JANOG 46 Meeting 「サービスレベルを再考しよう」

クラウドにおける
信頼性の考え方、方向性とアプローチ

アマゾン ウェブ サービス ジャパン株式会社
セキュリティソリューションアーキテクト
中島 智広

© 2020, Amazon Web Services, Inc. or its Affiliates.

自己紹介

中島　智広（Tomohiro Nakashima）

AWS Security Solutions Architect

お客様のセキュリティの取り組みをクラウド利活用の視点からご支援

Background

前職時代にASやデータセンターネットワーク、DNSなど、オンプレミスのインフラ設計・運用に従事、JANOGとはその頃からのご縁

© 2020, Amazon Web Services, Inc. or its Affiliates.

はじめに

よりよいサービスレベルを実現するには

お客様のアーキテクチャに踏み込む必要がある

© 2020, Amazon Web Services, Inc. or its Affiliates.

お話すること

どのようにお客様とサービスレベルやそれを実現するアーキテクチャの議論をしていくのか？

クラウド事業者がお客様と実施している
リファレンスやフレームワークを用いた取り組み

をご紹介します。

© 2020, Amazon Web Services, Inc. or its Affiliates.

前回お話したこと

クラウドが前提とする
責任共有モデルの考え方とその効能

定期健康診断 Well-Architectedの取り組みとその効能

【参考】JANOG 45 Meeting常識が変わる責任共有のカタチ
https://www.janog.gr.jp/meeting/janog45/program/responseshare

© 2020, Amazon Web Services, Inc. or its Affiliates.

責任共有モデル

Design for Failureを前提として

事業者/利用者のやるべきことへの集中をもたらす

これが責任共有モデルです。

https://aws.amazon.com/jp/compliance/shared-responsibility-model/

© 2020, Amazon Web Services, Inc. or its Affiliates.

前提となる重要なこと

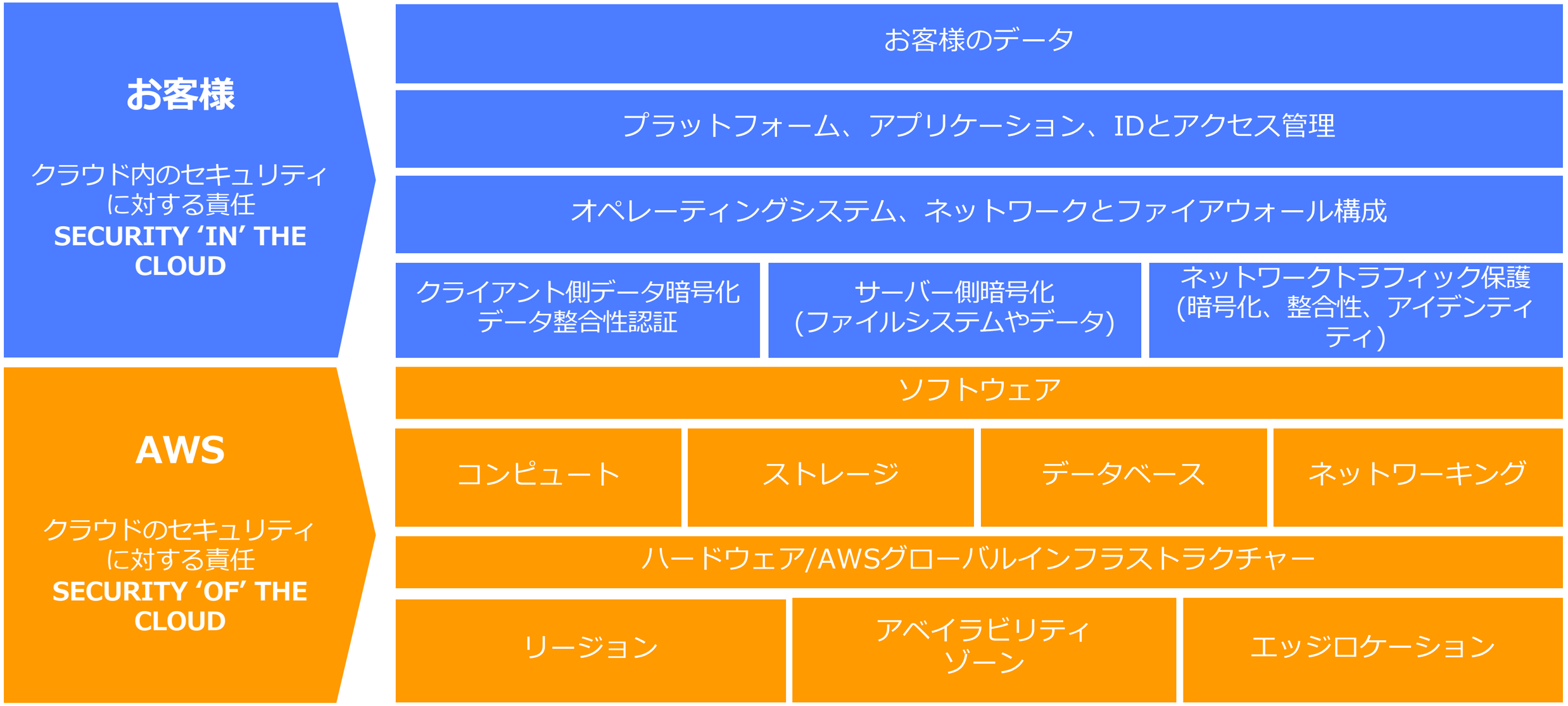
“Everything fails all the time.”

