



Your agent needs more than an API

Pieter Cailliau - VP of Product @ Redis

© 2026 Redis Ltd. All rights reserved.

About me

Produced in Belgium

- `isinstance(me, SoftwareEngineer) # @ TomTom`
- Consultant @ neo4j
- Solutions Architect @ Redis
- Product Manager @ Redis

- Triathlon, but “I’m only really alive when I’m on my bike”
- Recently relocated to Girona

- @cailliaup



Redis © 2026 Redis Ltd. All rights reserved.

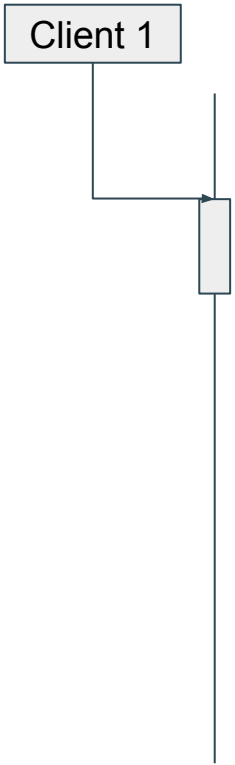
move on Pieter … talk me Redis

Redis © 2024 Redis Ltd. All rights reserved.

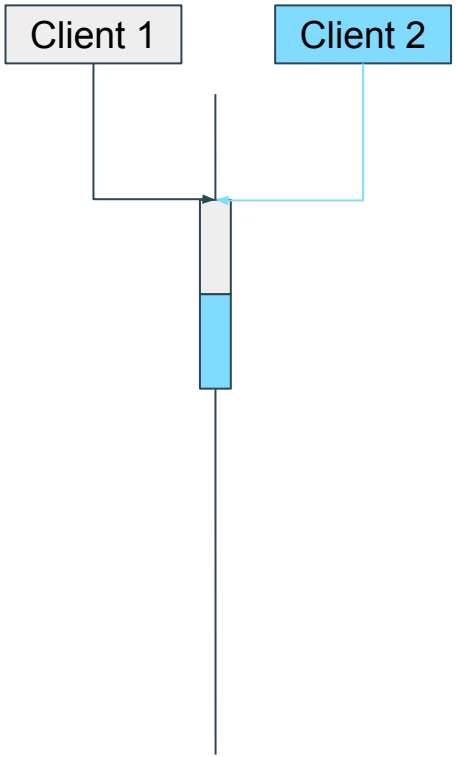
Redis (/ˈrɛdɪs/;[8][9] **Remote Dictionary Server**)[8] is an in-memory key–value database, used as a distributed cache and message broker, with optional durability.[10] Because it holds all data in memory and because of its design, Redis offers low-latency reads and writes, making it particularly suitable for use cases that require a cache.

Redis © 2026 Redis Ltd. All rights reserved.

