AIエージェントが変えるZabbix運用の未来

SCSK株式会社
ITインフラサービス事業グループ
基盤ソリューション事業本部
テクノロジーサービス部　第四課

自己紹介

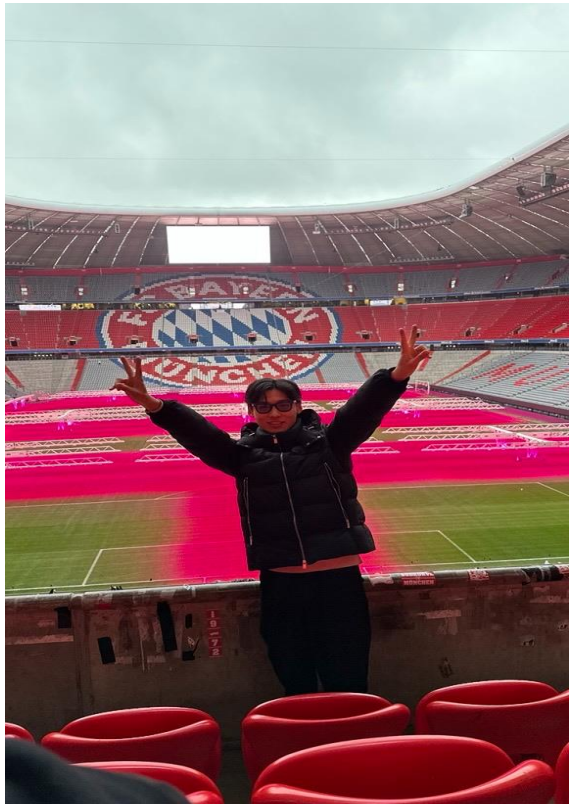
新沼　俊介 （ Niinuma Shunsuke ）

所属

SCSK株式会社
ITインフラサービス事業グループ
基盤ソリューション事業本部
テクノロジーサービス部第四課

業務内容

・2025年　新卒入社
・Zabbixのサポート＆構築業務担当



目次

- 概要
- 機能紹介

  Case1：ホスト登録
  Case2：アイテム作成＆トリガー作成
  Case3：マップ作成
  Case4：指定期間における監視データの集計
  Case5：GUI未対応の正規表現エクスポート
  Case6：障害発生時の初動対応リファレンス

- まとめ

～ 概要 ～

システム監視における3つの壁

企業のシステム監視において
広く利用されているOSSツール**“Zabbix”**
そのシステム監視における課題とは？

システム監視における3つの壁

エンジニア不足

- 人材確保が困難
- 属人化による負荷集中
- 運用チームの小規模化

監視設定の複雑化

- 監視対象ごとに異なる設定方法
- 設定ミスのリスク

データ活用不足

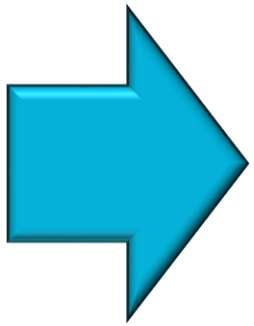
- データ活用の仕組み不足
- データ分析に時間がかかり障害対応が遅れる

AIによる解決策（ Zabbix × AI ）

エンジニア不足

監視設定の複雑化

データ活用不足



AIを新たな「労働力」へ

AIによる監視設定の自動化

AIによるインシデント対応支援

AIエージェントが解決する3つの壁と各機能紹介

AIを新たな「労働力」へ

AIによる監視設定の自動化

AIによるインシデント対応支援

- Case1 ： ホスト登録
- Case2 ： アイテム＆トリガー作成
- Case3 ： マップ作成
- Case4 ： 監視データ集計
- Case5 ： 正規表現エクスポート
- Case6 ： 初動対応リファレンス

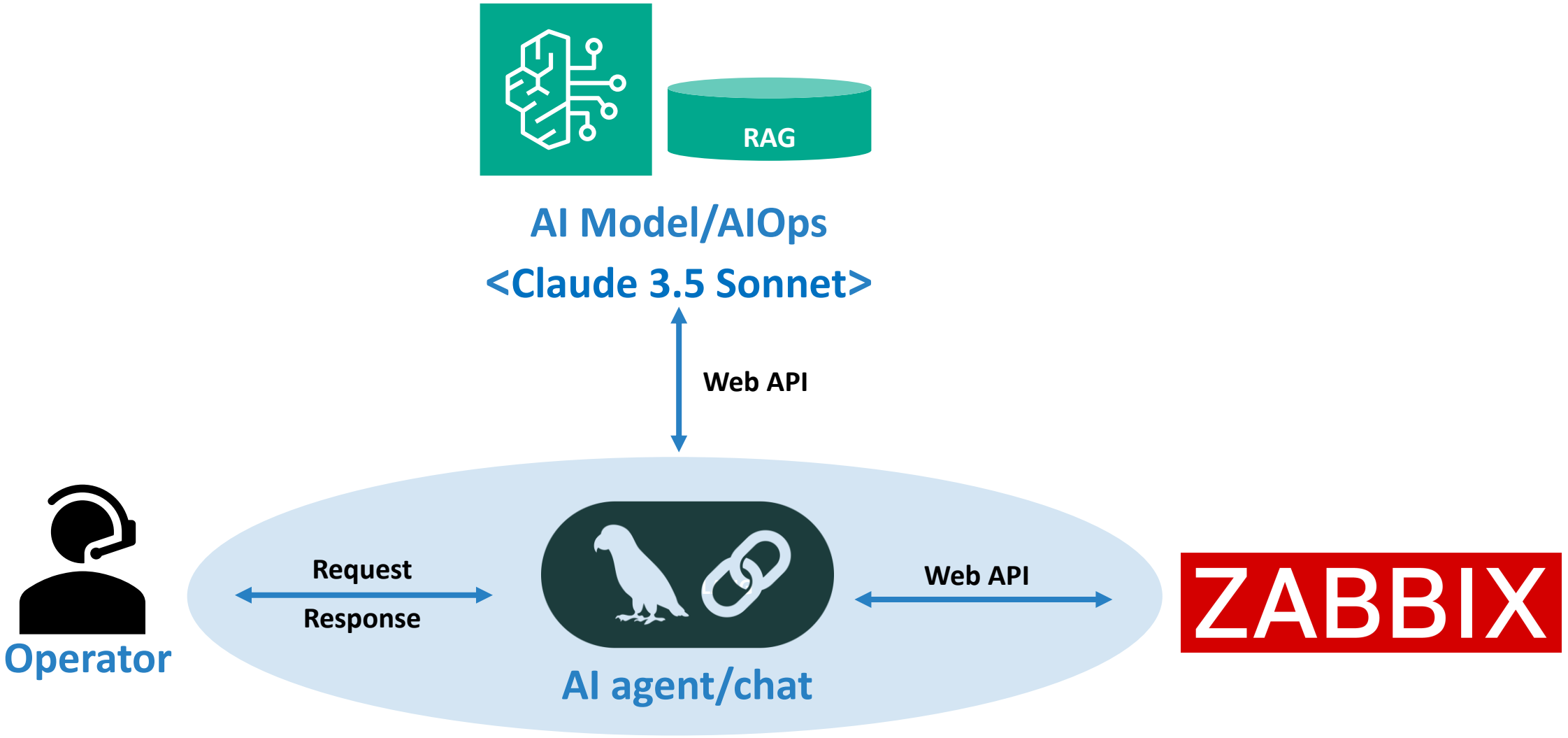
AIエージェントとは

＜AIエージェント＞

単なる質疑応答に留まらず、状況把握をして、必要なツールを操作して、
タスクを完了する能力を持つ。

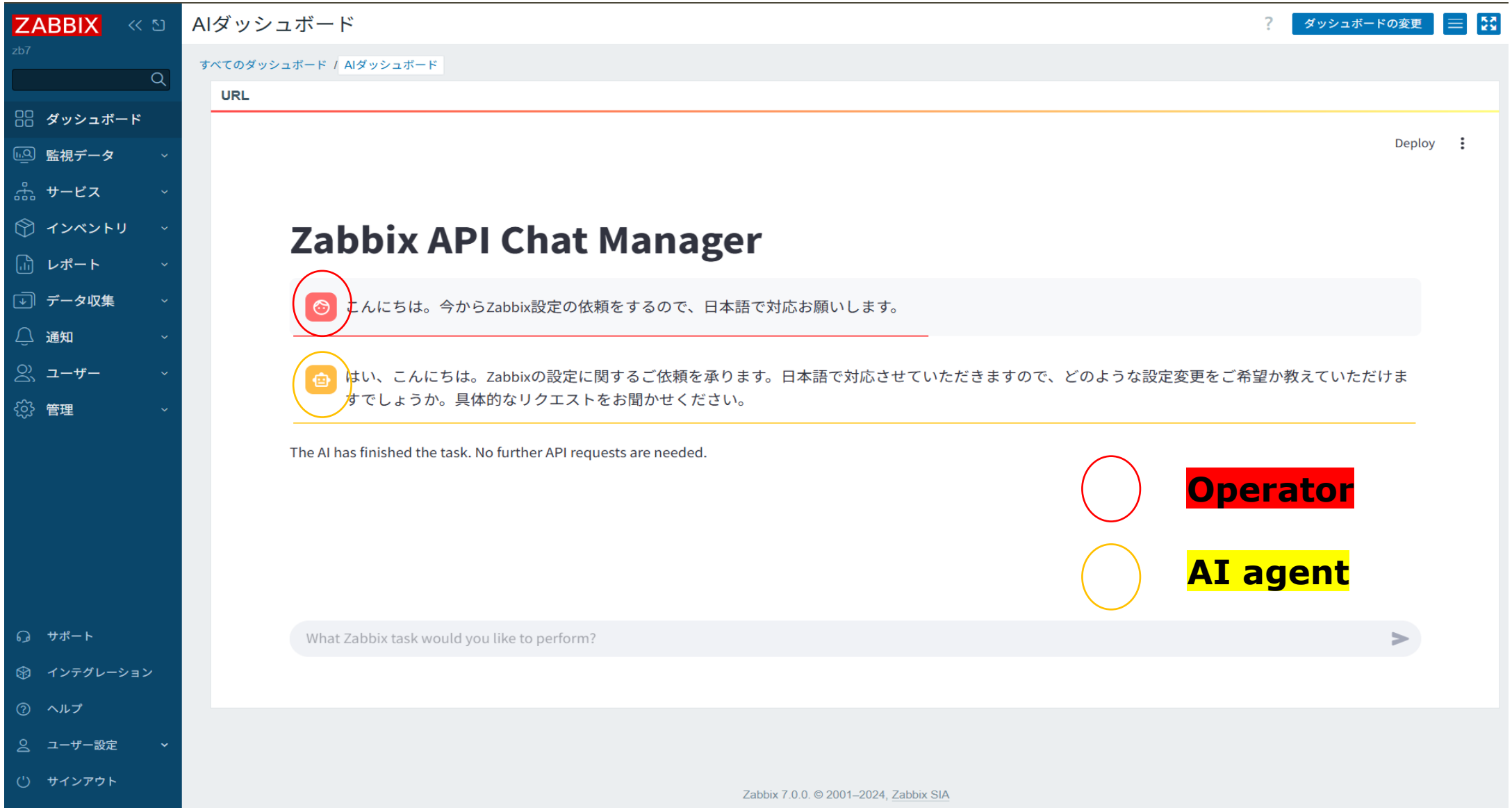
比較	ChatGPT（生成AI）	AIエージェント
基本構成	単体のAIモデル	生成AI＋ツール連携 （API、DB、スケジューラー等）
機能	質問応答、文章作成	タスク実行、業務支援
（例）	質問に答える先生	指示して動くアシスタント
活用シーン	アイデア出し、要約、メール作成	業務の自動化

