Decentralized computing and personal data stores

Nicolas Seydoux, PhD

Developer at Inrupt, Solid community member

Introductory use-case: Reasoning in "smart" homes

Typical smart home use case

- Remote control for connected devices
- Data gathering, **analytics and reporting**
- **Scenario-based** automation

Use case: smart home deployment

- The user has deployed:
  - Smart plugs, to monitor power consumption
  - Smoke detectors
  - Connected thermostat and weather station
  - Lighting system with presence sensors...

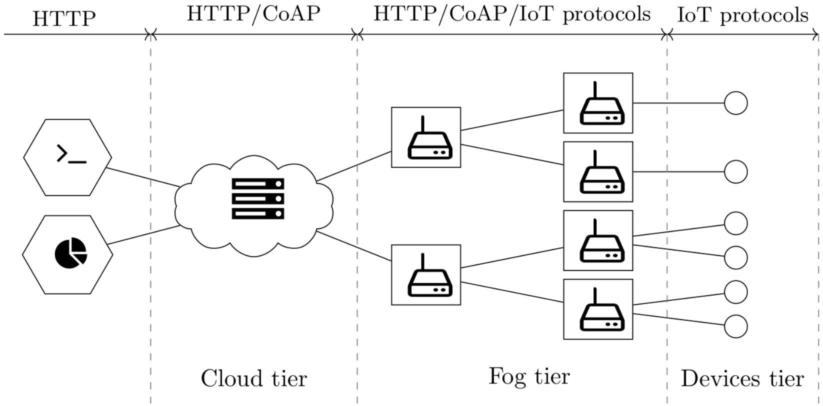
Use-case apps

- App A: Rule-based reasoning
    - Light schedule scenario
    - Smoke detection
    - Temperature preferences
- App B: Machine-learning based
    - Power consumption profile

Typical approach and its limitations

Cloud-based IoT service platforms

- Data is sent from local network to remote platform
- Cloud-based platform allows automation, analytics, remote control...



Service provider dependency

- The data is controlled by the service provider
- The user cannot migrate easily to another platform

Privacy concerns

- Smart home data is collected in an intimate setting.
- Potential inference of personal information:
    - Presence patterns at home
    - Religious practices

Interoperability issues

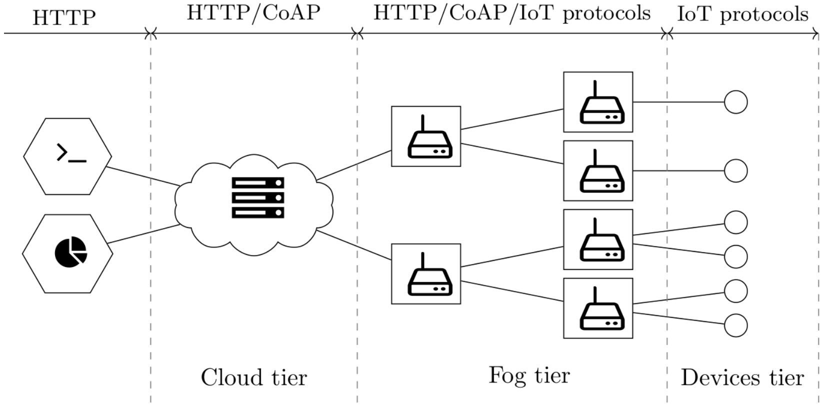
- The data is controlled by the app
- Systems may be isolated in silos

Security vulnerabilities

- Confidentiality
  - Concentration of many users data in one place
  - Data must be sent to a third party
- Availability
  - Single point of failure

Shifting from Cloud to Fog computing

- The user keeps the data on premise
- Dispatching computing instead of data
- Inference results flow in the hierarchy too



[N. Seydoux et al., 2019]

Shifting from app-centric to user-centric

- Each user is in control of their data
- The data store may be self-hosted, or hosted by a trusted party
- Migrating from one store to another is possible
- Applications read data from the user-controlled store

Solid, access control on top of Linked Data

[S. Capadisli et al., 2021]

The user

- They control the data.
- They use apps to get services.

The app

- It consumes the data: **reasoning**.
- It produces more data: **inference**.

The Pod

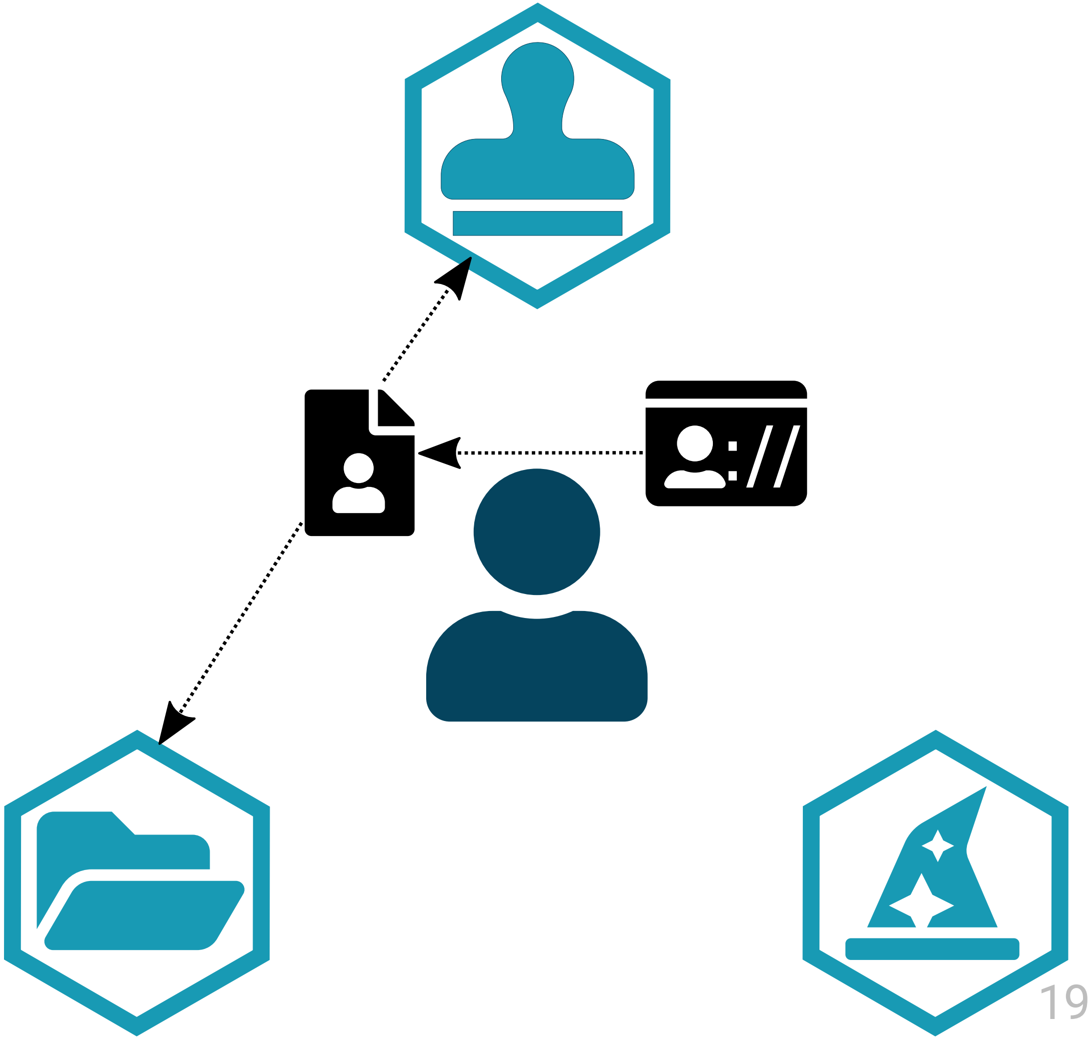
- It hosts the data.
- It enforces access permissions.

