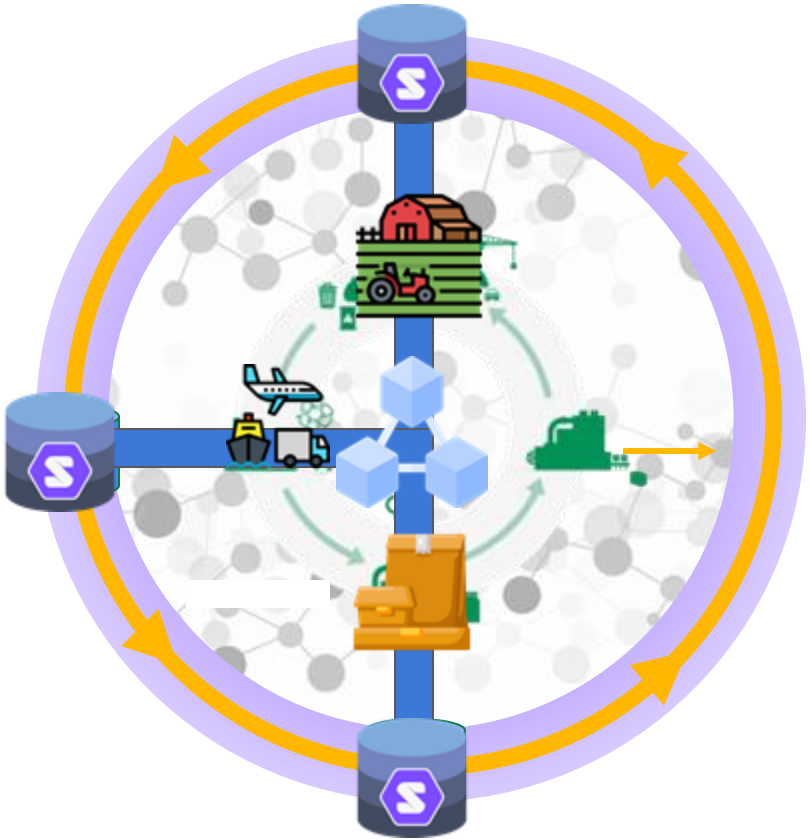
Use case: Product Shipping

A **Farmer** ships a product to a **Packager** using a **Transporter**.

The **Farmer** shares the product details with the **Packager**, but not with the **Transporter**.

The **Transporter** creates verifiable shipment records that are only readable by the actors involved (i.e., **Farmer** and **Packager**).

The **Packager** can read and verify the authenticity and data integrity of the received products.



- Food Supply Chains
- How can blockchain improve Food Supply Chains
- Open Challenges for BC-based solutions
- A hybrid approach is needed
- Solution
- Use Case: Food Recall
- **Architecture**
- Demo

Architecture

WebIDs to identify actors within the Solid ecosystem.

DIDs to identify actors and resources on the BC.

An actor's Solid Pod serves as off-chain data store that is solely under the control of the actor.

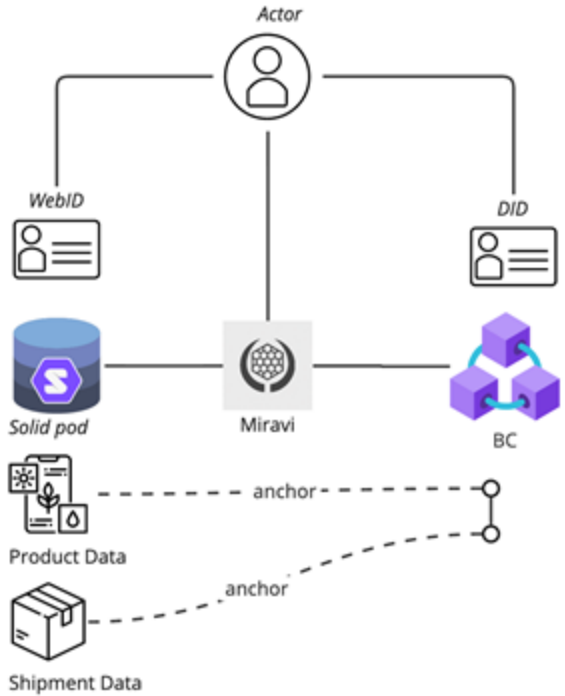
Data is encapsulated as a Verifiable Credential, which is stored on the actor's Solid Pod and anchored on the blockchain.

Miravi allows actors to

1) Query the decentrally stored data.

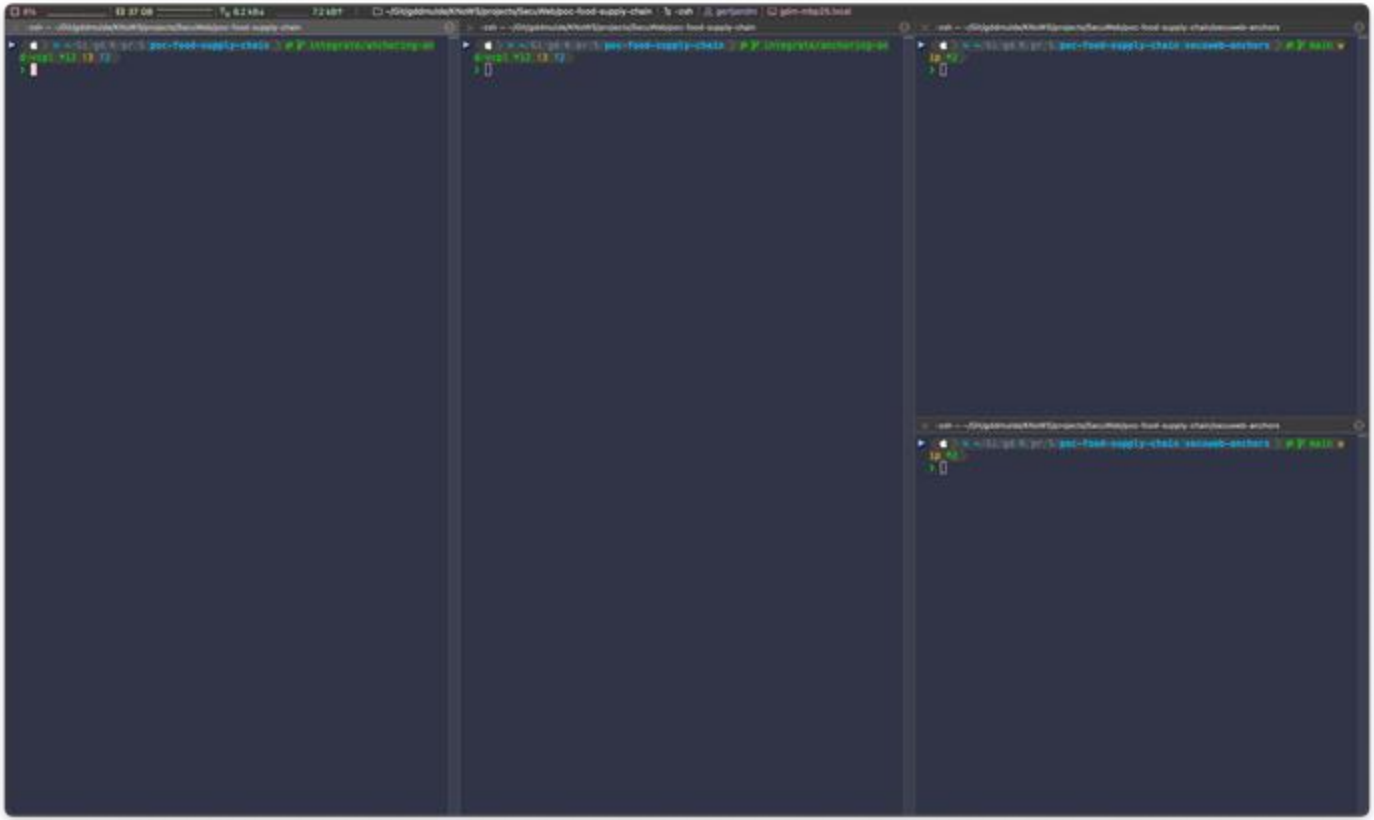
2) Verify the data's signature.

3) Verify the on-chain hash of the data.



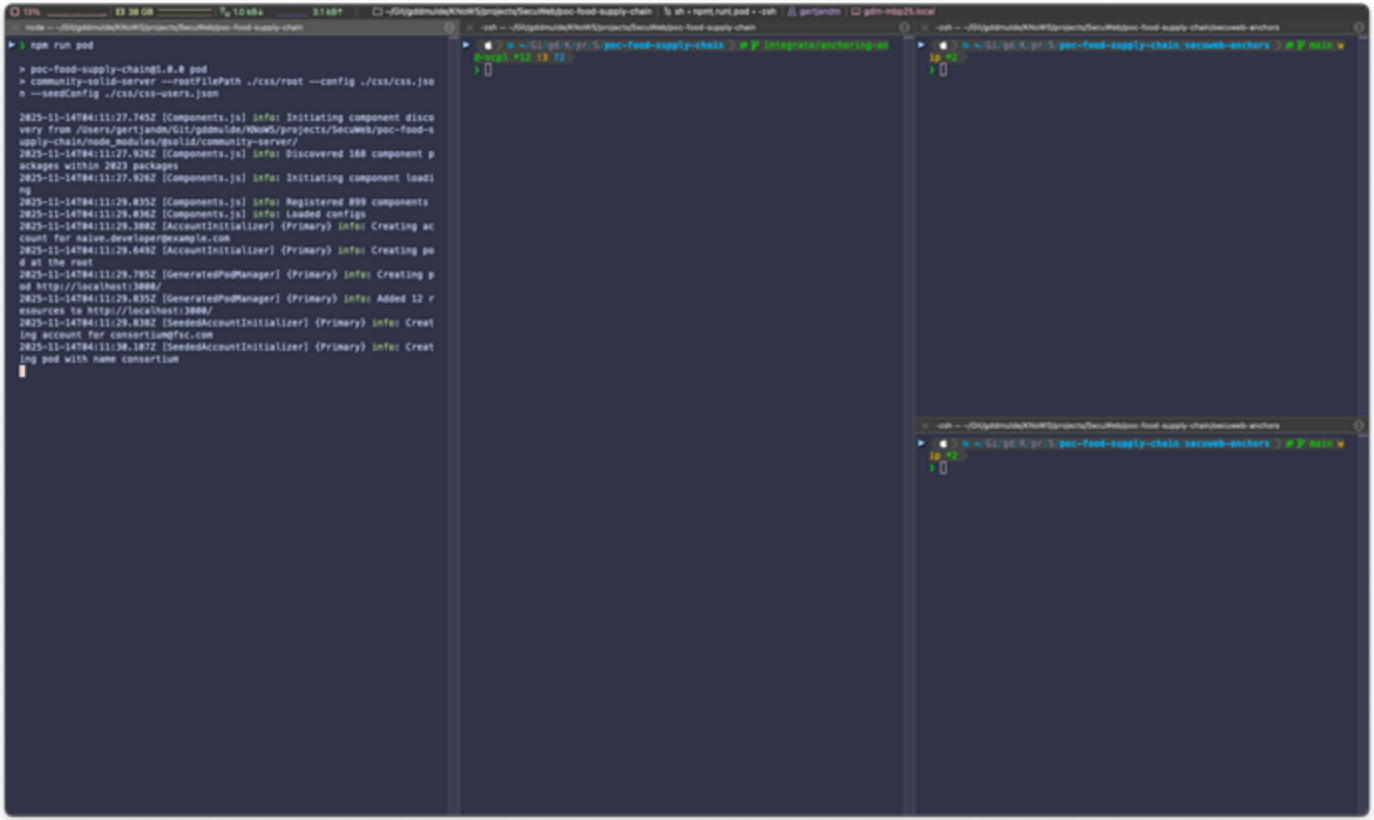
- Food Supply Chains
- How can blockchain improve Food Supply Chains
- Open Challenges for BC-based solutions
- A hybrid approach is needed
- Solution
- Use Case: Food Recall
- Architecture
- **Demo**

