



EDB Agent Governance

リリース *1.0.0*

EnterpriseDB Corporation

2026 年 07 月 06 日

目次

第 1 章	What you can do	3
1.1	どこから始めるべきか	3
1.2	ハイブリッドマネージャーでエージェントガバナンスを使用する	3
第 2 章	Key concepts	5
2.1	エアマン MCP	5
2.2	セッション	5
2.3	ステップ	5
2.4	目的	5
2.5	インスタンス	5
2.6	クラスター	6
2.7	Loki と LogQL	6
2.8	同期ウォーターマーク	6
第 3 章	Architecture and data flow	7
3.1	データパス	7
3.2	コンポーネントの責任	7
3.3	エージェントアクティビティのタグ付け方法	8
3.4	ログエントリのフォーマット	8
3.5	クエリパス	9
3.6	バックグラウンド同期	9
3.6.1	同期パイプライン	9
3.6.2	アダプティブチャンク分割	10
3.6.3	再試行ポリシー	10
3.6.4	再同期	10
第 4 章	Deploying and configuring	11
4.1	コンテナスタックを理解する	11
4.2	前提条件	11
4.3	導入前構成	11
4.3.1	必須	12
4.3.2	オプション	12
4.4	事前構築されたイメージを使用した展開	13
4.5	ID プロバイダーの管理	13
4.6	デプロイメントの保護とモニタリング	13
第 5 章	Connecting data sources	15
5.1	インスタンスタイプ	15
5.2	HM インスタンスの追加	15
5.2.1	HM インスタンスがセッションデータを表示するための要件	16
5.3	Loki インスタンスの追加	16

5.4	インスタンスの削除	16
5.5	インスタンスの再同期	16
第 6 章	Browsing sessions	19
6.1	セッションリストの検索	19
6.2	セッションテーブルの読み取り	19
6.3	セッションの検索	20
6.4	結果によるフィルタリング	20
6.5	ページサイズの構成	20
6.6	セッションを開く	20
第 7 章	Inspecting a session	21
7.1	ページヘッダーの検査	21
7.2	ステップバイステップの再構成の表示	21
7.2.1	折りたたまれたステップ	21
7.2.2	展開されたステップ	21
7.2.3	アコーディオンモード	22
7.3	フラット監査ログの表示	22
7.3.1	監査ログ列	22
7.3.2	監査ログの検索	22
7.3.3	ページサイズの構成	22
第 8 章	Securing access and handling credentials	23
8.1	ユーザーの認証	23
8.2	上流ソースの認証	23
8.3	資格情報の保存	24
8.4	ネットワークの保護	24
8.5	データの取り扱い	24

EDB Agent Governance は、Postgres データベースで動作する AI エージェントの可観測性と監視機能を提供するスタンドアロンアプリケーションです。これにより、データベース管理者、セキュリティチーム、および AI オペレーションエンジニアは、AI エージェントがエンタープライズデータと相互作用する方法について監視、監査、および推論できます。

アプリケーションは、1 つ以上の上流ログソース EDB Hybrid ManagerHM 展開またはスタンドアロンの Grafana Loki エンドポイントに接続して、Airman Model Context ProtocolMCP サーバーを介して AI エージェントが生成する Postgres クエリログを取得します。これらのログは、エージェントの動作をフィルタリング、検査、監査するためのツールを使用して、構造化セッション単一の AI エージェントのインタラクション内で実行される SQL ステートメントの論理グループとして表示されます。修飾されたログソースから読み取るため、HM 管理データベースに制限されません。

第1章 What you can do

- **エージェントセッションを再構成します** — 期間、ターゲットデータベース、Postgres ロール、エラーの重大度などの完全なメタデータを使用して、AI エージェントが実行したすべての SQL ステートメントを表示します。
- **マルチプルのデータソースにまたがる監査** — 一度に複数の HM または Loki ソースに接続し、1 か所からそれらのすべてのクラスターを監査します。
- **フラットな監査証跡を読み取る** — セッション内のすべてのステートメントを、レビュー用に最適化された単一のフィルタリング可能なテーブルとしてスキャンします。
- **キャッシュされたインクリメンタル同期に依存アプリケーションはセッションデータをインクリメンタルに同期およびキャッシュするため、訪問のたびに完全なログ履歴を再照会せずに監査ビューがすばやくロードされます。**

1.1 どこから始めるべきか

- *Key concepts* アプリケーション全体で使われるキーワードセッション、ステップ、目的、インスタンス、および Airman MCP がアクティビティをタグ付けする方法。
- *Architecture and data flow* — エージェントアクティビティが Postgres ログからアプリケーションにどのように伝わるか。
- *Deploying and configuring* — アプリケーションの展開と構成を行う管理者用。
- *Connecting data sources* — HM または Loki インスタンスを登録して、エージェントセッションが表示されるようにします。
- *Browsing sessions* — クラスターで検出された AI エージェントセッションを検索およびフィルタリングします。
- *Inspecting a session* — 単一のセッションを段階的に検査し、その監査ログを読み取ります。
- *Securing access and handling credentials* — 認証、資格情報ストレージ、およびデータの処理方法。

