Agentic AI Foundation

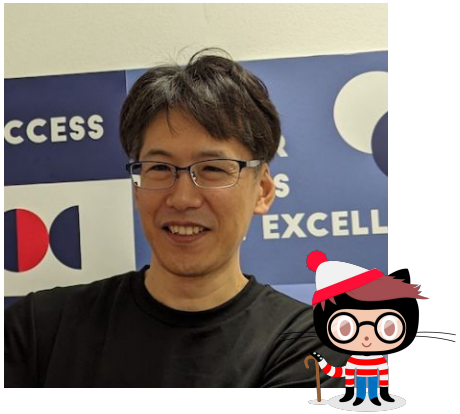
AGNTCon + MCPCon Japan

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

本セッションでは、AIエージェントをサービス提供する際の課題、特に永続化についてお話します。また、それらを解決するオープンソースのインフラストラクチャ、mem9とdrive9について説明します。

- エージェントにおける永続化の課題
- mem9：メモリの永続化
- drive9：成果物の永続化
- まとめ

エージェントの永続化における課題

AIエージェントをサービスとして提供する場合、ユーザーデータの保存が課題となる。ローカル保存では永続化とデータ分離の問題が生じる。

01

長期的な継続性の欠如

現代のハーネスが基本とするローカル保存では、環境停止と共に状態（State）、記憶（Memory）、成果物が消失する。スケーラビリティや耐障害性も課題

02

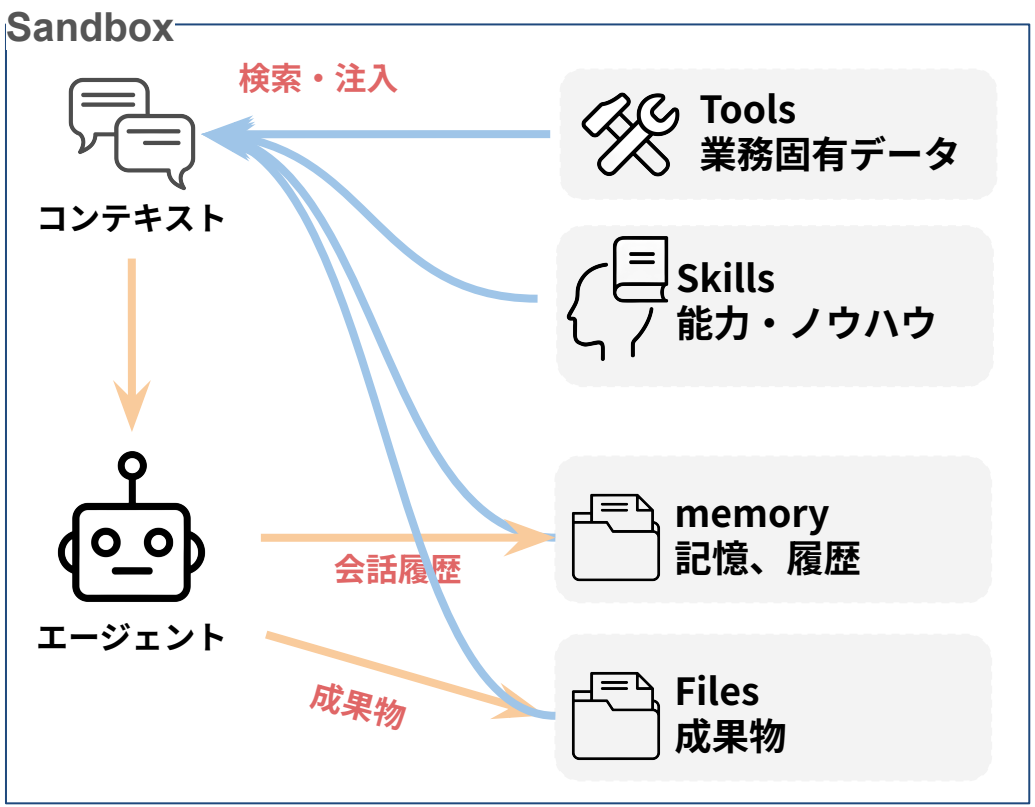
データ分離の課題

ユーザー単位で固有の情報（メモリや成果物含む）を分離する必要があるが、ローカル保存ではこれは難しい。

解決策としての外部永続化

従来のアプリケーションにおけるデータベースのように、エージェントの成果物と記憶を安全に保持する専用の**「永続化レイヤー」**が不可欠

永続化が必要なユーザーデータ



基本的に Read-Only

Tools と Skills はエージェント起動時に読み込まれるため、事前にSandboxにベイク可能

Read-Write

会話中に発生する履歴や成果物は動的に書き込まれる。

Sandboxの廃棄とともに失われないよう永続化する必要がある

エージェントのための永続化サービス

