Agentic AI Foundation

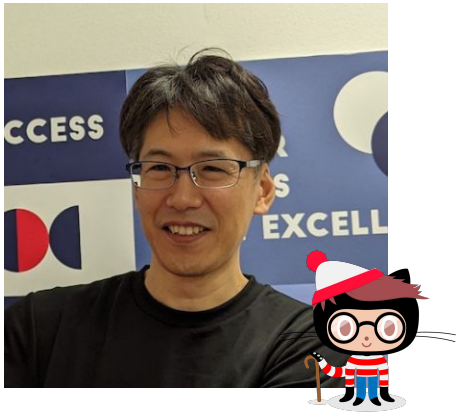
AGNTCon + MCPCon Japan

Where the **agentic stack** is **being built**.

Externalizing Agent State

Memory and File System Management for Sandboxed Coding Agent

Tadatoshi Sekiguchi, PingCAP



Tadatoshi Sekiguchi
X: @bohnen

Technology Evangelist
PingCAP

Agenda

In this session, we will discuss the challenges of serving AI agents, particularly around persistence. We will also introduce mem9 and drive9, open-source infrastructure designed to solve these challenges.

- Challenges of Agent Persistence
- mem9: Memory Persistence
- drive9: Artifact Persistence
- Summary

Challenges of Agent Persistence

When serving AI agents as a service, persisting user data becomes a key challenge. Local storage introduces issues with persistence and data isolation.

01

Lack of Long-Term Continuity

With local storage, which modern harnesses rely on, states, memories, and artifacts are lost when the environment stops. Scalability and fault tolerance are also key challenges.

02

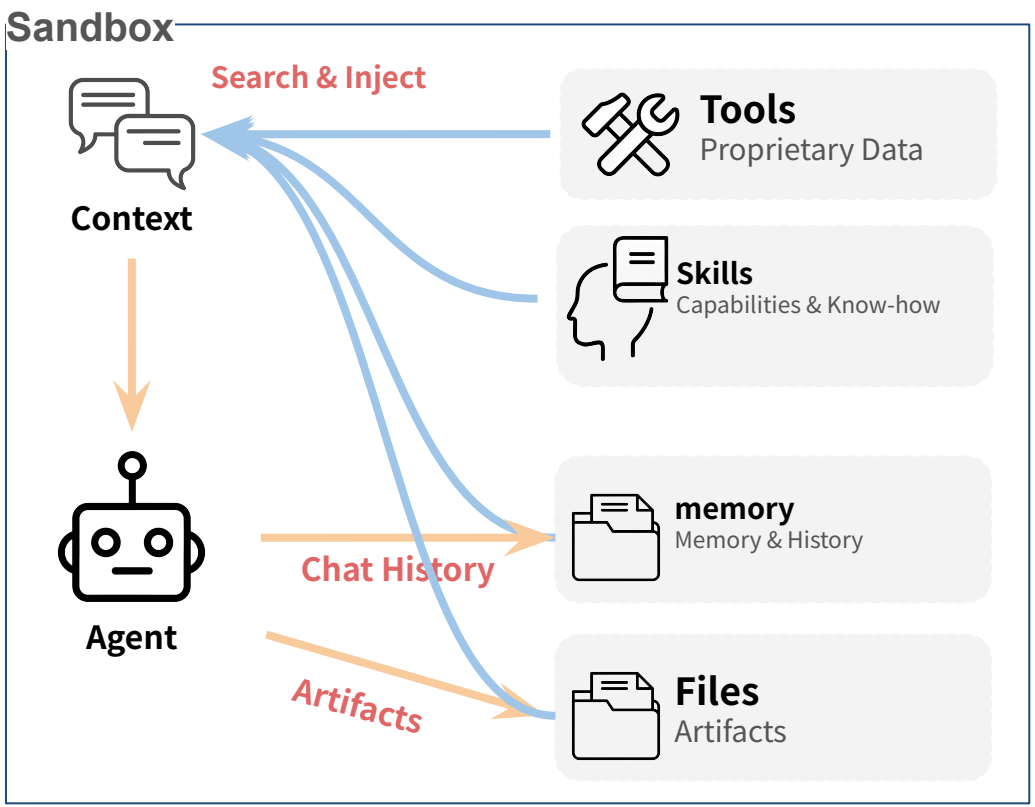
Data Isolation Challenges

It is necessary to isolate user-specific information (including memory and artifacts), but this is highly difficult with local storage.

External Persistence as a Solution

Just like databases in traditional applications, a dedicated **"Persistence Layer"** that securely stores agent artifacts and memory is essential.

User Data Requiring Persistence



Basically Read-Only

Tools and Skills are loaded when the agent starts, so they can be pre-baked into the Sandbox.

Read-Write

Histories and artifacts generated during conversation are dynamically written.