1.2 ハイブリッドマネージャーでエージェントガバナンスを使用する

Postgres クラスターがハイブリッドマネージャーによって管理されている場合、HM 固有の統合の詳細については、[Governance in Hybrid Manager](#) を参照してください。マシンユーザー API キー、HM の Loki パイプライン、およびプロジェクトとクラスター検出。

第2章 Key concepts

これらの用語を確認して、AI ガバナンス監査ログビューアを理解および効果的に使用します。

2.1 エアマン MCP

Airman MCP は、AI エージェントの Postgres アクセスを仲介するモデルコンテキストプロトコル MCP サーバーです。エージェントが呼び出すことができるツールとして Postgres 操作を公開し、`airman:<purpose>/<session-short>` フォームの Postgres `application_name` 変数を介して実行されるすべてのクエリーにタグ付けするため、後でアクティビティを再構成できます。ビューアはこれらのタグに完全に依存して、エージェントアクティビティをグループ化および属性付けます。

2.2 セッション

セッションは、単一の Airman MCP インタラクション内で AI エージェントによって実行される SQL ステートメントの論理グループです。各セッションは、ビューアでの監査の主な単位です。クラスターを開くとそのセッションが表示され、セッションを開くと、それに含まれるステートメントが表示されます。

Airman MCP は、セッションショート MCP セッショントークンの最初の 8 つの 16 進文字でセッションを識別します。セッション short を共有するすべてのログエントリは、同じセッションに属します。

2.3 ステップ

ステップは、セッション内の単一の SQL ステートメント、つまり 1 つの解析された Postgres ログレコードです。各ステップには、SQL テキストとその実行メタデータターゲットデータベース、Postgres ロール、バックエンドプロセス ID、期間、エラーの重大度が含まれます。

2.4 目的

目的は、`AIRMAN_MCP_PURPOSE` 環境変数たとえば `billing`、`support`、または `analytics` などを通して Airman MCP インスタンスごとに構成された宣言されたインテントラベルです。これは、Airman が各クエリーに書き込むタグの最初のセグメントとして表示され、セッションリストとセッションの詳細の両方に表示されます。目的はビューアではなくエージェント側で設定されます。

2.5 インスタンス

インスタンスは、登録された上流データソース—ハイブリッドマネージャー HM 展開またはスタンドアロンの Loki エンドポイントのいずれかです。ビューアは一度に複数のインスタンスに接続し、それらのすべてからクラスターとセッションを集約できます。[Connecting data sources](#) を参照してください。

2.6 クラスター

クラスターは、HM によって管理されるか、Loki によって監視される Postgres クラスターです。ビューアは、監査のエントリポイントとしてクラスターをリストします。クラスターを選択し、それに対して検出された AI エージェントセッションを確認します。

2.7 Loki と LogQL

Loki は、ビューアが読み取った Postgres クエリログを保存するログ集計システムです。クエリ言語 LogQL で表現されるラベルベースのクエリ用に最適化されています。HM には Loki パイプラインが含まれています。Postgres ログを受信するスタンドアロンの Loki インスタンスにビューアをポイントすることもできます。ビューアのバックエンドは LogQL クエリーを構成して、クラスターのセッションタグ付けのログエントリを見つけます。

2.8 同期ウォーターマーク

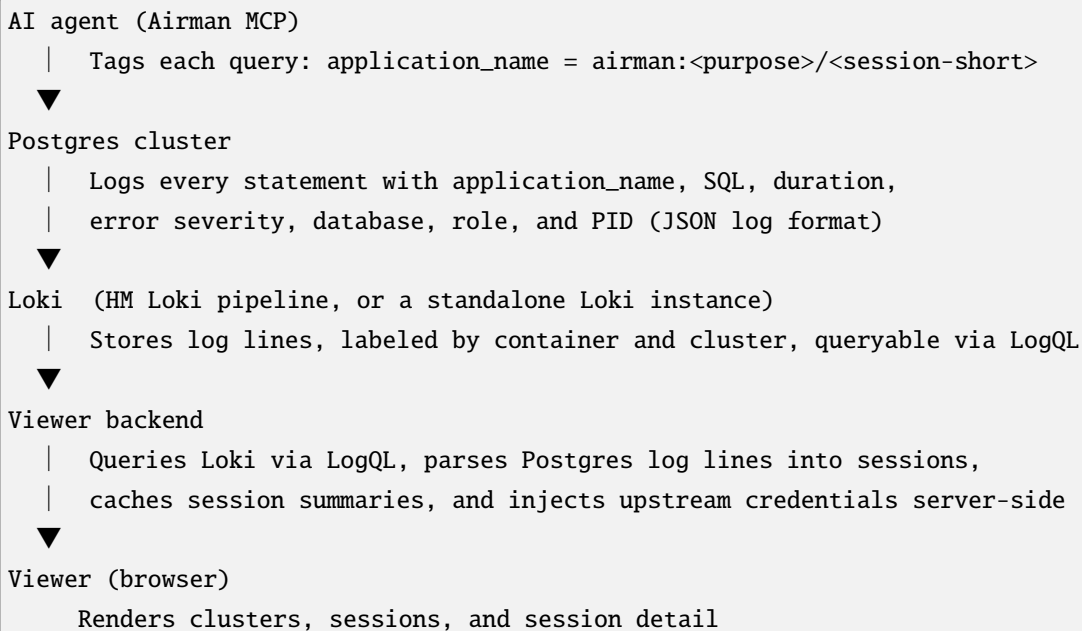
同期ウォーターマークは、インスタンスごと、クラスターごとのレコードで、ビューアのバックエンドが Loki からセッションデータをキャッシュした範囲を追跡します。最も古い既知のエントリーと、キャッシュがどのくらいの距離まで書き込まれているかを記録します。これにより、訪問するたびに完全なログ履歴を再照会するのではなく、高速なインクリメンタル同期が可能になります。[Architecture and data flow](#) を参照してください。

