Hungryserv

A Homeserver Optimized for Unfederated Use-Cases

Sumner Evans

26 August 2022

Beeper

A bit about me

My name is Sumner, I'm a **software engineer at Beeper**.

- I graduated from Colorado School of Mines in 2019 with a master's in CS.
- I teach as an adjunct professor at my alma mater.
- I enjoy skiing, volleyball, and football (soccer).
- I'm a 4th degree black belt in ATA taekwondo.

I became interested in Matrix when I was the chair of the ACM chapter at Mines. I was looking for an open source chat platform for the club to use, and Matrix fit the bill!

A bit about me

My name is Sumner, I'm a **software engineer at Beeper**.

- I graduated from Colorado School of Mines in 2019 with a master's in CS.
- I teach as an adjunct professor at my alma mater.
- I enjoy skiing, volleyball, and football (soccer).
- I'm a 4th degree black belt in ATA taekwondo.

I became interested in Matrix when I was the chair of the ACM chapter at Mines. I was looking for an open source chat platform for the club to use, and Matrix fit the bill!

What I work on at Beeper

I am on the newly created *Scaling* team.

Our current objective is to prepare Beeper for rocket-ship growth.

I was previously part of the *Bridges* team.

Notable projects include:

- Writing the LinkedIn bridge
- Implementing massive stability improvements in the Signal bridge
- Implementing incremental infinite backfill in our WhatsApp and Facebook bridges

What I work on at Beeper

I am on the newly created *Scaling* team.

Our current objective is to prepare Beeper for rocket-ship growth.

I was previously part of the *Bridges* team.

Notable projects include:

- Writing the LinkedIn bridge
- Implementing massive stability improvements in the Signal bridge
- Implementing incremental infinite backfill in our WhatsApp and Facebook bridges

Overview

1. A bit about Beeper
2. A bit about Beeper's current architecture
3. Hungryserv

This talk is interactive!

If you have questions at any point, feel free to interrupt me.

A bit about Beeper

Beeper's mission

Our mission is to:

> *make it easy for everyone on Earth to chat with each other.*

We specifically chose the word "chat" rather than "communicate" because we are focusing on *people talking to one another*.

Beeper's mission

Our mission is to:

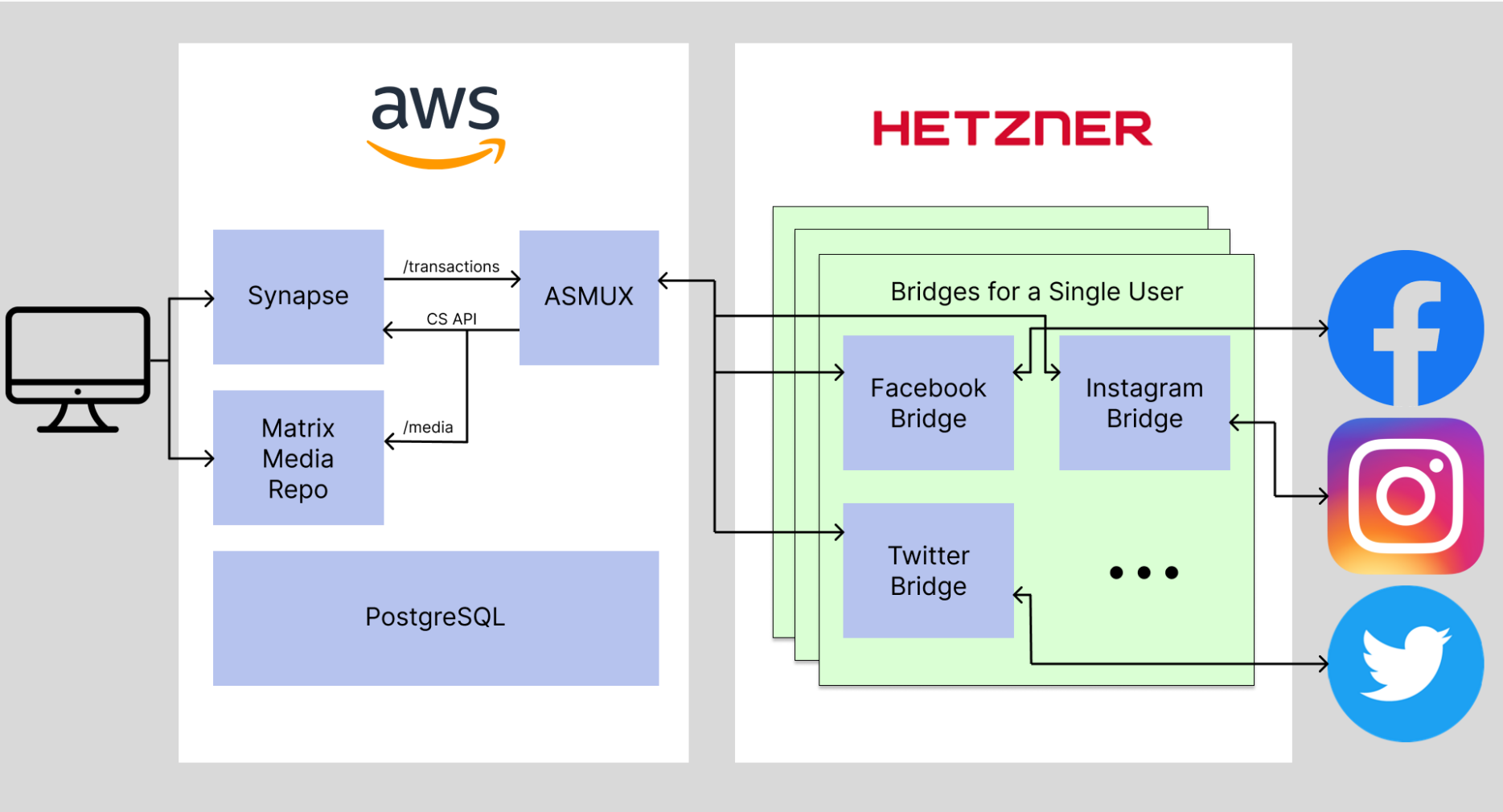
> *make it easy for everyone on Earth to chat with each other.*

We specifically chose the word "chat" rather than "communicate" because we are focusing on *people talking to one another*.

How we are getting there

A bit about Beeper’s current architecture

A diagram



Each user gets their own bridge for each network they connect.

Advantages of our architecture

- Each users' bridge is **isolated**
- We can deploy **different bridge versions** to different sets of users
- Bridges can be stopped and started individually
- Double puppeting

Advantages of our architecture

- Each users' bridge is **isolated**
- We can deploy **different bridge versions** to different sets of users
- Bridges can be stopped and started individually
- Double puppeting

Advantages of our architecture

- Each users' bridge is **isolated**
- We can deploy **different bridge versions** to different sets of users
- Bridges can be stopped and started individually
- Double puppeting

Advantages of our architecture

- Each users' bridge is **isolated**
- We can deploy **different bridge versions** to different sets of users
- Bridges can be stopped and started individually
- Double puppeting

Disadvantages of our architecture

- We have to run a **lot** of bridges
- Synapse is not designed for a dynamic numbers of bridges
- If two users join the same chat on an external network, we end up with two rooms with the same data
- We run Synapse on AWS which is relatively expensive
- Synapse becomes a bottleneck for all traffic

Disadvantages of our architecture

- We have to run a **lot** of bridges
- Synapse is not designed for a dynamic numbers of bridges
- If two users join the same chat on an external network, we end up with two rooms with the same data
- We run Synapse on AWS which is relatively expensive
- Synapse becomes a bottleneck for all traffic

Disadvantages of our architecture

- We have to run a **lot** of bridges
- Synapse is not designed for a dynamic numbers of bridges
- If two users join the same chat on an external network, we end up with two rooms with the same data
- We run Synapse on AWS which is relatively expensive
- Synapse becomes a bottleneck for all traffic