Demo setup

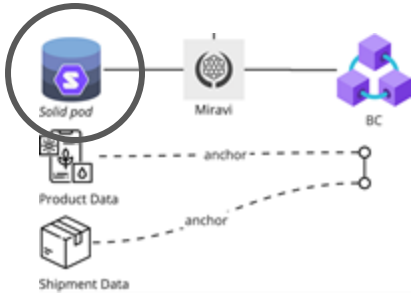


1. Set up Solid Pods
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.

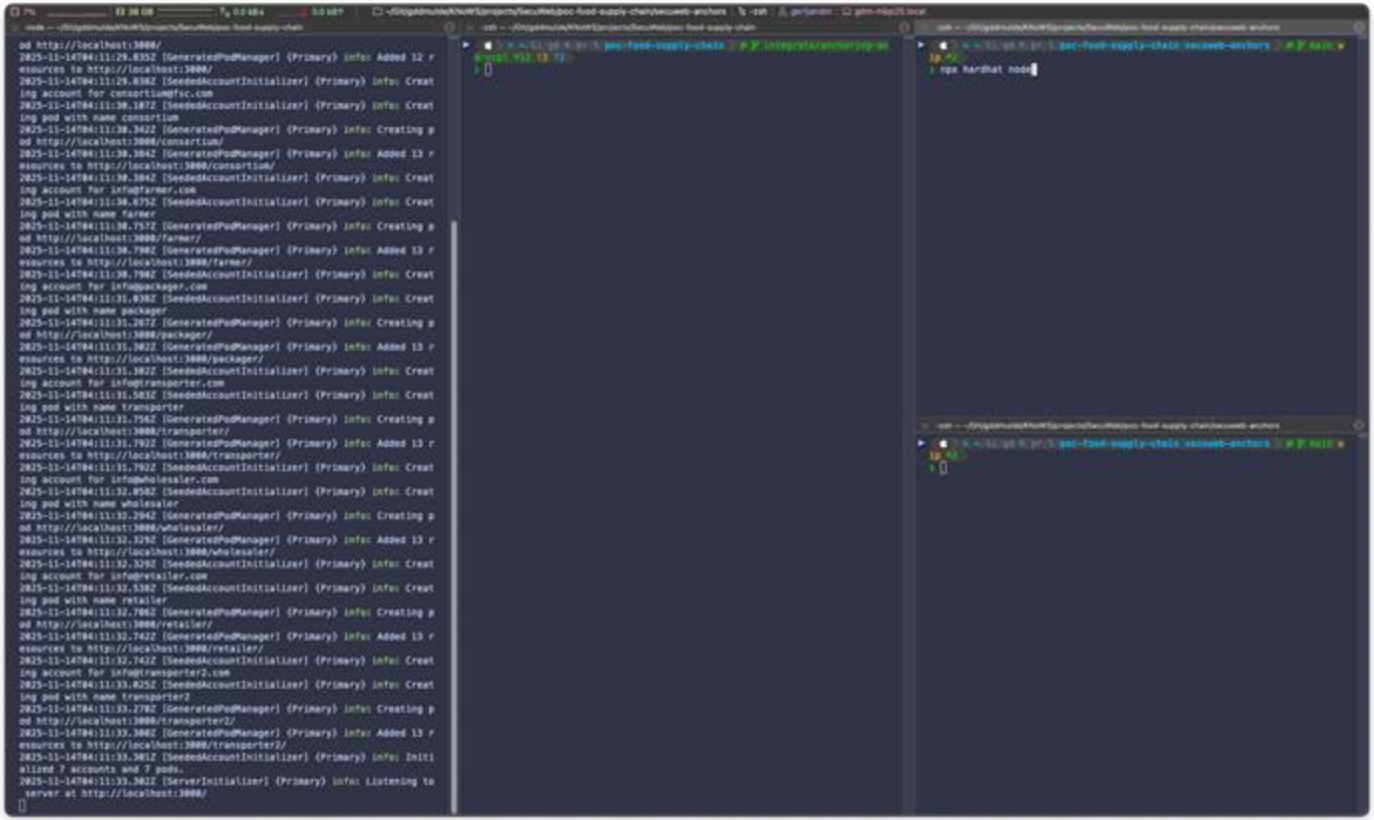
Demo setup



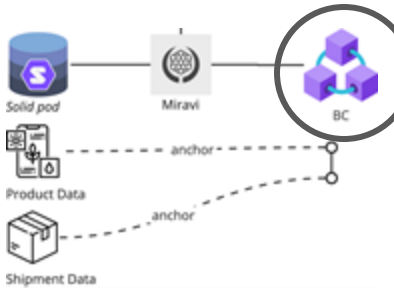
1. **Set up Solid Pods**
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



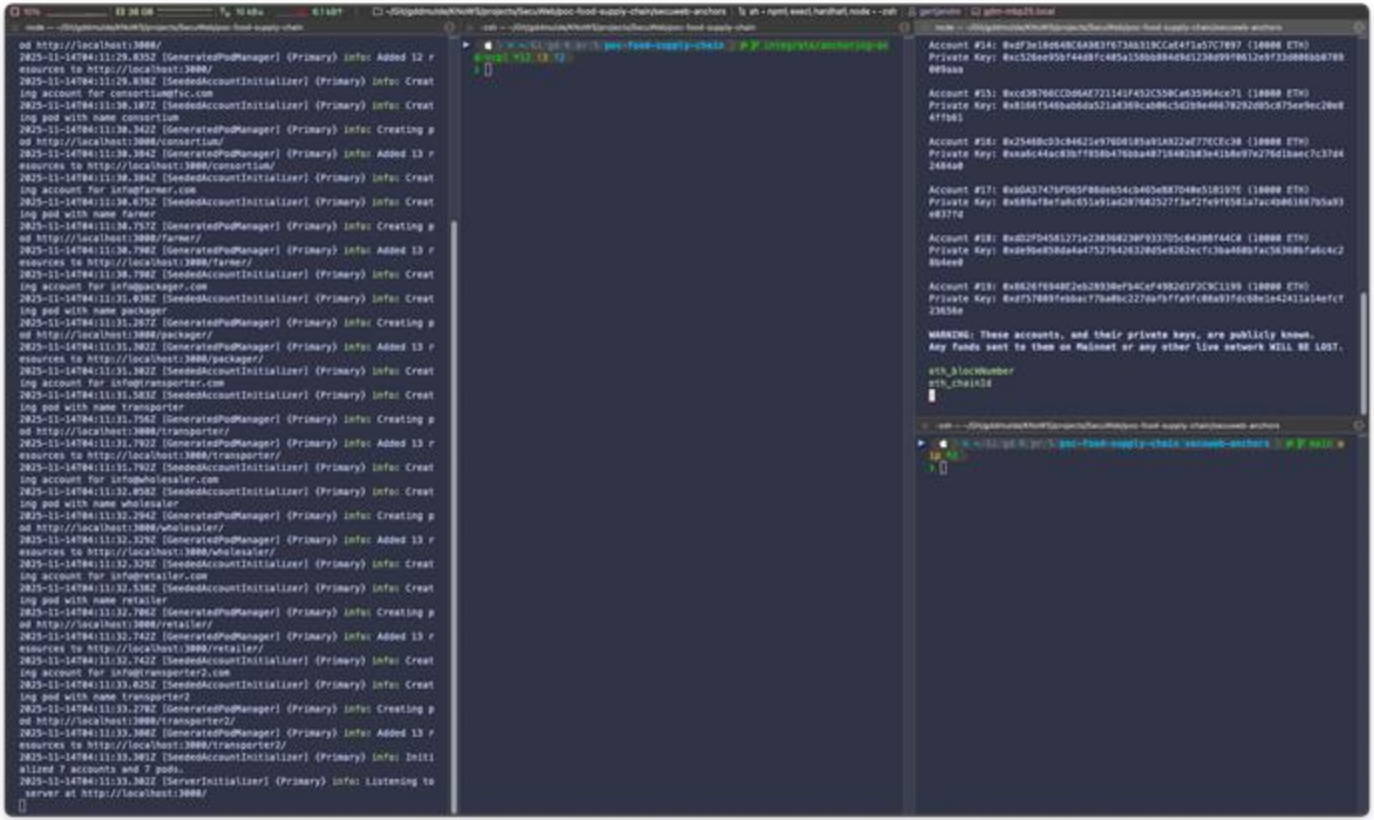
Demo setup



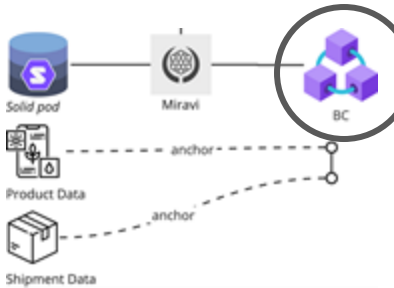
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



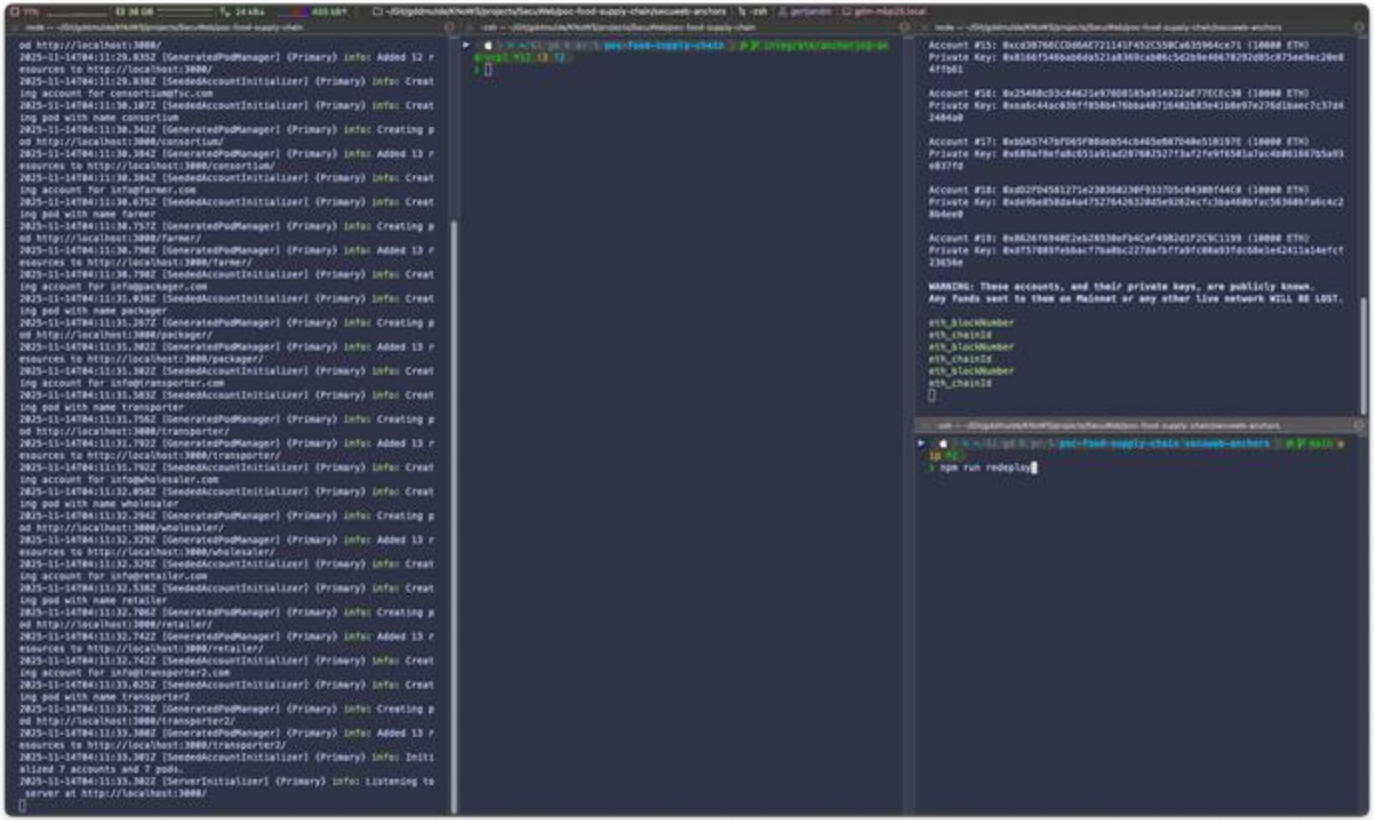
Demo setup



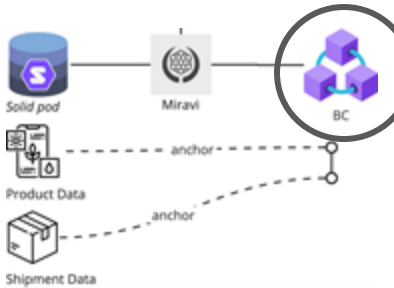
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