The Identity Provider

- It authenticates the user.

The WebID

- An IRI identifying the user.
- WebID profile links to user data.



Authentication patterns

Two patterns to access data

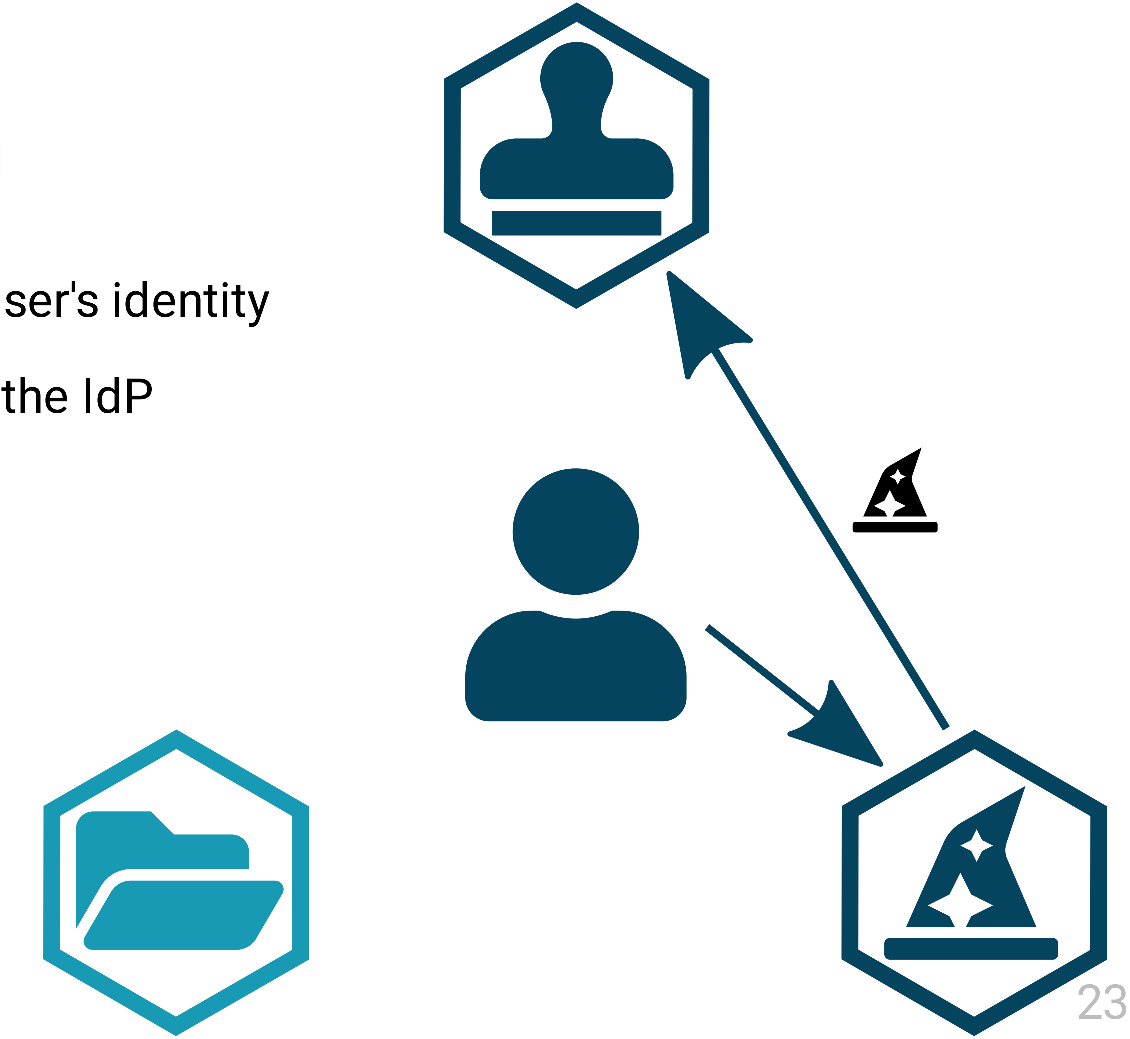
- The User logs in the Client
- The Client acts on its own behalf (bot)

[Coburn, A. et al, 2022]

