aws modern apps

AWS でモダンアプリケーションを構築する

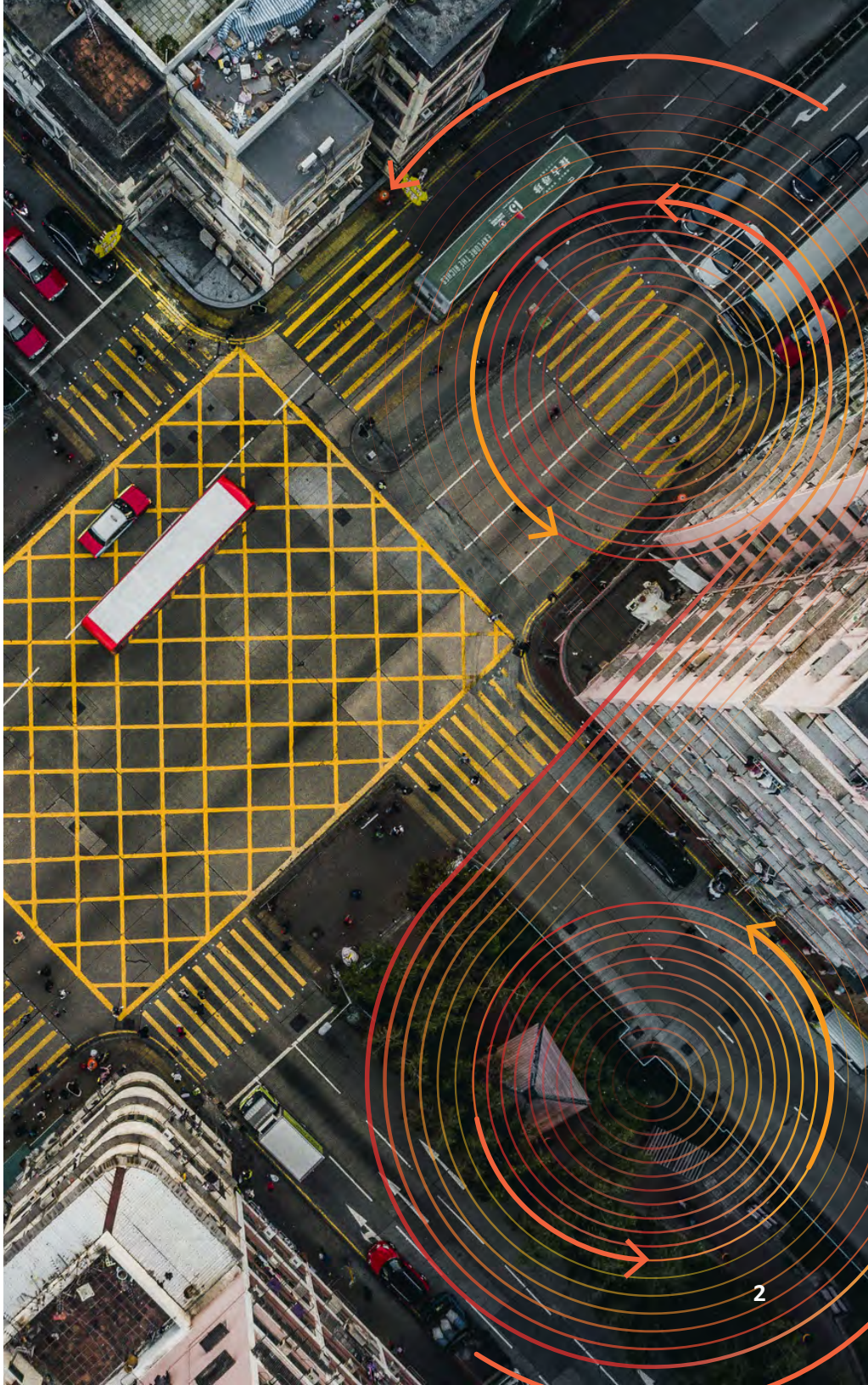
管理の軽減。迅速な構築。さらなるイノベーション。

モダンアプリケーションによって顧客価値を提供する方法が変化している

今後数年間で新たに構築されるアプリケーションは 5 億を超え、過去 40 年間で開発されたアプリケーション数の合計よりも多くなります[1]。多くの組織では、技術の管理と新機能の提供とのバランスを取ることに苦労しながらアプリケーションを構築しています。

クラウドは俊敏性を約束するものの、俊敏性が自動的に実現するわけではありません。イノベーションの加速、データの活用、新しいカスタマーエクスペリエンスの構築を目指す組織では、アプリケーションの構築とオペレーションの方法をモダナイズする必要があります。モダンアプリケーションは、モジュール式のアーキテクチャパターン、サーバーレスの運用モデル、俊敏なデベロッパープロセスの組み合わせで構築します。

この日本語ガイドでは、お客様の組織でモダンアプリケーション開発の基盤を築くために役立つ 3 つのパスをご案内します。また、AWS を使ったモダンアプリケーション開発が組織のイノベーション、コスト削減、市場投入までの時間短縮、信頼性の向上にどのように役立つのかを説明します。



aws modern apps

[1] *IDC PlanScape: Connected Cloud Architectures to Enable the Future of Digital Infrastructure、2020 年 10 月*