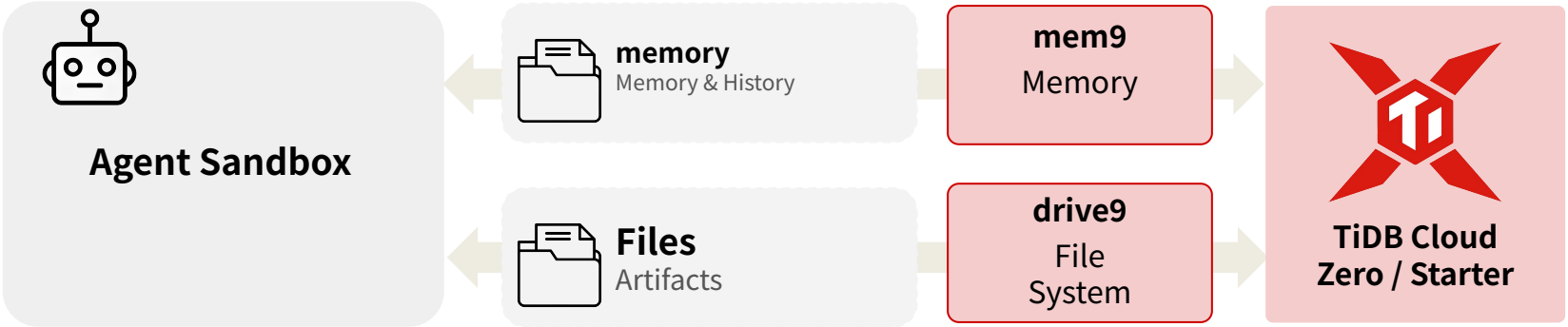
They must be persisted to avoid being lost when the Sandbox is discarded.

Persistence Services for Agents

While databases (KV, RDB, Vector) are commonly used as a persistence layer for agents:

- **Each DB requires its own unique connection method and tools, making setup complex.**
- **Saving memory and artifacts often requires multiple data models and processing.**

Therefore, we have built database-backed services for storing memory and artifacts, and released them as OSS.



Use Cases

When offering AI agents as a service, the handling of memory and deliverables differs by objective. Typical use cases are presented below.

Memory

Effective for services requiring personalization, where retaining past conversations and rules is critical.

- AI Companion
- Personal Agent (Claw)
- Coding Agent

Filesystem

Effective for services where deliverables are critical, trial-and-error is frequent, and long-term retention is needed.

- Builder (Web/App)
- Digital Worker
- Coding Agent

mem9:
Persistent Memory For Agents

Service site: https://mem9.ai
Github: https://github.com/mem9-ai/mem9

mem9：Memory for Agents

Challenge: Agents can retain memory in local files, but it is lost along with the sandbox.

Goal: Retain memory across agent sessions (without complex operations).

- Across agent sessions
- Across agent platforms

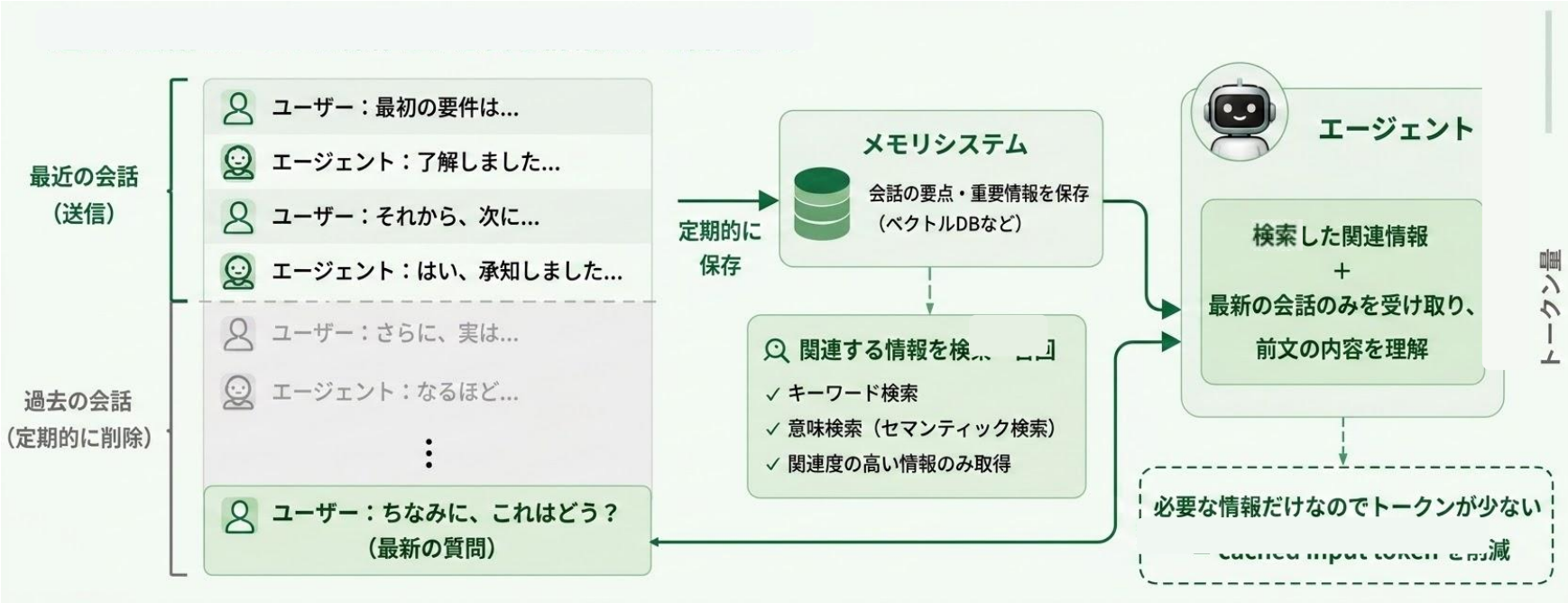
OpenClaw / Hermes / Claude / ChatGPT
OpenCode / DeepSeek / Dify / API