システム構成図



チャット画面

・AIダッシュボードによるチャット機能



～ 機能紹介 ～

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

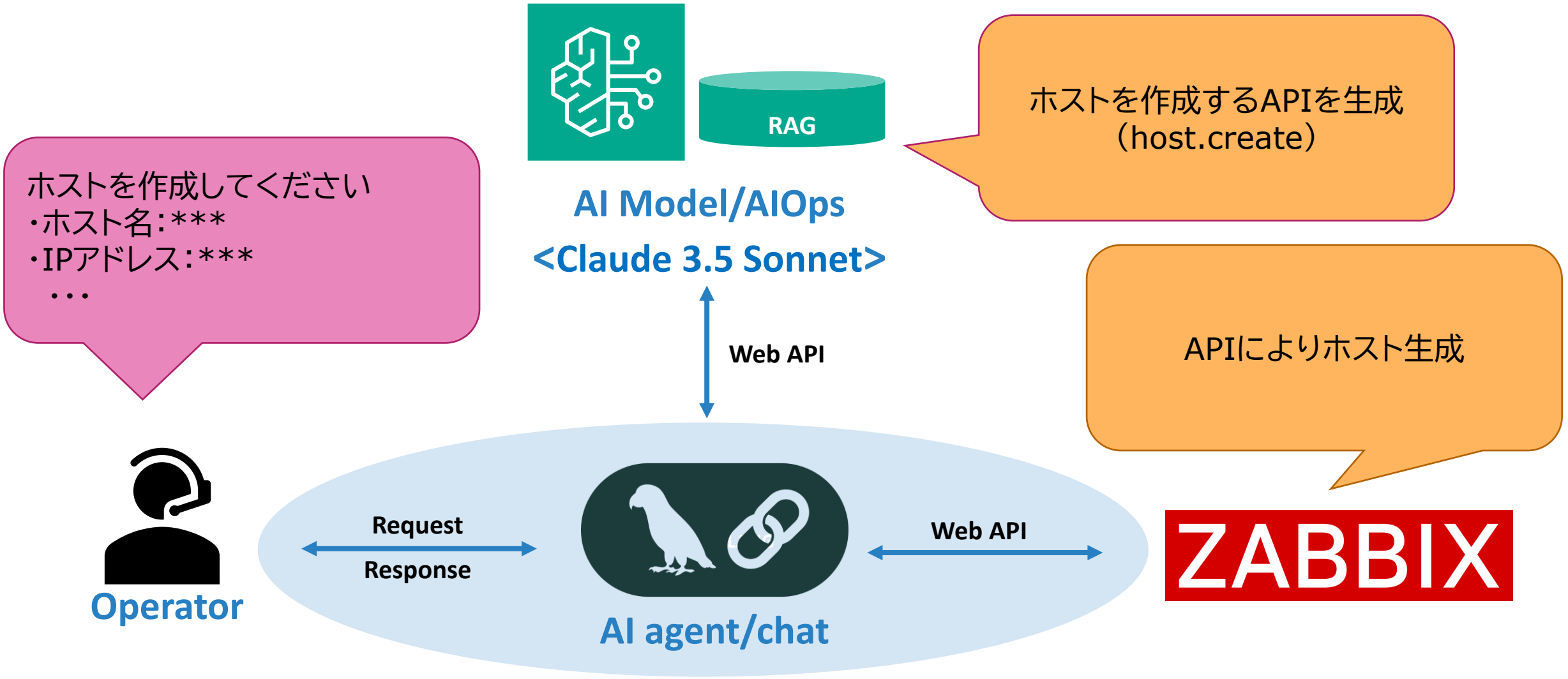
Case3：マップ作成

Case4：指定期間における監視データの集計

Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

システム構成図



Case1：ホスト登録（単体）

・ホスト登録依頼

新規サーバーの監視設定を構成してください。

ホストの作成を依頼する

承知いたしました。新規サーバーの監視設定を構成するために、いくつか情報が必要です。以下の点について教えていただけますでしょうか？

1. 新規サーバーのホスト名（または表示名）
2. サーバーのIPアドレス
3. 監視に使用するテンプレート（例：Template OS Linux by Zabbix agent など）
4. ホストグループ（既存のグループ名または新規作成）
5. Zabbixエージェントのポート番号（デフォルトは10050）

これらの情報をご提供いただければ、適切なAPIリクエストを作成できます。

必要な情報をヒアリングしてくれる

The AI has finished the task. No further API requests are needed.

What Zabbix task would you like to perform?

Case1：ホスト登録（単体）

・ホスト登録依頼

承知いたしました。新規サーバーの監視設定を構成するために、いくつか情報が必要です。以下の点について教えていただけますでしょうか？

1. 新規サーバーのホスト名（または表示名）
2. サーバーのIPアドレス
3. 監視に使用するテンプレート（例：Template OS Linux by Zabbix agent など）
4. ホストグループ（既存のグループ名または新規作成）
5. Zabbixエージェントのポート番号（デフォルトは10050）

これらの情報をご提供いただければ、適切なAPIリクエストを作成できます。

1: WEBServer001 2: 192.168.11.100 3: Linux by Zabbix agent 4 : AI-CREATE 5: 10050

1：WebServer001
2：192.168.11.100
3：Linux by Zabbix agent
4：AI-CREATE
5：10050

Case1：ホスト登録（単体）

・AIエージェントによるレスポンス

承知いたしました。新規サーバーの監視設定を構成するために、Zabbix APIを使用してホストを作成します。

まず、ホストグループとテンプレートのIDを取得する必要があります。そのために、次の2つのサブリクエストを実行します。

<API-SUB>{"jsonrpc": "2.0", "method": "hostgroup.get", "params": {"output": ["groupid"], "filter": {"name": ["AI-CREATE"]}}, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 1}</API-SUB>
このリクエストで "AI-CREATE" ホストグループのIDを取得します。もし存在しない場合は、新しく作成する必要があります。

<API-SUB>{"jsonrpc": "2.0", "method": "template.get", "params": {"output": ["templateid"], "filter": {"name": ["Linux by Zabbix agent"]}}, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 2}</API-SUB>
このリクエストで "Linux by Zabbix agent" テンプレートのIDを取得します。

これらのサブリクエストの結果を使用して、メインのホスト作成リクエストを作成します。ホストグループIDを `groupid` に、テンプレートI `templateid` に置き換えてください。

<API-MAIN>{"jsonrpc": "2.0", "method": "host.create", "params": {"host": "WEBServer001", "interfaces": [{"type": 1, "main": 1, "useip": 1, "ip": "192.168.11.100", "dns": "", "port": "10050"}], "groups": [{"groupid": "groupid"}], "templates": [{"templateid": "templateid"}], "inventory_mode":

・ホストグループのIDを取得するAPIリクエストを生成

・APIによりZabbixで生成

The AI needs you to run this API request to gather data:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": ["groupid"],
        "filter": {
            "name": [
                "AI-CREATE"
            ]
        }
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 1
}
```

API Response:

```
{
  "jsonrpc" : "2.0"
  "result" : [
    0 : {
      "groupid" : "35"
    }
  ]
  "id" : 1
}
```

・Zabbix APIを実行

Case1：ホスト登録（単体）

・レスポンス結果画面

素晴らしいです！APIレスポンスから、新しいホストが正常に作成されたことが確認できました。

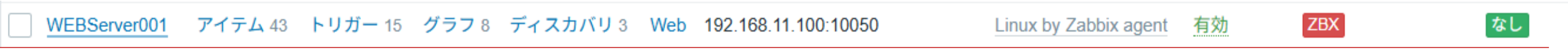
結果の解釈:

- "result": {"hostids": ["11760"]} は、新しく作成されたホストのIDが 11760 であることを示しています。

これで、以下の設定で新しいホストが正常にZabbixに追加されました：

1：WebServer001
2：192.168.11.100
3：Linux by Zabbix agent
4：AI-CREATE
5：10050

・作成されたホスト



Case1：ホスト登録（一括登録）

・条件を指定し、一括でホスト登録をリクエスト

以下のデータに基づき、Zabbixへの新規ホストの一括登録を実行してください。

＜ホスト名 ，IPアドレス ，テンプレート ，ホストグループ ＞
①AppServer01 ，192.168.2.11 ，Linux by Zabbix agent ，AI-CREATE
②AppServer02 ，192.168.2.12 ，Linux by Zabbix agent ，AI-CREATE
③AppServer03 ，192.168.2.13 ，Linux by Zabbix agent ，AI-CREATE
④AppServer04 ，192.168.2.14 ，Linux by Zabbix agent ，AI-CREATE
⑤AppServer05 ，192.168.2.15 ，Linux by Zabbix agent ，AI-CREATE

1：AppServer01~05
2：192.168.2.11~15
3：Linux by Zabbix agent
4：AI-CREATE

Case1：ホスト登録（一括登録）

・APIによりZabbixで生成

URL

Deploy

The AI asks you to execute this API request to modify Zabbix:


```
{
  "jsonrpc": "2.0",
  "method": "host.create",
  "params": [
    {
      "host": "AppServer01",
      "interfaces": [
        {
          "type": 1,
          "main": 1,
          "useip": 1,
          "ip": "192.168.2.11",
          "dns": "",
          "port": "10050"
        }
      ],
      "groups": [
        {
          "groupid": "35"
```


What Zabbix task would you like to perform?

・Zabbix APIを実行

・レスポンス結果画面

URL

Deploy

ありがとうございます。APIレスポンスを確認しました。

結果は成功を示しています。5台のホストが正常に登録されたことが確認できます。各ホストに割り当てられたホストIDは以下の通りです：

1. AppServer01: 11752
2. AppServer02: 11753
3. AppServer03: 11754
4. AppServer04: 11755
5. AppServer05: 11756

これで、要求された5台のホスト（AppServer01からAppServer05）が以下の設定で正常にZabbixに登録されました：

- 指定されたホスト名とIPアドレス
- ホストグループ "AI-CREATE"
- テンプレート "Linux by Zabbix agent"
- Zabbixエージェントのデフォルトポート (10050)

これで一括登録のプロセスは完了しました。新しく追加されたホストはZabbix上で監視可能な状態になっています。

追加の設定や確認が必要な場合は、お知らせください。また、他に何かご質問やご要望がありましたら、お気軽にお申し付けください。

Case1：ホスト登録（一括登録）

1：AppServer01~05
2：192.168.2.11~15
3：Linux by Zabbix agent
4：AI-CREATE

・一括登録されたホスト

名前 ▲	アイテム	トリガー	グラフ	ディスカバリ	Web	インターフェース		
AppServer01	アイテム 43	トリガー 15	グラフ 8	ディスカバリ 3	Web	192.168.2.11:10050	Linux by Zabbix agent	有効
AppServer02	アイテム 43	トリガー 15	グラフ 8	ディスカバリ 3	Web	192.168.2.12:10050	Linux by Zabbix agent	有効
AppServer03	アイテム 43	トリガー 15	グラフ 8	ディスカバリ 3	Web	192.168.2.13:10050	Linux by Zabbix agent	有効
AppServer04	アイテム 43	トリガー 15	グラフ 8	ディスカバリ 3	Web	192.168.2.14:10050	Linux by Zabbix agent	有効
AppServer05	アイテム 43	トリガー 15	グラフ 8	ディスカバリ 3	Web	192.168.2.15:10050	Linux by Zabbix agent	有効

Case1：ホスト登録（一括登録）

AIエージェントにより**ホスト登録が“人手いらず”に。**
定型業務からエンジニアを開放

AIを新たな「労働力」へ

AIによる監視設定の自動化

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

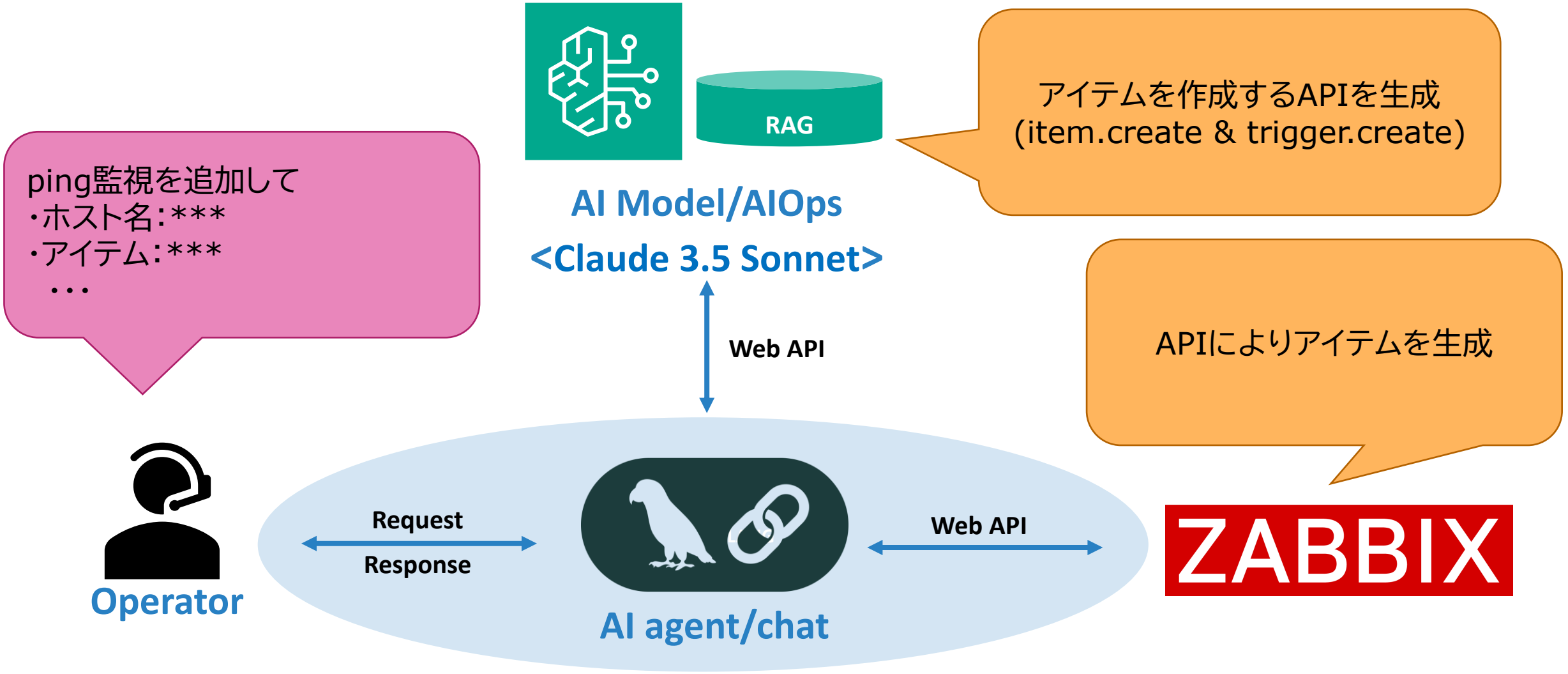
Case3：マップ作成

Case4：指定期間における監視データの集計

Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

システム構成図



Case2：アイテム作成

・アイテム作成依頼

ホストAppServer01 に、ICMP Ping監視を追加設定してください。

対象ホストにPing監視のアイテムの作成を依頼

かしこまりました。ホストAppServer01にICMP Ping監視を追加設定いたします。この設定を行うために、以下の情報が必要です：

1. ホストAppServer01のホストID
2. 使用するICMP Pingテンプレートの名前またはID

これらの情報を確認させていただきます。

まず、ホストAppServer01のホストIDを取得するために、以下のAPIリクエストを作成します：

<API-SUB>{ "jsonrpc": "2.0", "method": "host.get", "params": { "output": ["hostid"], "filter": { "host": ["AppServer01"] } }, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 1 }</API-SUB>
このAPIリクエストを実行して、ホストAppServer01のホストIDを取得してください。

次に、ICMP Pingテンプレートの情報を取得するために、以下のAPIリクエストを作成します：

<API-SUB>{ "jsonrpc": "2.0", "method": "template.get", "params": { "output": ["templateid"], "filter": { "name": ["Template ICMP Ping"] } }, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 2 }</API-SUB>
このAPIリクエストを実行して、ICMP Pingテンプレートのテンプレートを取得してください。

これらの情報が揃いましたら、ICMP Ping監視の追加設定を行うためのAPIリクエストを作成いたします。

対象のホストIDを自動取得して、アイテム生成

S {"jsonrpc": "2.0", "result": {"itemids": ["104693"]}, "id": 2}

API response: {"jsonrpc": "2.0", "result": {"itemids": ["104693"]}, "id": 2}

Zabbix APIを実行

Case2：アイテム作成

・レスポンス結果画面

API response: {"jsonrpc": "2.0", "result": {"itemids": ["104693"]}, "id": 2}

素晴らしいです！APIリクエストが成功し、新しいICMP Pingアイテムが正常に作成されました。

結果から、以下の情報が分かります：

- 新しく作成されたアイテムのID: 104693

これで、AppServer01にICMP Pingの監視アイテムが追加されました。このアイテムは、サーバーがICMP Pingに応答するかどうかを1分ごとにチ
す。

次のステップとして、以下のようなアクションを検討できます：

1. トリガーの作成: ICMP Pingが失敗した場合にアラートを発生させるトリガーを設定する。
2. グラフの作成: ICMP Pingの応答時間を視覚化するグラフを作成する。
3. スクリーンやダッシュボードへの追加: 新しいアイテムをスクリーンやダッシュボードに追加して、監視を容易にする。

これらの追加設定のいずれかを行いたい場合は、お知らせください。具体的なAPIリクエストを提供します。

また、他のサーバーにも同様の設定を適用したい場合や、他の監視項目を追加したい場合も、遠慮なくお申し付けください。

Case2：アイテム作成

・作成されたアイテム

名前:ICMP ping
タイプ:シンプルチェック
キー:icmpping
データ型:数値(整数)
監視間隔:1m
ヒストリ:90d
トレンド:365d

Case2：トリガー作成

・トリガー作成依頼

今作成したアイテムに、トリガーを作成してください。

作成したアイテム「ICMP ping」にトリガーをつける

はい、承知しました。ICMP Pingアイテムに対してトリガーを作成します。このトリガーは、ICMPの応答がない（つまり、サーバーが到達不能
ラートを発生させます。

以下のAPIリクエストを使用して、トリガーを作成します：

<API-MAIN> { "jsonrpc": "2.0", "method": "trigger.create", "params": { "description": "AppServer01 is unreachable by ICMP", "expression":
"last(/AppServer01/icmpping)=0", "priority": 4, "status": 0, "type": 0, "recovery_mode": 0, "recovery_expression": "", "manual_close": 0, "comments": "T
not responding to ICMP ping requests.", "url": "", "tags": [{ "tag": "server", "value": "AppServer01" }, { "tag": "service", "value": "ping" }] }, "auth":
"4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 3 } </API-MAIN>
このトリガーの設定内容は以下の通りです：

APIリクエストを生成

1. 説明: "AppServer01 is unreachable by ICMP"
2. 式: "last(/AppServer01/icmpping)=0" （最後のICMP pingが0（失敗）の場合にトリガー）
3. 重要度: 4 （高）
4. ステータス: 0 （有効）
5. タイプ: 0 （シングル）
6. 復旧モード: 0 （式）
7. 手動クローズ: 0 （不可）
8. コメント: サーバーがICMP pingに応答していないことを説明
9. タグ: サーバー名とサービス（ping）を識別するためのタグを追加

このAPIリクエストを実行してください。正常に作成された場合、新しいトリガーIDが返されるはずです。実行後、結果を教えてください。

Case2：トリガー作成

・APIによりZabbixで生成

The AI asks you to execute this API request to modify Zabbix:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": {
        "description": "AppServer01 is unreachable by ICMP",
        "expression": "last(/AppServer01/icmpping)=0",
        "priority": 4,
        "status": 0,
        "type": 0,
        "recovery_mode": 0,
        "recovery_expression": "",
        "manual_close": 0,
        "comments": "The server is not responding to ICMP ping requests.",
        "url": "",
        "tags": [
            {
                "tag": "server",
                "value": "AppServer01"
            },
            {
                "tag": "service",
                "value": "ping"
            }
        ]
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 3
}
```