Authentication patterns: User login

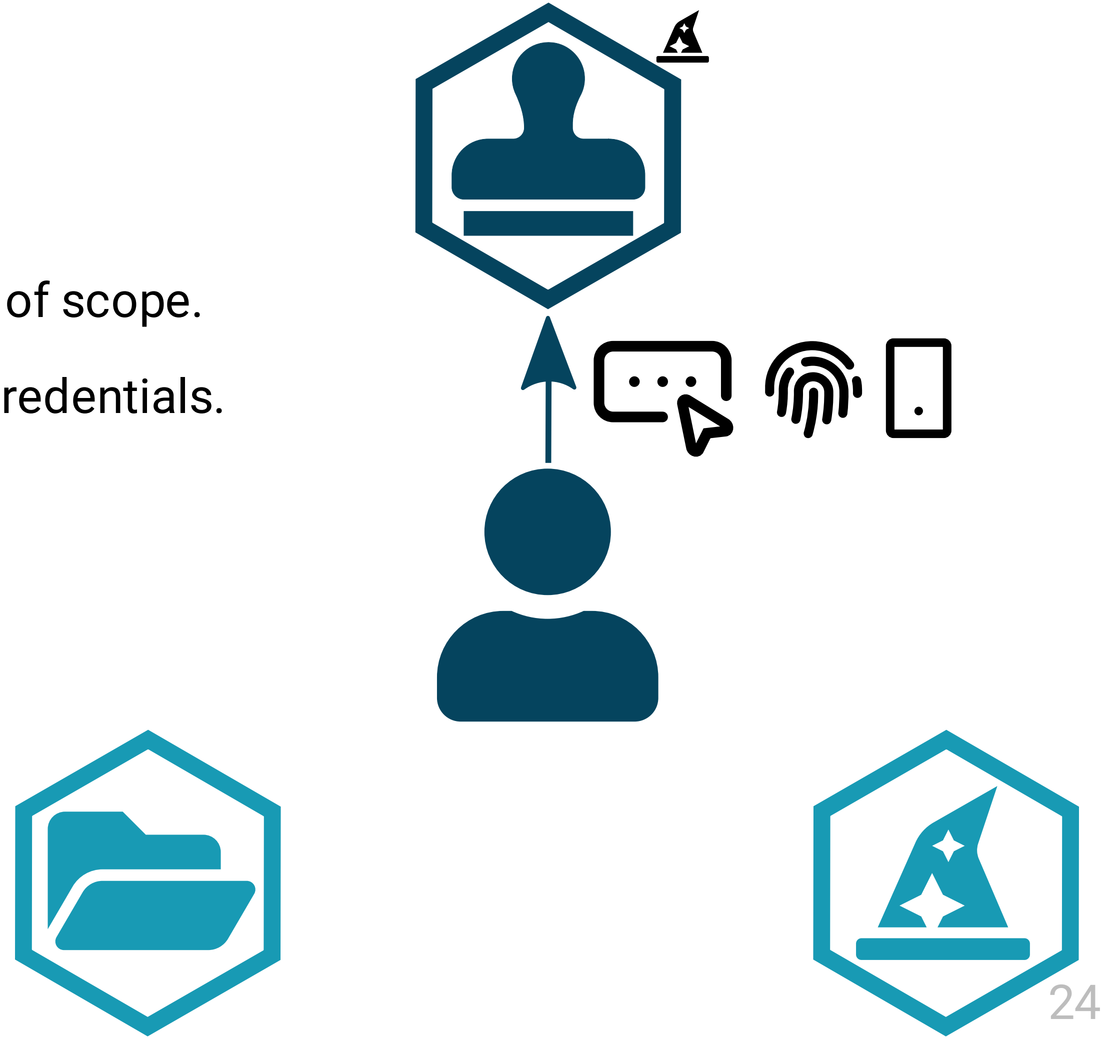
User initiates login

- The Client doesn't manage the User's identity
- The Client dedirects the User to the IdP



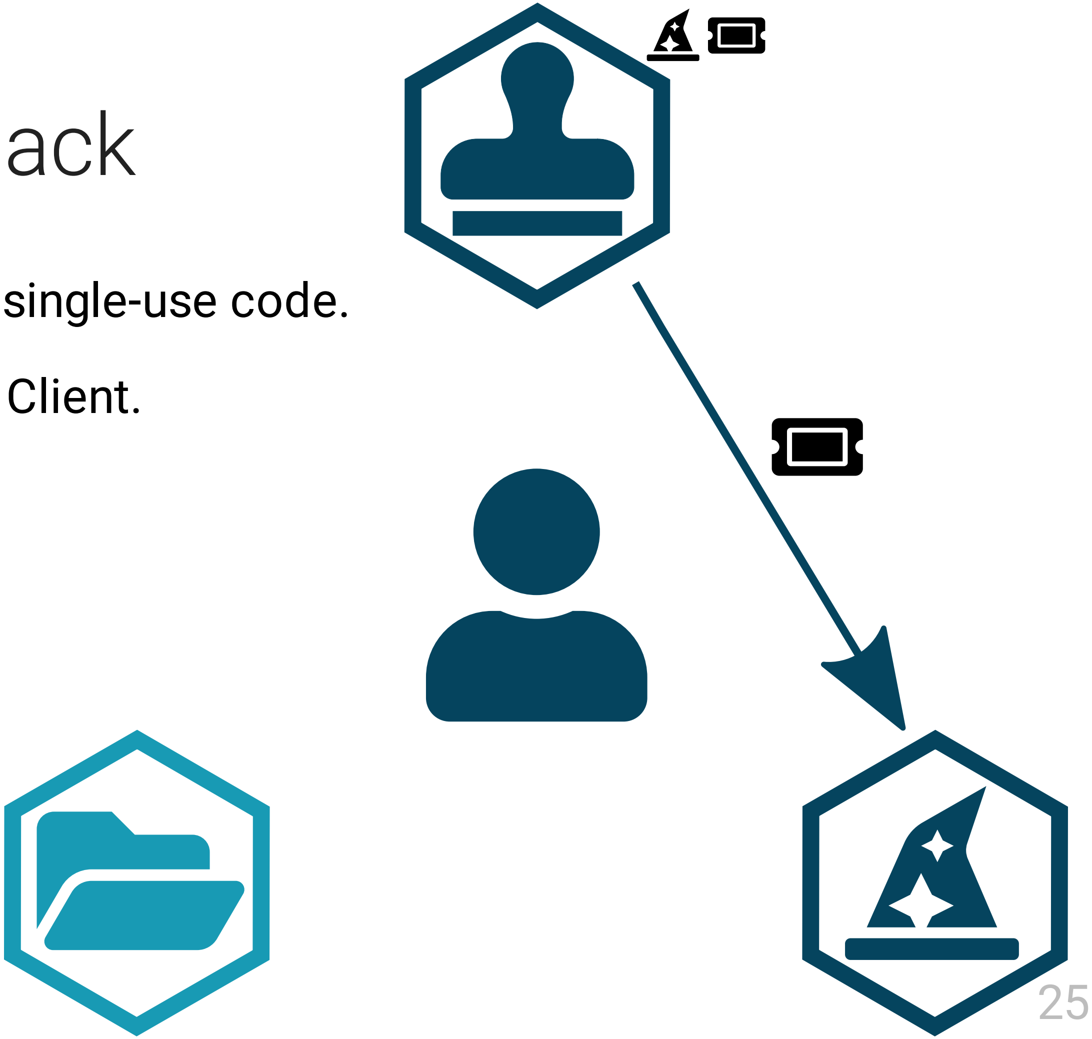
User logs in

- The authentication method is out of scope.
- The Client never sees the User's credentials.