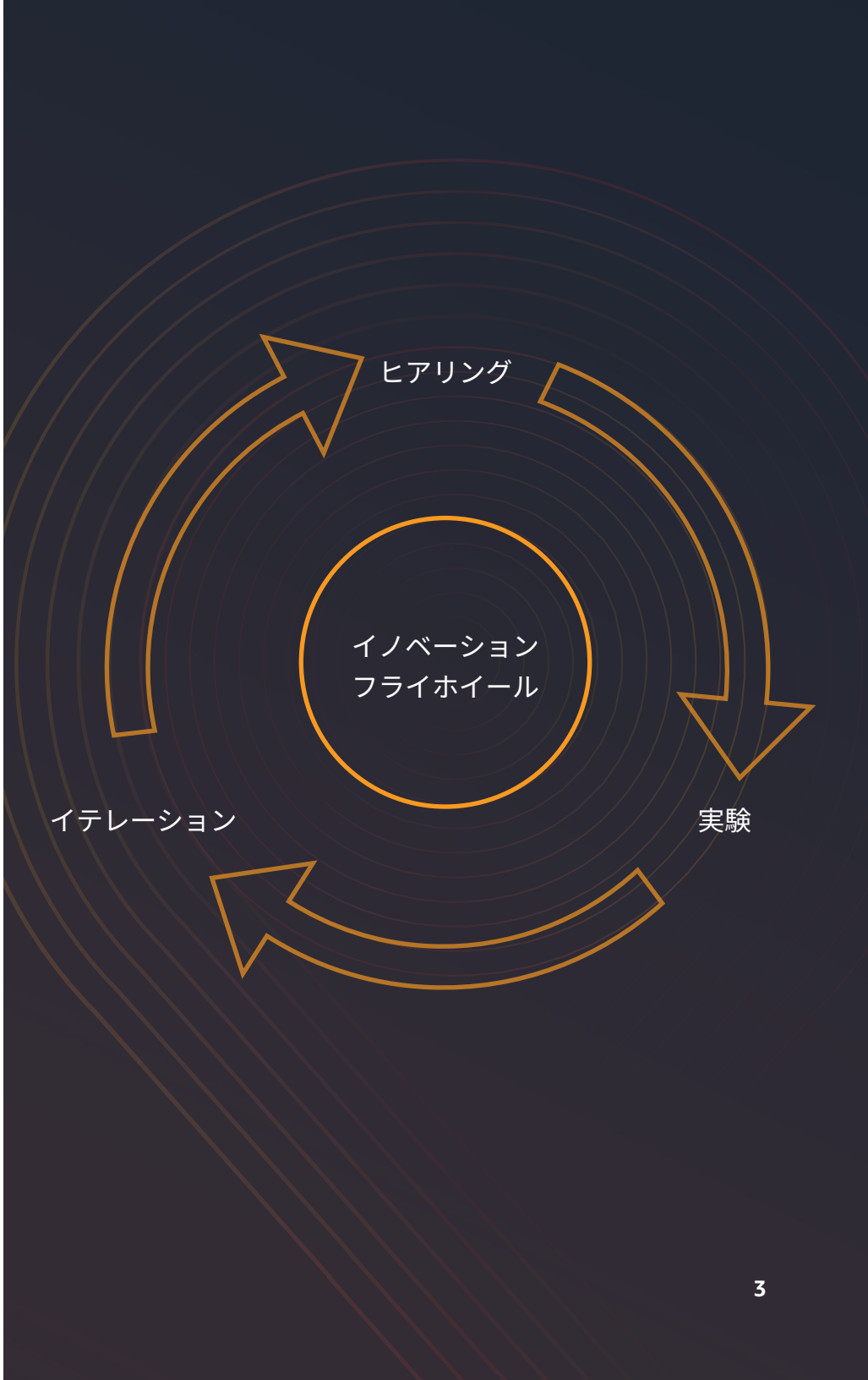
イノベーションとは
顧客の意見を聞くこと

Forrester Research は最近発行したビジョンレポート「*Digital Rewrites the Rules of Business*」の中で、デジタルイノベーターの顧客中心の思考方式を定義付けました。以下の抜粋では、こうした最近のイノベーターの基本的な使命を説明しています。

「…デジタルアセットとエコシステムを活用して、常により良い結果を顧客に提供すると同時に、カスタマーエクスペリエンス、オペレーション、エコシステム、イノベーションにデジタル思考を採り入れることで、運用性をさらに向上させます」。

顧客を重視することとは、顧客の視点に立って逆算しながらビジネス上の決断を下すことです。また、満足度が高い成果を顧客に提供できるように製品やサービスの向上に絶えず取り組むことを意味します。さらに、顧客が本当に大切にしていることに耳を傾け、顧客に代わって発明とイテレーションを行い続けられるようにすることを意味します。これは「イノベーションのフライホイール」と呼ばれます。

そのベーシックな概念は、イノベーションの推進力は顧客の需要から始まり、顧客からのフィードバックで改善され、継続的に (かつ収益が上がる形で) 繰り返され、需要が変化するとサイクル全体がもう一度開始されるというものです。チームが独自のイノベーションフライホイールを早く回転できるほど、モダンアプリケーションの構築が進み、競合他社よりも優位な立ち位置を確保できます。