Zabbix APIを実行

Case2：トリガー作成

・レスポンス結果画面

素晴らしいです！トリガーが正常に作成されました。

APIレスポンスから、以下の情報が分かります：

- 新しく作成されたトリガーのID: 25576

これで、AppServer01のICMP Pingアイテムに対するトリガーが設定されました。このトリガーは以下のように動作します：

1. AppServer01がICMP Pingに応答しない（つまり、icmppingの値が0）場合、トリガーが発動します。
2. トリガーが発動すると、"AppServer01 is unreachable by ICMP" というメッセージが表示されます。
3. このトリガーの重要度は "高" に設定されています。

次のステップとして、以下のようなアクションを検討できます：

1. アクションの作成: このトリガーが発動した際に、特定のユーザーに通知を送るなどのアクションを設定する。
2. グラフの作成: ICMP Pingの応答時間を視覚化するグラフを作成する。
3. ダッシュボードへの追加: 新しいアイテムとトリガーの状態をダッシュボードに追加して、監視を容易にする。

これらの追加設定のいずれかを行いたい場合は、お知らせください。具体的なAPIリクエストを提供します。

また、他のサーバーにも同様の設定を適用したい場合や、他の監視項目を追加したい場合も、遠慮なくお申し付けください。

Case2：トリガー作成

・作成されたトリガー

名前：**AppServer01 is unreachable by ICMP**
条件式：**last(/AppServer01/icmpping)=0**
正常イベントの生成：条件式
障害イベント生成モード：単一
正常時のイベントクローズ：すべての障害
ステータス：有効

Case2：トリガー作成

Case 1同様に、AIエージェントにより
アイテム＆トリガー作成が“人手いらず”に。
自動化により、設定ミスを削減

AIを新たな「労働力」へ

AIによる監視設定の自動化

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

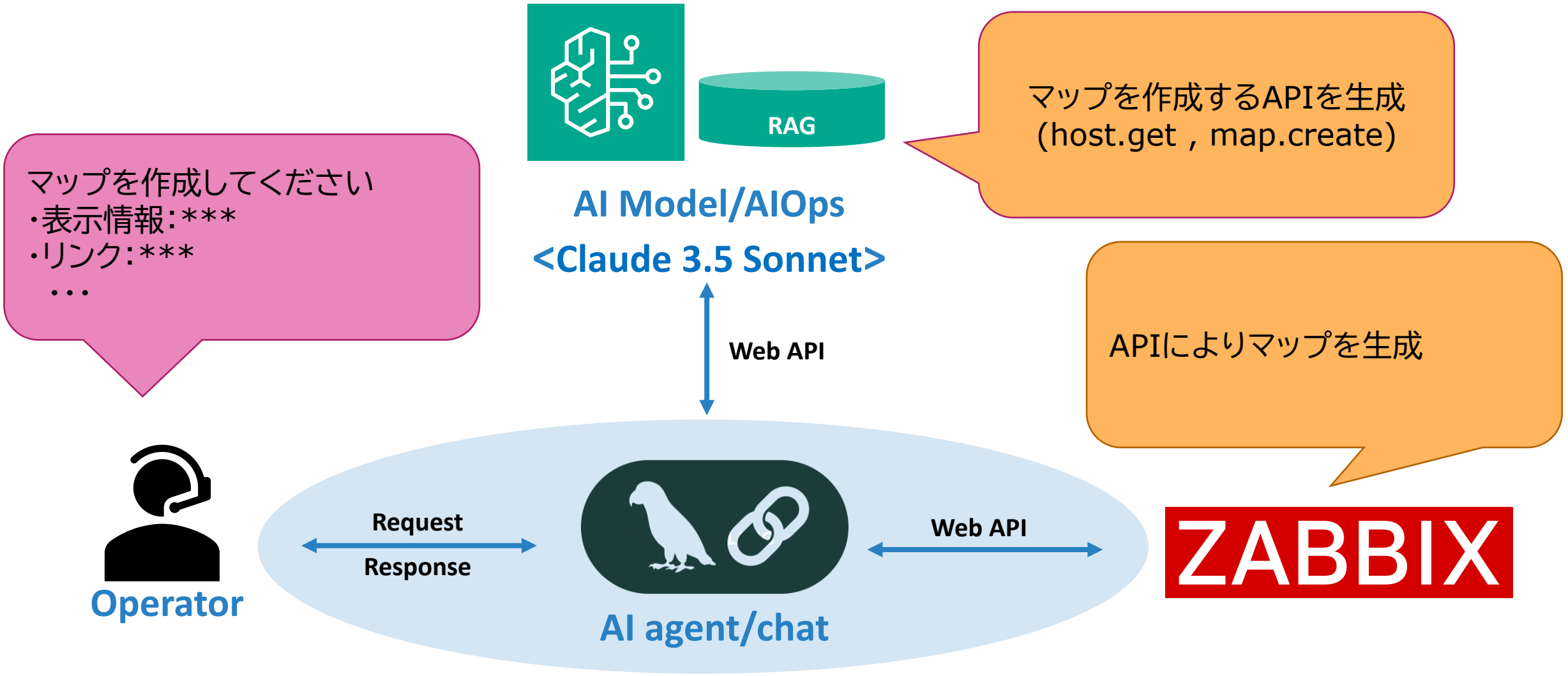
Case3：マップ作成

Case4：指定期間における監視データの集計

Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

システム構成図



Case3：マップの作成

・本ケーススタディで作成するマップ

①ネットワークマップ ： 物理・論理構成を可視化

（効果）

◆ 障害発生時の影響範囲を即座に把握

◆ 運用保守の効率化

②WEB/AP/DB 三階層マップ ： アプリ構成を階層的に表現

（効果）

◆ 問題発生個所の特定が容易

◆ 稼働率の可視化による品質管理

Case3：マップの作成(ネットワークマップ)

・ホストグループ“AI-CREATE”に属するホストでマップ作成
・各ホストの障害や深刻度を確認
・WebサーバーからAppサーバーへのpingステータス確認可能

・ネットワークマップの作成依頼

The AI has finished the task. No further API requests are needed.

ホストグループ"AI-CREATE"に含まれるすべてのホストを使用して、新規でネットワークマッピングを作成してください。
＜詳細設定＞
マップ名：AI-CREATE NetworkMap
表示情報：各ホストアイコンの下に、現在の障害の深刻度を表示させてください。
リンク：WebServerから、すべてのAppServerに対して、pingステータスを示す線を追加してください。

ホストグループのIDは自動取得して、マップのサイズは1000*1000で、他の設定はデフォルト値でお願いします。

Case3：マップの作成

・ホストグループを取得

ホストグループ"AI-CREATE"に含まれるすべてのホストを使用して、新規でネットワークマッピングを作成してください。 <詳細設
CREATE servers network map 表示情報：各ホストのアイコンの下に、ホスト名と現在の障害の深刻度を表示させてください。 リンク
ら、すべてのAppサーバーに、pingステータスを示す線を追加してください。

ホストグループのIDは自動取得して、その他の設定はデフォルト値で作成をお願いします。

はい、承知しました。新規でネットワークマッピングを作成するために、いくつかの手順を踏む必要があります。まず、ホストグ
を取得し、そのグループに含まれるホストを取得します。その後、ネットワークマップを作成します。

まず、ホストグループ"AI-CREATE"のIDを取得しましょう。

<API-SUB>{"jsonrpc": "2.0", "method": "hostgroup.get", "params": {"output": ["groupid"], "filter": {"name": ["AI-CREATE"]}}, "auth":
"4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 1}</API-SUB>
このAPIを実行して、ホストグループのIDを取得してください。

The AI needs you to run this API request to gather data:

情報取得のAPIリクエストの生成

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": ["groupid"],
        "filter": {
            "name": ["AI-CREATE"]
        }
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 1
}
```

API Response:

Zabbix APIを実行

```
{
  "jsonrpc" : "2.0"
  "result" : [
    0 : {
      "groupid" : "35"
    }
  ]
  "id" : 1
}
```

・ホストを取得し、Web,Appサーバーを識別

次に、このホストグループに含まれるすべてのホストを取得し、Webサーバーとアプリケーションサーバーを識別する必要があります。以下
してホスト情報を取得しましょう。

<API-SUB>{"jsonrpc": "2.0", "method": "host.get", "params": {"output": ["hostid", "host", "name"], "groupids": "35", "selectInterfaces": ["ip"], "selectTa
"extend"}, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 2}</API-SUB>
このAPIを実行して、ホストの情報を取得してください。

The AI needs you to run this API request to gather data:

情報取得のAPIリクエストの生成

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host", "name"],
        "groupids": "35",
        "selectInterfaces": ["ip"],
        "selectTags": "extend"
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 2
}
```

API Response:

Zabbix APIを実行

```
{
  "jsonrpc" : "2.0"
  "result" : [
    0 : {
      "hostid" : "11746"
      "host" : "WebServer01"
      "name" : "WebServer01"
      "interfaces" : [
        0 : {
          "ip" : "192.168.11.100"
        }
      ]
      "tags" : []
    }
    1 : {
      "hostid" : "11752"
      "host" : "AppServer01"
```

Case3：マップの作成

・APIリクエストを自動実行

ZabbixがAPIリクエストを実行し、マップを生成


```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "AI-CREATE servers network map",
        "width": 800,
        "height": 600,
        "selements": [
            {"selementid": 1, "elements": [{"hostid": "11746"}], "elementtype": 0, "iconid_off": "2", "label": "WebServer01\r
            {"selementid": 2, "elements": [{"hostid": "11752"}], "elementtype": 0, "iconid_off": "2", "label": "AppServer01\r
            {"selementid": 3, "elements": [{"hostid": "11753"}], "elementtype": 0, "iconid_off": "2", "label": "AppServer02\r
            {"selementid": 4, "elements": [{"hostid": "11754"}], "elementtype": 0, "iconid_off": "2", "label": "AppServer03\r
            {"selementid": 5, "elements": [{"hostid": "11755"}], "elementtype": 0, "iconid_off": "2", "label": "AppServer04\r
            {"selementid": 6, "elements": [{"hostid": "11756"}], "elementtype": 0, "iconid_off": "2", "label": "AppServer05\r
        ],
        "links": [
            {"selementid1": 1, "selementid2": 2, "color": "00CC00"},
            {"selementid1": 1, "selementid2": 3, "color": "00CC00"},
            {"selementid1": 1, "selementid2": 4, "color": "00CC00"},
            {"selementid1": 1, "selementid2": 5, "color": "00CC00"},
            {"selementid1": 1, "selementid2": 6, "color": "00CC00"}
```


Case3：マップの作成

・レスポンス結果画面

素晴らしいです！ネットワークマップが正常に作成されました。

APIレスポンスから、新しく作成されたマップのIDは "33" であることがわかります。

まとめると、以下の操作が完了しました：

1. ホストグループ "AI-CREATE" (ID: 35) に含まれるすべてのホストを取得しました。
2. "AI-CREATE servers network map" という名前の新しいネットワークマップを作成しました（マップID: 33）。
3. マップには以下の要素が含まれています：
    - WebServer01とすべてのAppServerが配置されています。
    - 各ホストのアイコンの下に、ホスト名と現在の障害の深刻度を表示するラベルが設定されています。
    - Webサーバーからすべてのappサーバーへのリンク（緑色の線）が追加されています。

