FUJITSU

クラウド運用管理

AWS Well-Architected Framework とは ～解説と事例、IPA非機能要求グレードとの比較～

2026年1月
富士通株式会社 ジャパングローバルゲートウェイ 堀口 洋

はじめに

クラウドコンピューティングがビジネス基盤となり、デジタルトランスフォーメーションが加速する現代では、多くの企業がAWSをはじめとするクラウドプラットフォームをすでに導入、あるいは更なる利用拡大を検討しています。一方で、クラウドを最大限に活用し、継続的にビジネス価値を創出するには、単なるワークロードの展開では不十分です。セキュリティの脆弱性、予期せぬコストの増大、パフォーマンスボトルネック、非効率な運用といった課題は、設計段階での見落としに起因することが少なくありません。

これらの既存の課題を解決し、また課題を未然に防ぎ、ビジネス目標達成に貢献するクラウドアーキテクチャを確立するために、AWSが長年にわたり培ってきた知見とベストプラクティスが “**AWS Well-Architected Framework**” です。

Well-Architected Frameworkの6つの柱

Well-Architected Framework は、2024年11月6日版では、以下の６つの ”**柱**” で構成されています。

６つの柱	内容
運用上の優秀性	組織運営とワークロード運用の継続的改善を通じてビジネス価値の最大化を図るベストプラクティス。
セキュリティ	データとシステム保護、脅威検出、インシデント対応でリスク軽減を図るベストプラクティス。
信頼性	予期せぬ障害から回復し、ワークロードが期待通り継続的に機能することを図るベストプラクティス。
パフォーマンス効率	最新技術と適切なリソース選定により、スケーラブルで効率的なクラウドアーキテクチャの実現を図るベストプラクティス。
コスト最適化	クラウド財務管理を実践、費用対効果を追求して継続的な支出の最適化を図るベストプラクティス。
持続可能性	環境への影響を最小限に抑えるためのベストプラクティス。（2021年12月に追加された最新の柱）

※本稿におけるリンクは、2026年1月13日時点で確認しています。
今後の Well-Architected Framework のアップデートにより、リンク先やその内容は変更される可能性があります。

これらの６つの ”**柱**” は、それぞれ独自の観点を持ちながらも、相互に関連し合っています。Well-Architected Framework は、これらの柱に沿ってワークロードを評価する優れた指針となり、改善すべき領域の特定に貢献します。

それでは、Well-Architected Framework の６つの柱を１つ１つ見ていきましょう。

運用上の優秀性のベストプラクティス領域

Well-Architected Framework の "**運用上の優秀性**" は、クラウド環境でのワークロードの運用を効果的かつ効率的に実行し、継続的に改善していくための重要な指針が提供されています。

この柱は、以下の４つのベストプラクティス領域で構成されています。

４つの領域	概要	実践例
1 組織	組織運営の効率性や持続可能性を高めて、ビジネス成果への寄与を図る領域です。 顧客ニーズ評価やガバナンス・コンプライアンス要件評価により組織の優先順位の決定、組織構築において責任と所有権の明確化、組織の文化醸成を通じて適時適切なコミュニケーションやエスカレーションの奨励、メンバーの学習の促進、といった項目が記載されています。	ニーズ/要件の把握、脅威状況の評価、役割分担の明確化、トップの参画、メンバーの学習の奨励、等
2 準備	監視、構成管理やデプロイ管理といった運用設計や、運用体制の構築に関する領域です。 主要業績評価指標の設定から各種テレメトリーの実装、バージョン管理、自動化、設計標準共有、小規模な変更、安全なデプロイとロールバック、人材確保、といった項目が記載されています。	主要KPIの定義、各種テレメトリーでワークロード可視化、変更時のテストと失敗に備えた対応方針策定、等
3 運用	ワークロードの運用状況の把握、運用で生じるイベントの管理を通じて、安定したワークロード運用を図る領域です。 ワークロードのメトリクス、ログ、トレース分析による正常性把握、KPIに基づく実践的なアラートとダッシュボード作成、運用目標測定と可視化、改善の優先順位付け、イベント・インシデント・問題管理のプロセスの確立、アラートごとの対応プロセス、ビジネス影響に基づくイベント優先度決定、イベント発生時のエスカレーションや顧客とのコミュニケーション計画、自動化といった項目が記載されています。	Amazon CloudWatchでKPIに基づく異常検知と通知を自動化、AWS Systems Manager Incident Managerでインシデント管理とランブックの自動実行、等
4 進化	オペレーションの継続的な改善と学習を通じて進化を図る領域です。 継続的改善プロセス、インシデント後分析、フィードバックループ、ナレッジ管理、改善推進要因の定義、メトリクスレビュー、教訓の共有といった項目が記載されています。	継続的改善のプロセス導入、インシデント分析で予防措置検討、ナレッジ管理、等