AWS でモダンアプリケーションを構築すれば市場投入を迅速に行えます。構築およびリリースのサイクルを高速化し、運用上の間接諸経費を軽減することにより、デベロッパーは新しい機能を迅速に構築できます。モジュール式のアーキテクチャによって、アプリケーション全体をリスクにさらすことなく、チームが個々のアプリケーションコンポーネントを実験することができ、イノベーションが加速します。開発ライフサイクルの各段階のテスト手順とモニタリングを自動化することで、信頼性を向上させることができます。従量課金制の料金体系によって、過剰なプロビジョニングやアイドル状態にあるリソースへの支払いコストを削減し、総保有コスト (TCO) を改善することができます。

モダンアプリケーションを構築するには、既存のアプリケーション基盤を再検討する必要があるかもしれません。アーキテクチャの変更は組織にとっては劇的な変化ではありますが、一気に舵を切る必要はありません。クラウドで新しいモダンアプリケーションを構築するために飛躍的な変貌を遂げる組織も多いですが、その他の多くの組織ではハイブリッドなアプローチによってチーム別、ワークロード別に一歩ずつ様子を見ながら進めます。

50%

デジタルトランスフォーメーションを実現するために 2023 年までに割り当てられることが予想される情報通信技術の割合

67%

競争力を維持するためには今よりスピードを上げる必要があると考える経営幹部の割合

90%

2025 年までにクラウドネイティブになることが予想される新たなアプリケーションの割合

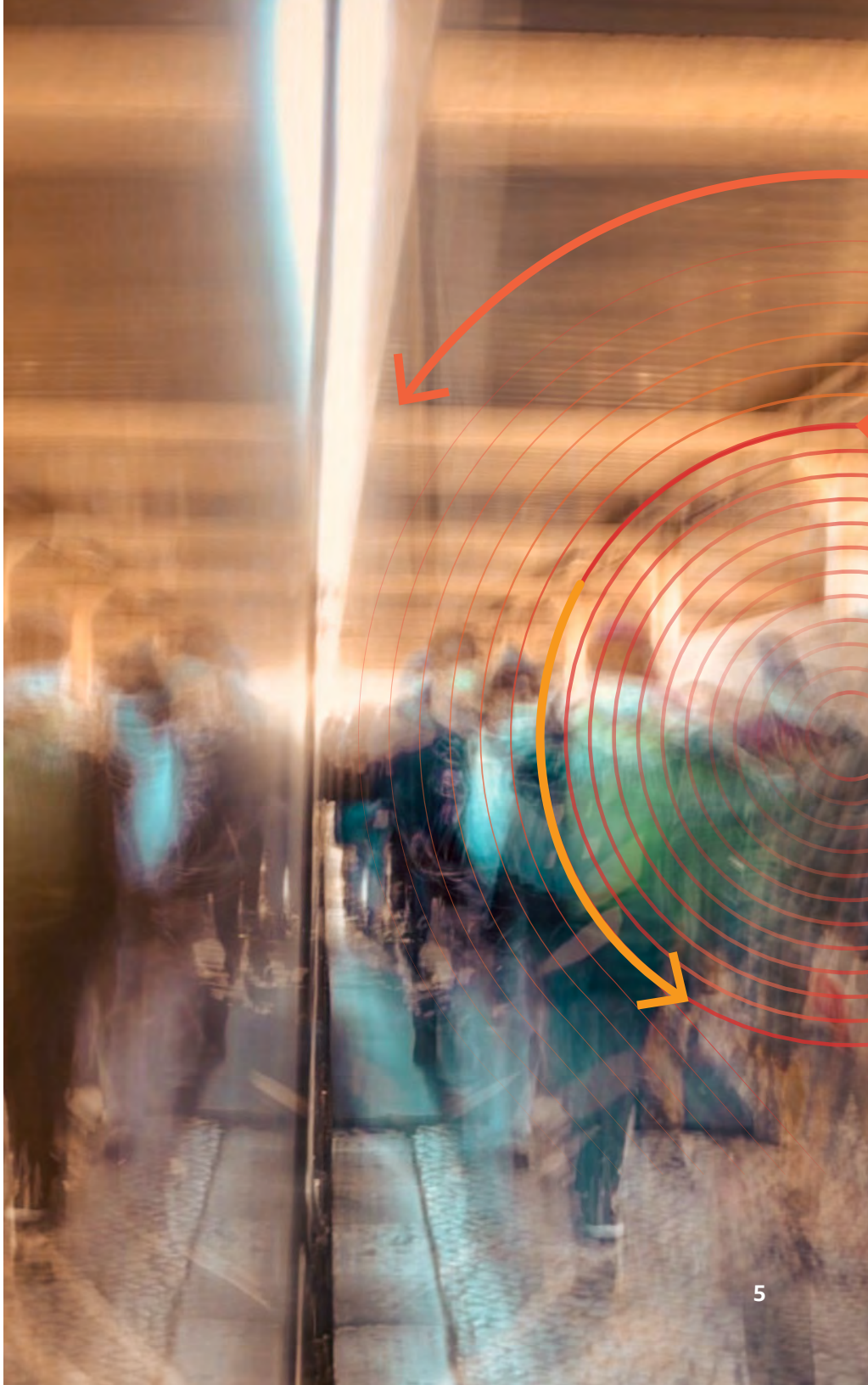
Amazon.com のアプリケーションを構築し、何百万もの AWS のお客様にサービスを提供してきた当社の実績から、アプリケーションをモダナイズするビジョンを現実のものにし、その工程でビジネス価値を生み出すために実行すべき 3 つのパスを解明しました。

