SUMMIT

Tokyo

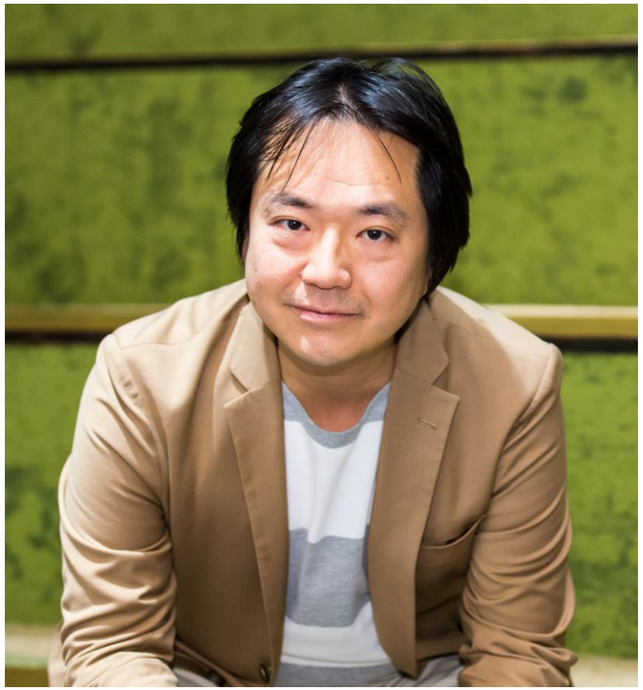
E03-06

今さら聞けない耐障害性の話
～クラウドで信頼性を高めるためには～

角川宗近
シニアクラウドサポートエンジニア
Premium Support
アマゾン ウェブ サービス ジャパン株式会社

aws SUMMIT

自己紹介



角川宗近 (すみかわ むねちか)
　シニア クラウドサポートエンジニア

AWSサポートのエンジニアとして国内外のお客様の技術課題の解決する

専門分野
　VPCなどのネットワーク系サービス
　CloudFront

本セッションの対象者と目的

対象者

- クラウド入門者
- オンプレミス環境からアマゾン ウェブ サービス (AWS)への移行を検討中のお客様
- AWS上でシステムを新規開発を始めようとしているお客様

目的

- クラウドでの信頼性の考え方について指針を与える

本日の内容

- 障害に対するオンプレミスとクラウドのアプローチの違い
- 信頼性に関わる4つの要求項目
  - 個々の要求項目でマインドセットをどう変えるか
- さらなる信頼性向上に向けて
- まとめ

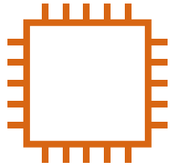
障害に対するアプローチの違い

オンプレミス

壊れにくい部品で構成してシステムの信頼性を上げる

クラウド

壊れた部品をクイックに交換してシステムの信頼性を上げる



調達時間の差が理由の一つ

調達時間の差

	オンプレミス	クラウド
対象	物理サーバ	仮想サーバ
調達	数週間～数ヶ月	数分
交換	数時間～数日	数分

調達が大変なので長く使いたい
交換が大変なので壊れないでほしい

前提条件が変わる
クラウドで信頼性を高めるために、
マインドセットの変革が必要

信頼性とは

システムが正常に動作し、利用ユーザに影響が出ないこと
　障害以外にも信頼性に影響する要因がある
本セッションのスコープ
　非機能要求グレードから信頼性に関わる4つの要求項目を抽出

	要求項目
可用性	稼働率, 災害対策
性能・拡張性	リソース拡張性
運用・保守性	試験用環境の設置

1. 稼働率

システム停止を許容時間内に抑えること

リスク: ハードウェア障害
アクション: 冗長化によるダウンタイム最小化と、正常状態への復旧

	オンプレミス	クラウド
前提	サーバ交換に時間がかかる	交換は数分で完了する
マインドセット	壊れにくいハードウェアを選定 冗長構成を組む	自動復旧するアーキテクチャ