エージェントの永続化レイヤーとしてはデータベース（KV, RDB, Vector）もよく利用されているが

- **DB毎に固有の接続方式やツールが必要となり、セットアップも煩雑**
- **メモリや成果物の保存に複数のデータモデルや、加工が必要となることも多い**

そのため、**データベースをバックエンドとした、メモリや成果物保存のためのサービス**を構築し、OSSで公開している。

ユースケース

サービスとしてAIエージェントを提供する場合、目的により記憶と成果物をどのように扱うかが異なる。ここでは典型的なユースケースを紹介する。

Memory

パーソナライズが必要で、過去の会話の記憶やルールの記憶が重要なサービスで有効

- AIコンパニオン
- パーソナルエージェント（Claw）
- コーディングエージェント

Filesystem

成果物が重要で、試行錯誤が多く、成果物の保存期間が長いサービスで有効

- ビルダー（Web/App）
- デジタルワーカー
- コーディングエージェント

mem9:
Persistent Memory For Agents

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

mem9：Memory for Agents

課題：エージェントはローカルファイルで記憶を保持できるが、サンドボックスとともに失われる

目的：エージェントのセッションを超えて記憶を保持したい。（複雑な運用を行わない形で）

- エージェントのセッション間でも
- エージェントプラットフォーム間でも

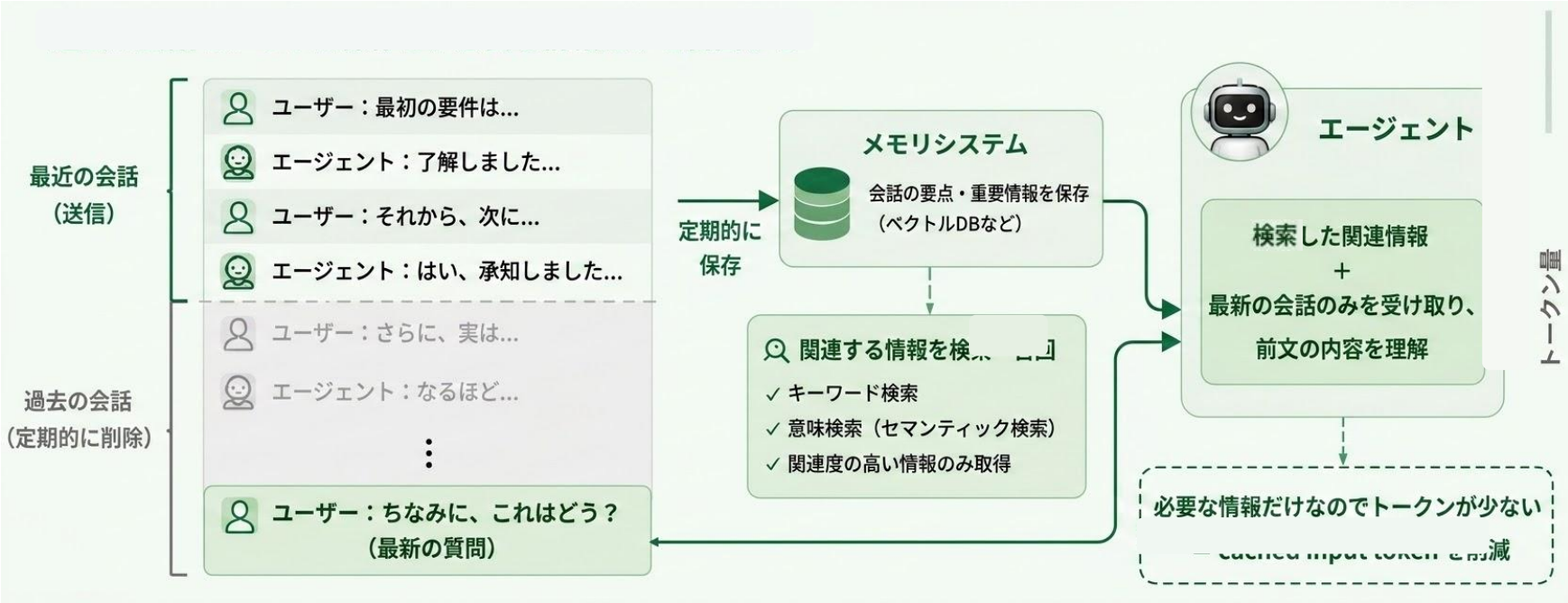
OpenClaw / Hermes / Claude / ChatGPT
OpenCode / DeepSeek / Dify / API

Mem9 Managed Service

Self Hosted (OSS)

