



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

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、インフラストラクチャのライフサイクル管理が大きな課題となる。これはエージェントの実行環境・利用インターバルが異なるため

01

エフェメラルなサンドボックス環境

エージェントはリクエストごとに起動し、一定時間で停止するサンドボックス上で動作するため、環境停止と共に状態（State）や記憶（Memory）が消失する

02

長期的な継続性の欠如

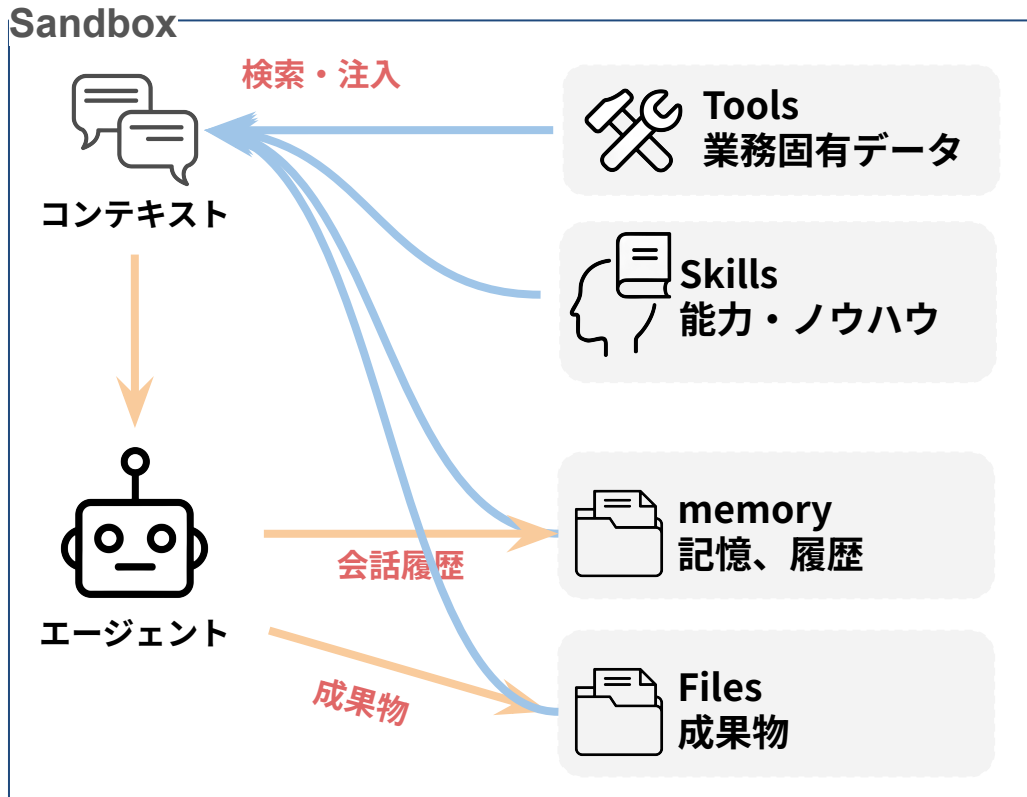
数日後、あるいは数ヶ月後という長いスパンで作業を再開するようなユースケースをサポートする必要がある

03

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

サンドボックスと永続化



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つつ、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。