Mem9 Managed Service

Self Hosted (OSS)

Primary focus in this session

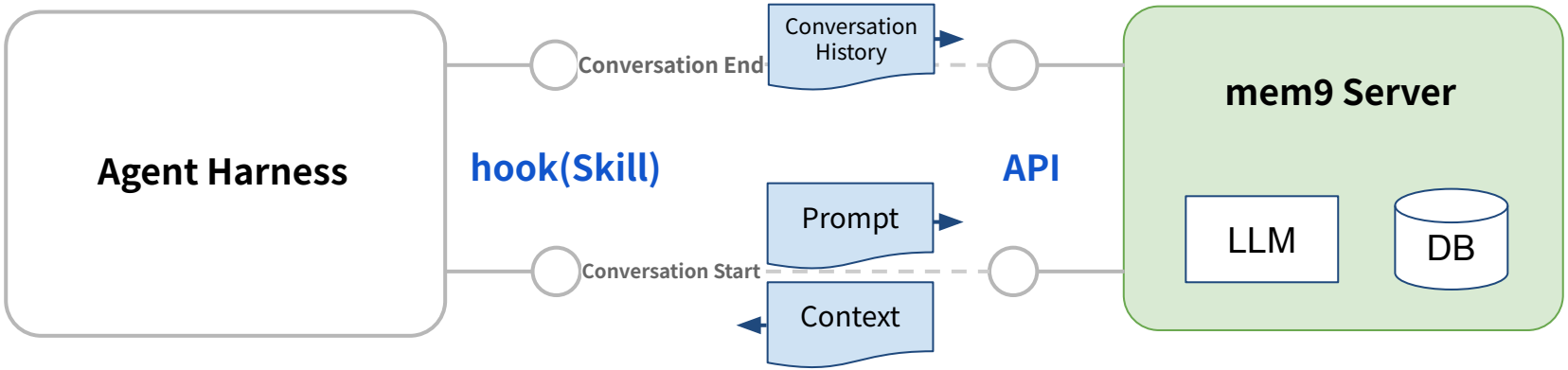
mem9: Overview



Interface with Agents

Leverage event hooks in the agent to send conversations and inject them into the context.

- **At Conversation End:**
  - Send and save conversation history, while extracting facts to be recorded.
  - Uses LLM for fact extraction, and Embedding Model for saving conversations to DB.
- **At Conversation Start:**
  - Search for relevant facts based on user prompts, and include them in the context along with recent conversation history.



Memory Storage and Extraction

01

Save Conversation History

Record full interaction history without omission

→

02

Fact Extraction (LLM)

Extract only key facts worth remembering from conversations

→

03

Reconcile

Decide to integrate, overwrite, or delete existing facts

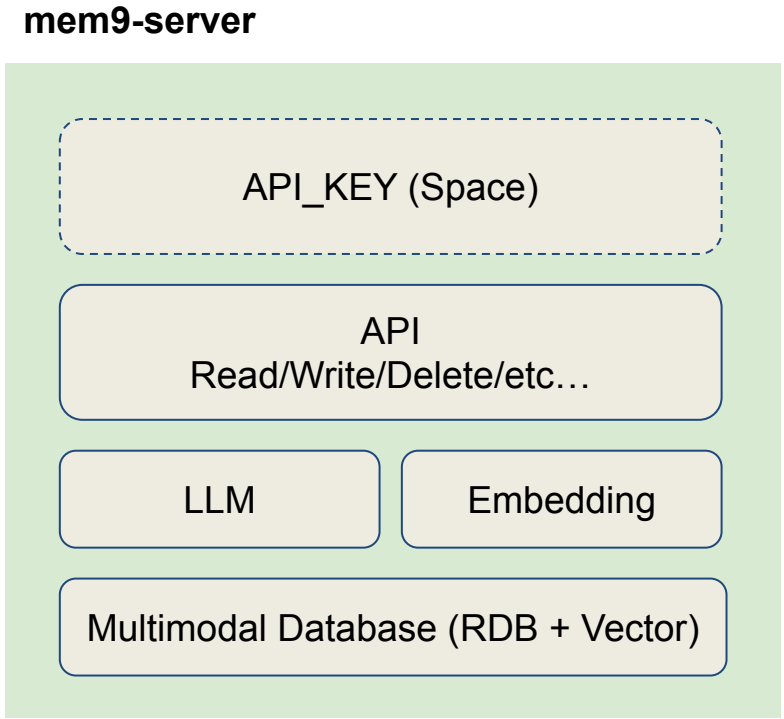
Memory Utilization: Context Insertion into New Conversations