1. 稼働率: オンプレミスの場合

稼働率	年間最大停止時間	アプリケーション例
99%	3日15時間	バッチ処理、ジョブの実行
99.9%	8時間45分	プロジェクト管理などの内部ツール
99.95%	4時間22分	電子商取引
99.99%	52分	動画配信、放送システム
99.999%	5分	ATM取引、通信システム

- 一度でもサーバが壊れると稼働率の達成は難しい
- 冗長構成は必須だが、片系状態が長引くとリスクが高まる
  → 障害を回避したい → 壊れにくいハードウェアがほしい

出典: ホワイトペーパー 「信頼性の柱」 https://d1.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf

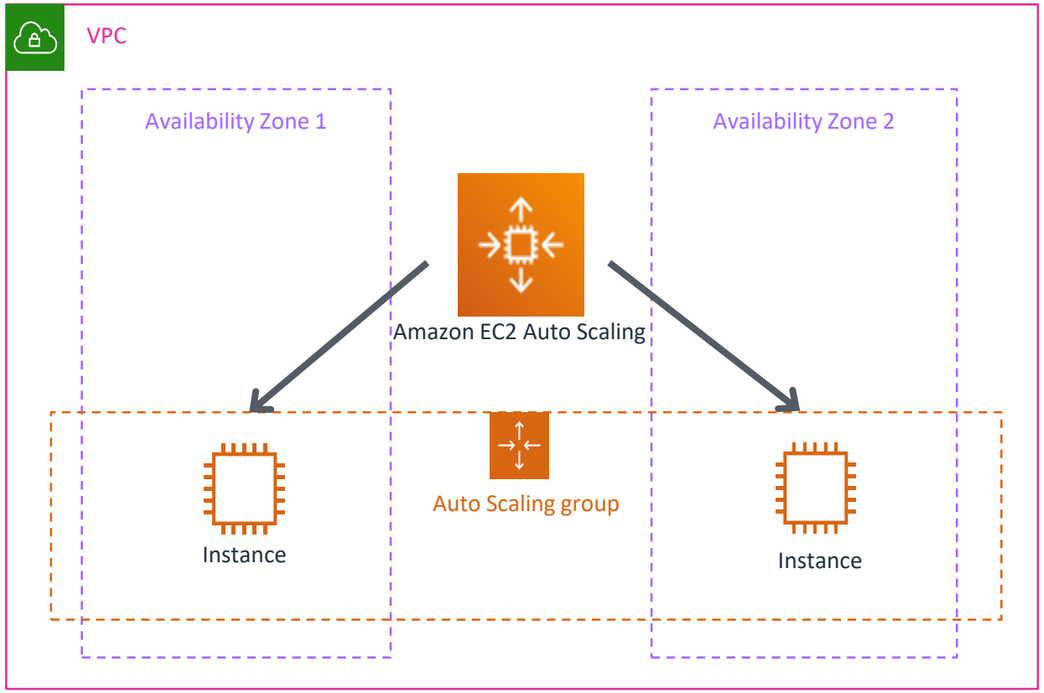
1. 稼働率: AWSの場合 (1)

- 交換自体は数分で終わる
- 重要なのは自動化による素早い完全復旧

サービス例:

Amaon EC2 Auto Scaling

EC2インスタンスを監視し、正常な状態に復旧



1. 稼働率: AWSの場合 (2)

アベイラビリティゾーン (AZ)

- １つ以上のデータセンターで構成される、互いに独立したインフラ
  - 自然災害、電源、ネットワーク等を考慮
- 同時障害のリスクが減るため、マルチAZによる冗長構成を推奨