1 マネージドコンテナサービスへのリプラットフォーム。既にオンプレミスでコンテナを実行中の組織やアプリケーションをコンテナに移行することを検討している組織では、ワークロードを AWS のコンテナサービスにリプラットフォームすることでオペレーションを簡素化し、オーケストレーションやインフラストラクチャのプロビジョニングなどの間接管理コストを削減できます。

2 新しいアプリケーションをサーバーレスアーキテクチャで構築。新しいアプリケーションや機能を構築する場合は、サーバーレス技術と目的別データベースを使用して俊敏性を最大化し、最新の開発ツールを使って開発を加速することをお勧めします。

3 モダン Dev+Ops モデルへの変換。企業文化の変化をもたらすような大規模なモダンアプリケーションを構築するには、セキュリティとガバナンスを高い水準で維持しつつ、DevOps サービスとツールを活用します。

俊敏性の強化、コストの削減、ビジネスの成功に寄与する優れたアプリケーションの構築にどのように役立つかを示しながら、それぞれのパスについて詳細に説明します。モダンアプリケーションへの移行に際しては、スタート地点はそれぞれ違っても、ゴールはみな同じでなければなりません。安全性、信頼性、拡張性があり、お客様やパートナーがすぐ利用できるアプリケーションがゴールです。



モダンアプリケーション開発への 3 つのパス

モダンアプリケーション開発により、クラウド内でのソフトウェアの設計、構築、管理を効率的に進めることができます。この実証済みの方法により、開発チームの俊敏性、アプリケーションの信頼性とセキュリティが軒並み向上するため、高機能製品をより短期間で構築およびリリースすることが可能となります。組織のあらゆる種類のアプリケーションの構築をお手伝いしてきた当社の経験から、モダンアプリケーション開発の 3 つのソリューションの柱をご紹介し、お客様のモダナイゼーションをお手伝いします。

モダンアプリケーションへのパス

1. マネージドコンテナサービスへのリプラットフォーム
2. 新しい、セキュアなアプリケーションをサーバーレスアーキテクチャで構築
3. モダン Dev+Ops モデルへの変換

マネージドコンテナサービスへのリプラットフォーム

コンテナは、アプリケーションを実行し、デプロイするための軽量でポータブルな手段です。多くの場合、モダナイゼーションの最初のステップは、既存のアプリケーションをコンテナ化することです。アプリケーションをコンテナに移行することを検討している場合は、AWS Fargate で Amazon Elastic Kubernetes Service (Amazon EKS) や Amazon Elastic Container Service (Amazon ECS) などのAWSマネージドサービスへワークロードをリプラットフォームするとよいでしょう。マネージドコンテナサービスは、スケーラビリティ、信頼性、セキュリティ、可用性を向上させながら、運用上の負担を軽減します。AWS のマネージドコンテナサービスによって、コンテナの管理を気にする必要がなくなる分、サーバーレスコンピューティングを用いたモダンアプリケーション開発に必要なスキルアップのためのトレーニングにリソースを集中できます。

Vanguard®

「AWS Fargate Spot によってユニットコストが下がった結果、AWS への移行をさらに後押しすることになりました。この最適化によって、当社では毎月より多くの資産価値を提供し続けています。効率性の向上によって得られた価値を株主に還元することは、当社全体のミッションの中核となるものです」

– Vanguard クラウドビジネスオフィス、シニアマネージャー、Tim Treston 氏

2015 年に AWS に移行を開始して以来、Vanguard Group では大きな躍進を遂げました。例えば、AWS のサービスを利用することで自社の IT 部門でサーバーを管理する必要がなくなりました。デベロッパーには革新的な新しいマイクロサービスを構築したり、現在のアプリケーションを強化したりするための時間が増え、市場投入までのスピードが 3 か月から 24 時間に短縮されました。

しかし、同社が得られたメリットは市場投入までのスピードだけではありません。Vanguard では、AWS Fargate と並ぶフルマネージドのコンテナオーケストレーションサービスである Amazon Elastic Container Service (Amazon ECS) の使用を選択しました。AWS Fargate はサーバーレスコンピューティングサービスの 1 つで、クラウド上で、安全でリサイズ可能なコンピューティング性能を提供するウェブサービスである Amazon Elastic Compute Cloud (Amazon EC2) で必要なプロビジョニングや管理が不要になります。さらに、AWS Fargate 用の新しいオプションを購入して活用することで、この金融サービス会社ではユニットコストを 50% 削減しました。

詳しいストーリーを読む »

アプリケーション開発のモダナイゼーションとは、デベロッパーがアプリケーション構築の俊敏性を高めることができるようにするサービス、プラクティス、戦略を導入することです。また、モダンインフラストラクチャのスピードと信頼性によって、デベロッパーはプロトタイプから数百万人のユーザーまでオートスケーリングするセキュアなアプリケーションを提供できるようになるため、イノベーションを加速し、変化に迅速に対応できるようになります。