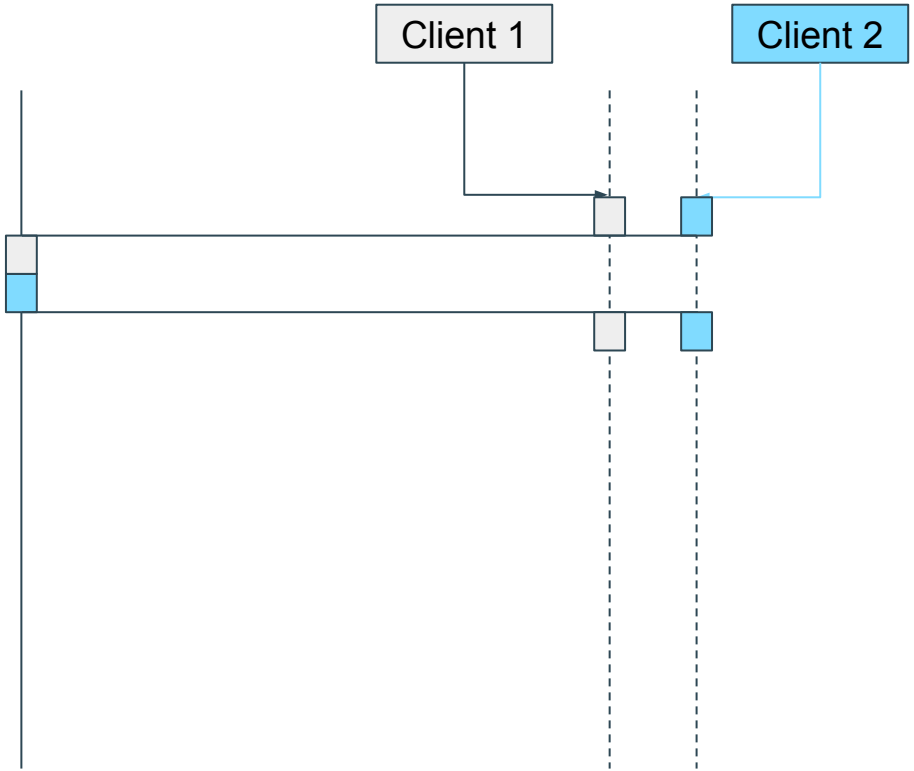
Redis - main thread



Redis - main thread

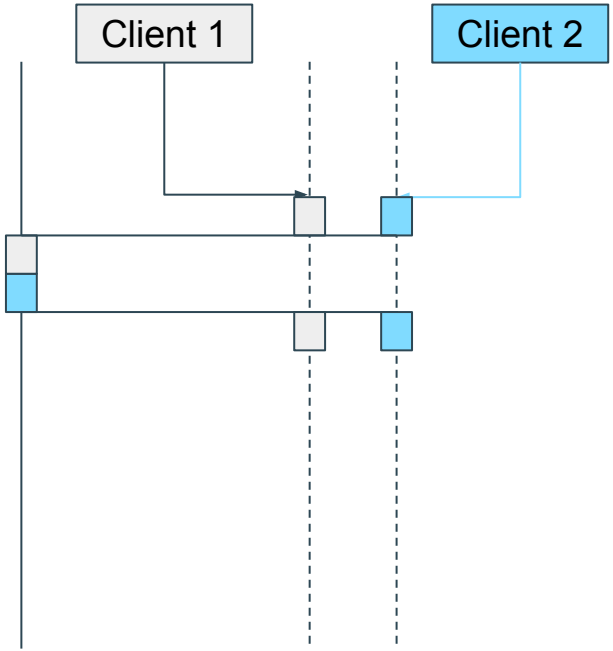


Redis - Scale up - io-threads

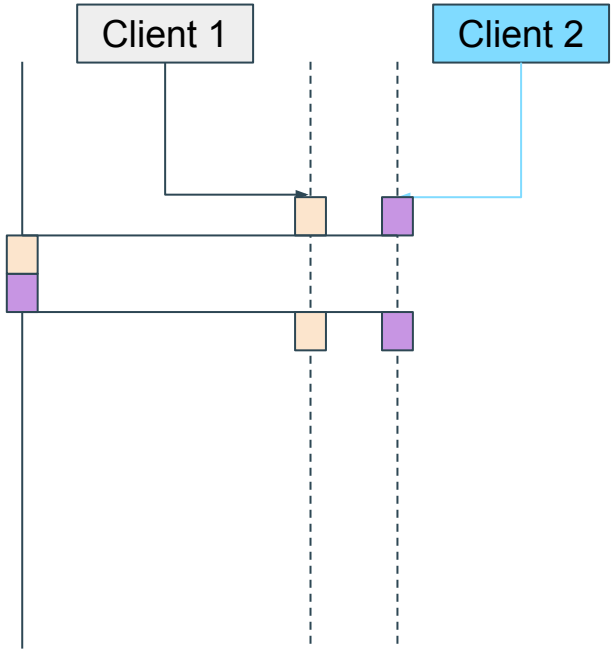


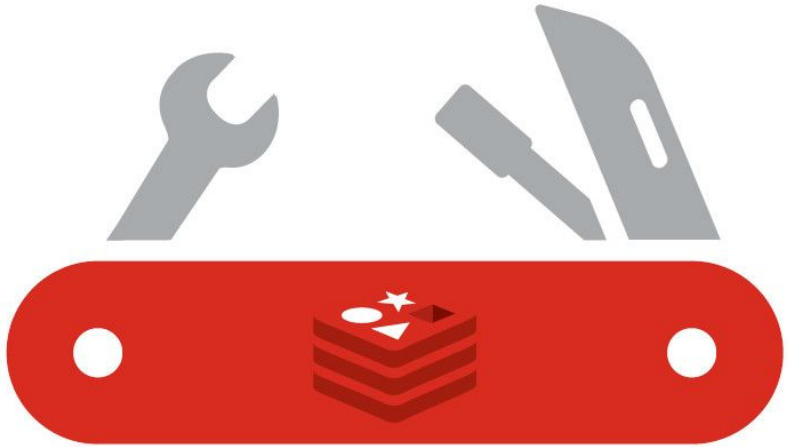
Redis - Scale out - Clustering

Redis - P1 - Slots 0- 8191



Redis - P2 - Slots 8192 - 16383





Redis © 2026 Redis Ltd. All rights reserved.