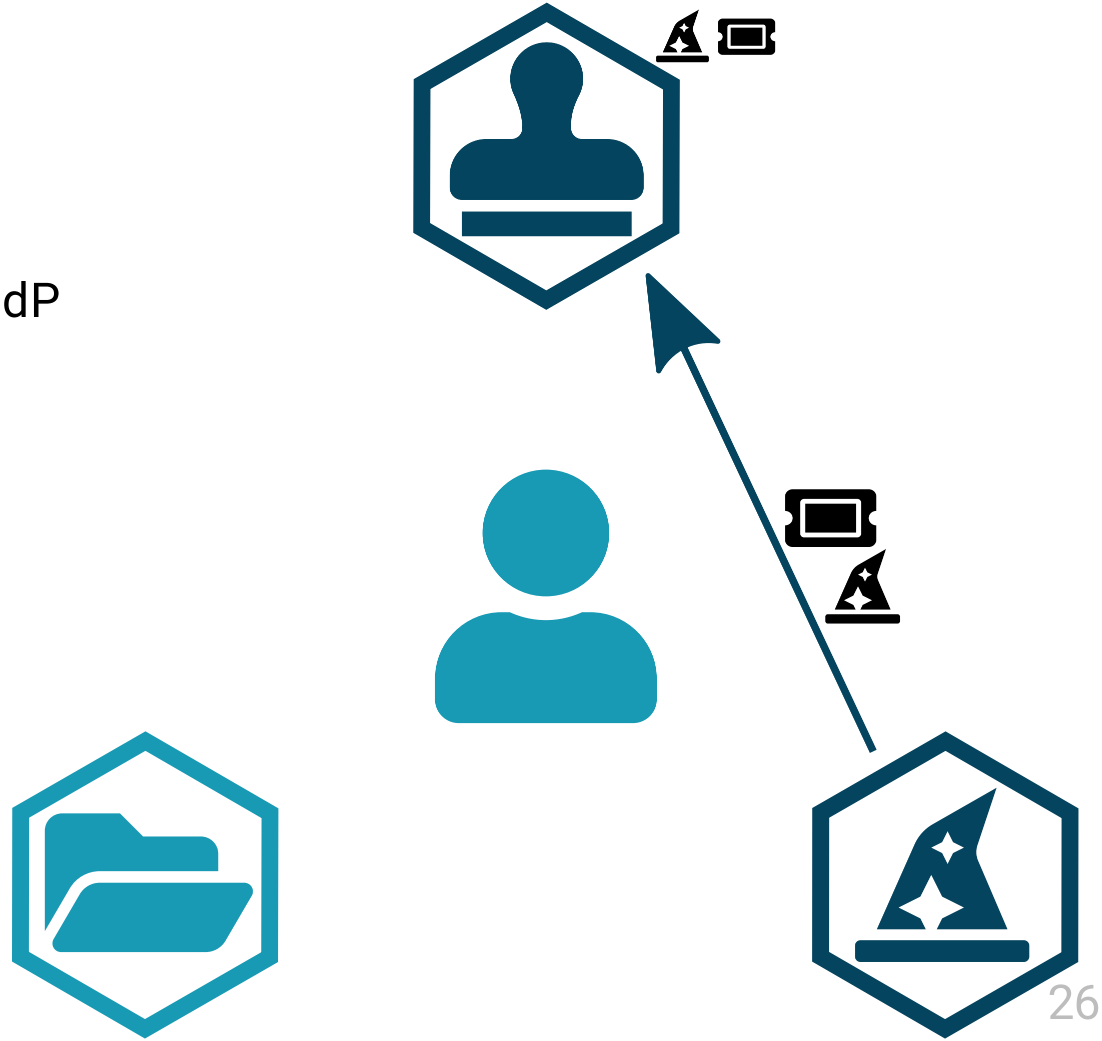
IdP redirects User back

- The IdP provides the Client with a single-use code.
- The IdP asociates the code to the Client.