セキュリティのベストプラクティス領域

Well-Architected Framework の "**セキュリティ**" の柱は、データやシステムの保護、脅威検出、インシデント対応でリスクの軽減を図るといったベストプラクティスが提供されています。

この柱は、以下の７つのベストプラクティス領域で構成されています。

７つの領域	概要	実践例
1 セキュリティ基盤	ワークロードを安全に運用し、脅威やリスクを低減する、セキュリティ基盤の確立を図る領域です。 マルチアカウント戦略、ルートユーザー管理、管理目標の特定と検証、脅威モデルの活用、セキュリティ脅威情報の更新、セキュリティ統制の自動化といった項目が記載されています。	AWS OrganizationsやAWS Control Towerでアカウント分離と共通ガードレール適用、AWS Security Hubで継続的に脅威状況の評価と自動対応、等
2 Identity and Access Management	人とマシンの認証やアクセス権限を安全かつ効率的に管理し、リスクの低減を図る領域です。 MFA等の強力な認証、一時的な認証の利用とシークレット管理、一元化されたIDプロバイダーの活用、認証情報の定期監査とローテーション、最小特権付与といった項目が記載されています。	多要素認証（MFA）の強制、ユーザーやアプリケーションへの最小権限付与、AWS IAM Identity Center を導入してSAML/OIDC連携でID管理を一元化、AWS Secrets Manager を使って、認証情報の安全な保存と自動ローテーション、等
3 検出	セキュリティイベントを検出し、調査と対応を効率化するための仕組み整備に関する領域です。 ログ取得と保存、標準化された場所へのログ等の情報集約、相関付けによるアラート強化、非準拠リソースの修復の手順整備や自動化といった項目が記載されています。	AWS CloudTrailとAmazon CloudWatch Logsでログ取得、Amazon S3にログ集約し一元管理、AWS Security Hubで検出した非準拠のリソースの自動修復、等

セキュリティのベストプラクティス領域（続き）

7つの領域（続き）	概要	実践例
4 インフラストラクチャの保護	ネットワークおよびコンピューティングリソースを保護し、脅威や不正アクセスを防ぐための領域です。 ネットワーク分離とトラフィック制御、トラフィック検査と対応自動化、脆弱性管理、強化イメージの利用したプロビジョニング、手動管理削減、ソフトウェア整合性検証、コンピューティングリソースの保護の自動化、といった項目が記載されています。	セキュリティグループやネットワークACLによるアクセス制御、AWS WAFによるWebアプリケーション保護、Amazon Inspectorで脆弱性をスキャンしてイメージを強化、等
5 データ保護	データの分類や、保管時および転送時のデータ保護等、データのセキュリティに関する領域です。 データ分類とデータのライフサイクル管理、機密性に基づくデータ保護統制、データのアクセス制御、暗号化キー管理、保管時および転送時のデータ暗号化といった項目が記載されています。	Amazon S3バケットへデータアップロード時の自動暗号化適用、HTTPSを用いた通信暗号化、等
6 インシデントへの対応	インシデント発生に備えた体制整備や対応準備、インシデント対応後の学習に関する領域です。 関係者の特定、インシデント対応計画や手順書の整備、フォレンジック機能実装、アクセス権やツールの事前準備、定期的な訓練の実施、インシデント発生後の教訓の活用による再発防止策の検討、といった項目が記載されています。	インシデント対応計画策定、インシデント対応用プレイブック作成、インシデント調査ツールの事前デプロイ、等
7 アプリケーションのセキュリティ	設計、開発、デプロイのライフサイクル全体を通じてアプリケーションのセキュリティ確保を図る領域です。 セキュリティ訓練、テスト自動化、定期的なペネトレーションテスト、コードレビュー、パッケージと依存関係の一元化、CI/CDでデプロイ、CI/CDのパイプラインでセキュリティ特性の評価、開発者の意識向上、といった項目が記載されています。	AWS CodePipelineでCI/CD実装により自動デプロイ、Amazon CodeGuru SecurityやAmazon Inspectorでパイプラインのセキュリティ特性を継続的評価、等

信頼性のベストプラクティス領域

Well-Architected Framework の “**信頼性**” の柱は、ワークロードが期待通りに機能し、インフラストラクチャやサービスでの予期せぬ負荷や障害から迅速に回復し、必要に応じて動的にリソースを獲得できる能力を確保するためのベストプラクティスが提供されています。

