

Title: SRE Agent の資料作成ノート

Date: 2026-01-27

Slug: SRE-Agent-Planning

Lang: ja-jp

Category: notebook

Tags: azure, SRE Agent, AIOps

Summary: SRE Agent の概要資料を作成するための勉強ノート

本稿は **Microsoft Learn の記述に基づく事実のみ**を材料として、PowerPoint 化しやすいように「説明文（話す文章）+スライド要点」に再整理したノートです。

引用の出し方:

- 本文中は [章-番号] の参照番号のみを付けます（URL を本文に直書きしません）。
 - 各章の末尾に 参考（番号付き / URL + 短い引用）をまとめます。
-

アジェンダ（案）

大枠は「機能紹介」と「デモ」。各スライドは「説明文（話す文章）+要点」で作る。

機能紹介（構成案）

- イントロ: 何ができる/何ができない（承認が必要な write、英語UIなど）[1-2][1-3]
- 導入前提: RBAC とネットワークの落とし穴[2-1][2-2]
- セキュリティ: “誰が/何で/どの範囲で” の3要素で説明[3-1]
- 実行: Review/Autonomous の違いと同意モデル[5-1][5-5]
- 運用: Incident management と response plan で何が自動化できるか[7-1][7-2]
- コンテキスト: Memory system とチーム共有の前提[8-2][8-9]
- 拡張: Subagent / Tools / Connectors / MCP の関係と境界[9-2][9-3]

デモ（構成案）

- デモA: “App Service が遅い/タイムアウト” → Cosmos DB 429/ホットパーティション切り分け（既存）
- デモB: “散発的タイムアウト” → Cosmos DB 408 切り分け（既存）

まとめ（構成案）

- 実運用で効くポイント（RBAC/同意/スコープ/メモリ/拡張）を早見表で締める[11-2]
-

0. 公式ページ（最小セット）

- Overview[0-1]
 - Create and use an agent[0-2]
 - Roles and permissions overview[0-3]
 - Managed identity[0-4]
 - Agent run modes[0-5]
-

- Scheduled tasks[0-6]
- Incident management[0-7]
- Incident response plan[0-8]
- Memory system[0-9]
- Subagent builder overview[0-10]
- Connectors[0-11]
- Custom logic (Python)[0-12]
- Custom MCP server[0-13]
- FAQ[0-14]

参考 (第0章)

- [0-1] <https://learn.microsoft.com/en-us/azure/sre-agent/overview> — “AI-powered monitoring, troubleshooting, and remediation capabilities.”
- [0-2] <https://learn.microsoft.com/en-us/azure/sre-agent/usage> — “Make sure that your user account has the `Microsoft.Authorization/roleAssignments/write` permissions”
- [0-3] <https://learn.microsoft.com/en-us/azure/sre-agent/roles-permissions-overview> — “consists of three main components”
- [0-4] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-managed-identity> — “Azure SRE Agent has its own managed identity”
- [0-5] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “generates an execution plan and waits for your consent”
- [0-6] <https://learn.microsoft.com/en-us/azure/sre-agent/scheduled-tasks> — “automate workflows such as monitoring, maintenance, and security checks”
- [0-7] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-management> — “receives alerts from ... Azure Monitor alerts ... PagerDuty ... ServiceNow”
- [0-8] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-response-plan> — “Filters ... Execution mode ... Customize instructions”
- [0-9] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “gives agents the knowledge they need to troubleshoot effectively”
- [0-10] <https://learn.microsoft.com/en-us/azure/sre-agent/subagent-builder-overview> — “Incident response plans or scheduled tasks trigger subagents.”
- [0-11] <https://learn.microsoft.com/ja-jp/azure/sre-agent/connectors#what-are-connectors> — “通信コネクタ ... ナレッジコネクタ ... カスタムコネクタ ... MCPサーバー エンドポイント”
- [0-12] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#create-a-python-tool> — “左側のナビゲーションから [Builder>Subagent builder] ... 作成>ツール ... Pythonツール”
- [0-13] <https://learn.microsoft.com/en-us/azure/sre-agent/custom-mcp-server> — “only accessible through subagents and aren't directly accessible to main Azure SRE Agent.”
- [0-14] <https://learn.microsoft.com/en-us/azure/sre-agent/faq> — “Avoid relying solely on group-based role assignments ... Use the Check Access feature”

1. SRE Agent の基本拳動 (Overview)

何の機能か (説明)

Azure SRE Agent は、運用に関する監視・トラブルシューティング・リメディエーションを AI で支援する、チャットボット型の体験として提供されます。[1-1]

何が大事か（スライド要点）

- 「診断（read）」と「変更（write）」が同じ会話の流れで扱える（ただし write は承認が前提）。[1-2]
- チャット UI の言語制約（英語のみ）があるので、社内展開時は運用手順・プロンプトを英語で標準化する設計が必要です。[1-3]
- エージェント作成時に Application Insights / Log Analytics / Managed Identity が自動作成されるため、運用データや権限の“置き場”が増える点を理解しておく必要があります。[1-4]

会話→診断→変更の流れ

```
graph LR
    U["運用担当者"] --> Chat["ポータル チャット"]
    Chat --> Agent["Azure SRE Agent"]
    Agent --> Read["Read: 診断"]
    Agent --> Plan["Write: 実行計画"]
    Plan --> Consent{"同意しますか？"}
    Consent -->|はい| Act["アクション実行"]
    Consent -->|いいえ| Stop["停止"]

    Agent --> AI["Application Insights"]
    Agent --> LA["Log Analytics ワークスペース"]
    Agent --> MI["マネージド ID"]
```

参考（第1章）

- [1-1] <https://learn.microsoft.com/en-us/azure/sre-agent/overview> — “AI-powered monitoring, troubleshooting, and remediation capabilities.”
- [1-2] <https://learn.microsoft.com/en-us/azure/sre-agent/overview> — “any action that an agent takes on your behalf requires your approval.”
- [1-3] <https://learn.microsoft.com/en-us/azure/sre-agent/overview> — “English is the only supported language in the chat interface.”
- [1-4] <https://learn.microsoft.com/en-us/azure/sre-agent/overview> — “Azure Application Insights / Log Analytics workspace / Managed Identity”

2. 作成前提（Usage）

何が必要か（説明）

SRE Agent を作成するには、ユーザー側に Microsoft.Authorization/roleAssignments/write の権限が必要です。[2-1]

また、ネットワーク的には *.azuresre.ai への到達性が前提になります。[2-2]

スライド要点（「導入で詰まりやすい点」）

- 権限要件は「SRE Agent リソースの作成」だけでなく、背後で行われる role assignment を通すために重要です。[2-1]
- *.azuresre.ai がブロックされると、Portal 側の体験が破綻するケースがあるため、最初にネットワーク確認を入れるのが現実的です。[2-2]

作成の前提チェック

```
flowchart TD
    S["開始"] --> R{"roleAssignments/write を持つ？"}
    R -->|いいえ| FixRBAC["必要な権限を付与"]
    R -->|はい| N{"*.azuresre.ai に到達可能？"}
    N -->|いいえ| FixNW["FW で allowlist を追加"]
    N -->|はい| OK["エージェント作成の準備 OK"]
```

参考（第2章）

- [2-1] <https://learn.microsoft.com/en-us/azure/sre-agent/usage> — “Make sure that your user account has the Microsoft.Authorization/roleAssignments/write permissions”
- [2-2] <https://learn.microsoft.com/en-us/azure/sre-agent/usage> — “Add *.azuresre.ai to the allowlist in your firewall settings.”

3. セキュリティモデル（3要素 + 境界）

何を説明すべきか（説明）

SRE Agent の権限は、(1) ユーザーが SRE Agent に対して持つロール、(2) エージェント自身の managed identity、(3) 実行モード（run modes）の3要素で説明できます。[3-1]

この3要素は「誰が（RBAC）」「何で（エージェントのID）」「どの範囲で（実行モード+同意/資格情報）」を分解して説明するのに向いています。[3-6]

スライド要点（役割分離の説明）

- SRE Agent のユーザー ロールは 3 種類（Admin / Standard User / Reader）として定義されています。[3-2]
- エージェントの managed identity は Reader / Privileged の permission level を持ち得ます。[3-3]
- “Autonomous” は無条件ではなく、incident management plan のコンテキストに限定されます。[3-4]
- エージェント側のスコープが優先されるという「境界」の説明は、権限迂回の懸念に対する答えになります。[3-5]

補足（説明）：

- 私は要点を「RBAC でユーザー機能を制御」「MI は閲覧者/特権のレベルを持つ」「実行モードは同意の扱いに影響」「特権昇格防止のためエージェントのアクセス許可が優先」の4点で説明しま

す。[3-7]

セキュリティモデルの3要素

```
flowchart LR
    RBAC["ユーザー ロール\n(Admin/Standard/Reader)"] --> UX["ユーザーができること\n(ポータル/チャット)"]
    MI["エージェントのマネージド ID\n権限レベル: Reader/Privileged"] --> ACT["エージェントができること\n(Azure 上の操作)"]
    RM["実行モード\nConsent / Credentials"] --> FLOW["アクション実行の流れ"]

    ACT --> BND["境界:\nエージェントのスコープ/権限が優先"]
    FLOW --> BND
```

参考（第3章）

- [3-1] <https://learn.microsoft.com/en-us/azure/sre-agent/roles-permissions-overview> — “consists of three main components”
- [3-2] <https://learn.microsoft.com/en-us/azure/sre-agent/roles-permissions-overview> — “Three primary roles (*SRE Agent Admin*, *SRE Agent Standard User*, and *SRE Agent Reader*)”
- [3-3] <https://learn.microsoft.com/en-us/azure/sre-agent/roles-permissions-overview> — “either *Reader* or *Privileged* access”
- [3-4] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “you can only enable autonomous mode in the context of an incident management plan”
- [3-5] <https://learn.microsoft.com/en-us/azure/sre-agent/roles-permissions-overview> — “Agent permissions take precedence ... to prevent privilege escalation”
- [3-6] <https://learn.microsoft.com/ja-jp/azure/sre-agent/roles-permissions-overview#key-concepts> — “セキュリティ モデルは、次の 3 つの主要なコンポーネントで構成”
- [3-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/roles-permissions-overview#security-model-at-a-glance> — “ロールベースのアクセス制御 ... アクセス許可レベル ... 操作モード ... エージェントのアクセス許可 ... 優先”

4. Managed Identity (Reader/Privileged と OBO)

何のための仕組みか（説明）

SRE Agent は「自身の managed identity」を持ち、管理対象の resource groups に対して、その identity を使って操作します。[4-1]

ポータル作成時に “Reader / 特権付き (Privileged) ” のアクセス許可レベルを選べます。[4-5]

スライド要点 (“Reader で始めて大丈夫か” の説明）

- Permission level が Reader の場合、必要に応じてユーザーに一時的な昇格を要求し、OBO フローで完了させます。[4-2]

- managed resource group に対して事前に割り当てられるロールとして、Log Analytics Reader / Azure Reader / Monitoring Reader が挙げられています。[4-3]
- 権限を絞るときは「特定権限を剥がす」ではなく「resource group をスコープから外す」方針で整理します。[4-4]

補足（説明）：

- エージェントは「ユーザーの同意」と「適切な RBAC 割り当て」が揃う場合にのみアクションを実行します。[4-6]
- レビュー モードは明示的同意、インシデント対応計画のコンテキストでは暗黙的同意として扱います。[4-7]

MI と OBO の流れ

```

sequenceDiagram
    autonumber
    participant User as 運用担当者
    participant Agent as Azure SRE Agent
    participant Entra as Entra ID (OBO)
    participant Azure as Azure Resource

    User->>Agent: write アクションを依頼
    Agent-->>User: 同意を要求
    User-->>Agent: 承認
    Agent->>Entra: 一時的な昇格を要求 ( OBO )
    Entra-->>Agent: 一時的な資格情報
    Agent->>Azure: マネージド ID / 委任資格情報で実行
    Azure-->>Agent: 結果
    Agent-->>User: 結果を報告
  
```

参考（第4章）

- [4-1] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-managed-identity> — "Azure SRE Agent has its own managed identity"
- [4-2] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-managed-identity> — "prompts the user for a temporary elevation using OBO flow"
- [4-3] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-managed-identity> — "preconfigured with the following role assignments"
- [4-4] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-managed-identity> — "You can't directly remove specific permissions ... remove the entire resource group from the agent's scope."
- [4-5] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-managed-identity#agent-permissions> — "ポータルからエージェントを作成するときに、次のいずれかのアクセス許可レベルを適用できます"
- [4-6] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-managed-identity#agent-actions> — "ユーザーの同意 ... 適切な RBAC 割り当てがある場合にのみアクションを実行"
- [4-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-managed-identity#agent-actions> — "レビュー モード ... 明示的な同意 ... インシデント対応計画のコンテキスト ... 暗黙的な同意"