Our goal is to make the most **common patterns for Redis in FastAPI** available to everyone **whilst avoiding tons of boilerplate code**.

Redis © 2026 Redis Ltd. All rights reserved.

About the FastAPI Redis SDK design

LIFESPAN

Easy way to attach the Redis SDK lifecycle to the lifespan of the FastAPI app



ASYNC FIRST

Follows the FastAPI principles of async first, but still support sync endpoints



DEPENDENCY INJECTION

DI all the way
- factories for standard behaviour
- *Backend dependencies for advanced recipes



Redis © 2024 Redis Ltd. All rights reserved.

Right… but we’re going to build an agent, no?

© 2024 Redis Ltd. All rights reserved.

"Nah man, oncall shift isn't too bad.
You just have to get used to it."
Michael, age 24:



Redis © 2026 Redis Ltd. All rights reserved.

On call agent requirements

Let’s help Michael

- Michael needs to drop off his kids
  - Needs to continue working on his mobile device
- Michael wants his life back
  - Fast metrics lookups - cache where possible
- Michael needs to go shopping in a Faraday Cage
  - No agent responses are lost when connection drops
- Michael’s on call time ends at some point
  - Multi user app
- Though Michael is on a TIP (Token Improvement Plan), his boss has a token budget
  - Limit the number of calls

Redis © 2026 Redis Ltd. All rights reserved.

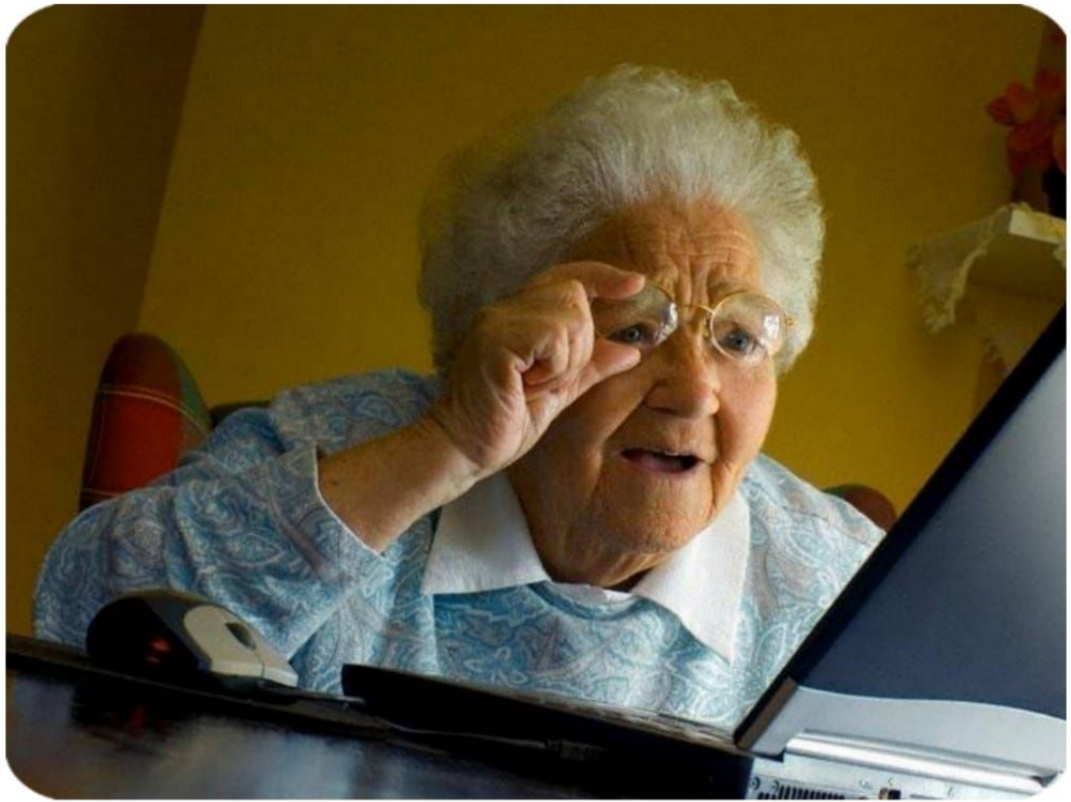
Show me the code already

© 2024 Redis Ltd. All rights reserved.

