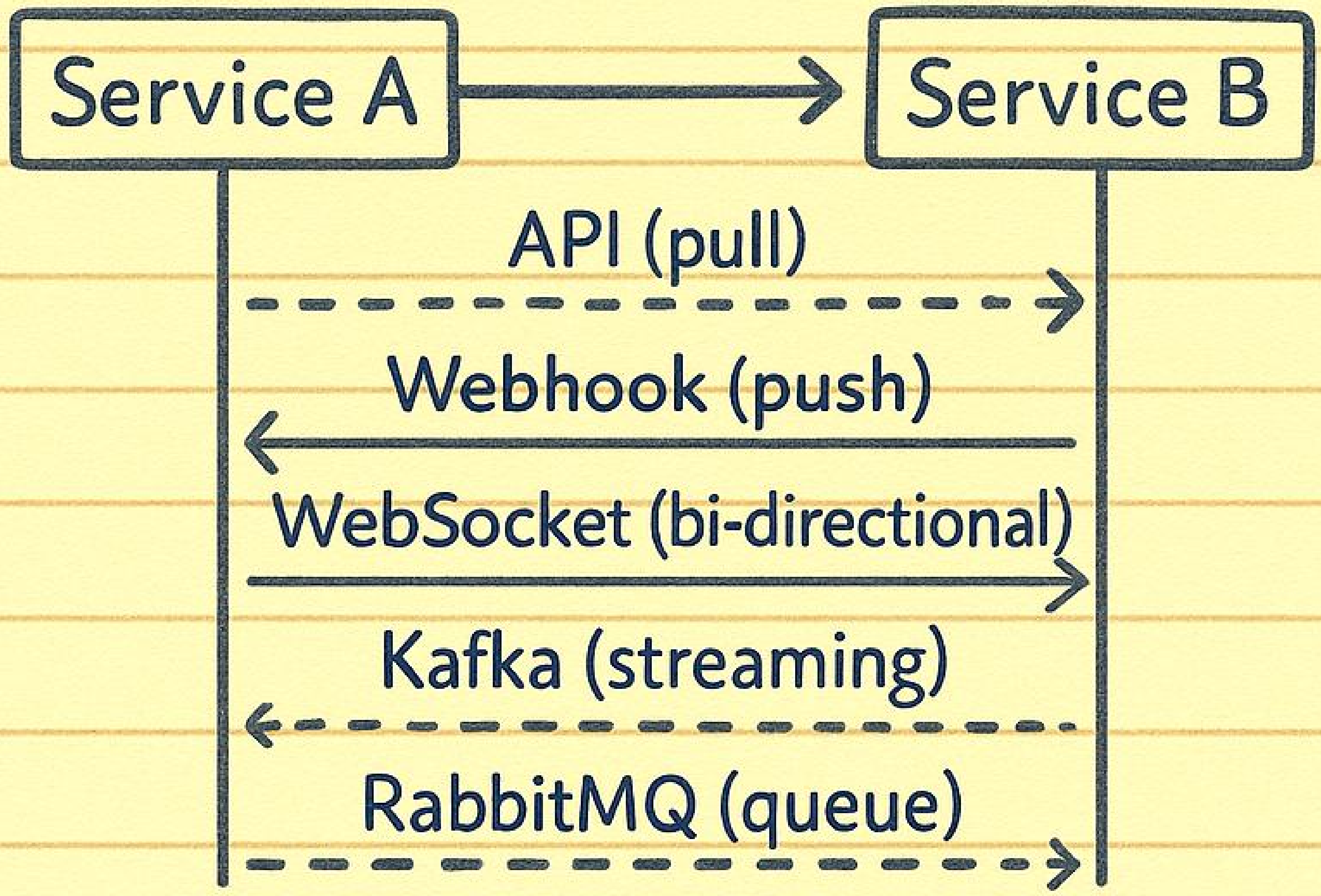
HOW DO SERVICES TALK TO EACH OTHER?

It's not just APIs.