5. Run modes (Consent と Credentials の分解)

何が起きているか (説明)

SRE Agent の write action では、(a) 実行計画に対する同意 (Consent) と、(b) 権限が足りない場合の一時的な資格情報アクセス (Credentials) を分けて扱います。[5-1]

既定はレビュー モードで、実行プランを生成して同意を待ってからアクションを実行します。[5-5]

Review mode は、実行計画を生成したうえで consent を待ってから実行します。[5-2]

Credentials の許可が必要になった場合は、OBO フローで一時的に資格情報を使い、作業完了後に revoke されます。[5-3]

Autonomous mode は "implicit consent" として扱われますが、無制限ではなく incident management plan のコンテキストに限定される点が重要です。[5-4]

また、自律モードを「どのコンテキストでも」有効化できるわけではありません。スコープを制限して安全な境界内で動かします。[5-6]

フロー図 (Review mode)

```
flowchart TD
    A["エージェントが実行計画を生成"] --> B{"同意しますか？"}
    B -->|いいえ| Z["停止（実行なし）"]
    B -->|はい| C["エージェントが実行を試行"]
    C --> D{"必要な資格情報がある？"}
    D -->|はい| E["計画を実行"]
    D -->|いいえ| F{"一時的な資格情報を付与（OBO）？"}
    F -->|いいえ| Z
    F -->|はい| E
    E --> G["終了"]
```

実行モードを「権限の2軸」で理解する (整理)

実行モードの本質は、write (変更) アクションに対して次の2点をどう扱うかです。[5-2][5-3]

- **Consent (同意)** : 生成された実行プランに基づいて "実行してよい"か
- **Credentials (資格情報)** : エージェントの権限が不足する場合に、ユーザーの資格情報へ一時的アクセス (OBO) を許可するか[5-3]

この2軸は、次の3要素と組み合わせて考えると混乱しにくいです。

- **ユーザーロール (SRE Agent 内の RBAC)** : そのユーザーがポータル/チャットで何ができるか[5-7]
- **エージェントのマネージド ID (MI) 権限レベル** : エージェントが自分の権限だけで何ができるか (Reader / Privileged) [5-8]
- **実行モード (Review / Autonomous)** : 同意の扱い (明示/暗黙) と、Autonomous の有効範囲 (incident plan の文脈のみ) [5-4][5-10]

補足（重要）：

- エージェントのスコープ/権限が優先される（特権の“バックドア”昇格を防ぐ）という境界があるため、「ユーザーが外側で強権限」でも、SRE Agent 経由でできるとは限りません。[5-9]

早見表（Read / Write × Review / Autonomous）

この章では「エージェントにアクセス許可があるか？」を **“MI (+管理対象スコープ)” でその操作ができるか**として扱います。[5-8][5-9]

1) Read-only（読み取り）

MIで実行できる？	実行モード	何が起きるか（要点）
はい	Review	MI の権限で読み取りを実行する[5-1]
いいえ	Review	必要なら OBO による一時アクセスを求める（ユーザーが拒否すると停止）[5-3]
はい	Autonomous	MI の権限で読み取りを実行する[5-1]
いいえ	Autonomous	必要なら OBO による一時アクセスを求める（ユーザーが拒否すると停止）[5-3]

ポイント:

- 読み取りは、Review/Autonomous の違い（=同意の扱い）が出にくい。差が出るのは主に write。[5-1]

2) Write（変更）

MIで実行できる？	実行モード	何が起きるか（要点）
はい	Review	実行プランへの同意を求め、同意があれば MI の権限で実行[5-2]
いいえ	Review	実行プランへの同意を求め、同意後に OBO による一時アクセスを求める（拒否で停止）[5-2][5-3]
はい	Autonomous	暗黙的同意のもとで MI の権限で実行（ただし incident plan の文脈に限定）[5-4][5-10]
いいえ	Autonomous	必要なら OBO による一時アクセスを求める（拒否で停止）[5-3][5-4]

シナリオ（ユーザー・ロール × MI権限 × 実行モード）

ここでは、代表的な組み合わせを「誰が（SRE Agent 内ロール）」「エージェントが（MI）」「どのモードで（Review/Autonomous）」で整理します。[5-7][5-8]

シナリオA: ユーザーロール=Reader、MI=Reader

- ねらい: 監視/状況把握、根本原因の仮説立て、レポート生成。
- Review: 診断 (read) は進められるが、write は同意・実行の操作を担う想定ではない。
- Autonomous: incident plan の文脈に限定されるため、Reader が自律実行の主体になる設計には向きません。[5-10]

結論:

- **"閲覧中心"**の運用で、実行は管理者へエスカレーションする前提が安全。

シナリオB: ユーザーロール=Standard User、MI=Reader ("まず安全に"の典型)

- Review:
 - 実行プランは生成できるが、write の最終実行は同意・権限・運用ルールの壁に当たりやすい。
 - MI=Reader なので、実行に不足があれば OBO (資格情報の一時アクセス) 要求が発生し得る。[5-3][5-8]
- Autonomous:
 - 同意が暗黙扱いでも、MI=Reader だと不足が出やすく、結局 OBO が必要になり "完全自律" になりにくい。[5-3][5-4]
 - そもそも incident plan の文脈でしか有効化できない。[5-10]

結論:

- Standard User は **診断→提案（実行計画）**まで。
- 実行は管理者へ引き継ぐ分業が設計しやすい。

シナリオC: ユーザーロール=Admin、MI=Reader ("人が承認し、必要時だけ昇格")

- Review:
 - 管理者が明示的に同意し、エージェントは実行を試みる。[5-2]
 - MI が不足する場合、OBO による一時アクセスを求め、承認されれば実行する（拒否で停止）。[5-3][5-8]
- Autonomous:
 - 同意は暗黙扱いになるが、MI が不足するケースでは OBO が必要 (=人の関与が残る)。[5-3][5-4]

結論:

- **Review と Autonomous の差は「同意が省略されるか」。**
- MI=Reader では、権限不足がある限り OBO が残りやすい。

シナリオD: ユーザーロール=Admin、MI=Privileged ("自動化を進めたい")

- Review:
 - 管理者が同意し、MI の権限で実行できる範囲が増える (OBO を減らしやすい)。[5-2][5-8]
- Autonomous:
 - incident plan の限定スコープで、暗黙的同意のもとで実行できる範囲が増える。[5-4][5-10]

結論:

- 自律性を上げるには、**MI を Privileged にするだけでなく、管理対象スコープ設計（対象RG/許可する操作）**が重要。[5-8][5-9]

シナリオE: “Adminでも失敗する” 代表例（境界の確認）

- 管理者ロールは「SRE Agent 内の操作権」を付与するが、Azure 側（MIの権限/スコープ）で許されなければ実行できない。[5-9]
- 例: ロールバック/スケール等を試みても、MI のスコープ外、または必要 RBAC が割り当てられないなければ失敗する。

使い分け（運用設計の結論）

- Review（既定）**: “提案→人が確認→実行” の安全な導線。教育/監査/誤実行防止に向く。[5-2]
- Autonomous**: incident plan の文脈に限定して「境界内で自律」を狙う。範囲設計（スコープ/許可操作）とセットで考える。[5-10]
- MI=Reader**: 原則 read 中心。write は OBO に寄りやすいので、責任分界（誰が承認/資格情報を出すか）を決める。[5-8]
- MI=Privileged**: OBO 依存を減らしやすいが、スコープ/境界の設計が前提になる。[5-8][5-9]

参考（第5章）

- [5-1] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “Consent / Credentials”
- [5-2] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “generates an execution plan and waits for your consent”
- [5-3] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “Any access to user credentials are revoked once the action is complete.”
- [5-4] <https://learn.microsoft.com/en-us/azure/sre-agent/agent-run-modes> — “implicit consent”
- [5-5] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-run-modes#review-vs-autonomous-mode> — “既定では … レビュー モード … 実行プランを生成し、同意を待ってからアクションを実行”
- [5-6] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-run-modes#review-vs-autonomous-mode> — “どのコンテキストでも自律的に作業できるのではなく … 制限付きスコープ”
- [5-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/roles-permissions-overview> — “3 つの主要 ロール … 管理者 … 標準ユーザー … 閲覧者”
- [5-8] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-managed-identity> — “Reader … 必要なアクションに昇格されたアクセス許可が必要な場合 … OBO フロー … / 特権付き … 承認されたアクションを実行”
- [5-9] <https://learn.microsoft.com/ja-jp/azure/sre-agent/roles-permissions-overview> — “エージェントのアクセス許可 … 優先 … 特権エスカレーションを防ぐ”
- [5-10] <https://learn.microsoft.com/ja-jp/azure/sre-agent/agent-run-modes> — “インシデント管理計画のコンテキストでのみ自律モードを有効にできます”

6. Scheduled tasks

何の機能か（説明）

Scheduled tasks は、 monitoring / maintenance / security checks といったワークフローをスケジュール実行する機能です。[6-1]

作成方法（説明）

作成は UI から行えます。チャット中に依頼したり、incident response の一部として自動生成させることもできます。[6-2]

スケジュール（自然言語）から cron への変換を支援する “Draft the cron for me” と、指示文を改善する “Polish instructions” を使えます。[6-3][6-4]

作り方と実行パス

```
flowchart TD
    A["Scheduled task"] --> B{"作成方法は？"}
    B --> UI["UI から手動作成"]
    B --> Chat["チャット中に依頼"]
    B --> IRP["インシデント対応の一部として自動生成"]

    UI --> Exec["スケジュール実行"]
    Chat --> Exec
    IRP --> Exec

    Exec --> Out["Monitoring / maintenance / security checks"]
```

参考（第6章）

- [6-1] <https://learn.microsoft.com/en-us/azure/sre-agent/scheduled-tasks> — “automate workflows such as monitoring, maintenance, and security checks”
- [6-2] <https://learn.microsoft.com/en-us/azure/sre-agent/scheduled-tasks> — “create these tasks manually, request them during a chat ... allow the agent to generate them autonomously as part of incident response”
- [6-3] <https://learn.microsoft.com/en-us/azure/sre-agent/scheduled-tasks> — “Draft the cron for me”
- [6-4] <https://learn.microsoft.com/en-us/azure/sre-agent/scheduled-tasks> — “Polish instructions”

7. Incident management / Incident response plan

何の機能か（説明）

Incident management は、 Azure Monitor alerts / PagerDuty / ServiceNow などの incident management platform からアラートを受け取り、分析・対応する仕組みです。[7-1]

Azure Monitor は既定の統合で、最小限のセットアップで使えます。一方、PagerDuty / ServiceNow などは追加のセットアップが必要です。[7-6]

補足（説明）：

- Incident management の設定は、Azure portal でエージェントを開いて **[インシデントプラットフォーム]** タブから入ります。[7-7]
- 既定では Azure Monitor アラートがインシデント管理プラットフォームとして構成されます。[7-8]

Incident response plan の位置づけ（説明）

Incident response plan は、インシデントに対して「どう検知し、どう確認し、どう軽減するか」を定義する計画です。

カスタム計画では、(1) 対象インシデントのフィルター、(2) 自律性レベル（実行モード）、(3) 問題解決のためのカスタム命令を使って、エージェントの対応を調整します。[7-15]

補足（説明）：

- エージェントの動作は、インシデント対応計画の構成によって異なります（昇格して人手解決に回す/ユーザーに代わって解決する等）。[7-15]

スライド要点（Incident response plan の定義）

- 目的: インシデント対応のやり方を「計画」として固定化する。[7-15]
- 調整レバー: `filters / execution mode / custom instructions`。[7-15]
- 効果: “検知→分析→人手介入 or 自律是正” の分岐が設計できる。[7-15]

既定値（説明）

Incident management を有効化した場合の既定として、Azure Monitor alerts と接続し、low priority を全サービス対象にして review mode で処理します。[7-3]

補足（説明）：

- サポートされているインシデント管理プラットフォームとして PagerDuty / ServiceNow が挙げられています。[7-16]
- Azure Monitor をインシデント管理システムとして使う機能は試験段階で、まだ完全には機能していません。[7-17]