第3章 Architecture and data flow

AI エージェントアクティビティは、エージェントが実行する SQL から Postgres ログと Loki を介して、読み取った画面に移動します。パスを理解することは、ビューアーに表示できるものと、セッションデータが表示される前に同期が必要な場合がある理由を推論するのに役立ちます。

3.1 データパス

AI エージェントが実行する各クエリは、ビューアに監査可能なセッションとして表示される前に4つのステージを通過します。



3.2 コンポーネントの責任

3つのコンポーネントはデータベースの異なる部分进行处理し、バックエンドがすべての上流アクセスを仲介するため、資格情報がブラウザに到達することはありません。

ビューアブラウザシングルページアプリケーションは、クラスター、セッション、およびセッションの詳細画面をレンダリングし、クライアント側のルーティングを管理し、すべてのデータのバックエンドを呼び出します。

ビューアバックエンド 次の軽量サービス。

- インスタンス構成と、関連するアップストリーム資格情報を保存します。

- インスタンスタイプに基づいて、各要求を正しい HM または Loki エンドポイントにルーティングします。
- 生の Postgres JSON ログ行を構造化されたセッションおよびステップデータに解析します。
- セッションの概要をキャッシュし、段階的に同期して、冗長なクエリを回避します。
- プロキシ化されるすべての要求に適切な認証ヘッダーを挿入するため、資格情報がブラウザーに到達することはありません。

アップストリームソース HM 展開 API を介してプロジェクト、クラスター、および Loki ログクエリを公開し、マシンユーザー API キーで認証します。または、オプションのベアラートークン認証を使用して、直接照会されるスタンドアロンの Loki インスタンス。

3.3 エージェントアクティビティのタグ付け方法

Airman MCP は、開くすべての接続に Postgres `application_name` セッション変数を自動的に設定します。

```
SET application_name = airman:<purpose>/<session-short>;
```

そこで

- `purpose` — AIRMAN_MCP_PURPOSE たとえば `billing`、`support`、または `analytics` を介して Airman インスタンスごとに構成された静的ラベル。設定されていない場合は、デフォルトの `_` になります。
- `session-short` — MCP セッショントークンの最初の 8 つの 16 進文字。すべてのクエリーを 1 つのインタラクションにグループ化するために使用されます。

タグは Postgres セッションレベルで設定され、その接続の後続のすべてのステートメントに対して持続します。

3.4 ログエントリのフォーマット

JSON ロギングを有効にすると、Postgres は次のようなエントリを生成し、Loki が取り込みます。

```
{
  "ts": "2026-05-22T14:30:45.123Z",
  "record": {
    "application_name": "airman:billing/a1b2c3d4",
    "message": "Duration: 45.2 ms statement: SELECT customer_id, email FROM_
↪customers WHERE active = true",
    "database_name": "banking_db",
    "user_name": "agent_read",
    "command_tag": "SELECT",
    "error_severity": "",
    "log_time": "2026-05-22 14:30:45.123 UTC",
    "process_id": "12345",
    "session_id": "789",
```

(次のページに続く)

(前のページからの続き)

```

    "sql_state_code": "",
    "query_id": "0"
  }
}

```

バックエンドは `message` フィールドから SQL テキストと期間を抽出し、残りのフィールドをステップメタデータに直接マッピングします。

3.5 クエリパス

ビューアは、表示しているものに応じて、さまざまな Loki クエリを発行します。

セッションリスト クラスターを開くと、バックエンドは同期ウォーターマークを読み取り、Loki からキャッシュされていない履歴をバックフィルし、キャッシュされたセッション概要をロードし、最新の日のライブクエリを追加し、結果をマージします—セッションの並べ替え最近見た人によって。