この柱は、以下の４つのベストプラクティス領域で構成されています。

４つの領域	概要	実践例
1 基礎	信頼性を維持するためのサービスクォータとネットワーク設計の基盤の確立に関する領域です。 クォータと制約の認識/管理、クォータの監視、ネットワーク冗長化、パブリック接続の高可用性、拡張性/可用性を考慮したサブネット設計、ハブアンドスポーク構成といった項目が記載されています。	状況に応じたクォータ上限引き上げ申請の実施、Amazon CloudFront活用でパブリック接続の高可用性確保、等
2 ワークロードアーキテクチャ	ワークロードのアーキテクチャ設計と分散システムの障害対策を通じて、信頼性向上を図る領域です。 ビジネスドメイン重視のサービス設計、ワークロードの疎結合な設計やマイクロサービス化、べき等性の確保、スロットリング、再試行制御、フェイルファストとキューの制限、タイムアウト設定、ステートレス化、緊急対応策実装など、障害の予防・緩和・耐性向上に向けた設計指針に関する項目が記載されています。	ワークロードをAWS LambdaやAmazon ECSのコンテナでセグメント化、Amazon SQSで非同期処理、Amazon API Gatewayでリクエストのスロットルを管理、等
3 変更管理	システム変更に伴う影響を最小限に抑え、可用性を維持するための管理手法に関する領域です。 モニタリングの設計と改善、アラームのリアルタイムな通知と自動対応、ログ分析、リソース取得やスケーリングの自動化、機能/回復力テスト、イミュータブルなインフラの導入、デプロイにランブック活用や自動化、といった項目が記載されています。	CI/CDパイプラインで自動デプロイ、本番環境の変更前のステージング環境でのテスト実施、等
4 障害管理	障害の影響を最小限に抑え、迅速な復旧と継続運用に関する領域です。 SLA設計、バックアップの自動化と暗号化、障害分離やDR戦略の検討、復旧テスト、フェイルオーバーなどリカバリ自動化、定期的な障害対応訓練の実施やカオスエンジニアリングなど、障害対応力を高めるための仕組みや運用手法に関する項目が記載されています。	マルチAZ等で障害分離、アラーム時のインスタンス自動再起動、フェイルオーバーテストの定期実施、等

パフォーマンス効率のベストプラクティス領域

Well-Architected Framework の “**パフォーマンス効率**” の柱は、コンピューティングリソースの効率的な利用を図り、変化する要件やスケーリングに対応する能力を実現、維持するためのベストプラクティスが提供されています。
この柱は、以下の５つのベストプラクティス領域で構成されています。

５つの領域	概要	実践例
1 アーキテクチャの選択	クラウド環境で最適なパフォーマンスを実現するためのアーキテクチャ選定を支援する領域です。 クラウドサービス理解、専門家の知見活用、コスト考慮、トレードオフ評価、ポリシー/リファレンスアーキテクチャ利用、ベンチマーク比較、データ駆動型アプローチで意思決定、といった項目が記載されています。	AWS Well-Architected Framework のガイダンスを活用、等
2 コンピューティングとハードウェア	クラウド環境で最適なコンピューティング資源を選定・活用するための領域です。 インスタンスタイプの選定、メトリクス収集、ライトサイジング、動的スケーリングといった項目が記載されています。	CPU集約型ワークロードでC系インスタンス選択、Amazon EC2 Auto Scalingで需要連動型スケーリング設定、等
3 データ管理	ワークロードに最適なストレージ選定と、ストレージのパフォーマンス向上に関する領域です。 データ特性に応じたストア選定、設定評価、メトリクス収集、クエリの性能向上、キャッシュ活用といった項目が記載されています。	低レイテンシーのキーバリューアクセスにAmazon DynamoDB活用、等
4 ネットワークとコンテンツ配信	ネットワーク構成とコンテンツ配信の最適化に関する領域です。 マネージドサービス選定、VPNや専用線の活用、適切なプロトコルやロードバランシング等のルーティング選定、最適なロケーション選定、メトリクス分析による設定の最適化といった項目が記載されています。	Webの静的コンテンツへのAmazon CloudFront活用、等
5 プロセスと文化	組織全体でパフォーマンス意識を高め、継続的な評価とパフォーマンス改善の文化の醸成を目的とする領域です。 KPI設定とモニタリング、改善プロセスの整備、負荷テスト、定期的なメトリクスの見直しといった項目が記載されています。	定期的なロードテスト実施によるボトルネック特定、パフォーマンス低下のアラート設定、等

コスト最適化のベストプラクティス領域

Well-Architected Framework の “**コスト最適化**” の柱は、クラウド財務管理を実践、費用対効果を追求し、クラウドでの支出を継続的に最適化するするためのベストプラクティスが提供されています。
この柱は、以下の５つのベストプラクティス領域で構成されています。

