

The Matrix Conference 2026

Integrating Matrix Chat into Waldur for Sovereign Conversations

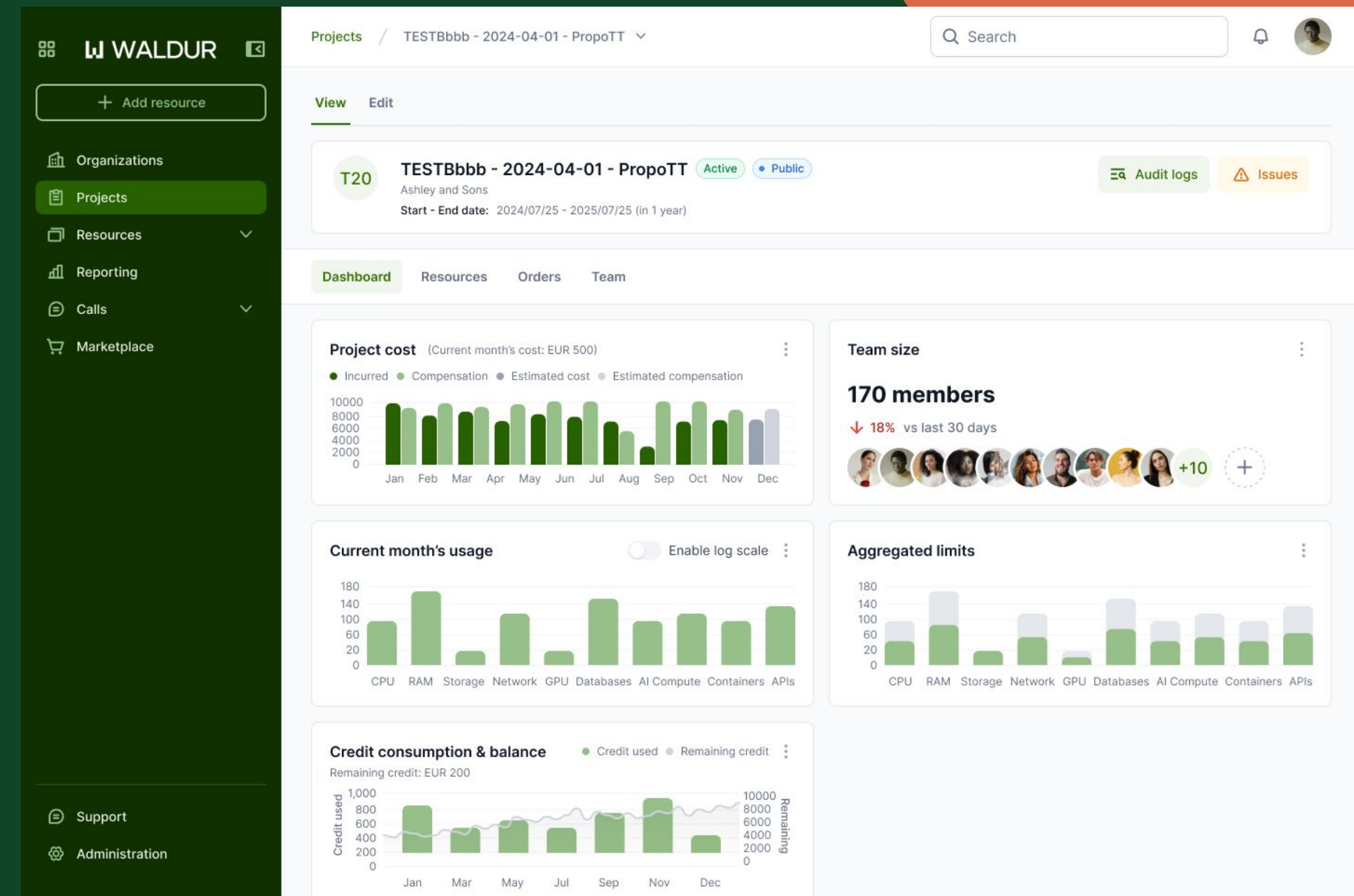
Hendrik Jaks · OpenNode

@hendrik.jaks:matrix.org

Waldur in 60 seconds

Organisation → Project → Resources

- Open-source, multi-tenant self-service portal for cloud and HPC
- Slurm allocations, virtual machines, GPU hours: ordered, approved and billed in one place
- People hold roles on projects: admin, manager, member
- The **role** is what lets you touch a resource