セッションの詳細 セッションを開くと、バックエンドはそのクラスターにスコープし、セッションショートにフィルタリングした LogQL クエリを発行し、返された各ログラインをステップの順序付けたリストに解析します。

バックエンドは、次の形式の LogQL クエリを使用します。

```
{container="postgres", cnp_cluster_id=~"<clusterId>.*"} |= "airman:"
```

セッションの詳細では、`session short` がラインフィルタとして `airman:` を置き換えます。

3.6 バックグラウンド同期

ページが読み込まれるたびに高価なフルレンジの Loki クエリを回避するため、バックエンドはセッション概要をキャッシュし、ウォーターマークベースのインクリメンタル同期で最新の状態に保ちます。

3.6.1 同期パイプライン

クラスターのセッションリストが要求されると、バックエンドは次のことを行います。

1. **ウォーターマークを読み取ります** —最も古い既知のエントリのインスタンスごと、クラスターごとのレコードと、キャッシュがどの程度まで移入されたか。最初の同期では、両方ともゼロです。
2. **最も古いエントリを見つけます** —最初の同期で、30 日間、90、次に 365 にわたって逆方向に検索して、キャッシュするフルタイム範囲を確立します。何も見つからない場合、今日が最も古いものとして扱われます。
3. **バックフィル履歴** —最初のエントリから今日までキャッシュを 1 日単位で埋め、セッションの概要を `upsert` し、ウォーターマークを進めます。進行状況は、ステータスエンドポイントを介して報告されます。
4. **キャッシュされたセッションのロード** —クラスターのキャッシュされたすべての概要を読み取ります。

5. **今日のライブデータを取得します** — 過去 24 時間に単一のクエリーを実行します。このデータは常に新しく、キャッシュされません。

次に、バックエンドは、キャッシュされたセッションとライブセッションをセッションショートごとにマージし、歩数を合計し、最初と最後に表示されたタイムスタンプを調整し、最後に表示された順に並べ替えます。

3.6.2 アダプティブチャンク分割

高冗長クラスターは、Loki のクエリごとの制限を超える場合があります。1 日分のチャンクが高すぎると Loki が報告すると、バックエンドはウィンドウを半分に分割し、各半分の独立して再試行し、最小チャンクサイズ 1 時間まで下げます。1 時間のチャンクでもまだ失敗する場合は、エラーをログに記録し、そのウィンドウをスキップします。

3.6.3 再試行ポリシー

アップストリーム Loki 呼び出しは、再試行ポリシーでラップされます。一時的なエラー 502、503、504 の場合、指数バックオフ 2 秒から開始、2 倍、30 秒で上限 30 秒で最大 5 回の再試行。再試行不可エラー 400、401、404、500 は、すぐに失敗します。

3.6.4 再同期

Configure ページから再同期をトリガーすると、バックエンドはインスタンスのキャッシュセッションとウォーターマークをクリアし、バックグラウンドジョブを開始します。Loki インスタンスの場合、単一のスタブクラスターを再同期します。HM インスタンスの場合、すべてのプロジェクトとクラスターを検出し、それらを並列同期します。要求はすぐに返され、ジョブが実行されると、インスタンスのステータスが **Syncing** から **Done** または **Error** に進みます。 [Connecting data sources](#) を参照してください。

第4章 Deploying and configuring

EDB Agent Governance 監査ログビューアを展開および構成します。これらの設定の背後にある認証モデルと資格情報の処理については、[Securing access and handling credentials](#) を参照してください。

4.1 コンテナスタックを理解する

監査ログビューアは、4つのコンテナの Dex フロントスタックとして展開されます。

Service	Role
<i>nginx</i>	Terminates HTTPS, serves the viewer's static assets, and routes API and single-sign-on requests to the right service so the browser talks to a single origin.
<i>bff</i>	The backend — OIDC client of Dex, instance management, and the HM/Loki proxy that parses logs and caches sessions.
<i>dex</i>	Identity federation — brokers login to your upstream identity providers.
<i>bootstrap</i>	A one-shot helper that generates secrets and the internal certificates the stack needs on first run.

共有 Postgres サービスはスタックをバックし、インスタンス構成とセッションのバックエンドのデータベース、および ID プロバイダーコネクタの Dex データベースをホストします。

4.2 前提条件

- Docker 24 以降と Compose v2 を使用するコンテナホスト。
- ユーザーが信頼する認証局からの、ドメインの TLS 証明書とキー。自己署名証明書は、ローカルテストで使用できます。
- 監査する少なくとも1つのアップストリームデータソース [HM instance with a machine user API key](#)、または AI エージェントワークロードから Postgres ログを受信するスタンドアロンの Loki インスタンス。