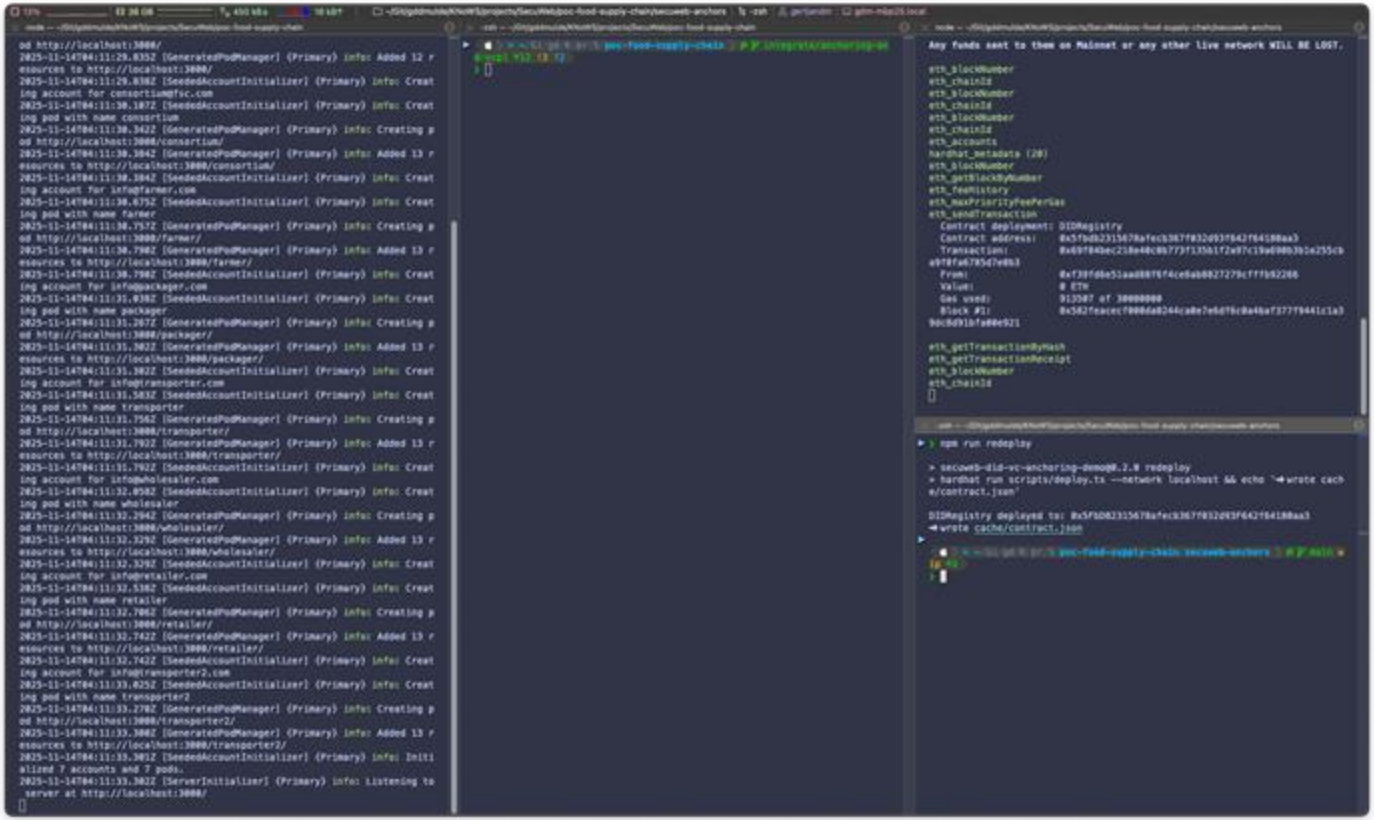
Demo setup



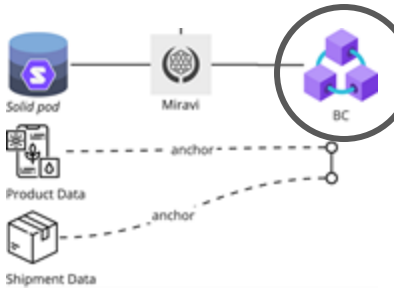
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



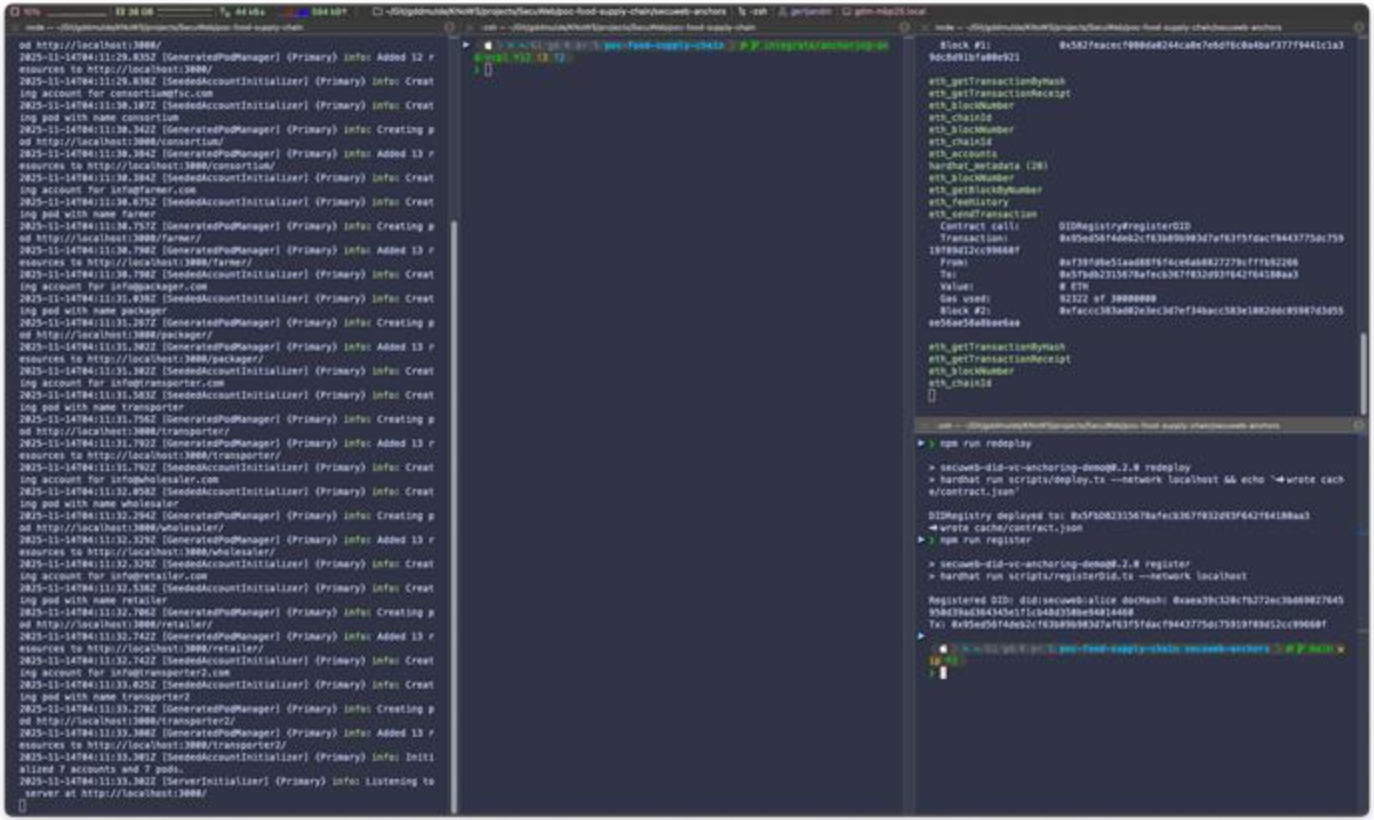
Demo setup



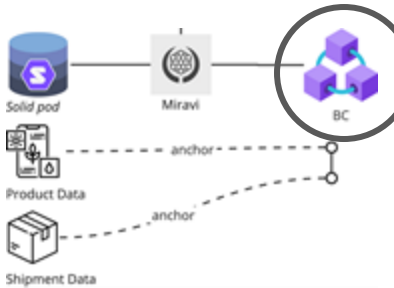
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