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、スコープが限定されて引き継ぐのは容易でない。

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）

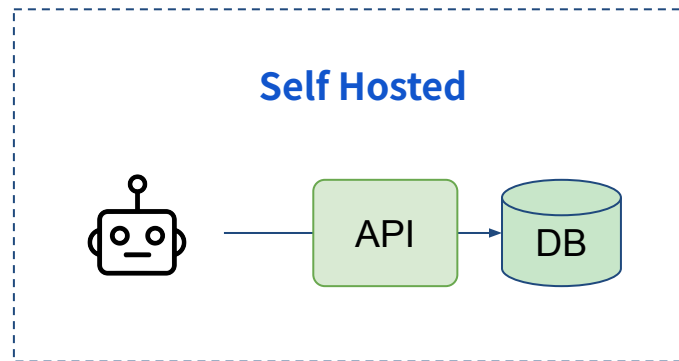
- エージェントのセッション間でも
- エージェントプラットフォーム間でも

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

Mem9 Service



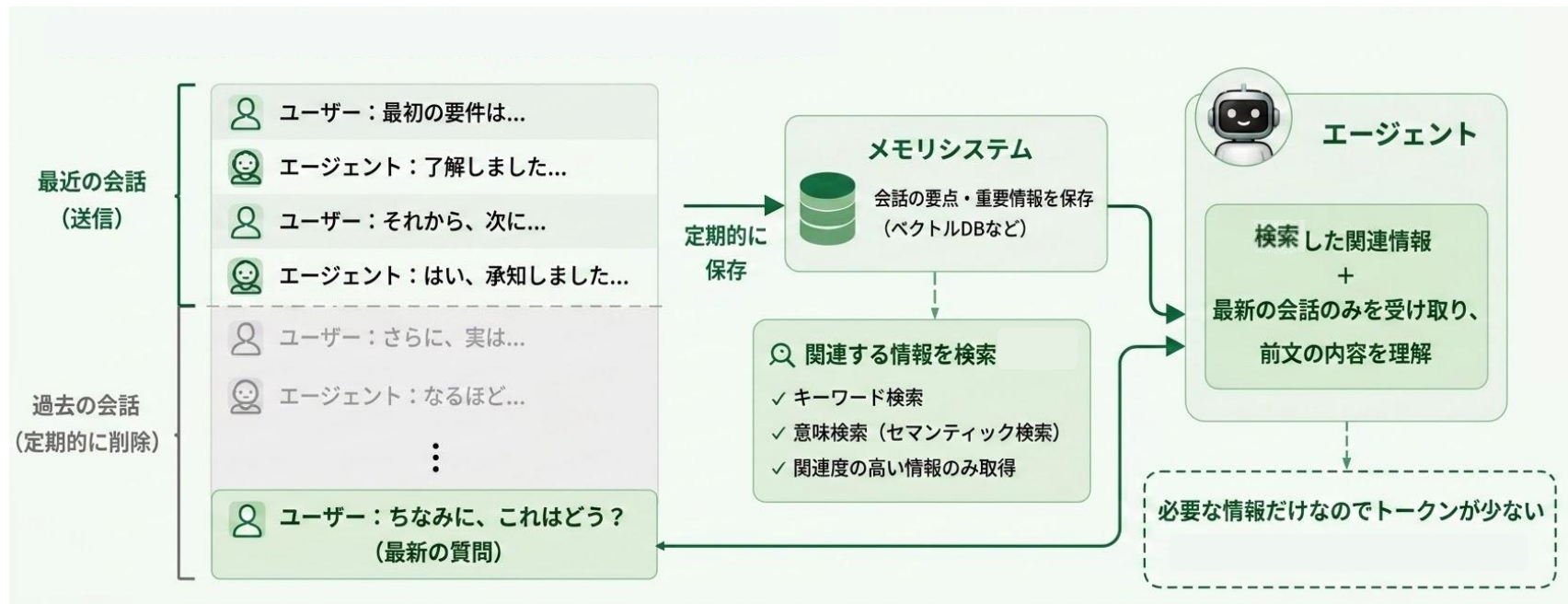
Self Hosted



こちらを中心に説明

mem9 : 概要

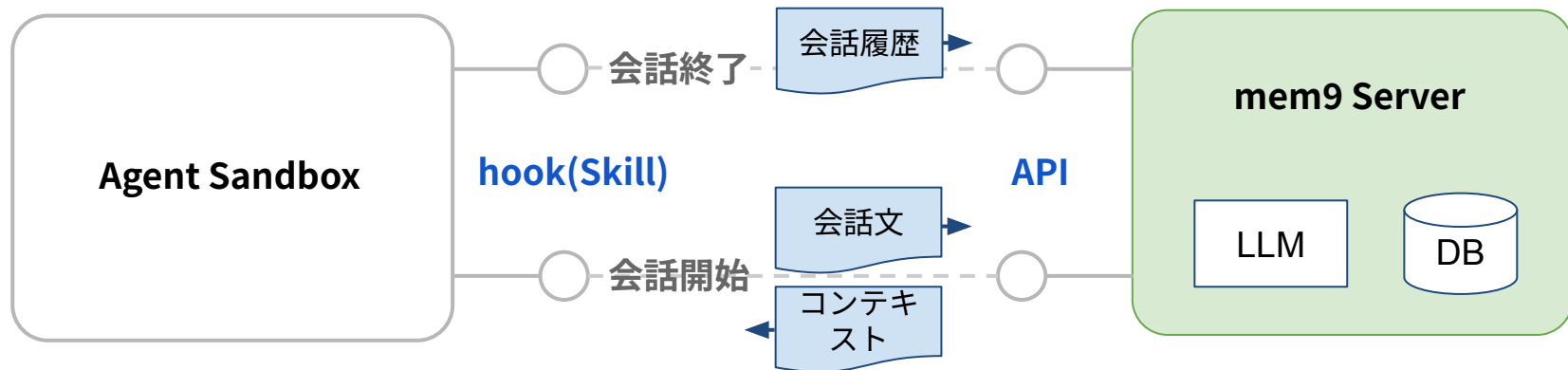
TiDBやPostgreSQLのような、ベクトル型を備えたDBを使って実現するエージェントメモリ