4.3 導入前構成

ブートストラップヘルパーを実行する前に、次の値を設定します。ブートストラップヘルパーは、その他すべてを自動的に生成および永続化します。

4.3.1 必須

Setting	Required	Description
Public base URL	Yes	パブリックオリジン — スキームとホスト、パスと後続のスラッシュなし、`https://governance.example.com`など。OIDC 発行者とリダイレクト URI は、そこから派生します。
Bootstrap admin email	Yes	最初のブレイクグラス管理者のメールアドレス。デフォルトの管理許可リストも。
Postgres password	Production	共有 Postgres サービスのパスワード。
Image registry	Production	事前構築されたイメージのレジストリ。
Image tag	Production	イメージタグ — リリースバージョンにピン留めします。

ブートストラップヘルパーが生成する値共有クライアントシークレット、セッション署名キー、トークン暗号化キー、およびブートストラップ管理の識別子とパスワードのハッシュは、自動的に書き込まれ、手動で設定しません。

4.3.2 オプショナル

運用展開には次の設定をお勧めします。

Setting	Default	Description
Database connection	Points at the shared Postgres service	バックエンドのデータベースの接続文字列。
Upstream egress to private ranges	Blocked	登録されたインスタンスがプライベートネットワーク範囲に到達できるかどうか。HM または Loki 上流がプライベート IP に存在するオンプレミス展開にのみ許可されます。 Securing access and handling credentials を参照してください。
OIDC CA bundle	—	自己署名またはプライベート CA OIDC 発行者を信頼する CA バンドル。OIDC TLS 検証はオンのままです。これは、信頼をバイパスするのではなく、追加します。
Session idle timeout	2h	アイドルセッションの有効期限が切れるまでの時間。
Session absolute lifetime	12h	アクティビティに関係ないセッションの最大有効期間。
Admin allowlist	The bootstrap admin email	`groups` クレームに `governance-admin` が含まれるユーザーに加えて、管理者アクセスが付与されたメール。

4.4 事前構築されたイメージを使用した展開

事前にビルドされた EDB 公開イメージを使用します。ソースからビルドする必要はありません。

1. **TLS 証明書を提供します。** スタックが想定する場所に証明書とプライベートキーを配置しますフルチェーンとプライベートキー。
2. **展開を構成します。** サンプル環境ファイルをコピーし、必要な値-少なくともパブリックベース URL とブートストラップ管理電子メール、およびイメージレジストリとタグ**必須**を参照してください。
3. **シークレットを生成して開始します。** 運用開始スクリプトを実行します。共有シークレット、セッションキーと暗号化キー、ブートストラップ管理の識別子とパスワード、および内部証明書を生成するブートストラップヘルパーを実行し、スタックをアップします。
4. **サインインとフェデレート。** ブラウザーで展開を開き、ブートストラップヘルパーが記録したパスワードを使用して、ブートストラップ管理者としてサインインします。次に、**設定 →ID プロバイダー**の下に上流の ID プロバイダーを追加し、日常のログインをそれに切り替えます。

注釈

latest を実行するのではなく、リリースバージョンにイメージタグを固定するため、展開が再現可能です。

4.5 ID プロバイダーの管理

設定 →ID プロバイダーの下で監査ログビューアで、ユーザーがサインインする ID プロバイダーを構成します。

1. 組織の ID プロバイダーの OIDC、SAML、または LDAP コネクタを追加します。
2. フェデレーション後、日常のログインをブートストラップ管理からフェデレーションプロバイダーに切り替え、ブートストラップ管理をブレイクグラスアカウントとして維持します。
3. 管理者許可リストを、管理者アクセスを保持する人に制限します。

4.6 デプロイメントの保護とモニタリング

展開が実行されたら、次のことに注意してください。

- **データボリュームを暗号化します。** 共有 Postgres ボリュームには、インスタンスアクセスキー、セッション、および ID プロバイダーコネクタのシークレットが保持されます。保存時暗号化ストレージに保存します。ボリュームが失われると、データベースは空の状態で作成されます。インスタンスとコネクタは、バックアップから再追加するか、復元する必要があります。[Securing access and handling credentials](#) を参照してください。
- **ブートストラップ資格情報を保護します。** 記録されたブートストラップ管理パスワードを機密として扱い、パスワードを安全に保存したらファイルを削除します。

- 上流の TLS 検証をオンのままにします。運用中の上流の HM または Loki ソースの TLS 検証を無効にしないでください。
- **Monitor liveness.** Dex は、バックエンドと nginx の起動をゲートするヘルスエンドポイントを公開します。コンテナのステータスと nginx 上流エラーを介して全体的な活性度を観察します。

第5章 Connecting data sources

1つ以上の上流データソースを登録して、監査ログビューアで AI エージェントセッションを利用できるようにします。インスタンスは、登録されたアップストリームデータソースです。EDB Hybrid ManagerHM 展開またはスタンドアロンの Grafana Loki エンドポイントのいずれかです。ビューアは同時にマルチプルのインスタンスに接続し、それらすべてのクラスターとセッションを集約できます。アクセスキーを含むインスタンス構成はサーバーサイドに保存され、ブラウザーに送信されることはありません。