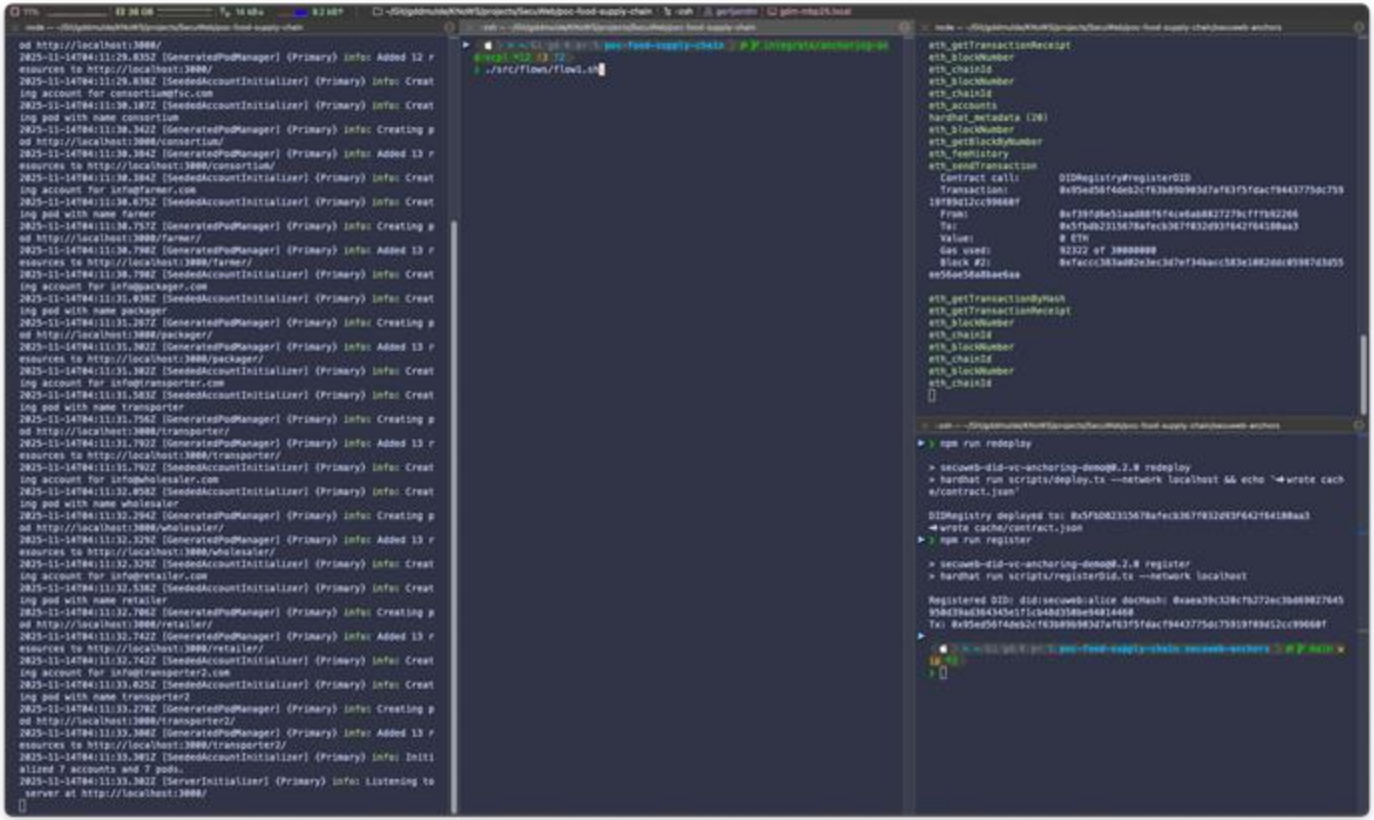
Demo setup



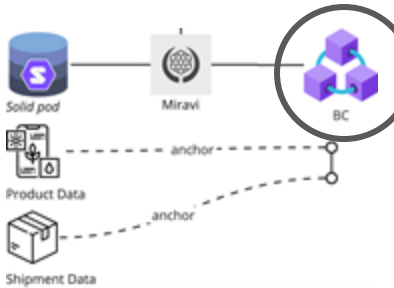
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



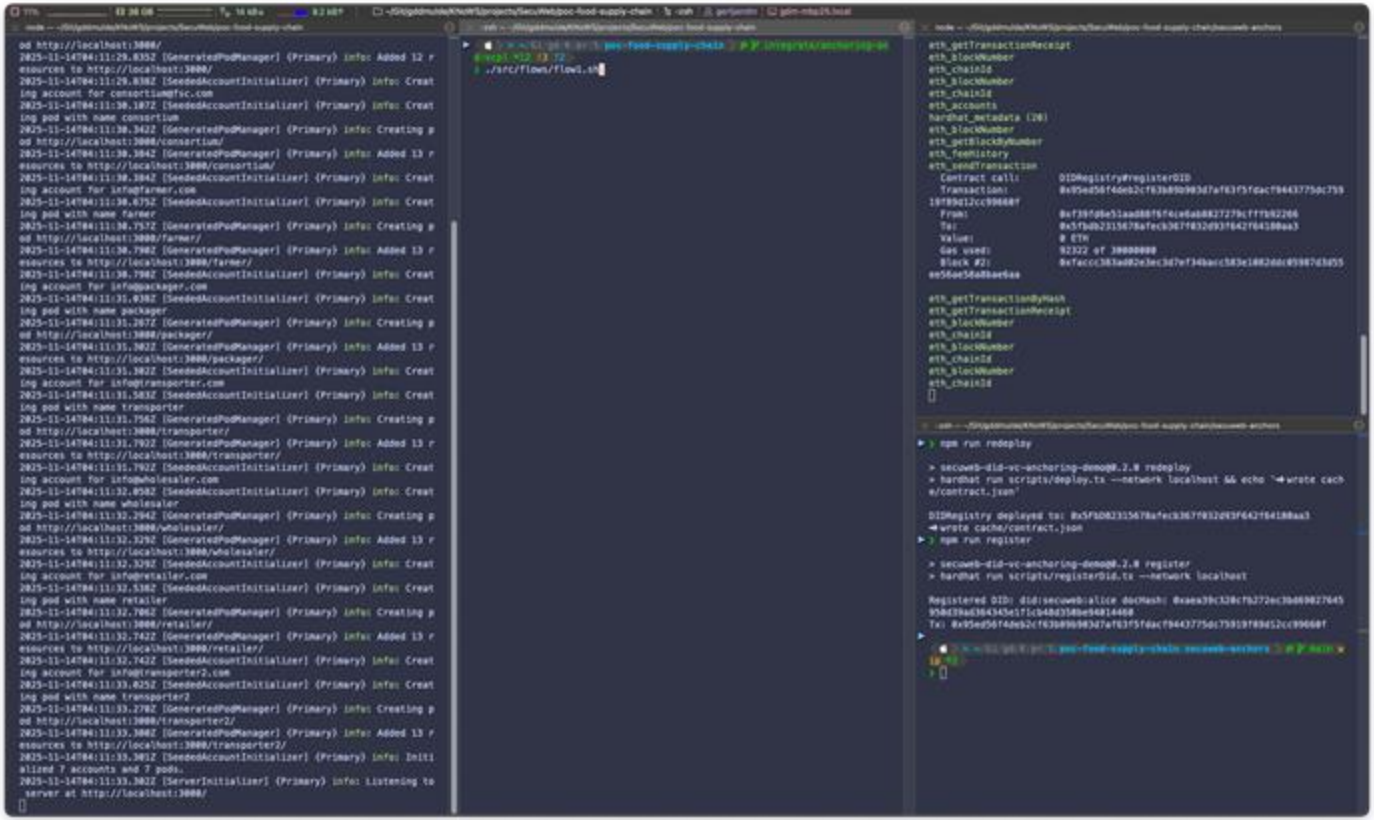
Demo setup



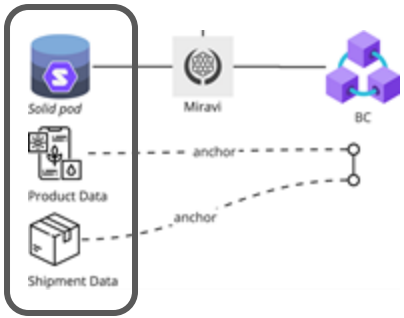
1. Set up Solid Pods
2. **Set up blockchain**
3. Store data as Verifiable Credential on the Solid Pods.
4. Anchor every Verifiable Credential on the blockchain.



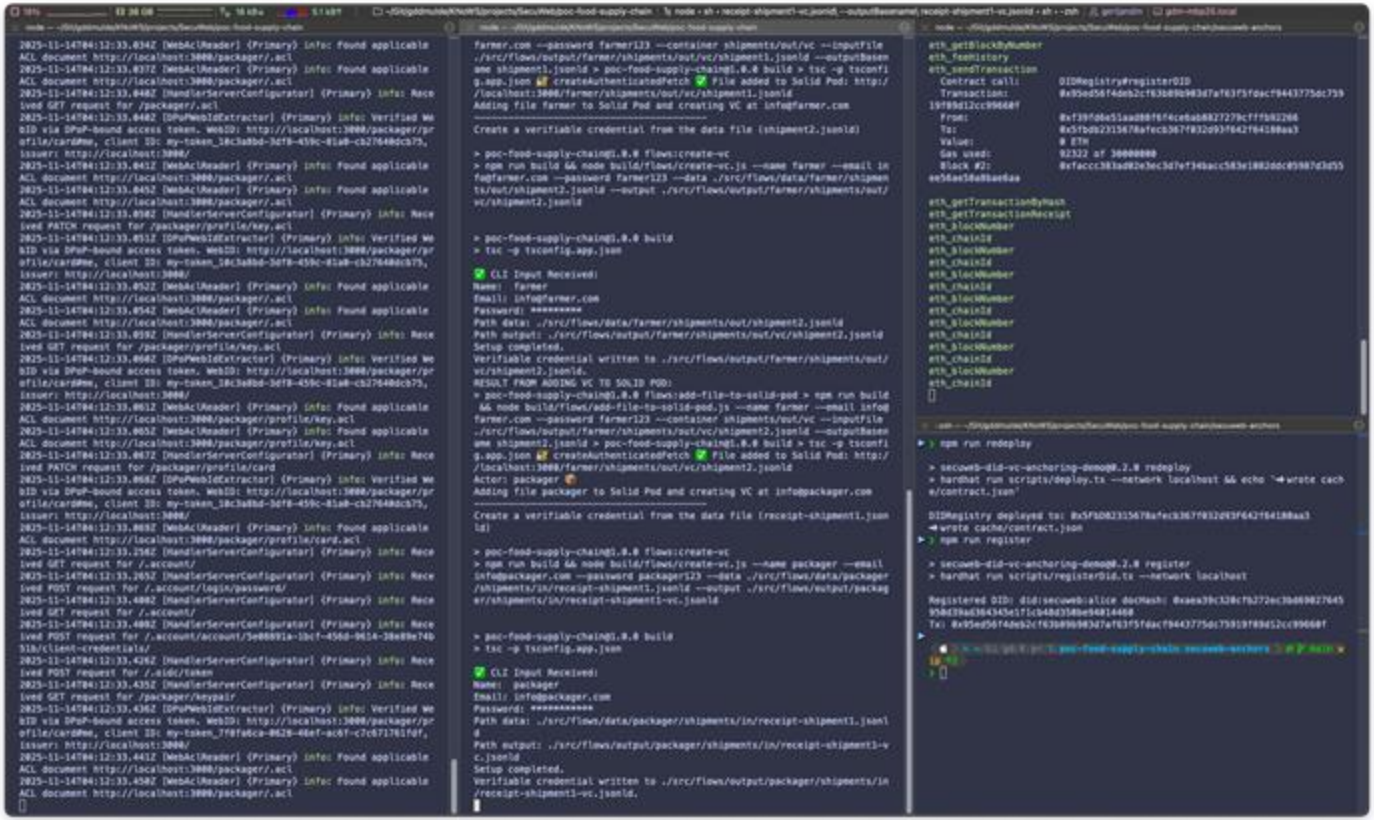
Demo setup



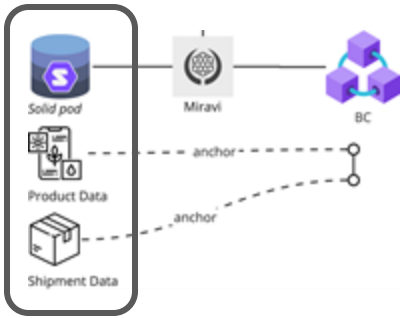
1. Set up Solid Pods
2. Set up blockchain
3. **Store data as Verifiable Credential on the Solid Pods.**
4. Anchor every Verifiable Credential on the blockchain.