2. 災害対策

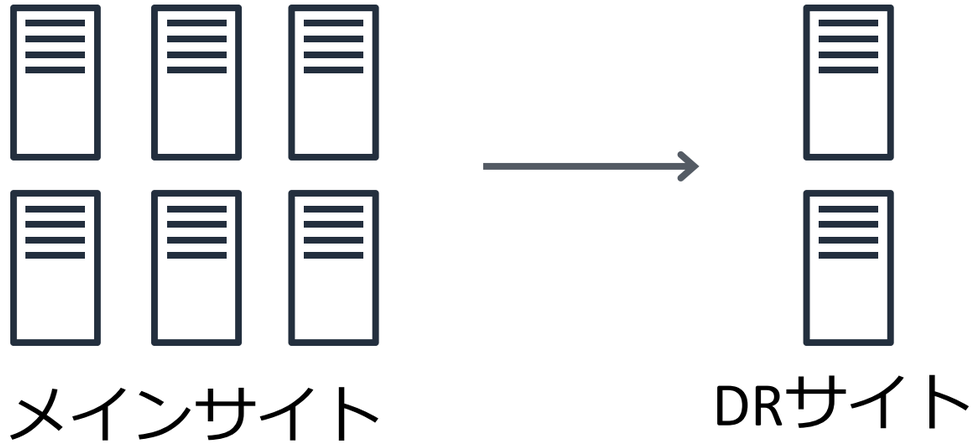
災害時にビジネスが継続できること
リスク：自然災害、停電などによる大規模障害
アクション: 定めらた時間内での復旧(DR)と業務の再開(BCP)

	オンプレミス	クラウド
前提	DR環境維持にコストがかかる	DR環境を維持する必要はない
マインドセット	メインサイトのサブセットをDRサイトに構築する	メインサイトと同一構成のDRサイトを災害時に構築

DR: Disaster Recovery, BCP: Business Continuity Plan

2. 災害対策: オンプレミスの場合

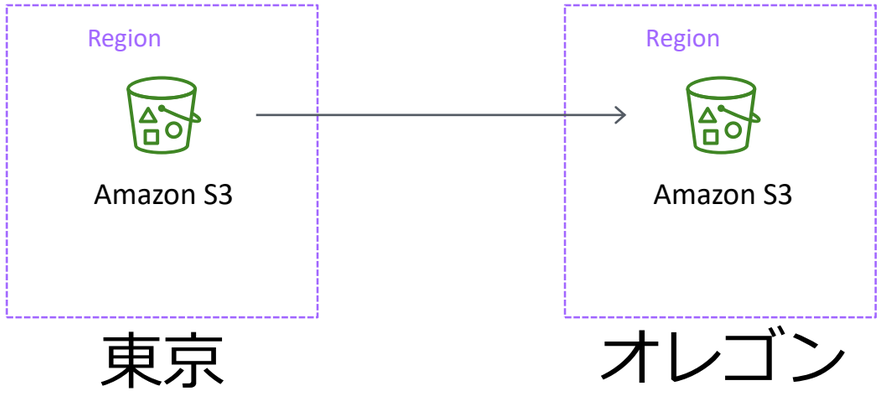
災害が起きてからの調達は間に合わない
　事前にDRサイトを構築しておく
使わないかもしれないハードウェアに対する投資
　現実的には重要度からDR対象を絞り込む必要性