すべてのものはいつでも壊れうる

—Werner Vogels, CTO AWS

© 2020, Amazon Web Services, Inc. or its Affiliates.

責任共有モデル



https://aws.amazon.com/jp/compliance/shared-responsibility-model/

© 2020, Amazon Web Services, Inc. or its Affiliates.

AWS Well-Architected Framework

システム設計・運用の”大局的な”考え方と
ベストプラクティス集

クラウドではWell-Architectedの取り組みにより

責任共有モデルに基づくお客様ワークロードの

設計・運用の継続的改善を推進しています。

皆様の効率的なTransformationをWell-Architected FrameworkとAWS の SA が**お手伝いします**

© 2020, Amazon Web Services, Inc. or its Affiliates.

AWS Well-Architected Framework

システム設計・運用の”大局的な”考え方と ベストプラクティス集


運用性


セキュリティ

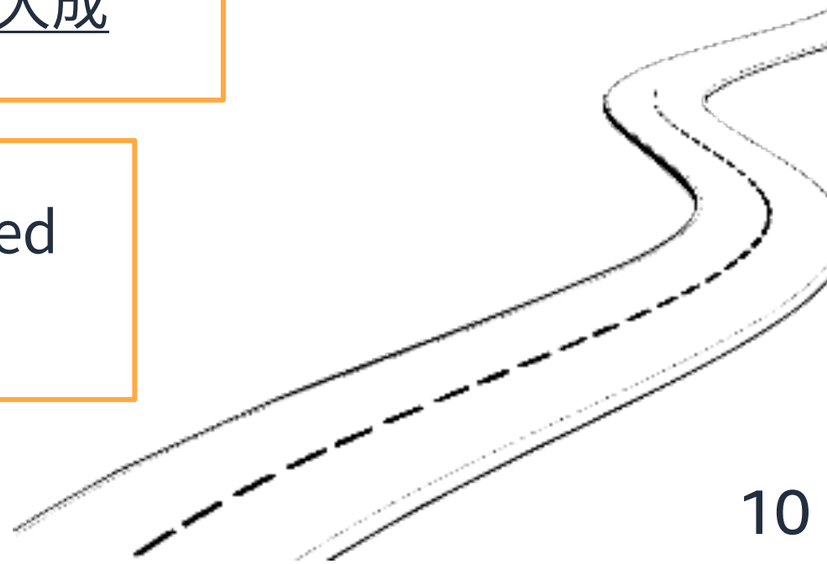

信頼性


パフォーマンス効率


コストの最適化

AWSのソリューションアーキテクト(SA)が、**10年以上**に渡り、**数多くのお客様**へ技術支援をしてきた経験の集大成

皆様の効率的なTransformationをWell-Architected FrameworkとAWS の SA が**お手伝いします**



© 2020, Amazon Web Services, Inc. or its Affiliates.

Well-Architected レビューの効能

設計・構築段階の確認により

- セキュリティや信頼性のリスクを発見
- 最適化や改善すべきポイントが明確に
- アーキテクチャの理解度が高まる

一度だけではなく、定期的に取り組みによって…

- 脆弱性やベストプラクティスとの乖離について
  お客様と認識を共有し発展的な会話をしやすくなる
- ノウハウがお客様社内に蓄積される

【参考】JANOG 45 Meeting 「責任共有モデルと Well-Architectedの取り組み」
https://www.janog.gr.jp/meeting/janog45/application/files/6115/7965/6729/janog45-responseshare-nakashima-01.pdf

© 2020, Amazon Web Services, Inc. or its Affiliates.