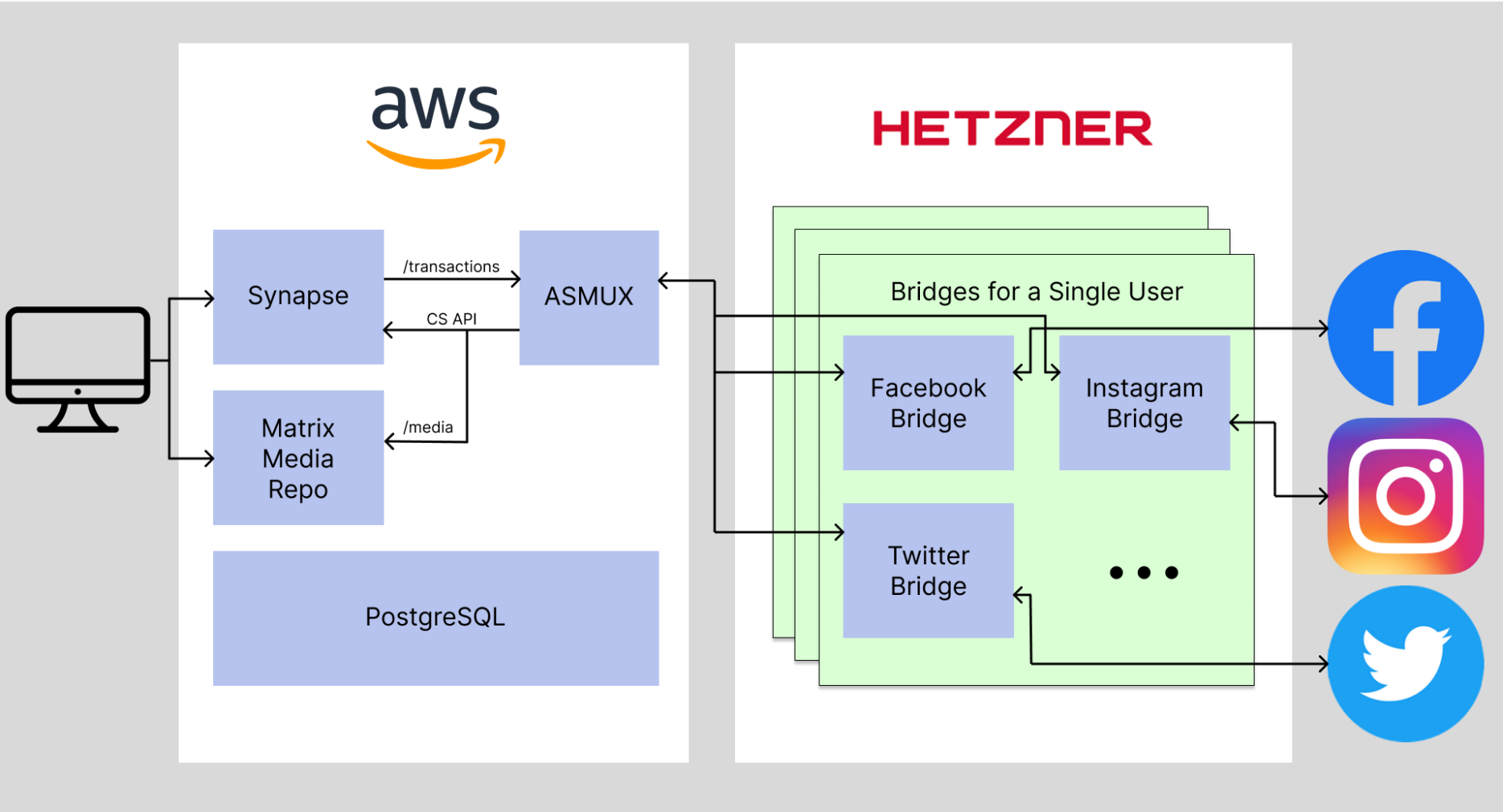
Disadvantages of our architecture

- We have to run a **lot** of bridges
- Synapse is not designed for a dynamic numbers of bridges
- If two users join the same chat on an external network, we end up with two rooms with the same data
- We run Synapse on AWS which is relatively expensive
- Synapse becomes a bottleneck for all traffic

Disadvantages of our architecture

- We have to run a **lot** of bridges
- Synapse is not designed for a dynamic numbers of bridges
- If two users join the same chat on an external network, we end up with two rooms with the same data
- We run Synapse on AWS which is relatively expensive
- Synapse becomes a bottleneck for all traffic

Back to the diagram



The core software in this architecture is Synapse.