Homeport, project page

64

Active deployments

58.8K

Users across deployments

20

Countries

4.6K

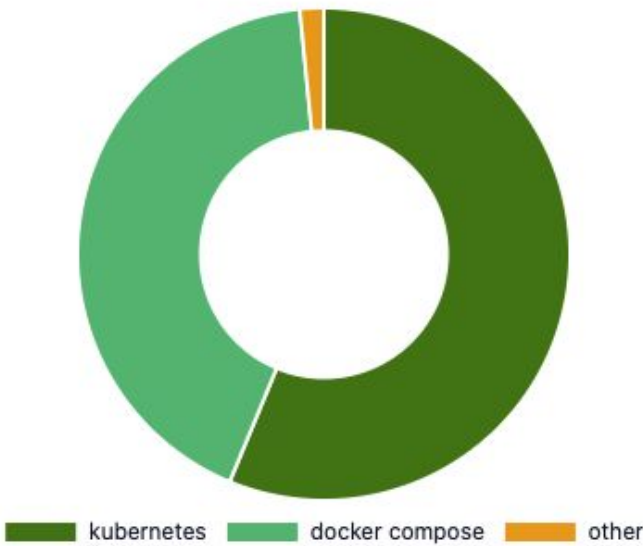
Offerings across deployments

Deployments that reported in the last 12 months · latest report 18. oktober 2026

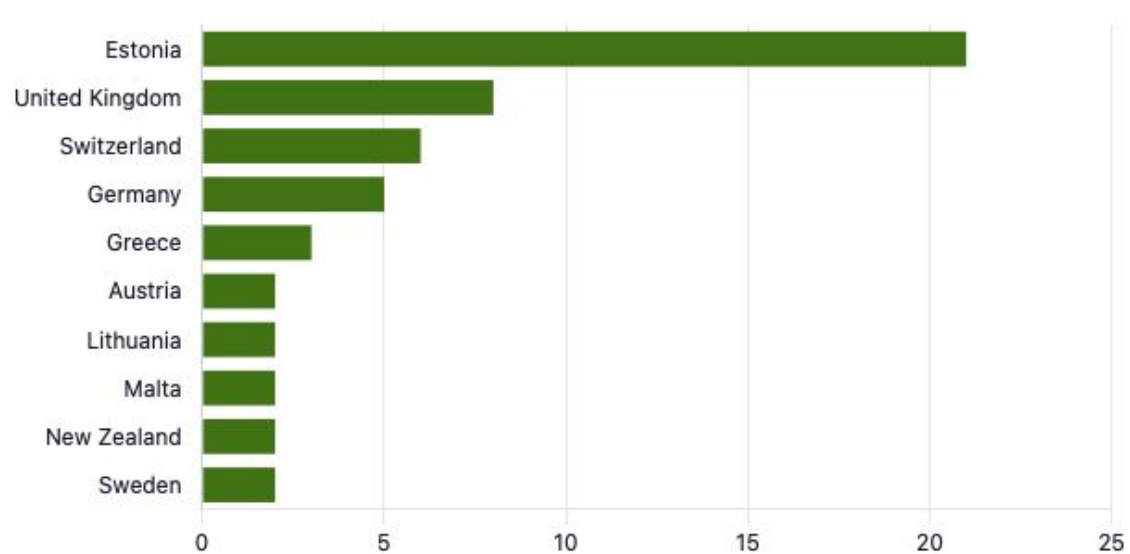
Global deployment distribution



Deployment types



Top countries



Where it runs

Riigipilv

Estonian Government
Cloud

ETAIS

Estonian Scientific
Computing Infrastructure

HPC centres

CSCS, LUMI, Cambridge

Projects routinely hold personal and sensitive data: health, registry and research data under agreements

Support today: five apps, one problem

Slack

Signal

email

ticket tracker

“someone’s DMs”

One incident, five apps, zero shared timeline

Three concrete failures

- 01 Context split across tools**
Slack, Signal, email, tickets, DMs: nobody holds the whole thread
- 02 No durable history tied to the project**
The history leaves with the person who had it
- 03 Sensitive conversations in third-party consumer apps**
Infrastructure details under a foreign jurisdiction



What we needed, written as constraints

- Self-hosted, works in isolated networks, no per-user licence
- Membership **derived** from existing authorisation: no second admin surface
- History that outlives staff turnover
- One login: your Waldur account is your Matrix account
- **Nothing leaves** the deployment's jurisdiction

