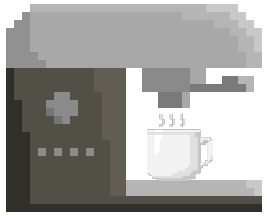
Implementing a Matrix client inside WorkAdventure

-

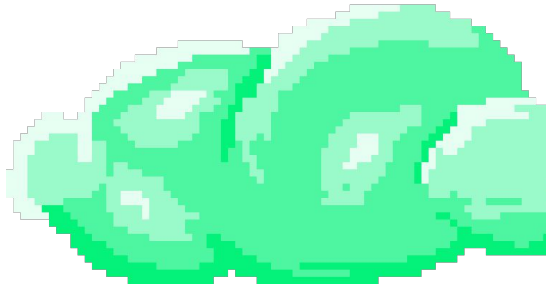
A developer feedback

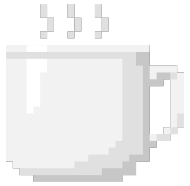
WorkAdventure
Your workplace. Better.

the matrix conference



Summary





Part 1

WorkAdventure?

Part 2

Choosing a SDK

Part 3

Connection to Matrix

Part 4

Managing encryption

Part 5

What's next?

“me”

David Négrier
CTO @ WorkAdventure

@moufmouf

@david_negrier

joind.in/user/moufmouf

@dan:workadventu.re

WorkAdventure
Your workplace. Better.

What is **WorkAdventure?**

A platform to build virtual universes

WorkAdventure allows anyone to build a **virtual place** where he/she can meet with other people, online.

WorkAdventure reproduces the **sense of belonging** to a place and being **physically present**, even when you are **remote**.

Demo
time!

WorkAdventure

(Almost) Open Source

APGL v3 restricted by Commons Clause

WorkAdventure

Distributed

Self-host your servers
Jump from world to world
Host your maps on any web server

WorkAdventure

Stateless

No database

WorkAdventure

Stateless

No database

⬇️

No message history
No contacts
No friends

WorkAdventure

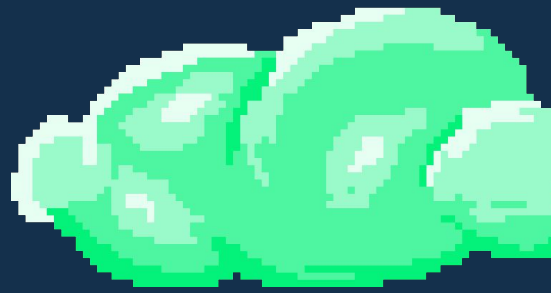
Stateless

No database

⬇️

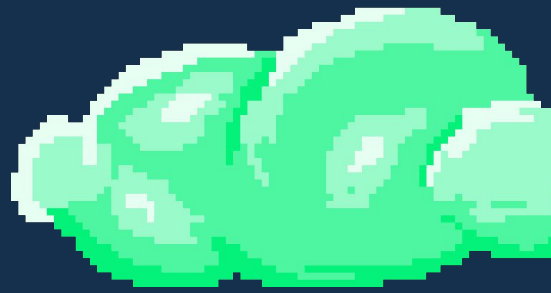
No message history
No contacts
No friends

Issue



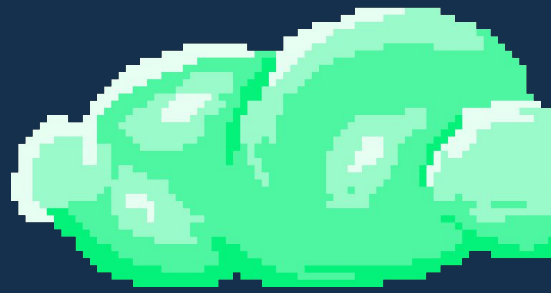
Need a way to **contact users** when they are not online

Issue



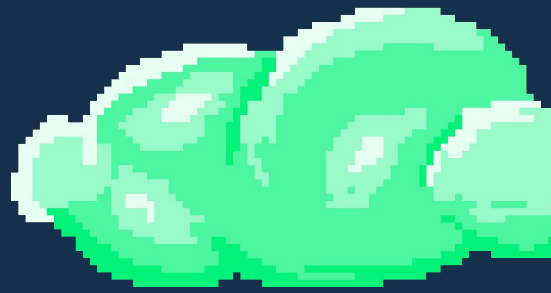
Need **conversation history**

Issue



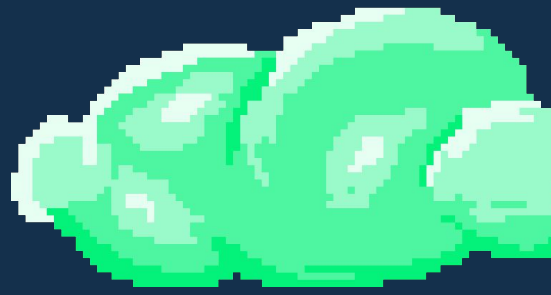
Need **large group chats**

Issue



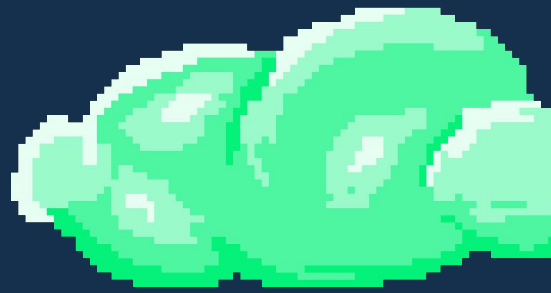
Need **messaging system**

The perfect messaging system



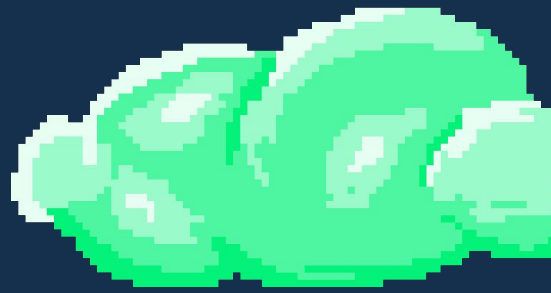
- Open
- Distributed
- Have a large community

The perfect messaging system



XMPP !

