JAPAN | JUNE 20,21 2024

aws SUMMIT

AIOps で障害分析を効率化してみよう（配布用資料）

障害対応とは

ここでは以下のようなプロセスと定義する

イベント検知	障害分析	暫定対処	本格対応	再発防止
• 監視からの通知 • ユーザの連絡	• システム状況の確認 • ビジネスへの影響確認 • 原因調査	• 復旧を目的とした処置 • 復旧確認	• （必要に応じた）恒久処置	• 類似障害の発生を抑止する対策 • ノウハウ化 • 復旧手順の整備

障害対応の難しさ

学習機会が少なく、初学者を教育しづらい

ノウハウが暗黙知になりがち

一人でシステムをすべて理解するのは難しく、チーム間の協力が必須

生成 AI は障害対応を助けるか？

大規模言語モデルが得意なこと

- 生成 AI 、特に大規模言語モデル (Large Language Model, LLM)は、高い自然言語処理能力を持つことで知られている

LLM のユースケース例

文書要約	質問応答	雑談・対話	文書の 生成・校正
感情分析	情報抽出	テキスト分類	コード生成

LLM が活用できそうな障害対応のタスク例

イベント検知	障害分析	暫定対処	本格対応	再発防止
• 監視からの通知 • ユーザの連絡	• システム状況の確認 • ビジネスへの影響確認 • 原因調査	• 復旧を目的とした処置 • 復旧確認	• （必要に応じた）恒久処置	• 類似障害の発生を抑止する対策 • 障害情報の蓄積 • 復旧手順の整備
• イベント通知文言の理解や要約	• ログから原因と関連する箇所の抽出 • ログの依存関係の整理	• ソースコードの理解や改修 • 設計書などドキュメントの理解		• 報告資料や手順書、障害情報の作成

LLM が活用できそうな障害対応のタスク例

イベント検知	障害分析
• 監視からの通知 • ユーザの連絡	• システム状況の確認 • ビジネスへの影響確認 • 原因調査
• **イベント通知文言の理解や要約**	• **ログから原因と関連する箇所の抽出** • **ログの依存関係の整理**

