ENOG 87 MEETING

AWS でできる耐障害性の向上とカオスエンジニアリングのすすめ

菊池之裕

Sr. Solutions Architect Network Specialist
Amazon Web Services Japan G.K.

自己紹介

- 名前：菊池　之裕(きくち　ゆきひろ)
- シニアソリューションアーキテクト ネットワークスペシャリスト
- ロール：Network系サービスについてのご支援
- 経歴: ISP,IXP,VPN運用、開発を経てネットワーク機器、仮想ルータ販売会社のプリセールス、プロダクトSEからAWSへ（8年目）
- 好きな AWS サービス:AWS Transit Gateway,AWS Direct Connect, AWS Marketplace

Agenda

- AWS概要
- グレー障害について
- グレー障害の検出
- グレー障害のAWSでの対応例
- カオスエンジニアリングと障害の訓練について
- まとめ

今日持って帰っていただきたいこと

- グレー障害とは何かを理解する
- 耐障害性をあげる方法について理解する
- カオスエンジニアリングと、訓練の方法について理解する

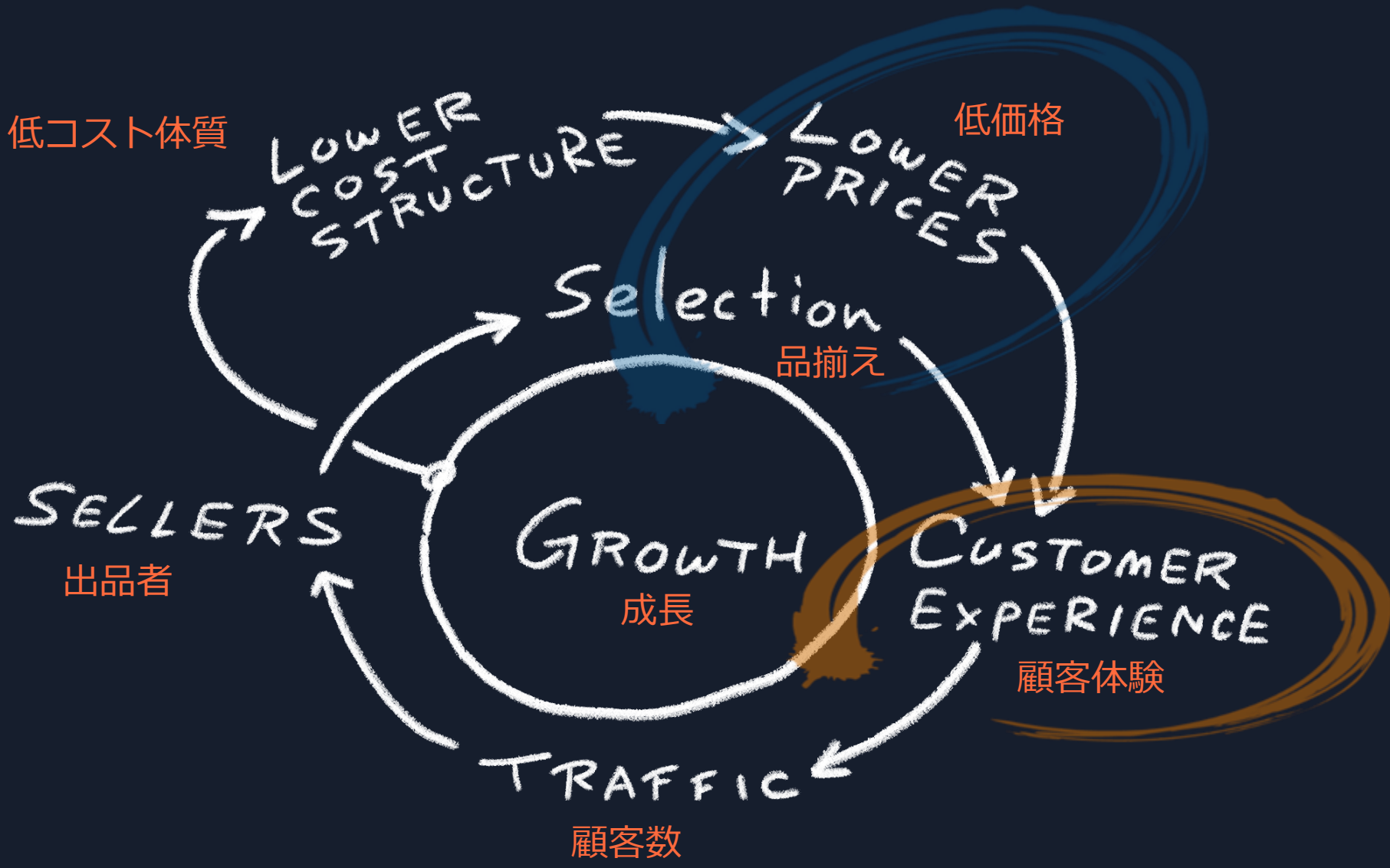
AWSとは何か？

1. Amazonの課題から生まれたAWS
2. AWSの特徴

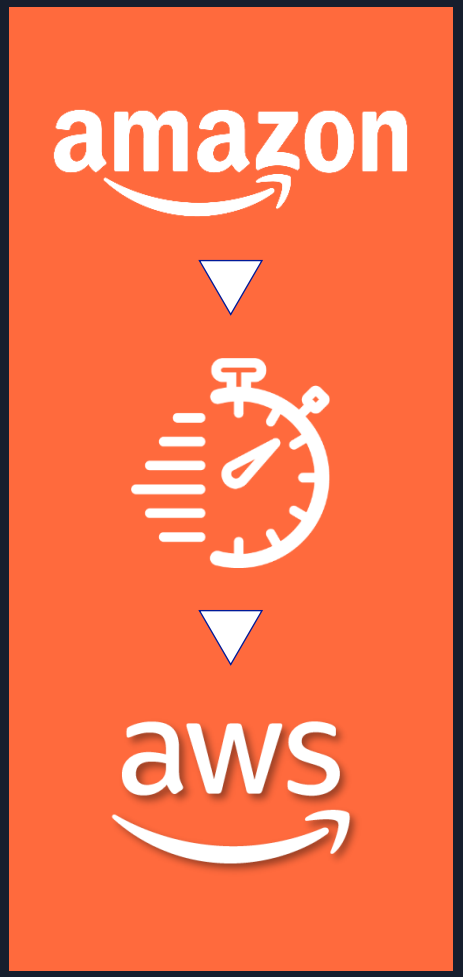
Our mission

地球上で、もっともお客様を大切にする企業であること

Amazon のビジネスモデル



Amazon の IT が抱えた課題の克服が AWS に



課題

- ビジネスの**進化のスピードに基盤が追従しない**
- お客様からのフィードバック（市場のニーズ）をサービスの充実化に繋げられない

対応策

- 多目的・多機能の巨大なアプリケーションを
  機能ごとにサービスとして細分化（マイクロサービス化）
- **サービス間の依存関係を排除**し、個々に拡張・進化可能
- API ベースで運用できる基盤に刷新（自動化）

この基盤を元にして、
2006年に一般向けサービスとして
Amazon Web Services が事業を開始

クラウドの真価とは価値創造に集中できること

スモールスタートで
すぐに使い始められる

必要なときに必要なだけ
使うことが可能

アイディアから
実装までの時間を短縮

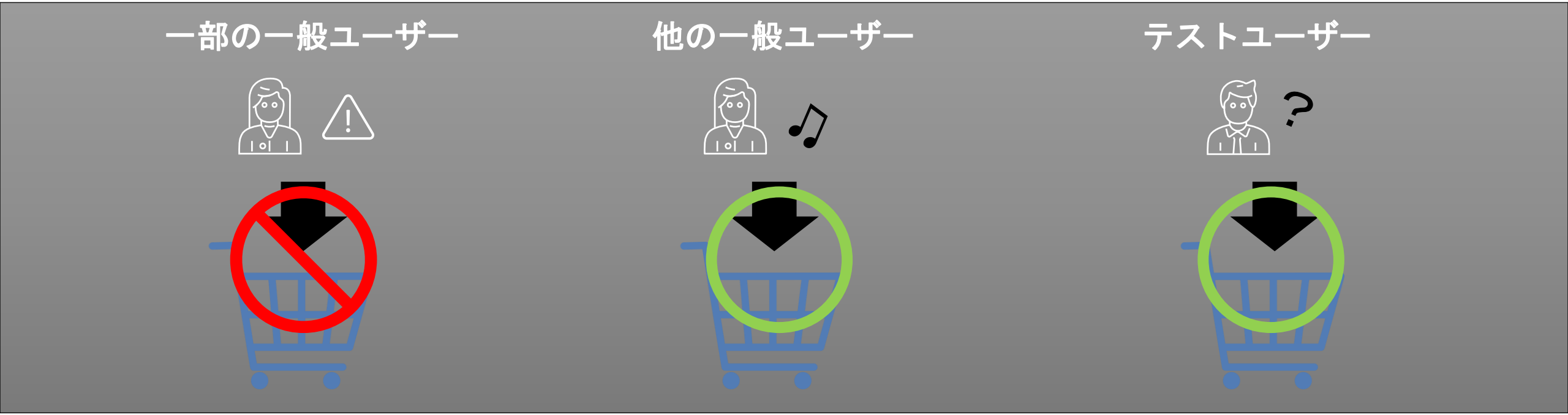
200 を超えるクラウドサービスで あらゆるワークロードをサポート

- コンピューティング
- モバイル
- AR と VR
- エンドユーザーコンピューティング
- ストレージ
- データベース
- ネットワークとコンテンツ配信
- AWS コスト管理
- 機械学習と AI
- IoT
- ロボット工学
- ビジネスアプリケーション
- メディアサービス
- 分析
- マネジメントとガバナンス
- 開発者用ツール
- サーバーレス
- アプリケーション統合
- Game Tech
- 量子テクノロジー
- カスタマーイネーブルメント
- 移行と転送
- ブロックチェーン
- セキュリティ・ID・コンプライアンス
- 人工衛星
- コンテナ

グレー障害の定義と発生シナリオ

気づきにくいユーザー体験の低下

- ある EC サイトで、一部のユーザーのみカートの操作が失敗
- 商品は表示されているのに、カートに入れようとすると時々エラーになる
- テストユーザーは正常であり、運用チームによる動作確認では問題を再現できない