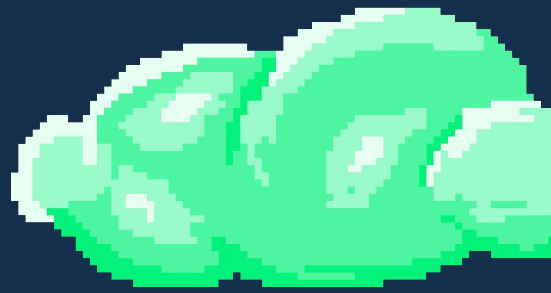
The perfect messaging system



- Client
- Client
- Client
- Client
- Client

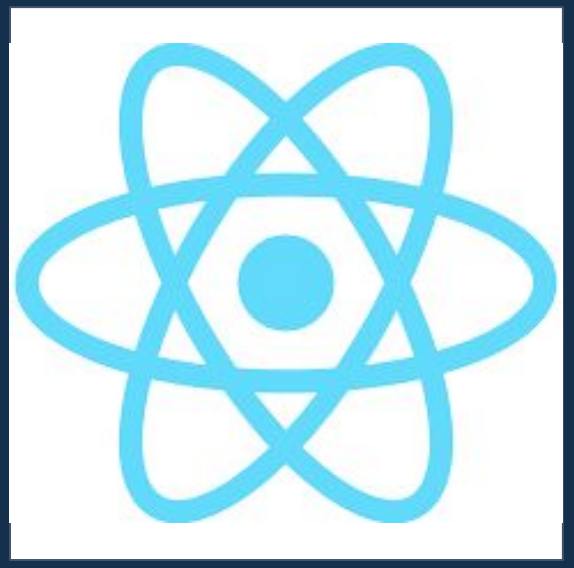
Planning the **migration!**

Choosing a SDK



matrix-js-sdk

matrix-rust-sdk

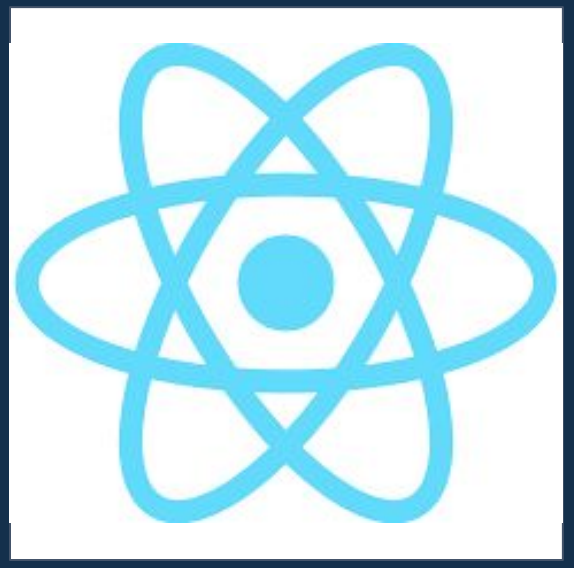


matrix-react-sdk

Choosing a SDK

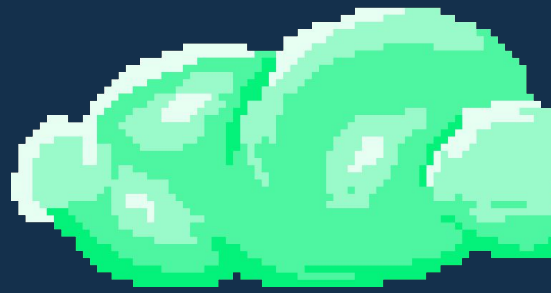
matrix-js-sdk

matrix-rust-sdk



matrix-react-sdk

Choosing a SDK



matrix-rust-sdk

matrix-rust-sdk is an implementation of a Matrix client-server library in Rust.

Status

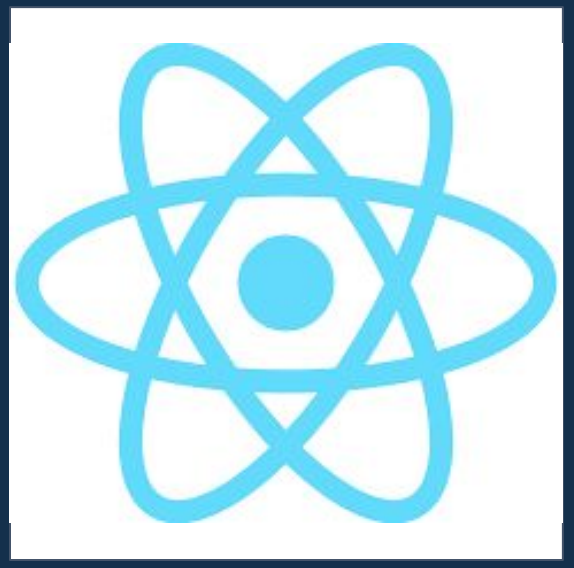
The library is in an alpha state, things that are implemented generally work but the API will change in breaking ways.

If you are interested in using the matrix-sdk now is the time to try it out and provide feedback.