スライド要点（既定の応答計画）

- 既定の接続先: Azure Monitor alerts。[7-3]
- 既定の対象: 影響を受ける全サービスの低優先度。[7-3]
- 既定の実行: Review mode。[7-3]

テスト（説明）

Incident response plan は過去インシデントに対して test mode で実行できます。test mode は read-only です。[7-4]

補足（説明）：

- 期待どおりに動作するかを、既存インシデントに対して実行して確認します。[7-18]
- 新規/既存どちらのインシデント対応計画でもテストできます。[7-18]

スライド要点（テストモード）

- 過去インシデントで事前検証できる（安全に挙動確認）。[7-18]
- テストモードは常に read-only。[7-4]

応答計画をカスタマイズする（説明）

インシデント対応計画は、(1) 管理サービスの選択、(2) フィルター適用、(3) 自律レベルの設定、(4) プロンプト コンテキスト（カスタム命令）のカスタマイズを通じて作成できます。[7-19]

フィルター（filters）（説明）

プランを実行するインシデントを、次のフィルターで決められます。[7-20]

- インシデントの種類（例：既定 / メジャー / セキュリティ）[7-20]
- 影響を受けたサービス（すべて/名前で選択）[7-20]
- Priority（優先度）[7-20]
- タイトルに含まれるもの（タイトル文字列の一致）[7-20]

自律レベル（execution mode）（説明）

応答計画内で、エージェントの自律性レベルを選べます。[7-21]

- Review: 診断し、提案されたアクションのレビューと承認の後にのみ軽減/変更します。[7-21]
- Autonomous: 分析し、軽減策またはリソース変更を個別に実行します。必要なアクセス許可がない場合は、昇格されたアクセス許可への一時的なアクセスを付与するように求められます。[7-21]

カスタム命令（custom instructions）とツール（説明）

SRE Agent は過去インシデントを調べて、過去の解決方法に基づくプロファイルと、より詳細なコンテキスト（命令）を生成できます。[7-22]

生成された命令は編集してカスタマイズでき、独自の命令に置き換えることもできます。[7-23]

また、応答計画の内容に応じて、インシデント対応時に使用するツールの一覧が生成されます。ツールは追加/削除でき、カスタム命令を変更した場合はツール一覧を再生成して同期を維持できます。[7-24]

図（パワポ用・下書き）

図1: 「応答計画」の構成要素（1枚で説明）

```
flowchart LR
    Plan["Incident response plan\n(インシデント対応計画)"]
    F["Filters\n対象インシデントの絞り込み"] --> Plan
    M["Execution mode\nReview / Autonomous"] --> Plan
    I["Custom instructions\n解決方針・手順の文脈"] --> Plan
    T["Tools list\n使うツールの選択"] --> Plan
    Plan --> Out["実行時のエージェント挙動\n(分析 / 推奨 / 是正 / 更新)"]
```

```

classDef core fill:#111827,stroke:#111827,color:#ffffff;
classDef box fill:#F3F4F6,stroke:#9CA3AF,color:#111827;
class Plan core;
class F,M,I,T,Out box;

```

図2: 実行時の流れ（構成で分岐が変わる）

```

flowchart LR
    subgraph P["Incident management platform"]
        Alert["Alert / Incident"]
        end

        subgraph A["Azure SRE Agent"]
            Recv["Incident management\nが受信"]
            Thread["初期分析つきの\nチャット スレッド"]
            Mode{"Execution mode"}
            Rec["推奨を提示"]
            Rem["是正を実行\n(状況により自動終了)"]
            end

        subgraph H["運用担当"]
            Approve["レビュー/承認"]
            end

        subgraph P2["Incident management platform"]
            Update["インシデント更新/終了"]
            end

        Alert --> Recv --> Thread --> Mode
        Mode -->|Review| Rec --> Approve --> Update
        Mode -->|Autonomous| Rem --> Update

        classDef lane fill:#ffffff,stroke:#D1D5DB,color:#111827;
        classDef action fill:#EEF2FF,stroke:#6366F1,color:#111827;
        classDef decision fill:#FEF3C7,stroke:#F59E0B,color:#111827;
        class P,A,H,P2 lane;
        class Recv,Thread,Rec,Rem,Update action;
        class Mode decision;

```

エージェントの応答（説明）

- インシデントが検出されると、最初の分析を含む新しいスレッドがチャット履歴に作成されます。 [7-9]
- 閲覧者モードでは推奨を提示し、人間の介入が必要です。自律モードでは構成に応じてインシデントを自動終了したり是正措置を実行でき、管理プラットフォーム側のインシデント更新/終了も行う場合があります。 [7-10]
- 管理プラットフォーム側の構成により、SRE Agent に送るインシデントの種類（例: 低優先度は送るが高優先度は人間が対応）を制御できます。 [7-11]

- インシデントハンドラーをカスタマイズすることで、(例) 自律性レベル、使えるツール、カスタム手順などを制御できます。[7-12]

ダッシュボード（説明）

Incident management タブには、エージェントが管理するインシデントの一元ビュー（主要メトリック、保留中のインシデントなど）を提供するダッシュボードがあります。[7-13]

集計された視覚化と AI によって生成された根本原因分析を提供し、傾向把握や対応計画の最適化に使えます。[7-14]

インシデント処理の流れ

```

flowchart LR
    A["プラットフォーム側のアラート\\nAzure Monitor / PagerDuty / ServiceNow"] -->
    B["Incident management が受信"]
    B --> C["初期分析つきの\\n新しいチャット スレッドを作成"]
    C --> D{"実行モード"}
    D -->|Review| V["推奨を提示\\n人が介入"]
    D -->|Autonomous| AU["是正を実行\\nインシデントを自動終了する場合あり"]
    V --> E["運用担当が承認/対応"]
    AU --> F["プラットフォーム側の\\nインシデントを更新/終了"]
    E --> F

```

PagerDuty 連携の注意（説明）

PagerDuty 統合では User API key が必要です。General API key では acknowledge ができません。[7-5]

参考（第7章）

- [7-1] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-management> — “receives alerts from ... Azure Monitor alerts ... PagerDuty ... ServiceNow”
- [7-2] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-response-plan> — “Filters ... Execution mode ... Customize instructions”
- [7-3] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-response-plan> — “Processes all low priority incidents ... Runs in review mode”
- [7-4] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-response-plan> — “In test mode ... always operates in a read-only mode.”
- [7-5] <https://learn.microsoft.com/en-us/azure/sre-agent/incident-management> — “You must use a User API key ... General API keys don't allow the agent to acknowledge incidents properly”
- [7-6] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#platform-integration> — “Azure Monitor (既定の統合) には最小限のセットアップ ... PagerDuty や ServiceNow ... 追加のセットアップ”
- [7-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#platform-integration> — “エージェントを開き、[インシデント プラットフォーム] タブ”
- [7-8] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#platform-integration> — “既定では、Azure Monitor アラートは ... 構成されます”

- [7-9] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#agent-responses> — “インシデントが検出されると ... 新しいスレッドがチャット履歴に作成”
- [7-10] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#agent-responses> — “閲覧者モード ... 自律モード ... 自動的に終了 ... 是正措置 ... インシデントを更新または終了”
- [7-11] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#agent-responses> — “構成設定を制御することで ... インシデントの種類を制御”
- [7-12] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#agent-responses> — “インシデントハンドラーをカスタマイズ ... 自律性レベル ... 使用できるツール ... カスタム手順”
- [7-13] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#dashboard> — “ダッシュボード ... すべてのインシデントの一元的なビュー ... 主要なメトリック ... 保留中のインシデント”
- [7-14] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-management?tabs=azmon-alerts#dashboard> — “集計された視覚化と AI によって生成された根本原因分析”
- [7-15] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan> — “インシデントを検出、確認、軽減する方法を定義 ... 自律レベルを設定 ... カスタム指示”
- [7-16] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#default-settings> — “サポートされているインシデント管理プラットフォーム ... PagerDuty ... ServiceNow”
- [7-17] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#default-settings> — “Azure Monitor ... 試験段階 ... まだ完全には機能していません”
- [7-18] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#test-a-response-plan> — “過去の問題に対して計画をテスト ... テストモード ... 常に読み取り専用 ... 新規および既存 ... テスト”
- [7-19] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#customize-a-response-plan> — “管理サービスの選択 ... フィルター ... 自律レベル ... プロンプトコンテキストのカスタマイズ”
- [7-20] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#filter-incidents> — “インシデントの種類 ... 影響を受けたサービス ... Priority ... タイトルに含まれるもの”
- [7-21] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#set-an-autonomy-level> — “Review ... レビューと承認 ... Autonomous ... 必要なアクセス許可がない場合 ... 一時的なアクセス”
- [7-22] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#define-custom-instructions> — “以前のインシデント ... 過去の解決方法 ... プロファイル ... より詳細なコンテキストを生成”
- [7-23] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#define-custom-instructions> — “生成された命令をカスタマイズ ... 置き換え”
- [7-24] <https://learn.microsoft.com/ja-jp/azure/sre-agent/incident-response-plan?tabs=new-plan#define-custom-instructions> — “ツールの一覧を生成 ... 追加または削除 ... ツールの一覧を再生成して同期”

8. Memory system

何の機能か（説明）

Memory system は、トラブルシューティングを効果的に行うために、runbooks・チーム標準・サービス固有のコンテキストを与える仕組みです。[8-1]

構成（説明）

Memory system のコンポーネントは User Memories / Knowledge Base / Documentation connector / Session insights の4つです。[8-2]

補足（説明）：

- Session insights はセッションからエージェントが生成するメモリ（自動）です。[8-7]

（補足：4コンポーネントの役割の切り分け）

コンポーネント	目的	設定のしやすさ	最適な用途
User memories	チーム知識のクイック保存（チャットコマンド）	インスタント	チーム標準、サービス構成、運用ワークフローのパターン
Knowledge Base	Runbook 等の直接アップロード	クイック（ファイルアップロード）	静的 runbook、トラブルシューティングガイド、内部ドキュメント
Documentation connector	Azure DevOps の自動同期	構成が必要	頻繁に更新されるドキュメント（ライブドキュメント）
Session insights	セッションから自動生成される学習	自動	トラブルシューティングパターン、過去の解決から学習

この表の理解は「何を、どこに置くべきか」を決めるための前提になります。[8-14]

取り出し方（説明）

SearchMemory は4コンポーネントを横断検索するツールです。Custom subagents ではツール追加が必要です。[8-3]

SearchNodes も同様に全メモリシステムを検索しますが、追加で **entityType** などのフィルターオプションをサポートし、**includeNeighbors** を **true** にすると検索結果に加えて接続されたノード（リソースや関連インシデント等）も返します。[8-8]

拡張検索パラメーター（事実）：

パラメーター	役割
entityType	Incident / Service / Resource など、エンティティ種別で結果をフィルターする
includeNeighbors	true の場合、一致したノードだけでなく接続されたノードも返す

`includeNeighbors` により「一致ノード + つながっているリソース/サービス/関連インシデント/リンク文書」まで含めたリレーションシップの把握が可能になります。[8-15]

データ投入の注意（説明）

秘密情報（secrets/credentials/API keys）は保存しません。[8-4]

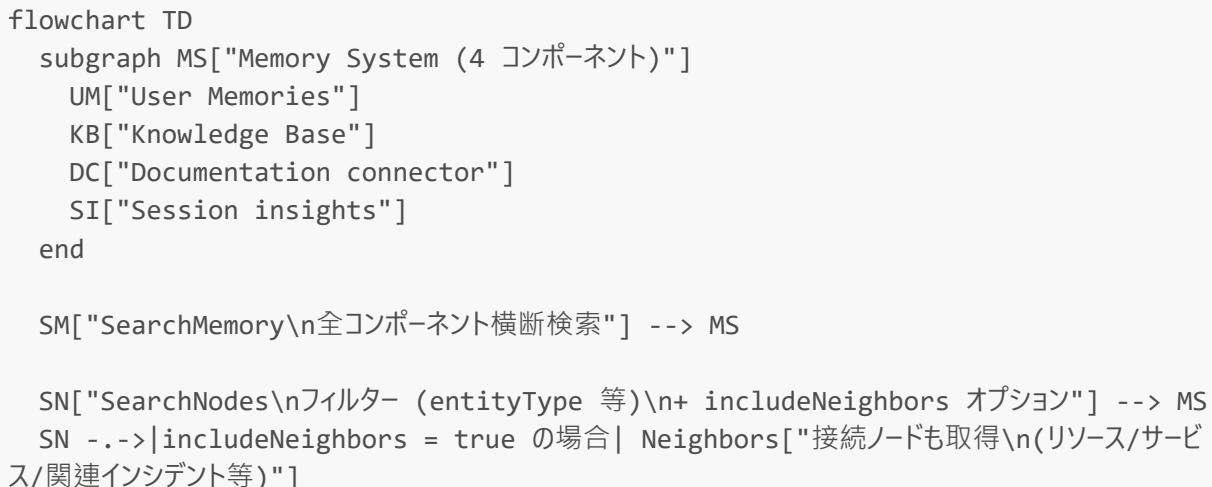
メモリはチームで共有され、検索用にインデックスが作成されます。[8-9]