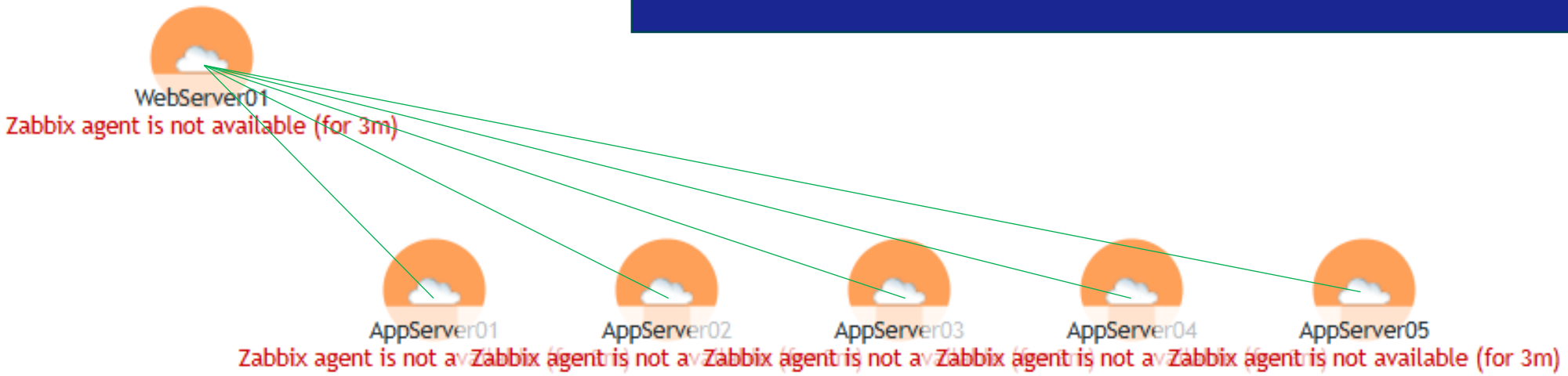
このマップは現在Zabbixシステムに保存されており、Zabbixのウェブインターフェースから確認することができます。必要に応じて、ウェブインターフェースから微調整を行うこともできます。

他に何か追加や変更が必要な点はありますか？または、Zabbixに関する他の質問はありますか？

Case3：マップの作成(ネットワークマップ)

・作成されたマップ

各ホストで、現在の障害と深刻度を確認可能
WebサーバーからAppサーバーへのping通信も確認可能



Case3：マップの作成(三階層マップ)

・三階層マップの作成依頼

「AI-CREATE」ホストグループに属するホストを使用してマップを作成してください。

マップ上に線を引き、WEB、AP、DBの3つの層に明確に区分けしてください。

ホストグループ“AI-CREATE”に属するホストで、WEB, AP, DBの三階層に分けてマップを作成

ご要望ありがとうございます。「AI-CREATE」ホストグループに属するホストを使用してマップを作成し、WEB、AP、DBの3つの層に区分けするマップを作成するために、以下の情報が必要です。

1. マップの名前
2. マップのサイズ（幅と高さ）
3. 各層（WEB、AP、DB）に属するホストの情報

これらの情報を教えていただけますでしょうか？また、ホストグループ「AI-CREATE」に属するホストの一覧を取得する必要がありますので、はそのAPIリクエストを作成してもよろしいでしょうか？

詳細情報をヒアリング

1. 「AI-CREATE Three_Layer Map」
2. 800×600
3. WebServer, AppServerなど自動取得してください。

マップ名、サイズ等を指定

ありがとうございます。マップの名前とサイズの情報を確認しました。ホストの情報を自動取得するために、まず「AI-CREATE」ホストグループに属するホストの一覧を取得します。その後、ホスト名に基づいて各層（WEB、AP、DB）に分類します。

まずは、ホストグループ「AI-CREATE」に属するホストの一覧を取得するAPIリクエストを作成します。

<API-SUB>{ "jsonrpc": "2.0", "method": "host.get", "params": { "output": ["hostid", "host", "name"], "groupids": "AI-CREATE", "selectInterfaces": ["interfa
"ip"] }, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 1 }</API-SUB>
このAPIリクエストを実行していただけますでしょうか？結果を教えていただければ、それに基づいてホストを分類し、マップ作成のAPIリクエストを作成いたします。

APIリクエストを生成

Case3：マップの作成(三階層マップ)

・APIリクエストを自動実行

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "AI-CREATE Three_Layer Map",
        "width": 800,
        "height": 600,
        "selements": [
            {
                "selementid": 1,
                "elements": [{"hostid": "11746"}],
                "elementtype": 0,
                "iconid_off": "185",
                "label": "WebServer01",
                "x": 200,
                "y": 50
            },
            {
                "selementid": 2,
                "elements": [{"hostid": "11752"}],
                "elementtype": 0,
                "iconid_off": "185",
                "label": "AppServer01",
                "x": 100,
                "y": 250
            },
            {
                "selementid": 3,
                "elements": [{"hostid": "11753"}],
                "elementtype": 0,
                "iconid_off": "185",
                "label": "AppServer02",
                "x": 250,
                "y": 250
            },
            {
                "selementid": 4,
                "elements": [{"hostid": "11754"}],
                "elementtype": 0,
                "iconid_off": "185",
                "label": "AppServer03",
                "x": 400,
                "y": 250
            },
            {
                "selementid": 7,
                "elementtype": 4,
                "iconid_off": "185",
                "label": "DBServer (Dummy)",
                "x": 400,
                "y": 450
            },
            {
                "selementid": 8,
                "elementtype": 4,
                "iconid_off": "2",
                "label": "WEB Layer",
                "x": 50,
                "y": 20,
                "width": 700,
                "height": 150,
                "border_type": 1
            },
            {
                "selementid": 9,
                "elementtype": 4,
                "iconid_off": "2",
                "label": "AP Layer",
                "x": 50,
                "y": 200,
                "width": 700,
                "height": 150,
                "border_type": 1
            },
            {
                "selementid": 10,
                "elementtype": 4,
                "iconid_off": "2",
                "label": "DB Layer",
                "x": 50,
                "y": 400,
```

Zabbix APIを実行

API Response:


```
{
  "jsonrpc" : "2.0"
  "result" : {
    "sysmapids" : [
      0 : "35"
    ]
  }
  "id" : 1
}
```

API response: {"jsonrpc": "2.0", "result": {"sysmapids": ["35"]}, "id": 1}

Case3：マップの作成(三階層マップ)

・レスポンス結果画面

素晴らしいです！APIリクエストが成功し、新しいマップが作成されました。

APIの応答から、以下の情報が分かります：

- マップが正常に作成されました。
- 作成されたマップのID（sysmapid）は "35" です。

これで、「AI-CREATE Three_Layer Map」という名前の3層構造のマップが作成されました。このマップには以下の特徴があります：

1. サイズ: 800x600 ピクセル
2. WEB、AP、DBの3つの層が含まれています
3. 各層は枠で囲まれ、ラベルが付いています
4. 実際のホスト（WebServer01とAppServer01-05）が適切な層に配置されています
5. DBサーバーはダミーとして追加されています
6. ホスト間の接続が線で表現されています

このマップは以下の方法で確認できます：

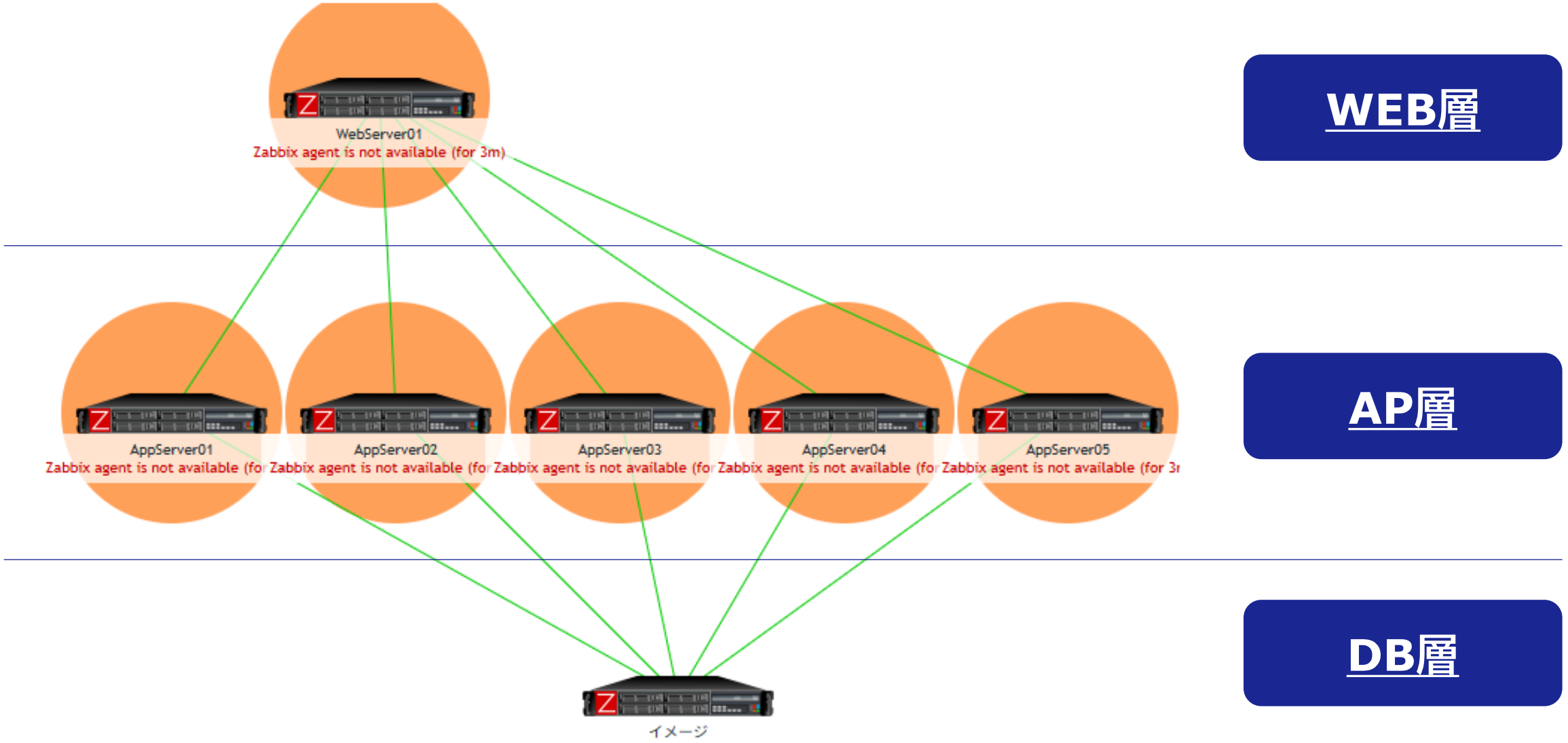
1. Zabbixの管理画面にログインします。
2. 「監視」>「マップ」に移動します。
3. 「AI-CREATE Three_Layer Map」という名前のマップを探し、クリックして表示します。

マップの外観や構成に調整が必要な場合は、お知らせください。例えば、要素の位置の調整、色の変更、追加の情報の表示などが可能です。
このマップに関連して他に何か必要なことがあればお知らせください。

The AI has finished the task. No further API requests are needed.

Case3：マップの作成(三階層マップ)

・作成されたマップ



Case3：マップの作成

AIが監視対象ネットワークを自動認識・マッピングし、
その情報を活用することで、監視設定の自動化に貢献。

AIによる監視設定の自動化

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

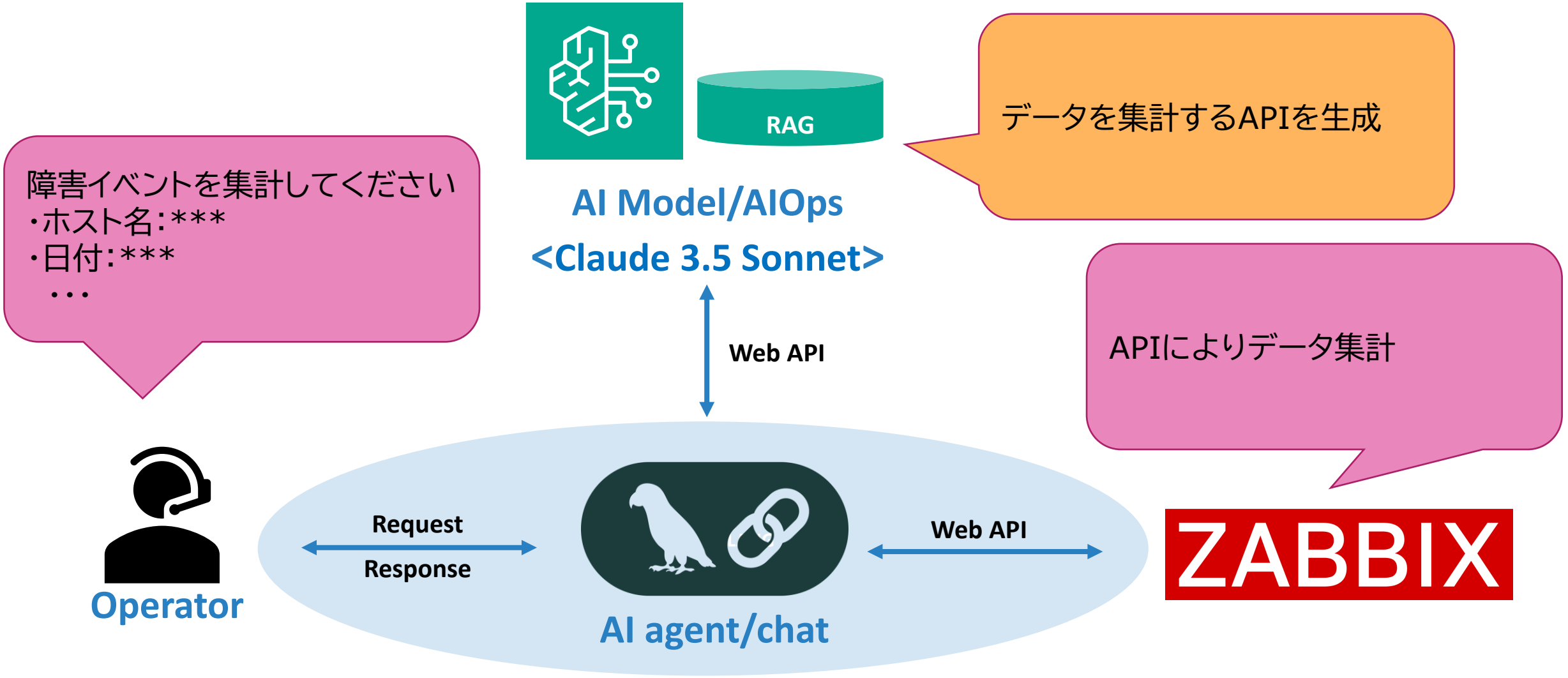
Case3：マップ作成

Case4：指定期間における監視データの集計

Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

システム構成図



Case4：監視データの集計

・監視データの集計依頼

Zabbix API Chat Manager

今からZabbixの設定依頼をします。日本語で対応してください。

承知いたしました。Zabbixの設定依頼について、日本語で対応させていただきます。どのような設定変更をご希望でしょうか？具体的なリクエストをお聞かせください。ご要望に応じて、ZabbixAPIを使用して設定変更を行うためのリクエストボディを作成いたします。

不明な点や追加情報が必要な場合は、確認させていただきますので、よろしくお願いいたします。それでは、Zabbixの設定依頼についてお聞かせください。

The AI has finished the task. No further API requests are needed.

指定期間の障害イベントをホストごとに集計

2025年10月1日に発生した問題イベントを、ホストごとに集計してください。

Case4：監視データの集計

・監視データの集計

1. 2025年10月1日（UTC）のイベントを取得します（time_fromとtime_tillで指定）
2. ホスト情報を含めて問題イベントを取得します
3. イベントIDでソートし、最大10000件のイベントを取得します（制限に注意）

APIの実行結果を受け取った後、以下の手順で結果を処理してください：

1. 取得したイベントデータをホストIDでグループ化します
2. 各ホストのイベント数をカウントします
3. ホストIDをホスト名に置き換えます（先ほど取得したhost.getの結果を使用）
4. ホスト名と問題イベント数のリストを作成します

この処理により、2025年10月1日に発生した問題イベントがホストごとに集計されます。結果は「ホスト名：イベント数」の形式で表示されま

何か追加の情報や修正が必要な場合は、お知らせください。

ホスト情報取得のAPIリクエストの生成

The AI needs you to run this API request to gather data:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectGroups": "extend"
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
```

Zabbix APIを実行

Case4：監視データの集計

・監視データの集計

The AI asks you to execute this API request to modify Zabbix:


```
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": ["eventid", "objectid", "clock", "ns", "r_eventid", "r_clock", "r_ns", "correlationid", "userid", "name", "
        "time_from": "1727722800",
        "time_till": "1727809199",
        "sortfield": ["eventid"],
        "sortorder": "DESC",
        "limit": 10000
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 1
}
```


障害イベント情報取得のAPIリクエストの生成

API Response:


```
{
  "jsonrpc" : "2.0"
  "result" : []
  "id" : 1
}
```


Zabbix APIを実行

API response: {"jsonrpc": "2.0", "result": [], "id": 1}

Case4：監視データの集計

・レスポンス結果画面

対象ホストごとに、指定期間のイベント情報を出力

Case4：監視データの集計

AIが監視データを集計し、その情報を活用することで、
対応の優先度判断や初動の迅速化に貢献。

AIによるインシデント対応支援

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

Case3：マップ作成

Case4：指定期間における監視データの集計

Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

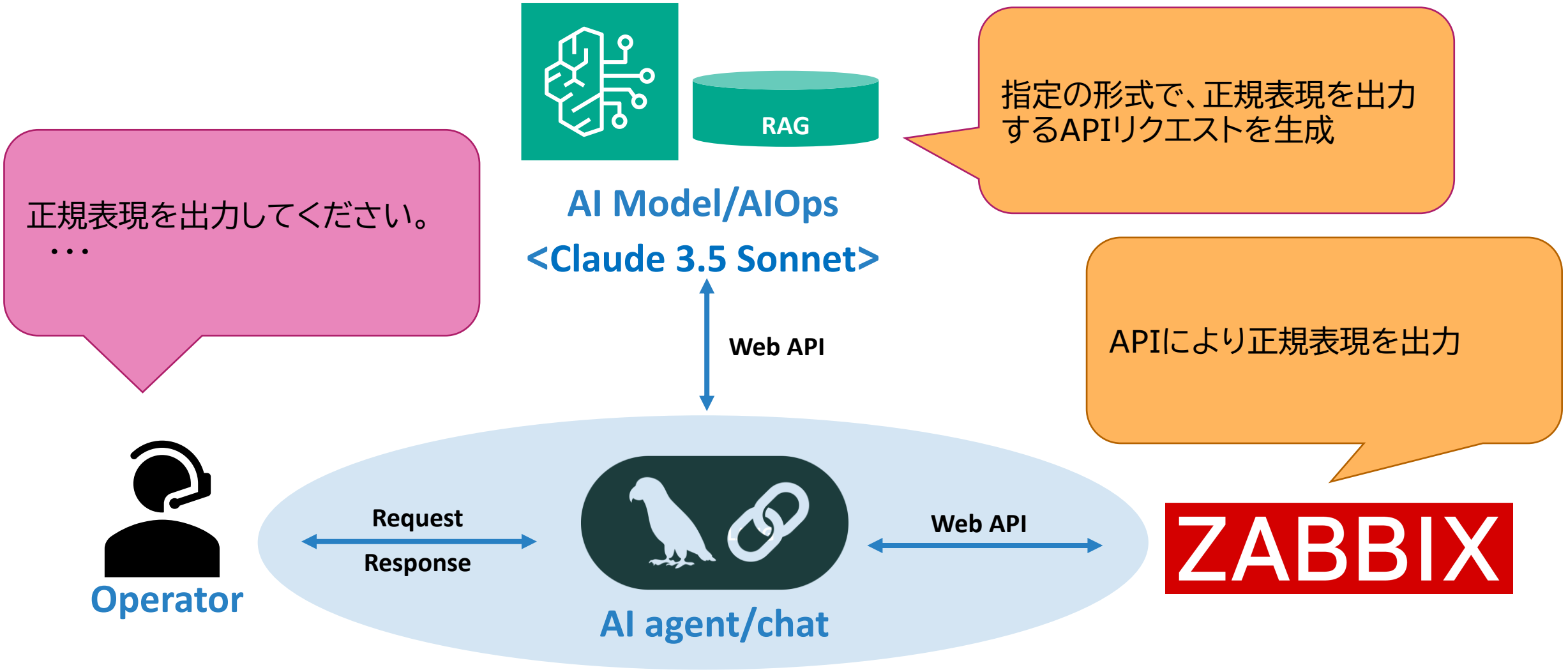
Case5：正規表現エクスポート

・Zabbix環境の正規表現

正規表現

名前	条件式
File system for Discovery	1 ⇒ ^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$ [結果が真]
File systems for discovery	1 ⇒ ^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$ [結果が真]
Network interfaces for discovery	1 ⇒ ^Software Loopback Interface [結果が偽] 2 ⇒ ^(In)?[Ll]oop[Bb]ack[0-9._]*$ [結果が偽] 3 ⇒ ^NULL[0-9.]*$ [結果が偽] 4 ⇒ ^[Ll]o[0-9.]*$ [結果が偽] 5 ⇒ ^[Ss]ystem$ [結果が偽] 6 ⇒ ^Nu[0-9.]*$ [結果が偽]
Storage devices for SNMP discovery	1 ⇒ ^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$ [結果が偽]
Windows service names for discovery	1 ⇒ ^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$ [結果が偽]
Windows service startup states for discovery	1 ⇒ ^(automatic|automatic delayed)$ [結果が真]