- Hybrid search organized **Facts based on user instructions** to retrieve results
- Latest **conversation history** recording recent interactions

mem9 Architecture

- Space: Isolated memory (namespace)
  - Databases are also separated
- LLM for fact extraction (OpenAI API compatible)
- Embedding for vectorization
- Database for storage
  - Requires RDB + Vector DB
  - TiDB, PostgreSQL, etc.

Open-source models can also be used for LLM and Embedding (when self-hosted)



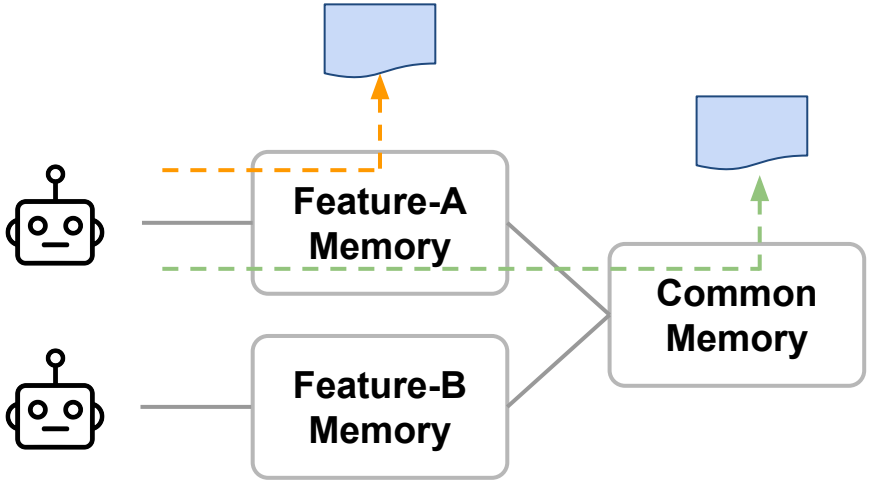
Space Chain: Memory Dependencies

Memory used by agents can have dependencies.

As shown in the diagram on the right:

- First query the assigned Feature memory. If facts with **sufficiently high accuracy** are found, adopt them.
- If not found, query the common memory and adopt those facts.

By combining them, you can configure flexible agent memories.