５つの領域	概要	実践例
1 クラウド財務管理を実践する	コスト最適化と財務管理の仕組みを組織的に整備するための領域です。 責任者の設定、財務部門と技術部門の連携、予算と予測、プロセスへのコスト意識の導入、異常検知と通知、プロアクティブなモニタリング、最新情報把握、文化醸成、ビジネス価値の数値化など、継続的なコスト管理と最適化に関する項目が記載されています。	AWS Budgetsで予算を設定、AWS Cost ExplorerでAWSコストを可視化、AWS Trusted Advisorで コスト最適化の推奨を確認、等
2 経費支出と使用量の認識	クラウド利用におけるコストと使用状況の管理・最適化を継続的に実践するための領域です。 ポリシー策定、目標設定、メトリクスの確立、コスト管理ツール設定、不要リソース削除のプロセス化などの項目が記載されています。	AWS Cost Explorerによる日々のコストモニタリング、タグ付け戦略導入によるコストの分類、等
3 費用対効果の高いリソース	クラウドサービスの選定やリソース構成の場面で、費用対効果とコスト最適化を追求する領域です。 コスト予測に基づいた最適な構成と料金モデル選択、共有リソース検討、データ転送最適化といった項目が記載されています。	一時的な処理ではスポットインスタンス、長期稼働のデータベースへリザーブドインスタンス適用、等
4 需要を管理しリソースを供給する	ワークロードの需要変動に応じてリソース供給の最適化を図る領域です。 季節的傾向など考慮した需要分析、バッファリングやスロットリングの実装、動的リソース供給による過不足の抑制といった、リソースの利用効率を高めるための設計と運用に関する項目が記載されています。	Amazon EC2 Auto Scalingで負荷状況に合わせたインスタンス自動調整、夜間や週末での開発環境の自動停止、等
5 継続的最適化	クラウド環境におけるサービス評価と運用効率の継続的な最適化を図る領域です。 定期レビュー、新サービスの導入可否や既存サービスの置き換え検討、リアーキテクトの可能性評価、運用工数削減や人的エラー低減に資する運用作業の自動化の推進、といった項目が記載されています。	AWS Compute Optimizerなどを用いた定期的なリソース最適サイズ確認と推奨事項への変更、等

持続可能性（サステナビリティ）のベストプラクティス領域

Well-Architected Framework の “**持続可能性（サステナビリティ）**” の柱は、エネルギー効率の向上や、リソース利用の最適化等を通じて環境負荷を低減し、クラウドワークロードが環境へ及ぼす影響を最小限に抑えることを目指すベストプラクティスが提供されています。

この柱は、以下の６つのベストプラクティス領域で構成されています。

６つの領域	概要	実践例
1 リージョンの選択	ワークロードの全体的な環境負荷低減のため、持続可能性の高いAWSリージョンを選択する領域です。 カーボンフットプリント削減や再生可能エネルギー利用に注力しているリージョン選択について記載されています。	再生可能エネルギー利用に注力したAWSリージョンでワークロード展開、等
2 需要に合わせた調整	クラウドリソースをワークロードの需要に応じて柔軟に調整、最適化するための領域です。 動的スケーリング、持続可能性を勘案したSLA、地理的配置の最適化、未使用リソースの削除、運用作業用リソースの最適化、バッファリング/スロットリングの実装、といった項目が記載されています。	Amazon EC2 Auto Scalingでインフラを動的スケール、Amazon CloudFrontで地理的配置を最適化、等
3 ソフトウェアとアーキテクチャ	持続可能性目標に向けてソフトウェア（利用者アプリケーション）とアーキテクチャの最適化を図る領域です。 非同期ジョブとスケジュールジョブの最適化、未使用リソースの削除、高コストなコードの最適化、顧客が利用するデバイスや機器の最適化、データのアクセスに最適なストレージやアーキテクチャの選定といった項目が記載されています。	Amazon ECR における未使用イメージの自動クリーンアップ、等
4 データ	持続可能性目標に向けて効率的なデータ管理を実践するための領域です。 データ分類、データのアクセスに最適なストレージやアーキテクチャの選定、ライフサイクル管理、不要データの削除、共有ストレージの活用、ネットワーク間のデータ移動抑制、バックアップデータの最小化など、ストレージ効率とエネルギー消費削減に関する項目が記載されています。	低アクセスのデータをAmazon S3 Glacier ストレージクラスへ移行、冗長なデータ複製の回避、等
5 ハードウェアとサービス	持続可能性目標の達成に向けてハードウェアとサービス選定の最適化を図る領域です。 最小限のハードウェア使用、エネルギー効率のよいインスタンスタイプ選定、マネージドサービス活用、アクセラレーター活用といった項目が記載されています。	最新のインスタンスタイプ（Gravitonなど）への移行検討、GPUを含むインスタンスタイプの選定、等
6 プロセスと文化	持続可能性の取り組みを推進/定着させるための組織文化やプロセスに関する領域です。 持続可能性の目標の周知と浸透、持続可能性の改善を迅速に導入できるプロセス整備、ワークロードの最新状態維持、ビルド環境利用率の向上、マネージド型Device Farmでのテストといった項目が記載されています。	持続可能性目標の共有、最新のインスタンスタイプの適用検討、CI/CDの使用率増加、等