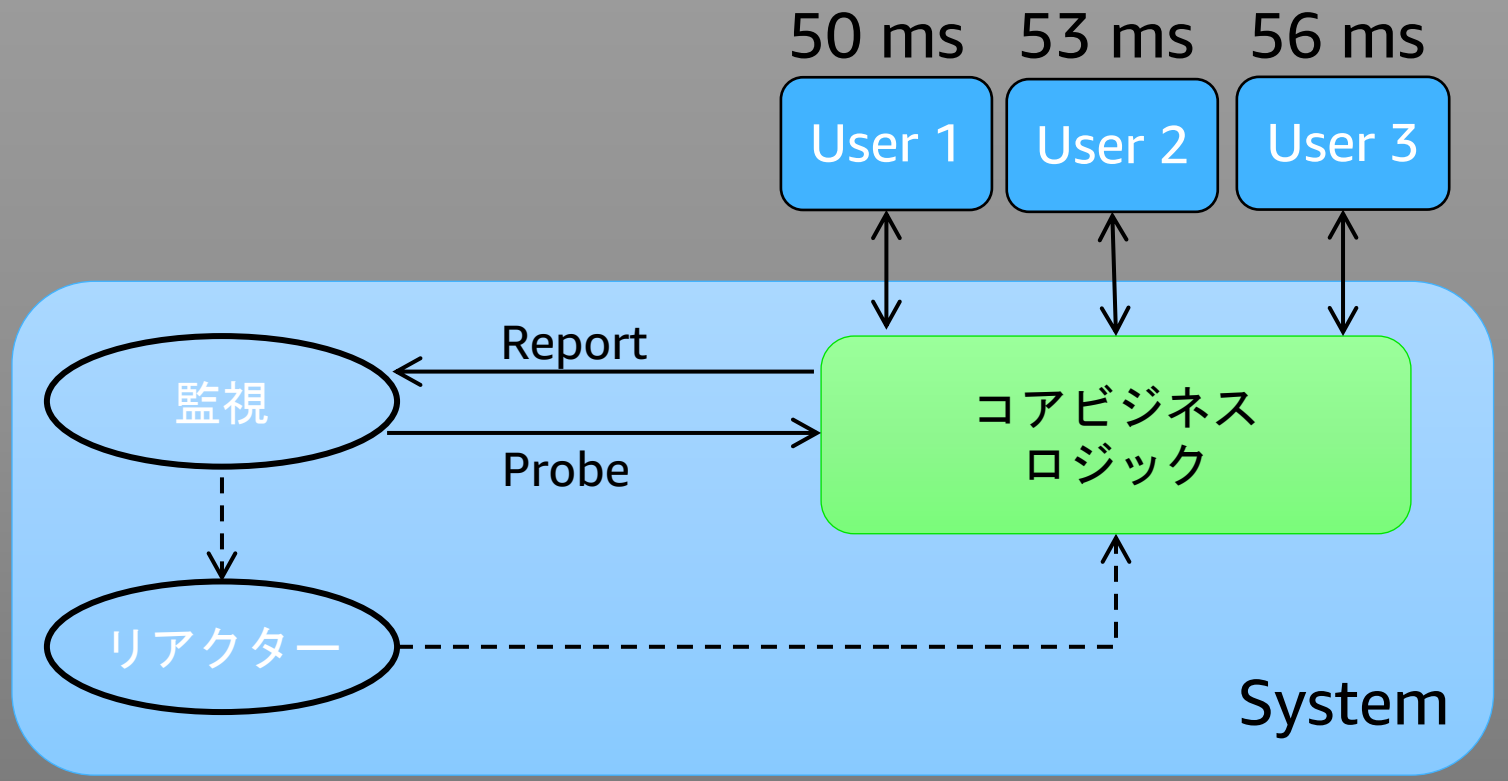


グレー障害

完全な停止ではないが、サービスの質が低下している状態

Average latency :
53 ms

Latency threshold :
60 ms

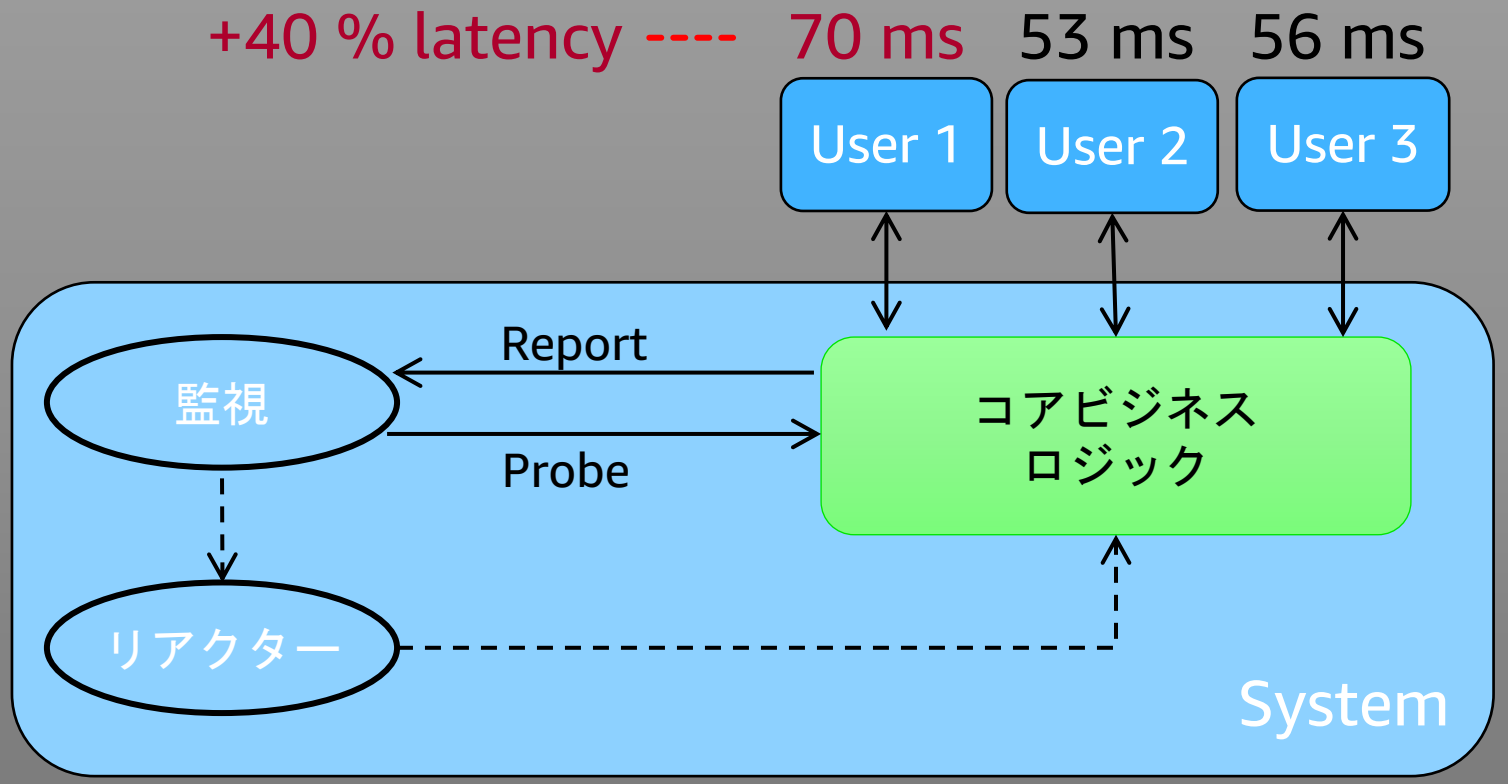


グレー障害

完全な停止ではないが、サービスの質が低下している状態

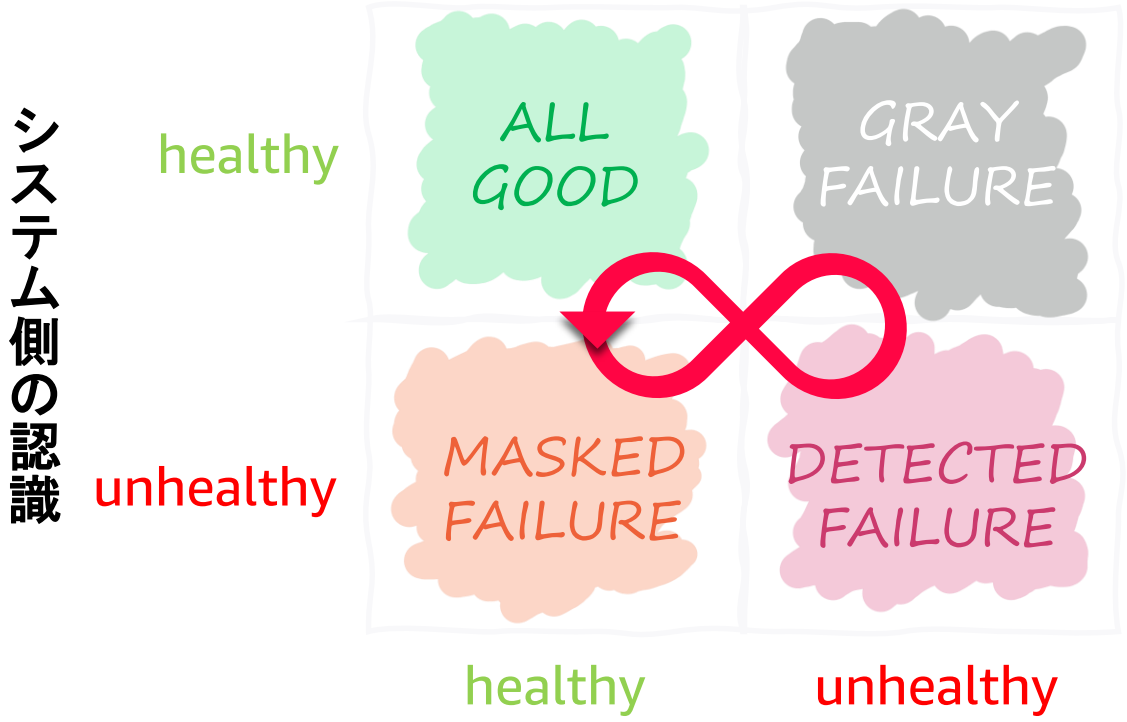
Average latency :
59.66 ms

Latency threshold :
60 ms



視点別の **Observability**

システムの状態は、誰がどこから見ているかによって、異なって見えることがある



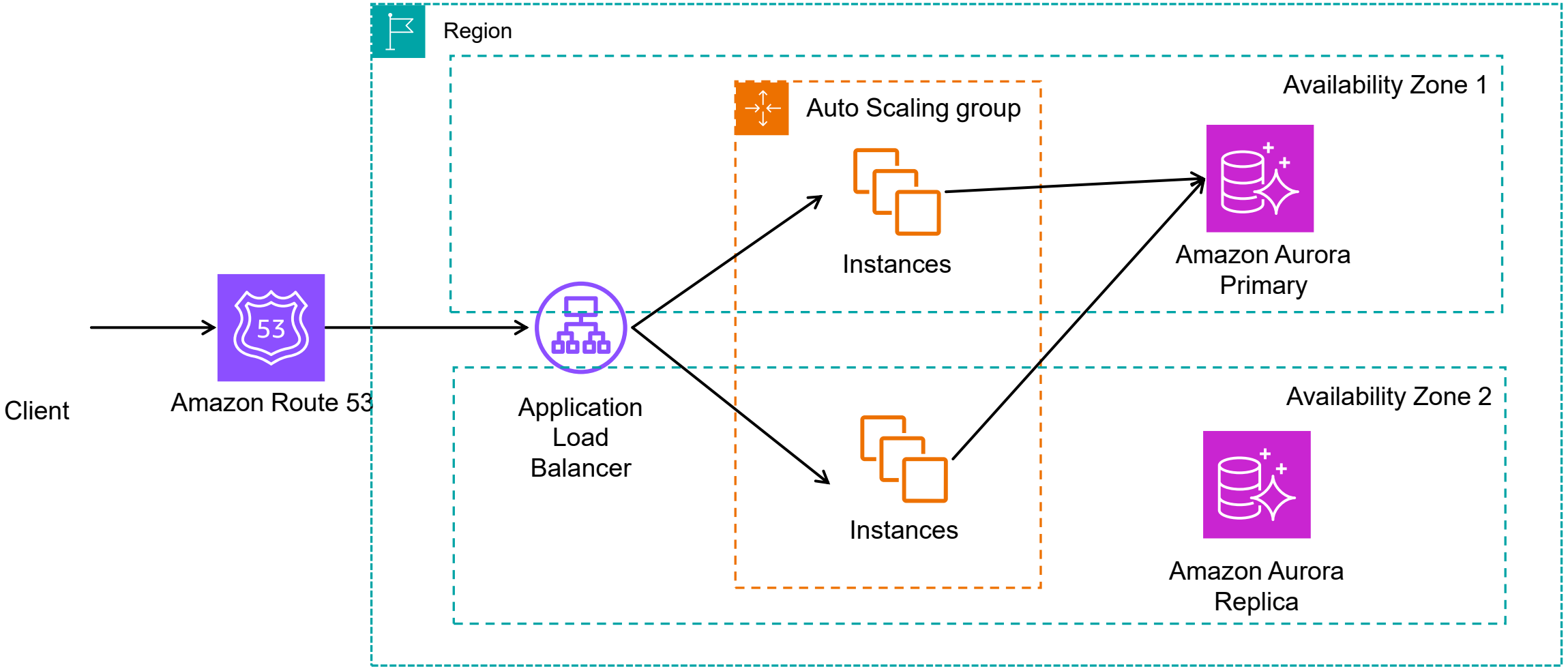
ALL GOOD：
すべての視点で正常な状態

GRAY FAILURE（グレー障害）：
システム全体としては正常に見えるが、
一部のユーザーやアプリケーションで問題が発生

DETECTED FAILURE：
システムとして問題を検知できている状態

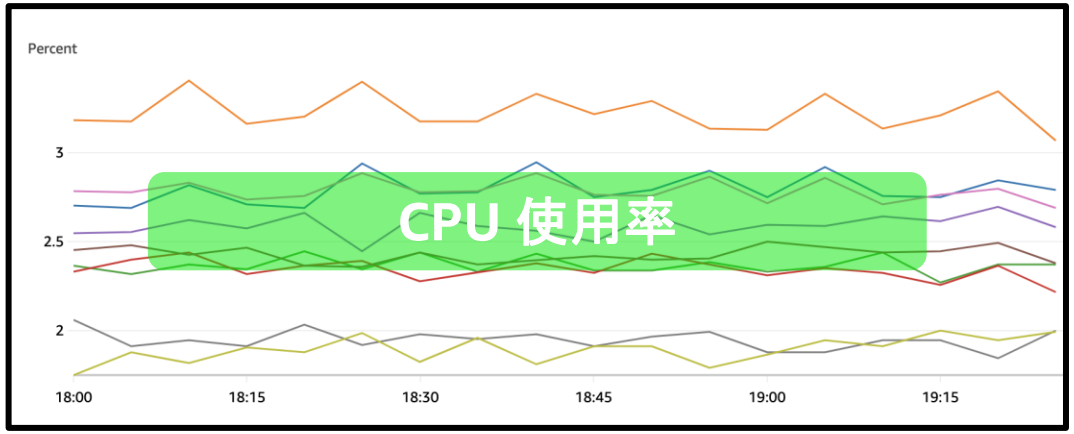
MASKED FAILURE：
問題はあるがユーザーは検知できていない状態

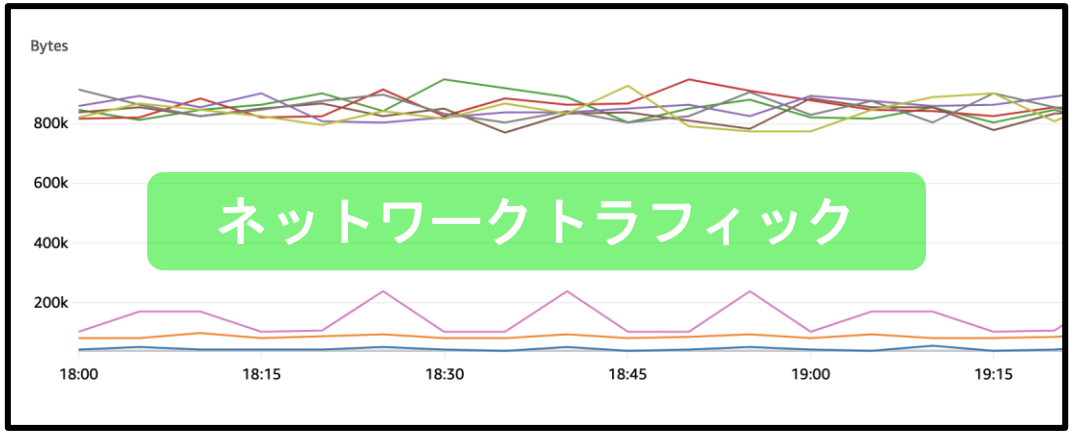
EC サイトのアーキテクチャー

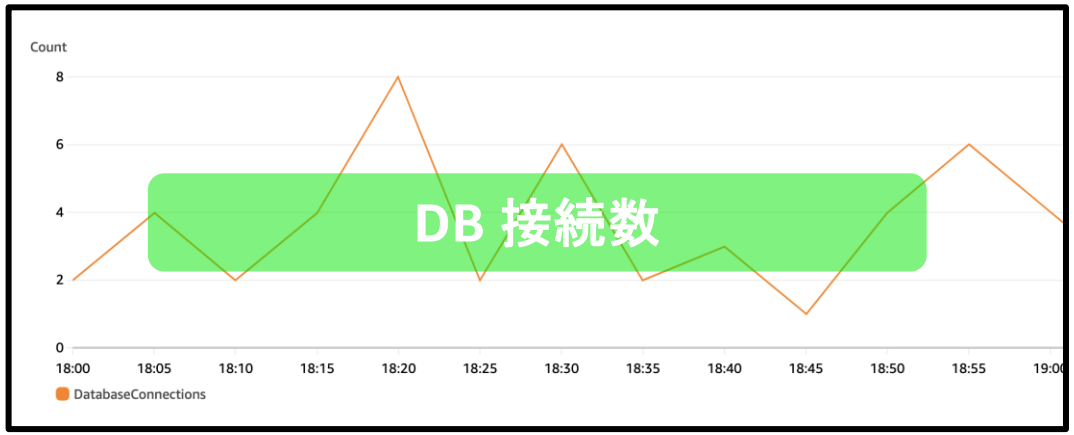


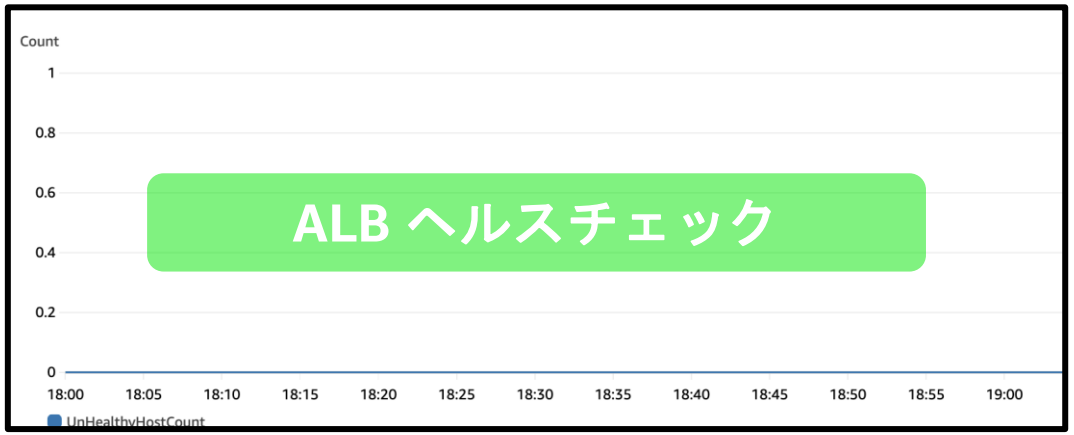