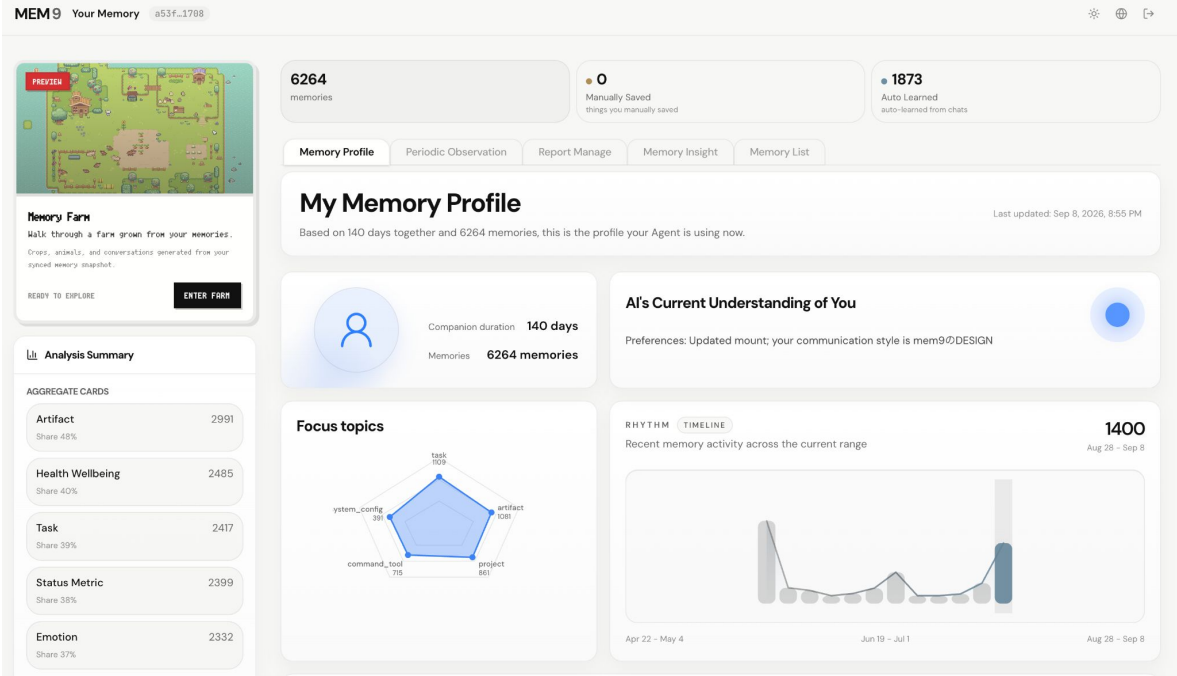


mem9: Leveraging Memory

Analyze agent memories using Mem9, or use it as a starting point for agent improvement.

- Topic summary
- Tag cloud
- Trend changes

In addition to the dashboard, analysis can be performed directly using the database.



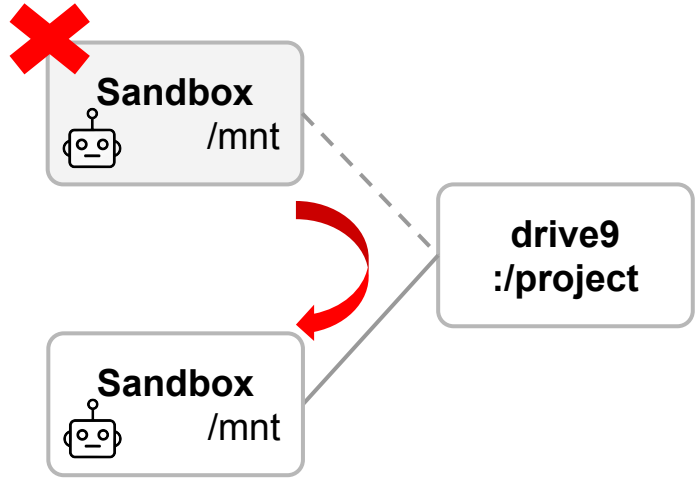
drive9:
Persistent Filesystem For Agents