2. 災害対策: AWSの場合

コンピューティングリソースは災害が起きたタイミングで起動
→ 投資を最小限に抑えられる
DR対象リストの検討不要

データは常にバックアップをとっておく
サービス例: S3 クロス リージョン レプリケーション



3. リソース拡張性

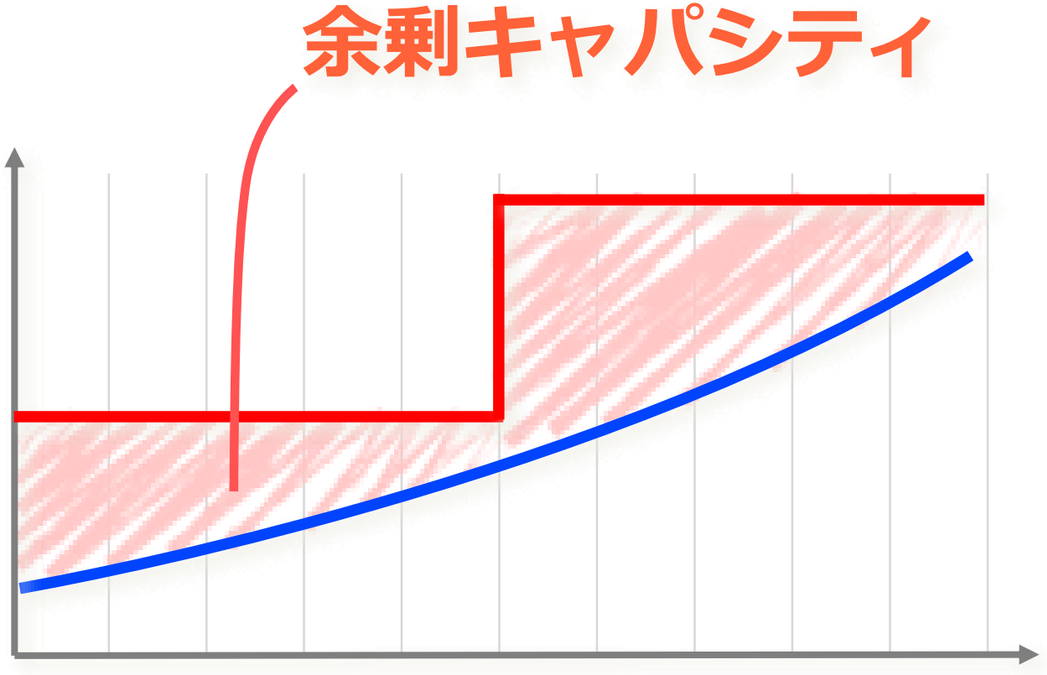
将来にわたって性能面でサービスレベルが維持できること
リスク: 将来のアクセス増
アクション: 必要十分なハードウェアリソースの確保

	オンプレミス	クラウド
前提	スペックは変更できない	スペックの変更は数分
マインドセット	将来のピーク需要を予測し、 要求スペックを決定	予測は不要

3. リソース拡張性: オンプレミスの場合

将来のピークを見積もる(キャパシティプランニング)
精度の高い需要予測は難しい
　予測 > 実需: 過剰コスト
　予測 < 実需: ハードウェアの更新
長く使いたいからマージンを取る

いずれにしろ導入当初はオーバースペック



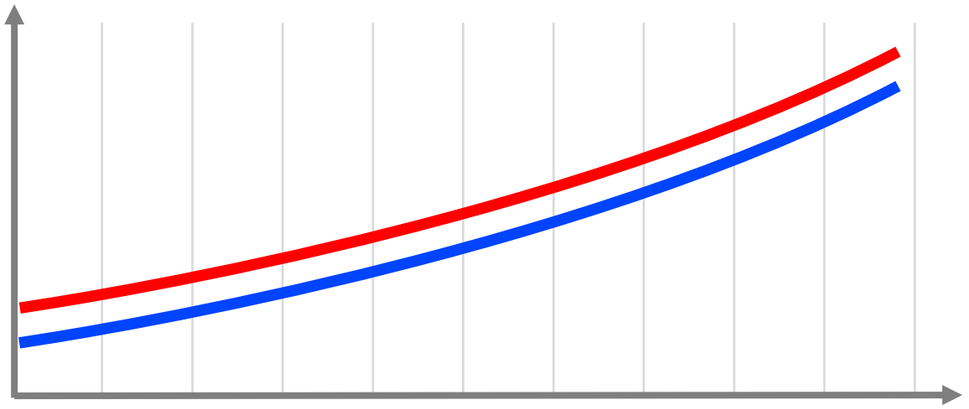
3. リソース拡張性: AWSの場合

- スペック変更は数分で完了
- 実際の負荷から決定

サービス例: EC2インスタンス

- CPU, メモリ: 3ステップ
  1. インスタンスの停止
  2. インスタンスタイプの変更
  3. インスタンスの起動
- ディスク: 無停止で拡張可能 (Elastic Volume)

インスタンスタイプ	vCPU	メモリ (GiB)
m5.large	2	8
m5.xlarge	4	16
m5.2xlarge	8	32
m5.4xlarge	16	64
m5.12xlarge	48	192
m5.24xlarge	96	384