Choosing a SDK

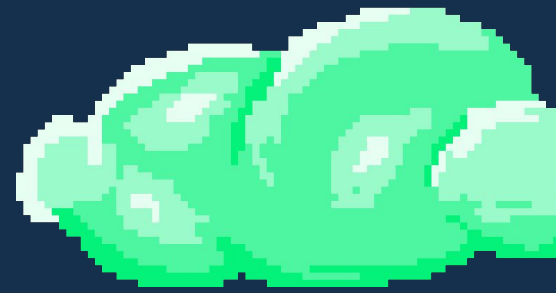
matrix-js-sdk

matrix-rust-sdk

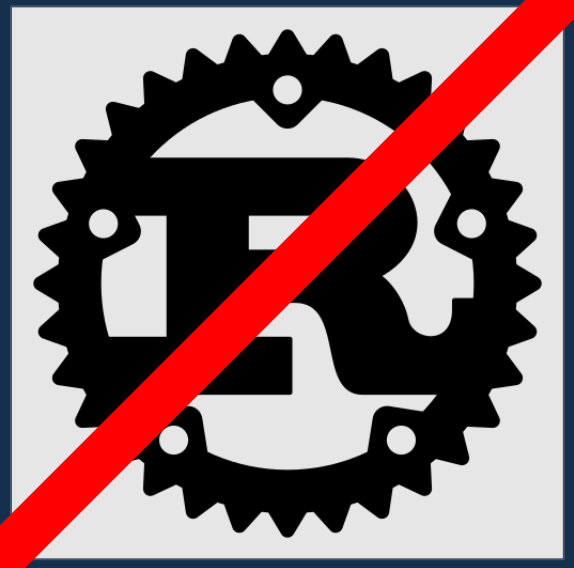


matrix-react-sdk

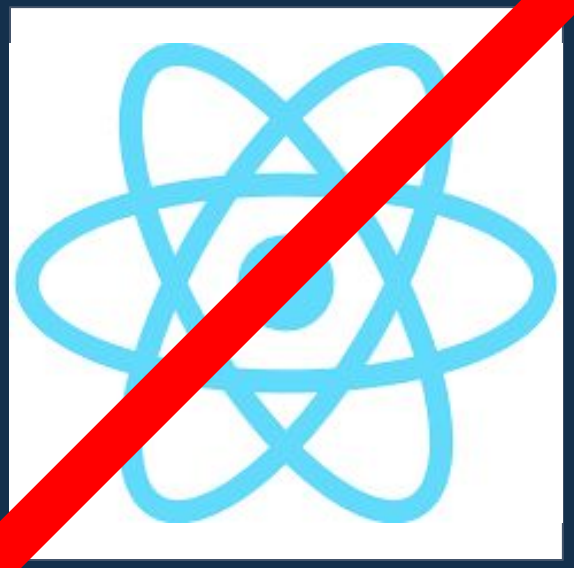
Choosing a SDK



matrix-js-sdk



matrix-rust-sdk



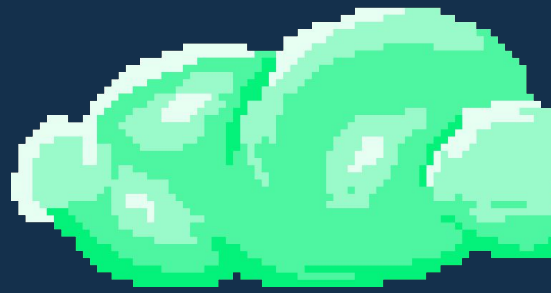
matrix-react-sdk

Choosing a SDK

matrix-js-sdk

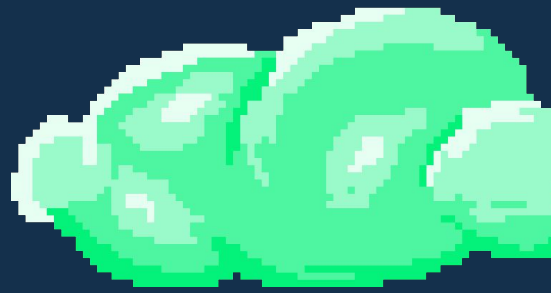
Starting the development

Valid information sources



- matrix-js-sdk README
- matrix-js-sdk Source code
- Matrix Specification
- Matrix rooms!
- #community:matrix.org (it took us 6 month to find!)
- matrix-react-sdk / cinny (code that is using matrix-js-sdk)

Community



Matrix rooms related to Matrix are all in a space:

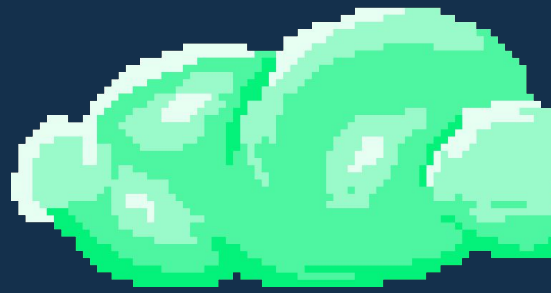
#community:matrix.org

People are super helpful

Lack a proper online archive (a lot of valuable knowledge is lost) => 😢 *archive.matrix.org* 😢

Connecting
to Matrix

Are we OIDC yet?

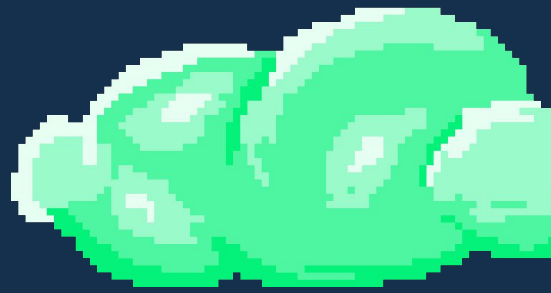


WorkAdventure allows OIDC connections.

Matrix is migrating to OIDC…

Let’s try!

OIDC Native vs Aware clients



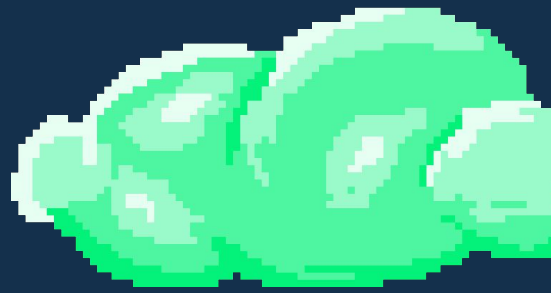
Native client

Fully uses OIDC when talking to an OIDC enabled homeserver.