Configure ページでインスタンスを管理し、ホームページの **Configure Instances** ボタンまたはヘッダーの **Settings** → **Configure Instances** からアクセスします。

5.1 インスタンスタイプ

Type	Description	Authentication	Discovery
HM	EDB Hybrid Manager データプレーン	Machine user API key, sent in the <i>x-access-key</i> header	Auto-discovers projects and clusters
Loki	ダイレクト Loki ログストア	Optional <i>Authorization</i> header	Creates a single stub project and cluster automatically

5.2 HM インスタンスの追加

1. **Configure Instances** ページを開きます。
2. **Add New** を選択します。
3. フォームに入力します。
 - **Type** — HM.
 - **Name** — 説明的なラベル、例、Production EU.
 - **Base URL** — HM API エンドポイント、例、https://hm.example.com:8443.
 - **Access Key** — マシンユーザー API キー。これらのキーは通常、`hmak_` または `baak_` で始まります。
4. **Add Instance** を選択します。

成功すると、ビューアのバックエンドは HM インスタンスからプロジェクトとクラスターを検出し、これらのクラスターは数秒以内にホームページに表示されます。

5.2.1 HM インスタンスがセッションデータを表示するための要件

HM インスタンスが AI エージェントセッションを表面化する場合、以下が満たされる必要があります。

- Postgres ステートメントロギングは、ターゲットクラスター `log_statement = 'all'` または `log_min_duration_statement = 0` で有効になっています。
- クラスターは、ビューアのバックエンドからネットワークを介して到達可能です。
- AI エージェントワークロードは、トレースを有効にして Airman MCP を介して実行されるため、アクティビティがタグ付けされて検出可能になります。

HM 管理ポータルでマシンユーザー API キーを作成します。ビューアはそれをサーバーサイドに保存し、ブラウザーに返すことはありません。資格情報の処理の詳細については、[Securing access and handling credentials](#) を参照してください。

5.3 Loki インスタンスの追加

1. **Configure Instances** ページを開きます。
2. **Add New** を選択します。
3.
 - **Type** — Loki.
 - **Name** — 説明的なラベル、例、Dev Loki .
 - **Base URL** — Loki HTTP エンドポイント、例、`http://loki.internal:3100` .
 - **Authorization** — オプション。ベアラートークンまたはその他の承認ヘッダー値 Loki エンドポイントが必要とする場合。
4. **Add Instance** を選択します。

Loki インスタンスの場合、ビューアは単一のスタブプロジェクトとクラスターを作成し、すべての Loki セッションがその下に表示されます。

5.4 インスタンスの削除

Configure ページの各インスタンス行には、**Delete** ボタンがあります。それを選択すると、確認のポップオーバーが表示されます。確認時に、ビューアはインスタンスレコード、そのキャッシュされたセッションデータ、および同期ウォーターマークを削除します。

削除は元に戻せません。インスタンスを再追加するには、上流からの履歴を再同期する必要があります。

5.5 インスタンスの再同期

バックエンドは、高速な取得のためにセッション概要をキャッシュします。上流データが修正またはバックフィルされたとき、キャッシュが一貫していないように見えるとき、またはキャッシュデータベースが失われたときに、完全な再同期を強制します。

1. **Configure** ページで、インスタンス行を見つけます。

2. **Resync** ボタン 同期アイコンを選択します。
3. バックエンドはインスタンスのキャッシュデータをクリアし、完全なバックグラウンド同期を開始します。

インスタンスは、その同期ステータスを **Syncing** スピナー付き、**Done** セッションカウント付き、または **Error** メッセージ付きとして示します。再同期はバックグラウンドで実行されるため、完了している間ビューアの使用を継続できます。同期の仕組みの詳細については、[Architecture and data flow](#) を参照してください。

i 注釈

****Resync**** ボタンは、開発者ツールが有効になっている場合にのみ表示されます。それ以外の場合、バックエンドは、クラスターへの最初のアクセスに自動的に同期します。

第6章 Browsing sessions

セッションリストを使用して、クラスターで検出されたすべての AI エージェントセッションを監査します。各セッションは完全なインタラクション、つまり AI エージェントが単一の Airman MCP セッション内で実行したすべての SQL ステートメントを表します。

6.1 セッションリストの検索

Home ページには、構成済みのすべてのインスタンスのすべてのクラスターがリストされます。各クラスターカードには、クラスター名前、そのステータス、およびプロジェクトが表示され、検索ボックスがリストをフィルタリングします。クラスターカードを選択して、**セッションリスト**を開きます。

構成済みのインスタンスのロードに失敗すると、**Home** ページにインスタンスごとのエラーアラートが表示されます。インスタンスがまだ構成されていない場合、[**Configure Instances**] ボタンでゼロ状態が表示されます。 [Connecting data sources](#) を参照してください。