API	Ask me when you need something.
Webhook	I'll call you when something changes.
WebSocket	Let's stay connected forever.
Kafka	I'll keep talking. Tune in when you want
RabbitMQ	Message delivered. Pick it up anytime.

Swipe → Break down these 5 in the simplest way possible (with jokes too)

WHY SO MANY COMMUNICATION METHODS? 🤔



Service A

Service B

API (pull)

Webhook (push)

WebSocket (bi-directional)

Kafka (streaming)

RabbitMQ (queue)

Need quick info? → Use an API

Want a ping on updates? → Use a Webhook

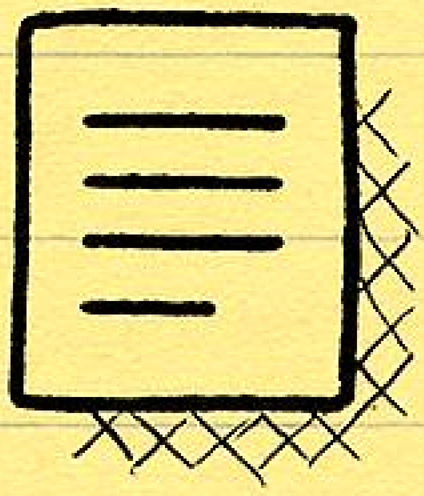
Need real-time sync? → Go for WebSocket

Streaming events nonstop? → Hello Kafka

Slow consumers? → Queue it with RabbitMQ

Different tools. Different problems.
But all trying to make your services
talk better 🧠

API — The Classic Ask-and-Get

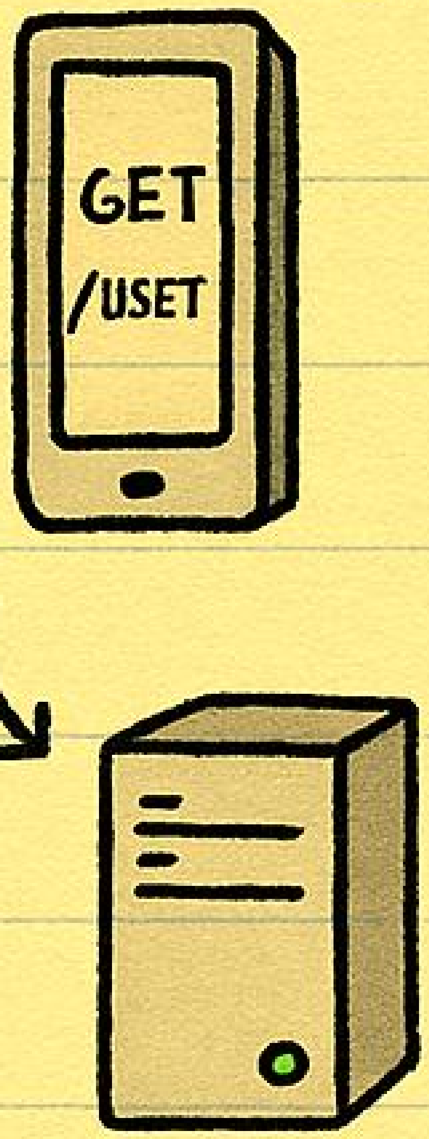


What is it?

"Client asks, server replies."
A synchronous, request-respons model.
No state is remembered between calls.

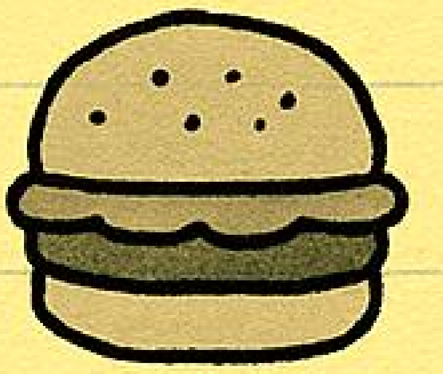
How it works:

You → "Hey, give me the data!"
Server → "Here it is. Goodbye."



Best for:

- ✓ Mobile/web apps fetching user data
- ✓ Microservice A asking Microservice B for detasils
- ✓ Frontend-backend communication



Analogy: Like ordering food when you're hungry. You ask. You wait. You get.