多くのモダンアプリケーションはサーバーレスファーストで構築されています。これはお客様がアプリケーションスタック全体の俊敏性を高めるために、サーバーレスの導入を優先する戦略です。サーバーレス技術によって、物理サーバーの管理が必要なくなり、オートスケーリングの実現、高可用性が最初から組み込まれている、従量課金制請求などのメリットが得られます。またサーバーやランタイムの管理と運用に煩わされることなく、製品のイノベーションに集中して、市場投入までの時間を短縮することができます。

さらに、ウェブやモバイルのフロントエンドツールとサービスも、AWS に構築することができます。このインフラストラクチャの信頼性によって、企業は世界中で自動的にスケーリングでき、安全で可用性の高いアプリケーションの提供を実現することができます。

スケーラブルなモダンアプリケーションを構築するための主な検討事項

アーキテクチャのパターン：マイクロサービス

モノリシックアプリケーションは現時点では管理が簡単かもしれませんが、企業規模が拡大すると、アプリケーションの責任権限の割り振り方など、さまざまな課題が生じます。オーナーシップの文化を強固にしてもよいのですが、最終製品に所有者意識を持つのを避けたくなるような依存がアプリケーションアーキテクチャ自体にあると、規模を拡大しづらくなります。拡大と変更を迅速に行うアプリケーションに対し、私たちがマイクロサービスアーキテクチャの構築を勧める理由はそこにあります。マイクロサービスとは、オーナーシップの文化をアーキテクチャで表現したものです。複雑なアプリケーションをコンポーネントに細分し、各部分を 1 つのチームが単独で所有、運営できるようにします。

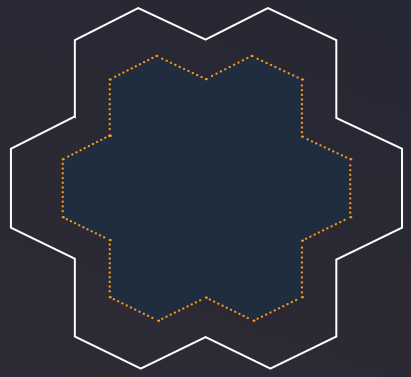
モノリシックアプリケーションでは、多数のデベロッパーが共有のリリースパイプラインにすべての変更をプッシュするため、ライフサイクルのあちこちで衝突が生じます。開発中、現場のエンジニアは他のデベロッパーのコードを壊さないように変更を調整する必要があります。新しい機能を活用するには、共有ライブラリを全員が同時にアップグレードする必要がありますが、その説得は簡単ではありません。また、機能の重要な修正プロセスをすぐに進める必要がある場合でも、その修正を進行中の変更とマージする必要があります。

開発後に配信パイプラインを通じて変更をプッシュする場合、間接諸経費の問題にも直面します。コードをわずか 1 行変更する場合でも、エンジニアには煩雑な作業が待っています。例えば、変更を早めに調整し、コードをマージして、リリース内の競合を解決し、アプリ全体を再構築し、一連のテストを実行し、デプロイし直すといった具合です。

マイクロサービスアーキテクチャを持つアプリケーションは、各アプリケーションプロセスを 1 つのサービスとして実行する独立したコンポーネントで構成されます。各サービスはビジネス機能向けに構築され、それぞれ 1 つの機能を実行します。各サービスは独立して実行され、1 つの開発チームによって管理されるため、アプリケーションの特定の機能に合わせてアップデート、デプロイ、スケールすることができます。例えば、オンラインショップのカートはセール期間中により多くのユーザーに利用される可能性があります。マイクロサービスでは、軽量の API、イベント、ストリームを使用して、詳細に定義されたインターフェイス経由で相互にデータがやり取りされます。アプリケーションのスケーラビリティと回復力の向上およびコスト削減を両立するために、AWS のお客様は、データの変更に反応してアクションがトリガーされるイベント駆動型アーキテクチャを利用する割合が急速に高まっています。

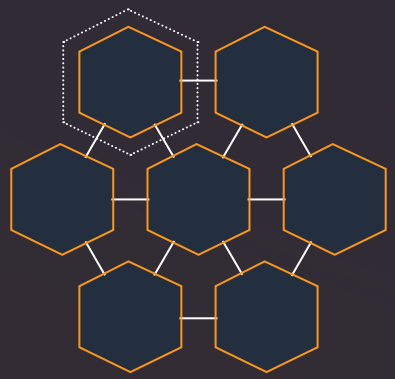
総合型 vs 専用型：2 種類のアプリケーション

モノリスアプリケーション



- すべてを実行
- 単一アプリケーション
- アプリ全体のデプロイが必要
- 単一データベース
- テクノロジーレイヤーを中心に構成
- 各ランタイムインスタンスに存在
- アプリケーション全体に 1 つのテクノロジースタック

マイクロサービス



- 1 つのことを実行
- 最小限の機能サービス
- 個別にデプロイ、連携して機能
- それぞれ独自のデータストアを所有
- ビジネス能力を中心に構成
- 状態を外部化
- マイクロサービスごとにテクノロジーを選択可能