今回は**イベント検知から障害分析**に注目する

LLM が障害分析の支援をするには？

学習機会が少なく、初学者の立ち上げが難しい

→ LLM が網羅的にログを見ることで、障害時の確認漏れを防ぐ

ノウハウが暗黙知になりがち

→ログに対する理解不足・確認漏れを LLM が補助

一人でシステムをすべて理解するのは難しく、チーム間の協力が必須

→チーム間コミュニケーションをシステムでサポート

横断検索したログやトレースを LLM に与え、原因を推測できるか試みる

LLM が障害分析をするためのポイント

障害分析の流れ

ログなどの情報収集 → 集めた情報の理解・分析 → 障害原因の仮説構築

ポイントその１
ログなど、集めるべき情報が明確かつ、自動化できる

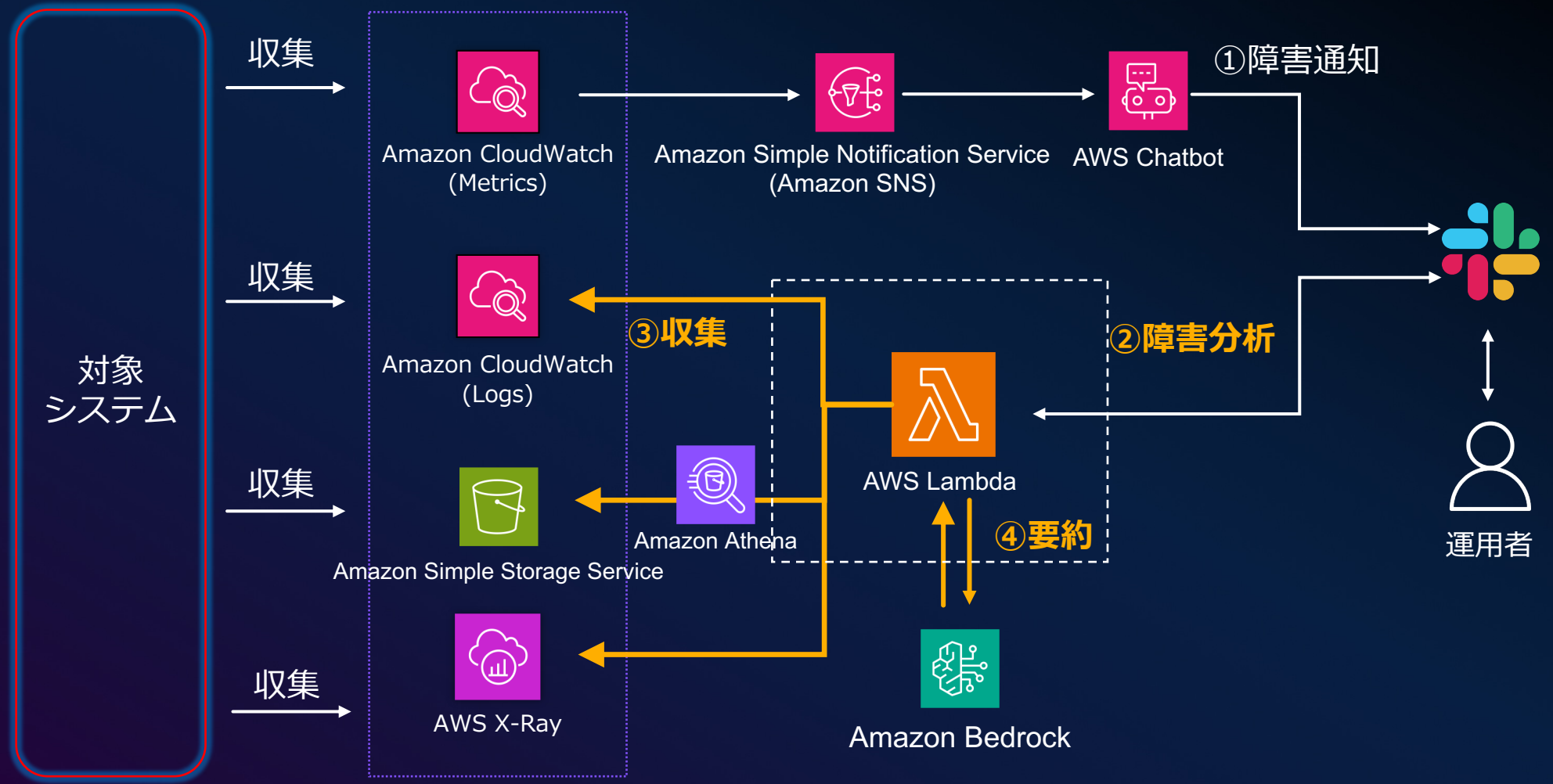
ポイントその２
LLM が理解しやすいプロンプトが構築できる

LLM の推論の流れ

コンテキスト収集 → プロンプトの構築 LLM による推論 → 推論結果出力

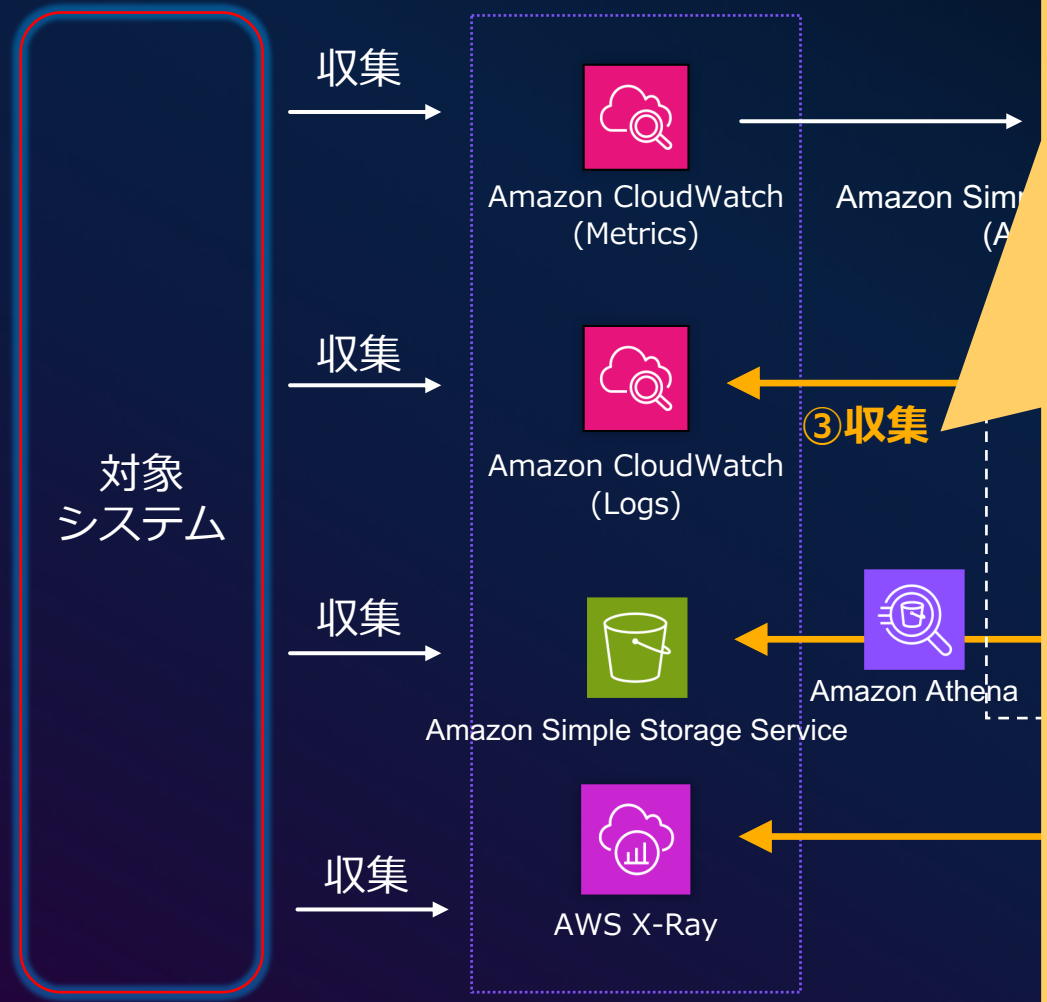
ソリューション例

ログから障害原因の仮説候補を提案する



ログから障害原因の仮説候補を提案する

ログから障害原因の仮説候補を提案する



情報収集結果例

```
[
  [
    {
      field: "@message",
      value: {
        Version: "0",
        Type: "Service",
        ServiceName: "Dev-TargetEcsApp-EcsAppServiceABC8901F-9Q81rSPeDQGh",
        ClusterName: "Dev-TargetEcsApp-EcsAppCluster7C7DF57B-vc2x3xgyPKvK",
        Timestamp: 1716900120000,
        DesiredTaskCount: 2,
        RunningTaskCount: 0,
        PendingTaskCount: 0,
        DeploymentCount: 1,
        TaskSetCount: 0,
        CloudWatchMetrics: [
          {
            Namespace: "ECS/ContainerInsights",
            Metrics: [
              { Name: "DesiredTaskCount", Unit: "Count" },
              { Name: "RunningTaskCount", Unit: "Count" },
              { Name: "PendingTaskCount", Unit: "Count" },
              { Name: "DeploymentCount", Unit: "Count" },
              { Name: "TaskSetCount", Unit: "Count" },
            ],
            Dimensions: [["ServiceName", "ClusterName"]],
          },
        ],
      },
    },
```