Knowledge Base の仕様（説明）

Knowledge Base は `.md` と `.txt` を扱い、1ファイル最大 16MB です。[8-5]

さらに、1回のアップロード内で合計 100MB の制限があります。[8-10]

メモリ構成と検索



User memories の操作（説明）

User memories は `#remember` / `#forget` / `#retrieve` のチャットコマンドを使います。[8-6]

`#remember` は将来の会話のために fact/standard/context を保存し、`#forget` は保存済みメモリを検索して削除し、`#retrieve` はエージェントの推論をトリガーせずに検索・表示します。[8-11]

補足（仕組み: 事実）：

- `#remember` で保存した内容は、（Learnの記述では）OpenAI を用いた埋め込み（embedding）として格納され、Azure AI Search に保存されます。保存後に Agent Memory saved. の確認が表示されます。[8-16]
- `#forget` は意味的検索で一致候補を見つけて削除し、 Agent Memory forgotten: [deleted content] の確認が表示されます。[8-17]
- `#retrieve` は保存メモリを意味的に検索し、上位5件をもとに「合成された回答」と「個々のメモリ」の両方を表示します（推論をトリガーせずに確認したい用途）。[8-18]

User Memories の実用例（提案）

前提:

- メモリに secrets/credentials/API keys を保存しません（運用手順で徹底）。[8-4]
- チームで共有され検索用にインデックス化されるので、「個人メモ」ではなく「運用の構成情報」として扱います。[8-9]

実用例 1: “最初の一言”を短くする（環境コンテキスト）

- ねらい: 毎回の前提説明（prod/stg、主要リソース、既知の監視ポイント）を省略し、診断にすぐ入る。
- 例（保存）：

```
#remember
In our environment:
- Production: <subscription alias>, <resource-group>
- Primary workload: <service-name> (App Service / Container Apps)
- Logs: <Log Analytics workspace alias>
- APM: Application Insights enabled
```

実用例 2: Runbook と“判断基準”を固定する（チーム標準）

- ねらい: “何を確認したら次に何をするか”を、記憶（標準手順）として持たせる。
- 例（保存）：

```
#remember
Runbook standard:
1) Confirm symptom, time window (UTC), and impact
2) Collect evidence: errors/exceptions + dependency latency + resource metrics
3) Produce a short incident summary (What/When/Where/Impact/Suspected cause/Next actions)
```

実用例 3: “検索だけ”を安全に使う（確認・棚卸し）

- ねらい: エージェントの推論を走らせず、保存済みメモリを確認する。
- 例（取得）：

```
#retrieve runbook
```

実用例 4: 更新時の事故を減らす（削除）

- ねらい: 交代/変更で古くなった運用前提を残さない。
- 例（削除）：

```
#forget on-call rotation
```

スライド要点:

- User Memories は「チーム標準の短文化」に向く（毎回の前提説明を削れる）。[8-6][8-11]
- “共有され検索される” 前提なので、運用のコンフィグとして扱う（秘密情報は入れない）。[8-4] [8-9]

Session insights (説明)

セッション分析情報は、各セッションから「症状、解決手順、根本原因、落とし穴」をキャプチャして検索可能なメモリになり、関連する過去の分析情報が今後のセッションで自動的に取得されます。[8-12]

分析情報は「会話完了後に定期的に（約30分ごと）自動生成」されるか、「チャット フッターで[セッション分析情報の生成]を選ぶと約30秒でオンデマンド生成」されます。[8-13]

補足（エージェントがキャプチャする内容: 事実）：

- 観察された症状（同様パターン認識）
- 動作した手順（実証済みの解決パス提案）
- 根本原因（より可能性の高い原因へのショートカット）
- 落とし穴（同じ間違いの回避）
- 指定したコンテキスト（環境事実の記憶）
- 関連するリソース（同一リソースの過去問題と接続）[8-19]

補足（「見に行く」価値: 事実）：

セッション分析情報を定期的にレビューすることで、繰り返し発生する問題、知識のギャップ、runbook の不足、テレメトリの欠如といったパターンを発見できます。これらのパターンに基づいて、コードや構成の修正、サブエージェントの作成、runbook の更新、ログ・メトリックの追加、アラートの調整といった具体的な改善につなげることができます。[8-20]

補足（インサイトの構造: 事実）：

各分析情報は「タイムライン（最大8つのマイルストーン）」「エージェントのパフォーマンス（うまくいったこと/改善領域/主要な学習）」「調査品質スコア（1~5）」を含みます。[8-21]

Quickstart (Learn の推奨手順: 事実)

Learn では、段階的に導入する流れが提示されています。[8-22]

1. User memories から始める（#remember でチーム知識を即時保存）[8-22]
2. 主要ドキュメントを Knowledge Base にアップロードする（.md / .txt、最大 16MB / ファイル）。アップロード後、システムがインデックス化し、SearchMemory で取得できるようになります。[8-5][8-23]
3. Session insights を確認し、何がうまくいったか/不足していたコンテキストを見て「知識ギャップ」を埋める（必要ならメモリやドキュメントを追加）。[8-24]
4. （任意）Documentation connector を使い、Azure DevOps リポジトリのドキュメントを自動同期する（設定 > Connectors から、リポジトリ URL と managed identity を指定してインデックスを開始）。[8-25]

Knowledge Base (詳細: 事実)

ポータルでのアップロードは **設定 > Knowledge Base** から行い、ファイルの検証/アップロード/インデックス作成が自動で行われます。[8-23]

さらにエージェントは **UploadKnowledgeDocument** ツールで、Knowledge Base へ直接アップロードできます（調査中に得た手順の保存、インシデント解決からの runbook 追加、UIなしでの追加）。[8-26]

UploadKnowledgeDocument の仕様（事実）：

- **fileName: .md / .txt** 必須
- **content**: プレーンテキストまたは Markdown の全文
- **triggerIndexing**: 省略時 **true** (バッチでは **false** にできる) [8-26]

制約・挙動（事実）：

- 最大ファイルサイズ: 16MB
- 同名ファイルが存在する場合は上書き
- (Learnの記載では) Azure Blob Storage にアップロードし、検索可能なインデックス作成をトライガーする[8-26]

Azure Portal で「どこで」「どう実装するか」（事実）

ここでは、Memory system を Azure Portal の UI で“実装（運用に載せる）”する際の操作パスをまとめます。

前提: まず Azure portal で **Azure SRE Agent** を検索して開き、一覧から対象のエージェントを選択します（チャット UI が開けば OK）。[8-27]

1. User memories (チーム知識の即時保存)

- 場所: エージェントの **チャット**
- やること: **#remember** / **#forget** / **#retrieve** をそのまま入力して実行します。[8-6][8-11]

2. Knowledge Base (runbook のアップロード)

- 場所: エージェントの **Settings > Knowledge base**
- やること: **Add file** を選ぶか、ドラッグ&ドロップで **.md / .txt** をアップロードします（最大 16MB / ファイル）。アップロード後、ポータル側で検証/アップロード/インデックス作成が実行され、**SearchMemory** で検索可能になります。[8-5][8-23]

3. Session insights (改善ループの確認)

- 場所: エージェントの **Settings > Session insights**
- 生成: 会話終了後に定期的（約30分）に自動生成、またはチャットフッターの **Generate Session insights** (日本語 UI では「セッション分析情報の生成」) でオンデマンド生成（約30秒）できます。[8-13]

4. Documentation connector (Azure DevOps リポジトリの同期)

- 場所: エージェントの **Settings > Connectors**

- やること: **Add connector** から **Documentation connector** を選び、Azure DevOps の repository URL と managed identity を指定します。コネクタが自動でインデックスを開始します。[8-25]

補足（実装の考え方）：

- まず **#remember** で「運用の前提」を短く保存し、次に Knowledge Base に“手順化された runbook”を入れ、最後に Session insights で「不足していた前提」を埋めていく、という順が Learn の Quickstart と整合します。[8-22][8-24]

参考（第8章）

- [8-1] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “gives agents the knowledge they need to troubleshoot effectively”
- [8-2] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “consists of four complementary components”
- [8-3] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “SearchMemory tool retrieves all memory components ... Custom subagents: Add SearchMemory”
- [8-4] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “Don't store secrets, credentials, API keys”
- [8-5] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “Upload .md or .txt files (up to 16 MB each).”
- [8-6] <https://learn.microsoft.com/en-us/azure/sre-agent/memory-system> — “chat commands (#remember, #forget, #retrieve)”
- [8-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#memory-components> — “セッションの分析情報 ... セッションからエージェントによって生成されたメモリ ... 自動”
- [8-8] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#advanced-search-parameters> — “SearchNodes ... フィルターオプション ... includeNeighbors を true ... 接続されているノードも返されます”
- [8-9] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#tool-configuration> — “チームはメモリを共有し、システムによって検索用にインデックスが作成されます”
- [8-10] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#supported-file-types-and-limits> — “要求ごと ... 合計 100 MB”
- [8-11] [#remember ... #forget ... #retrieve”](https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#chat-commands)
- [8-12] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#session-insights> — “症状、解決手順、根本原因、および落とし穴をキャプチャ ... 検索可能なメモリ ... 自動的に取得”
- [8-13] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#when-insights-are-generated> — “約 30 分ごと ... [セッション分析情報の生成] ... (約 30 秒)”
- [8-14] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#memory-components> — “4 つの補完的なコンポーネント ... 目的 ... 設定 ... 最適な用途”
- [8-15] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#advanced-search-parameters> — “entityType ... includeNeighbors ... 接続されたノードも返す”
- [8-16] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#remember-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E6%83%85%E5%A0%B1%E3%82%92%E4%BF%9D%E5%AD%98%E3%81%99%E3%82%8B> — “OpenAI ... 埋め込み ... Azure AI Search ... Agent Memory saved.”

- [8-17] [https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#forget-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%83%A1%E3%83%A2%E3%83%AA%E3%82%92%92%E5%89%8A%E9%99%A4%E3%81%99%E3%82%8B](https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#forget-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%83%A1%E3%83%A2%E3%83%AA%E3%82%92%E5%89%8A%E9%99%A4%E3%81%99%E3%82%8B) — “意味的な分析 ... Agent Memory forgotten”
 - [8-18] [https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#retrieve-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%83%A1%E3%83%A2%E3%83%AA%E3%81%AB%82%AF%E3%82%A8%83%AA%E3%82%92%92%E5%AE%9F%E8%A1%8C%E3%81%99%82%8B](https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#retrieve-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E3%83%A1%E3%83%A2%E3%83%AA%E3%81%AB%E3%82%AF%E3%82%A8%E3%83%AA%E3%82%92%E5%AE%9F%E8%A1%8C%E3%81%99%E3%82%8B) — “上位 5 つの一致 ... 個々のメモリ ... 合成された回答”
 - [8-19] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#what-the-agent-captures> — “観察された症状 ... 動作した手順 ... 根本原因 ... 落とし穴 ... コンテキスト ... 関連するリソース”
 - [8-20] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#find-opportunities> — “繰り返し発生 ... 知識不足 ... Runbook ... テレメトリ ... アラート”
 - [8-21] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#insight-structure> — “タイムライン ... エージェントのパフォーマンス ... 調査品質スコア ... 1 から 5”
 - [8-22] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#quickstart> — “ユーザー モリ ... ドキュメント ... 自動同期に拡張”
 - [8-23] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#uploading-documents> — “設定 > ナレッジベースmd または .txt ... インデックス ... SearchMemory”
 - [8-24] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#3-review-session-insights> — “設定 > Session の分析情報 ... 知識のギャップ ... メモリまたはドキュメントを追加”
 - [8-25] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#4-connect-a-repository-optional> — “設定 > Connectors ... ドキュメント コネクタ ... Azure DevOps ... マネージド ID”
 - [8-26] <https://learn.microsoft.com/ja-jp/azure/sre-agent/memory-system#uploading-documents-with-agent-tools> — “UploadKnowledgeDocument ... fileName ... content ... triggerIndexing ... 上書き ... Blob Storage ... インデックス”
 - [8-27] <https://learn.microsoft.com/en-us/azure/sre-agent/usage#chat-with-your-agent> — “In the Azure portal, search for and select Azure SRE Agent ... Locate your agent ... select it”
-