Well-Architected Framework の活用ステップ

Well-Architected Framework の基本的な使い方は、以下の3つのステップで構成されます。

1．レビュー（評価）

ワークロードの現状を、Well-Architected Framework の6つの柱に基づいて評価します。

2．改善（計画と実施）

レビューで特定された改善機会やリスクに対し、優先順位をつけ、具体的な改善計画を立てて実行します。

作業項目	概要
1 優先順位付け	リスクレベル（高リスク、中リスクなど）やビジネスへの影響度に基づいて、対応すべき課題を決定します。
2 改善計画の策定	専門家からのアドバイス等を参考に、具体的なアクションプランを作成します。 これには、技術的な実装だけでなく、運用プロセスの変更や組織的な取り組みも含まれる場合があります。
3 実装	計画に基づき、ワークロードやプロセスの改修を行います。一貫性と効率性を高めることを目論み、 実装においては IaC（Infrastructure as Code）を活用して自動化を推進することが推奨されます。

3．継続的な改善（維持）

クラウド環境は常に変化し、ビジネス要件も進化します。そのため、Well-Architected Framework による評価と改善は、一度実施したら終わり、ではありません。**定期的にレビューを実施し、継続的にワークロードを最適化**していきます。

Well-Architected Framework の活用タイミング

Well-Architected Framework は、ワークロードのライフサイクルの様々な段階で活用できます。
活用にあたって効果的なタイミングを以下にまとめます。

タイミング	効果的である理由
1 **新規ワークロードの設計・構築時** クラウドで新しいアプリケーションやシステムをゼロから設計する際、またはPoC（概念実証）を行う前のタイミングです。	最初から Well-Architected Framework のベストプラクティスを勘案して設計することにより、後工程の大規模な手戻りを防ぎ、手戻りのコストとリスクを大幅に削減できます。正しい基盤を早期に確立できます。
2 **オンプレミスからクラウドへの移行計画時** 既存オンプレミスシステムをAWSへ移行する際の計画段階のタイミングです。	移行後のクラウド環境が、ビジネス要件を満たすだけでなく、クラウドのメリットを最大限に享受しかつ運用しやすい状態になるよう事前に設計を評価・調整できます。 移行戦略全体を Well-Architected Framework の観点から見直すことで、費用対効果の高い移行を実現できます。
3 **既存ワークロードの定期的な評価・最適化時** 稼働中の重要なワークロードについて、定期的（例: 四半期や半年に一度）、または大きな変更（例: 機能追加、リファクタリング）を行う前のタイミングです。	ビジネス環境や技術は常に変化するため、一度最適化されたシステムでも時間の経過とともにギャップが生じ得ます。 定期的なレビューにより、常に最新のベストプラクティスを適用し、パフォーマンス、セキュリティ、コストなどの継続的な最適化を図ります。
4 **具体的な課題が発生した時** 特定の運用上の問題（例: システムのレスポンス低下、予期せぬ高額請求、セキュリティ脆弱性の発見）が発生したタイミングです。	Well-Architected Framework の6つの柱は、これらの問題の原因特定と、解決策の発見に役立つ体系的なフレームワークを提供します。 問題のある特定の柱に焦点を当ててレビューすることで、根本原因を突き止め、効果的な改善策を導き出すことができます。
5 **監査やコンプライアンス要件への対応時** 業界の規制への対応や、社内監査の準備をするタイミングです。	Well-Architected Framework は、多くのコンプライアンス要件と整合したセキュリティのベストプラクティスが豊富に含まれています。 これらのベストプラクティスを考慮した設計実施と文書化は、監査対応をスムーズにする助けとなります。

Well-Architected Framework は "**完璧な設計**" を１回で作るためのものではなく、"**より良い設計を継続的に追求する**" ためのツールとプロセスを提供します。常に改善を意識して、適切なタイミングで Well-Architected Framework を積極的に活用することが、クラウド活用の成功の鍵と言えるでしょう。

Well-Architected Framework の活用事例①

それでは、Well-Architected Framework の活用事例を見ていきましょう。
次の事例は、クラウド移行時に、運用管理のミドルウェアを検討する際の事例になります。

事例：商用ミドルウェアの利用を継続するか、置き換えるかの検討

オンプレミス環境からのAWS移行では、商用ミドルウェアのオープンソース化やマネージドサービスへの変更は、多くの案件で検討される事例になります。
Well-Architected Framework のベストプラクティスでは、以下が該当します。