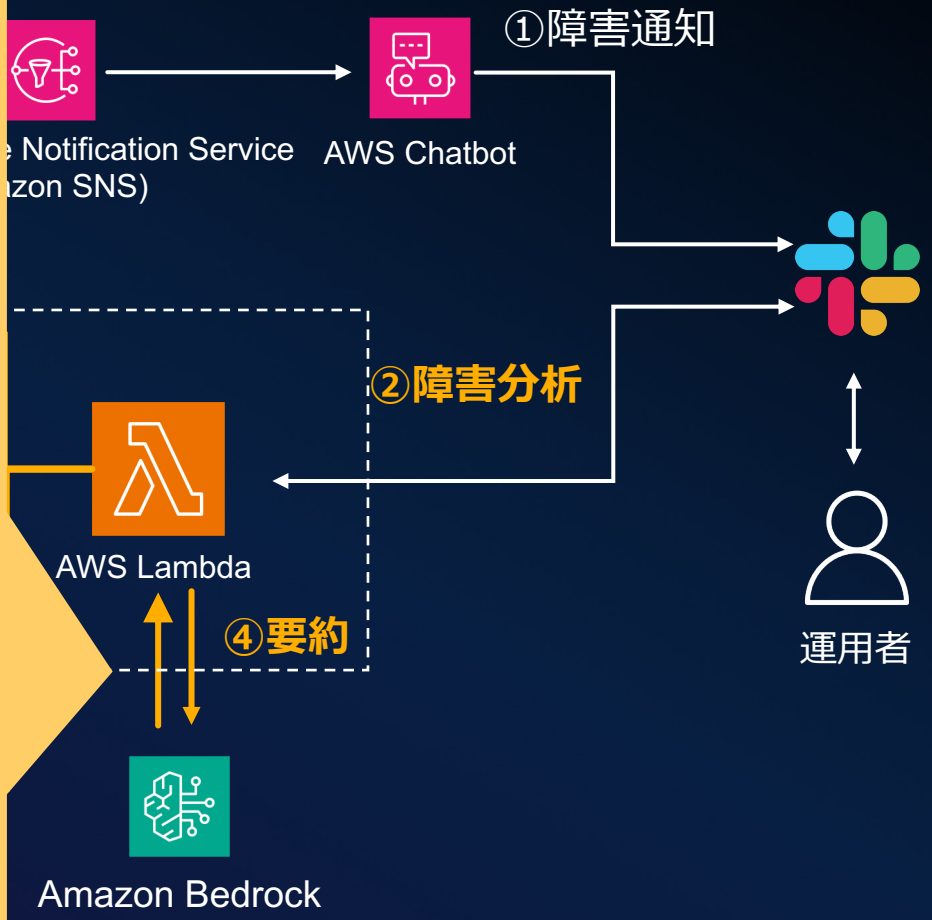
CloudWatch Logs のログ抜粋

ログから障害原因の仮説候補を提案する

プロンプト例

```
AWS上で稼働するワークロードを監視・運用するエージェントです。必ず日本語で回答してください。
あなたが担当するワークロードは、CloudFront、ALB、ECS on EC2、DynamoDBで構成されており、ECS on EC2上にSpringアプリケーションがデプロイされています。
現在、運用管理者から ${エラーの概要} という事象が発生したとの連絡がありました。
あなたは、<logs>タグに与えられたCloudFrontやALB、ECSなどのログを確認し、発生した事象の根本原因を推測してください。
根本原因を記述する際に、参考にしたログの内容についても記載し、運用管理者が実際のログを確認しやすくしてください。
<logs>
  <ApplicationLogs>${取得したアプリケーションログ}ApplicationLogs>
  <ALBAccessLogs>${取得したALBのログ}</ALBAccessLogs>
  <CloudTrailLogs>${取得したCloudTrailのログ}</CloudTrailLogs>
  <XrayTraces>${取得したトレース情報}</XrayTraces>
</logs>

発生した事象の根本原因 :
```



ただし、回答をそのまま信頼できるわけではない

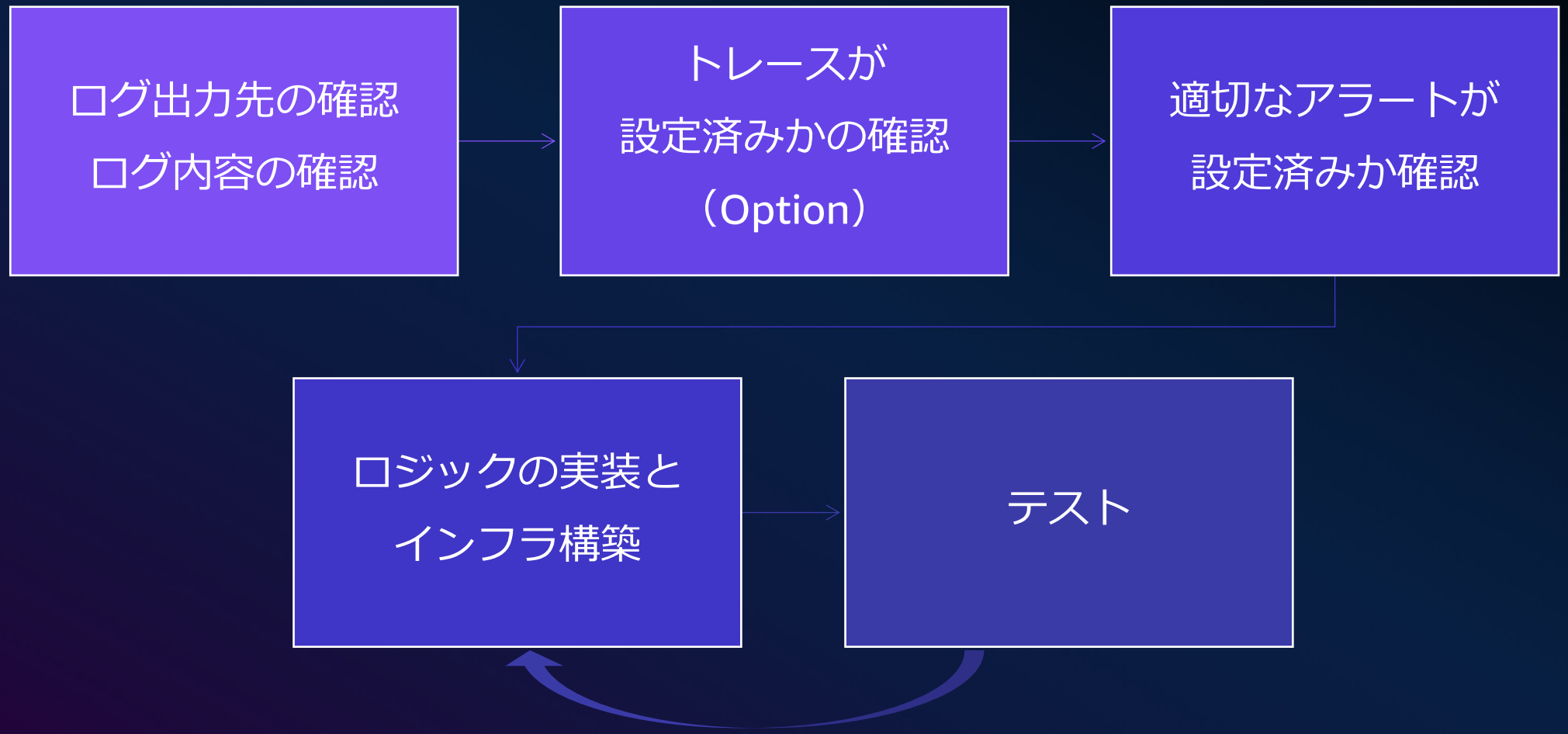
- 生成 AI のハルシネーションにより、**誤った内容を含む可能性がある**
  - もし、誤った内容を受け入れてしまうと、重大なミスになりかねない
- 回答はあくまで参考情報として、**自分で確認することが求められる**