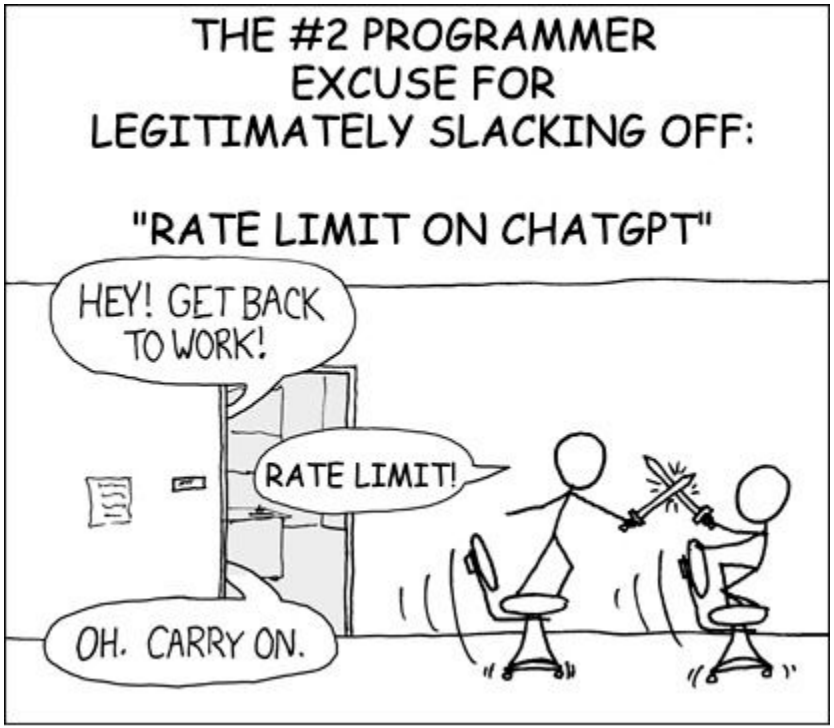


© 2024 Redis Ltd. All rights reserved.

Rate limit exceeded? What does that mean?



Redis © 2024 Redis Ltd. All rights reserved.



© 2024 Redis Ltd. All rights reserved.

Really, show me the code already!

© 2024 Redis Ltd. All rights reserved.

Session management



1 Session modeling with STRING

- Standard session management

2 Unlock session modeling with HASH

- Avoid serialization/deserialization
- Reduce bandwidth consumption

3 Unlock session modeling with HASH and hash field expiration

- More control



4 Unlock HASH-based searchable sessions

- Cross-session searches
- Search by product, location, timestamp, etc.

5 Unlock session modeling with JSON

- Supports nested objects
- JSONPath session lookups

6 Unlock JSON-searchable sessions

- Index at arbitrary depth
- Multi-value indexing

Redis © 2024 Redis Ltd. All rights reserved.

Streams

Redis © 2024 Redis Ltd. All rights reserved.