柱	領域	質問	ベストプラクティスとリスク		かいつまんだポイント	IPA非機能要求グレード区分
コスト最適化	費用対効果の高いリソース	COST 5 サービスを選択するときは、どのようにコストを評価するのですか?	COST05-BP04 コスト効率の高いライセンスを提供するソフトウェアを選択する	低	ソフトウェアを使うワークロードで、商用ソフトウェアではなく、積極的にオープンソースソフトウェアの活用でコスト削減を図っているか、といった問いかけです。 たとえば、Windowsの利用よりもLinuxがコスト効率が良いというような内容が言及されています。	IPA非機能要求グレード範囲外
			COST05-BP05 組織の優先順位に従ってコストが最適化されるようにこのワークロードのコンポーネントを選択する	中	オープンソースソフトウェア、あるいはライセンス料金が不要なサービス活用で支出の抑制を図っているかしているか、といった問いかけです。 商用ミドルウェアではなく、マネージドのサービス利用を検討する、例えば、OracleからAmazon Aurora PostgreSQLに移行、といった対応が想定されます。	IPA非機能要求グレード範囲外

よく検討の議題に上がる商用のミドルウェアの一つに、運用管理を担う**“ジョブ管理”**があります。AWSでも、**AWS Step Functions** というジョブ管理的な機能を持つマネージドサービスがあります。このサービスは、ジョブ管理の置き換えでよく検討の遡上に上りますが、**機能で比較すると細かいところに差異**がありますので注意が必要です。機能の検討例では、ジョブの再実行や、祝日や営業日などのカレンダー対応があります。一例で、**ジョブの再実行**について見ていきましょう。

ジョブの再実行では、**AWS Step Functions** でも **Redrive** という機能があります。この機能により、失敗したワークフローを再実行することが可能です。ただし、失敗した箇所からの再実行は出来ますが、あるジョブが失敗した場合に任意のジョブから再実行というような柔軟な制御は出来ません。ジョブ管理のミドルウェアでは、異常終了したジョブが含まれるジョブネットで、様々な再実行の方法が提供されています。

💡 富士通のジョブ管理ミドルウェア **Fujitsu Software Systemwalker Operation Manager** および、ジョブ管理ミドルウェアをベースにSaaSサービスとして提供する **Fujitsu Workload Operations Integrator** では、多くのお客様との間で長年培ってきたジョブ管理の知見をもとに、ジョブ異常時の対応に柔軟にフィットする複数のジョブ再実行方式をご用意しています。

Systemwalker Operation Manager V17.1.1 活用ガイド（2025年8月版）第2章 運用・監視編
2.1 異常終了したジョブネットの対処をしたい

Fujitsu Workload Operations Integrator ユーザーズガイド ジョブ管理サービス編（2025年10月版）
14.5 ジョブをリカバリ操作する

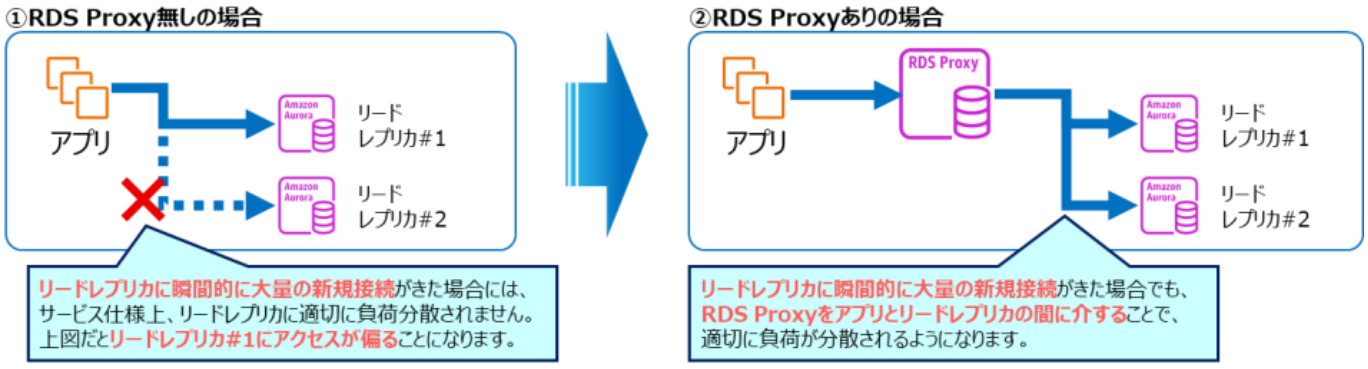
機能単位での検討に加えて、ミドルウェアの変更に影響を受ける**ジョブの数**といった観点も注意が必要です。ジョブネット数が数百、ジョブ数が数千から万の単位となるケースもあります。また、既存のジョブは、**冪等性などの機能を商用ミドルウエアにオフローディング**することで、ジョブを簡易に作っているケースもあります。このような観点を鑑み、商用ミドルウェアの継続利用（あるいは商用ミドルウェアから発展したSaaSへの移行）と、ジョブを AWS Step Functions 対応に作り替えとで、初期費用と運用費用を合わせたトータルコストで比較することが、コスト最適で費用対効果の高いリソースの検討で重要となります。