Examples

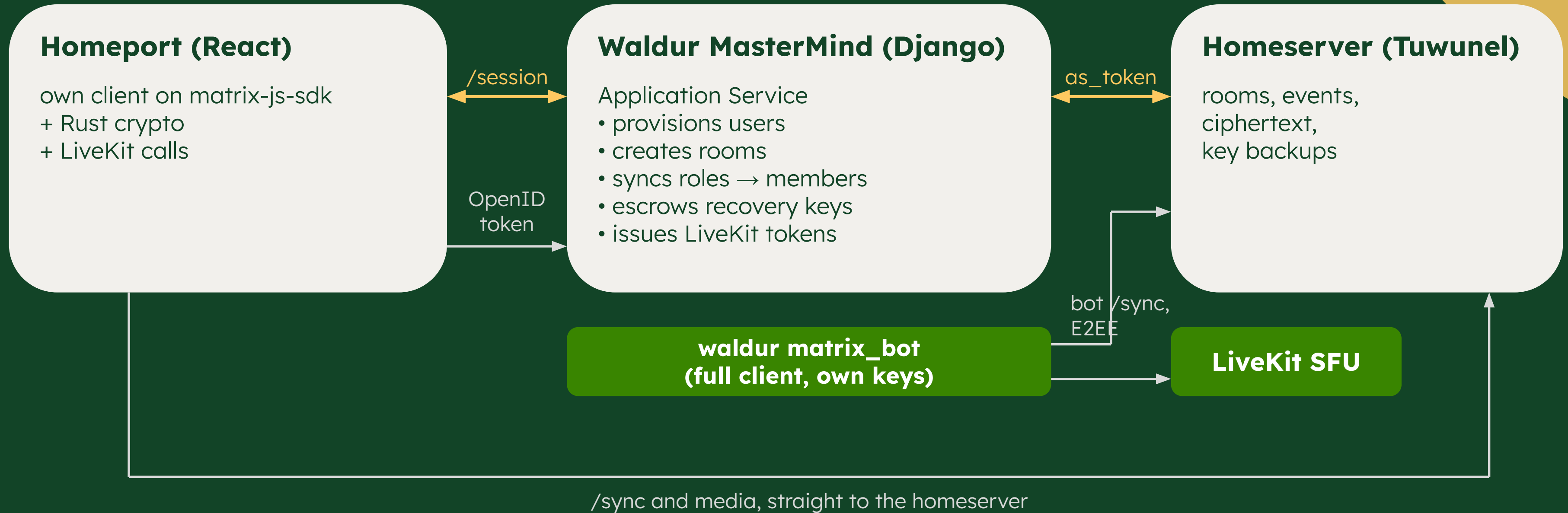
Tchap

BwMessenger

BundesMessenger

openDesk

Architecture



Waldur owns **identity and membership**. The homeserver owns **messages**. The browser owns the **keys**, and Waldur holds the **recovery key**.

Where the Application Service gets stretched

- Rooms exist before anyone is in them. Membership comes from Waldur roles, not from invites.
- Act as any user with `?user_id=` : join, leave, display name, device list, key backup. No device, no token, nothing to accept.
- Key backups written for people who have never opened a client.

Awkward: deleting a device needs UIA, which an appservice cannot do. So we log in on that device ID and call `/logout` from it.



Identity, SSO, zero-touch

Web, inside Waldur

POST `/api/matrix/session/` → appservice
login on a fresh device

Refresh tokens, **never stored** by Waldur

Each tab is a device; stale ones pruned nightly

Element / mobile

Homeserver OIDC against the same IdP

Same `sub`, same Matrix ID, same account

Or a generated password: shown once, stored
nowhere (one setting, oidc or password)

- We **refuse** to adopt accounts we didn't create.
- **Zero-touch:** Waldur registers itself on the homeserver at deploy time.