エネルギーとサービスを提供する英国の多国籍企業 Centrica では、アプリケーションのアーキテクチャを設計する方法を変更することで、コストを削減し、俊敏性を高めようと考えていました。同社はこれらの目標を達成するため、マイクロサービスアーキテクチャにリファクタリングし、サーバーレス戦略を採用することにしました。

組織を変革するために、Centrica ではサーバーレス作業グループを設置し、代表チームと協力してパイロットの構築をスタートしました。この成功を受けて、他のチームも同じアプローチを採用したため、今ではサーバーレスは組織全体に浸透しています。

Centrica ではサーバーレスを採用したことで、以前はできなかった顧客の問題をリアルタイムで確認し、対応することができるようになりました。

動画を見る »

サーバーレス運用モデル

サーバーレスを徹底

アーキテクチャのパターンとソフトウェアの配信プロセスを変更すると、ビジネスの主力機能以外の活動を排除できる運用モデルの採用を検討するようになるでしょう。迅速なイノベーションを可能にする俊敏性を手に入れるためには、マイクロサービスアーキテクチャを構築し、アプリケーションのモニタリング、プロビジョニング、コスト管理、デプロイ、セキュリティ、ガバナンスなどを自動化して、ソフトウェアを運用およびデプロイすることをお勧めします。サーバーレスファースト戦略を採用し、可能な限りサーバーレステクノロジーを選ぶことで、AWS での運用のメリットを最大限に高めることができます。

サーバーレスの運用モデルでは、サーバーをプロビジョニングまた管理することなく、アプリケーションやサービスの構築と実行が可能になります。そのため、サーバーの管理が不要になり、柔軟にスケールできます。また従量課金制が採用され、高可用性が自動化されます。このモデルでは基礎的な細部に煩わされることなく、お客様に価値を提供するアプリケーションの構築および管理に集中することができます。

まったく新しいアプリケーションを構築する、レガシーアプリケーションを移行する、など、どのような目的であれ、コンピューティング、データ、インテグレーションのために基礎的なサーバーレス環境を構築すると、クラウドによってもたらされる俊敏性の恩恵を最大限に受けることができます。

AWS でのサーバーレスの定義

AWS では、サーバー運用という画一的で面倒な作業を取り除くことを「サーバーレス」と呼んでいます。サーバーレスが重要な理由は、アプリケーションをサポートするためのインフラストラクチャの管理や規模の拡大ではなく、アプリケーションの構築自体に集中できることにあります。サーバーレスの運用モデルに関する原則を 4 つ紹介します。

1. **サーバー管理不要** — サーバーのプロビジョニングや維持が不要です。インストール、維持、管理を要するソフトウェアやランタイムがありません。

2. **柔軟なスケーリング** — アプリケーションを自動的に、あるいは、サーバー単位ではなく利用単位 (スループット、メモリなど) を切り替えて容量を調整することで、スケールできます。

3. **従量課金制** — サーバー台数でなく、一貫したスループットや実行時間といった利用分に対してのみ支払いが発生します。

4. **自動化による高可用性** — サーバーレスアプリケーションには高可用性と耐障害性が組み込まれています。これらの機能は、アプリケーションを実行するサービスによってデフォルトで提供されるため、お客様がアーキテクチャを設計する必要はありません。

サーバーレス運用モデルは急速なイノベーションが必要な高成長企業に最適です。サーバーレスにすることで、作業ペースが上がり、ビジネスを差別化する活動にのみ常に照準を合わせ、イノベーションフライホイールを高速で回転できます。

AWS Lambda と AWS のマネージドコンテナサービスの活用

コンテナおよびサーバーレスコンピューティングの台頭によって、クラウドコンピューティングの選択肢はインスタンスだけではなくなりました。モダンアプリケーションに最適なコンピューティングを選択するために、最初にいくつか確認することがあります。インフラストラクチャを自己管理することで、ビジネスの成果は向上しますか? そのための専門知識は持っていますか? その労力に見合った価値を生み出すと思いますか?

サーバー管理をオフロードすることを選択するお客様は増えており、そのために Amazon ECS および Amazon EKS といったコンテナサービスや、AWS Lambda のようなイベント駆動型サーバーレスコンピューティングサービスを導入しています。

現実には、ほとんどのお客様は両方を組み合わせて使用しています。約 80% の AWS コンテナのお客様は AWS Lambda も導入しています [2]。両方のオプションを活用することで、AWS インフラストラクチャとの高度な統合が可能なフルマネージドサービス、幅広いユースケースのサポート、複雑性からの抽象化、幅広いパートナーエコシステムなど、数々のメリットがあります。

[2] *Datadog State of Serverless、2020 年*

どのように決断しますか?