システムメトリクスは正常

監視メトリクスに問題はなく、システムは正常に動いているように見える



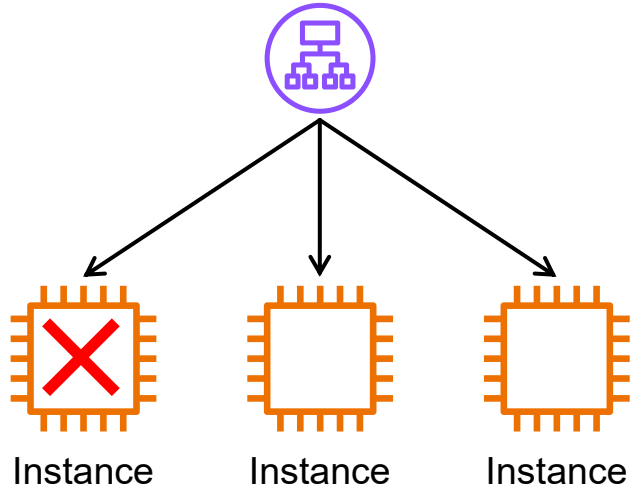




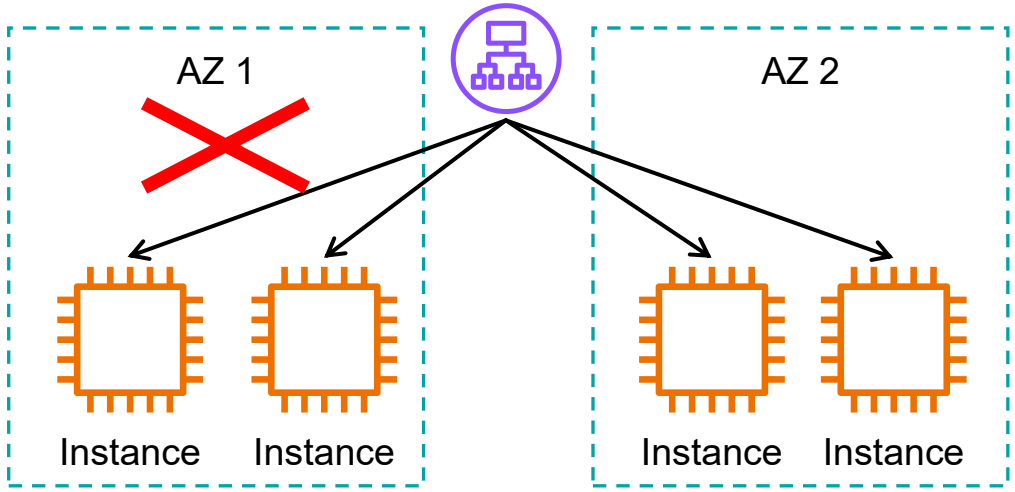


グレー障害の代表的なパターン

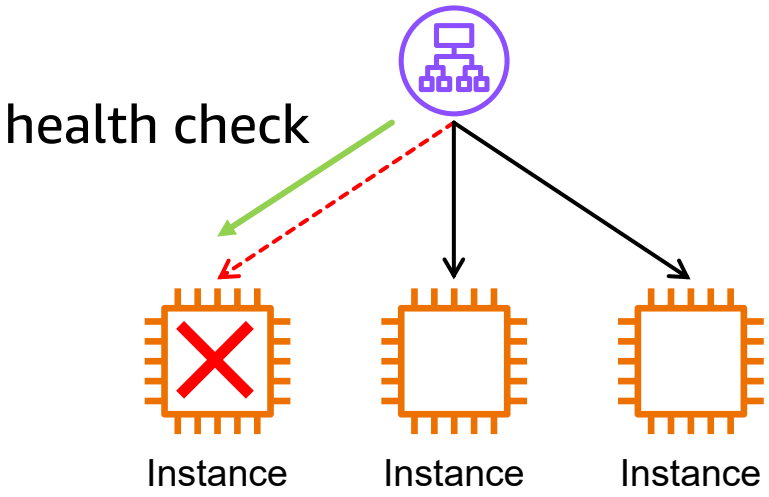
特定インスタンスでの部分的な障害



AZ レベルでの品質低下

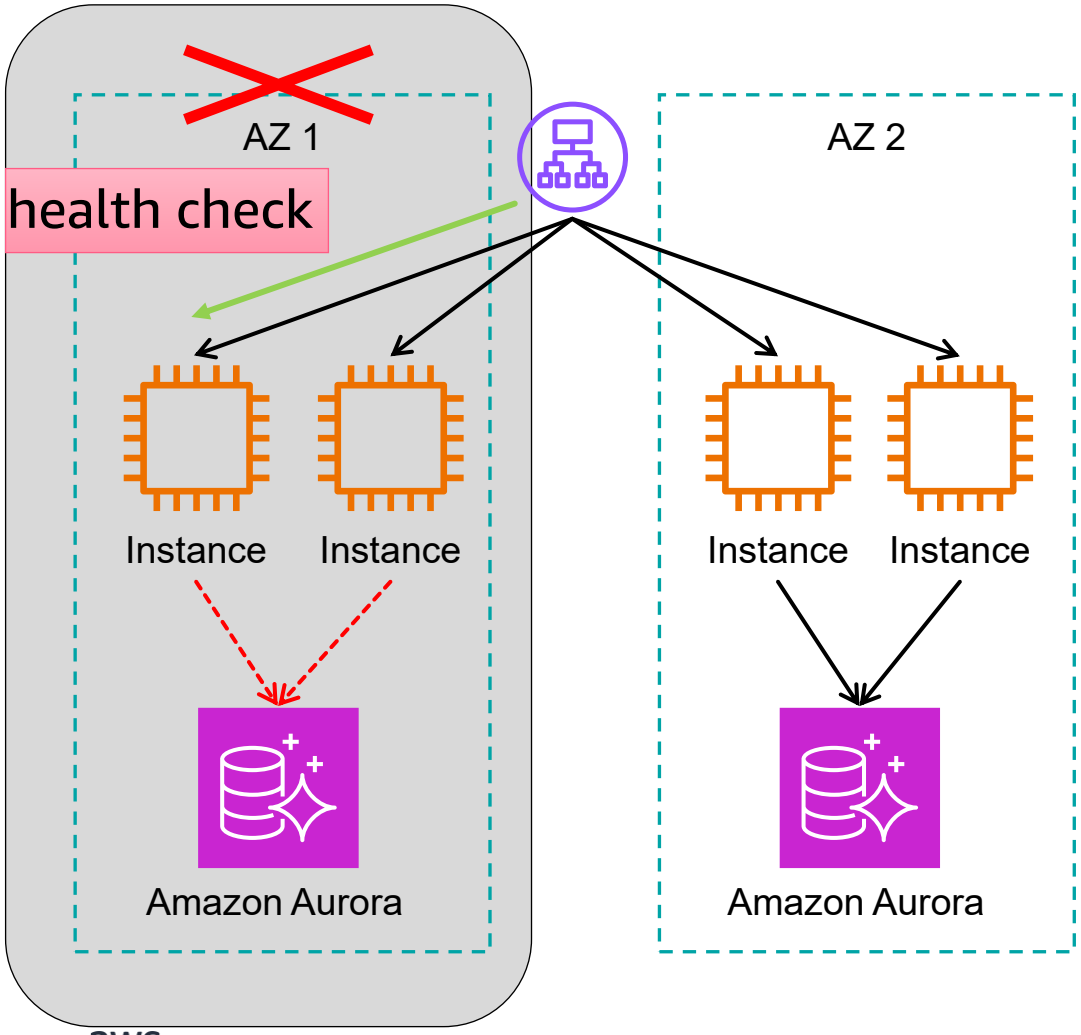


例 1：特定インスタンスで発生する部分的な障害



- Auto Scaling Group 内の複数インスタンスのうち、1 台で断続的な遅延が発生
- 他のインスタンスは正常に動作している
- 問題のあるインスタンスでも **ALB のヘルスチェックは通過**している
- 失敗しているのは一部のリクエストのみ
  - 商品一覧表示：〇
  - カート追加：×
  - 注文処理：〇

例 2 : AZ レベルでの品質低下



- 特定 AZ 内のすべてのインスタンスで問題が発生している
- 他の AZ では正常に動作している
  - ユーザーはリクエストの半数が遅延やエラー
- 特定の AZ で発生する EC2 のネットワーク品質低下により、その AZ で稼働するインスタンスからデータベースへの接続が断続的に遅延
- **ALB のヘルスチェックは通過**

ヘルスチェックでは検出できないのか？

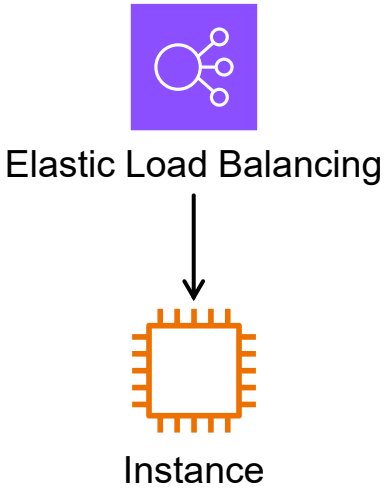
シャローヘルスチェックとディープヘルスチェック

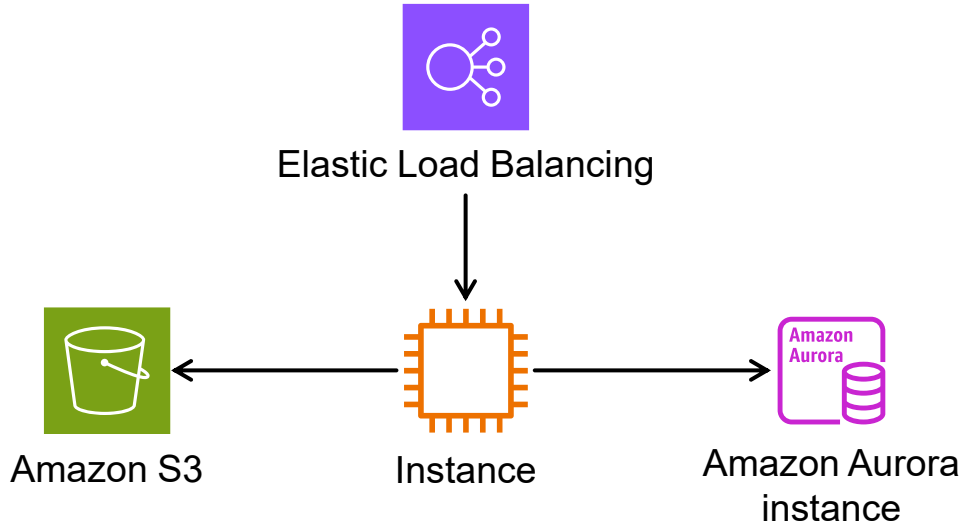