Service site: https://drive9.ai/
Github: https://github.com/mem9-ai/drive9

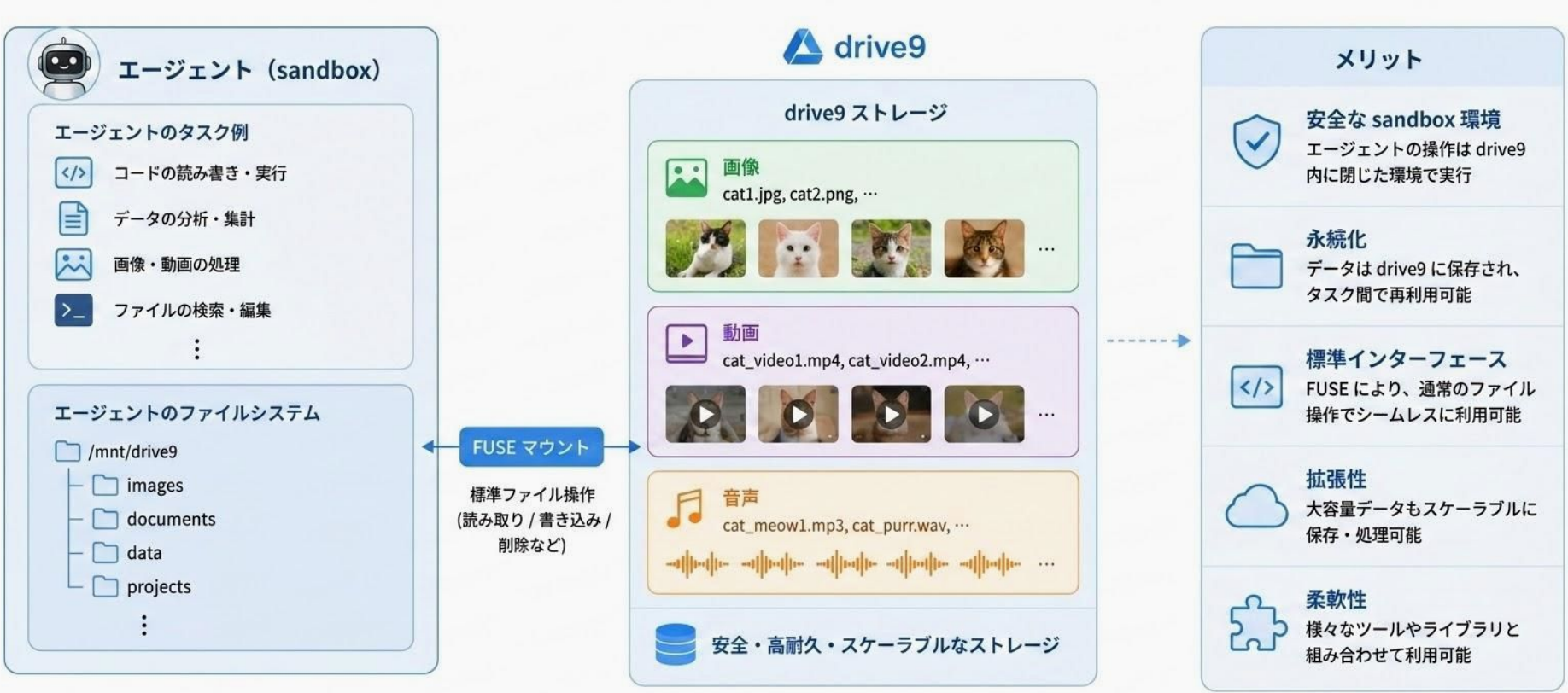
drive9: Filesystem for Agents

Sandboxes are disposable. Workspaces should not be.

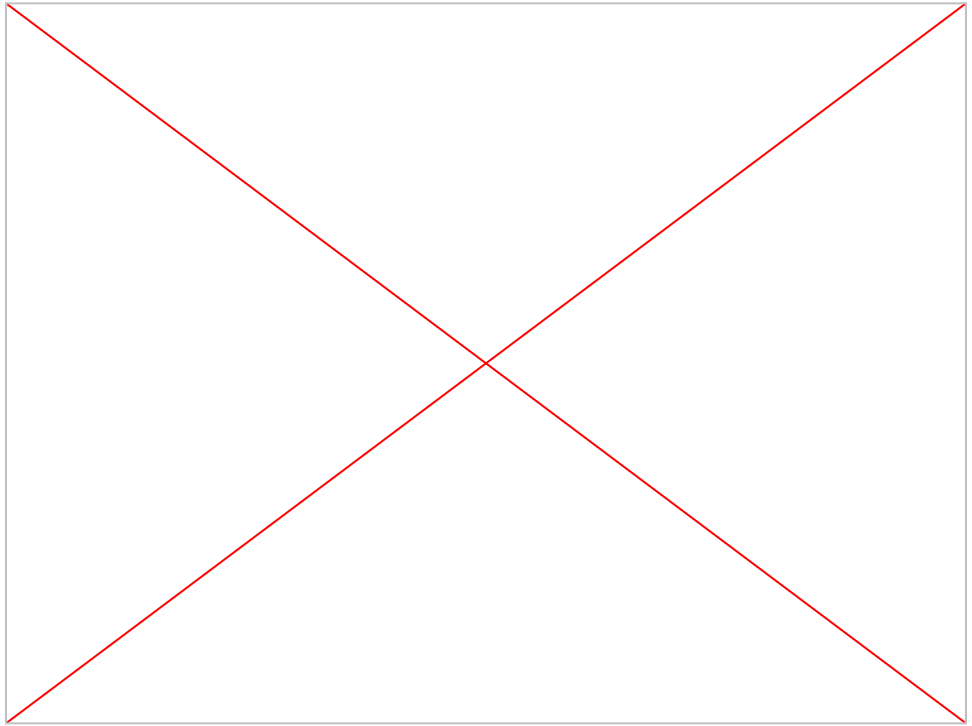
- Cloud workspaces controllable at runtime
- Mountable from any sandbox, supporting conflict detection and parallel trials
- No need to teach agents new tools—standard Unix tools and git work out of the box