4. 試験用環境の設置

ソフトウェア更新のために試験環境を設置する
リスク: 本番環境でバグが発生し、サービスが停止する
アクション: リリース前の事前テスト

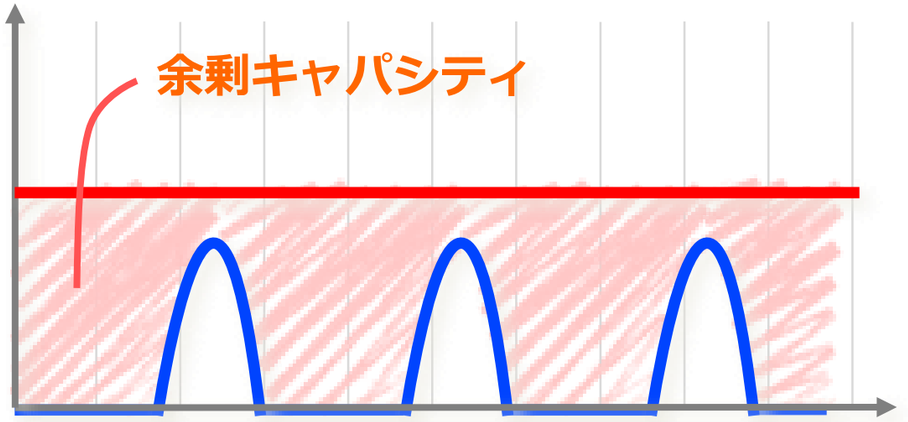
	オンプレミス	クラウド
前提	テスト環境の維持にコストがかかる	テスト環境を維持しない
マインドセット	本番よりスペックを下げてテストする	本番と同一条件でテストを行う

4. 試験用環境の設置: オンプレミスの場合

本番と同一構成のテスト環境を別途用意する
　時々しか使わないハードウェアへの投資

コスト削減から何らかの妥協が入る
　例1: 本番よりスペックを下げてテストをする
　　10000ユーザー同時接続 → 1000ユーザー
　例2: フェイルオーバーテストを省略する

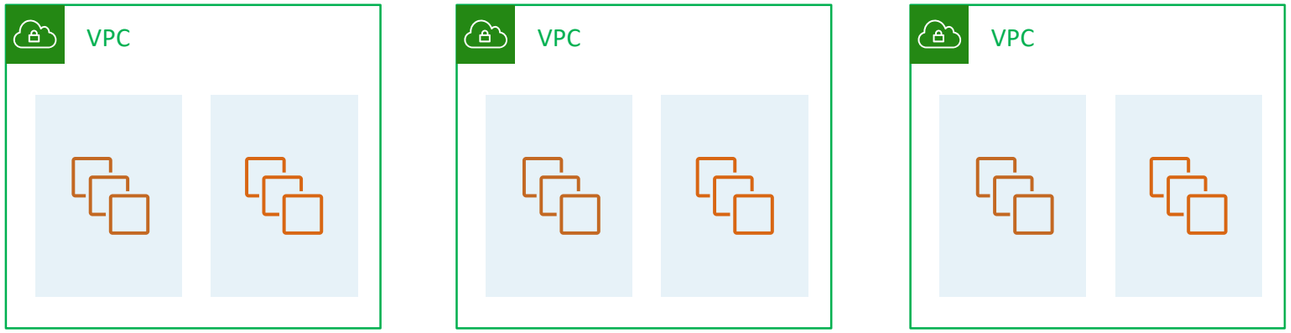
バグの見逃しから、信頼性の低下につながる



4. 試験用環境の設置: AWSの場合

同一スペックの試験環境を必要なタイミングで起動
テストが終われば環境全体を破棄
テストに使った時間分だけがコスト

サービス例: Amazon VPC
　IPアドレスなどを同一にした独立した仮想ネットワークを構築
　複数環境も構築可能



ここまでのまとめ

クラウドではマインドセットの変革が求められる

稼働率	障害からの自動復旧を重視する
災害対策	本番環境と同一のDR環境を構築する データは常にバックアップを取る
リソース拡張性	負荷をモニタリングし、実際のメトリックからスペックを調整する
試験用環境の設置	本番環境と同一のテスト環境を構築する