シャロー（浅い）ヘルスチェック

- サーバーとアプリケーションの状態を確認
- TCP 接続確認、HTTP 200 応答確認、EC2 ステータスチェック
- **外部依存関係はチェックしない**

ディープ（深い）ヘルスチェック

- サーバーの状態に加え、外部依存関係も確認
- DB へのクエリ実行や S3 アクセスなども含む
- エンドツーエンドでの機能確認が可能
- **依存関係の問題について過剰に検知**

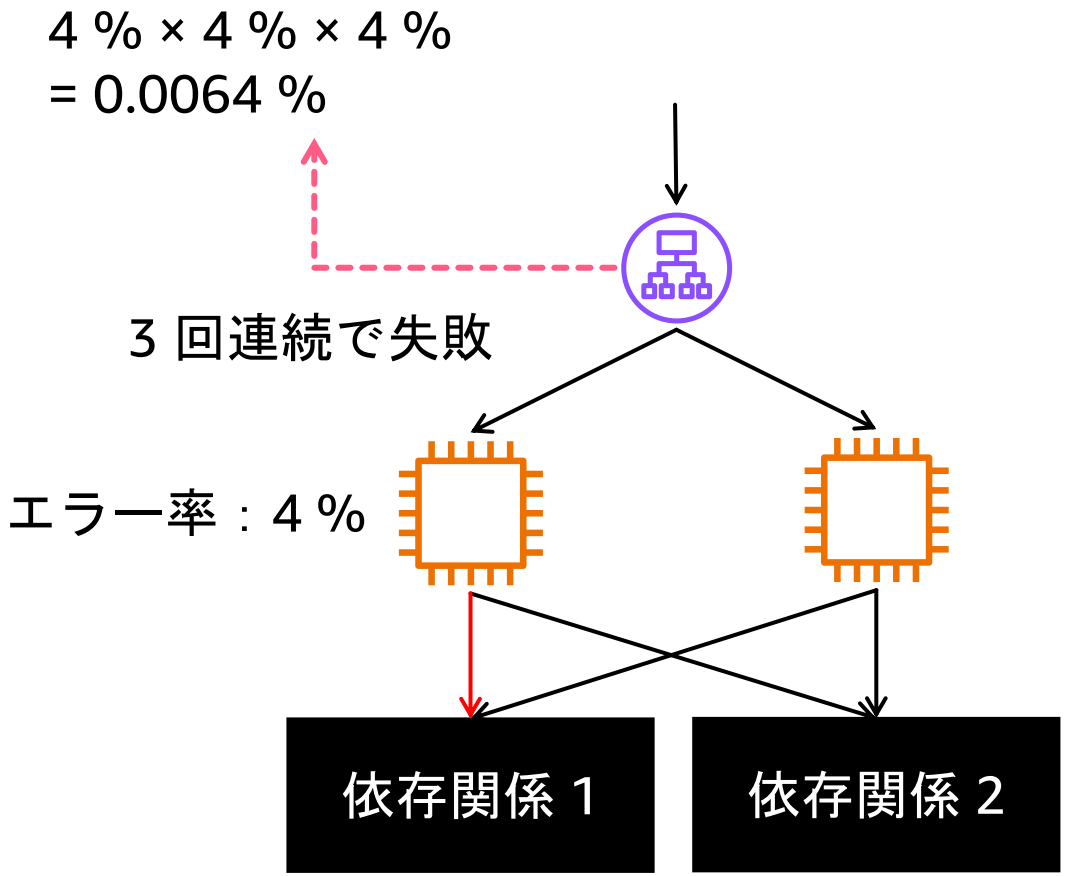




ヘルスチェックで検出できない小さなエラー

- あるサーバーでのエラーレート 4 %
- 3 回連続でヘルスチェックが失敗したらアラームを発行
- 3 回連続で失敗する確率は 0.0064 %

リクエストの 4 % に問題が起きているが検出できず、サービス品質低下



グレー障害の検出

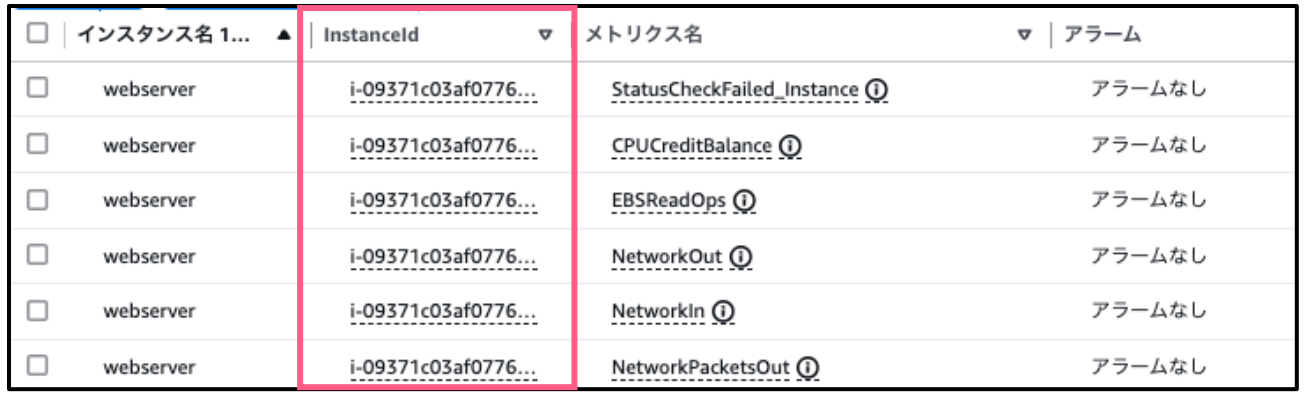
見えづらい問題をどう検出するか

- システム全体では正常に見える中、部分的に発生している異常を見つけ出す
- ヘルスチェックのような基本的な監視では、問題を捉えきれない

Amazon CloudWatch を活用したグレー障害の検出

ディメンションを活用した柔軟な監視

- ディメンションは、メトリクスを分類するためのラベル（InstanceID 等）
- ディメンションには、障害分離境界（ AZ-ID 等）やビジネスコンテキスト（CustomerID 等）を含めることができる
- EMF（Embedded Metric Format）を使用することでログデータに直接メトリクスを埋め込むことが可能