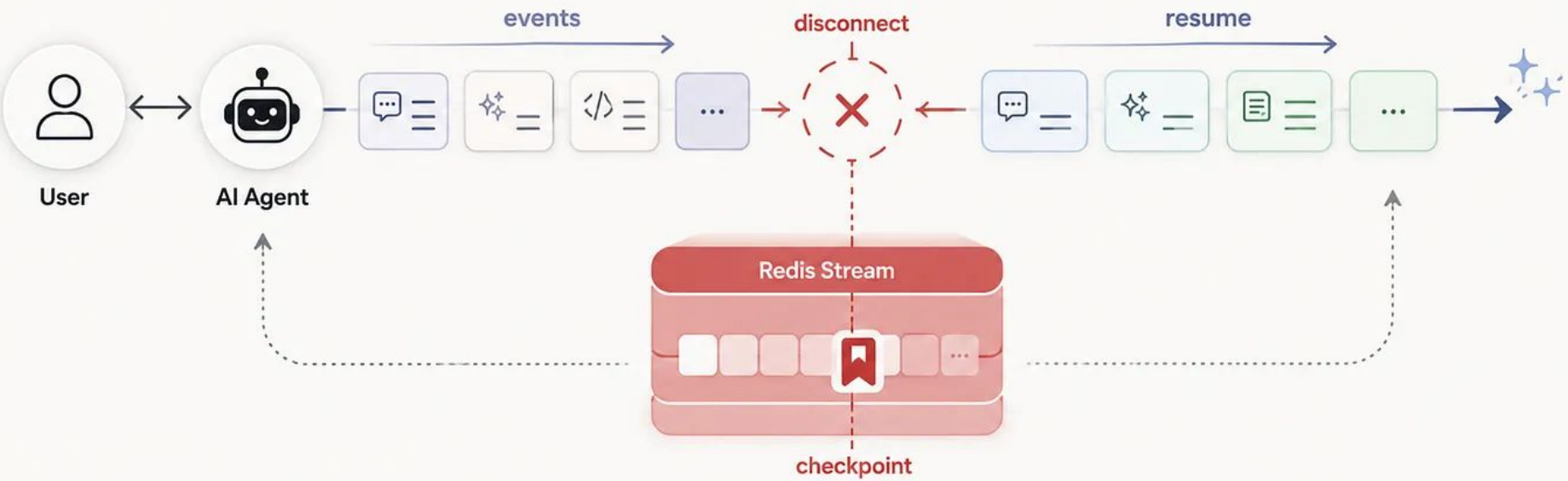
© 2026 Redis Ltd. All rights reserved.



Redis © 2026 Redis Ltd. All rights reserved.

Building Resumable Real-Time LLM Streaming

with Redis Streams





Pieter Cailliau • You
Product Leader at Redis
Sep 10 • Edited

Great engineering write-up from the team at Glance on building resumable, real-time LLM streaming.

What I particularly liked is that this isn't a "we chose Redis" story. They start with the problem, work through direct SSE, gRPC streaming and Kafka, and eventually land on Redis Streams.

The interesting part for me is the pattern behind it. Agents don't just need to stream responses to humans. They need to maintain state across disconnects, infrastructure changes, tool calls and long-running work — and let the human pick up exactly where things left off.

We're seeing this pattern more and more in agentic architectures: using Redis as the state layer between agents and the humans interacting with them.

As Aditya Singh explains in his blog post, a Redis Stream per turn gives each interaction an ordered, replayable history. The agent can keep working independently of the connection, and the client can reconnect and resume without losing the thread.

Really nice example of Redis Streams in a real-world agent architecture.
Worth a read: https://lnkd.in/dJ5hcNg4



Building Resumable Real-Time LLM Streaming with Redis Streams
engg.glance.com

503 7 24

893,387 impressions View analytics

Redis © 2026 Redis Ltd. All rights reserved.