Membership sync: roles → membership & power levels

Power levels

100

@waldur-bot

the only 100

50

Project admin, staff, support

whoever may create the room

0

Any other project role

talk and join calls

role granted

→ invite + join-as-user + PL + notice

role revoked

→ kick, unless another role keeps access

user deactivated

→ log out all devices, kick, leave the call, lock account

project deleted

→ notice, everyone out, call ended, archive



Chat inside the Waldur UI

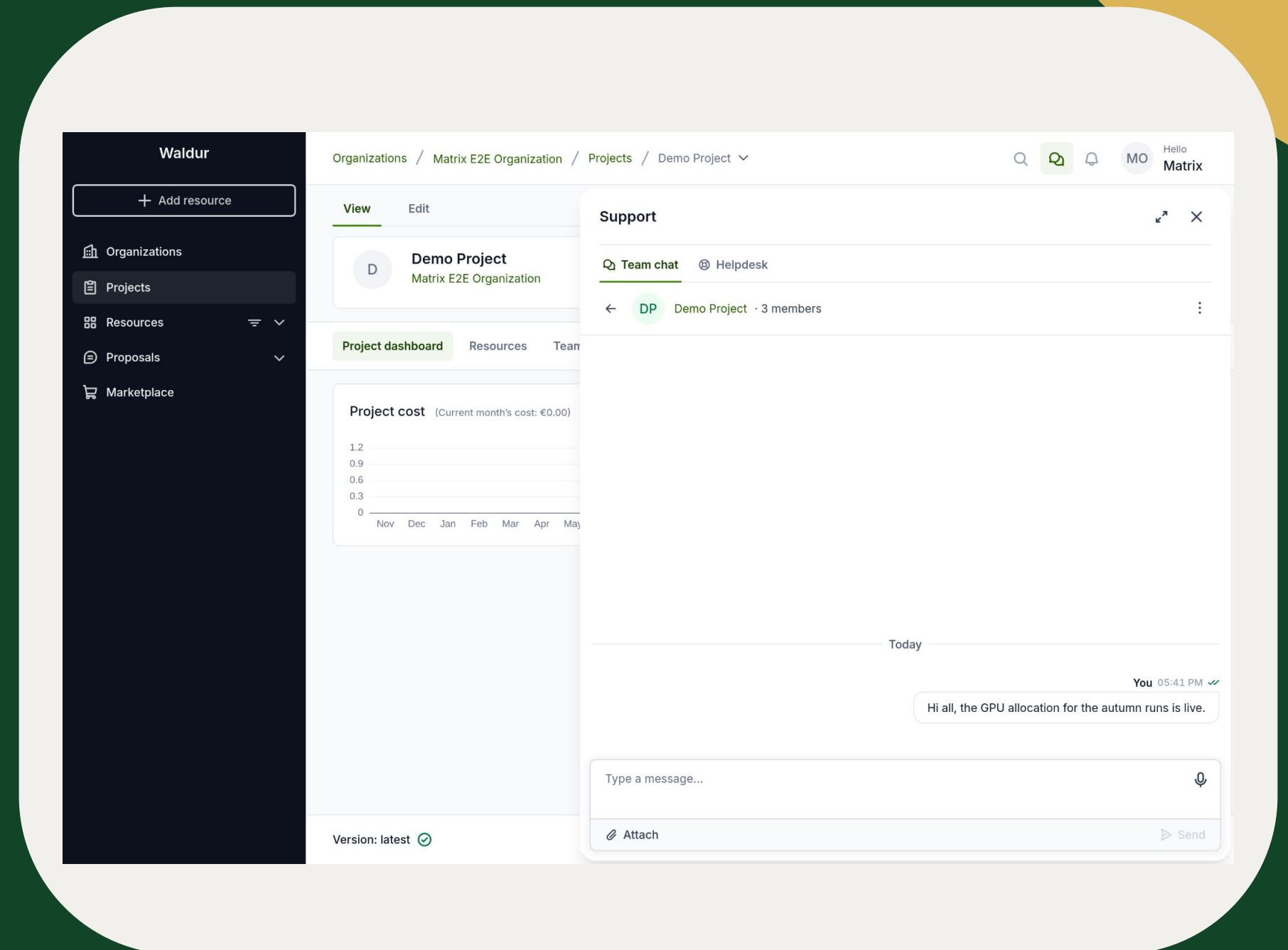
- Own React client on matrix-js-sdk 43 + Rust crypto.
- Not Element, not a widget
- Why: display names from Waldur, bundle budget, one UI

Have

Replies, edits,
reactions
Mentions, typing,
unread counts
Encrypted files,
voice messages
Calls

Traded away

Server-side search
DMs
Non-Waldur rooms



Prefer Element? Open Element: same account, same rooms.