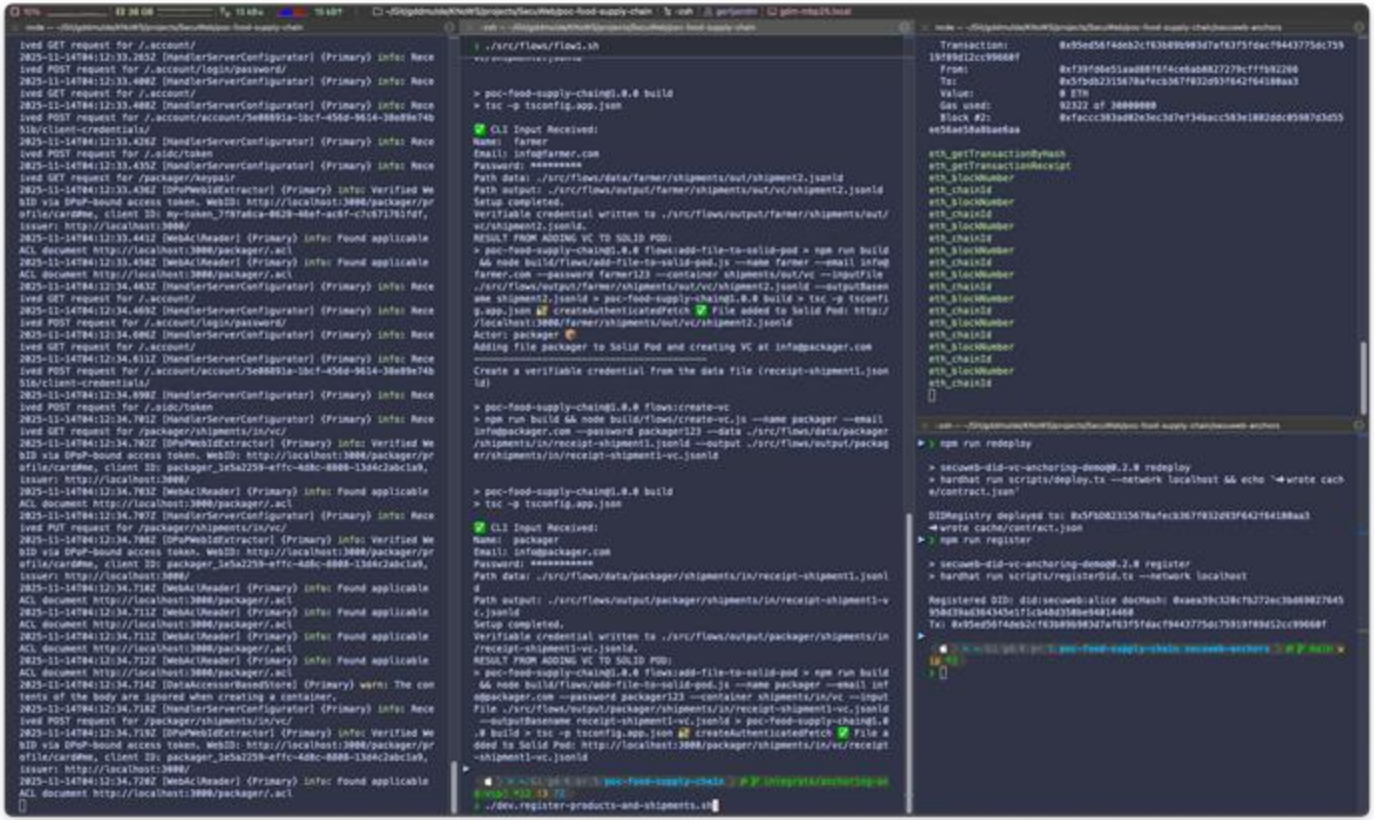
Demo setup



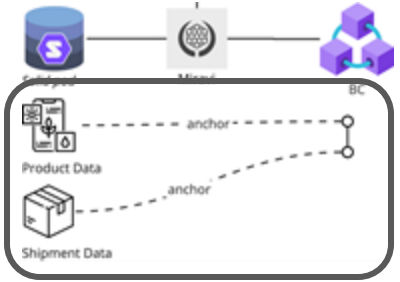
1. Set up Solid Pods
2. Set up blockchain
3. **Store data as Verifiable Credential on the Solid Pods.**
4. Anchor every Verifiable Credential on the blockchain.



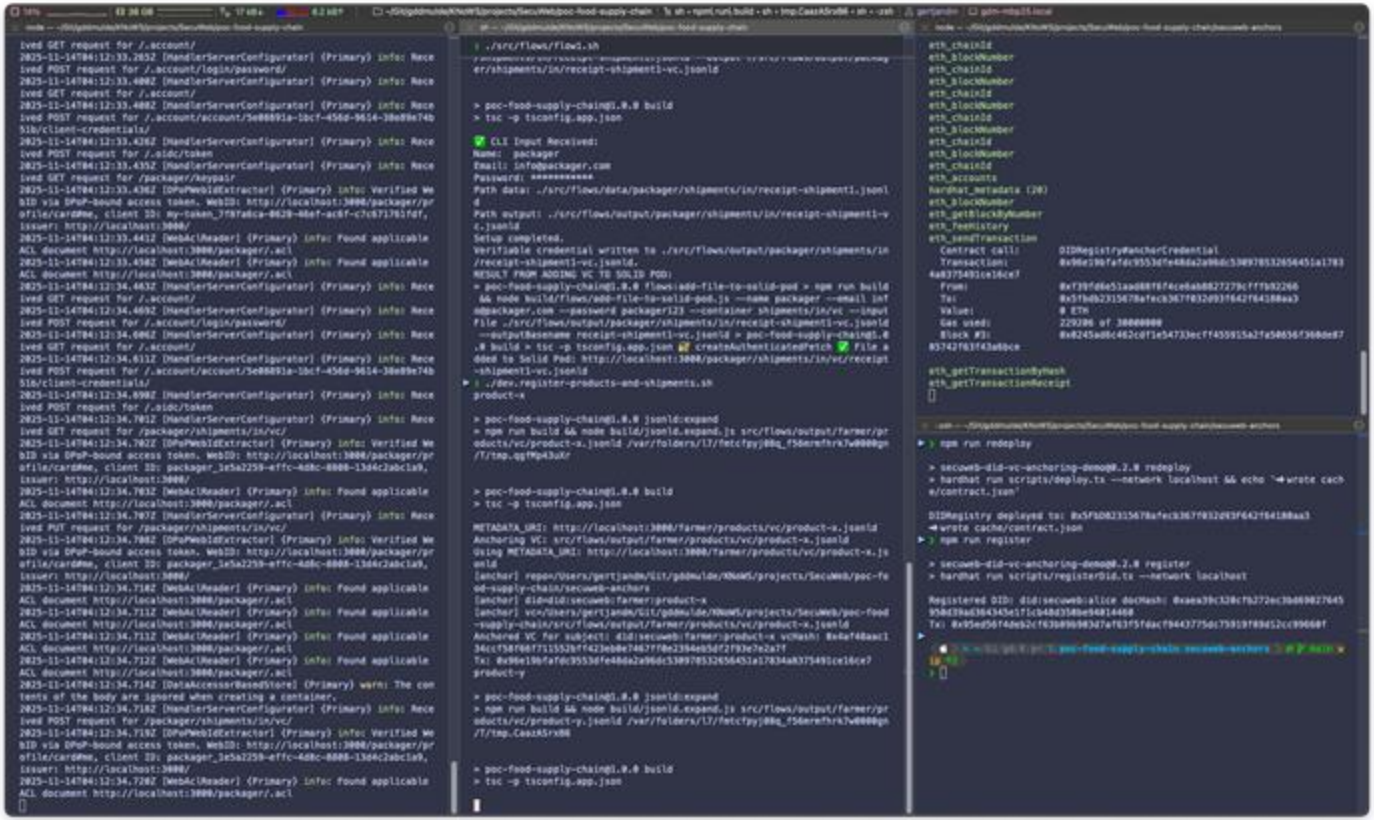
Demo setup



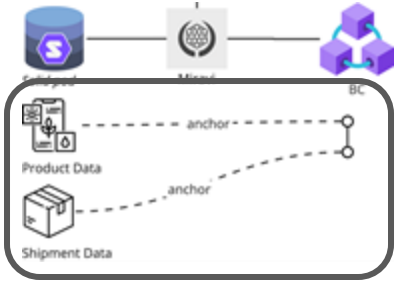
1. Set up Solid Pods
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. **Anchor every Verifiable Credential on the blockchain.**



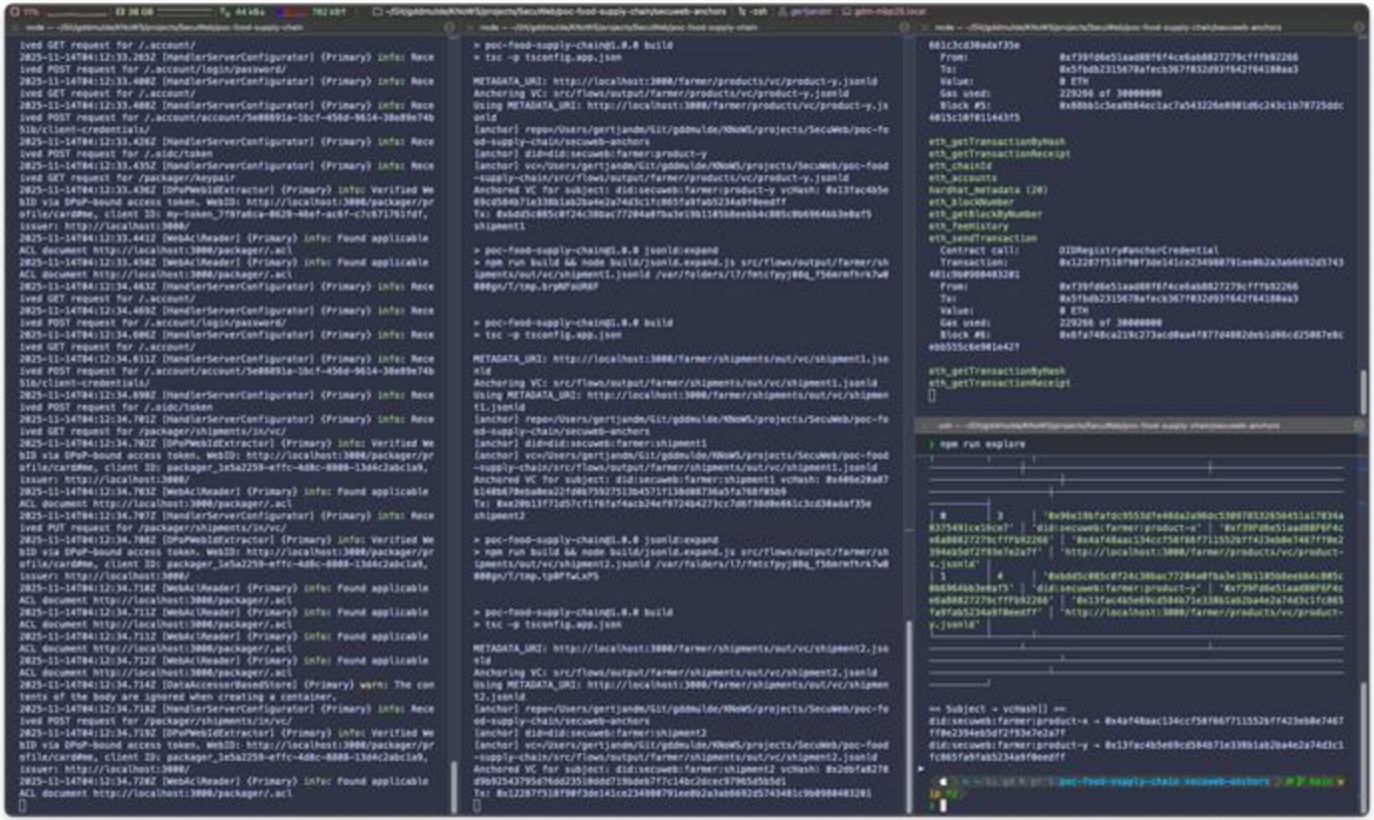
Demo setup



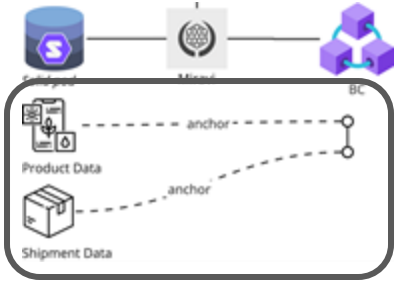
1. Set up Solid Pods
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. **Anchor every Verifiable Credential on the blockchain.**