Stats

Because we encourage users to connect every chat network, we have a lot of puppet users.

On average, each user brings 4716 puppeted users.

All of these puppets represent real people on other networks, so they each generate their own traffic.

On average, each user and their puppets collectively account for 100k events!

None of that traffic is federated, but it has to go to Synapse which is designed with federation front-and-center!

Stats

Because we encourage users to connect every chat network, we have a lot of puppet users.

On average, each user brings 4716 puppeted users.

All of these puppets represent real people on other networks, so they each generate their own traffic.

On average, each user and their puppets collectively account for 100k events!

None of that traffic is federated, but it has to go to Synapse which is designed with federation front-and-center!

Stats

Because we encourage users to connect every chat network, we have a lot of puppet users.

On average, each user brings 4716 puppeted users.

All of these puppets represent real people on other networks, so they each generate their own traffic.

On average, each user and their puppets collectively account for 100k events!

None of that traffic is federated, but it has to go to Synapse which is designed with federation front-and-center!

A few more notes

- **Synapse is aggressively federated**
  Synapse has no special handling of federated vs unfederated rooms.
- **Synapse does not handle message floods well**
  Synapse falls over when users bridge some large Telegram chats and Discord servers.
- **Deleting rooms from Synapse is hard**
  We end up with lots of *dead* rooms when users delete their bridges and never turn it back on (or need a hard bridge reset).

A few more notes

- **Synapse is aggressively federated**
  Synapse has no special handling of federated vs unfederated rooms.
- **Synapse does not handle message floods well**
  Synapse falls over when users bridge some large Telegram chats and Discord servers.
- **Deleting rooms from Synapse is hard**
  We end up with lots of *dead* rooms when users delete their bridges and never turn it back on (or need a hard bridge reset).

A few more notes

- **Synapse is aggressively federated**
  Synapse has no special handling of federated vs unfederated rooms.
- **Synapse does not handle message floods well**
  Synapse falls over when users bridge some large Telegram chats and Discord servers.
- **Deleting rooms from Synapse is hard**
  We end up with lots of *dead* rooms when users delete their bridges and never turn it back on (or need a hard bridge reset).