例えば、
LLM の回答抜粋：
「 ECS タスクの起動失敗と、ALB / ECS 間の通信障害が主なエラー原因と考えられます。DynamoDB のレイテンシの改善も性能向上につながると思われます。 」

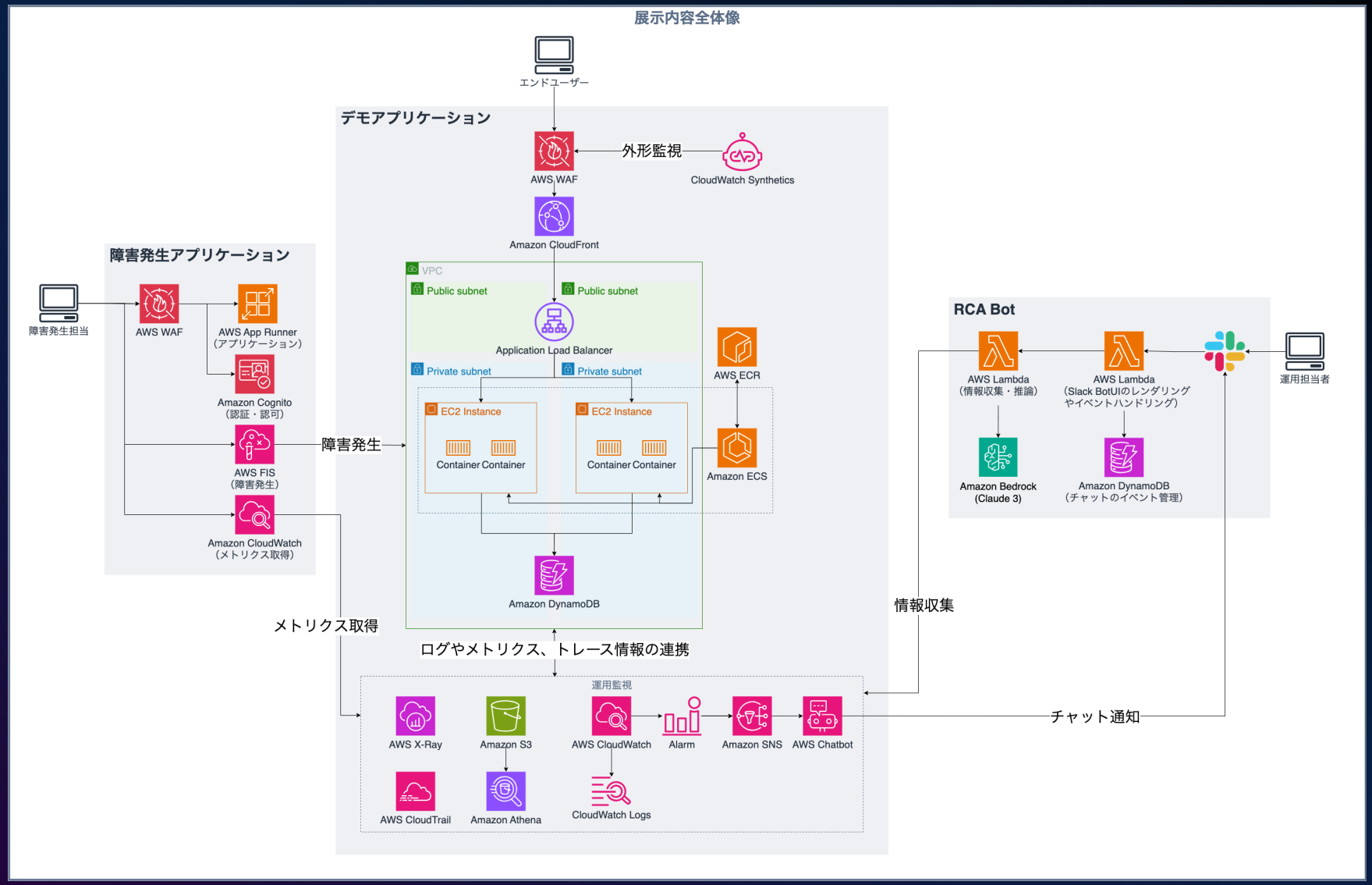
➡ **平常時に問題はなく、レイテンシの発生は障害起因。また、性能向上につながるかは不明**