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

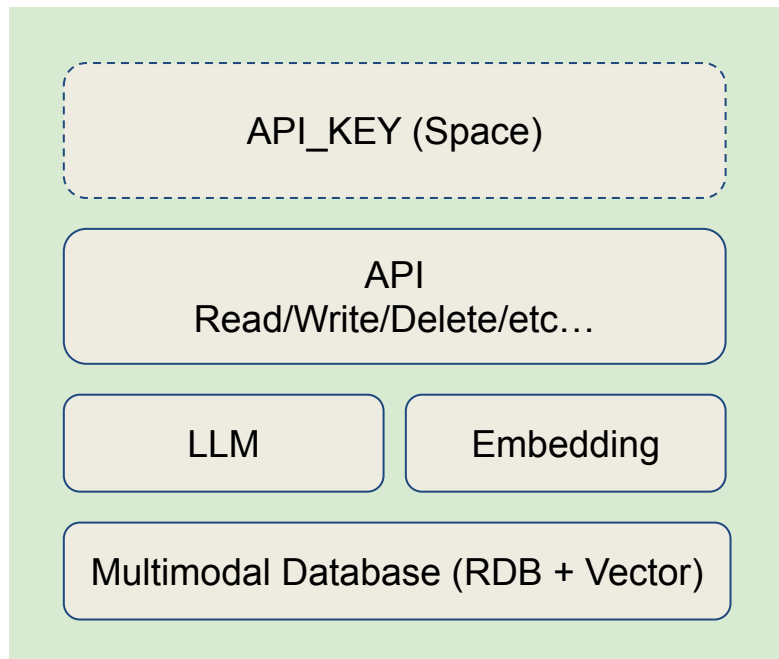
- ユーザーの指示に基づき、整理されたFactをハイブリッド検索して結果を取得
- 直近のやり取りを記録した最新の会話履歴