AWSには変革を支える種々の仕組み、サービスがある

さらなる信頼性向上に向けて

aws SUMMIT

信頼性をさらに上げるには

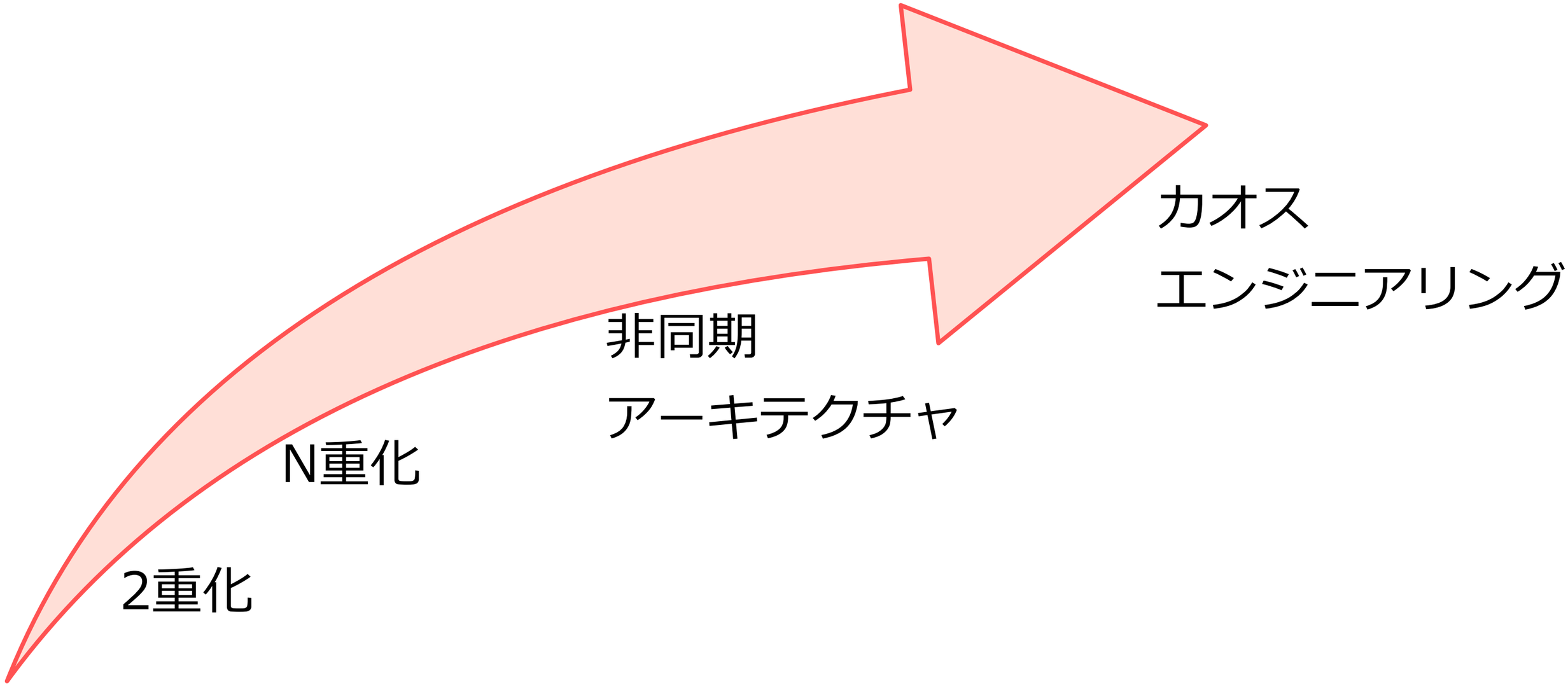
クラウドの特性を活かしたアーキテクチャに移行することで、想定外の障害に耐えられるシステムにする