開発にむけて

開発のステップ



デモ展示のアーキテクチャ全体像



開発 Tips 1/2

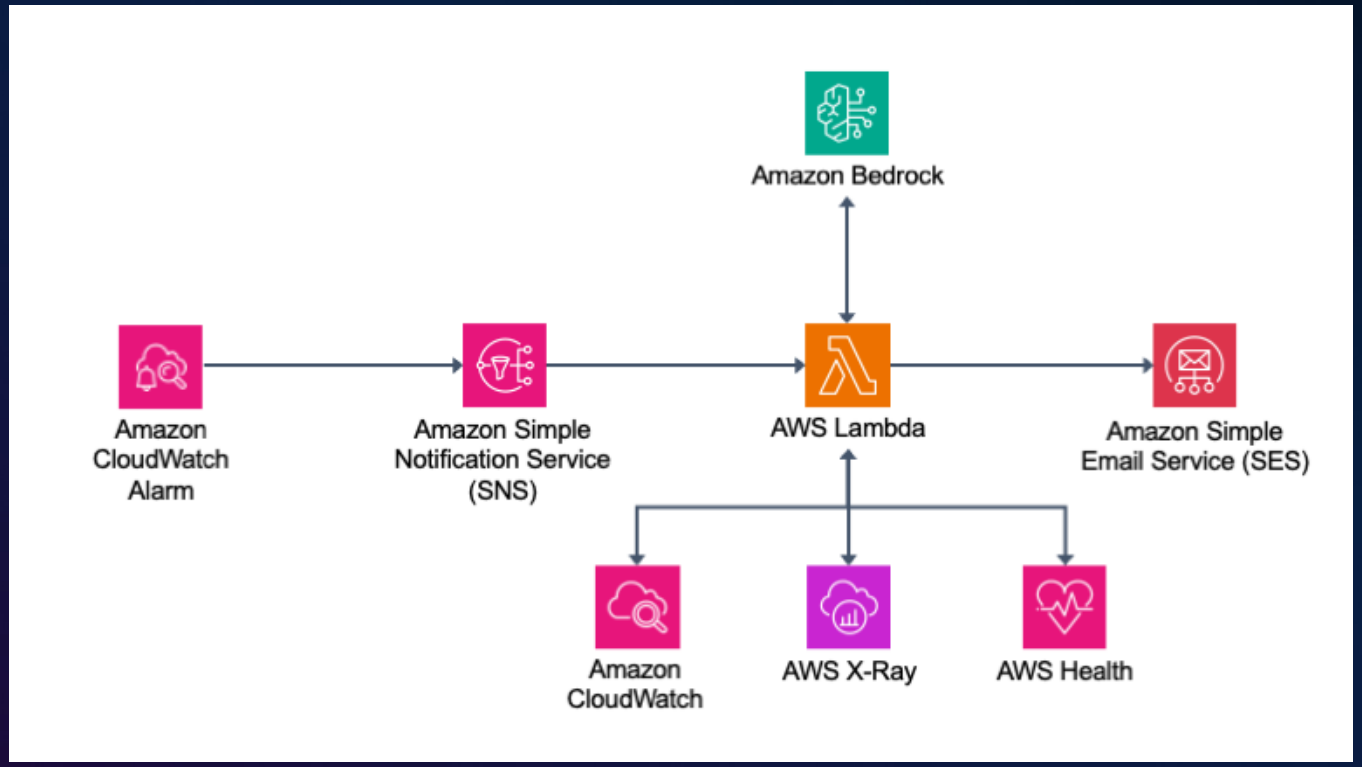
- **ログの設計が重要**
  - ログ設計が正しく行われていないと、LLM も理解できない
  - 例：誤ったログレベルで出力している、適切なログメッセージになっていない
- **コンテキストを充実させるために、トレースの活用を考える**
  - トレースはあるリクエストがどこでエラーになったかたどりやすくする情報
  - コンテキストに追加すると精度が上がる可能性がある
- **障害分析のトリガーとなるアラームは必須**
  - イベントの発生時間から、必要なログが発生している時間を推定する
  - 今回のデモでは、入力する IF を用意したが、自動化することも可能