インスタンス名 1...	InstanceId	メトリクス名	アラーム
webserver	i-09371c03af0776...	StatusCheckFailed_Instance	アラームなし
webserver	i-09371c03af0776...	CPUCreditBalance	アラームなし
webserver	i-09371c03af0776...	EBSReadOps	アラームなし
webserver	i-09371c03af0776...	NetworkOut	アラームなし
webserver	i-09371c03af0776...	NetworkIn	アラームなし
webserver	i-09371c03af0776...	NetworkPacketsOut	アラームなし

EMF のディメンション設定例

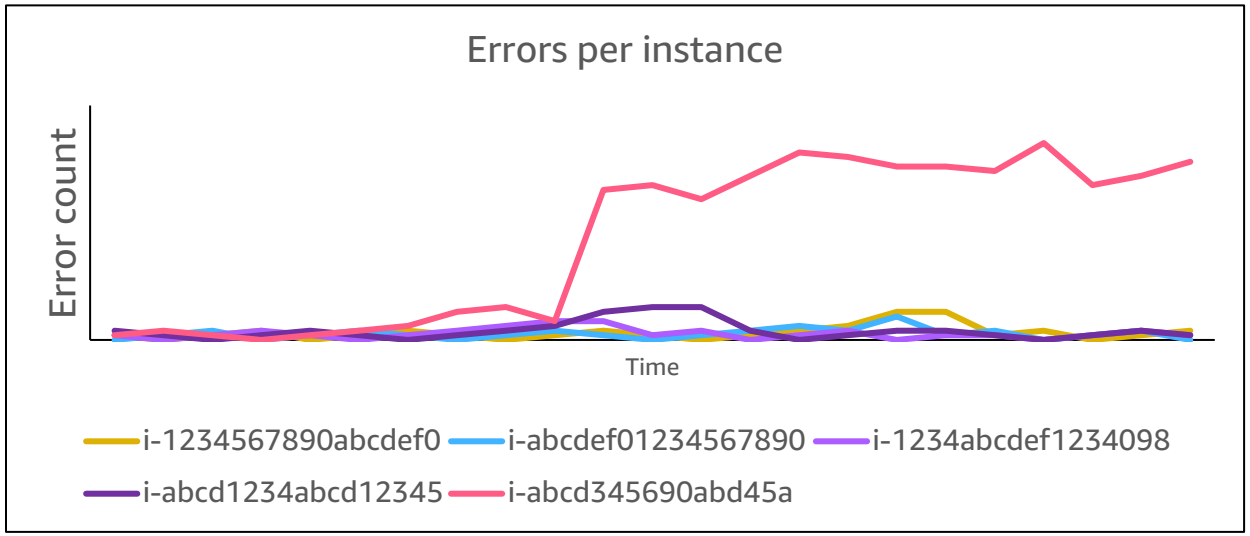
```
"Dimensions": [
        ["AZ-ID", "CustomerId", "Region"],
        ["AZ-ID", "Region"]
      ]
```

高カーディナリティな（値の種類が非常に多い）ディメンションをどう分析するか？

Contributor Insights の活用

Contributor Insights を使用することで、膨大な数の要素から重要な情報を抽出

高カーディナリティデータを分析する



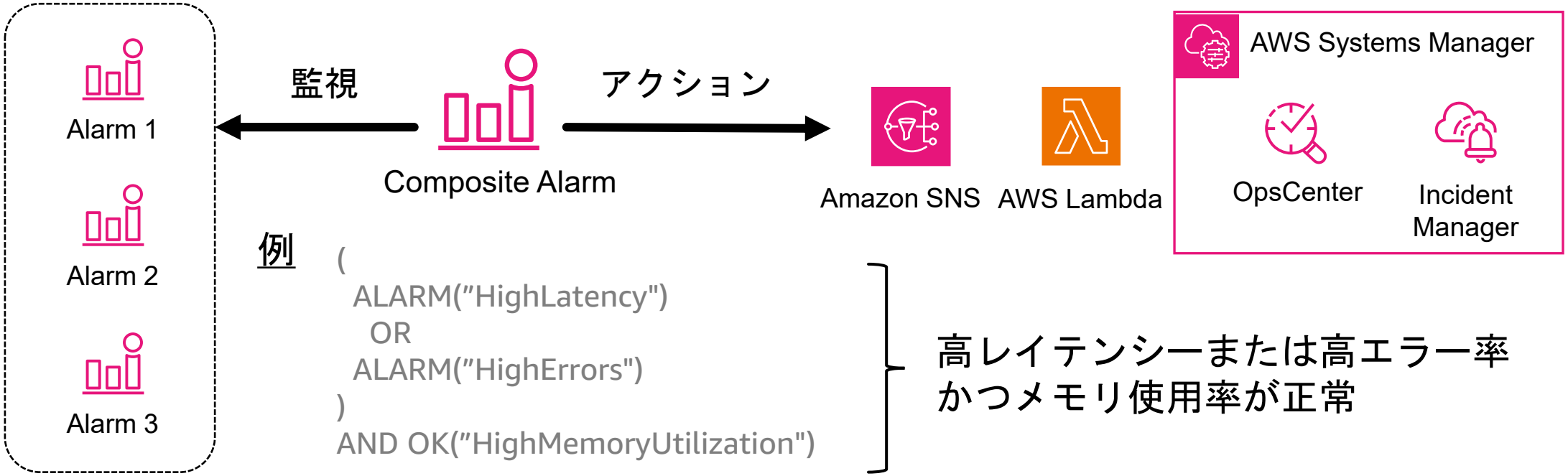
- CloudWatch Logs の時系列データに対する上位 N のコントリビューターの分析を簡素化
- システムとアプリケーションのパフォーマンスに影響を与えている人またはものをリアルタイムに確認
- グラフ化やアラームの設定も可能

例：

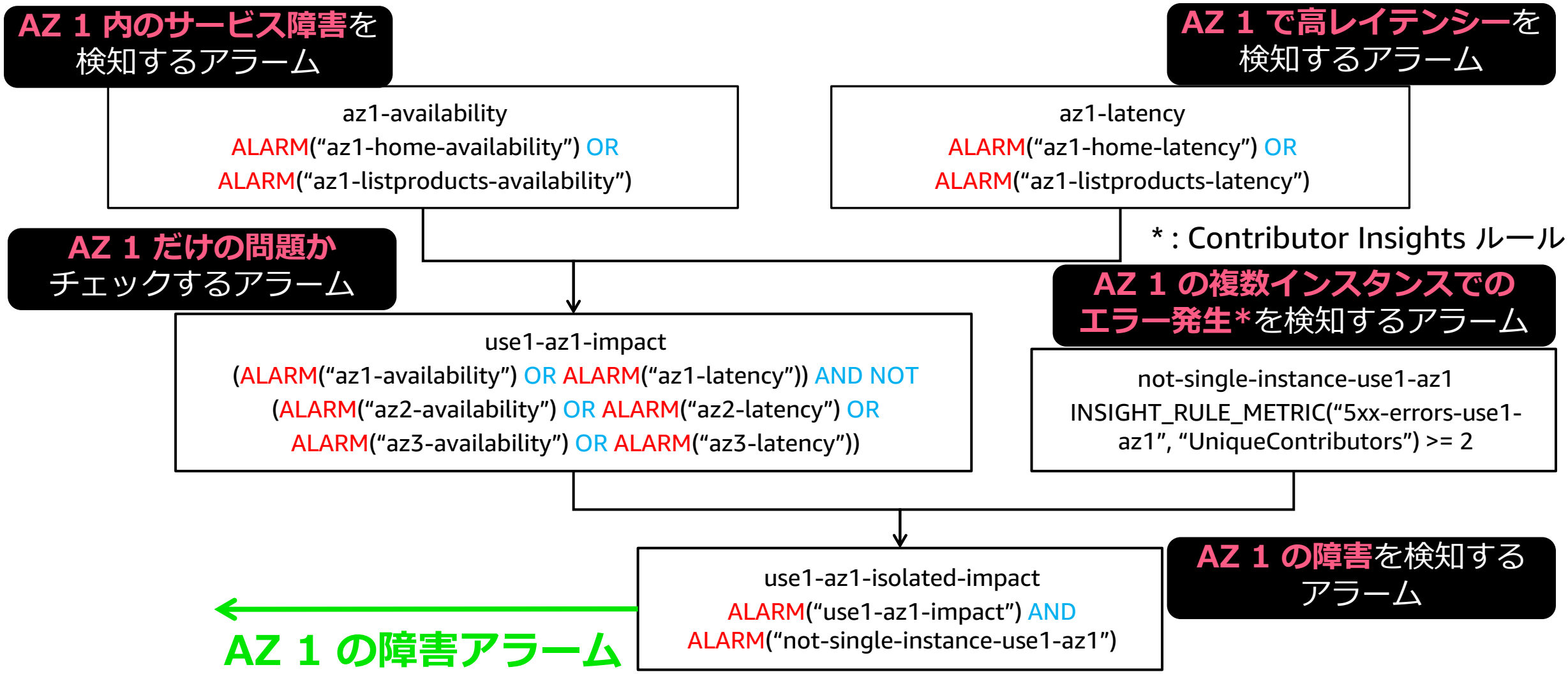
- エラーを最も多く返す API /インスタンス
- 特定のステータスコードが頻出する URL

複合アラーム

- **複数のアラームの状態を組み合わせて監視**
- 特定の条件の組み合わせが満たされた場合にのみアクションを実行
- AND / OR / NOT というシンプルなアラームロジックで設定



複合アラームを応用して特定 AZ の問題を検出



グレー障害のAWSでの対応例

グレー障害対応の選択肢

状況に応じて適切な手段を選択する

	インスタンス単位	アプリケーション	ロケーション切り替え
特徴	• 比較的シンプルな実施手順 • 効果は該当リソースに限定	• 事前の実装と設計が必要 • より柔軟な対応が可能	• 物理的な問題から完全な退避が可能 • 広範な問題に有効
例	• インスタンスの再起動 • 新規インスタンスへの置き換え	• グレイスフルデグラデーション • サーキットブレーカー • エクスポネンシャルバックオフによるリトライ 等	• AZ のシフト • リージョンフェイルオーバー