Aware client

Aware of OIDC but will still use existing auth types (e.g. *m.login.sso*) to auth with an OIDC enabled homeserver.

OIDC Native vs Aware clients



Nativ

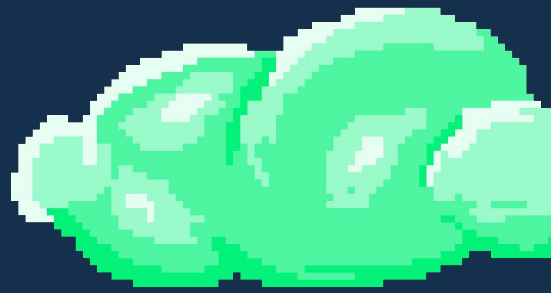
F ıking to
a homeserver.

Aware client

Aware of OIDC but will still use existing auth types (e.g. *m.login.sso*) to auth with an OIDC enabled homeserver.

Future!

Trying the OIDC Native way



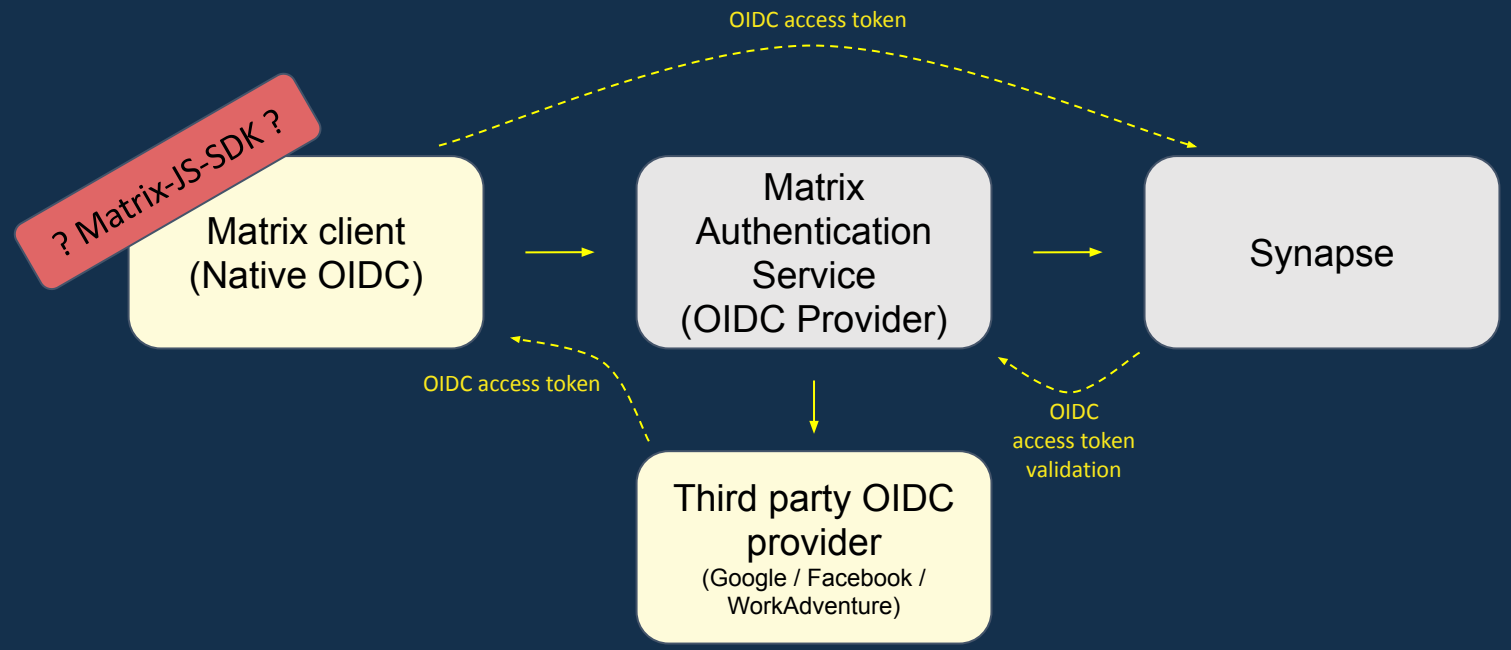
Synapse does not yet support OIDC native connections.

Need to add a compatibility layer:

MAS

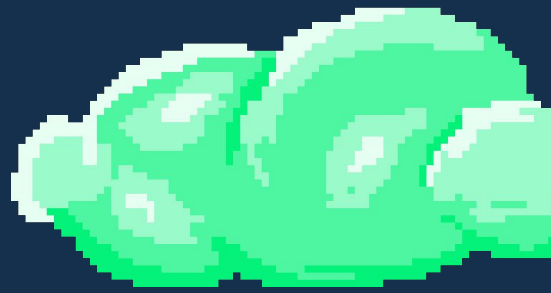
(Matrix Authentication Service)

Trying the OIDC Native way



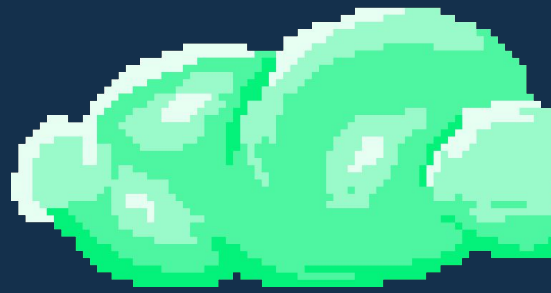
Are we OIDC yet?

Are we OIDC yet?



The **Matrix-JS-SDK** is **not compatible** with
Native OIDC servers…
…**yet**

OIDC Native vs Aware clients



Future!