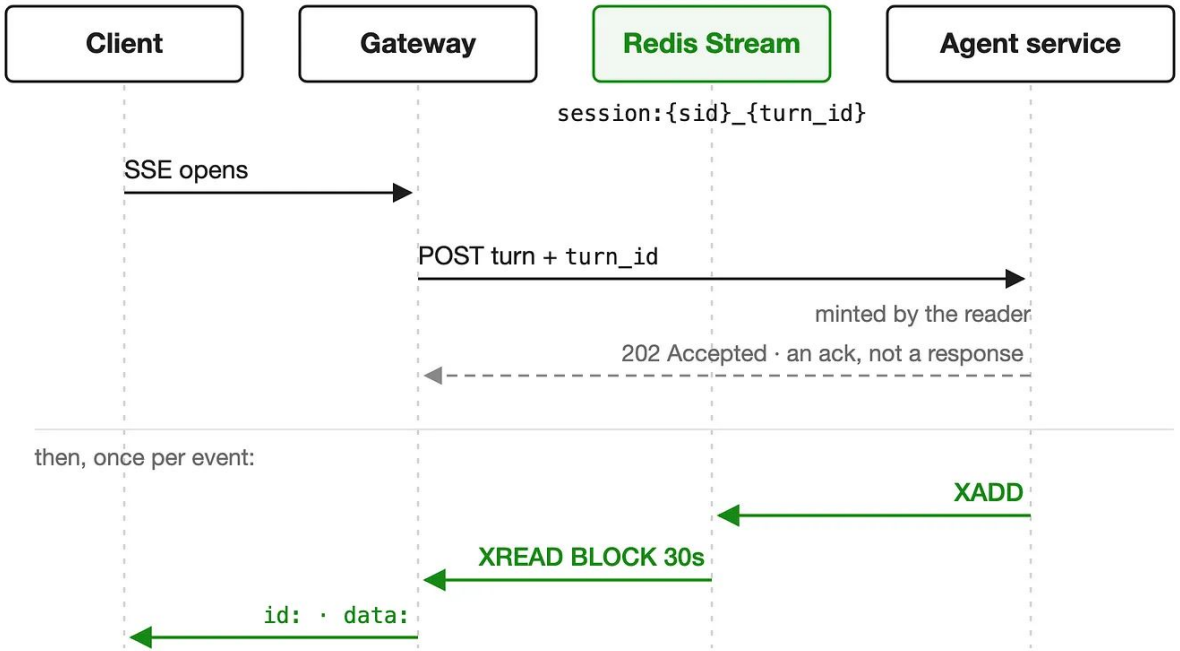
What We Wanted

Four constraints shaped the design.

1. **Server-side streaming.** A turn is not one response. It emits events over tens of seconds: tool selections, tool results, progress updates, and the final answer. The client needs to show motion throughout.
2. **Resumption at turn granularity.** Our clients live on consumer networks. If one drops mid-turn, it should reconnect and continue *where it left off* without restarting or losing events.
3. **No topology coupling.** The gateway should not need to know *which* agent instance is executing a turn. Any gateway pod should be able to serve any turn.
4. **Survival across deploys.** A rollout should not sever every in-flight turn. Events cannot live only inside a process that may disappear, and reconnecting cannot depend on reaching the machine that started the turn.

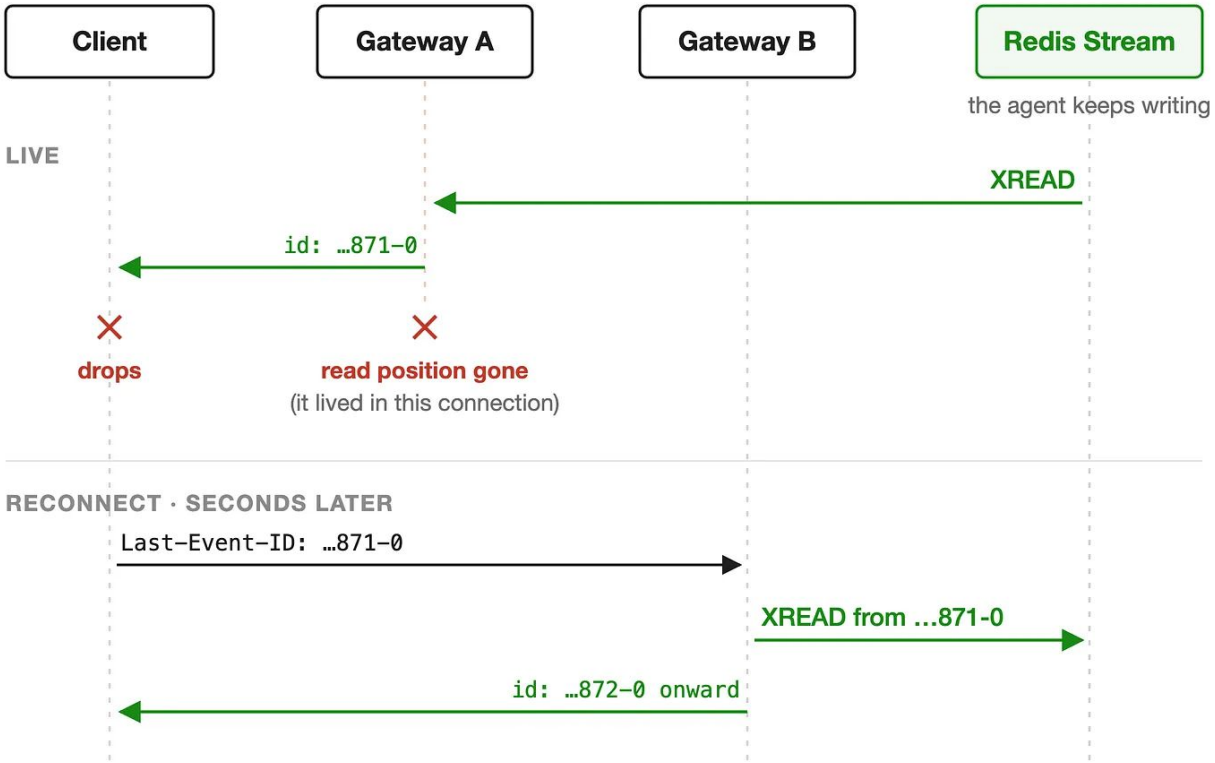
Constraint 2 turned out to be the important one. It sounds like a streaming requirement, but it is really about **addressability**.