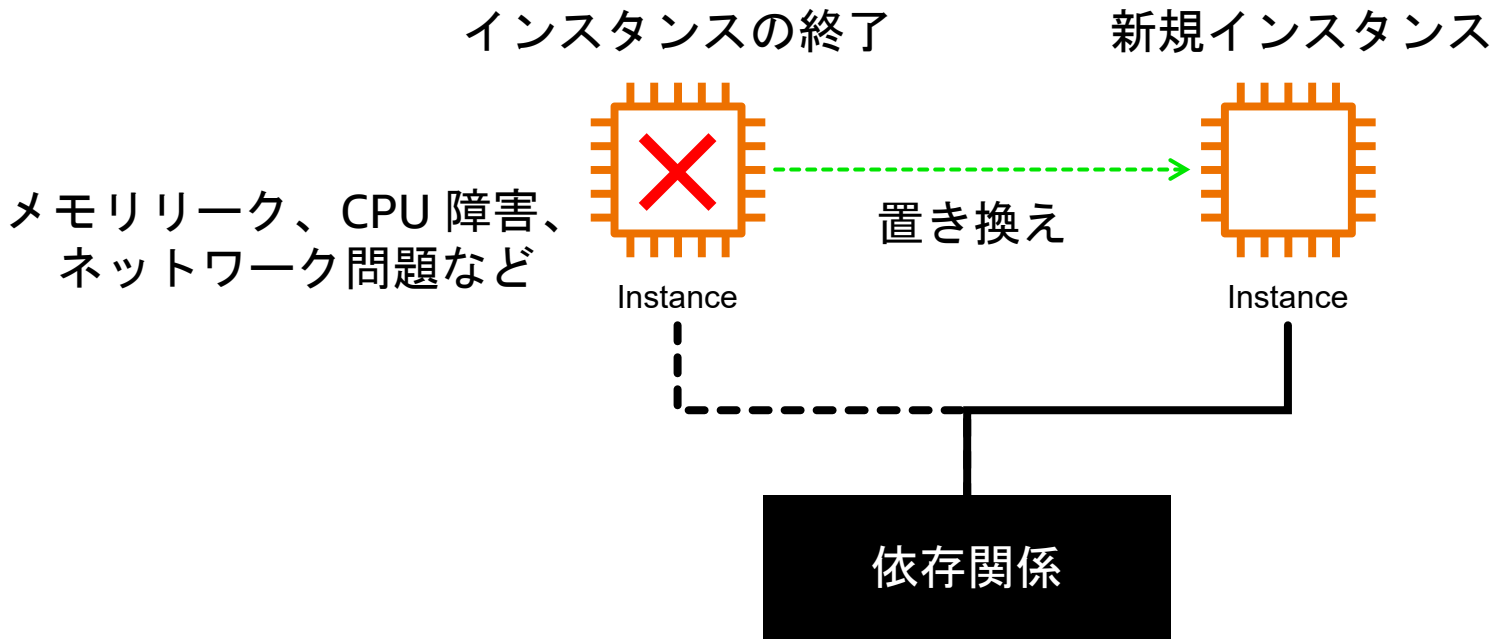
グレー障害対応の選択肢

状況に応じて適切な手段を選択する

	インスタンス単位	アプリケーション	ロケーション切り替え
特徴	• 比較的シンプルな実施手順 • 効果は該当リソースに限定	• 事前の実装と設計が必要 • より柔軟な対応が可能	• 物理的な問題から完全な退避が可能 • 広範な問題に有効
例	• インスタンスの再起動 • 新規インスタンスへの置き換え	• グレイスフルデグラデーション • サーキットブレーカー • エクスポネンシャルバックオフによるリトライ 等	• AZ のシフト • リージョンフェイルオーバー

インスタンス単位の対応

- 新しいインスタンスに置き換えることで、解消する可能性もある
- AZ全体に影響が出ているような場合には適していない
- 新しいインスタンス起動と初期化に時間がかかる



グレー障害対応の選択肢

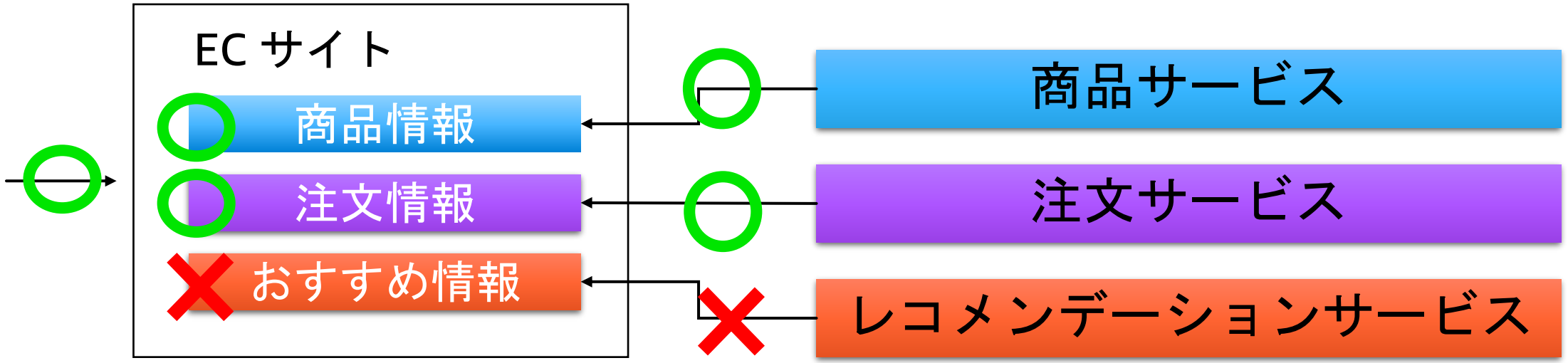
状況に応じて適切な手段を選択する

	インスタンス単位	アプリケーション	ロケーション切り替え
特徴	• 比較的シンプルな実施手順 • 効果は該当リソースに限定	• 事前の実装と設計が必要 • より柔軟な対応が可能	• 物理的な問題から完全な退避が可能 • 広範な問題に有効
例	• インスタンスの再起動 • 新規インスタンスへの置き換え	• グレイスフルデグラデーション • サーキットブレーカー • エクスポネンシャルバックオフによるリトライ 等	• AZ のシフト • リージョンフェイルオーバー

グレイスフルデグラデーション

Graceful Degradation 優美（上品）な劣化

一部のコンポーネントが停止しても、システム全体は停止せず
機能制限（デグレード）しながらサービスを提供し続ける能力



一部の情報が表示されなくても顧客は
引き続きECサイトの主要機能を利用可能

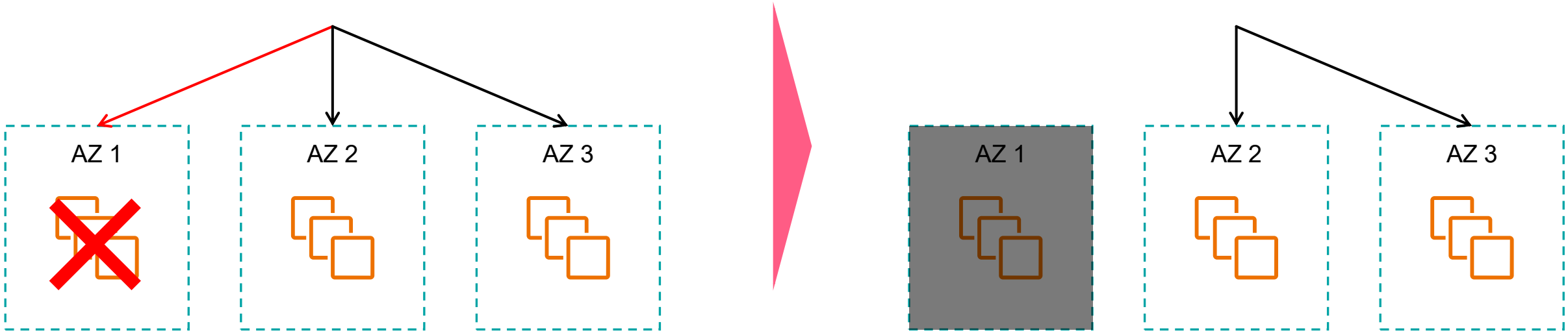
グレー障害対応の選択肢

状況に応じて適切な手段を選択する

	インスタンス単位	アプリケーション	ロケーション切り替え
特徴	• 比較的シンプルな実施手順 • 効果は該当リソースに限定	• 事前の実装と設計が必要 • より柔軟な対応が可能	• 物理的な問題から完全な退避が可能 • 広範な問題に有効
例	• インスタンスの再起動 • 新規インスタンスへの置き換え	• グレイスフルデグラデーション • サーキットブレーカー • エクスポネンシャルバックオフによるリトライ 等	• AZ のシフト • リージョンフェイルオーバー

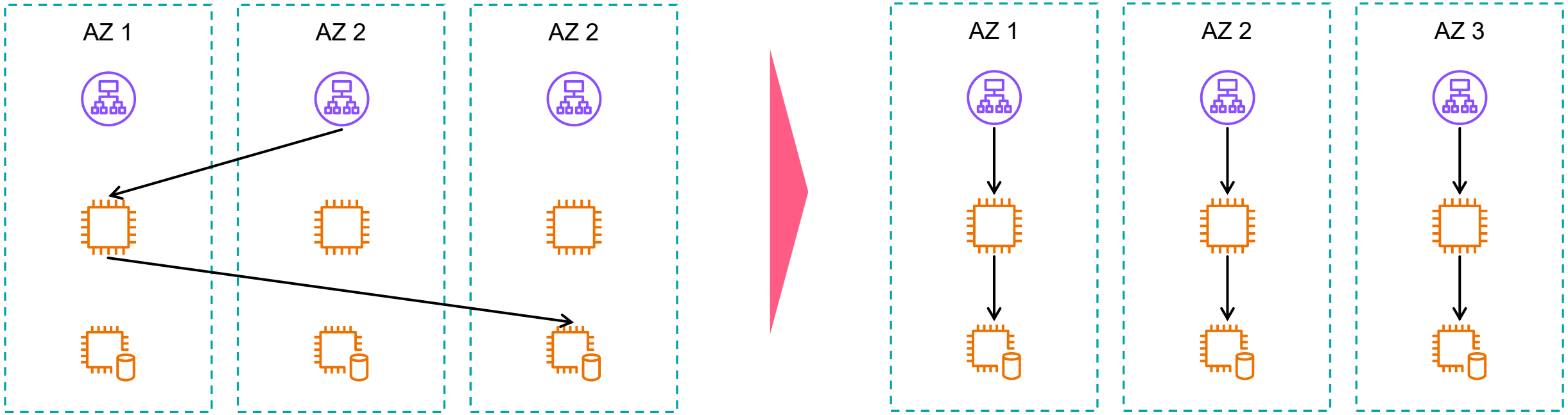
アベイラビリティゾーンの切り離し

- **問題のある AZ を一時的に切り離す**ことで、システム全体の健全性を保つ
- API レスポンス低下やコントロールプレーン障害など**広範な問題への対応**が可能
- 障害の原因特定に時間をかけずに、**まず影響を回避**できる



アベイラビリティゾーンの独立性
Availability Zone independence (AZI)

- 異なる AZ 間の不要な通信を防ぎ、AZ 内でリソースの利用を完結させる
- AZ 間のデータ転送による遅延とコストを最小限に抑える
- AZ 障害等の際、影響を受けている AZ からの退避を可能にする