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）

mem9-server



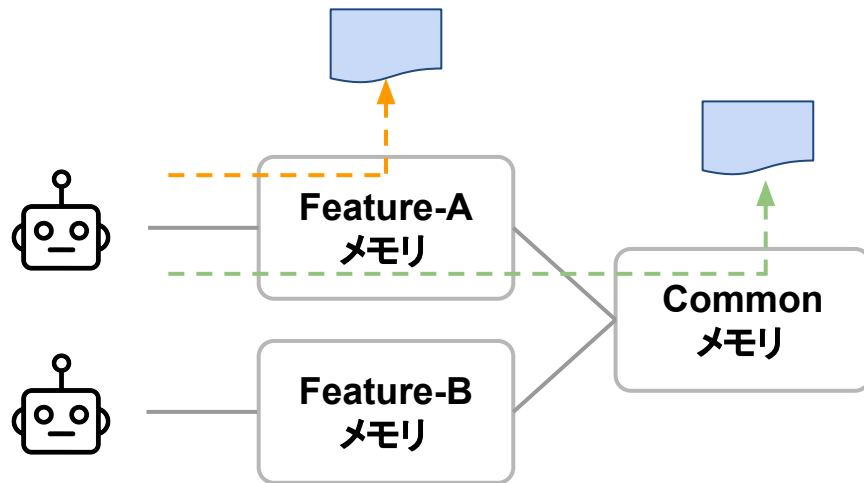
Space Chain：依存関係のあるメモリ

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。

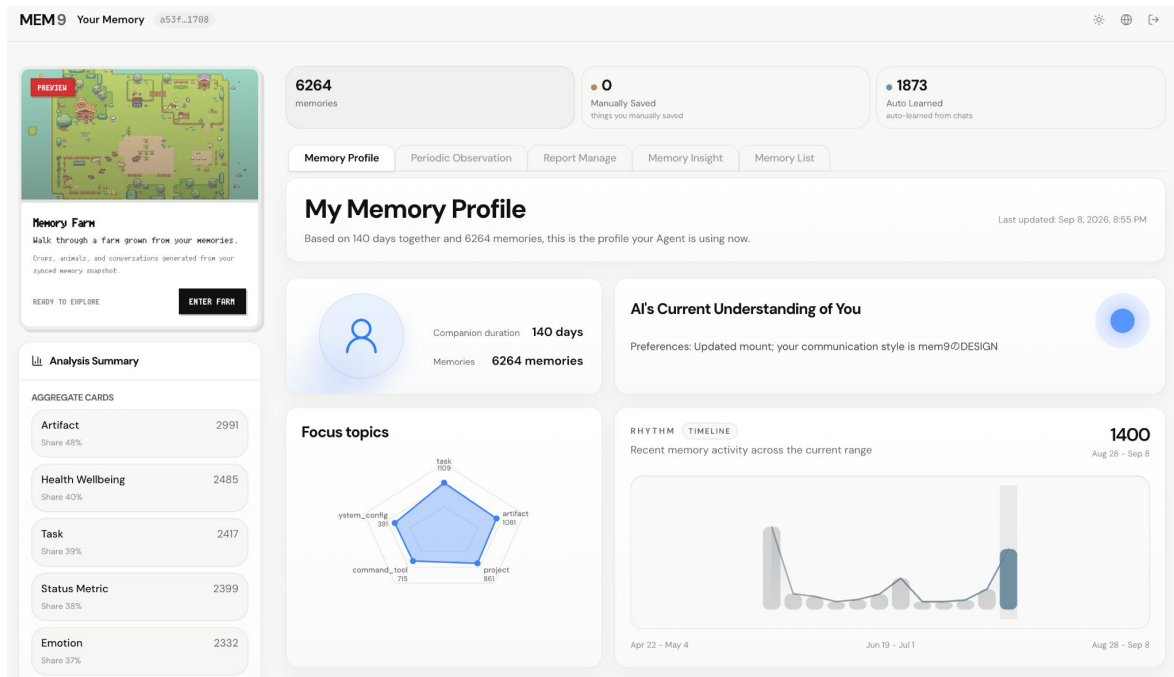


mem9：記憶の活用

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



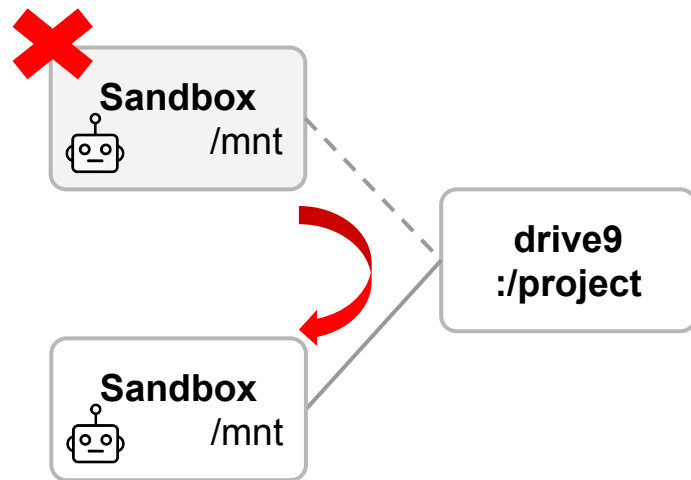
drive9: Persistent Filesystem For Agents