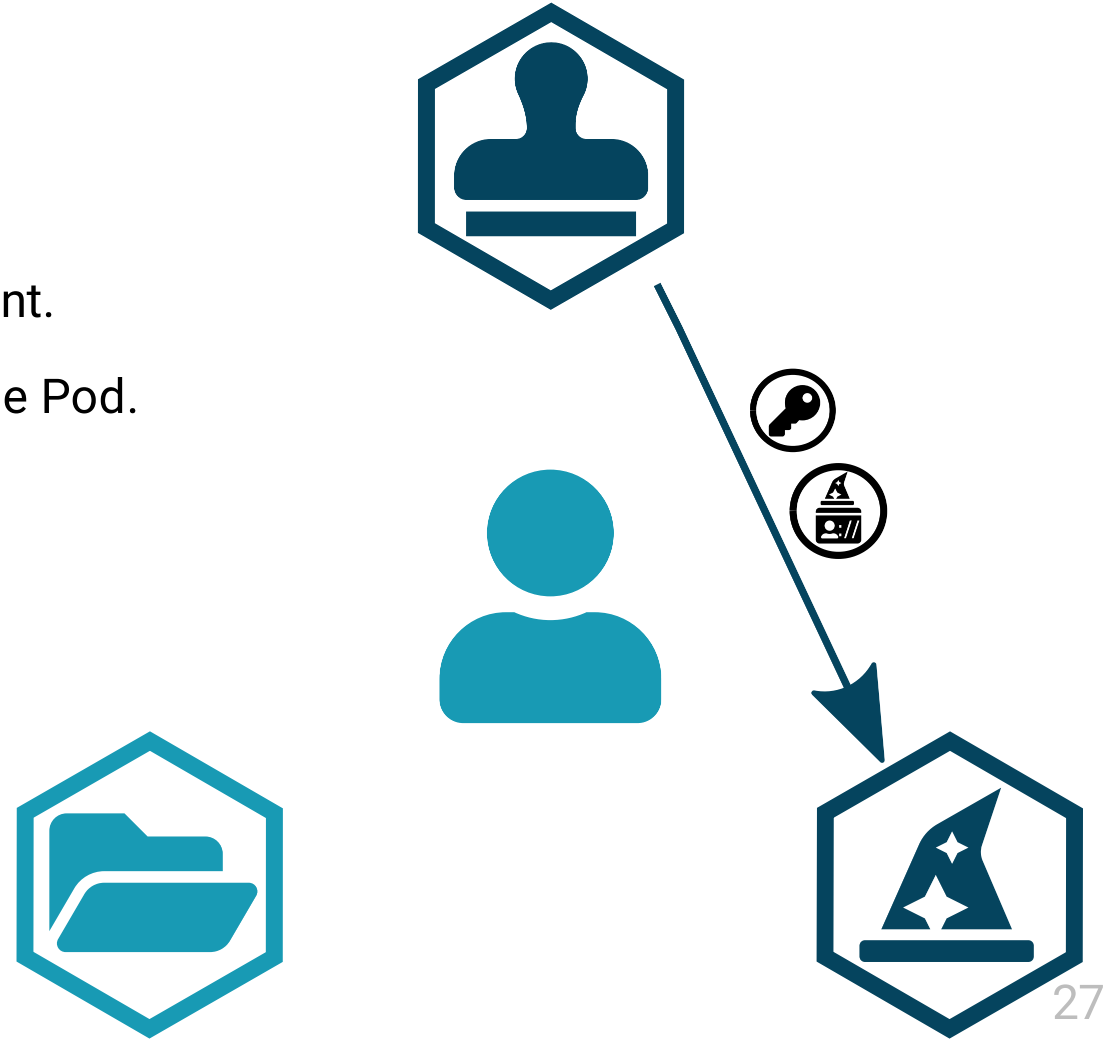
Client sends code

- The Client sends the code to the IdP



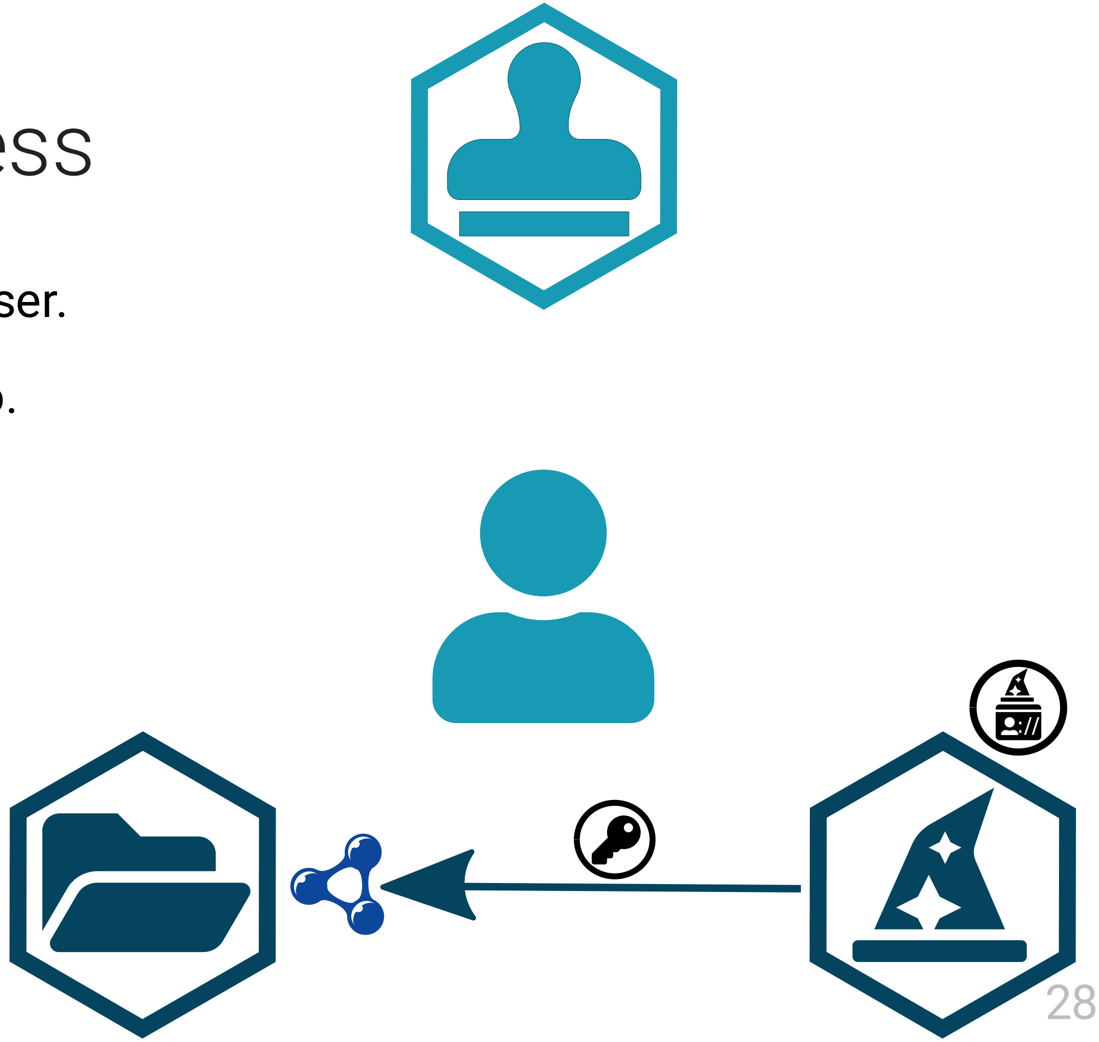
IdP sends tokens

- The ID token is meant for the Client.
- The Access Token is meant for the Pod.



Authenticated access

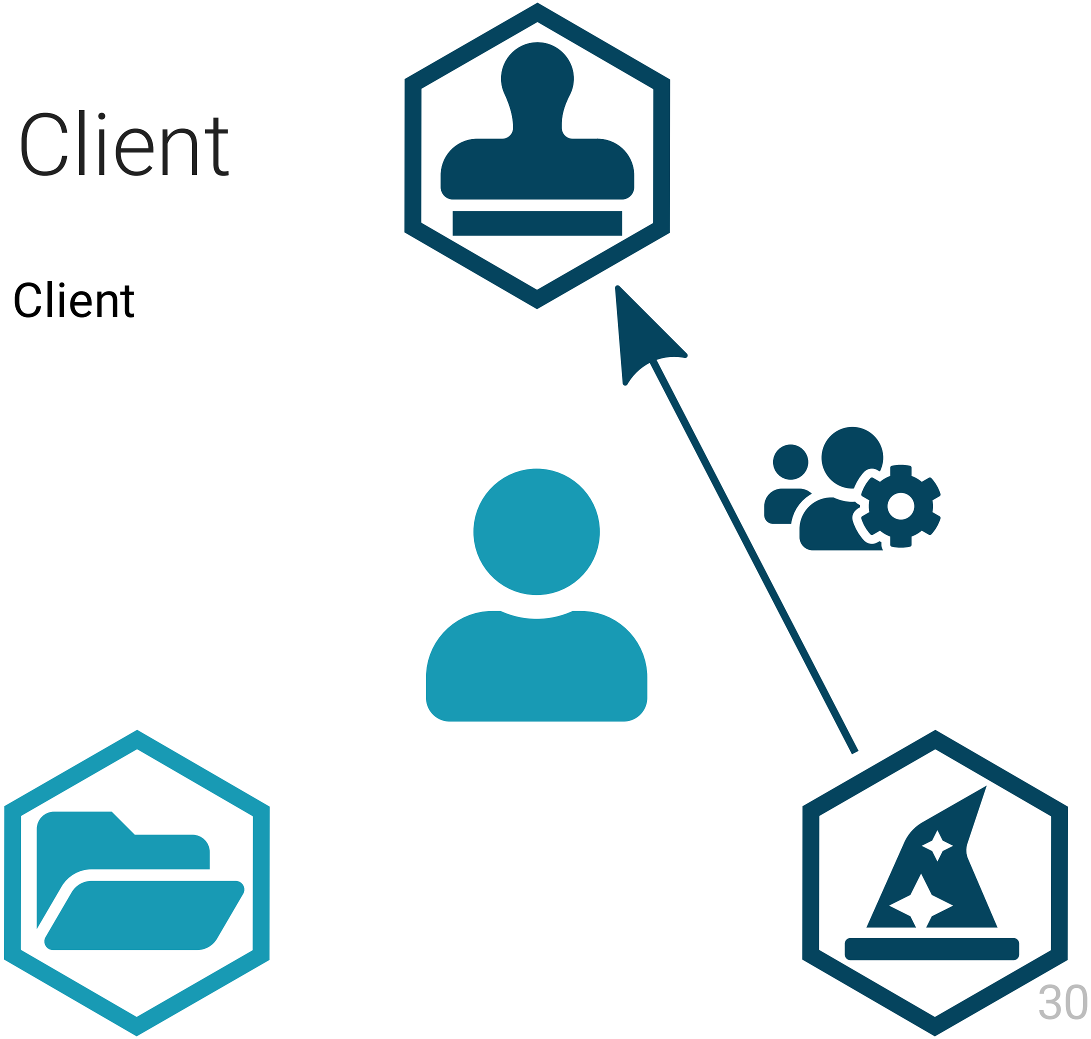
- The Client acts on behalf of the User.
- The Client has its own identity too.