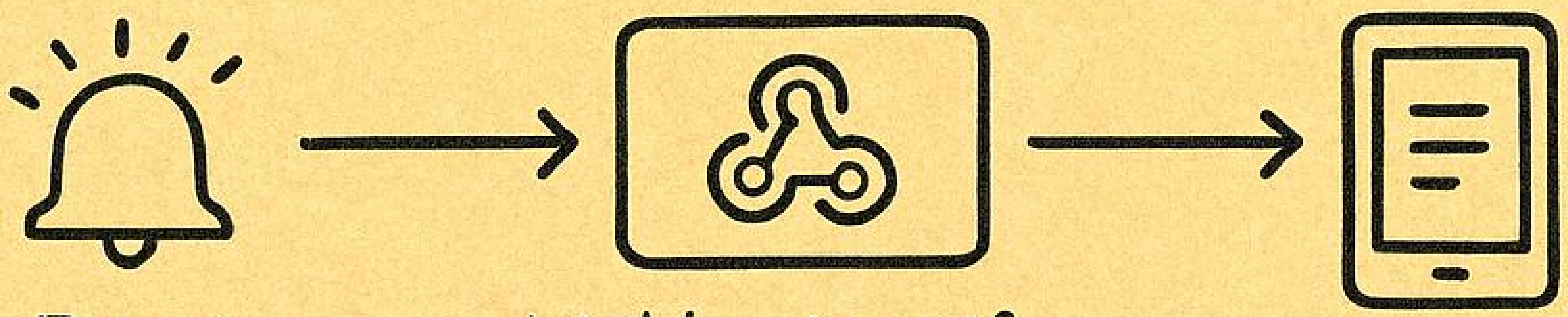
Keep it stateless. Keep it quick. APIs love simplicity.

WEBHOOK – DON'T CALL ME, I'LL CALL YOU ☎

WHAT IS IT?

"The server notifies you when something changes." Event-driven, async communication – no need to ask again and again.

HOW IT WORKS:



Event happens

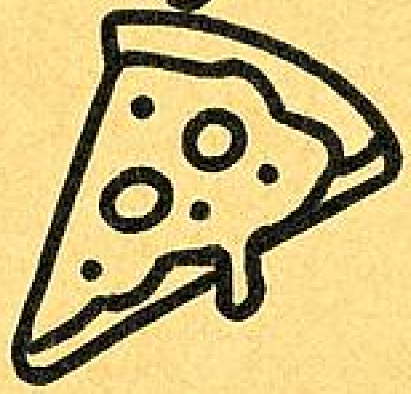
Webhook notifies automatically

You react instantly

BEST FOR:

✓ Payment success notifications
✓ GitHub → Jenkins triggers
✓ Slack alerts for events

ANALOGY: Like getting a pizza delivery SMS: "Hey! Your order is outside."



Tip: Efficient. Instant. But depends on the sender showing up on time.

WebSocket – Let's Stay on This Call Forever

What is it?

A persistent, bi-directional connection. Once the connection is open, both client and server can send messages anytime.

How it works:

- Connected once.
- Talk freely.
  Send + receive anytime

Best for:

- Live chat apps (WhatsApp Web, Messenger)
- Real-time stock & trading dashboards
- Collaborative tools (Google Docs, Figma)

Analogy: Like staying on a video call all day. You talk. They talk. No disconnection needed.

Kafka – I'll Keep Talking. You Just Tune In.

What is it? A distributed, fault-tolerant streaming platform. Data is published to a topic. Any number of consumers can subscribe and consume independently.