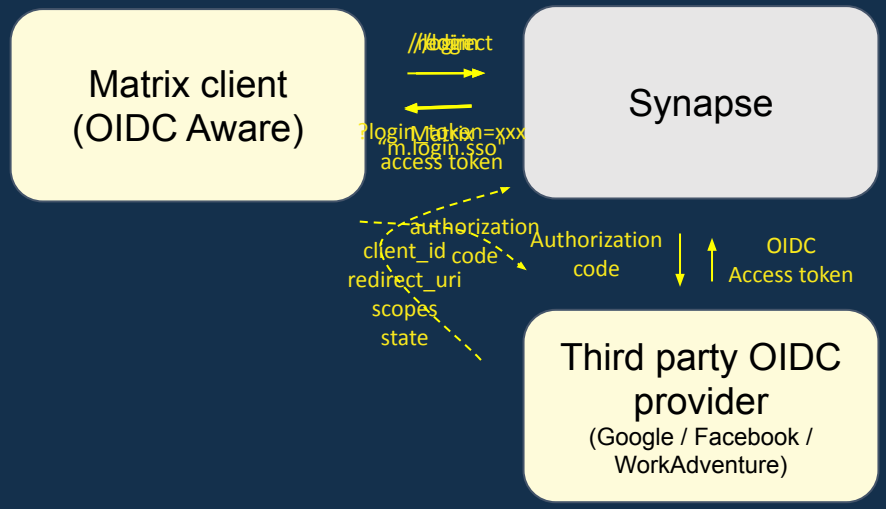
Native client

Fu[illegible]iking to a[illegible]omeserver.

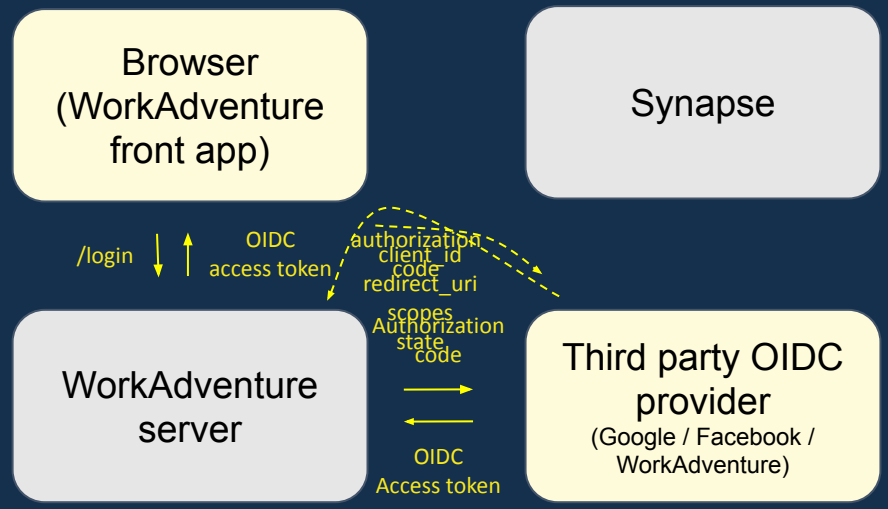
Aware client

Aware of OIDC but will still use existing auth types (e.g. *m.login.sso*) to auth with an OIDC enabled homeserver.