開発 Tips 2/2

- **コンテキストに含める情報の種類は様々。試行錯誤が必要**
  - アプリケーションログ、CloudFront、ALB が出力するアクセスログ、CloudTrail のログ、セキュリティサービスの検知結果など利用できるものは多い
  - 今回のデモでは、コンテナのアプリケーションログ、ALB のアクセスログ、CloudTrail のログ、Xray のトレースを利用した
  - AWS は API があるので、各種ログの収集も SDK で容易に実装可能
- **コンテキストの量が長大になるので、大きなコンテキストウィンドウを持つ LLM の利用を推奨**
  - 例えば、Claude モデルは200,000 トークンのコンテキストウィンドウを持つ
  - Amazon Bedrock を利用すれば、大きなコンテキストウィンドウの LLM が出ても切り替えやすい

類似ソリューション

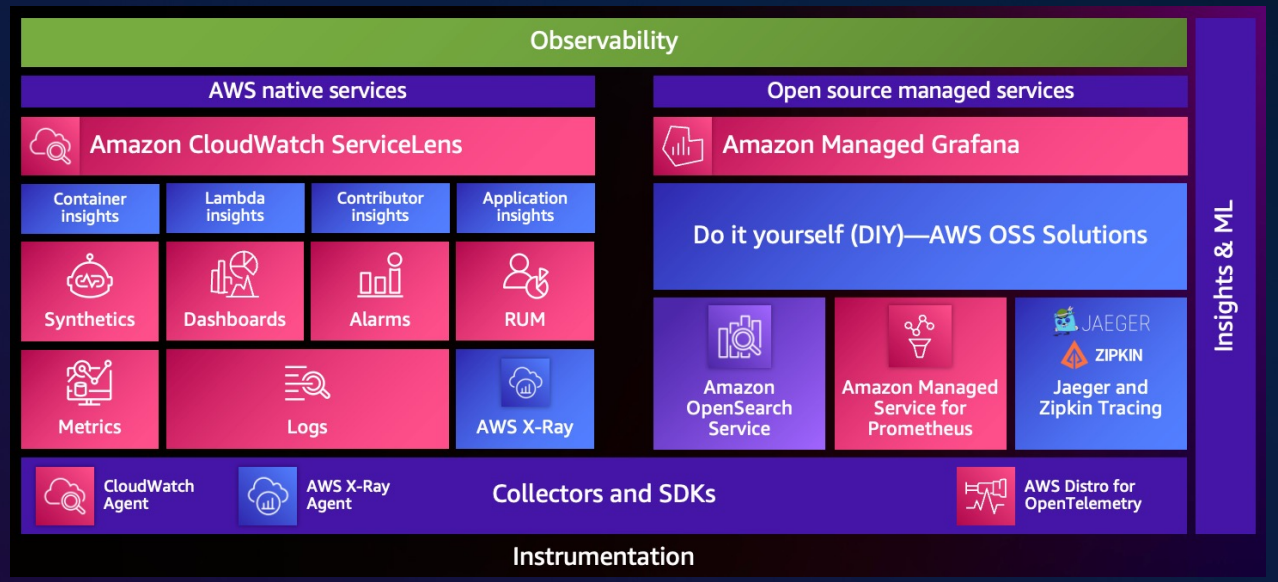
- アラートが発生すると自動的にメールで障害分析結果を送る仕組み
- GitHub に公開されているので、これをベースに開発を始めやすい



Blog: https://aws.amazon.com/jp/blogs/mt/alarm-context-tool/
Github: https://github.com/aws-samples/alarm-context-tool

Observability に取り組みたい方へ

- One Observability Workshop がおすすめ
- AWS における Observability を包括的に学べるワークショップ
  - ワークショップの後、自身の運用するワークロードへ必要なものを適用し、その後 AIOps に取り組んでいくと、LLM に渡す情報を集めやすい



https://catalog.us-east-1.prod.workshops.aws/workshops/31676d37-bbe9-4992-9cd1-ceae13c5116c/ja-JP