影響範囲を局所化して全体障害を防ぐ

2軸で考える

影響範囲 = ダウンタイム x 影響を受けたユーザー数

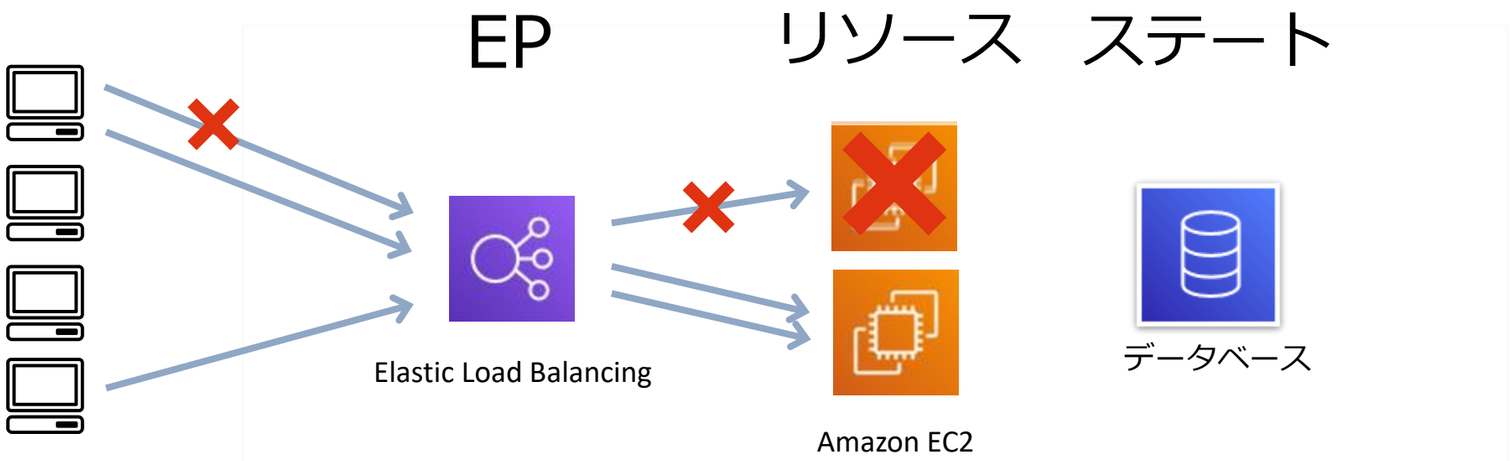
信頼性向上に向けた流れ



2重化

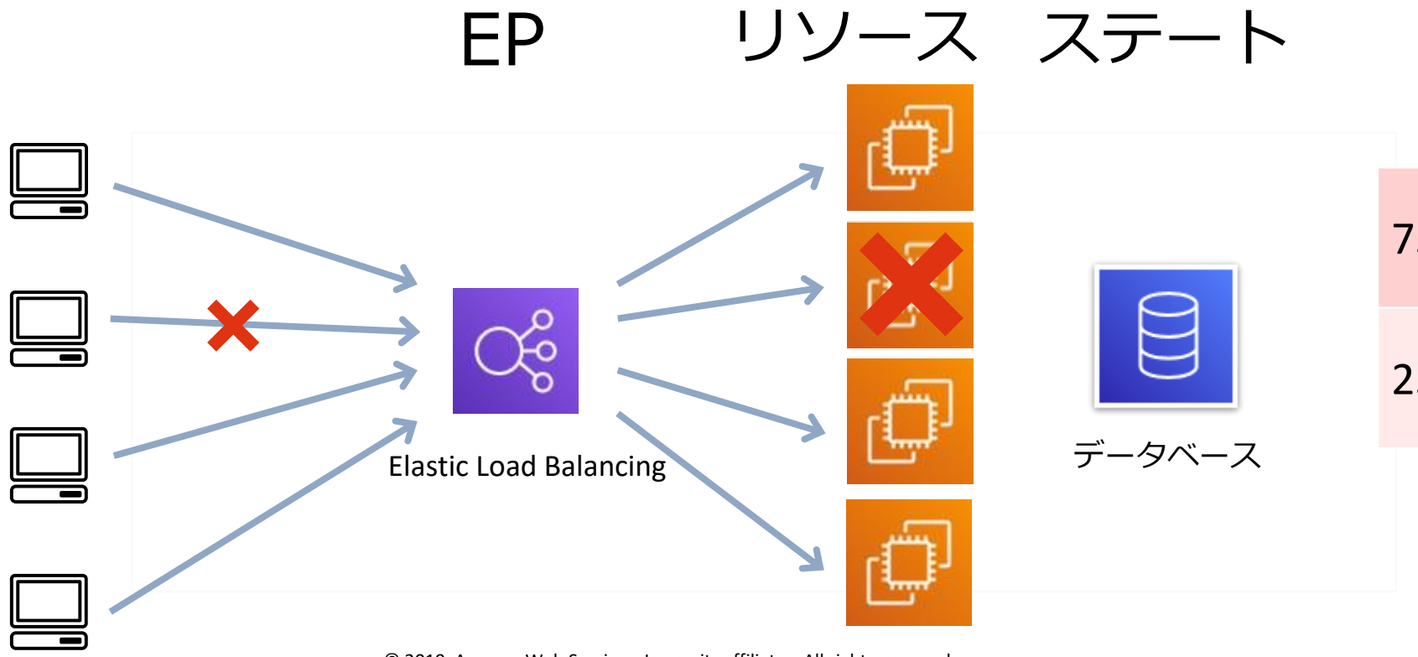
影響範囲: 障害時には50%のユーザが影響を受ける
　クライアントによるリトライで救える
　ユーザには遅延として見える