ここから本題

クラウド事業者がお客様と実施している
リファレンスやフレームワークを用いた取り組み

© 2020, Amazon Web Services, Inc. or its Affiliates.

信頼性の考え方、方向性

むかしは…

選択するサービスやコンポーネントそのものの
信頼性やサービスレベルが高いかどうか？

いま現在、特にクラウドでは…

信頼性の高いベストなアーキテクチャを
利用者の統制範囲でどう実現していくか？

© 2020, Amazon Web Services, Inc. or its Affiliates.

Well-Architected Framework 信頼性の柱 ホワイトペーパー

信頼性の高いワークロードを実装するための、
詳細なベストプラクティスのガイダンス

一環として分散システムの設計、復旧計画、変更の
処理方法の紹介や解説を含む

**お客様の考えを深めたり、新しいアイデアを導き出したり、
新しい質問に繋げるためのエッセンス**

https://d1.awsstatic.com/whitepapers/ja_JP/architecture/AWS-Reliability-Pillar.pdf

© 2020, Amazon Web Services, Inc. or its Affiliates.

Well-Architected Framework 信頼性の柱 設計5原則

- 障害から自動的に復旧する
- 復旧手順をテストする
- 水平方向にスケールして集合的なワークロードの可用性を高める
- キャパシティを勘に頼らない
- 自動化で変更を管理する

© 2020, Amazon Web Services, Inc. or its Affiliates.

可用性の正確なニーズを特定する

可用性目標の設定は高ければ高いほど良いわけではない

- 通常はアプリケーションの可用性を高く設計すれば労力や費用は増大する
- 網羅的な障害シナリオのもとで厳しいテストおよび検証条件が課される
- 非常に高いレベルの可用性を前提にソフトウェア開発のコストを積み上げると、システムのデプロイメント速度の足枷になる

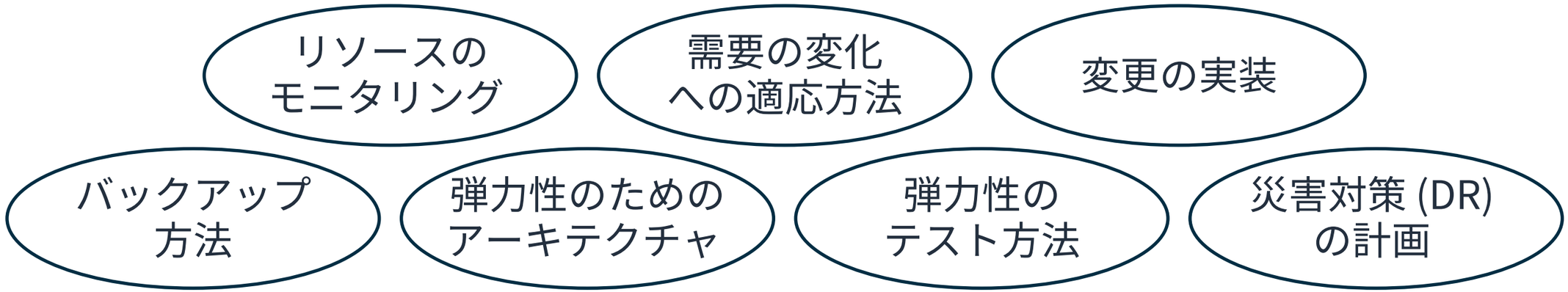
ライフサイクル全体にわたって適切な可用性の目標を設定することが重要

© 2020, Amazon Web Services, Inc. or its Affiliates.

可用性目標からワークロード設計へ

Well-Architectedフレームワーク信頼性の柱 ホワイトペーパー記載の例

可用性	最大利用不可能時間(年間）	アプリケーションのカテゴリ
99%	3 日と 15 時間	バッチ処理、データ抽出、転送、ジョブのロード
99.9%	8 時間 45 分	ナレッジ管理、プロジェクト追跡などの社内ツール
99.95%	4 時間 22 分	オンラインコマース、POS
99.99%	52 分	動画配信、ブロードキャストのワークロード
99.999%	5 分	ATM トランザクション、通信のワークロード



© 2020, Amazon Web Services, Inc. or its Affiliates.