9. Subagent builder / Tools / Connectors / Custom MCP

Subagent builder (説明)

Subagent builder は、運用業務を自動化・効率化するために、専門的なサブエージェントを作成・カスタマイズ・管理する機能です。[9-1]

サブエージェント ビルダーでできること：

- カスタム サブエージェントの作成（特定の業務に特化したエージェント）
- データ統合（監視ツールやナレッジソースへの接続）
- 自動トリガー設定（インシデント対応計画やスケジュールされたタスクによる起動）
- アクション連携（外部サービスとの連携）[9-4]

サブエージェントの設計手順：

1. 目的と運用スコープを定義
2. データソースを接続
3. システムツールや MCP 統合を関連付け

4. カスタム命令を設定
5. ハンドオフルールを構成[9-5]

補足：サブエージェントは、インシデント対応計画またはスケジュールされたタスクによってトリガー（起動）されるか、メインエージェントからのハンドオフによって呼び出されます。[9-6]

Python カスタム ロジック（説明）

サブエージェント ビルダー上で Python ツールを作成できます。作成手順は [**Builder > Subagent builder**] → **作成 > ツール** → **Python ツール** です。[9-7]

Python ツールは「タイムアウト 5~900秒（既定 120）」「`def main` を含む必要がある」「戻り値は JSON にシリアル化（dict/list/プリミティブ/None→null）」などの仕様です。[9-8][9-9]

さらに、ツールモード（自動/手動/隠れた）でエージェントがツールを呼ぶ方法を制御できます。Azure リソースへアクセスする必要がある場合は [**ID**] タブでマネージド ID アクセスを構成します。[9-10][9-11]

Connectors（説明）

コネクタは SRE Agent の機能を拡張する統合です。通信コネクタ（Outlook/Teams 等）とナレッジ/テレメトリの取り込み（Datadog/Dynatrace/New Relic 等）に加えて、任意の MCP エンドポイントへ接続するカスタム コネクタがあります。[9-12]

コネクタ設定は [**設定**] > [**コネクタ**] から種類（Outlook/Teams/カスタム MCP）を選びます。Outlook/Teams は OAuth、MCP は URL と資格情報または OAuth トークンを指定します。[9-13]

Custom MCP（説明）

Custom MCP server は HTTPS で到達可能なリモートホストが必須です。SRE Agent 内でローカル実行はできません。[9-2]

さらに、MCP のツールは main agent から直接は使えず、サブエージェント経由でのみアクセスできます。[9-3]

補足（説明）：

- カスタム MCP 接続では、コネクタが「MCP サーバー エンドポイント」「トランSPORT（SSE/HTTP）」「認証メカニズム」を定義します。[9-14]
- 追加は [**設定**] → [**コネクタ**] → [**コネクタの追加**] → **種類: MCP サーバー** を選び、名前/接続の種類（SSE/HTTP）/MCP サーバー URL/認証などを入力して検証します。[9-15]

オンプレミス機器に SRE Agent を使いたい場合（設計パターン）

結論としては「オンプレを Azure の管理プレーンに“載せる”」か「オンプレ/外部SaaSにあるデータを MCP で“引く”」の2択に分解すると整理しやすいです。

パターンA: オンプレを Azure リソース化して“スコープに入れる”

SRE Agent は、エージェントに関連付けたリソース グループ内のリソースへアクセスできます。[9-22]

そのため、オンプレの Windows/Linux (サーバー等) を **Azure Arc-enabled servers** として Azure リソース化し、(リソース グループに配置したうえで) そのリソース グループを SRE Agent の managed resource groups に含める、という設計が成立します。[9-23]

補足 (監視データの取り込み: 事実) :

Azure Arc の Connected Machine agent は「オンプレを Azure リソースとして扱えるようにするエージェント」ですが、監視テレメトリ収集 (Log Analytics 等) を置き換えるものではありません。オンプレ側で OS/ワークロードの監視やログ収集をするには別途 Azure Monitor Agent が必要になります。[9-23]

さらに Azure Monitor Agent は、オンプレ/他クラウドのマシンでも Azure Arc を介してサポートされ、ワークスペースへの認証に managed identity を使用します (Connected Machine agent の導入で作成される)。[9-24]

このパターンが向くケース:

- ・ 「オンプレも Azure portal 上で資産として扱いたい」 (タグ/ポリシー/一元管理の文脈)
- ・ “診断の一次データ”を Azure Monitor / Log Analytics に寄せたい

パターンB: MCP でオンプレ/外部SaaSのデータを“引く”

オンプレ機器の状態が Datadog など外部の観測基盤に集約されている場合は、SRE Agent の **Custom MCP server connector** を使い、外部システムに対する“問い合わせ手段”をサブエージェントに提供するのが整理しやすいです。[9-12][9-15]

注意点 (事実) :

- ・ MCP サーバーは **HTTPS** で到達可能なリモートホストが必須で、SRE Agent の中でローカル実行はできません。[9-2]
- ・ MCP ツールは **メインエージェントから直接は使えず**、サブエージェント経由でのみアクセスできます (= Datadog MCP を使うなら、Datadog問い合わせ専用サブエージェントを作るのが基本線)。[9-3]

このパターンが向くケース:

- ・ 「オンプレの操作/問い合わせAPIが社内にある」 → そのAPIを包む MCP を用意する
- ・ 「既に Datadog MCP 等がある」 → それをコネクタとしてつなぐ

「データをためる場所」についての整理 (運用上の注意)

SRE Agent の Memory system (User memories / Knowledge Base / Session insights) は、runbook やチーム標準など“手順と前提”を置く場所で、メトリクス/ログなどの“生のテレメトリ”を溜めるストアではありません (テレメトリは Azure Monitor/Log Analytics や Datadog 等に置く)。また、Memory system には秘密情報を保存しない前提です。[8-4][8-14]

Main agent / Subagent / Tools の関係

サブエージェントは、インシデント対応計画・スケジュールタスク・メインエージェントからのハンドオフの3つの方法で呼び出され、ツール・スキル・コネクタ・ナレッジの4つのカテゴリの機能を使用できます。

```

flowchart TD
    subgraph Triggers ["サブエージェントの呼び出し方法"]
        IRP["Incident Response Plan\n(インシデント対応計画)"]
        Schedule["Scheduled Tasks\n(スケジュールされたタスク)"]
        Main["メイン エージェント\n(Chat Handoff)"]
    end

    IRP --> Sub["サブエージェント"]
    Schedule --> Sub
    Main -->|ハンドオフ| Sub

    subgraph SubAgentCapabilities ["サブエージェントが使用できるもの"]
        direction TB

        subgraph Tools ["ツール"]
            Python["Python ツール\n(カスタムロジック)"]
            Kusto["Kusto クエリ\n(Log Analytics/App Insights)"]
            Metrics["Azure Metrics"]
            DevOps["DevOps ツール\n(GitHub/Azure DevOps)"]
            OtherTools["その他のシステムツール"]
        end

        Skills["スキル"]

        subgraph Connectors ["コネクタ (外部統合 )"]
            Outlook["Outlook\n(メール送信)"]
            Teams["Teams\n(通知送信)"]
            Telemetry["テレメトリソース\n(Datadog/Dynatrace/New Relic)"]
            MCP["カスタム MCP\n(他の SaaS システム)"]
            OtherConn["その他"]
        end

        subgraph Knowledge ["ナレッジ (Memory System) "]
            UserMem["User Memories"]
            KB["Knowledge Base"]
            DocConn["Documentation Connector"]
            SessionIns["Session Insights"]
        end
    end

    Sub --> Tools
    Sub --> Skills
    Sub --> Connectors
    Sub --> Knowledge

    MCP -.->|HTTPS/SSE/HTTP| External["外部 MCP サーバー"]

```

classDef triggerStyle fill:#e1f5ff,stroke:#0078d4,stroke-width:2px
 classDef mainStyle fill:#fff4ce,stroke:#f59700,stroke-width:3px
 classDef subStyle fill:#d4f1d4,stroke:#107c10,stroke-width:2px
 classDef capabilityStyle fill:#f3f2f1,stroke:#605e5c,stroke-width:1px

```
class IRP,Schedule,Main triggerStyle  
class Sub subStyle  
class Tools,Skills,Connectors,Knowledge capabilityStyle
```

Tool が多い前提での整理（提案）

添付画像のように、ポータル上の「ツール」はカテゴリ単位（例: DevOps / Diagnostics / Knowledge Base ...）で大量に並ぶ前提で設計します。

整理のしかた（例）：

- Diagnostics: Kusto (Log Analytics / App Insights) やメトリック取得、クエリ検証など
- DevOps: GitHub Issue / コメント、Azure DevOps Work Item、リポジトリ紐付けなど
- Knowledge Base: Memory/Knowledge の検索・投入
- Connectors: Outlook/Teams、外部監視/ナレッジ、カスタム MCP
- Custom logic: Python ツール（用途特化の“薄い関数”）[9-8][9-9]

スライド要点:

- “ツールが多い”のは正常（能力を足していくと増える）。設計で重要なのは「どれをいつ使うか」を主語にすること。
- main agent から直接使えないツールがある（MCP は subagent 経由）。[9-3]

Agent Units (AAU) の使用量を API で取得する (PowerShell)

ポータル（Network タブ）から確認できるとおり、Agent Units (AAU) の使用量は ARM 経由で取得できます。

エンドポイント（例）：

- 月次の合計: GET
`/subscriptions/<subId>/resourceGroups/<rg>/providers/Microsoft.App/agents/<agentName>/usages?api-version=2025-05-01-preview`
- 日次の内訳: GET
`/subscriptions/<subId>/resourceGroups/<rg>/providers/Microsoft.App/agents/<agentName>/dailyusages?api-version=2025-05-01-preview`

注意:

- Bearer token (`Authorization: Bearer ...`) は絶対に共有しません。
- `api-version` が `*-preview` のため、フィールド名や挙動が変わる可能性があります。

直接 GET (おすすめ)

```
$subscriptionId = "2107faa2-8e88-4c5d-8124-34b8c638de70"  
$resourceGroup = "rg-dev-rag"  
$agentName     = "rag-are"  
$apiVersion    = "2025-05-01-preview"
```

```

# ARM 用アクセストークン (Azure CLI)
$token = az account get-access-token --resource https://management.azure.com/ -
-query accessToken -o tsv
$headers = @{
    Authorization = "Bearer $token"
}

$baseUri = "https://management.azure.com"
$monthlyUri =
"$baseUri/subscriptions/$subscriptionId/resourceGroups/$resourceGroup/providers/
/Microsoft.App/agents/$agentName/usages?api-version=$apiVersion"
$dailyUri =
"$baseUri/subscriptions/$subscriptionId/resourceGroups/$resourceGroup/providers/
/Microsoft.App/agents/$agentName/dailyusages?api-version=$apiVersion"

$monthly = Invoke-RestMethod -Method GET -Uri $monthlyUri -Headers $headers
$daily = Invoke-RestMethod -Method GET -Uri $dailyUri -Headers $headers

$monthly | ConvertTo-Json -Depth 50
$daily | ConvertTo-Json -Depth 50

```

ARM の /batch でまとめて取得（ポータルと同じやり方）

```

$subscriptionId = "2107faa2-8e88-4c5d-8124-34b8c638de70"
$resourceGroup = "rg-dev-rag"
$agentName = "rag-are"
$apiVersion = "2025-05-01-preview"

$token = az account get-access-token --resource https://management.azure.com/ -
-query accessToken -o tsv
$headers = @{
    Authorization = "Bearer $token";
    "Content-Type" =
    "application/json"
}

$batchUri = "https://management.azure.com/batch?api-version=2015-11-01"

$body = @{
    requests = @(
        @{
            httpMethod = "GET"
            requestHeaderDetails = @{
                commandName = "getMonthlyUsage"
            }
            url =
            "/subscriptions/$subscriptionId/resourceGroups/$resourceGroup/providers/Microso
            ft.App/agents/$agentName/usages?api-version=$apiVersion"
        }
        @{
            httpMethod = "GET"
            requestHeaderDetails = @{
                commandName = "getDailyUsages"
            }
            url =
            "/subscriptions/$subscriptionId/resourceGroups/$resourceGroup/providers/Microso
            ft.App/agents/$agentName/dailyusages?api-version=$apiVersion"
        }
    )
}