Authentication patterns: Client login

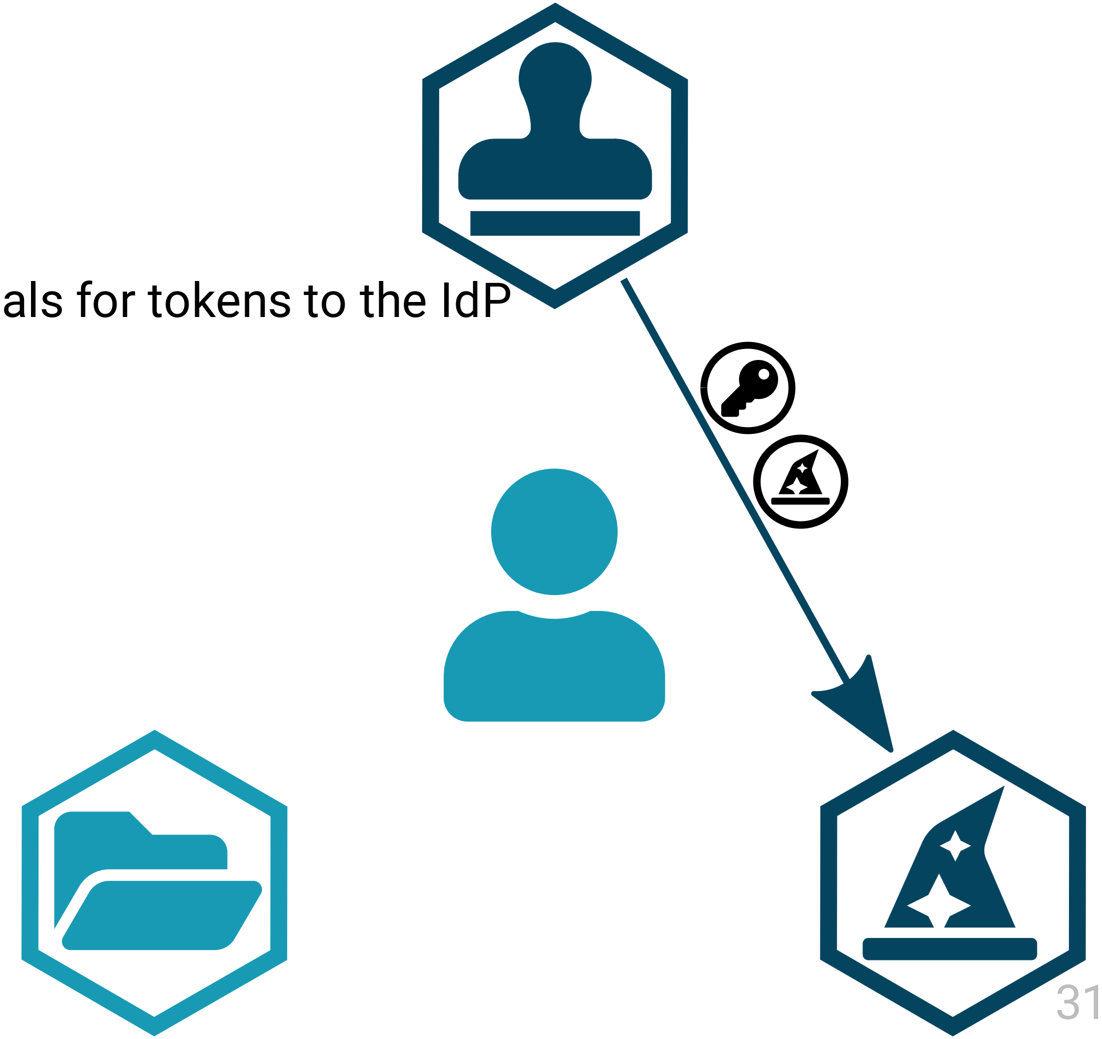
Developer registers Client

- The IdP has information about the Client
- The Client gets credentials.