こちらを中心に説明

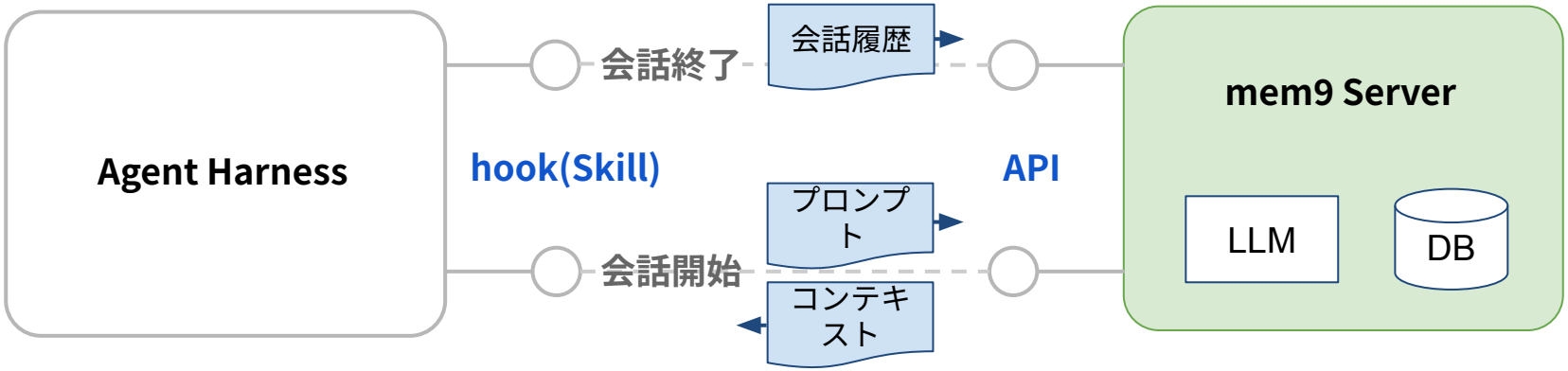
mem9：全体像


最近の会話
（送信）
過去の会話
（定期的に削除）
ユーザー：最初の要件は...
エージェント：了解しました...
ユーザー：それから、次に...
エージェント：はい、承知しました...
ユーザー：さらに、実は...
エージェント：なるほど...
ユーザー：ちなみに、これはどう？
（最新の質問）
定期的に
保存
メモリシステム
会話の要点・重要情報を保存
（ベクトルDBなど）
関連する情報を検
✓ キーワード検索
✓ 意味検索（セマンティック検索）
✓ 関連度の高い情報のみ取得
エージェント
検索した関連情報
＋
最新の会話のみを受け取り、
前文の内容を理解
トークン量
必要な情報だけなのでトークンが少ない

エージェントとのインターフェース

エージェントに備わっているイベントフックを利用して、会話の送信とコンテキストへの差し込みを実現

- **会話終了時：**
  - 会話履歴を送付し、保存すると同時に記録すべき事実を抽出
  - 事実の抽出にLLMを、会話のDBへの保存にEmbedding Modelを利用している
- **会話開始時：**
  - ユーザーからの指示文を元に、関連する事実を検索。直近の会話履歴とあわせてコンテキストに含める



メモリの保存と抽出

01

会話履歴の保存

やり取りの全履歴を漏れなく記録

→

02

Factの抽出（LLM）

会話から記憶すべき重要な事実のみを抽出

→

03

リコンサイル

既存の事実を統合・上書き・削除の判断

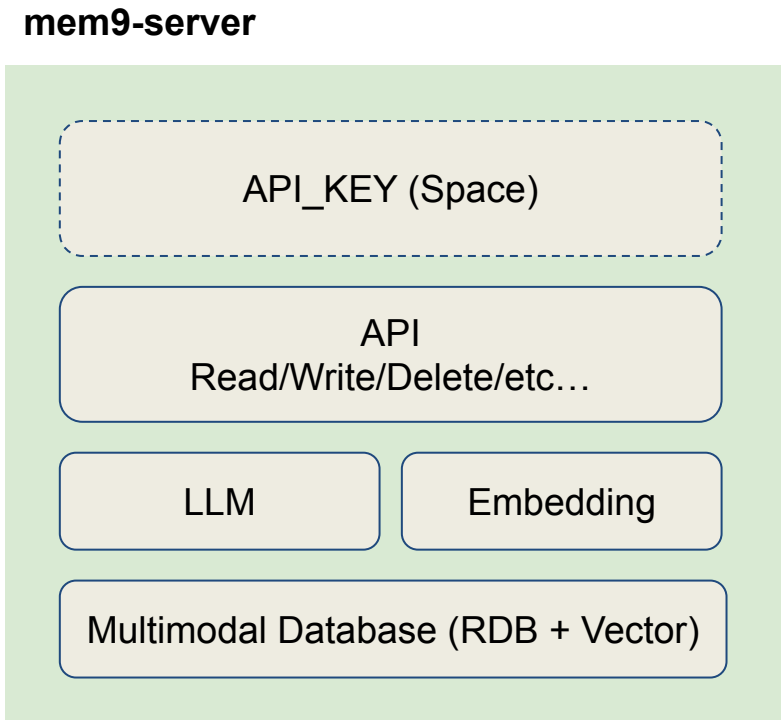
記憶の利用：新しい会話へのコンテキスト挿入

- ユーザーの指示に基づき、整理された**Factをハイブリッド検索**して結果を取得
- 直近のやり取りを記録した**最新の会話履歴**

mem9のアーキテクチャ

- Space: 分離されたメモリ（名前空間）
  - データベースも分かれる
- 事実抽出のためのLLM（OpenAI API互換）
- 埋め込みのためのEmbedding
- 保存のためのデータベース
  - RDB + VectorのDBを要求
  - TiDBやPostgreSQL等