```

```

} | ConvertTo-Json -Depth 10

$resp = Invoke-RestMethod -Method POST -Uri $batchUri -Headers $headers -Body
$body
$resp | ConvertTo-Json -Depth 50

```

date と value だけ抜き出す

以下のレスポンス（例）：

```
{
    "date": "2026-02-23",
    "name": { "value": "AgentUnits", "localizedValue": "Agent Units" },
    "value": 81.85395874999999
}
```

直接 GET の \$daily から (name.value == AgentUnits で絞り込み)：

```
$daily.value |
Where-Object { $_.name.value -eq "AgentUnits" } |
Select-Object -Property date, value
```

/batch の \$resp から（リクエスト順で [0]=monthly, [1]=daily の想定）：

```
$dailyFromBatch = $resp.responses[1].content

$dailyFromBatch.value |
Where-Object { $_.name.value -eq "AgentUnits" } |
Select-Object -Property date, value
```

日次コスト（USD）に変換する

前提（重要）：

- dailyusages の name.value == "AgentUnits" が **その日の合計AAU (Always-on + Active)** を返すのか、**Active分だけ**なのかは環境/APIの返し方で変わり得ます (*-preview)。
- 迷ったら、まず「dailyusages の合計」と「usages の月次合計」が概ね一致するかで当たりを付けます。

次のスクリプトは両方の解釈で日次コストを計算します（AAU 単価は East US 2 の値を入れてください）。

```

# 入力: `date` と `value` を持つ配列 (例: $daily2 = $daily.value | ... | Select-Object date,value)
$daily2 = $daily.value |
Where-Object { $_.name.value -eq "AgentUnits" } |
Select-Object -Property date, value

# 設定 (要調整)
$aauUnitPriceUsd = 0.10    # East US 2 の AAU 単価 (USD/AAU) に置き換え
$alwaysOnAgentCount = 1     # Always-on を有効にしているエージェント数
$alwaysOnHoursPerDay = 24   # 24固定でOK (途中でOFFにした日は実態に合わせる)

$alwaysOnAauPerDay = 4 * $alwaysOnAgentCount * $alwaysOnHoursPerDay

$dailyCost = $daily2 |
Sort-Object date |
ForEach-Object {
    $activeAau = [double]$_.value
    $activeSeconds = $activeAau / 0.25

    # 解釈A: dailyのAgentUnitsが「その日の合計AAU」
    $totalAau_assumeDailyIsTotal = $activeAau
    $usd_assumeDailyIsTotal = $totalAau_assumeDailyIsTotal * $aauUnitPriceUsd

    # 解釈B: dailyのAgentUnitsが「Active分だけ」
    $totalAau_assumeDailyIsActiveOnly = $alwaysOnAauPerDay + $activeAau
    $usd_assumeDailyIsActiveOnly = $totalAau_assumeDailyIsActiveOnly *
$aauUnitPriceUsd

    [pscustomobject]@{
        date = $_.date
        activeAau = [math]::Round($activeAau, 2)
        activeSeconds_est = [math]::Round($activeSeconds, 0)
        alwaysOnAau_assume = $alwaysOnAauPerDay
        usd_assumeDailyIsTotal = [math]::Round($usd_assumeDailyIsTotal, 2)
        usd_assumeDailyIsActiveOnly = [math]::Round($usd_assumeDailyIsActiveOnly,
2)
    }
}

$dailyCost | Format-Table -AutoSize

# 月次合計 (解釈A/B それぞれ)
$dailyCost |
Measure-Object -Property usd_assumeDailyIsTotal, usd_assumeDailyIsActiveOnly
-Sum |
Select-Object -ExpandProperty Sum

```

Azure Workbooks に組み込んで可視化する (ARM + JSONPath)

「dailyusages を取得 → AgentUnits だけ表にする → 日次コスト列を追加 → チャート化」までを Workbooks の中だけで完結できます。

ポイント:

- 「Log Analytics workspace を選べ」と出る場合は **Logs** データソースを選んでいます。ここでは **Azure Resource Manager (Preview)** を使います。

1) パラメータを用意する

Workbooks を **Edit** モードにして、次のパラメータを追加します。

- Subscription** (Subscription picker)
- ResourceGroup** (Text)
- AgentName** (Text)
- ApiVersion** (Text) : **2025-05-01-preview**
- AauUnitPriceUsd** (Text または Number) : AAU 単価 (USD/AAU)
- AlwaysOnAgentCount** (Text または Number) : Always-on を有効にしているエージェント数
- AlwaysOnHoursPerDay** (Text または Number) : **24**

2) ARM で dailyusages を取得し、JSONPath で表にする

Add query でクエリを追加し、次の設定にします。

- Data source:** Azure Resource Manager (Preview)
- Http Method:** GET
- Path:**
 - /subscriptions/{Subscription:id}/resourceGroups/{ResourceGroup}/providers/Microsoft.App/agents/{AgentName}/dailyusages
- Parameters:**
 - api-version: {ApiVersion}

次に Result Settings で Result Format を JSON Path にします。

- JSON Path Table** (どちらか)
 - 推奨 (AgentUnits のみに絞る) : **\$.value[?(@.name.value=='AgentUnits')]**
 - うまく絞れない場合: **\$.value.***

Columns (例) :

Column ID	Column JSON Path
date	\$.date
metric	\$.name.value
aau	\$.value

metric 列を出しておくと、AgentUnits 以外が混ざっていないか確認しやすいです。

任意: **Advanced settings** でこのクエリの **名前**を **Daily AAU (raw)** のように付けておきます（次の Merge で参照しやすくするため）。

3) Merge で計算列（日次コスト）を追加する

もう1つ **Add query** を追加して、次の設定にします。

- **Data source:** **Merge**
- **Add Merge**
 - **Merge Type:** **Duplicate table**
 - **Table:** さきほど名前を付けたテーブル（例: **Daily AAU (raw)**）

Run **Merge** を押したあと、ツールバーの **Add new item** で「他列の値から計算する列」を追加します。

計算したい列（例）：

- **activeAau = aau**
- **activeSeconds_est = activeAau / 0.25**
- **alwaysOnAau_assume = 4 * {AlwaysOnAgentCount} * {AlwaysOnHoursPerDay}**
- **usd_assumeDailyIsTotal = activeAau * {AauUnitPriceUsd}**
- **usd_assumeDailyIsActiveOnly = (activeAau + alwaysOnAau_assume) * {AauUnitPriceUsd}**

補足:

- **{...}** は Workbooks のパラメータ参照です。UI 上で数値として扱われるよう、パラメータ型は **Number** が選べるなら **Number** 推奨です。

4) 可視化（テーブル + 時系列チャート）

- テーブル: **date, activeAau, usd_assumeDailyIsTotal, usd_assumeDailyIsActiveOnly** を表示
- チャート: **Visualization** を **Time chart** にして、X 軸を **date**、Y 軸を **usd_assumeDailyIsTotal**（または **usd_assumeDailyIsActiveOnly**）に設定

5) 検算（任意）

迷ったら、別クエリで **usages**（月次合計）も取得しておき、
dailyusages(AgentUnits) の合計と概ね一致するかで解釈A/Bの当たりを付けます。

- **Path:**
`/subscriptions/{Subscription:id}/resourceGroups/{ResourceGroup}/providers/Microsoft.App/agents/{AgentName}/usages`
- **Parameters:** `api-version={ApiVersion}`

SubAgent の設計方法（他 MS AI ドキュメントの原則を援用）

SRE Agent の Subagent 仕様は製品ドキュメント中心ですが、**プロンプト (system/instructions) 設計の作法**は一般化できます。

要点（設計チェックリスト）：

- 役割と成果物を最初に書く (assistant の job) 。[9-16]
- 境界条件 (やらないこと) を明示する。[9-16]
- 出力フォーマット (例: Markdown の章立て、JSON のキー固定) を固定する。[9-16]
- 不確実なときの方針 (質問する/わからないと言う) を明示する。[9-16]
- 指示は具体的に、解釈の余地を減らす (Be Specific など) 。[9-17]

ツール呼び出しを安定させるコツ (設計) :

- ツールは「説明 (description) 」「引数の説明」を厚くし、システム指示で“いつ呼ぶか”を明示する。[9-18]
- 重要な確認が欠けている場合は、推測せず質問させる (例: 対象リソース/時間範囲) 。[9-18]

複数エージェントに分けるときの注意 (一般論) :

- 役割を分割して、主エージェントは “委譲と統合” に徹する (責務分離) 。[9-19]
- 深さや引用の伝播など、マルチエージェント固有の制約がある (設計で吸収する) 。[9-20]

シナリオ例 : HTTP 500 のインシデントを「証跡収集→Issue化」まで自動処理する

目的 :

- インシデントを診断し、**証跡 (KQL/メトリクス/例外) **を揃え、開発者が着手できる GitHub Issue を作成する
- 本番変更や自動緩和は行わない (診断 + 報告に限定)

エージェント構成 :

```
flowchart TD
  Incident["インシデント発生\n(Azure Monitor Alert)"] --> Main["Main Agent\n(インシデント受信・統括)"]
  Main -->|証跡収集が必要| Diag["DiagnosticsAgent\n(ログ/メトリクス収集)"]
  Diag -->|証跡収集完了| Main
  Main -->|Issue 作成が必要| GH["GitHubIssueAgent\n(Issue 作成)"]
  GH -->|Issue 作成完了| Main
  Main --> Done["完了報告"]
```

1) Main Agent の設定

ポータルでの設定項目 (Main エージェントは既定で存在) :

設定項目	内容
Instructions	インシデントを受信し、状況を整理して適切なサブエージェントにハンドオフする統括役
Tools	- SearchMemory (Memory systemへのアクセス) - 基本的な監視ツール (必要に応じて)

設定項目	内容
Handoff subagents	- DiagnosticsAgent - GitHubIssueAgent
Instructions の例:	

役割: HTTP 500 エラーのインシデント統括担当

あなたの責任:

1. Azure Monitor からインシデントアラートを受信する
2. 重要な情報を抽出する: サービス名、時間帯 (UTC) 、環境 (prod/stg)
3. インシデントの種類に基づいて、どのサブエージェントを呼び出すかを判断する
4. ユーザーに結果を要約して報告する

ハンドオフの判断ロジック:

- ログ/メトリクスの証跡が必要な場合 → DiagnosticsAgent にハンドオフ
- GitHub Issue の作成が必要な場合 → GitHubIssueAgent にハンドオフ

境界条件:

- 本番環境への変更は実行しない
- メモリに秘密情報を保存しない

サービス名や時間帯が不明な場合は、1つの明確な質問をする。

Handoff instructions の例:

このエージェントにハンドオフすべき場合:

HTTP 500エラーなどのアプリケーションエラーに関するインシデントの統括が必要な場合。Incident Response Plan、ユーザーの直接要求、または他のエージェントから、証跡収集やIssue作成の調整が必要な場合にハンドオフしてください。

入力:

インシデント情報 (アラート内容、影響範囲、検出時刻など)

出力:

完了報告 (収集された証跡、作成されたIssue URL、取られたアクション)

2) DiagnosticsAgent (Subagent) の設定

ポータルでの作成: [Builder > Subagent builder] → [Create > Subagent]

設定項目	内容
Subagent name	diagnostics-agent
Instructions	証跡収集専門エージェント

設定項目	内容
Handoff instructions	Main から受け取る情報と、Main に返す情報の形式を定義
Tools	<ul style="list-style-type: none"> - Log Query (Log Analytics) - App Insights Query - Azure Metrics - SearchMemory (必要に応じて)

Give access to knowledge base ON (runbook へのアクセス)

Instructions の例:

役割: インシデントの証跡収集担当

あなたの仕事:

1. Main Agent からインシデントコンテキストを受け取る (サービス、時間帯、環境)
2. Log Analytics と App Insights で関連するログ/例外を検索する
3. 問題を示す 3~5 個の重要なメトリクスを収集する
4. 発見事項を構造化された形式で要約する

使用するツール:

- Log Analytics: アプリケーションログ、例外、依存関係の確認用
- App Insights: リクエストレーメトリ、失敗の確認用
- Azure Metrics: リソースレベルのメトリクス (CPU、メモリ、リクエスト数) 用