クラスターを開くと、**セッションリスト**が自動的にロードされ、必要に応じてバックグラウンド同期がトリガーされるため、上流ソースからの最新データが表示されます。

6.2 セッションテーブルの読み取り

セッションテーブルは、カスタマイズ可能な列の可視性をサポートしています。

Column	Default	Sortable	Description
Session ID	Always visible	Yes	等幅識別子。それを選択して、 Session Detail ページを開きます。
Purpose	Visible	Yes	Airman MCP 構成から宣言された目的。
Started	Visible	Yes	セッション内の最初のステートメントのタイムスタンプ。
Commands	Hidden	No	セッション内の SQL コマンドタイプ SELECT、INSERT、UPDATE などを示すタグ。
Steps	Visible	Yes	実行された SQL ステートメントの合計数。

次のページに続く

表 1 – 前のページからの続き

Column	Default	Sortable	Description
Errors	Visible	Yes	重大度がエラーになったステートメントの数。ゼロより大きい場合、赤いバッジとして表示されます。

表示される列を変更するには、ツールバーの列の可視性ボタン歯車アイコンを選択し、チェックボックスリストを使用します。セッション ID は常に表示され、非表示にすることはできません。

6.3 セッションの検索

ツールバーの検索フィールドは、セッション ID と目的を超えて全文一致を実行します。用語を入力して **Enter** を押すか、オートフィルタリングを待ちます。Clear ボタンを使用して、検索をリセットします。

6.4 結果によるフィルタリング

ツールバーのフィルタを使用すると、結果でリストを絞り込むことができます。

Filter	Options	Description
Outcome	OK, Issues	正常に完了したセッションのみ、または少なくとも 1 つのエラーを含むセッションのみを表示します。

フィルタは単一選択です。別のオプションを選択すると、現在のオプションが置き換えられます。すべてのセッションを再度表示するには、それをクリアします。

6.5 ページサイズの構成

表には、デフォルトでページごとに 10 セッションが示されており、標準の前および次のナビゲーションを使用して 10、20、または 50 に構成可能です。

6.6 セッションを開く

セッション ID を選択して *Inspecting a session* ページを開きます。ここで、各 SQL ステートメントを段階的に検査し、そのセッションの完全な監査ログを読み取ることができます。

第7章 Inspecting a session

Session Detail ページを使用して、AI エージェントが実行したすべての SQL ステートメントとその実行メタデータを、ステップバイステップの再構成とフラット監査ログの2つのビューで検査します。

7.1 ページヘッダーの検査

ヘッダーには次のものが示されます。

- **セッション ID** — 大きな等幅のタイトルとしてのセッション識別子。
- **Started** — 最初のステートメントのフォーマットされたタイムスタンプ。
- **Purpose** — 宣言された目的を示すタグ利用可能な場合。
- **Steps** — 合計ステートメント数を示すバッジ。

7.2 ステップバイステップの再構成の表示

[**Reconstruction**] タブを使用して、セッションを時系列でステップ実行します。各ステップは、完全な実行コンテキストを含む単一の SQL ステートメントであり、拡張可能なアコーディオンとして表示されます。

7.2.1 折りたたまれたステップ

折りたたまれた各ステップは、一目で次のことを示します。

- ステップ番号、例、ステップ 1。
- SQL コマンドタイプのコマンドタグ **SELECT**、**INSERT**、**UPDATE**、**DELETE** など。
- ステートメントにエラーが発生した場合にのみ表示されるエラーの重大度タグ **ERROR**、**FATAL**、または **PANIC** の場合は赤。
- タイムスタンプと期間、右揃え—例 **14:30:45** ・ **45ms**。

7.2.2 展開されたステップ

ステップを選択すると、展開され、要求パネルが表示されます。これはデフォルトで表示され、以下を含みます。

- **SQL ステートメント** — 構文が強調表示されたコードブロック内のステートメント。
- **SQLSTATE コード** — **00000** 成功でない場合にタグとして表示されます。
- **プロパティ** — 実行メタデータのグリッド

Property	Description
Database	ターゲットデータベース名
User	使用される Postgres ロール
PID	バックエンドプロセス ID
PG Session ID	Postgres セッション識別子
Query ID	内部クエリプラン ID
Duration	ミリ秒単位の実行時間
application_name	完全な Airman タグ、例 <i>airman:billing/a1b2c3d4</i>

7.2.3 アコーディオンモード

デフォルトでは、一度に複数のステップを展開できます。設定でアコーディオンステップが有効になっている場合、ステップを展開すると、以前に開いたものが自動的に折りたたまれます。画面スペースが制限されている、多くのステップがあるセッションに役立ちます。

7.3 フラット監査ログの表示

[**Audit Log**] タブは、同じセッションデータをフラットテーブルとして表示し、一度にすべてのステップのスキャンとフィルタリング用に最適化されています。

7.3.1 監査ログ列

Column	Description
Time	タイムスタンプ、等幅、右揃え
SQL	ステートメントテキスト等幅、省略記号で切り捨てられます
Purpose	宣言された目的