LLMやEmbeddingにはオープンモデルも利用可能（セルフホストの場合）



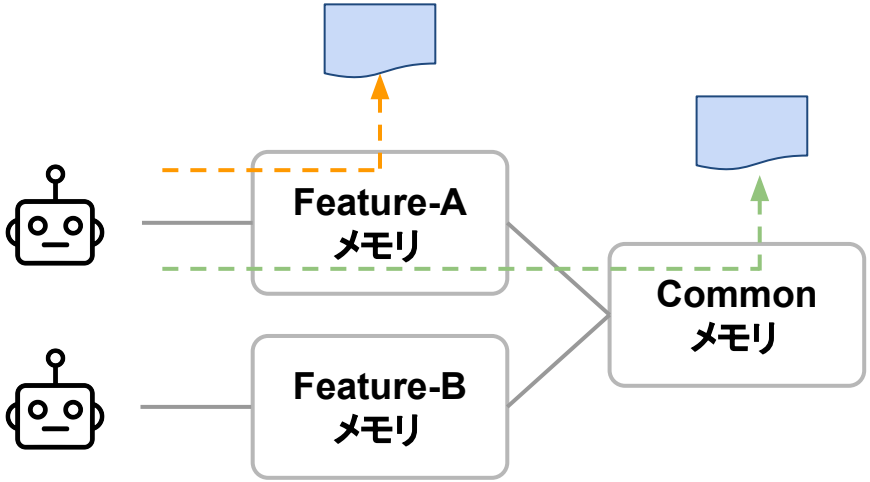
Space Chain：依存関係のあるメモリ

エージェントが利用するメモリに依存関係を持たせることができる。

右図のように、

- 自分の担当するFeatureのメモリにまず問い合わせ、そこで**十分精度の高い**事実が得られたらそれを採用
- もし得られなければ、共通のメモリに問い合わせ、その事実を採用