© 2026 Redis Ltd. All rights reserved.



The entry ID is the cursor. No connection between the two services outlives a few milliseconds.

© 2026 Redis Ltd. All rights reserved.



A's position was never the one that mattered. B resumes from the client's ID, and never learns that A existed or which agent pod is writing.

Redis © 2026 Redis Ltd. All rights reserved.

So we started to generalise this in the FastAPI Redis SDK

beta

© 2024 Redis Ltd. All rights reserved.

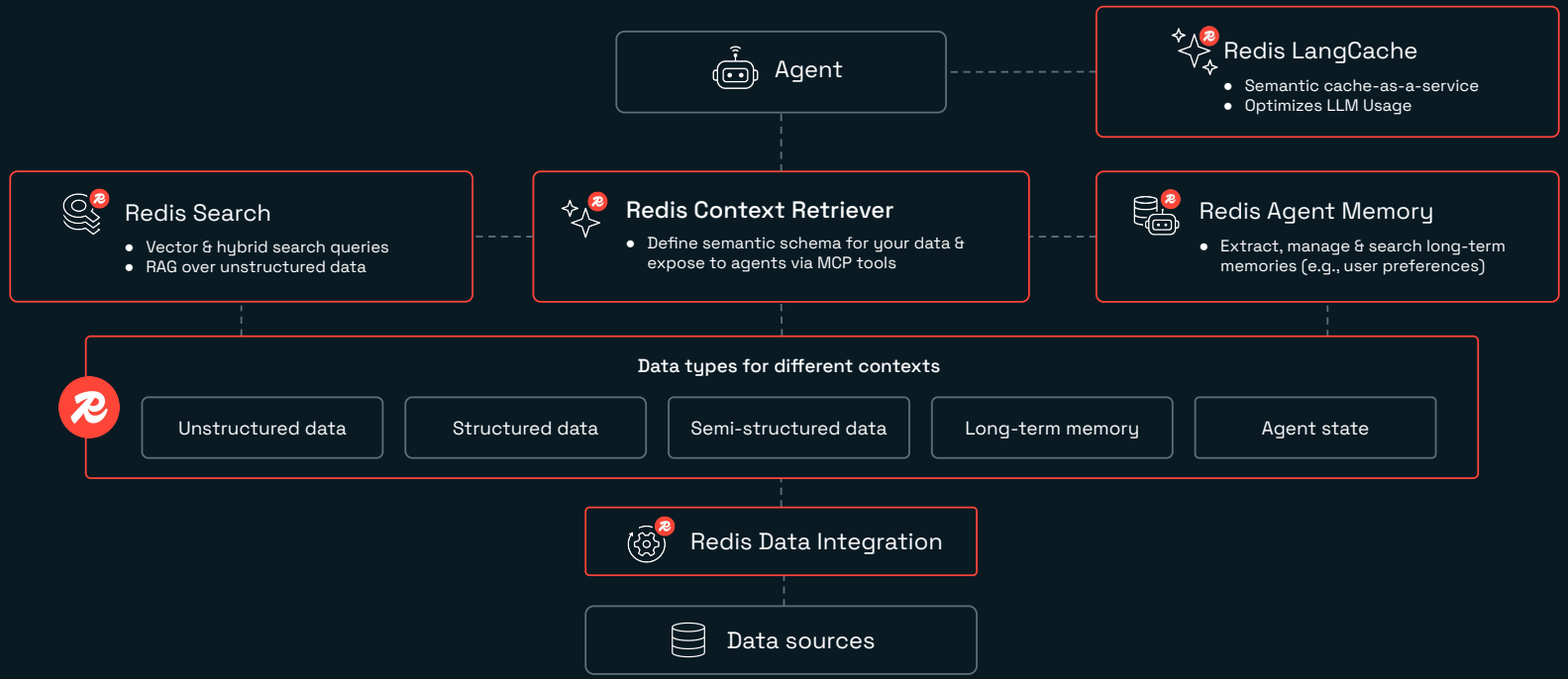


Stop assembling your agents' context layer. Introducing Redis Iris.