Solution: build a homeserver optimized for unfederated bridge traffic

Hungryserv

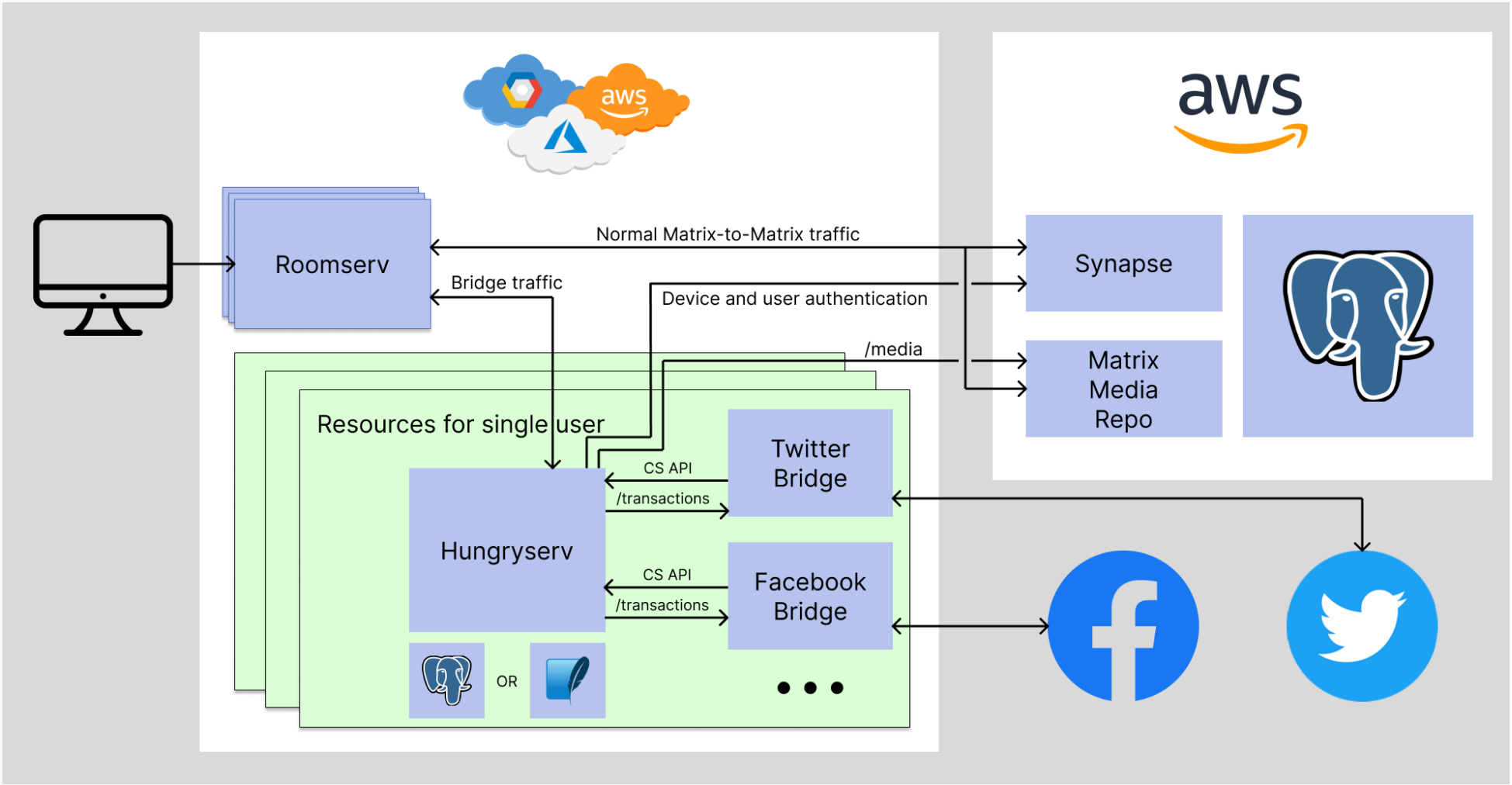
Why is it hungry?

Because it’s unfederated!

Why is it hungry?

Because it’s unfederated!

Our new architecture



Why unfederated?

Each bridge belongs to a single user, so federation is unnecessary.

> *When your DAG is a linked-list, don't store it as a DAG.*

Being unfederated means that we can avoid implementing many of the event auth rules.

- Event storage requires less metadata
- Deletions can be optimized
- Infinite backfill becomes an simple batch SQL operation

Why unfederated?

Each bridge belongs to a single user, so federation is unnecessary.

> *When your DAG is a linked-list, don't store it as a DAG.*

Being unfederated means that we can avoid implementing many of the event auth rules.

- Event storage requires less metadata
- Deletions can be optimized
- Infinite backfill becomes an simple batch SQL operation

Why unfederated?

Each bridge belongs to a single user, so federation is unnecessary.

> *When your DAG is a linked-list, don't store it as a DAG.*

Being unfederated means that we can avoid implementing many of the event auth rules.

- Event storage requires less metadata
- Deletions can be optimized
- Infinite backfill becomes an simple batch SQL operation

Goals, and non-goals

Goals

- Be fast
- Be compatible with Beeper clients (and mostly compatible with Element clients)
- Be API-compatible with the Appservice API for bridges

Non-goals

- Be federated
- Be fully spec compliant
- Be P2P

Goals, and non-goals

Goals

- Be fast
- Be compatible with Beeper clients (and mostly compatible with Element clients)
- Be API-compatible with the Appservice API for bridges

Non-goals

- Be federated
- Be fully spec compliant
- Be P2P

Demo!

Questions?