システム構成図



Case5：正規表現エクスポート

・正規表現をJSON形式で出力依頼

Zabbixの正規表現をJSON形式で出力依頼

Zabbixの正規表現をJSON形式で出力してください

かしこまりました。Zabbixの正規表現をJSON形式で出力するリクエストを作成いたします。この操作には、`regexp.get` APIメソッドを使用し

まず、必要な情報を確認させてください：

1. 出力したい正規表現の数に制限はありますか？
2. 特定の名前やIDの正規表現のみを取得したいですか？
3. 正規表現の詳細情報（説明文など）も含めて出力しますか？

これらの情報をお知らせいただければ、より適切なAPIリクエストを作成できます。

詳細情報をヒアリング

すべてのグローバル正規表現を取得

全てのグローバル正規表現の詳細情報を取得したいので、以下の指示でJSONリクエストを作成してください。

1. 取得する正規表現の数 制限なしで全てのグローバル正規表現を取得してください。
2. 特定の正規表現の指定 特定の名前やIDでの絞り込みは不要です。登録されている全てを出力してください。
3. 出力する詳細情報 全ての詳細情報を含めて出力してください。具体的には、正規表現アイテム（式）の内容、名前、コメント、タイプ（regexpid, name, regexp, test_string, comment, type）などの完全なデータを含めてください。

Case5：正規表現エクスポート

・APIリクエストを実行

ご指示ありがとうございます。ご要望に基づいて、全てのグローバル正規表現の詳細情報を取得するためのAPIリクエストを作成いたします。

<API-MAIN>{ "jsonrpc": "2.0", "method": "regexp.get", "params": { "output": "extend", "selectExpressions": "extend", "sortfield": "name" }, "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3", "id": 1 }</API-MAIN>
このリクエストの説明：

1. `"output": "extend"`：正規表現の全ての詳細情報（regexpid, name, test_string）を取得します。
2. `"selectExpressions": "extend"`：各正規表現に関連する式（expressions）の詳細情報も取得します。これには、式の内容（regexp）とタイ
（expression_type）が含まれます。
3. `"sortfield": "name"`：結果を名前でソートします。

このリクエストにより、Zabbixに登録されている全てのグローバル正規表現の完全な詳細情報がJSON形式で出力されます。制限は設けていな
で、登録されている全ての正規表現が取得されます。

このリクエストを実行することで、各正規表現のID、名前、テスト文字列、コメント、タイプ、および関連する式の詳細情報が取得できます

The AI asks you to execute this API request to modify Zabbix:

```
{
    "jsonrpc": "2.0",
    "method": "regexp.get",
    "params": {
        "output": "extend",
        "selectExpressions": "extend",
        "sortfield": "name"
    },
    "auth": "4fcbaf553b7a6e936d1da0043f5467fc1eb04e5c57fb93fd3aa74330da825ab3",
    "id": 1
}
```

Zabbix APIを実行

Case5：正規表現エクスポート

・レスポンス結果画面

APIからの応答を確認いたしました。Zabbixに登録されている全てのグローバル正規表現の詳細情報が正常に取得できました。応答には5つのバル正規表現が含まれています。各正規表現の概要は以下の通りです：

1. "File systems for discovery"
2. "Network interfaces for discovery"
3. "Storage devices for SNMP discovery"
4. "Windows service names for discovery"
5. "Windows service startup states for discovery"

これらの正規表現は、Zabbixのディスカバリールールや他の機能で使用されており、特定のパターンにマッチするアイテムを識別するのに役す。

各正規表現には以下の情報が含まれています：

- regexpid: 正規表現の一意のID
- name: 正規表現の名前
- test_string: テスト用の文字列
- expressions: 実際の正規表現パターンとその設定

Case5：正規表現エクスポート

・作成されたJSON形式の正規表現


```
{
  "jsonrpc" : "2.0"
  "result" : [
    0 : {
      "regexpid" : "1"
      "name" : "File systems for discovery"
      "test_string" : "ext3"
      "expressions" : [
        0 : {
          "expression" : "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$"
          "expression_type" : "3"
          "exp_delimiter" : ","
          "case_sensitive" : "0"
        }
      ]
    }
    1 : {
      "regexpid" : "2"
      "name" : "Network interfaces for discovery"
      "test_string" : "eth0"
```


Case5：正規表現エクスポート

これまで手作業で行っていた正規表現のインポートを、
AIエージェントで**JSON形式でエクスポート可能**に
なったことで、監視設定の自動化に貢献。

AIによる監視設定の自動化

Case1：ホスト登録

Case2：アイテム作成＆トリガー作成

Case3：マップ作成

Case4：指定期間における監視データの集計

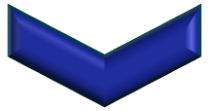
Case5：GUI未対応の正規表現エクスポート

Case6：障害発生時の初動対応リファレンス

Case6：初動対応リファレンス

課題背景

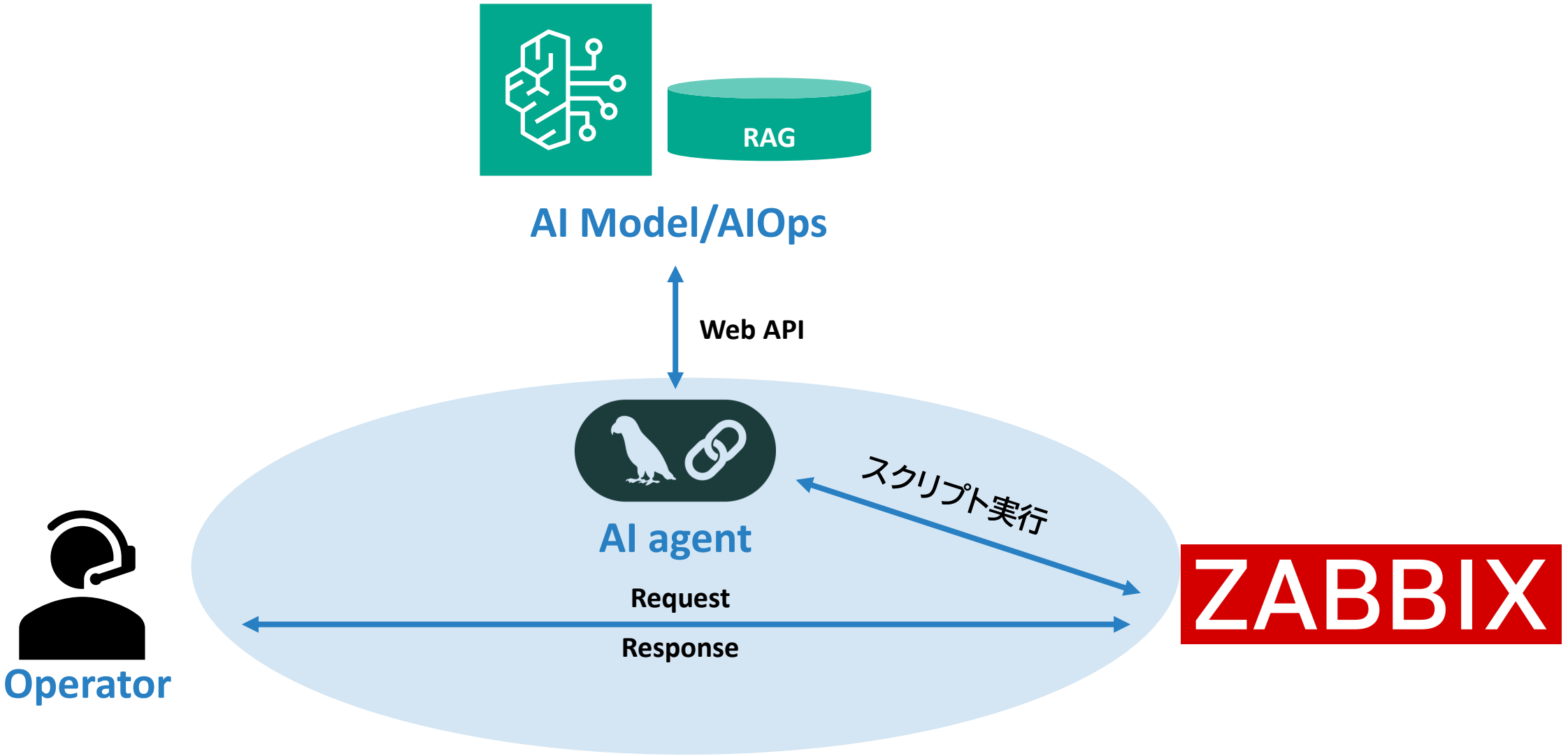
オペレーターは障害発生時、障害対応ナレッジに従い対応するが、**”発生ノード、エラー文字列、発生時刻など”**細かい条件指定があり、事象の把握と調査に時間を要すことが多い。
また調査結果の末、初動を誤るケースがある。