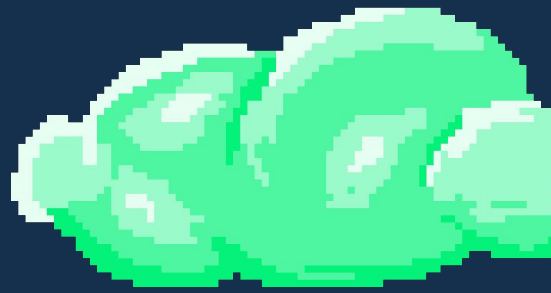
The OIDC Aware way



The OIDC Aware way

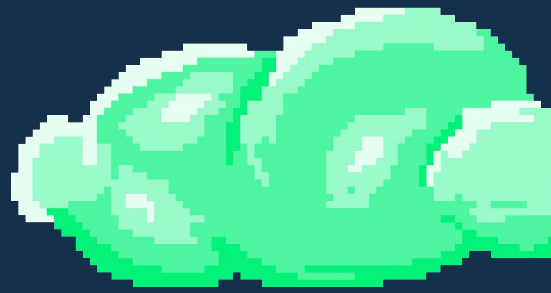


Managing 2 login processes



I don’t want to login **twice**

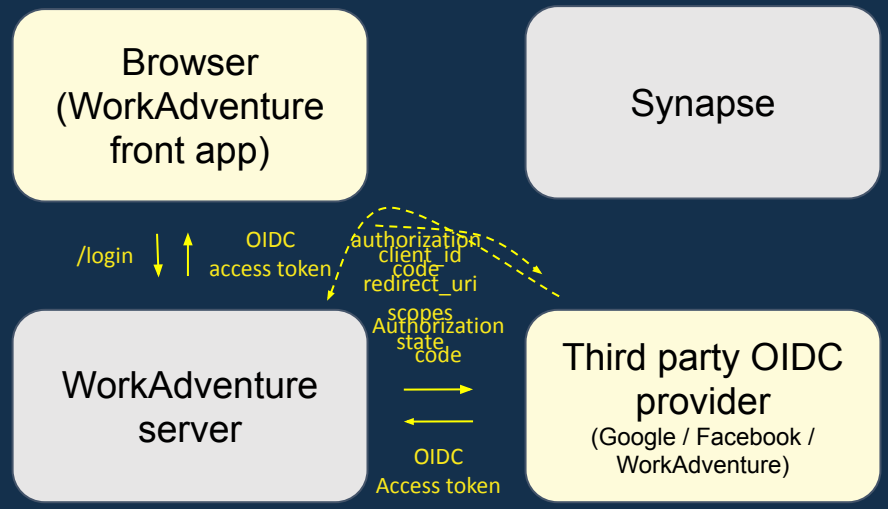
Managing 2 login processes



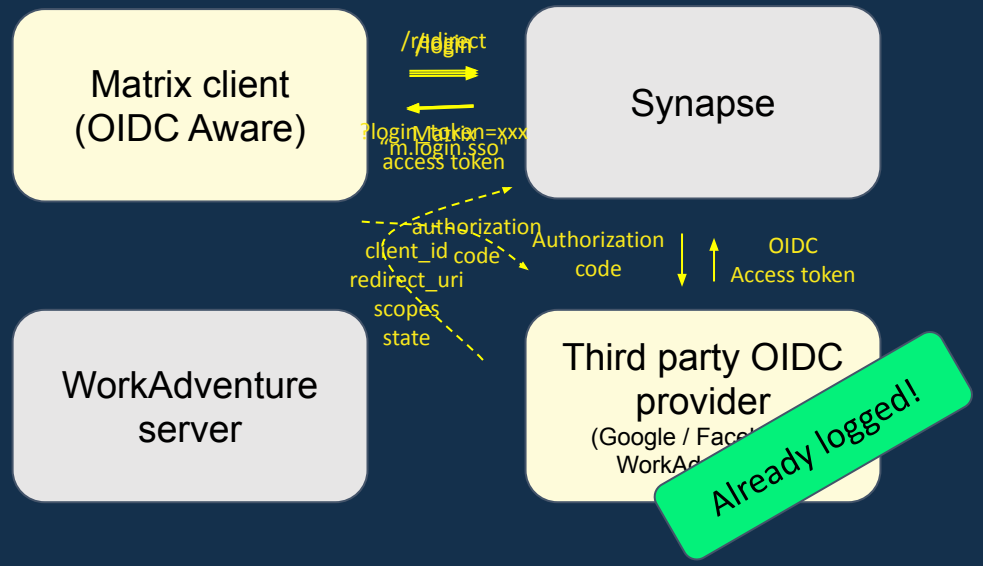
Can I **share** my WorkAdventure OIDC access token with Synapse?

The answer: **DON’T**

The OIDC Aware way

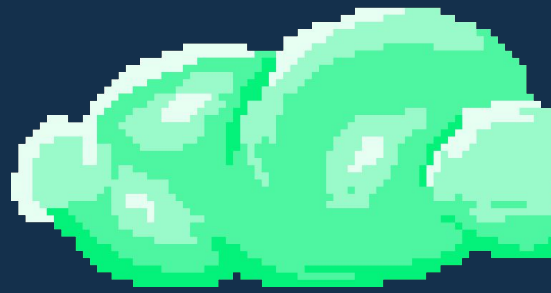


The OIDC Aware way



Implementing encryption

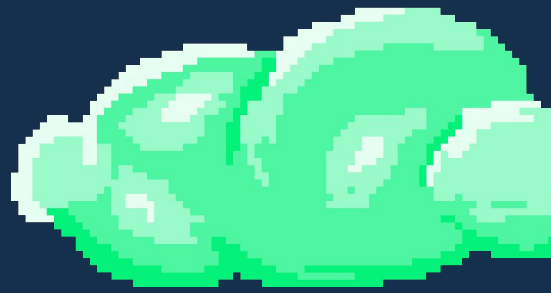
Implementing encryption



😨 Scary 😨

(and we were right to be scared)

Lifecycle of the local database



The Matrix-JS-SDK only supports **one user** at a time.

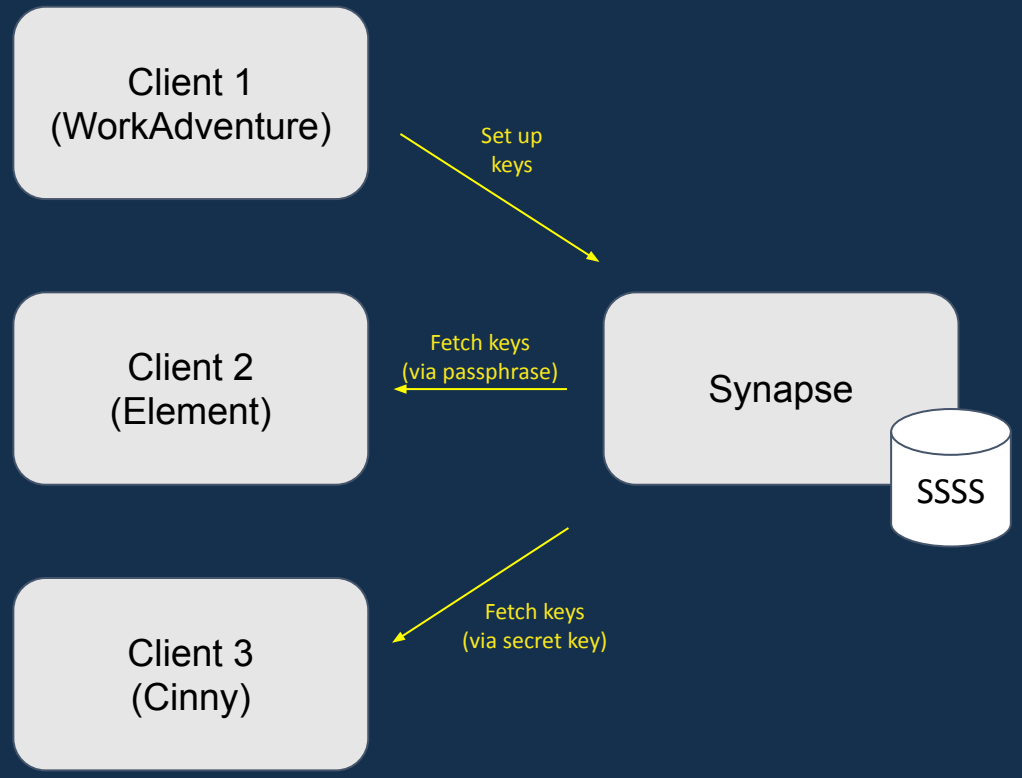
When someone else logs in, you must **delete the old data**.