Best for:

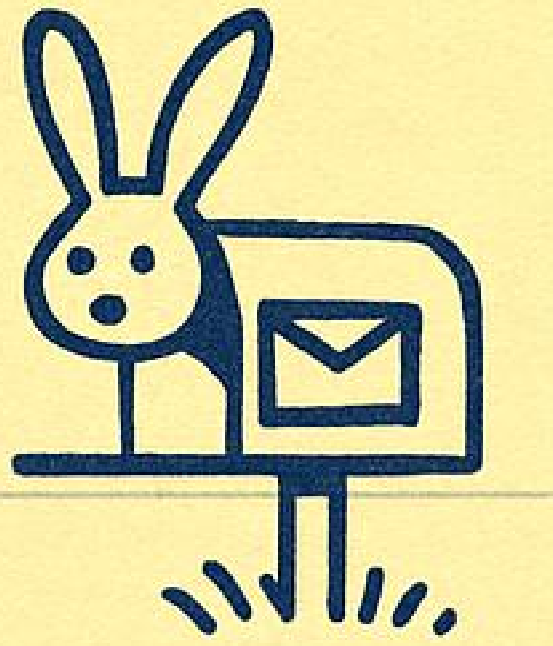
✓ Real-time analytics
✓ Event-driven microservices
✓ Logging, metrics, and clickstrem data

Analogy:

Like a radio station.
One keeps broadcasting.
Listeners tune in when they want.

Tip: High speed. High volume. High reliability. Just don't expect a reply. 😎

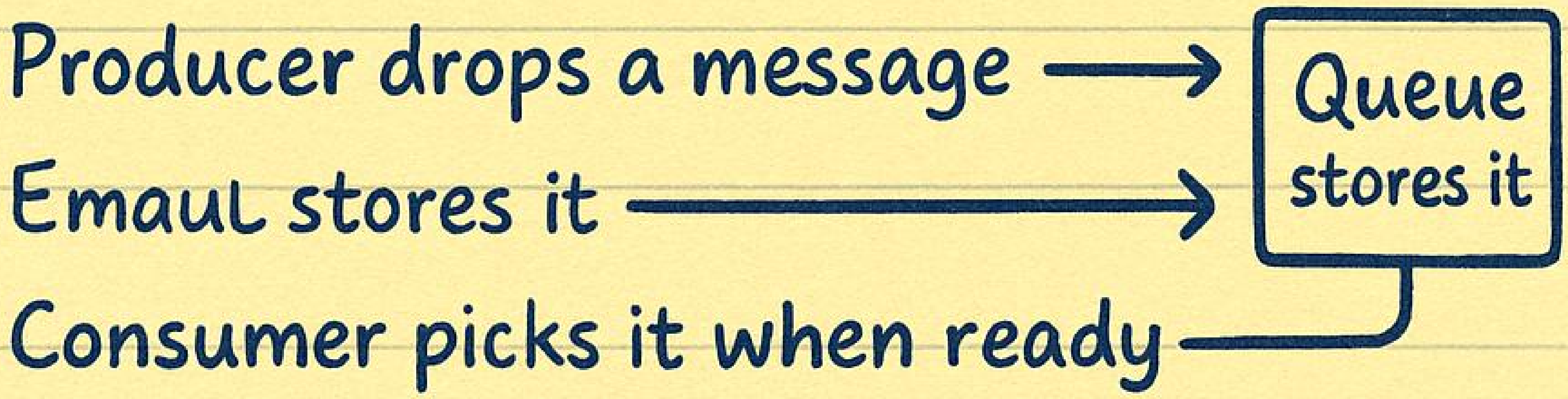
RabbitNQ – I Got Your Message. Chill.



What is it?

A message broker that queues messages until the consumer is ready. Asynchronous. Reliable. Works great when sender & receiver don't move at the same speed.

How it works:



Producer drops a message → Queue stores it

Emaul stores it →

Consumer picks it when ready

Like leaving a voice message.
They'll hear it when they're ready.

Perfect for slow consumers
or unreliable systems.
Because life gets busy.

Let’s Recap...
Who Talks How?

API → Ask when needed

Webhook → I’lll call you Hi

WebSocket → I’ll keep talking

Kafka → I’ll keep broadcasting

RabbitMQ → Here’s your message, pick it when ready.

Every method has its moment.
Choose wisely. Architect smartly.

Found it useful? Save + share this with your dev team. Or just drop a 💬 and tell me which one you’ve used the most.