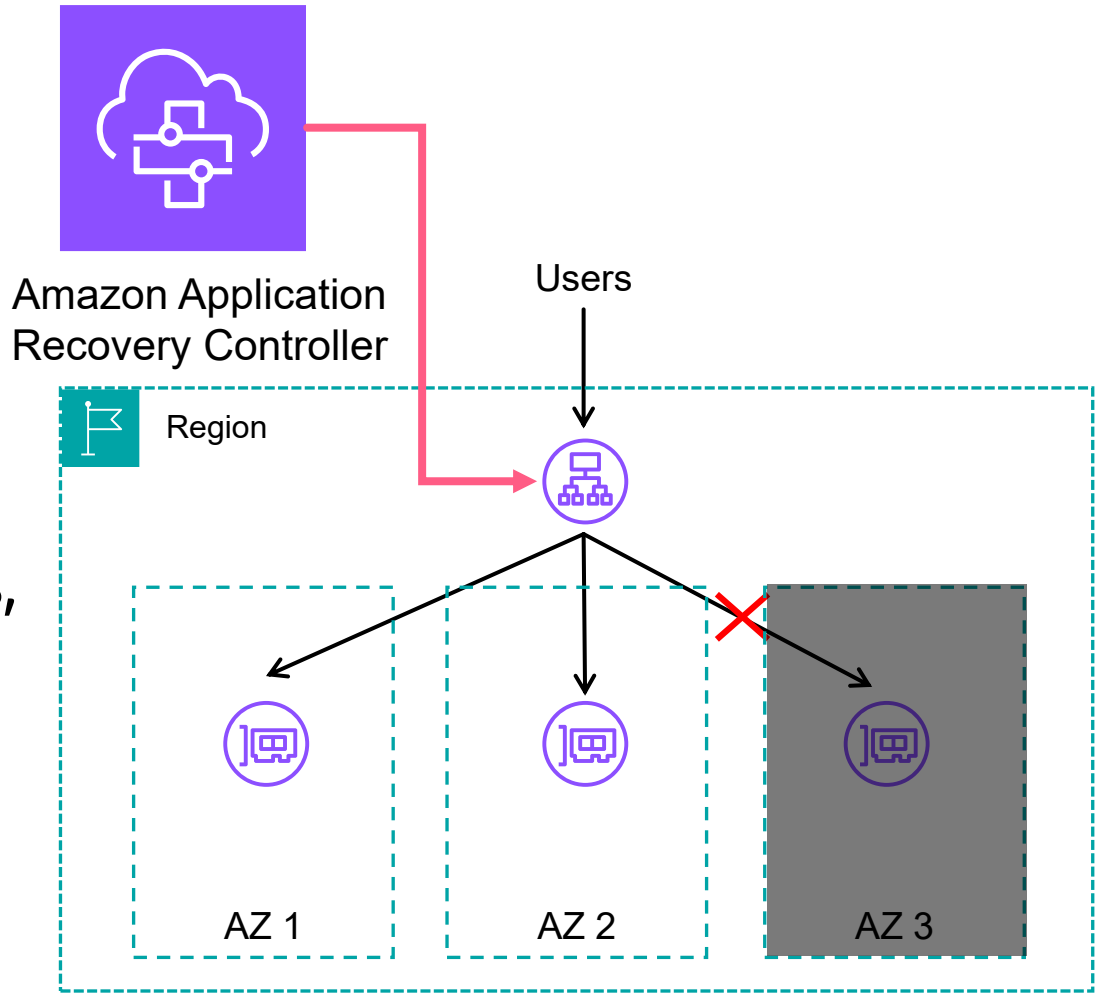
https://aws.amazon.com/jp/builders-library/static-stability-using-availability-zones/

どうやって問題のある AZ を切り離すか？

Amazon Application Recovery Controller (ARC)

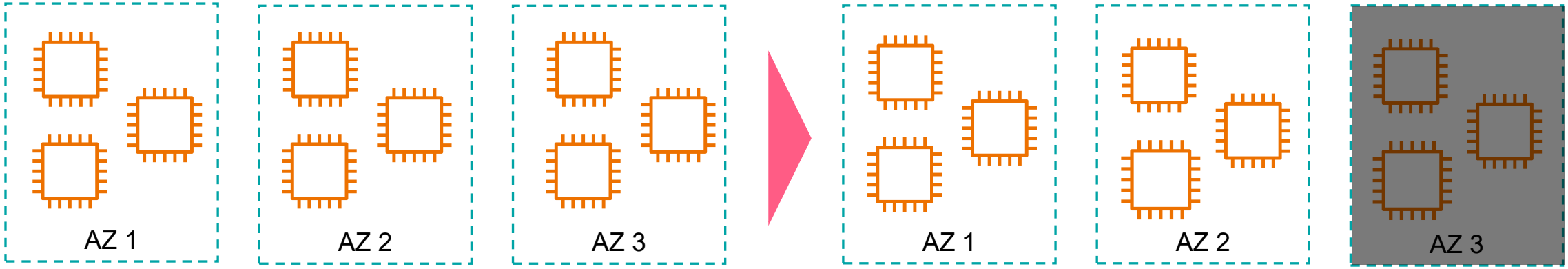
ゾーンシフト

- リージョン内のリソースに対して、**特定 AZ へのトラフィックルーティングを回避**
- トラフィックを遠ざける AZ と サポートされているリソースを選択する
- 指定有効期間の後、自動的に復帰
- ALB/NLB, Amazon EC2 Auto Scaling groups, Amazon EKS をサポート



ゾーンシフトにおけるリソース容量の考慮

- 1 つの AZ で障害が発生しても残りの AZ で負荷を処理できるよう、**事前に十分な容量をプロビジョニング**する
- 3 つの AZ を使用する場合、各 AZ は通常負荷の 66 % で動作するようにオーバープロビジョニング
  - アプリケーションに 6 インスタンスが必要な場合、各 AZ に 3 インスタンス（計 9 インスタンス）を配置する
  - 1 つの AZ がダウンしても、残りの AZ で合計 6 インスタンスを維持可能



ゾーンシフト実行の判断基準

ゾーンシフトの実行タイミングを判断するには、明確な SLO/SLI の設定が不可欠

サービスレベル目標（SLO）

- サービスの信頼性に関する目標値
- 顧客に提供するサービス品質の目標を定量的に示したもの

例：
- 注文処理の成功率を 99.9 % 以上に保つ
- ページ読み込み時間を 2 秒以内に抑える
- 買い物かご機能の可用性を 99.99 % 維持する
- 決済処理のエラー率を 0.1 % 未満に抑える

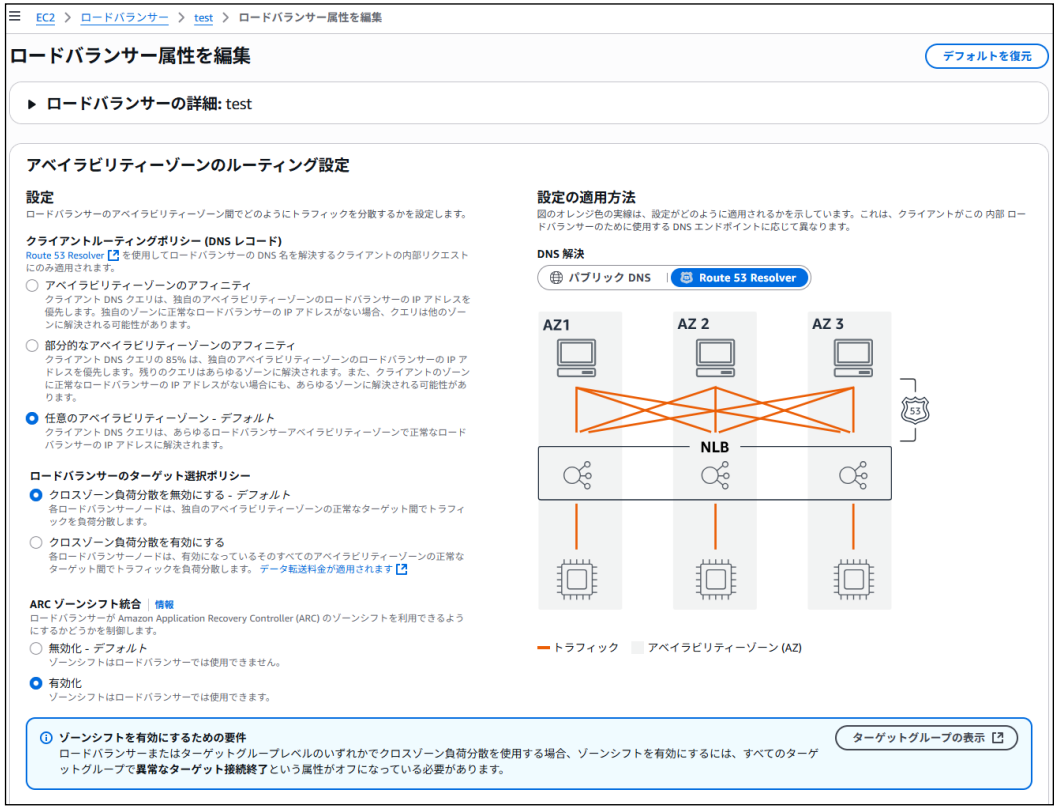
サービスレベル指標（SLI）

- SLO の達成度を測定するための具体的な指標
- 実際に測定可能な定量的なメトリクス

例：
- 注文 API の HTTP 200 レスポンス率
- トップページの Time to First Byte (TTFB)
- 買い物かご API のエラー率
- 決済トランザクションの成功率

例：AZ 障害が疑われ、SLO 違反が 15 分以上継続する場合、ゾーンシフトを開始

ゾーンシフトは手動で実行できる



ゾーンシフトを有効にしておけば、手動でゾーンシフトを実行可能
APIで自動実行も可能

ゾーンオートシフト

AZ に潜在的な影響がある場合、**AWS が自動的にトラフィックを別の AZ に移動**

- AWS の内部モニタリングツールを使用して早期に問題を検知
- 電源障害やネットワーク障害などの**潜在的な問題に自動対応**
- 明示的な有効化が必要
- 毎週 30 分間の練習実行が走る（練習実行をブロックする時間帯を定義することも可能）
- Amazon EventBridge を使用してモニタリングも可能

今日持って帰っていただきたいこと（再掲）

- グレー障害とは何かを理解する
- 耐障害性をあげる方法について理解する
- カオスエンジニアリングと、訓練の方法について理解する

カオスエンジニアリングと障害の訓練について

カオスエンジニアリングとは

Netflix 社がクラウド上で分散したアプリケーションを構築していく中でスタートした、本番環境に意図的に障害を注入し、その影響を観察する実験を通して、システムの可用性、信頼性を高めていく取り組み

https://principlesofchaos.org/ja/

「カオスエンジニアリングの原則」では、システムが本番環境における不安定な状態に耐える能力へ自信を持つためにシステム上で実験を行う訓練方法　と定義

カオスエンジニアリングの詳細原則

- 定常状態における振る舞いの仮説を立てる
- 実世界の事象は多様である
- 本番環境で検証を実行する
- 継続的に実行する検証の自動化
- 影響範囲を局所化する

https://principlesofchaos.org/ja/

カオスエンジニアリングの詳細原則

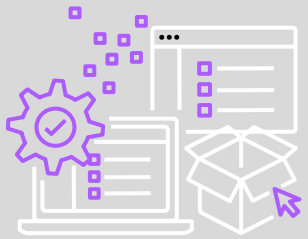
- 定常状態における振る舞いの仮説を立てる
- 実世界の事象は多様である
- 本番環境で検証を実行する
- 継続的に実行する検証の自動化
- 影響範囲を局所化する