drive9：Overview

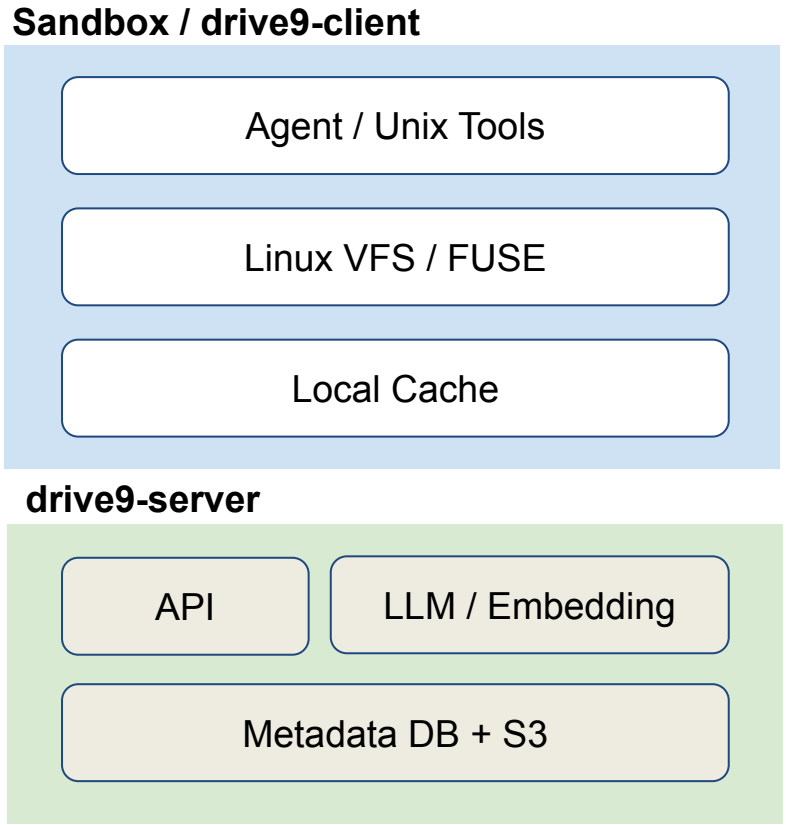


Demo: drive9



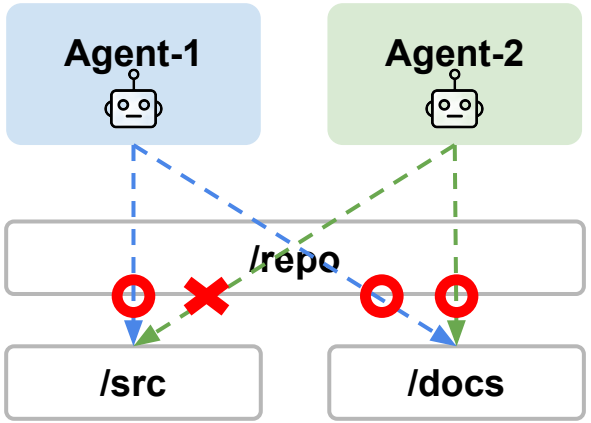
drive9: How It Works

- Virtual file system using FUSE
- Read/write just like standard files; Unix commands can be used as-is
- Files are written to local cache and uploaded asynchronously to the server
- Metadata is automatically assigned based on file type
- Metadata is generated for unstructured files using LLMs
- Actual data files are stored in S3 or other storage services



Scope Control

- Drive9 has a scope called a "tenant"
  - Each tenant has a separate namespace
- Can create short-term sub-contexts (tokens) for agent tasks
  - Access control can be granted on a path-by-path basis
- Enables fine-grained access control based on the agent's role

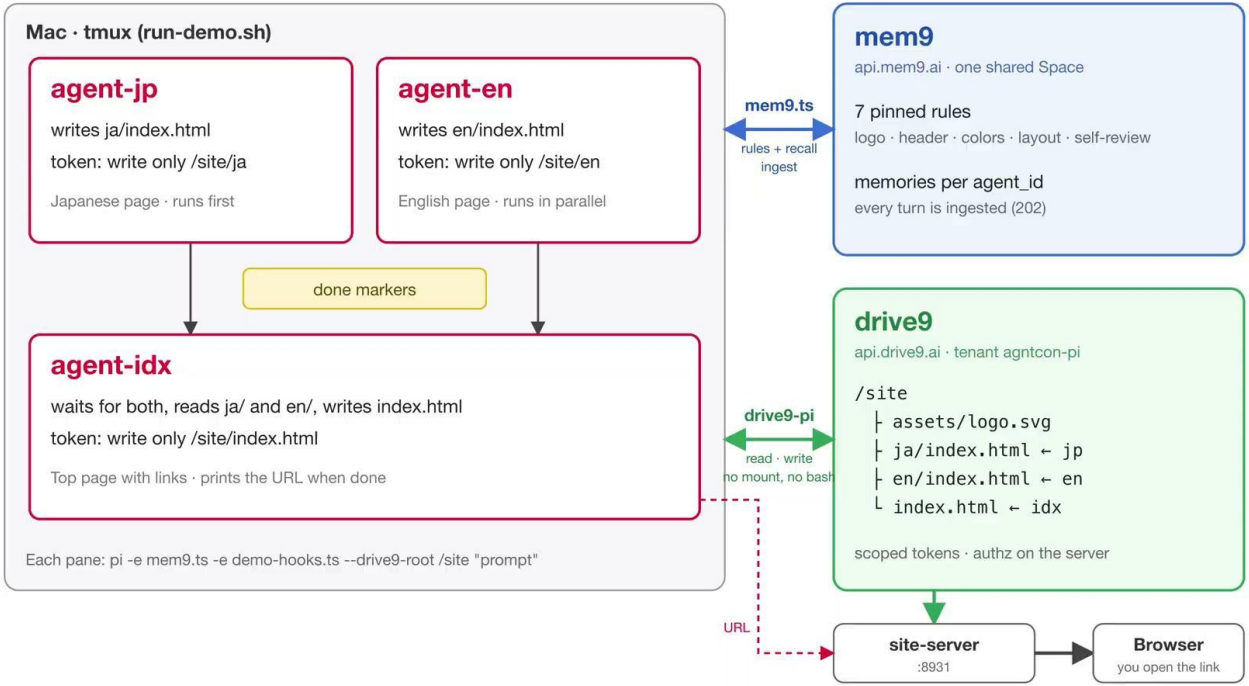



```
drive9 token issue agent-2 --ttl 1h \
  --allow /repo/docs:read,list,search,write
```