と、組み合わせてメモリを構成することができる。

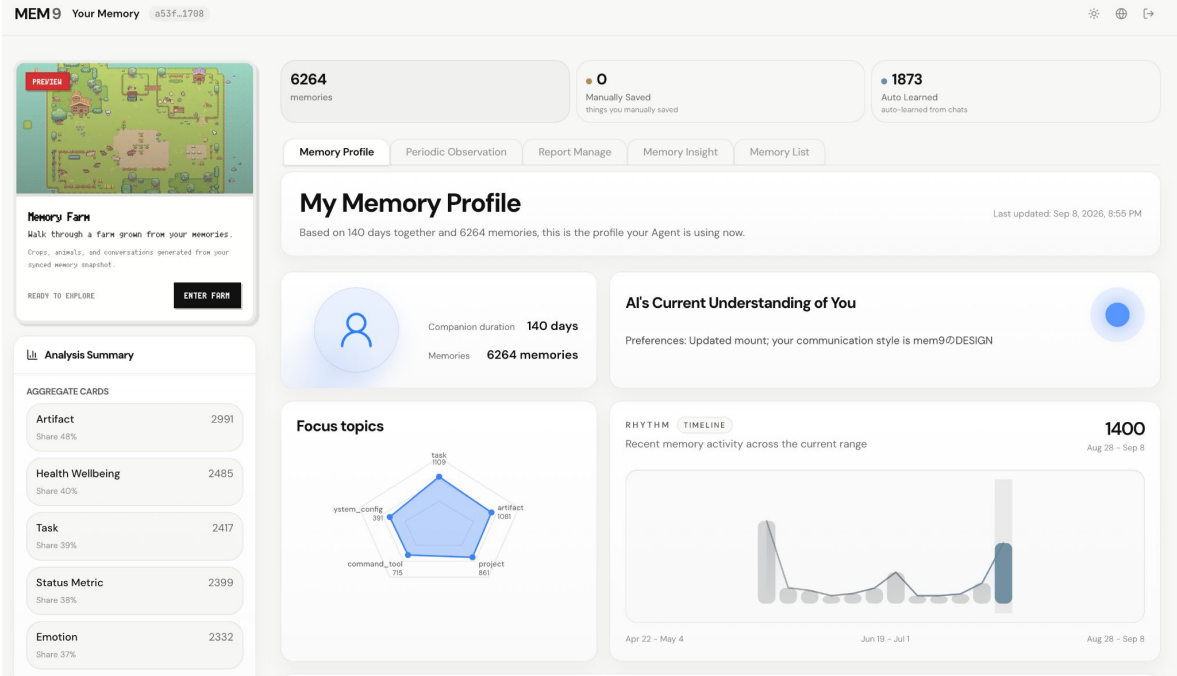


mem9：記憶の活用

Mem9を使ってエージェントの記憶を分析したり、エージェント改善の起点とすることもできる

- トピックサマリー
- タグクラウド
- トレンドの変化

ダッシュボード以外にも直接DBを使って分析もできる



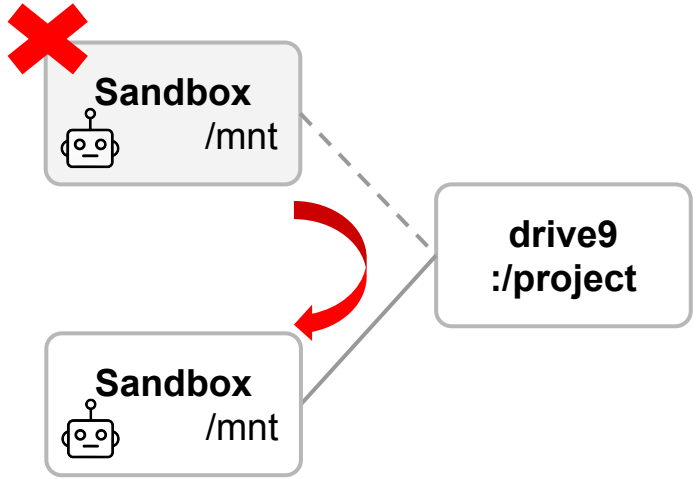
drive9:
Persistent Filesystem For Agents

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

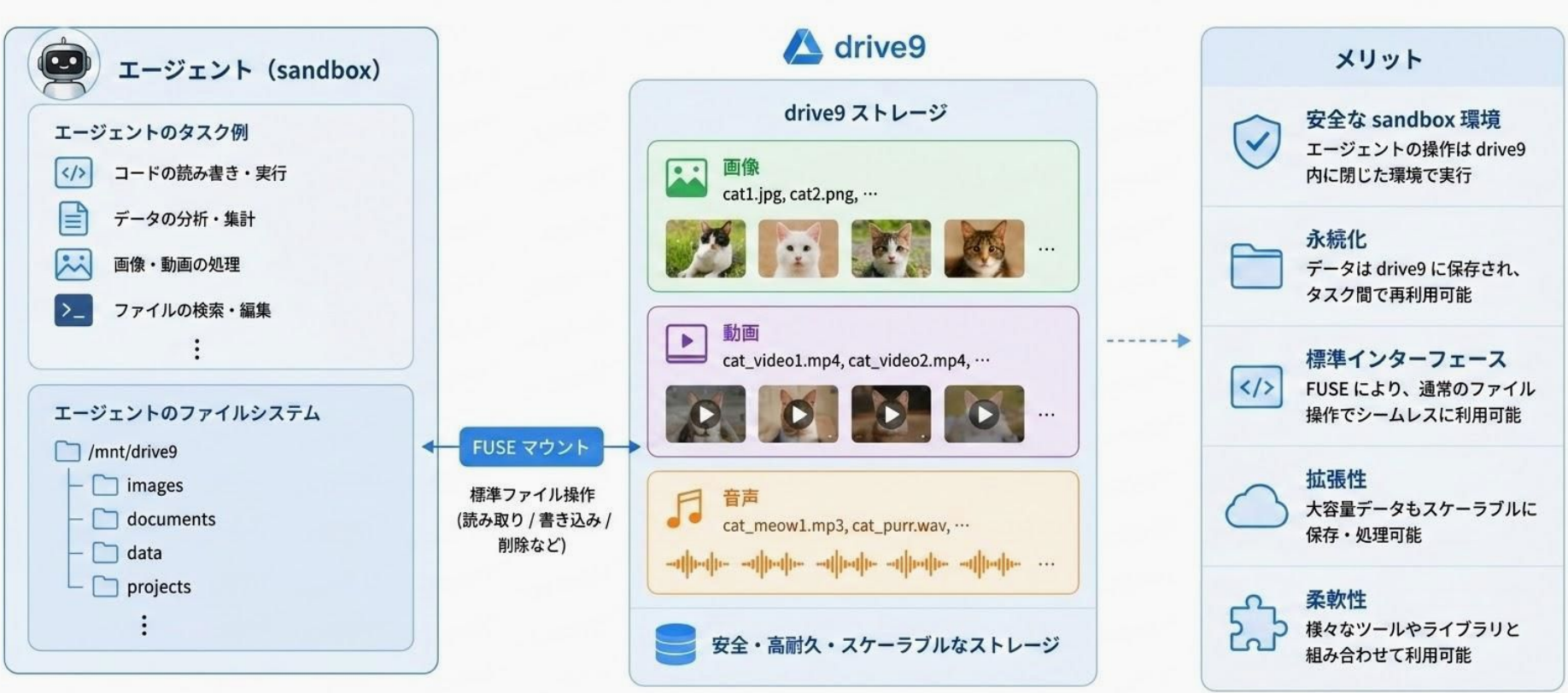
drive9：Filesystem for Agents

Sandboxes are disposable.
Workspaces should not be.