Service site: <https://drive9.ai/>

Github: <https://github.com/mem9-ai/drive9>

*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



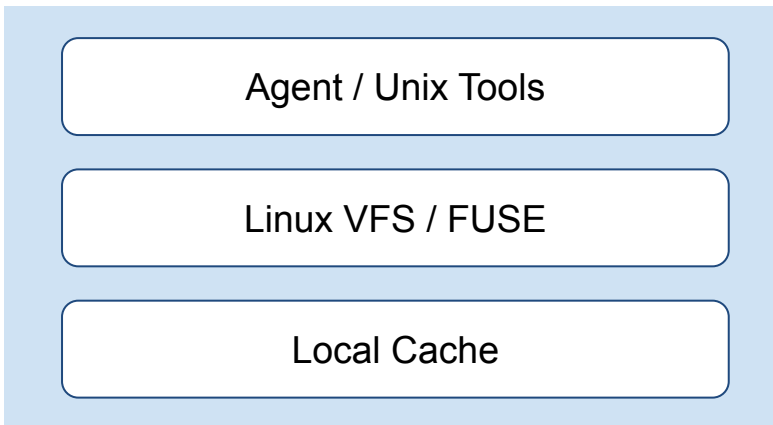
drive9全体像



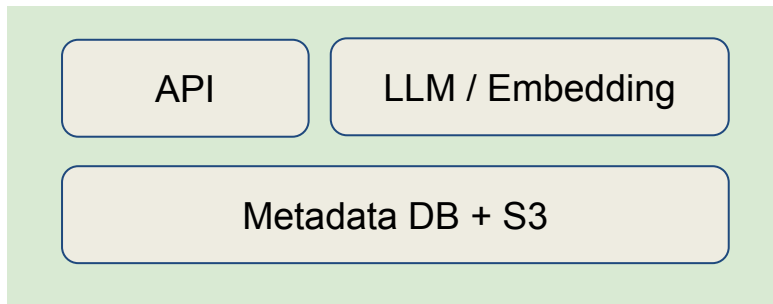
drive9 : 仕組み

- 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

Sandbox / drive9-client

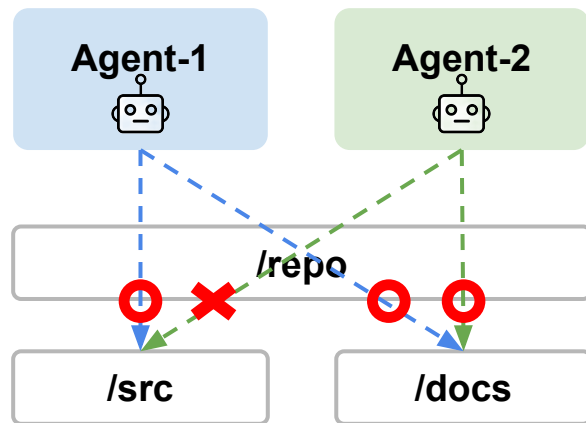


drive9-server



エージェント作業領域の制御

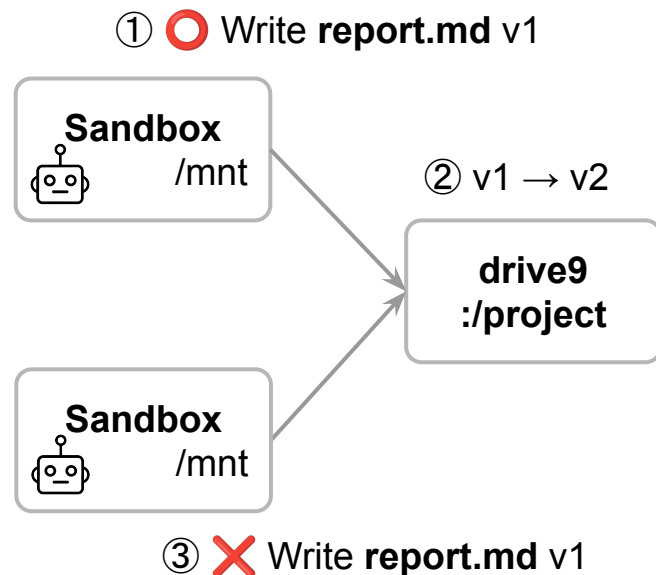
- Drive9はコンテキストと呼ばれるスコープを持つ
 - コンテキスト毎に名前空間も別
- エージェント作業用に、短期のサブコンテキスト（トークン）を作成することができる
 - このとき、パス単位でアクセス制御を付与できる
- エージェントの役割に応じて細かいアクセス制御を可能にすることが可能



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

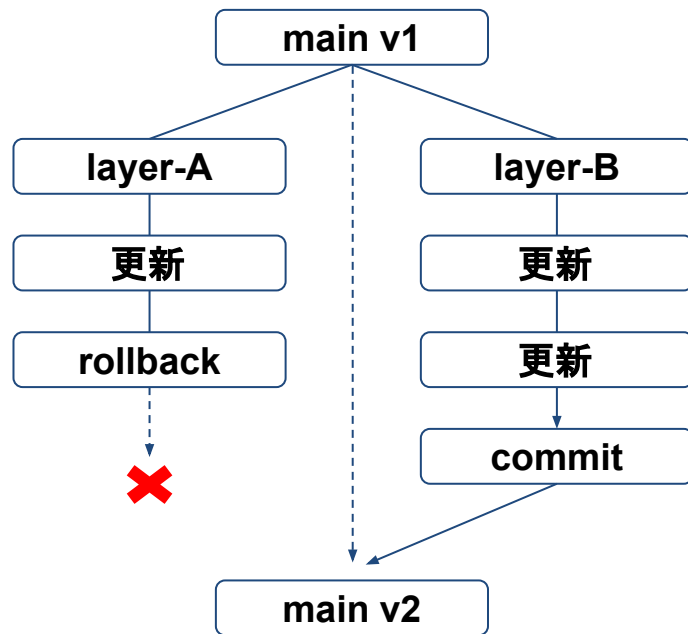
楽観ロックによる競合検知

- 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）



本セッションでは、エージェントサービスの動的なコンテキストと成果物をどこに置くか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、ハーネス部分の開発に注力・開発サイクルを効率よく回せるメリットがある。

また、ユーザー固有データを分離することで、エージェント自体はステートレスに実行することができるという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.