これらを簡単に実現できないか？

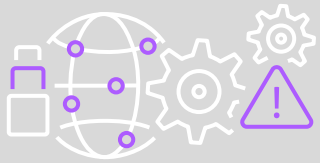
AWS Fault Injection Service（FIS）

障害注入実験を実行するためのフルマネージド サービス

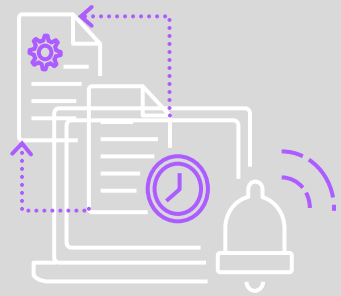
簡単に始められる



実環境の条件で

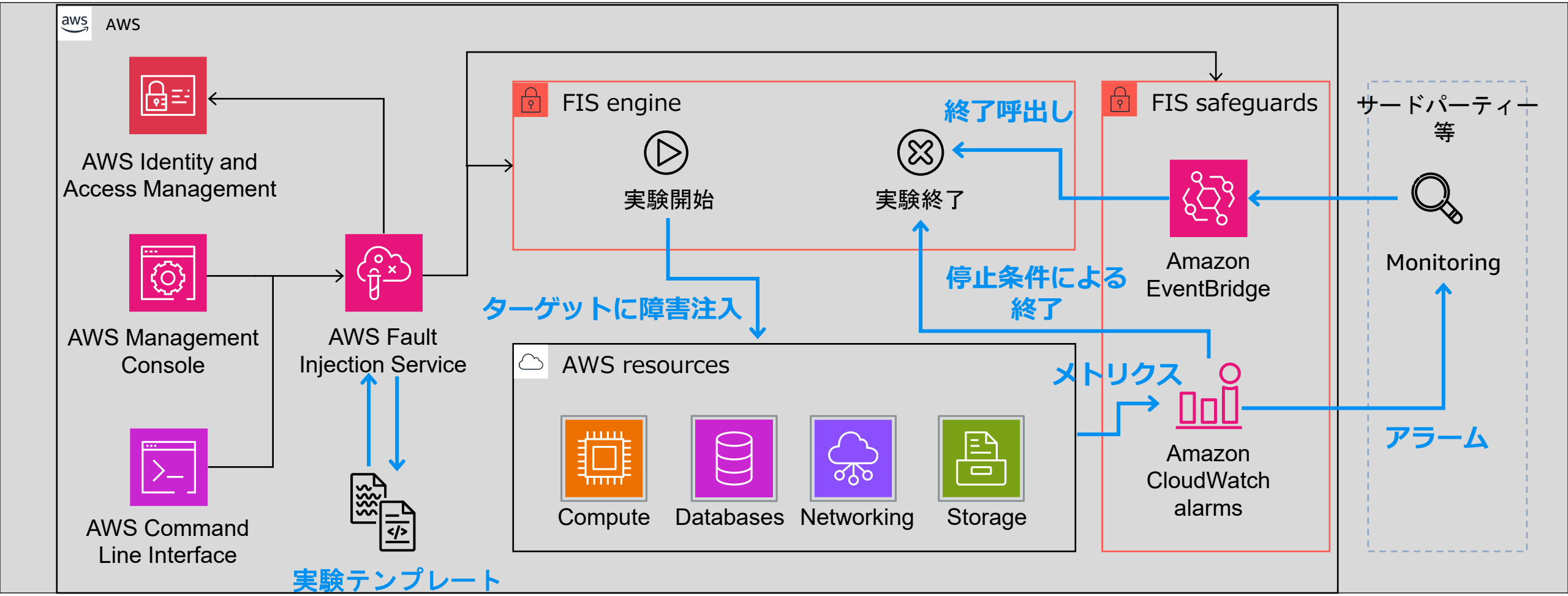


セーフガード



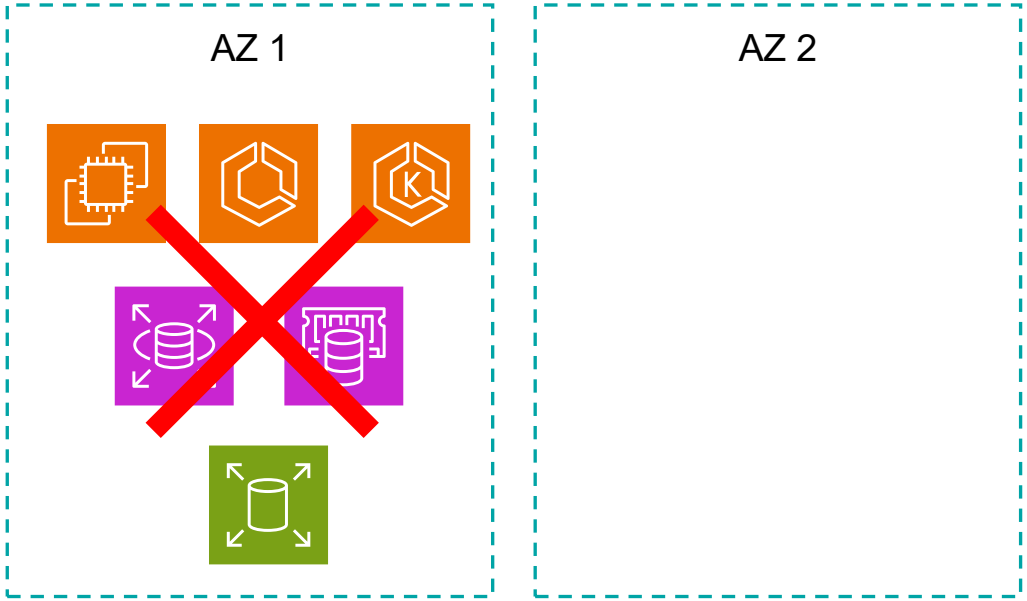
※ AWS Resilience Hub の1つの機能

AWS FIS を使った実験の流れ



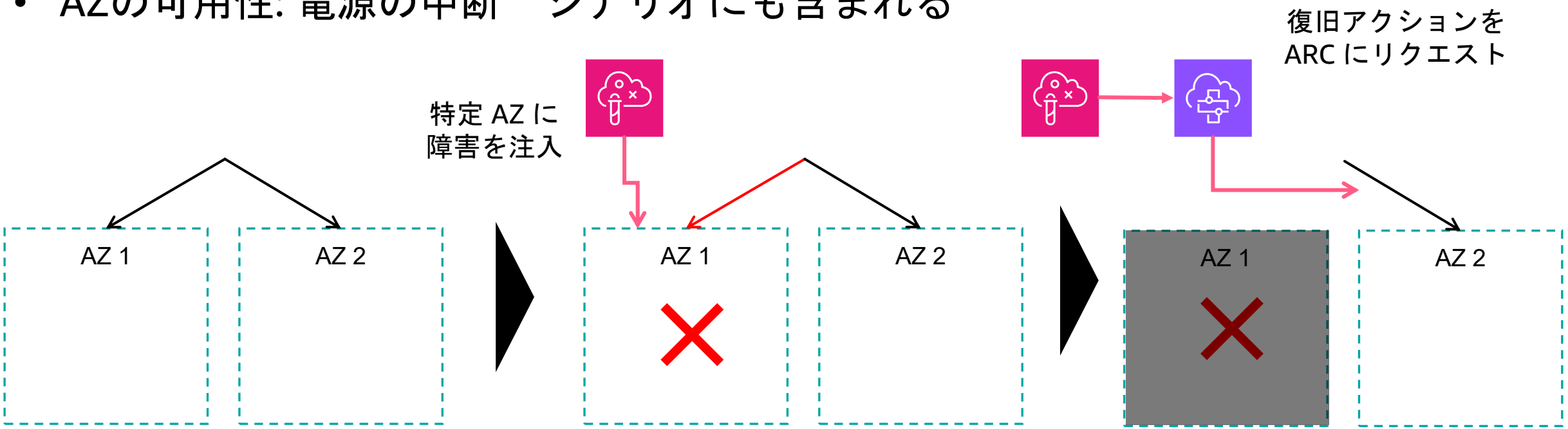
AWS FIS で AZ 障害をシミュレートして訓練する

- お客様がアプリケーションの耐障害性をテストするために適用できるイベントを定義したシナリオライブラリ
- **AZの可用性: 電源の中断**　のシナリオを使用して、AZ の電力が中断される状況をシミュレート



AWS FIS の復旧アクション

- aws:arc:start-zonal-autoshift という復旧アクションが提供されている
- 特定の AZ からのトラフィックシフトをシミュレートすることが可能
- FIS が ARC の API を使用してリクエストを送りゾーンシフトが開始
- AZの可用性: 電源の中断　シナリオにも含まれる



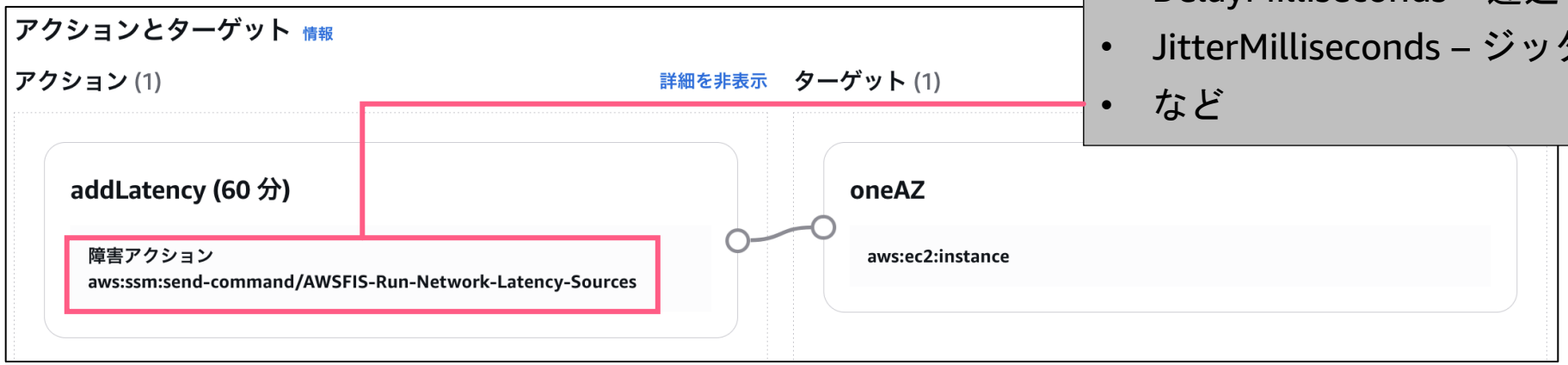
SSM ドキュメントでグレー障害シミュレート

AWS FIS の SSM ドキュメントアクションを使用してグレー障害をシミュレート

- SSM Agent がインストールされたインスタンスで実行可能
- グレー障害をシミュレートできる SSM ドキュメントアクションの例：
  - **AWSFIS-Run-Network-Latency-Sources :**
    特定ソースとの通信にネットワークレイテンシーとジッターを追加
  - **AWSFIS-Run-Network-Packet-Loss-Sources :**
    特定ソースとの通信にパケットロスを発生

ドキュメントパラメータ

- DelayMilliseconds – 遅延（ミリ秒）
- JitterMilliseconds – ジッター（ミリ秒）
- など



組織全体で取り組むレジリエンス文化の醸成

AWS Well-Architected フレームワーク：信頼性の柱

- 障害訓練を特別なイベントではなく
  日常的な活動として組織に定着させる
- 技術チームだけでなく、**実際の障害時に**
  判断を求められる関係者も含める
- 訓練で見つかった課題をプロセス改善につなげ、
  次の訓練で検証

まとめ

- グレー障害とは何かを理解する
- 耐障害性をあげる方法について理解する
- カオスエンジニアリングと、訓練の方法について理解する

参考資料

- AWSにおけるグレー障害の検出と対策
  https://www.youtube.com/watch?v=Nn07ZgWqI8I
- ミッションクリティカルシステムを AWS に載せるには?
  https://pages.awscloud.com/rs/112-TZM-766/images/AWS-43_AWS-Summit-2023_Resilience.pdf
- **第三十七回 ちょっぴり DD - AWS Fault Injection Service で AZ 障害を体験しよう - データベースの可用性向上対策の検証**
  https://www.youtube.com/watch?v=9KYuib6NNFM
- **カオスエンジニアリング—回復力のあるシステムの実践**
  https://www.oreilly.co.jp//books/9784873119885/

Thank you!

Yukihiro Kikuchi
kyukihi@amazon.co.jp