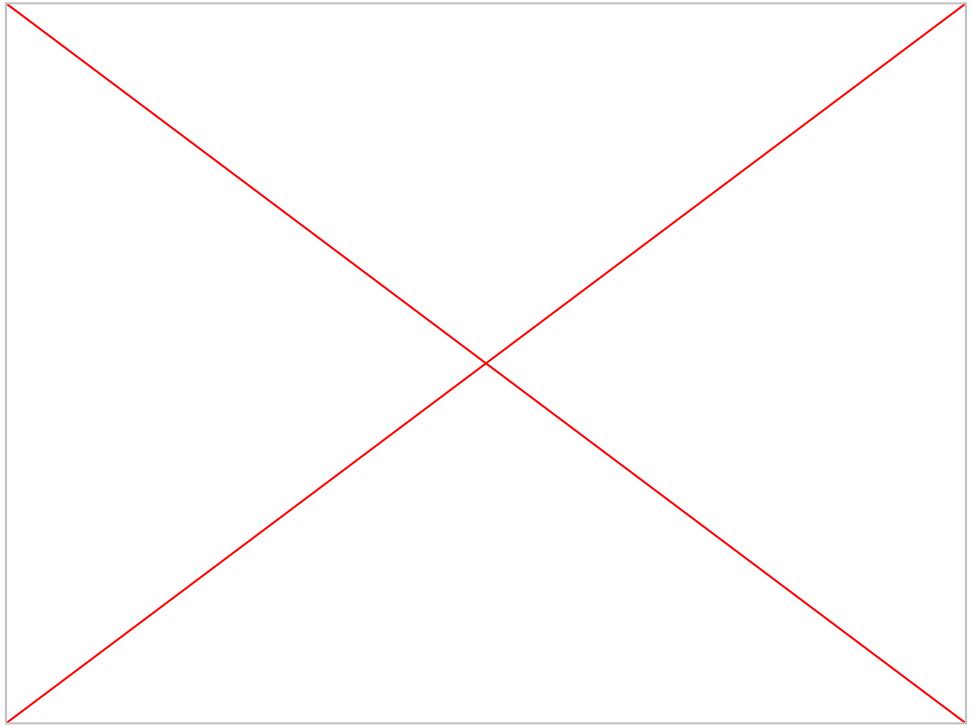
- ランタイムでコントロールできるクラウドワークスペース
- どのサンドボックスからもマウントでき、競合検出や並行試行をサポートする
- エージェントに新しいツールを使わせない。Unix Toolsやgitがそのまま利用できる