Well-Architected Framework の活用事例②

次の事例は、パフォーマンスで具体的な課題が発生した際の事例になります。

事例：Amazon Aurora の リードレプリカ が適切に負荷分散できていない

ある案件で、Amazon Auroraでリードレプリカを利用していましたが、同時に大量にアクセスがあった際に、リードレプリカが適切に負荷分散されていないことがわかりました。
Well-Architected Framework で観点を確認の上、サービスの仕様を確認したところ、Amazon Aurora ではリードレプリカで同時大量アクセスを適切に負荷分散することは難しいことが判明しました。
そこで最新の機能を確認、Amazon RDS Proxyの導入により、同時大量アクセスでも適切に負荷分散できることがPoCで検証できました。



出典：富士通 AWSコラム
2024年3月22日 Amazon Auroraでアプリごとにリードレプリカを分けつつオートスケールも設定したい

こちらの事例で確認し、改善ポイントとして抽出した Well-Architected Framework のベストプラクティスを以下にまとめます。

柱	領域	質問	ベストプラクティスとリスク		かいつまんだポイント	IPA非機能要求グレード区分
信頼性	変更管理	REL 7 需要の変化に対応するためには、どのようにワークロードを設計すればよいですか。	REL07-BP01 リソースの取得またはスケーリング時に自動化を使用する	高	オートスケール（スケールアウト/スケールイン）を活用しているか、といった問いかけです。 ➡○ リードレプリカを活用する構成は出来ていました。	B.3.5 サーバ処理能力増強
パフォーマンス効率	データ管理	PERF 3 ワークロード内のデータはどのように保存、管理、アクセスすればよいでしょうか?	PERF03-BP02 データストアで利用可能な設定オプションを評価する	中	ストレージのオプションを、適切に設定/見直ししているか、といった問いかけです。 ➡× 今回は、リードレプリカに同時に多数アクセスした場合の仕様の理解が不足していました。	B.3.3 ディスク拡張性
	ネットワークとコンテンツ配信	PERF 4 ワークロード内のネットワークリソースはどのように選択して構成すればよいでしょうか?	PERF04-BP04 ロードバランシングを使用してトラフィックを複数のリソースに分散する	高	ロードバランサーを使っているか、ロードバランサーで暗号化を終端しているか、といった問いかけです。 ➡× 今回は、リードレプリカの負荷分散ができる Amazon RDS Proxyが採用できていませんでした。最新の機能調査の観点で、Amazon RDS Proxy の知識が不足していました。	B.3.4 ネットワーク
	アーキテクチャの選択	PERF 1 ワークロードに適切なクラウドリソースとアーキテクチャを選択するにはどうすればよいでしょうか?	PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する	高	今後の改善として、以下の項目を運用作業に追加することを推奨しました。 • アーキテクチャに関する最新機能の継続的な調査 • 改善に向けた評価/検討	IPA非機能要求グレード範囲外

Well-Architected Framework の活用の仕方

事例①②での使い方のように、Well-Architected Framework のレビューでは、毎回すべてのベストプラクティスを参照する、という使い方をする必要はありません。必要に応じてベストプラクティスを抽出したり、全ベストプラクティスを使用したりと、**活用タイミングによって使い分け**を行うことが肝要です。

なお事例②の案件では、Well-Architected Framework のベストプラクティスとIPA非機能要求グレードとを合わせて、効率的にレビューを行っています。

この後は、日本の文化として根付く IPA非機能要求グレード と Well-Architected Framework とを比較し、Well-Architected Framework がどのようにまとめられているかを見ていきましょう。
まずは、IPA非機能要求グレードの概要について次の節に記載します。

IPA非機能要求グレードとは

IPA非機能要求グレードは、以下の目的で作成されたツール群です。

> **非機能要求グレードとは**
> 「非機能要求グレード」は、「非機能要求」についてのユーザと開発者との認識の行き違いや、互いの意図とは異なる理解を防止することを目的とし、非機能要求項目を網羅的にリストアップして分類するとともに、それぞれの要求レベルを段階的に示したものです。重要な項目から順に要求レベルを設定しながら、両者で非機能要求の確認を行うことができるツール群です。

出典：システム構築の上流工程強化（非機能要求グレード）紹介ページ | アーカイブ | IPA 独立行政法人 情報処理推進機構

IPA非機能要求グレードは、現在はIPAサイトでは更新が終了しておりますが、商談ではRFP（提案依頼書）などで現在でも活用されており、日本のシステム開発における標準的な枠組みとして定着しています。