出力形式 (Main Agent に返す) :

証跡サマリー

- **サービス**: <名前>
- **時間帯 (UTC)**: <開始> - <終了>
- **影響**: <エラー数 / 影響を受けたユーザー数>

主要な発見事項

1. <発見事項1 と KQL クエリ>
2. <発見事項2 と KQL クエリ>
3. <発見事項3 とメトリクス>

疑わしい原因

<短い仮説>

制約:

- 読み取り専用の操作のみ
- 重要な情報が欠けている場合は、Main Agent に確認を求める

Handoff instructions の例:

ハンドオフすべき場合:

Azure Monitor (Log Analytics、App Insights、Metrics) からインシデント証跡の収集が必要な場合。ログ/例外のクエリ、主要メトリクス (3~5個) の収集、KQLと疑わしい原因を含む構造化サマリーを作成します。読み取り専用の診断データ収集に限定。

ハンドオフすべきでない場合:

是正措置、構成変更、RBAC変更、Azure Monitor以外のデータソース、ビジネスインパクト計算、postmortem作成は対象外です。

入出力:

受取: サービス名、時間帯 (UTC) 、環境

返却: 証跡サマリー (発見事項、KQLクエリ、メトリクス、疑わしい原因)

ハンドバック先: Main Agent

3) GitHubIssueAgent (Subagent) の設定

ポータルでの作成: [Builder > Subagent builder] → [Create > Subagent]

設定項目	内容
Subagent name	github-issue-agent
Instructions	GitHub Issue 作成専門エージェント
Handoff instructions	Main から受け取る証跡をもとに Issue を作成
Tools	- GitHub (Issue 作成、コメント追加) - Teams (通知用、オプション)
Give access to knowledge base	ON (Issue テンプレートへのアクセス)

Instructions の例:

役割: インシデント用の GitHub Issue 作成担当

あなたの仕事:

1. Main Agent から証跡サマリーを受け取る
2. 以下を含む、適切に構造化された GitHub Issue を作成する:
 - 明確なタイトル
 - 影響の説明
 - タイムライン (UTC)
 - 証跡 (KQL クエリ、メトリクス)
 - 疑わしい原因
 - 開発者への提案アクション
3. Issue URL で作成を確認する

使用するツール:

- GitHub: 指定されたリポジトリに Issue を作成
- (オプション) Teams: Issue が作成されたときに通知を送信

Issue テンプレート:

```

---  

## 影響  

<ユーザーに対する影響を記述>  

## タイムライン (UTC)  

- 開始: <時刻>  

- 検出: <時刻>  

## 証跡  

<KQL クエリと主要メトリクスを説明と共に記載>  

## 疑わしい原因  

<証拠に基づく仮説>  

## 次のアクション  

- [ ] <アクション1>  

- [ ] <アクション2>  

---  

制約:  

- 常に UTC タイムスタンプを含める  

- 実際の KQL クエリを含める（説明だけではなく）  

- 利用可能な場合は関連ダッシュボードへのリンクを含める

```

Handoff instructions の例:

Main Agent から証跡がハンドオフされたとき:

- 受け取る情報: 証跡サマリー（発見事項、KQL クエリ、メトリクス、疑わしい原因）
- あなたのタスク: テンプレートに従って GitHub Issue を作成する
- 完了時: Issue URL を Main Agent に返す
- ハンドバック先: Main Agent

実行の流れ (Incident Response Plan で自動化) :

1. Azure Monitor Alert が発火 → Incident management が受信
2. Main Agent が起動され、インシデント情報を受け取る
3. Main Agent が DiagnosticsAgent に証跡収集をハンドオフ
4. DiagnosticsAgent が証跡を収集し、Main Agent に返す
5. Main Agent が GitHubIssueAgent に Issue 作成をハンドオフ
6. GitHubIssueAgent が Issue を作成し、Main Agent に Issue URL を返す
7. Main Agent がユーザーに完了報告（チャットスレッド）

設計のポイント :

- Main エージェントは「統括・ハンドオフ判断」に徹する（ツールは最小限）
- 各サブエージェントは「専門ツール」を持ち、明確な役割を持つ
- Instructions でハンドオフロジックを明示する
- Handoff instructions で入出力の契約を明確にする

- 実際の運用では、Incident Response Plan の設定で Main Agent を起動トリガーとして設定する

参考（第9章）

- [9-1] <https://learn.microsoft.com/en-us/azure/sre-agent/subagent-builder-overview> — “Incident response plans or scheduled tasks trigger subagents.”
- [9-2] <https://learn.microsoft.com/en-us/azure/sre-agent/custom-mcp-server> — “must host ... remotely and make them reachable over HTTPS ... doesn't support running MCP servers locally”
- [9-3] <https://learn.microsoft.com/en-us/azure/sre-agent/custom-mcp-server> — “only accessible through subagents and aren't directly accessible to main Azure SRE Agent.”
- [9-4] <https://learn.microsoft.com/ja-jp/azure/sre-agent/subagent-builder-overview#what-you-can-build-with-subagent-builder> — “カスタム サブエージェント ... データ統合 ... 自動トリガー ... アクション”
- [9-5] <https://learn.microsoft.com/ja-jp/azure/sre-agent/subagent-builder-overview#work-with-subagent-builder> — “主な目的と運用スコープ ... データ ソース ... システム ツールと MCP 統合 ... カスタム命令 ... ハンドオフルール”
- [9-6] <https://learn.microsoft.com/ja-jp/azure/sre-agent/subagent-builder-overview#work-with-subagent-builder> — “インシデント対応計画またはスケジュールされたタスクは、サブエージェントをトリガー”
- [9-7] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#create-a-python-tool> — “左側のナビゲーションから [Builder>Subagent builder] ... 作成>ツール ... Python ツール”
- [9-8] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#create-a-python-tool> — “タイムアウト (秒) | 5 ~ 900 ... 既定値は 120 ... Python コード ... def main 関数を含む必要があります”
- [9-9] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#write-the-main-function> — “システムは戻り値を JSON に自動的にシリアル化 ... 次の種類がサポート”
- [9-10] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#configure-tool-mode> — “自動 ... 手動 ... 隠れた”
- [9-11] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-logic-python#set-up-identity> — “Azure リソースにアクセスする必要があるツール ... [ID] タブ ... マネージド ID アクセスを構成”
- [9-12] <https://learn.microsoft.com/ja-jp/azure/sre-agent/connectors#what-are-connectors> — “通信コネクタ ... ナレッジコネクタ ... カスタムコネクタ ... MCP サーバー エンドポイント”
- [9-13] <https://learn.microsoft.com/ja-jp/azure/sre-agent/connectors#configure-a-connector> — “設定 ... コネクタ ... Outlook と Teams ... OAuth ... MCP URL と資格情報または OAuth トークン”
- [9-14] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-mcp-server#how-custom-mcp-connections-work> — “コネクタは次を定義 ... エンドポイント ... トランスポート ... 認証”
- [9-15] <https://learn.microsoft.com/ja-jp/azure/sre-agent/custom-mcp-server#add-a-custom-mcp-server-connector> — “Azure portal ... [設定] → [コネクタ] ... [コネクタの追加] ... MCP サーバー ... SSE ... HTTP ... URL ... 認証”
- [9-16] <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/concepts/advanced-prompt-engineering?view=foundry-classic#design-checklist> — “Start with the assistant's job ... Define boundaries ... Specify the output format ... Add a “when unsure” policy”
- [9-17] <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/concepts/prompt-engineering?view=foundry-classic#best-practices> — “Be Specific ... Order Matters ... Give the model an “out””
- [9-18] <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/function-calling?view=foundry-classic#prompt-engineering-with-functions> — “Provide more context in the system”

message ... Ask for clarification if a user request is ambiguous."

- [9-19] <https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/connected-agents?view=foundry-classic> — "break down complex tasks into coordinated, specialized roles"
- [9-20] <https://learn.microsoft.com/en-us/azure/ai-foundry/agents/how-to/connected-agents?view=foundry-classic#limitations> — "maximum depth of 2 ... not possible to guarantee citations ..."
- [9-21] <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ai-agents/build-secure-process#agent-instructions> — "Standardize instruction architecture ... Identity and tone ... Scope and boundaries ... Tool mandates ... Citation rules"
- [9-22] <https://learn.microsoft.com/en-us/azure/sre-agent/usage#chat-with-your-agent> — "Your agent has access to any resource inside the resource group that's associated with the agent."
- [9-23] <https://learn.microsoft.com/en-us/azure/azure-arc/servers/overview> — "machines outside of Azure ... treated as a resource in Azure ... included in a resource group ... Connected Machine agent ... doesn't replace ... Azure Monitor Agent"
- [9-24] <https://learn.microsoft.com/en-us/azure/azure-monitor/agents/azure-monitor-agent-supported-operating-systems#on-premises-and-in-other-clouds> — "supported on ... on-premises ... via Azure Arc-enabled servers ... authenticates ... using a managed identity ... created when you install the Connected Machine agent"

10. デモ:

注意: ここは「何をどう壊す/どう直す」を構成した **デモ案（提案）**です。

ただし、各ステップの“事実として言い切る部分”は Microsoft Learn の記述に基づき、参照番号で根拠を示します。

デモA: 「App Service が遅い/タイムアウトする」 = Cosmos DB の 429 (スロットリング) / ホットパーティションを切り分ける

ゴール（提案）

- "アプリ (App Service) の症状"から入り、根因が"バックエンド (Cosmos DB) のスループット/パートイション"であることを、Portal のメトリクスで確認していく。
- その後、Azure SRE Agent (Cosmos DB SRE Agent) に「診断の起点」を作らせ、運用の会話の型を作る。[10A-1]

全体像



事前準備（Azure portal での設定 / 事実+提案を分離）

A) Cosmos DB (NoSQL) を Azure portal で作る (事実)

1. Azure portal で **Azure Cosmos DB** を検索し、**Create** から **Azure Cosmos DB for NoSQL** を選ぶ。[10A-2]
2. **Review + create** → **Create** → **Go to resource** でアカウントを作成する。[10A-3]

B) Data Explorer で DB/Container を作る (事実)

1. Cosmos アカウントのメニューで **Data Explorer** を開く。[10A-4]
2. **New Container** を選び、DB と container を作成する (partition key とスループット (Manual/Autoscale) を設定する)。[10A-5][10A-6]

C) App Service を Azure portal で作り、アプリをデプロイする (事実+提案)

1. Azure portal で **App Services** → **+ Create > Web App** で App Service を作成する。[10A-7]
2. デモ用アプリは、(提案) Cosmos DB を読む/書くエンドポイント (例: `/healthz` と `/burst`) を持つように用意する。
3. Portal の案内に沿って進めるなら、App Service の **Deployment Center** から Git リポジトリを設定してデプロイする (例: **Source: External Git** → **Repository/Branch** を指定 → **Save**)。[10A-8]
4. デプロイ後は App Service の **Overview** で **Browse** を使い、アプリが起動することを確認する (起動に少し時間がかかることがある)。[10A-18]

D) App Service へ Cosmos 接続情報を注入する (事実)

1. App Service の左メニューで **Settings > Environment variables > Connection strings** を開く。[10A-9]
2. 新規追加は **Add**、反映は **Apply** (ダイアログ側) → **Apply** (Environment variables 画面側) で行う。[10A-19]
3. Cosmos DB の接続文字列を connection string として登録でき、App Service は環境変数として提供する (Cosmos DB は `DOCDBCONNSTR_` プレフィックスとして定義されている)。[10A-10][10A-11]
4. Portal 上では connection string の値は既定で非表示で、必要に応じて **Show value / Show values** で表示できる。[10A-20]

障害の作り方 (提案)

- “429 を出す”には、(提案) 低い RU/s の container に対して短時間に集中アクセスを発生させる。
- “ホットパーティション”を意図的に作るなら、(提案) 特定の partition key 値に偏ったリクエストを投げ続ける。

切り分け (Azure portal の手順 / 事実)

Step 1: 429 が出ているかを確認する (事実)

- 429 は “Request rate too large” で、Cosmos DB が rate limited を返していることを示す。[10A-12]
- まず Cosmos DB アカウントの **Insights > Requests > Total Requests by Status Code** で、429 の比率を確認する。[10A-13]

Step 2: 正常な範囲か、問題かを判断する (事実)

- 1~5% 程度の 429 は “healthy sign” として言及されている（エンドツーエンド遅延が許容なら追加対応不要）。[10A-14]
- 429 が高率かつ継続する場合、次のステップへ進む。