Cross-signing is complex

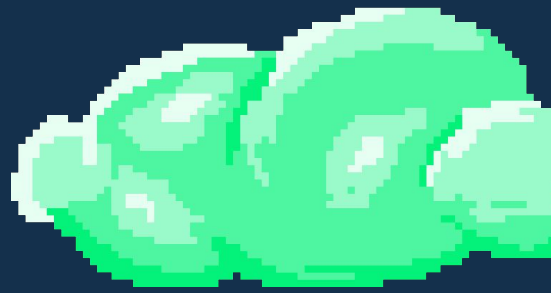
End-to-end encryption needs keys.

Those keys must be shared by clients of a same account.

The Synapse server can store those keys (Secure Secret Storage and Sharing)

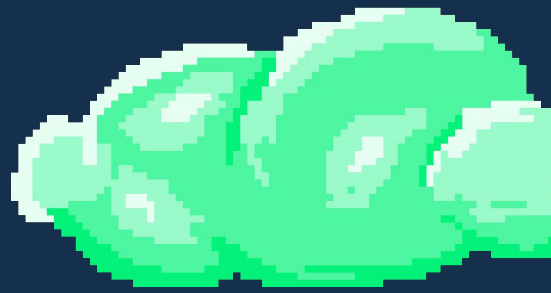


Cross-signing is complex



No documentation in the SDK

Cross-signing is complex



3. Implementing SSSS

SSSS is short for Secure Secret Storage and Sharing, which provides a method of storing secret data on the server, without the server administrator or anyone else having access to that data. As such, this is optimal for storing the private keys for cross-signing on the server so that all your devices can use them later on to generate signatures, and thus verify other people.

Unfortunately there doesn't seem to be any help from libolm for implementing SSSS so this will become rather cryptographic-y to implement. You will mainly need to find libraries for the following functions:

1. AES-CTR encryption and decryption
2. SHA256 HMAC calculation
3. Base58 encoding and decoding
4. PBKDF2

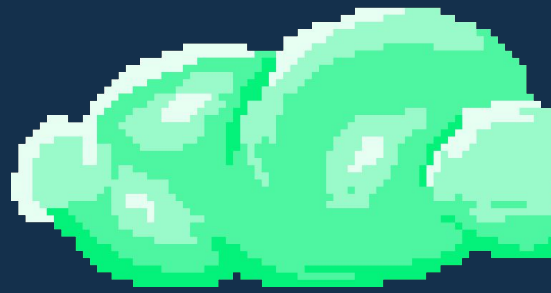
As the "sharing" aspect is simpler to implement, here it will be implemented first. It can later on also be easily used to debug & verify the functionality of the "secure secret storage" part. It is, however, still crucial to look at the general structure of the SSSS first.

TODO: This section doesn't list yet how to *create* keys, add this later on.

General Structure of SSSS storage

SSSS depends on keys with which the actual secrets are being encrypted to be stored. These keys need to be provided by the client. There is, however, metadata information on the keys in a users account_data. For that, there are entries

Use the code, Luke

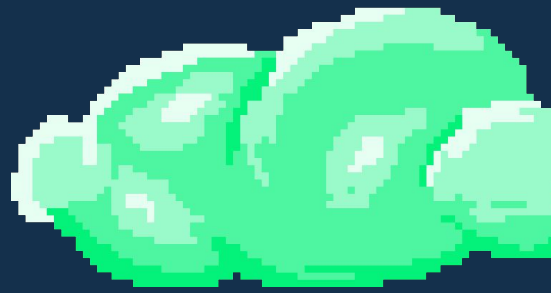


The **React SDK** and **Cinny** both use the JS SDK

We reverse engineered / copied the way they do.

ChatGPT provided a good starting point by converting **React** screens to **Svelte** semi-automatically.

Encryption is hard



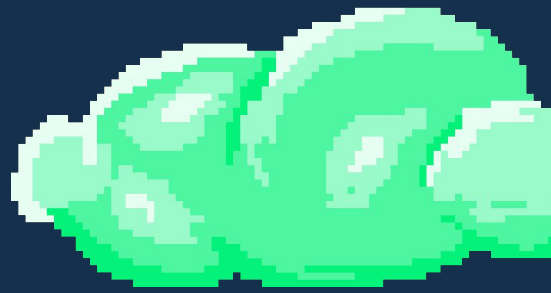
Still a lot to do (emoji / QRCode…)

If it was to do again, we would look at **Svelte to React wrappers** to avoid coding the UI again.

Demo
time!

What's **next?**

Next



Bridges!
Discord, **Slack**

WebRTC calls?

Connect **third party Matrix accounts** (i.e. matrix.org)

“me”

David Négrier
CTO @ WorkAdventure

@moufmouf

@david_negrier

joind.in/user/moufmouf

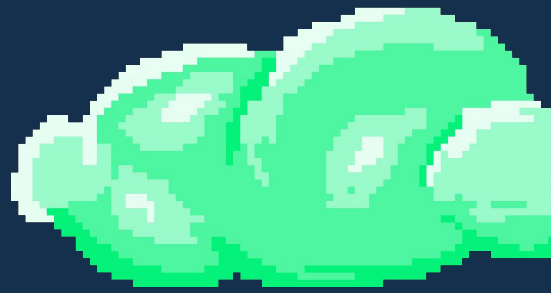
@dan:workadventu.re

WorkAdventure
Your workplace. Better.