障害分析以外のプロセス

LLM が活用できそうな障害対応のタスク例

残りのプロセスに生成 AI が適用できそうな領域はあるか？

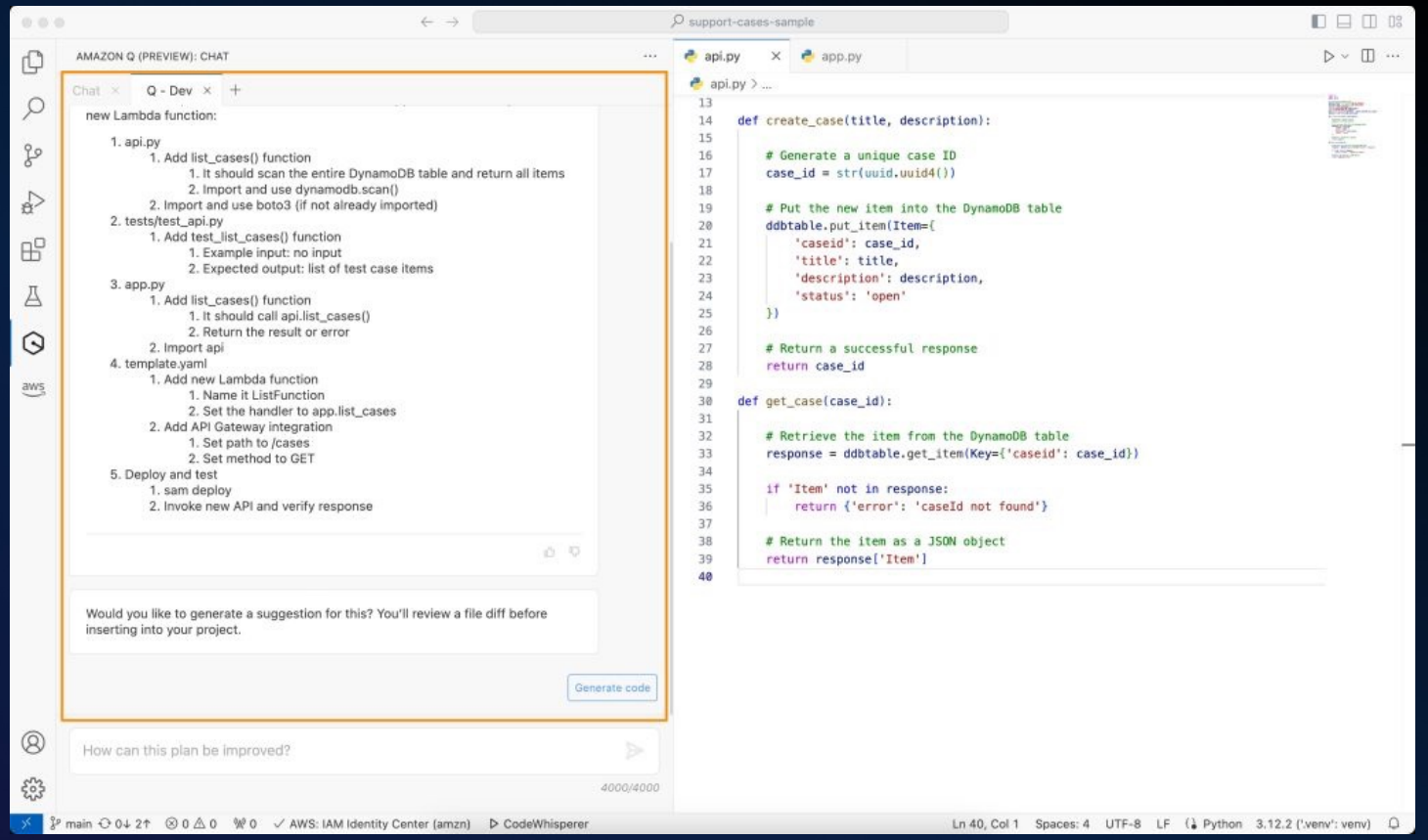
暫定対処	本格対応	再発防止
• 復旧を目的とした処置 • 復旧確認	• （必要に応じた）恒久処置	• 類似障害の発生を抑止する対策 • 障害情報の蓄積 • 復旧手順の整備

- ソースコードの理解や改修
- 設計書などドキュメントの理解

- 報告資料や手順書、障害情報の作成

ソースコードの理解支援や開発支援

- **Amazon Q Developer**
  - AWS 上のチャットアシスタント、IDE の拡張機能、CLI といったインタフェースを持つ
  - IDE の拡張機能は、コードの説明や変更提案、セキュリティスキャンなどを提供
  - 2024/6/6 時点で英語のみ対応



https://aws.amazon.com/jp/blogs/news/amazon-q-developer-now-generally-available-includes-new-capabilities-to-reimagine-developer-experience/

文書の校正など

- 文書校正は LLM の得意なタスクの一つです
- さらには、文書フォーマットや記述スタイルをコンテキストとして与えることで、適した形で出力することも期待できます

まとめ

まとめ

- 生成 AI を活用することで、**障害分析を効率化できる”可能性”がある**
  - ログなどの整理や、原因調査の手掛かり探しには、役立つ可能性がある
  - 生成AIはハルシネーションを起こすため、裏どりは必要になる
- 現状では**補助的に利用していく**のが望ましい

Thank you!

鈴木 陽三

suzukyz@amazon.co.jp