drive9：全体像

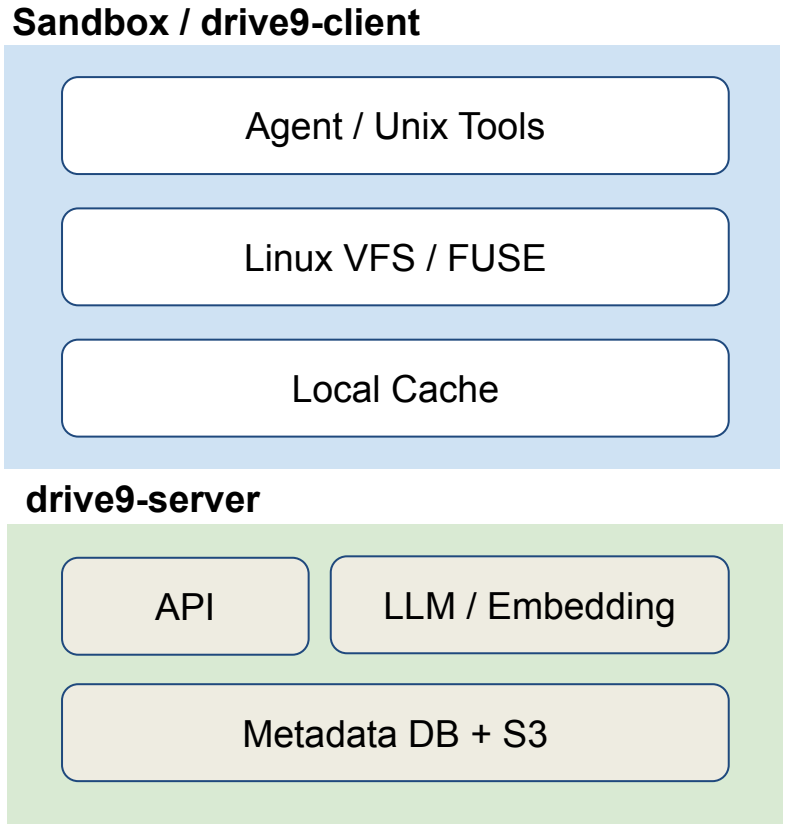

エージェント（sandbox）
エージェントのタスク例
コードの読み書き・実行
データの分析・集計
画像・動画の処理
ファイルの検索・編集
エージェントのファイルシステム
/mnt/drive9
images
documents
data
projects
FUSE マウント
標準ファイル操作
(読み取り / 書き込み / 削除など)
drive9
drive9 ストレージ
画像
cat1.jpg, cat2.png, ⋯
動画
cat_video1.mp4, cat_video2.mp4, ⋯
音声
cat_meow1.mp3, cat_purr.wav, ⋯
安全・高耐久・スケーラブルなストレージ
メリット
安全な sandbox 環境
エージェントの操作は drive9 内に閉じた環境で実行
永続化
データは drive9 に保存され、タスク間で再利用可能
標準インターフェース
FUSE により、通常のファイル操作でシームレスに利用可能
拡張性
大容量データもスケーラブルに保存・処理可能
柔軟性
様々なツールやライブラリと組み合わせて利用可能

Demo: drive9



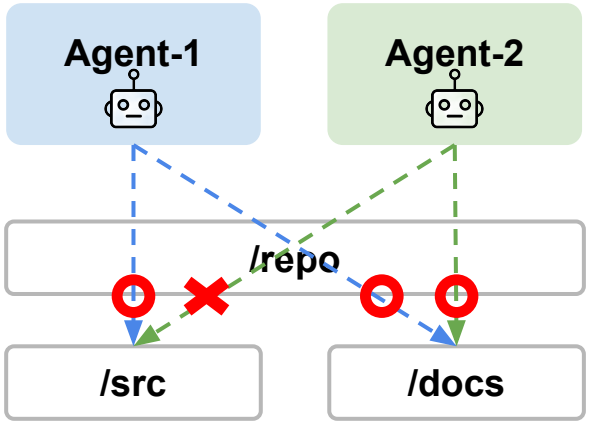
drive9：仕組み

- FUSEを使った仮想ファイルシステム
- 通常のファイル同様にread/write。Unixコマンドなどもそのまま利用可能
- ファイルはローカルキャッシュに書かれ、非同期でサーバに送信
- ファイルタイプに応じたメタデータ付与
- 非構造化ファイルもLLMによりメタデータ生成
- データファイル本体はS3などに書かれる



スコープの制御

- Drive9はテナントと呼ばれるスコープを持つ
  - テナント毎に名前空間も別
- エージェント作業用に、短期のサブコンテキスト（トークン）を作成することができる
  - このとき、パス単位でアクセス制御を付与できる
- エージェントの役割に応じて細かいアクセス制御を可能にすることが可能

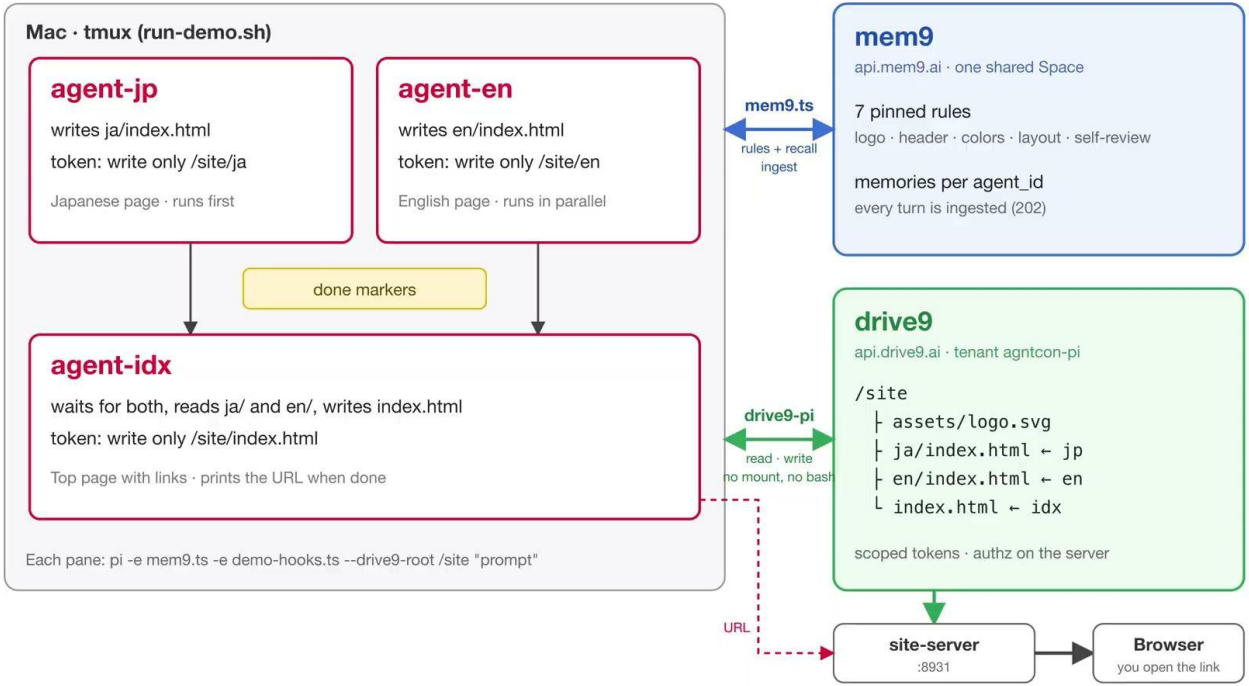



```
drive9 token issue agent-2 --ttl 1h \
  --allow /repo/docs:read,list,search,write
```