お客様が AWS Lambda を選択するのは、コードを記述することに主に注力しているチームがあり、既存のインスタンスやコンテナプラットフォームに制限がない場合です。AWS Lambda ではインフラストラクチャからの抽象度を最大化しているため、お客様は最速でリリースできます。そのため、AWS Lambda は新しいアプリケーションに最適です。

お客様がコンテナを選択するのは、多くの場合、既存のコンテナに投資をしている、オープンソースの Kubernetes を好んでいる、またはインフラストラクチャの管理や設定について特定の要件がある場合です。コンテナはコードをパッケージする最も一般的な方法であり、レガシーアプリケーションのモダナイゼーションには最適な選択です。

	AWS が管理	お客様が管理	
AWS Lambda サーバーレス関数	データソースの統合 物理ハードウェア、ソフトウェア、ネットワーク、設備 プロビジョニング	アプリケーションコード	
AWS Fargate サーバーレスコンテナ	コンテナのオーケストレーション、プロビジョニング クラスターのスケーリング 物理ハードウェア、ホスト OS/ カーネル、ネットワーク、設備	アプリケーションコード データソースの統合 セキュリティの設定と更新、ネットワークの設定	管理タスク
Amazon ECS/ Amazon EKS サービスとしてのコンテナ管理	コンテナのオーケストレーションコントロールプレーン 物理ハードウェア、ネットワーク、設備	アプリケーションコード データソースの統合 セキュリティの設定と更新、ネットワークの設定、ファイアウォール	作業クラスター 管理タスク
Amazon EC2 サービスとしてのインフラストラクチャ	物理ハードウェア、ネットワーク、設備	アプリケーションコード データソースの統合 スケーリング 管理タスク	セキュリティの設定と更新、ネットワークの設定 サーバーのプロビジョニング、管理、スケーリング、パッチ適用

AbiBird®

「AWS で運用している当社のテクノロジーにより、ますます多くの高齢者が自宅で自立して幸せに暮らすことができるようになると考えています」

– ATF Services および AbiBird、CEO (最高経営責任者)、Robin Mysell 氏

AbiBird は、オーストラリアとニュージーランドに 60 の支店を持つオーストラリアの企業グループである ATF Services が完全所有する部門です。同社では 350 人の正社員と契約社員が勤務しています。AbiBird では、家庭向けの赤外線センサーで構成されるサービスを提供しています。このセンサーを設置すると、スマートフォンにインストールしたアプリから高齢者の動きを見守ることができます。

AbiBird では、以前は Microsoft Azure で運用していましたが、サービスを運用し続けるためにクラウドプロバイダーに対して非常に多くのサポートチケットを作成していることに気付きました。同社では、より安定したスケーラビリティの高いクラウドを必要としていました。Abibird は AWS に移行し、現在はニーズに基づいてコンピューティングサービスの組み合わせを使用しています。使いやすく、スケーラビリティが高く、マネージド型であることから、バックエンドで AWS Lambda を使用しています。

AbiBird では AWS のコンテナサービスも使用しています。Amazon ECS で公開 API をホストし、仮想マシン群を自社で管理することなく AWS Fargate を使用してコンテナを実行しています。

2019 年に AWS でこのシステムを立ち上げてから、AbiBird ではサポートチケットを 1 枚も作成する必要がありませんでした。最小限の間接費のサポートで効率的に運用できるようになったのです。

詳しいストーリーを読む »

米国を代表する飲食ブランドの 1 つである Taco Bell は、米国内に 7,000 店を超えるレストランを展開しています。新型コロナウイルス感染症がまん延している間、Taco Bell ではお客様からのデリバリーの要望に応えるために、迅速にシフトする必要がありました。エンジニアリングおよび分析部門バイスプレジデントの Vadim Parizher 氏によると、Taco Bell では、ほぼすべてのインフラストラクチャを Amazon Web Services (AWS) で運用しており、AWS のサーバーレスを活用することで、サーバー管理よりも、ビジネスロジックやデータトランスフォーメーションの構築に集中して、メニューやレストランの情報をリアルタイムでデリバリーパートナーに提供することに力を入れています。「当社のメニューは、非常に複雑で複数のデジタルチャネルで共有する必要があります。サーバーレスはそのモデルに最適です」と、Parizher 氏は述べました。サーバーレスサービスを利用することで Taco Bell は初期コストを削減することができ、小規模でスタートして必要な分だけを支払い、サービスの利用が増えれば自動的にスケーリングすることができます。

動画を見る (英語) »

新型コロナウイルス感染症による打撃で、消費者の習慣は一夜のうちに変わりました。Coca-Cola では迅速に対応し、革新的な Freestyle ドリンクディスペンサーに、非接触で購入できる機能を加えました。Coca-Cola では、構築に AWS Lambda を使用することを選択し、その結果、チームはセキュリティ、レイテンシー、スケーラビリティよりもアプリケーションに集中できました。

AWS Lambda には、それらがすべて組み込まれているからです。新しいアプリケーションをわずか100 日で立ち上げ、現在では 52,000 台を超えるマシンに非接触機能が備わっています。

詳しいストーリーを見る (英語) »

「ユーザーエクスペリエンスには低レイテンシーが不可欠であり、それが AWS のサーバーレスソリューションに当社がコミットした理由です」

– Coca-Cola Freestyle 装置イノベーションセンター、チーフアーキテクト、Michael Connor 氏

モダン Dev+Ops は、文化的な理念、プラクティス、およびツールの組み合わせであり、組織が迅速かつ安全にソフトウェアを開発して、それを本番環境にリリースし、目標とする可用性とパフォーマンスが維持できるようにするものです。

AWS では一連の一般的で広く受け入れられている参考事例を特定し、これを導入することで、パフォーマンスの高い DevOps 組織を構築するためのメカニズムを提供しています。このアプローチでは継続的改善というシンプルなアイデアが、計画からコードの記述、デプロイおよびモニタリングまで、DevOps ライフサイクルのすべてに採用されています。