RTO、RPOを設定して、定期的なリカバリテストを実施

目標復旧時間(RTO)
組織のミッションまたはミッション/ビジネスプロセスに悪影響が及ぶ前の、ワークロードのコンポーネントが復旧フェーズにあるため利用できない全体的な時間の長さ

目標復旧時点(RPO)
組織のミッションまたはミッション/ビジネスプロセスに悪影響が及ぶ前の、ワークロードのデータが利用できなくなる可能性がある全体的な時間の長さ

© 2020, Amazon Web Services, Inc. or its Affiliates.

テストの信頼性を高める取り組み

本番環境のストレスに耐えられるようにワークロードを設計した後、ワークロードが意図したとおりに動作し、期待する弾力性を実現することを確認する唯一の方法が、テストを行うことです。

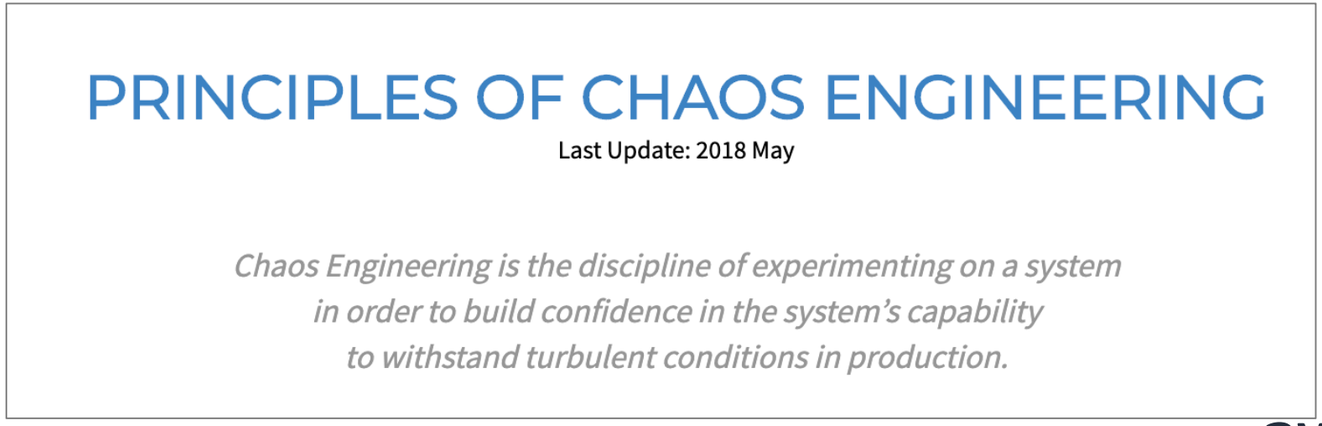
カオスエンジニアリングは実践されているアプローチの一つ

本番環境の分散システムが過酷な状況でも耐えられるとの確信を得るために、意図的に障害を起こし、実験するという取り組み

たとえば、障害を模して

- 経路を切り替えてみる
- インスタンスを停止してみる

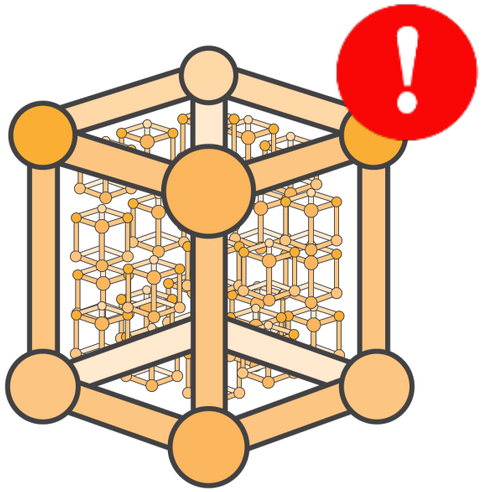
など



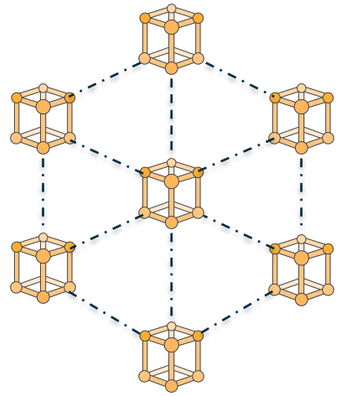
http://principlesofchaos.org/

© 2020, Amazon Web Services, Inc. or its Affiliates.