Demo: mem9 + drive9

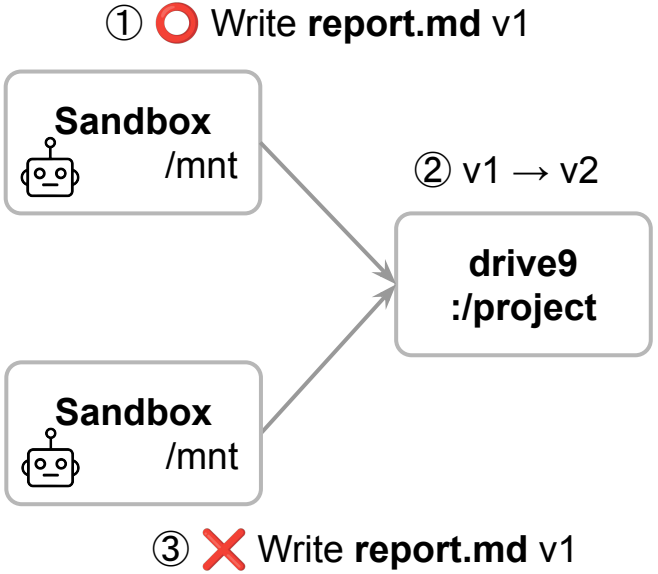
Three pi agents, one drive9, one mem9

Same pi, same extensions, same model. Only the role, the drive9 token and the mem9 agent_id differ.



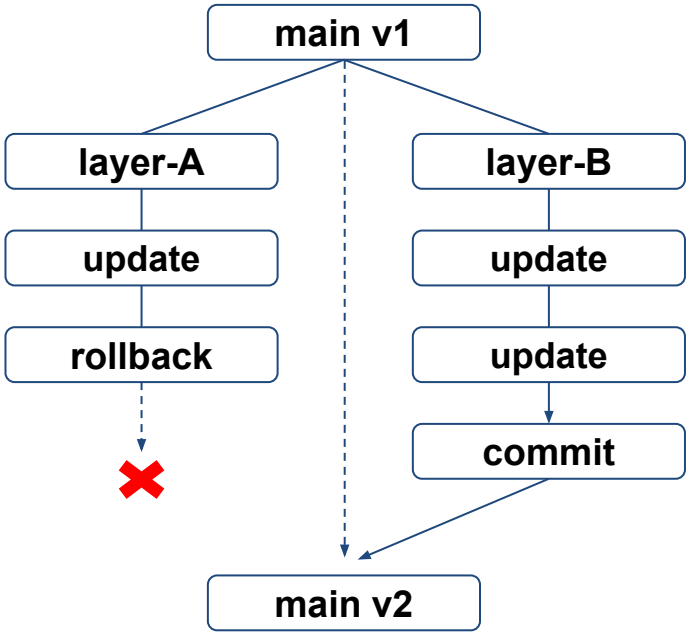
Conflict Detection via Opt. Locking

- drive9 has a conflict detection mechanism
- Has a version number as file metadata
  - This number increments on update
- Reads this number on Open, and specifies it to update on Write
- Errors as a conflict if the server-side version is ahead
- The failed file remains on the failing side



Layer Capability to Support Trials

- Create layers that overlay original data via Copy-on-Write
- Mount by specifying a specific layer
- Updates on a layer do not affect original data and are isolated to the layer
- Multiple layers can be created, useful for agent exploration and trials
- Layers are eventually committed to main, or rolled back (discarded)



Summary

In this session, we introduced mem9 and drive9 as solutions to the challenge of state persistence in agent services.

By decoupling agents into a harness and user-specific data, the harness becomes stateless. This clarifies the boundary between applications and data, offering distinct deployment advantages.

Additionally, both mem9 and drive9 share key design principles that serve as valuable implementation patterns for AI infrastructure:

- Hybrid search for information retrieval (FTS + Vector + RRF)
- Separation by namespaces and dynamic switching
- Information processing via LLMs (distillation and metadata enrichment)

Agentic AI Foundation

AGNTCon + MCPCon Japan

Where the **agentic stack** is **being built**.