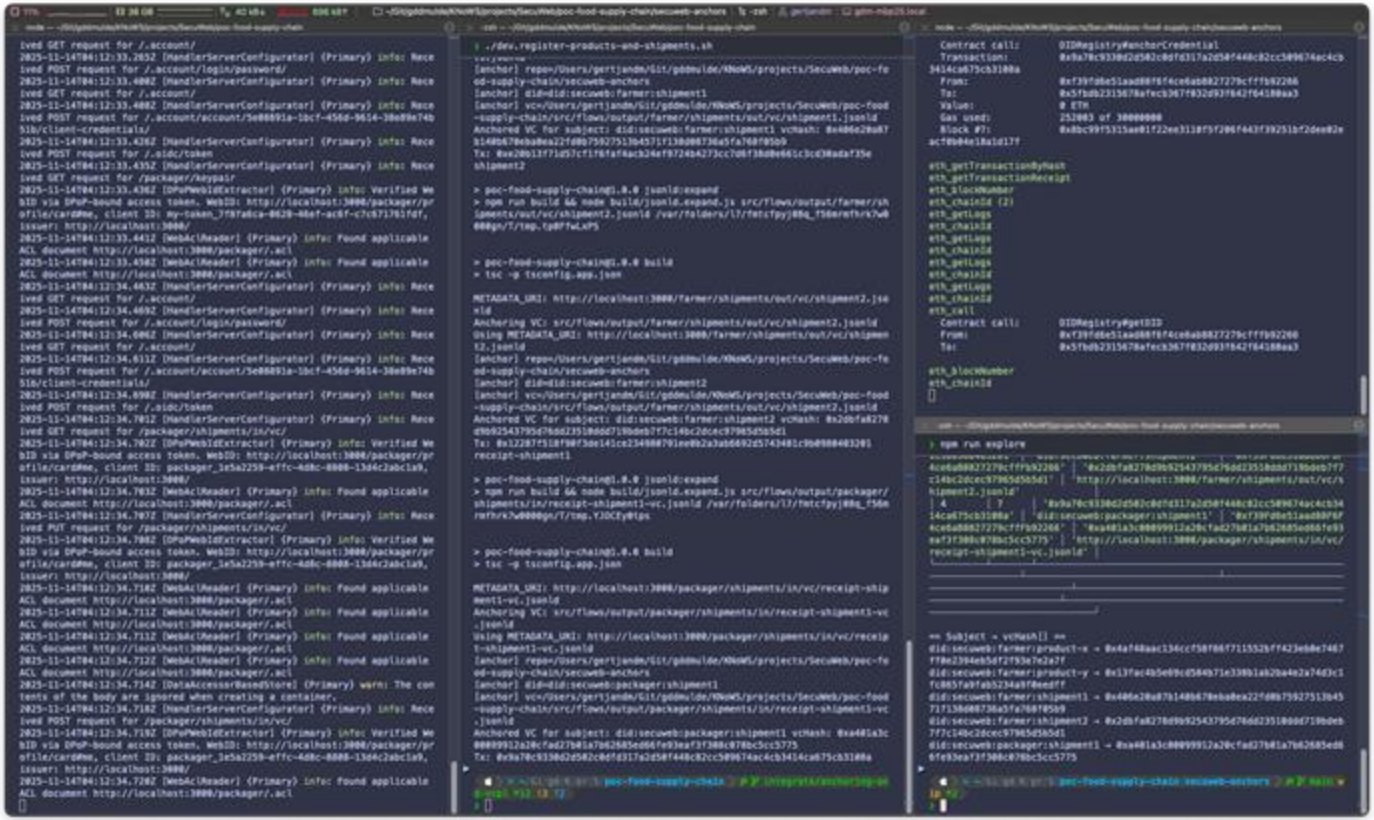
Demo setup



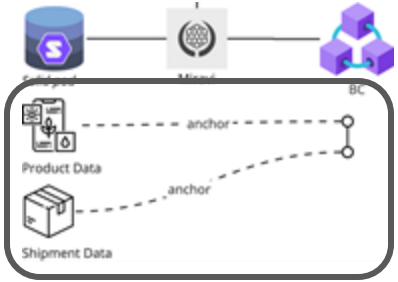
1. Set up Solid Pods
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. **Anchor every Verifiable Credential on the blockchain.**



Demo setup



1. Set up Solid Pods
2. Set up blockchain
3. Store data as Verifiable Credential on the Solid Pods.
4. **Anchor every Verifiable Credential on the blockchain.**

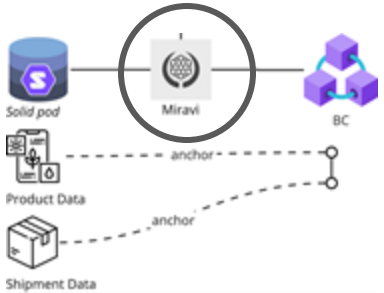


Demo setup: Miravi

```
> miravi-a-linked-data-viewer@1.0.0 dev
> vite

  VITE v4.5.13  ready in 115 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h to show help
```



Demo

Use case: Product Shipping

- Farmer has multiple products
- Farmer ships products to Packager, using a Transporter

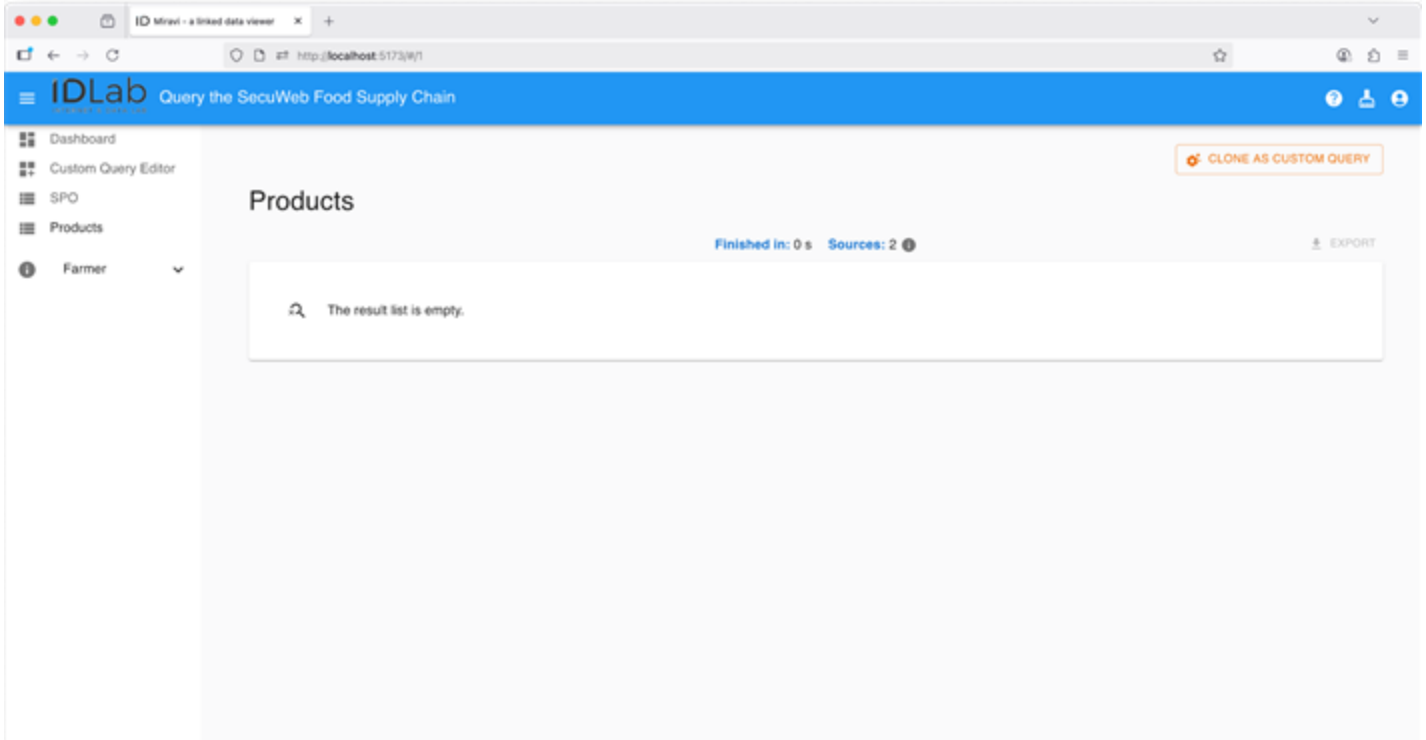
Features

Secure: Actors control which data to share with whom.

- Only the appropriate actors can access information about products and shipments. For example
  - Transporter can read shipment information, but cannot read product details
  - Packager can read both shipment information and product details

Verifiable: Actors can trust the data they are “seeing”

Unauthenticated



Actors control which data to share with whom.

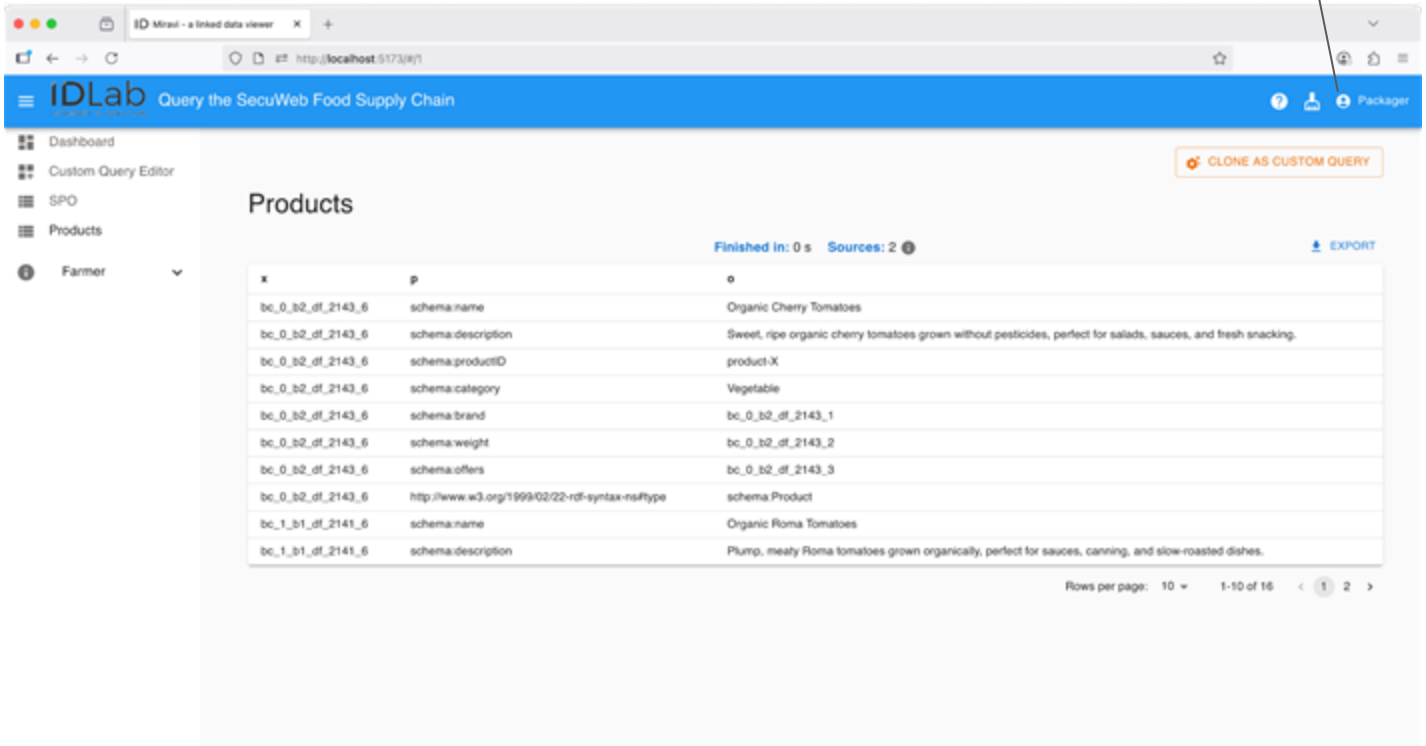
In this example, the Farmer decided that its product data is not publicly readable. Instead, only the actors involved in a business process (e.g., shipping a product) should be able to read the product data.