Demo: mem9 + drive9

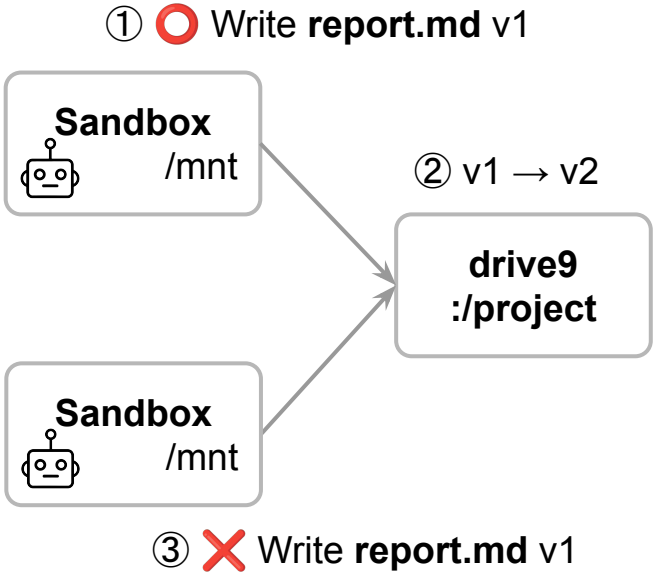
Three pi agents, one drive9, one mem9

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



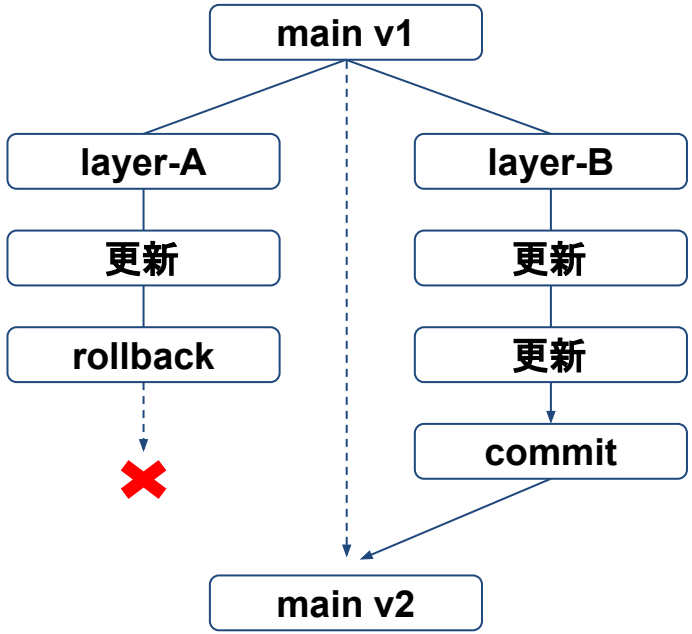
楽観ロックによる競合検知

- drive9は競合検知の仕組みを持つ
- ファイルのメタ情報として**バージョン番号**を持つ
  - この番号は更新時に増加する
- Openの際にこの番号を読み、Writeの際に番号を指定して更新する
- サーバ側のバージョンがそれより進んでいた場合は競合発生としてエラー
- 失敗側には失敗したファイルが残る



試行をサポートするレイヤー機能

- Copy-on-Writeにより元のデータにオーバーレイするレイヤーを作成
- レイヤーを指定してマウントできる
- レイヤー上の更新は元のデータに反映されず、レイヤーに限定される
- レイヤーは複数作成でき、エージェントの試行などに役立つ
- レイヤーは最終的に元（main）にcommit、または廃棄（rollback）



まとめ

本セッションでは、エージェントサービスの状態永続化という課題に対するソリューションとして、mem9とdrive9について説明した。

エージェントをハーネスとユーザー固有データに分けることで、ハーネスはステートレスになり、アプリケーションとデータの境界が明確になりデプロイ上のメリットも生じる。

また、mem9とdrive9の両方の設計において、下記の点は共通しており、AIインフラの実装パターンとして参考になるものと思われる。

- 情報検索時のハイブリッドサーチ（FTS + Vector + RRF）
- 名前空間による分離と動的な切り替え
- LLMによる情報の加工（蒸留やメタ情報付与）

Agentic AI Foundation

AGNTCon + MCPCon Japan

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