Step 3: ホットパーティションかどうかを見る (事実)

- ホットパーティションがあると、一部の partition key range が 100% に張り付き、429 が発生し得る。[10A-15]
- **Insights > Throughput > Normalized RU Consumption (%) By PartitionKeyRangeID** で偏りを確認する（高い PartitionKeyRangeID があればホットパーティション候補）。[10A-16]

Step 4: 何を直すべきかの分岐 (事実)

```

flowchart TD
S["症状:\nApp Service が遅い/タイムアウト"] --> A["Cosmos の 429 比率を確認"]
A --> C{"429 が低く、レインジも OK ? "}
C -->|はい| OK["追加対応なし"]
C -->|いいえ| B["PartitionKeyRangeID ごとの\nNormalized RU を確認"]
B --> D{"偏り/ホットパーティション ? "}
D -->|はい| HP["ホットパーティション疑い"]
D -->|いいえ| TH["全体スループット不足の可能性"]
HP --> FIX1["パーティション戦略を見直す\n( 中長期 )"]
TH --> FIX2["スループット (RU/s) 増\nまたは 1 操作あたり RU 削減"]
FIX2 --> META["メタデータの\nスロットリングも除外"]
META --> SYS["Insights > System\nMetadata Requests By Status Code"]

```

補足 (事実) :

- メタデータ操作 (DB/コンテナのCRUD、一覧、スループット確認など) は system-reserved RU limit があり、データ側の RU/s を増やしても効果がないケースがある (“increasing ... RU/s ... has no effect” と明記）。[10A-21]
- 調査導線として **Insights > System > Metadata Requests By Status Code** が提示されている。[10A-22]

SRE Agent で“会話の起点”を作る (事実+提案)

- Cosmos DB SRE Agent は「Azure SRE Agent を使って Cosmos DB のトラブルシューティングを簡素化する」AI-powered diagnostic tool と説明されている。[10A-1]
- セットアップは「SRE Agent を作成→Cosmos リソースを追加→Preview Upgrade Channel を有効化→会話開始」という流れで説明されている。[10A-17]
- (提案) チャットの型としては、**My App Service is slow. Check Cosmos throttling/hot partitions.** のように「症状→疑う観点」を明示して投げる。

参考 (デモA)

- [10A-1] <https://learn.microsoft.com/en-us/azure/cosmos-db/site-reliability-engineering-agent> — “AI-powered diagnostic tool ... simplify troubleshooting”

- [10A-2] <https://learn.microsoft.com/en-us/azure/cosmos-db/quickstart-portal#create-an-account> — “select **Create**, and then **Azure Cosmos DB for NoSQL**.”
- [10A-3] <https://learn.microsoft.com/en-us/azure/cosmos-db/quickstart-portal#create-an-account> — “select **Review + create** ... select **Create** ... select **Go to resource**”
- [10A-4] <https://learn.microsoft.com/en-us/azure/cosmos-db/quickstart-portal#create-a-database-and-container> — “select **Data Explorer**”
- [10A-5] <https://learn.microsoft.com/en-us/azure/cosmos-db/quickstart-portal#create-a-database-and-container> — “select the **New Container** option.”
- [10A-6] <https://learn.microsoft.com/en-us/azure/cosmos-db/how-to-create-container> — “Enter a **Partition key** ... Select **Autoscale** or **Manual** throughput”
- [10A-7] <https://learn.microsoft.com/en-us/azure/sre-agent/troubleshoot-azure-app-service> — “search for **App Services** ... select + **Create > Web App**.”
- [10A-8] <https://learn.microsoft.com/en-us/azure/sre-agent/troubleshoot-azure-app-service> — “select **Deployment Center** ... **Source** ... **Save**”
- [10A-9] <https://learn.microsoft.com/en-us/azure/app-service/configure-common#configure-connection-strings> — “select **Settings > Environment variables** ... select **Connection strings**.”
- [10A-10] <https://learn.microsoft.com/en-us/azure/app-service/configure-common#configure-connection-strings> — “Connection strings are always encrypted ... (encrypted at rest).”
- [10A-11] <https://learn.microsoft.com/en-us/azure/app-service/reference-app-settings#variable-prefixes> — “**DOCDBCONNSTR_** | Connection string to a database in Azure Cosmos DB.”
- [10A-12] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large> — “A “Request rate too large” ... indicates that your requests ... are being rate limited.”
- [10A-13] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large#request-rate-is-large> — “navigate to **Insights > Requests > Total Requests by Status Code.**”
- [10A-14] <https://learn.microsoft.com/en-us/azure/cosmos-db/monitor-normalized-request-units> — “if you see between 1-5% of requests with 429s ... this is a healthy sign ... No action is required.”
- [10A-15] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large#step-2-determine-if-theres-a-hot-partition> — “A hot partition ... can lead to 429 responses”
- [10A-16] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large#step-2-determine-if-theres-a-hot-partition> — “navigate to **Insights > Throughput > Normalized RU Consumption (%) By PartitionKeyRangeID**.”
- [10A-17] <https://learn.microsoft.com/en-us/azure/cosmos-db/site-reliability-engineering-agent> — “Create an Azure SRE Agent ... Add your Azure Cosmos DB resources ... Enable the Preview Upgrade Channel”
- [10A-18] <https://learn.microsoft.com/en-us/azure/sre-agent/troubleshoot-azure-app-service> — “select **Overview** ... select **Browse**”
- [10A-19] <https://learn.microsoft.com/en-us/azure/app-service/configure-common#configure-connection-strings> — “select **Add** ... select **Apply** ... select **Apply** on the **Environment variables** page.”
- [10A-20] <https://learn.microsoft.com/en-us/azure/app-service/configure-common#configure-connection-strings> — “values for connection strings are hidden ... select **Show value** ... **Show values**.”
- [10A-21] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large#rate-limiting-on-metadata-requests> — “There's a system-reserved RU limit ... increasing ...

RU/s ... has no effect"

- [10A-22] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-rate-too-large#rate-limiting-on-metadata-requests> — "Navigate to **Insights > System > Metadata Requests By Status Code.**"
-

デモB: 「App Service が散発的にタイムアウトする」 = Cosmos DB 408 (Request timeout) /ホットパーティション起点で切り分ける

狙い (提案) :

- アプリ側の "タイムアウト" を、Cosmos DB の 408 (Request timeout) として観測し、ホットパーティション/429 併発の可能性まで含めて切り分ける。

切り分けの軸 (事実) :

- 408 の原因の一つとして "Hot partition key" が挙げられており、判断に **Normalized RU Consumption** を使う導線が示されている。[10B-1]
- 408 が SLA 違反かどうかで「リトライで耐える」か「サポートへ」が分かれる。[10B-2]

参考 (デモB)

- [10B-1] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-time-out> — "Hot partition key ... Use the Normalized RU Consumption metric"
 - [10B-2] <https://learn.microsoft.com/en-us/azure/cosmos-db/troubleshoot-request-time-out> — "The application should handle this scenario and retry ... Contact Azure Support"
-

11. FAQ 由来のトラブルシュート早見表 (事実)

症状	Learn が挙げる原因	Learn が挙げる対処
portal が unresponsive	firewall が Azure domain をブロック	*.azuresre.ai を allowlist に追加
403/CORS、チャットできない	権限不足や割り当ての問題	Contributor/Owner を確認、グループ割り当てに頼らず直接付与、Check Access を使う

- *.azuresre.ai allowlist は portal unresponsive の対処として挙げられている。[11-1]
- 権限エラー対処として "Avoid relying solely on group-based role assignments" や "Check Access" が挙げられている。[11-2]

早見表の判断フロー

```
flowchart TD
S["開始: ポータル/チャットの問題"] --> A{"ポータルが応答しない?"}
A -->|はい| N["ネットワーク allowlist を確認"]
N --> F["*.azuresre.ai を allowlist に追加"]
A -->|いいえ| B{"403/CORS またはチャット不可?"}
```

```
B --> |はい| P[ "ロール割り当てを確認" ]
P --> Q[ "直接割り当てを優先\nCheck Access を使用" ]
B --> |いいえ| O[ "その他の原因を確認" ]
```

参考 (第11章)

- [11-1] <https://learn.microsoft.com/en-us/azure/sre-agent/faq> — “add *.azuresre.ai to the allow list”
- [11-2] <https://learn.microsoft.com/en-us/azure/sre-agent/faq> — “Avoid relying solely on group-based role assignments ... Use the Check Access feature”

12. 詳細な調査 (Deep investigation)

何のための機能か (説明)

詳細な調査は、SRE Agent で複雑な問題を診断するときに **透明性と精度を上げる**ための、仮説主導の調査モードです。[12-1]

標準的な調査（短いクエリで迅速に分析情報を返す）に対して、詳細な調査では「結論を急がず、複数の根本原因候補を検証してから軽減策を提案する」進め方を取ります。[12-1][12-2]

いつ使うか (スライド要点)

- 重大インシデントなど **影響が大きく複雑**な問題の調査。[12-1]
- 根本原因が複数疑われ、**体系的な検証**が必要なとき。[12-1]
- 戦略室（war room）的に、**推論プロセスを追跡**したいとき。[12-1]

どう進むか (スライド要点)

- 初期調査: ログ/メトリック/コンテキストを収集して土台を作る。[12-2]
- 仮説生成: 2~4 個の大まかな仮説（調査パス）を作る。[12-2]
- 検証: 各仮説を反復的にテストし、有効ならサブ仮説に分解して深掘りする。[12-2]
- 無効化も記録: 何を除外したかを残す（複雑なトラブルシュートで重要）。[12-2]
- 軽減策: 根本原因が確認できたら、実行可能な修復手順（ロールバック/リソース調整/設定修正など）を提案する。[12-2]
- 構造化出力: 検証済み/無効化の仮説を含めて、結果を明確に見せる。[12-2]

図 (仮説→検証のループ)

添付図の「仮説生成→テスト→（必要ならサブ仮説）」の考え方を、SRE Agent の詳細な調査の説明に合わせて書き起こした図。

```
flowchart TD
U["運用担当者"] --> Tg1["(任意) 詳細な調査を ON\nチャットの Deep investigation アイコン"]
Tg1 --> Q["調査クエリを送信"]
```

```

Q --> Init["初期調査\nログ/メトリック/コンテキスト収集"]
Init --> Gen["仮説生成\n2~4 個の大まかな仮説"]

subgraph Loop ["仮説検証ループ"]
    Pick["仮説を 1 つ選ぶ"] --> Test["テスト設計/実行\n反復チェック"]
    Test --> Eval{"仮説は支持される?"}
    Eval -->|"いいえ"| Inval["無効化として記録\n(除外理由を残す)"]
    Eval -->|"はい"| Sub["サブ仮説を生成\n寄与要因を深掘り"]
    Sub --> Test
end

Gen --> Pick
Inval --> More{"他の仮説を試す?"}
More -->|"はい"| Pick
More -->|"いいえ"| Mit["軽減策を提案\nロールバック/リソース調整/設定修正 など"]

Mit --> Out["構造化された出力\n検証済み/無効化の仮説 + 根拠 + 次アクション"]
Out --> End["調査終了"]

```

どう有効化するか（事実）

チャットで詳細な調査を使うには、クエリ送信前にチャットの **詳細調査（Deep investigation）アイコン**を選択します。[12-3]

参考（第12章）

- [12-1] <https://learn.microsoft.com/ja-jp/azure/sre-agent/deep-investigation> — “詳細な調査により…透明性と精度が向上”
- [12-2] <https://learn.microsoft.com/ja-jp/azure/sre-agent/deep-investigation> — “仮説主導のアプローチ…2~4…仮説…反復的なチェック…無効な仮説を文書化…軽減策…構造化された出力”
- [12-3] <https://learn.microsoft.com/ja-jp/azure/sre-agent/deep-investigation> — “クエリを送信する前に…詳細調査…アイコンを選択”

付録: Learn の公式チュートリアル（参照のみ）

- App Service を SRE Agent でトラブルシュートする手順がある。[A-1]
- Container Apps を SRE Agent でトラブルシュートする手順がある。[A-2]

参考（付録）

- [A-1] <https://learn.microsoft.com/en-us/azure/sre-agent/troubleshoot-azure-app-service> — “Troubleshoot an App Service app by using Azure SRE Agent Preview”
- [A-2] <https://learn.microsoft.com/en-us/azure/sre-agent/troubleshoot-azure-container-apps> — “Troubleshoot a container app by using Azure SRE Agent Preview”