⬆️ Control

Authenticated as **Packager**

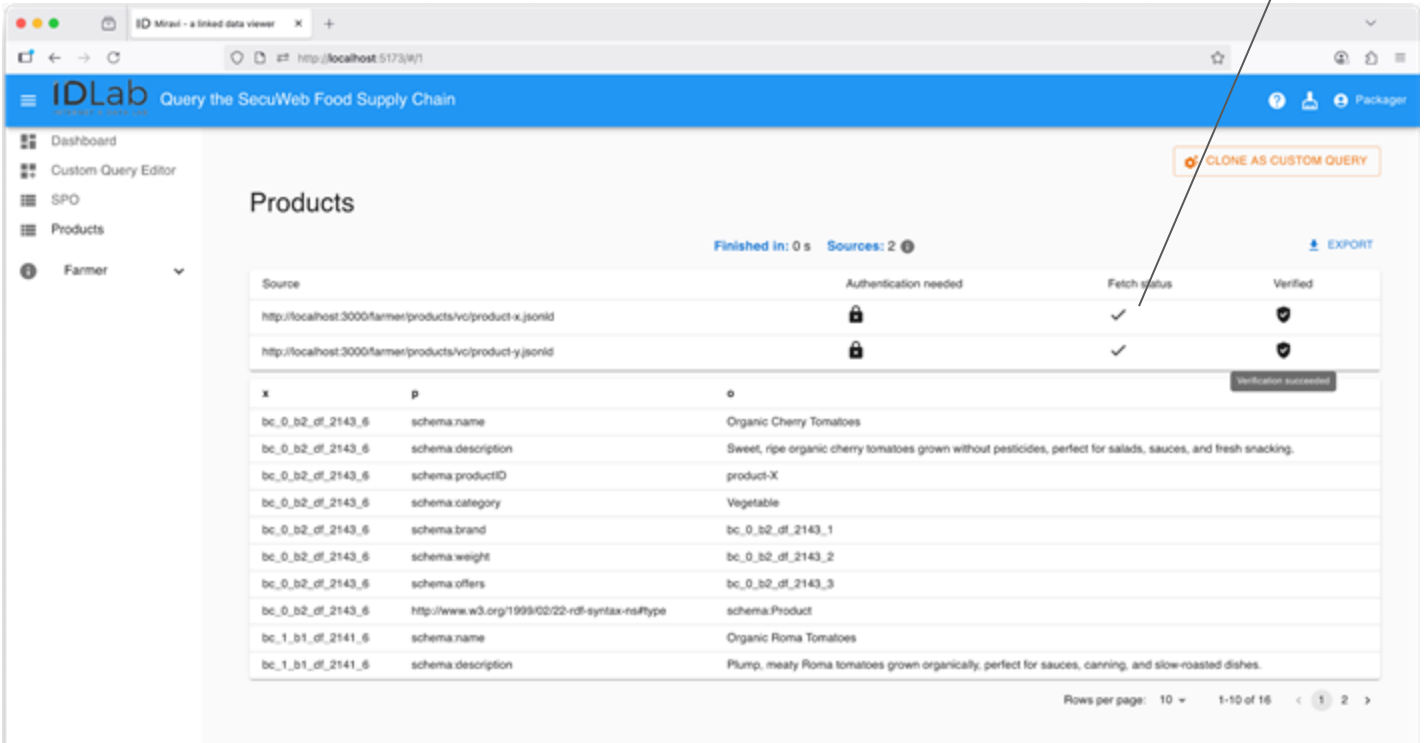
View: Products

Authenticated as **Packager**



Authenticated as **Packager**

View: Products



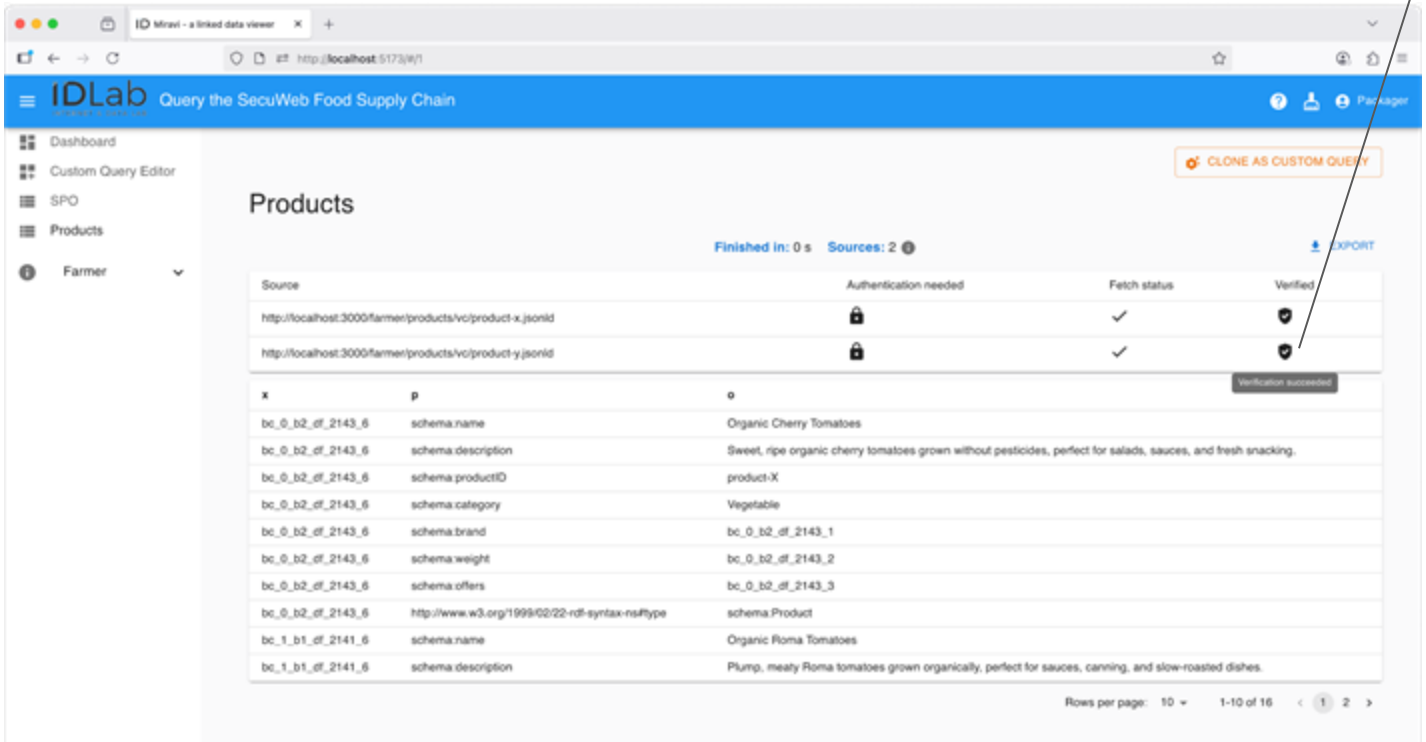
Access control
Packager has read access to product details

The Packager receives the Farmer's produces. Therefore, Packager can read the product data.

Control

Authenticated as **Packager**

View: Products



Verifiable product data

Actors can verify the authenticity and integrity of the data they have access to.