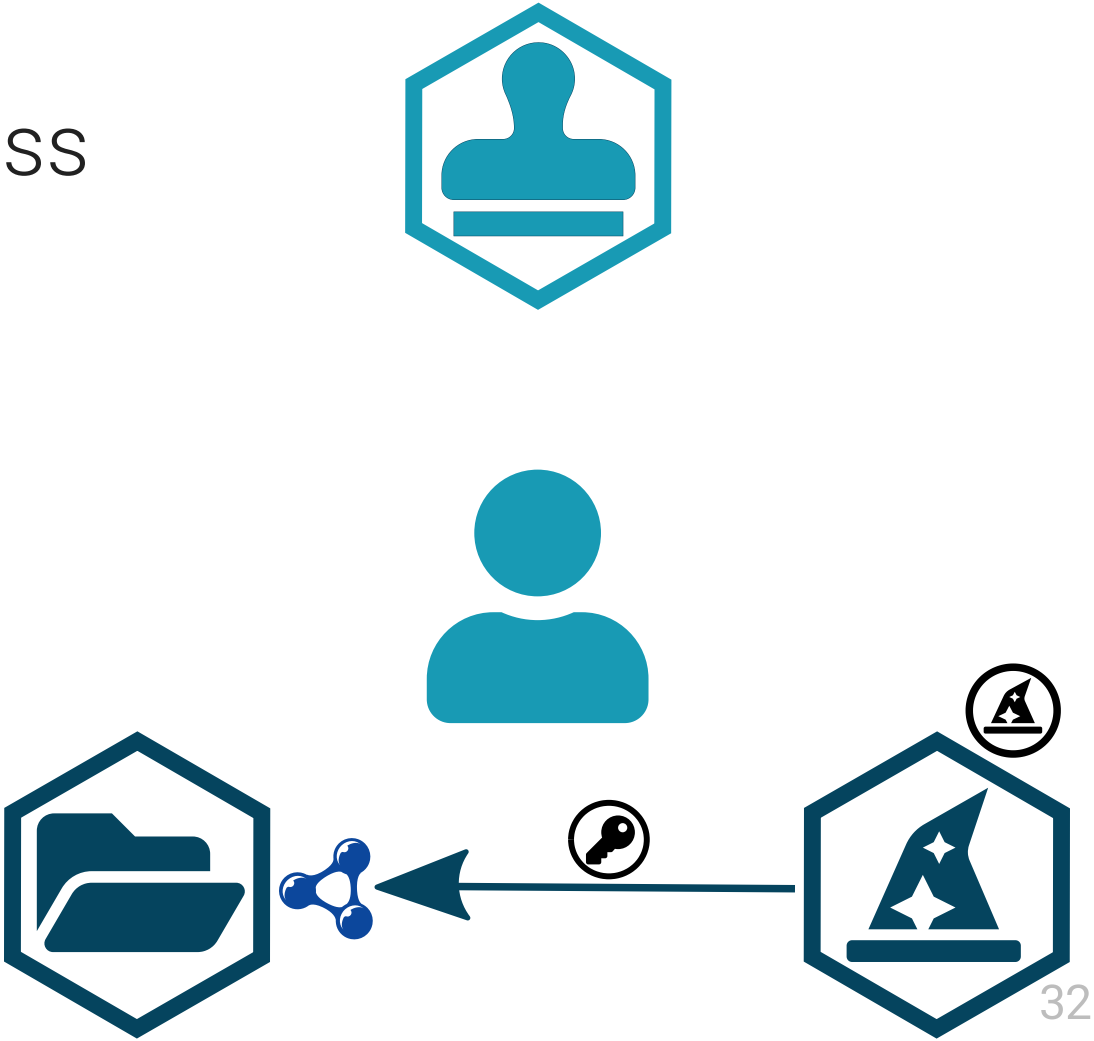
Client logs in

- The Client exchanges its credentials for tokens to the IdP



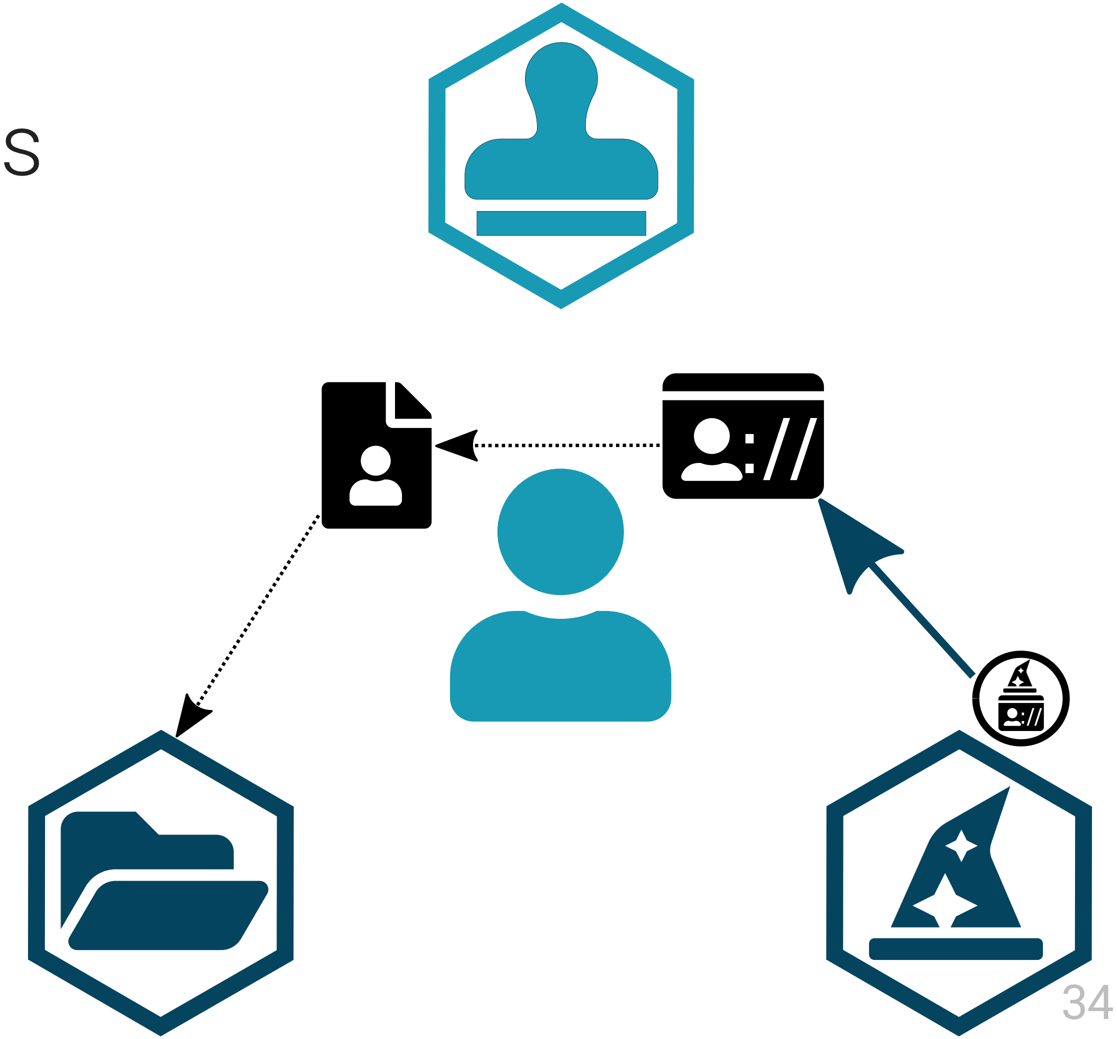
Authenticated access

- The Client acts on its own behalf.