アーキテクチャの方向性



モノリシック



マイクロサービス

課題

- 1 つのコンポーネントに障害が発生すると、アプリケーション全体に障害が及ぶおそれがある
- ビジネスの進化のスピードにアプリケーションの追随が追いつかない

対応策

- 機能の細分化を決定
- システム間の依存関係を排除
- 個々のサービスで自由にスケール
- 分散型のデータ管理へ移行
- API ベースで運用できる基盤に刷新

© 2020, Amazon Web Services, Inc. or its Affiliates.

ここまでのお話

- 責任共有モデルとWell-Architectedの取り組み
- 信頼性の考え方、方向性

  = ベストなアーキテクチャを利用者の統制範囲で実現していく

    - 可用性の正確なニーズを特定する
    - 可用性目標からワークロード設計へ
    - RTO、RPOを設定して、定期的なリカバリテストを実施
- アーキテクチャの方向性としてのマイクロサービス

© 2020, Amazon Web Services, Inc. or its Affiliates.

サービスレベルを再考しよう

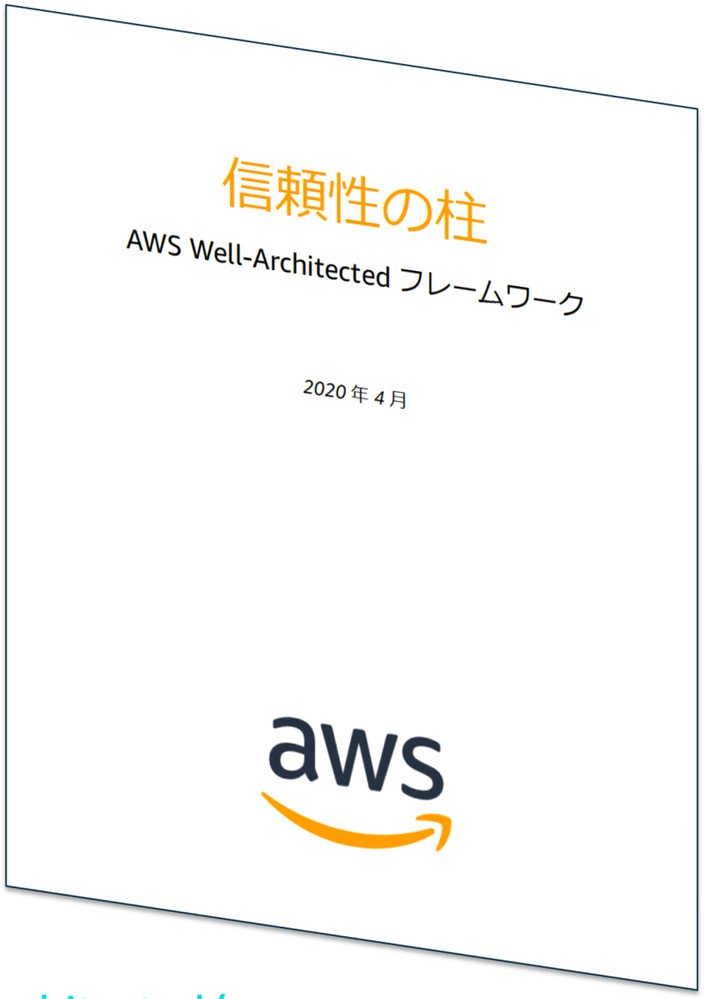
- システム全体の信頼性はお客様と事業者で責任を共有している、これはクラウドに限る話では無い
- よりよいアーキテクチャによってカスタマーサクセスに導くことが技術者の仕事
- お客様に新しい考え方を取り入れてもらうには苦労を伴うが、ベストプラクティスを伝え、実現可能性を顧客と共に確認しながらリードすることが正攻法
- リファレンスやフレームワークを再利用することは効率的なアプローチ

© 2020, Amazon Web Services, Inc. or its Affiliates.

まとめにかえて

AWS Well-Architected Framework
信頼性の柱 ホワイトペーパー

今回ご紹介した内容はその一部です。
お客様の考えを深めたり、
新しいアイデアを導き出したり、
新しい質問に繋げるためのエッセンスを
取り入れてみませんか？



https://aws.amazon.com/jp/architecture/well-architected/
https://d1.awsstatic.com/whitepapers/ja_JP/architecture/AWS-Reliability-Pillar.pdf

© 2020, Amazon Web Services, Inc. or its Affiliates.

Thank you!

© 2020, Amazon Web Services, Inc. or its Affiliates.