Powered by the Redis you already love

Redis © 2026 Redis Ltd. All rights reserved.

Redis is a real-time context engine built for fast, accurate AI apps.



Redis © 2026 Redis Ltd. All rights reserved.

Agent Memory makes your context compounding.

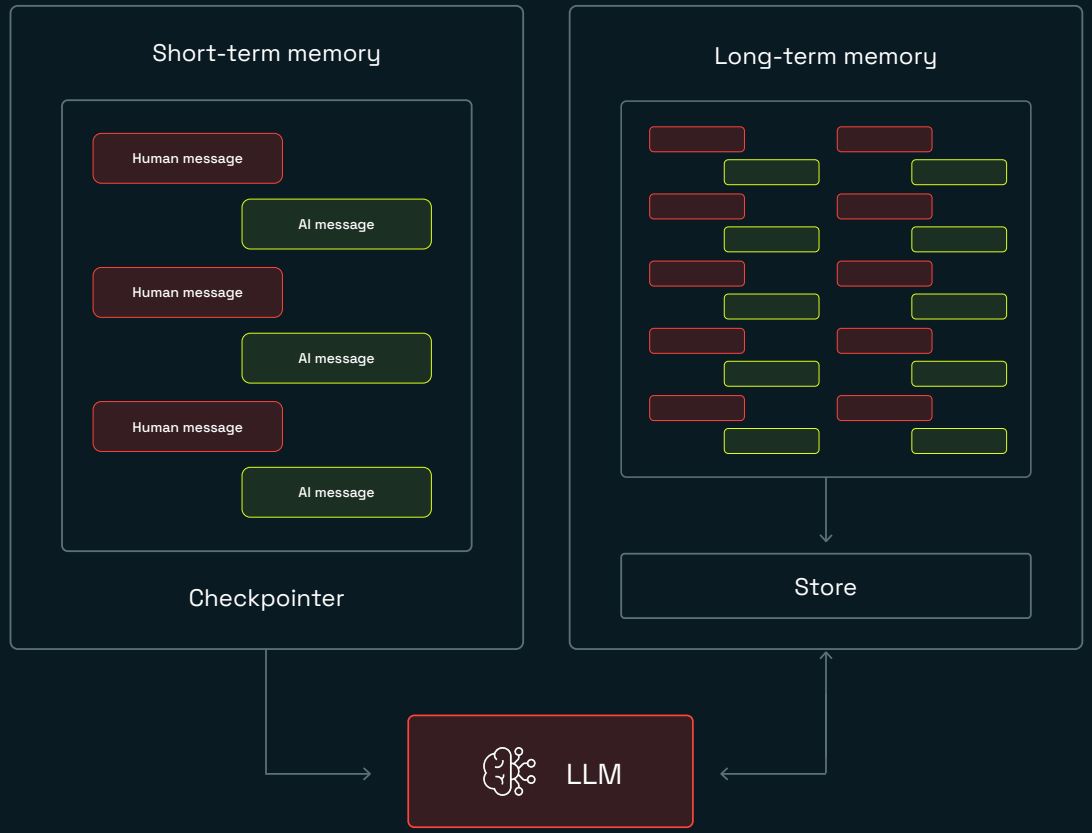
Store history and preferences, and retrieve key details for agents and customers.

Short-term memory is typically isolated per user and per session.

- → Agent state
- → Chat history
- → Summaries

Long-term memory is typically shared across sessions and sometimes across users.

- → Extracted preferences
- → Durable facts



Redis © 2026 Redis Ltd. All rights reserved.

What’s next for FastAPI Redis SDK?

In the short term …

- Add task queue scheduler support
- Distributed locks
- Redis Iris integration
- GA

Looking for contributors!

© 2024 Redis Ltd. All rights reserved.

Redis

FastAPI

Redis Iris



FastAPI Redis SDK



Thank you.

© 2026 Redis Ltd. All rights reserved.