IPA非機能要求グレードは、以下の区分で記載されています。

No	大区分	主な項目
1	可用性	継続性（目標復旧水準、稼働率など）、耐障害性、災害対策、回復性
2	性能・拡張性	業務処理量（通常業務量、業務量の増大度など）、性能目標値（レスポンス、スループットなど）、リソース拡張性（CPU/メモリ/ディスク/ネットワーク拡張性、サーバ処理能力増強）、性能品質保証（帯域保障、リソース専有、性能テスト、スパイク負荷対応）
3	運用・保守性	通常運用（バックアップ、運用監視など）、保守運用（計画停止、運用作業自動化、パッチ適用など）、障害時運用（復旧作業、障害復旧の自動化など）、運用環境（開発環境/テスト環境、リモート運用）、サポート体制、その他の運用管理方針（インシデント管理、変更管理など）
4	移行性	移行スケジュール、移行方式、移行対象（設備、データ量、移行媒体、データ変換量/変換ツール）、移行計画（作業分担、リハーサル、トラブル対処）
5	セキュリティ	前提条件/制約条件（社内規定、業界標準など）、セキュリティリスク分析/管理、セキュリティ診断、アクセス/利用制限（認証、権限管理など）、データの秘匿（通信経路やデータ保管先での暗号化）、不正追跡/監視（ログ監視など）、ネットワーク対策（アクセス制御、不正検知、DoS攻撃の回避など）、マルウェア対策、Web対策（セキュアコーディング、WAF導入）、セキュリティインシデント対応/復旧
6	システム環境・エコロジー	システム制約/前提条件、システム特性、適合規格、機材設置環境条件、環境マネージメント

Well-Architected FrameworkとIPA非機能要求グレードを組み合わせた活用方法

Well-Architected Framework は、多くのベストプラクティスをIPA非機能要求グレードの区分で理解を深めることが可能です。
一例として、"**スケールアップ、スケールアウト**" 関連のベストプラクティスを比較してみましょう。
IPA非機能要求グレードでは、項番 "**B.3.5 サーバ処理能力増強**" にてスケールアップやスケールアウトに関する項目があります。
ただスケールアップやスケールアウトといっても、評価軸としては、パフォーマンスの面もあればコスト効率化の面もあります。
Well-Architected Framework では、以下の表のように、**複数の柱、複数の領域にわたって、様々な角度から**スケールアップやスケールアウトに関連するベストプラクティスを提供しています。

柱	領域	質問	ベストプラクティスとリスク	
信頼性	変更管理	REL 7 需要の変化に対応するためには、どのようにワークロードを設計すればよいですか。	REL07-BP01 リソースの取得またはスケーリング時に自動化を使用する	高
			REL07-BP02 ワークロードの障害を検出したときにリソースを取得する	中
			REL07-BP03 ワークロードにより多くのリソースが必要であることを検出した時点でリソースを取得する	中
パフォーマンス効率	コンピューティングとハードウェア	PERF 2 コンピューティングリソースを選択し、ワークロードで使用するにはどうすればよいでしょうか?	PERF02-BP05 コンピューティングリソースを動的にスケールする	高
コスト最適化	経費支出と使用量の認識	COST 4 リソースはどのように廃止するのですか?	COST04-BP04 自動的にリソースを廃止する	低
	費用対効果の高いリソース	COST 6 コストターゲットに合わせて、リソースタイプ、リソースサイズ、およびリソース数を選択するには、どうすればよいですか?	COST06-BP03 メトリクスに基づいて自動的にリソースタイプ、リソースサイズ、リソース数を選択する	低
持続可能性	需要に合わせた調整	SUS 2 クラウドリソースを需要に合わせる方法	SUS02-BP01 ワークロードインフラストラクチャを動的にスケールする	中

このように、**ある観点について、様々な角度で評価軸を提供**してくれるのが Well-Architected Framework の特徴です。
一方、ワークロードをレビューする手順としては、**観点をまとめて、系統立ててレビュー**できると効率的です。
そこで私たち富士通では、Well-Architected Framework のベストプラクティスとIPA非機能要求グレードとをマッピングし、
Well-Architected Framework を IPA非機能要求グレードの記載順で系統立ててレビューできる資材を整備しています。
これによって、**どちらの観点でも漏れなく、かつ効率的に**お客様のワークロードをレビューできるよう取り組んでいます。

Well-Architected Frameworkレビューのご相談について

富士通は、**AWS Well-Architected パートナー**として認定されて以降、Well-Architected Framework を活用しています。
AWS上のワークロードで Well-Architected Framework レビューを実施したい等ございましたら、お気軽にご相談ください。

本コンテンツに関するお問い合わせは
■ クラウド運用管理 > 相談・問い合わせ
https://global.fujitsu/ja-jp/local/software/cloud-operation

富士通株式会社