WorkAdventure

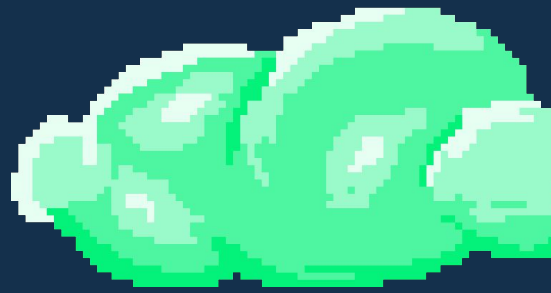
Your workplace. Better.

Plan



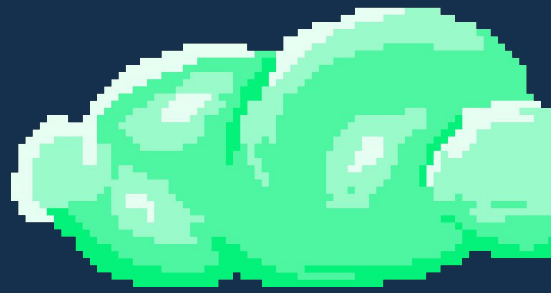
- What we do (screenshots, bringing people together)
- Open source (almost) / stateless / distributed
- Issue : when you are not here, you are not here
- Need a way to send messages to the outer world / messaging system (example: send a message from WA to a phone)
- Are we going to develop our own? No!
- We will use…. XMPP!

Plan



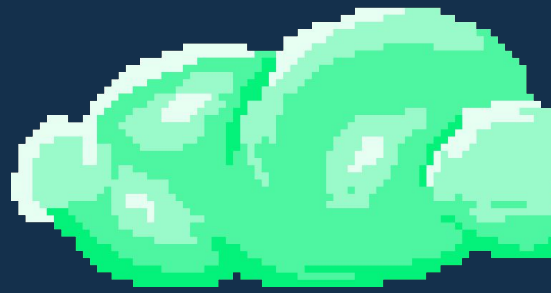
- XMPP? The “other” distributed open chat protocol
- We spent some time implementing it. It was moderately complex.
- Cool! Now, we can interact with other clients!
- …
- What clients?
- Gosh! No good clients out there.
- And then… Element :love:

Plan



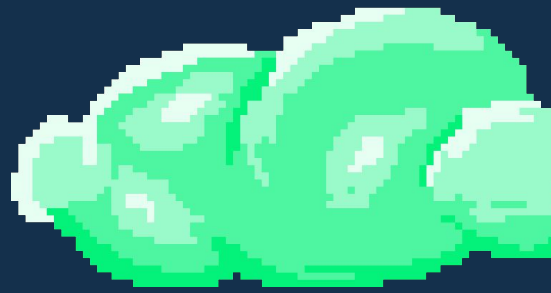
- Plan to migrate
- Choosing the tech stack => WA is a web-based platform
- slide state: js client, Rust client (can we use it in WASM? => no)
- Inside js client Olm VS WASM
- Other clients not going through a lib
- React kit
- … we are alone… until we understand that there are Matrix rooms for developers (wish there were online archives)

Plan



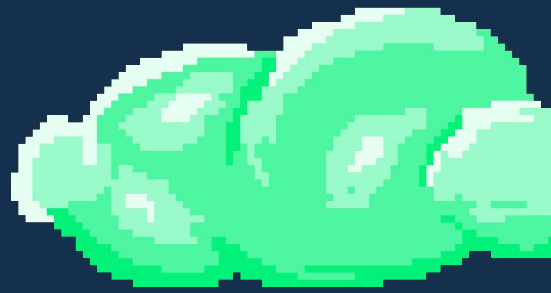
- Encryption?
- Scary! I’m not a mathematician, I know nothing about double ratchet things
- When should we delete the DB? Can we keep several users logged on the same DB (answer: no)
- How do you get your keys back
- How do you use the code of the client (hard)
- => Hopefully, React SDK uses the JS SDK.
- => ended up doing some React to Svelte work in GPT
- If this was to do again? => Svelte to React wrapper?

Plan



- Process pour créer une clef d’encryption (cross signing key) pas expliqué (détaillé dans la spec https://matrix.org/docs/older/e2ee-cross-signing/ mais expliqué dans le JS SDK) => on a regardé dans le React SDK
- Besoin d’upload la clef sur le secret storage (inclus dans Synapse)

Plan



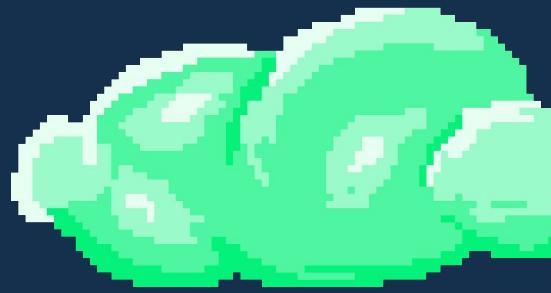
New doc:

https://matrix.org/docs/matrix-concepts/end-to-end-encryption/

La nouvelle doc ne parle pas du SSSS (Secret Storage), pourtant nécessaire.

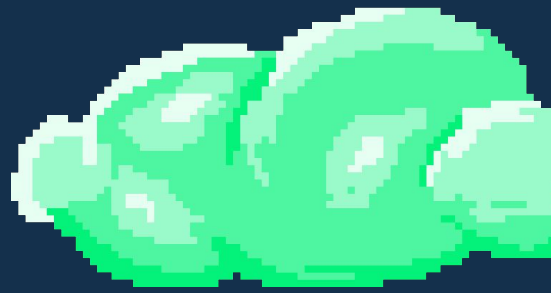
Ancienne doc:
https://matrix.org/docs/older/e2ee-cross-signing/

Plan



- The gap between Matrix and WA rooms
- Enter / Leave rooms quite often in WA!
- Invite sent by admin user associated with a zone in the map. Talk about the visibility

Plan



- What's next?
- => MatrixRTC? (4pm conf!)
- => Connect any account to your WA account