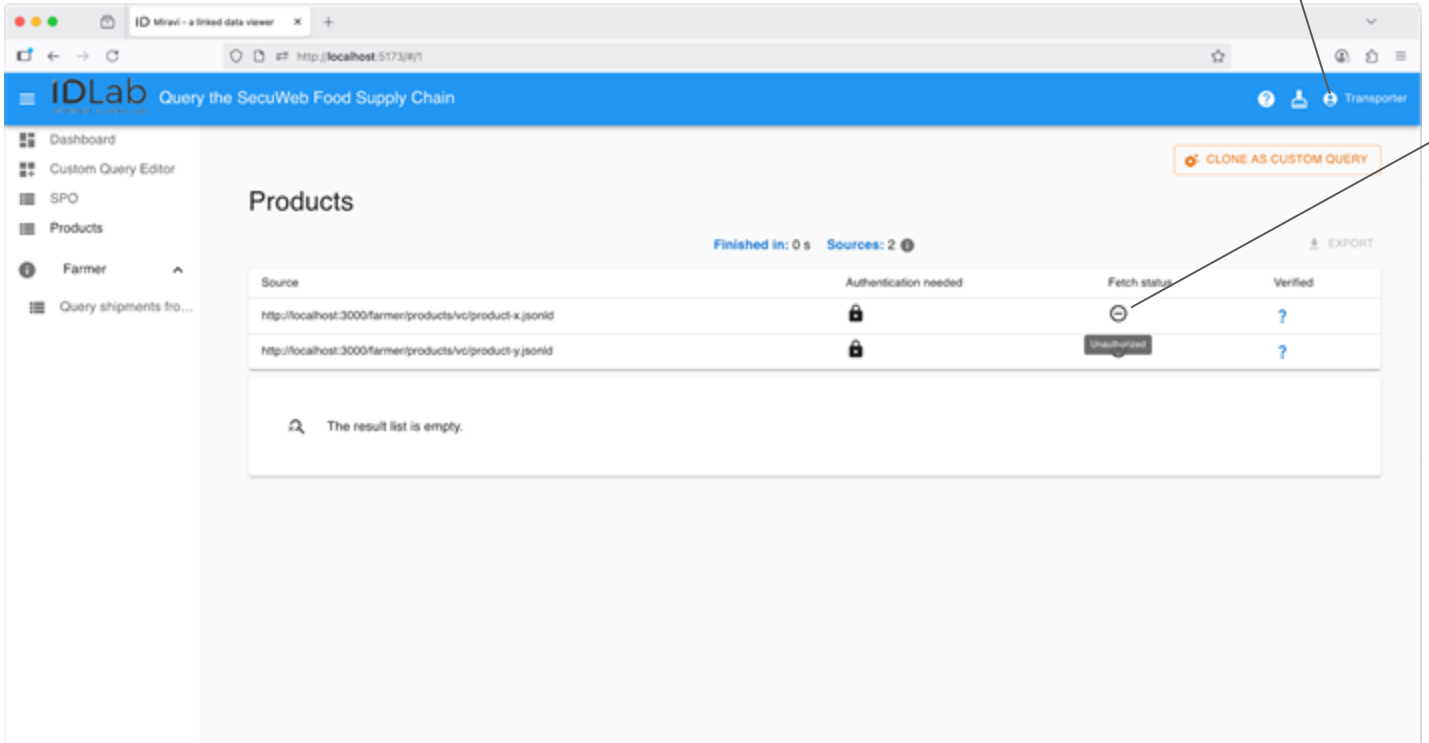
Trust

Authenticated as **Transporter**

View: Products

Authenticated as **Transporter**

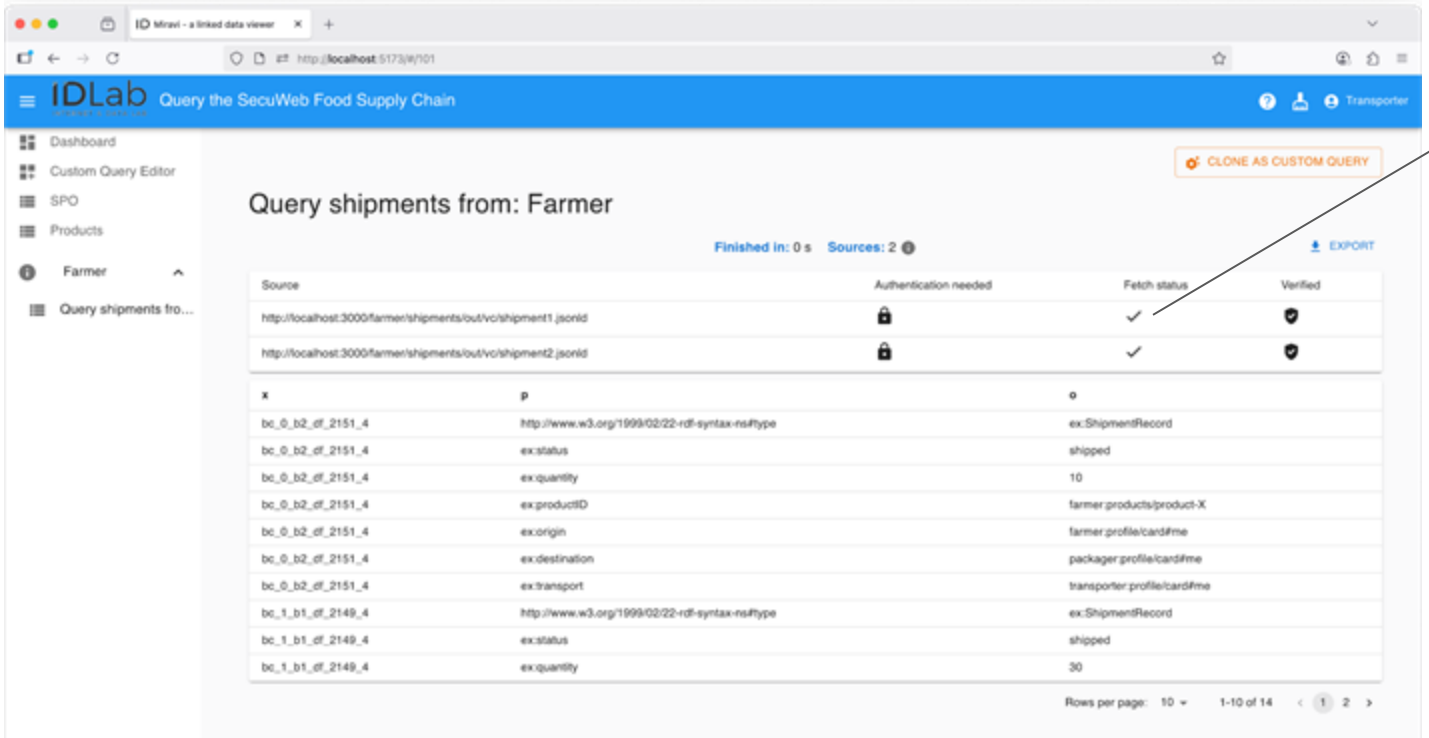
Access control
Transporter has no read access to product details



Authenticated as **Transporter**

View: Shipments

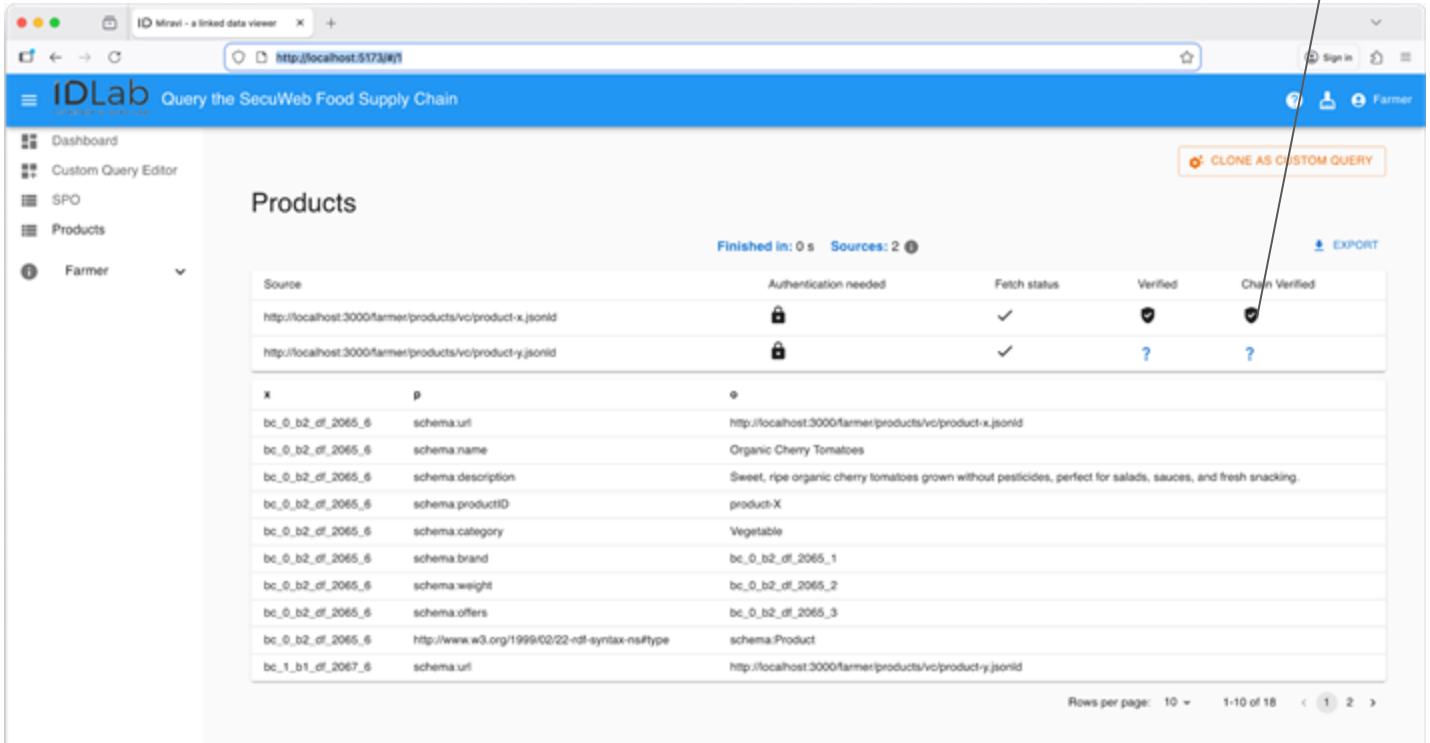
Access control
Transporter has read access to shipments



Authenticated as **Packager**

View: Products (with on-chain verification)

On-chain verification of product data



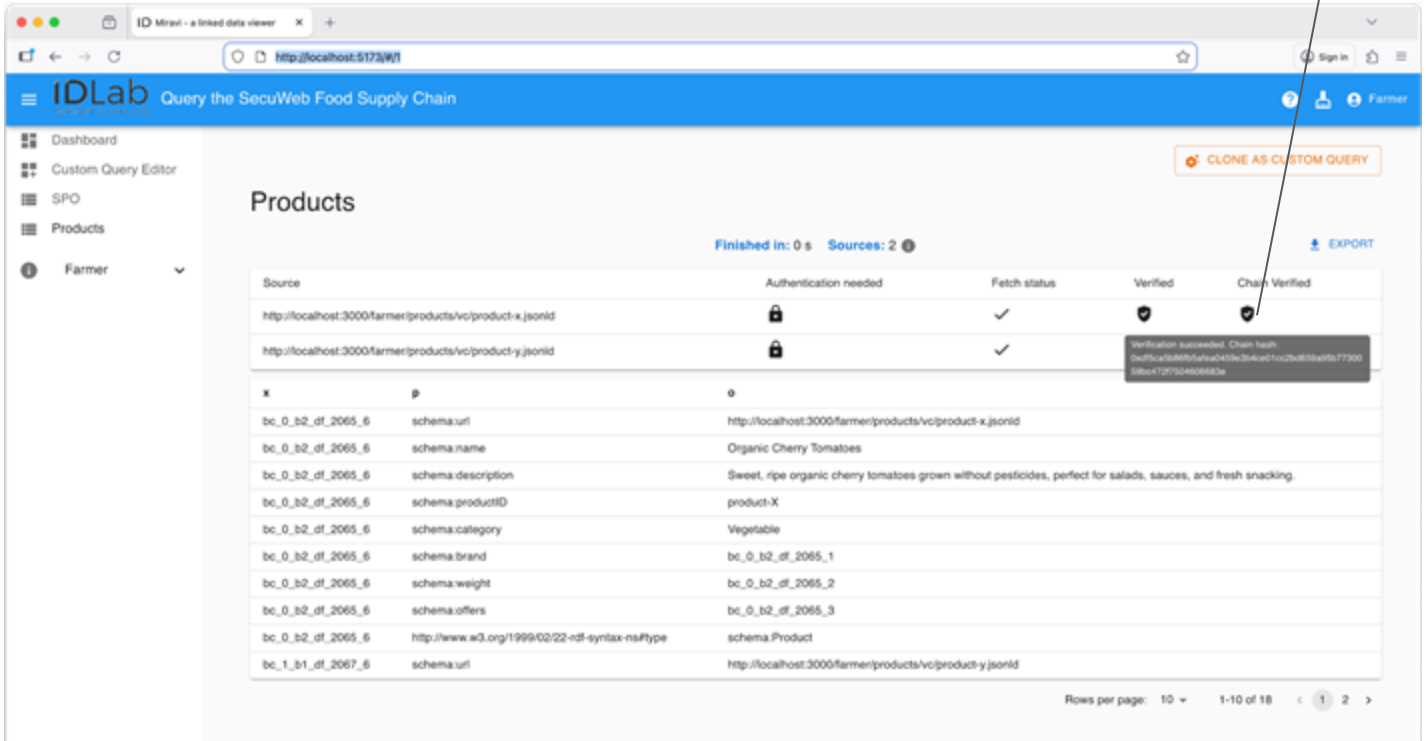
Actors can verify the on-chain hash of the anchored VC.

- Data Integrity
- Trust

Authenticated as **Packager**

View: Products (with on-chain verification)

On-chain verification of product data



Actors can verify the on-chain hash of the anchored VC.

- ⬆ Data Integrity
- ⬆ Trust

Next steps

Continue development to implement the overarching Food Recall use case.

Evaluation

- Performance by benchmarking different on- vs off-chain storage configurations.
- Security (using STRIDE)
- Privacy (using LINDDUN)

STRIDE: https://owasp.org/www-community/Threat_Modeling_Process
LINDDUN: https://linddun.org/

Questions?