Navigating the data

Discovering a user's data store



Getting the Pod root

Traversing to a container

Getting an RDF resource

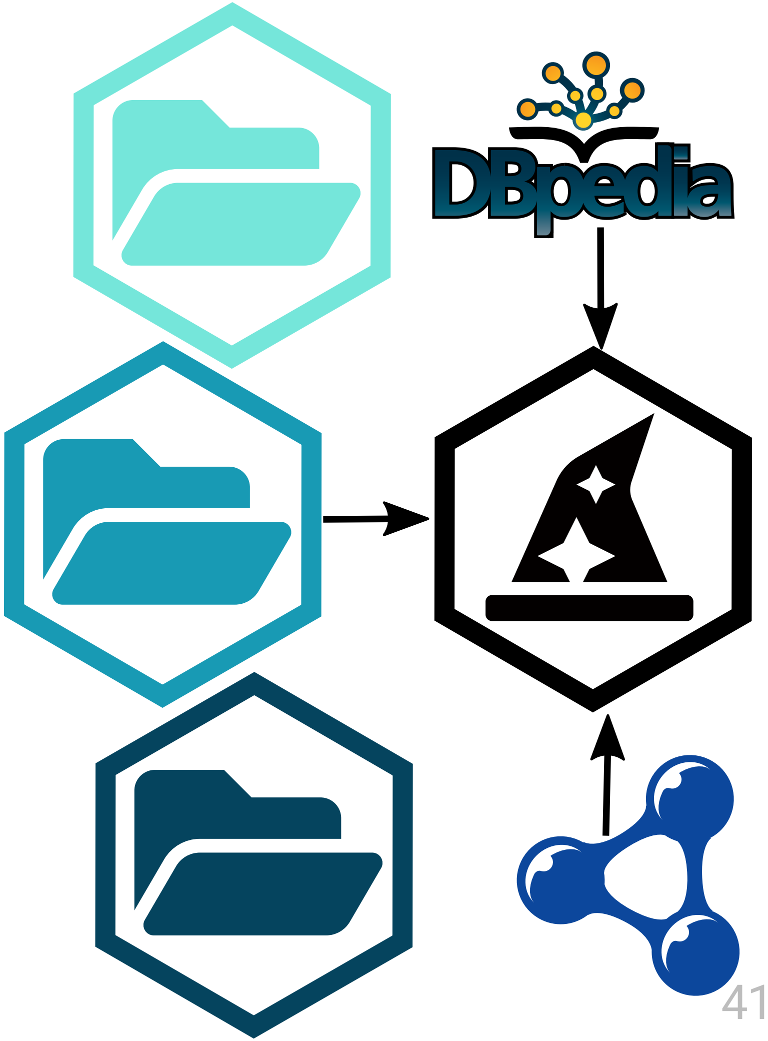
Disovering more resources

Getting a non-RDF resource

Reasoning against Solid Pods

Centralizing common, decentralizing individual

- Not all data is personal
    - Vocabularies/instances
    - Devices datasheets/local deployments
    - "Common sense"/user preferences
- **Private data** linked to **shared knowledge bases**



Reasoning patterns: client side

- Since the data is directly available to the client, all of the reasoning may happen in the browser
- Lightweight for the service provider
- Entierly user-centric
- User login authentication pattern

[Terdjimi, M. et a., 2018]

Client-side reasoning use case

- Suitable for scenarios centered on one user data
- Power consumption analytics, complex event processing…
- The user is logged in the app doing the reasoning

Reasoning patterns: server-side

- Data from multiple users is aggregated in a single graph
- Allows to identify patterns beyond individual users
- Bot login authentication pattern

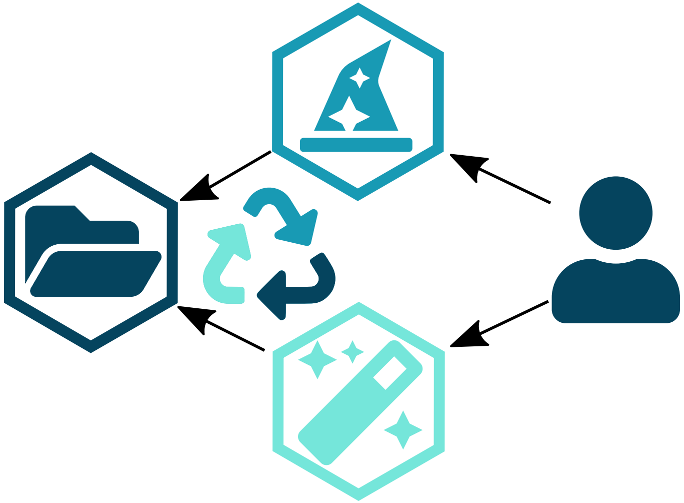
Server-side reasoning use case

- Suitable for scenarios aggregating data from multiple users
- Machine learning training, large-scale predictions...
- Individual user data is collected by the app doing the reasoning

Taking advantage of app/data separation

Beyond app silos

- All data for a user is available in their Pod
- Not all apps need to harvest: data is collected once, reused infinitely
- Power consumption data my be linked to thermostat and weather



Decentralizing chained inferences

- A Pod is a read/write storage
- Inferences on user data should be written back to the user's Pod
- Inferences chained through the Pod instead of sequentially

Challenges and discussion topics

Interoperability, symbolic and numeric AI

- Symbolic reasoning is a natural fit for Linked Data
- Numerical approaches extremely powerful too, but act as black boxes.
- How to make numerical models interoperable?

Data discoverability

- Data stored by Pod providers
- A given app or user may not have access to some data from a source
- User-centric apps only need one user's data
- How to find many small datasets to train a model?

From individual to collective and back

- User data needs to be consolidated into a single dataset for ML training
- Performance issue: N data stores => N requests
- Is it possible to adapt the result back to each individual user?

Maintening provenance

- A model is trained based on data from dynamic sources
- What provenance information should be captured to keep the model accurate?

Preserving privacy in sharing

- Anonymizing data before sharing it
- Temporal or spatial aggregtion at different scales

Conclusion

Reasoning and personal data stores

- Solid is a set of specifications centering the data around the user
- Different decentralized processing approaches are complementary
- Decoupling data from applications offers new opportunities

Bibliography

- A. Coburn, E. Pavlik, et D. Zagidulin, « Solid-OIDC ». [Online] 2022. https://solidproject.org/TR/oidc
- S. Capadisli, T. Berners-Lee, R. Verborgh, and K. Kjernsmo, “Solid Protocol,” 2021. https://solidproject.org/TR/protocol.
- N. Seydoux, K. Drira, N. Hernandez, et T. Monteil, « EDR: A Generic Approach for the Distribution of Rule-Based Reasoning in a Cloud-Fog continuum », Semantic Web Journal, 2019.
- M. Terdjimi, L. Médini, et M. Mrissa, « Web Reasoning Using Fact Tagging », in Companion Proceedings of the The Web Conference 2018.

- Credit to fontawesome for the icons (license)