次のステップに向けて:
　　エンドポイント、リソース、ステートをきちんと分離する



N重化

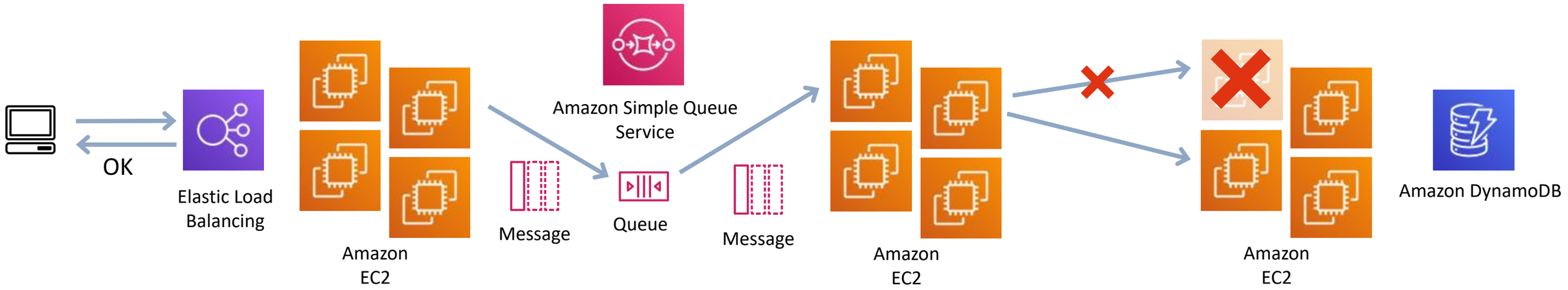
EC2インスタンスのスペックを下げて台数を増やす
　例： m5.4xlarge 2台 → m5.2xlarge 4台
水平方向にスケールして影響範囲を限定
負荷に応じて台数を調整する



75%	成功
25%	リトライで成功

非同期処理アーキテクチャ

キューを介した疎結合なアーキテクチャ
　リクエストを非同期に処理
　クライアントにはリクエストを受け付けたら”成功”を返す
　内部エラーがあっても、リトライによって最終的には成功状態に収束
部分障害をユーザーから隠蔽できる
　ユーザーのリトライ不要



カオスエンジニアリング

防災訓練を日常的におこなう取り組み
- 本番環境で意図的に障害を発生
- 障害の局所化と自動復旧を実際に確認
- ユーザーの使い方は多種多様、テスト環境では再現できない

実現までの道のりも重要
- 細かい改善を適用していく必要
- 継続的インテグレーション/継続的デリバリーの導入など

最終的には障害は深刻ではないイベントに

まとめ

オンプレミスとクラウドは信頼性を高めるためのアプローチが異なります
AWSを使えば、災害対策や試験用環境の構築に妥協は不要になります
モニタリング、障害監視、復旧、スケーリングを自動で行うことによって、システムの信頼性は増します
クラウドの特性を活かすことで、障害は深刻ではないイベントになります

オンプレミスからクラウドにマインドセットを変えて、
クラウドを最大限に活かしましょう

Thank you!

aws SUMMIT