7.3.2 監査ログの検索

ツールバーの検索フィールドは、SQL テキスト、目的、およびタイムスタンプにわたって一致します。用語を入力してテーブルをフィルタリングし、クリアしてすべてのステップを再度表示します。

7.3.3 ページサイズの構成

表には、デフォルトでページごとに 10 行が表示され、10、20、または 50 に構成可能です。

第8章 Securing access and handling credentials

ハイブリッドマネージャー HM によって管理またはスタンドアロンの Loki インスタンスを介して監視される Postgres クラスターで動作している監査 AI エージェント。監査ログビューアは、機密データを含む可能性のある Postgres クエリログを読み取り、監査対象の上流ソースの資格情報を保持します。2つの異なる認証レイヤーがあり、すべての資格情報はサーバーサイドでライブです。ブラウザーに公開されるものはありません。

Layer	Question it answers	Mechanism
User authentication	Who may sign in to the viewer?	Identity federation through Dex (OIDC)
Upstream authentication	How does the backend reach HM or Loki?	Machine user API keys

8.1 ユーザーの認証

上流の ID プロバイダーとフェデレーションする Dex を介してサインインします。

1. バックエンドは、Dex の OIDC クライアントです。Dex は、設定済みのコネクタ OIDC、SAML、または LDAP へのログインをブローカーし、どの ID プロバイダーがユーザーを認証したかを報告します。
2. 成功すると、バックエンドはサーバーサイドセッション、つまり署名された Cookie と、保存時に暗号化されたリフレッシュトークンと ID トークンをミントします。トークンはブラウザーに到達しません。
3. ID プロバイダーが構成される前に、ブートストラップ管理が存在します。最初のログインにそれを使用し、次にフェデレーション ID プロバイダーを追加して、毎日のログインをそれに切り替えます。
4. **管理ゲーティング**メールが管理許可リストに含まれている場合、または **groups** クレームに **governance-admin** が含まれる場合、ユーザーは管理者です。ID プロバイダー管理などの管理専用サーフェスは、これにゲートされます。

ID プロバイダーの構成については、[Deploying and configuring](#) を参照してください。

8.2 上流ソースの認証

マシンユーザー API キーを使用して、上流ソースへの認証を行います。HM データプレーンまたは Loki インスタンス全体へのアクセスを許可する共有資格情報。これらのキーは次のとおりです。

- HM 管理ポータルで作成される HM インスタンスの場合、または外部構成 Loki の場合。
- バックエンドの Postgres データベースに保存されます。

- プロキシ化されたすべての要求にサーバーサイドで挿入されます。
- ブラウザーへの応答には含まれません。サーバー上でクリアされ、省略されます。

バックエンドは、インスタンスタイプに基づいて、適切なヘッダーを追加します。

Instance type	Header	Format
HM	<i>x-access-key</i>	The raw key value, for example <i>hmak_abc123...</i>
Loki	<i>Authorization</i>	The full header value, for example <i>Bearer token123...</i>

8.3 資格情報の保存

バックエンドは、Postgres データベースに 2 つの資格情報ストアを維持します。

- ビューアのデータベースには、インスタンスアクセスキーとユーザーセッションが保持されます。
- Dex データベースは、OIDC クライアントシークレットや LDAP バインドパスワードなどの ID プロバイダーコネクタのシークレットを平文で保持します。

基になるストレージボリュームは、暗号化された保存ストレージ LUKS または dm-crypt、クラウドディスク暗号化、または同等のものに保持します。特に Dex コネクタ資格情報にはアプリケーションレイヤーの暗号化がないため、ボリュームレベルの暗号化が重要な保護となります。

8.4 ネットワークの保護

ビューアは、いくつかのネットワークレベルの保護を適用します。

TLS 検証。 OIDC TLS 検証—検出、JWKS、およびトークン交換用—は常にオンで、無効にすることはできません。自己署名またはプライベート CA 発行者を信頼するには、検証をバイパスするのではなくシステム信頼プールに追加される CA バンドルを提供します。

アップストリームの下り制御。 デフォルトでは、バックエンドはプライベートネットワーク範囲へのインスタンスの下りをブロックするため、登録されたインスタンスを使用して内部ネットワークをプローブすることはできません。クラウドメタデータ、ループバック、およびリンクローカルアドレスは、それに関係なくブロックされたままです。HM または Loki 上流がプライベート IP で合法的に存在するオンプレミス展開にのみプライベートレンジを許可します。

クロスオリジン要求。 ビューアとそのバックエンドは、通常の実操作で単一のオリジンから提供されるため、バックエンド API へのクロスオリジンアクセスは、明示的に許可したオリジンに制限されます。

8.5 データの取り扱い

実行された SQL テキストエージェントと、ターゲットデータベース、Postgres ロール、期間などの実行メタデータを含む Postgres クエリログを確認します。