E2EE with an operator-held recovery key

- Every room created with `m.room.encryption` (Megolm); the bot turns encryption on if it ever finds a room without it
- First session: browser bootstraps secret storage, cross-signing, key backup, dehydrated device. **Recovery key escrowed to Waldur before any key is uploaded**
- Every later session and Element on the phone unlock with it; keys only go to cross-signed devices

Threat model

The homeserver holds ciphertext; the operator, via Waldur, can decrypt.

E2EE protects the data from the server, the backups and the hoster, not from the operator. Keys live in Waldur's database, encrypted at rest, under the body that runs the platform.



History for new members

Year 1



Room created, encrypted.
Bot is a member from
day one

Year 2



Every Megolm session
lands in the bot's
Postgres crypto store

Year 3



You join. Bot writes the
room's sessions into your
key backup

- Only sessions from devices that were members
- Never overwriting keys you already have
- Membership re-checked with Waldur and the homeserver right before writing

A Server-side alternative to **MSC4268**



What got harder

1

E2EE vs the Application Service

/transactions only acks.
The bot runs its own /sync with its own keys; everything Waldur posts goes through an outbox.

2

Waldur is the call token service

MatrixRTC on LiveKit, Element Call-compatible.
The LiveKit token comes from Waldur: OpenID token, device, room membership, and the Waldur role.

Lose the role mid-call and LiveKit drops you.

3

Sessions are devices

Tuwunel keeps one refresh token per device, so two tabs logged each other out. Now every drawer session is its own device: 5-minute tokens, 10 per user, idle ones signed out nightly.



From **!status** to a ticket drafted by a model

Every project room has:

`!help` `!status` `!orders`
`!members` `!ticket`

Notices on role changes,
order states, staff joining

!ticket

Bot reads the conversation, fetches project facts, and a model on the operator's own infrastructure drafts a support ticket

Draft card posted as a reply; only the requester can Create / Edit / cancel, as themselves, via Waldur's API. No token in the room event

!ticket in the room

Bot decrypts last 20
messages + project
facts

**Model on operator
infra drafts**

Card in the room

**Requester: Create
/ Edit**

Commands advertised as MSC4391 state events, so any client that reads them gets a menu



Guardrails: a room is multi-party input

No tools

The model proposes, the human confirms. Worst case is a bad draft, not a bad action

Fail closed

Injection detected → no model call, template draft. PII redacted before the model sees anything

No Celery for commands

Decrypted text never touches the broker

Nothing lingers

Draft stored encrypted, erased on send / cancel / expiry; nothing of the conversation in logs or Sentry

Budgets

Per-user token quota, global budget, role gating. Over the limit: template draft

Demo

Recorded. What you will see:

- 01 | A project, its team, and the project's room in the chat drawer
- 02 | A call from the drawer; Element on a phone, same account
- 03 | !status, then !ticket → draft card → Create → the ticket in the helpdesk
- 04 | Role revoked → gone from the room. Diagnostics
- 05 | Panel meeting: 👍 in the call casts the vote, the privacy guard, the result in the room



5:12 · recorded

Close

Where we are

Live

Estonian Scientific
Computing Infrastructure

Testing

Estonian Government Cloud
Swiss EOSC Node
University of Cambridge HPC center

For operators

Admin dashboard

20 diagnostics checks

Zero-touch enrolment

**Backup / restore
runbooks**

What's next

Smaller improvements

Threads

Notifications: email, in-tab, push

Periodic reconciliation of rooms with roles

Making the integration more stable

More AI assistant commands: summarise a conversation, explain a failed order

Experiment: panel meetings

Waldur also runs research funding calls, and panels meet
A room per review panel, voting from the room

Thumbs-up or thumbs-down in the call casts the vote

Browser-local privacy guard hides the agenda when a phone appears, or you leave the frame

Proof of concept!

Federation between operators

Waldur already federates resources: national portal → Puhuri Core → LUMI.

The conversation about that resource should span them too. Each operator keeps its homeserver.

Project rooms are unfederated today, by design.

Sovereign conversations

The conversation about sensitive infrastructure, its history, the keys to it, and the model drafting inside it can all run on infrastructure **the operator controls**, with access derived from the **same authorisation as the resource**.

Close

Links

Backend

`github.com/waldur/waldur-mastermind`
`src/waldur_mastermind/matrix_chat/`

Frontend

`github.com/waldur/waldur-homeport`
`src/matrix/`

Docs

`docs.waldur.com` → Matrix chat overview

Contact

@hendrik.jaks:matrix.org · hendrik@opennodecloud.com · waldur.com
Feedback welcome: open an issue on either repo

Questions?