解決策

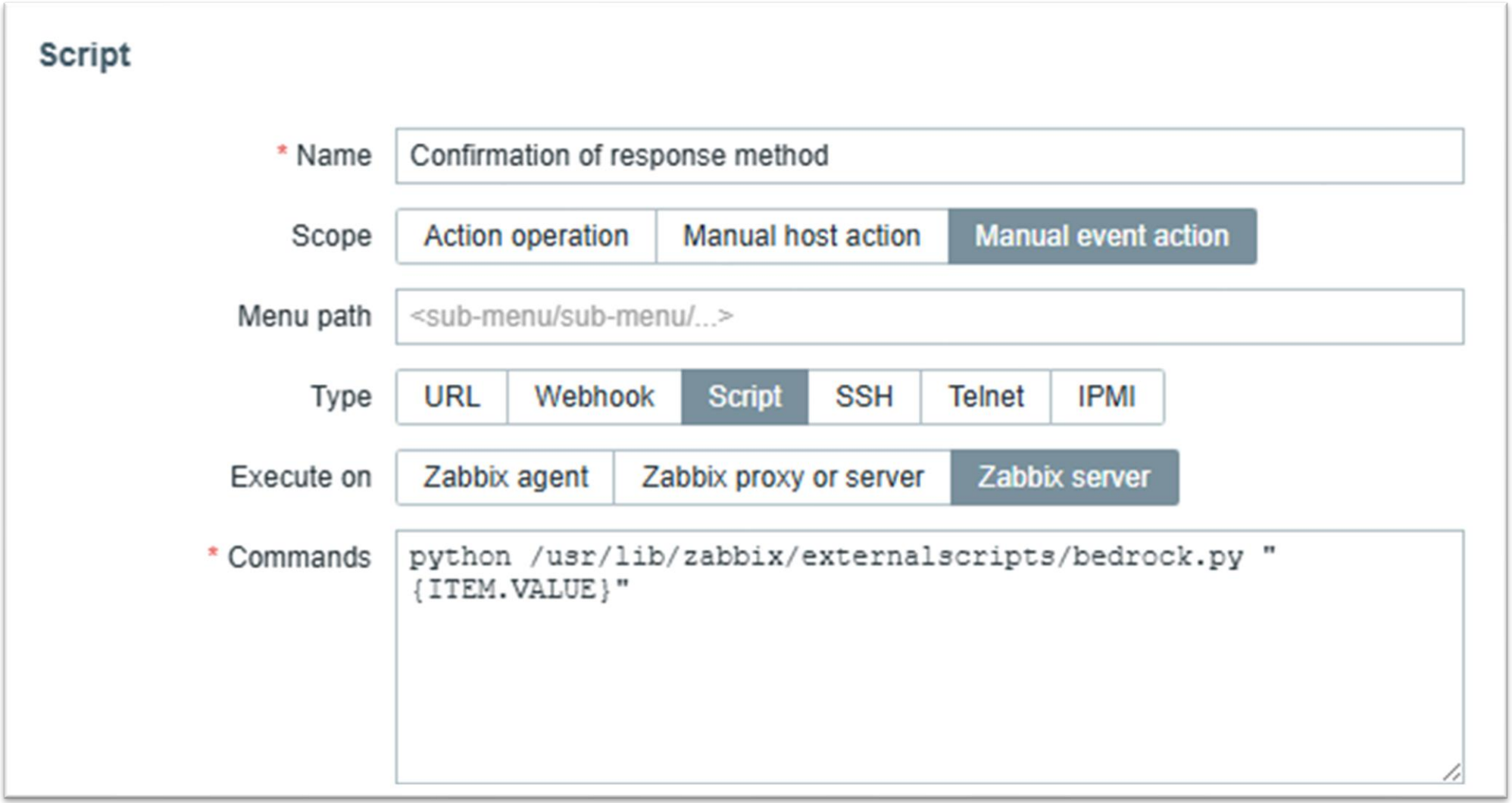
- **障害対応ナレッジを学習したRAGを作成**
- **ZabbixからRAGを検索するスクリプトを作成**

システム構成図



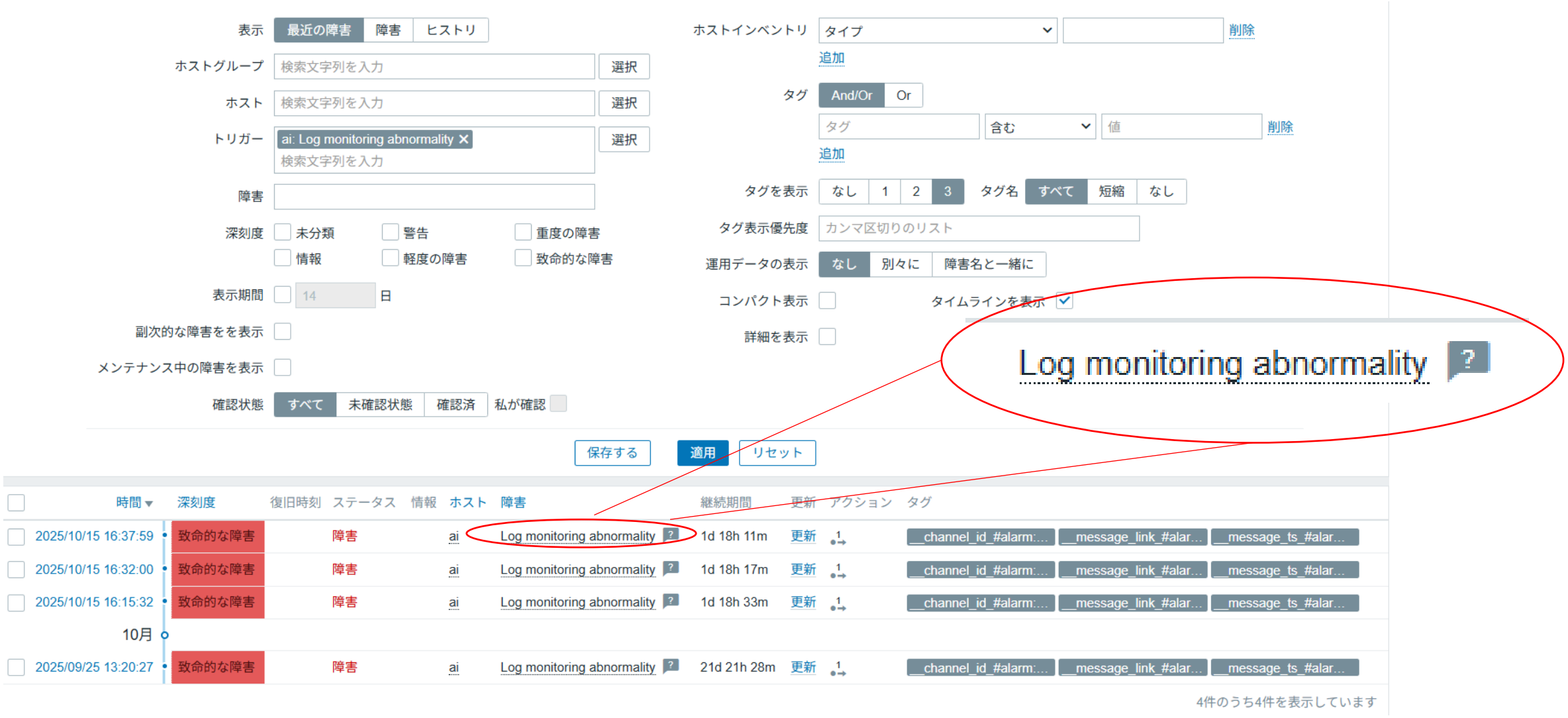
Case6：初動対応リファレンス

・エラースクリーンから直接実行できるように、スクリプトを設定



Case6：初動対応リファレンス

・監視データ＞障害　→　該当の障害をクリック

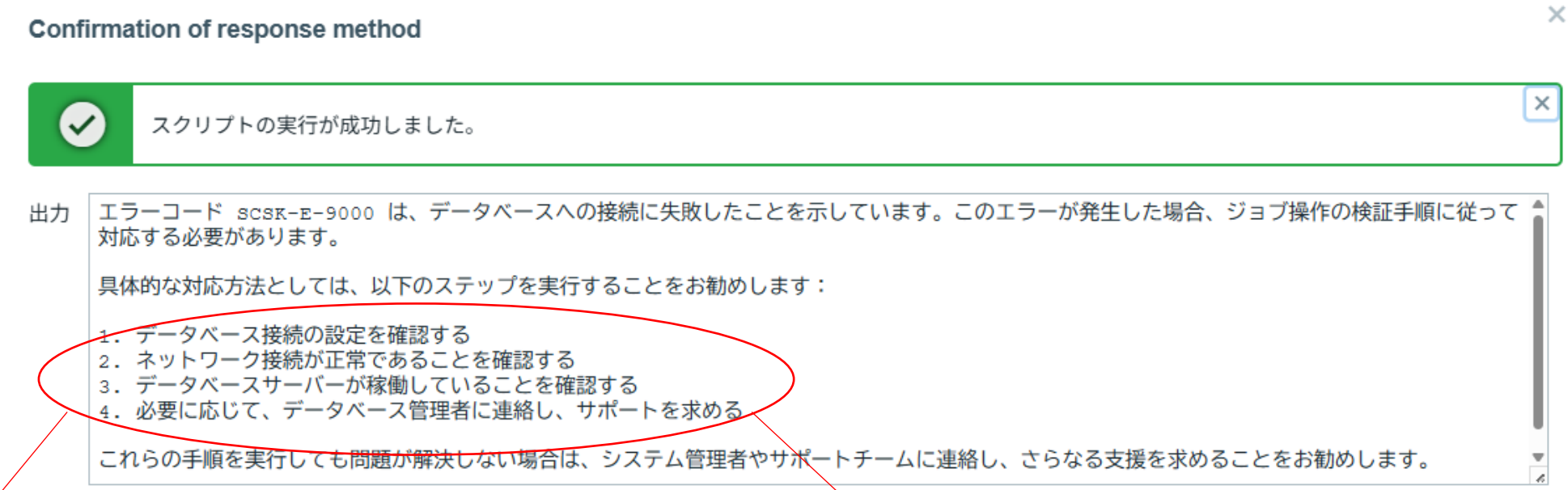


Case6：初動対応リファレンス

・Confirmation of response method をクリック

Case6：初動対応リファレンス

・スクリプト実行結果



1. データベース接続の設定を確認する
2. ネットワーク接続が正常であることを確認する
3. データベースサーバーが稼働していることを確認する
4. 必要に応じて、データベース管理者に連絡し、サポートを求める

・エラー内容、対応処理をアドバイス
エラーコード：**SCSK-E-9000**
意味：データベースへの接続に失敗しました。
対応：ジョブ操作の確認手順に従ってください。

Case6：初動対応リファレンス

初動対応リファレンスによって

・障害発生時の対応が迅速になる

＋

・障害対応の属人化の解消

AIを新たな「労働力」へ

AIによるインシデント対応支援

～ まとめ ～

機能紹介のまとめ

AIを新たな「労働力」へ

- ✅アシスタントとしては、有用
- 💡定型業務の効率化に貢献

AIによる監視設定の自動化

- ⚠️識別が必要
- 🔍制度や安全性の確認が必要

AIによるインシデント対応支援

- ✅アシスタントとしては有用
- 📌ただし、完全対応には限界あり

今後の展望

AIインフラコンシェルジュ

- AIがリソースを自動で割り当て
- 閾値の調整や対策の展開もAIが実施

予測型モニタリング

- AIが障害を予測
- 予測に基づき、交換部品の注文を自動化

グローバル知能ネットワーク

- 自社のZabbixが世界中の導入事例から学習
- 匿名パターンの共有により集合知を形成

SCSK

夢ある未来を、共に創る。