当社ではこのアプローチをモダン Dev+Ops と呼んでおり、コンプライアンス、可観測性、レジリエンス、インフラストラクチャなどのオペレーションタスクを開発プロセスの早い段階で共有し、さらに AI と機械学習 (AI/ML) を使って強化することで、デベロッパーとオペレーション部門との距離を縮めることに焦点を当てています。

デベロッパーの俊敏性：抽象化、自動化、標準化

マイクロサービスアーキテクチャを採用することで、俊敏性が上がり、これまでよりも迅速に動くことができます。つまり、リリースするべきものの構築がはかどるようになります。これは良いことなのですが、構築とリリースのプロセスがチームのペースに追いつかない場合、新しい機能を迅速に顧客に提供することができません。従来の開発プロセスとリリースパイプラインの遅れの主な原因は、手動プロセスとカスタムコードでした。カスタムコードによりエラーの余地が生まれるとともに、長期のメンテナンスが必要になるため、カスタムコードは長期にわたり負担となります。コードの変更や構築リクエストからテスト、デプロイまで、手動ステップはリリース速度にブレーキをかける最大の要因です。これを解決するには、抽象化、自動化、標準化が必要です。開発プロセスをスピードアップするには、本番対応のアプリケーションの開発および配信に必要なコードをできる限り抽象化します。特に、ビジネスロジックコード以外の行を抽象化します。抽象化を行うために、複雑なリソースのプロビジョニングと設定を軽減するフレームワークとツールを採用するのが 1 つの方法です。これにより、デベロッパーは迅速に作業できるだけでなく、開発プロセス全体でセキュリティ、プライバシー、信頼性、パフォーマンス、可観測性、拡張性のベストプラクティスを採用できます。開発フレームワークを利用すれば、ビジネスの成長を長期的にサポートするアーキテクチャを構築している確信を持つことができます。

ベストプラクティスのテンプレートに沿ってソフトウェアの配信プロセスを定義すると、クラウド環境におけるすべてのインフラストラクチャリソースのモデル作成とプロビジョニングを標準化できます。この「Infrastructure as Code」テンプレートを使用すると、チームは幸先の良いスタートを切ることができます。このテンプレートでは手動プロセスでなくコードを通じてアプリケーションのテクノロジースタック全体が提供されるためです。

オートメーションを通じて反復可能な動きを作成し、ソフトウェアの配信ライフサイクルを加速できます。継続的インテグレーションと継続的デリバリー (CI/CD) を通じてリリースパイプラインを自動化すると、良質のコードを迅速かつより頻繁にリリースするのに役立ちます。CI/CD を実践しているチームでは、より多くのコードをより迅速にリリースし、問題にすばやく対応しています。実際、Puppet の State of DevOps レポートによると、CI/CD プラクティスを採用するチームの障害率は採用していないチームの 5 分の 1 にとどまり、コミットからデプロイまでのスピードは 440 倍速く、デプロイの頻度は 46 倍高くなっています。[3] 中でも注目すべきは、プロセスやツールを管理する代わりに新しい機能とコードの作成にかける時間が 44% も長くなっていることです。

CI/CD パイプラインはモダンアプリケーションの新たな構築の場となっています。Amazon では、リリース速度を加速するために CI/CD の利用をスタートして大きな成果をあげています。年間デプロイ件数は数百万に達しており、毎年急速に拡大しています。AWS では、当社の経験をお客様に活用していただけるように、当社で使用しているツールをベースに、コードの迅速な提供に役立つ開発ツール一式を構築しました。

補足情報

継続的インテグレーション — (CI) とは、デベロッパーが定期的にコードに対する変更をセントラルリポジトリにマージし、その後自動的に構築してテストが実行されるソフトウェア開発手法です。継続的インテグレーションは、ほとんどの場合ソフトウェアリリースプロセスのビルドまたはインテグレーションの段階を指し、自動化要素 (CI やビルドサービスなど) および文化的要素 (例えば、頻繁に統合について学ぶこと) を必要とします。

継続的デリバリー — (CD) は、コードの変更によって本番環境へのリリースを自動的に準備するソフトウェア開発のプラクティスです。継続的デリバリーでは、ビルド後にテスト環境または本番環境にすべてのコード変更をデプロイすることで、継続的インテグレーションを拡張します。

Amazon がどのようにデプロイを自動化し、安全かつ手動操作なしでデプロイしているのかについてはこちらをご覧ください »

[3] *Puppet 2020 State of DevOps Report、2020 年*

カナダの Lululemon Athletica は AWS を利用することにより、数日どころか数分で開発環境を立ち上げ、その環境を自動化し、継続的インテグレーションとデプロイを実現しました。同社は、ヨガ関連アパレルなどの衣料を世界中にある 350 以上の拠点で販売しています。Lululemon は開発環境とテスト環境、および近日中にリリースされるモバイルアプリケーション環境を AWS クラウドで運用しています。

Lululemon では、AWS CloudFormation テンプレートと AWS CodePipeline を使用して、新しい本番稼働用アカウント構築の時間を、2　日から数分に短縮しました。俊敏性が高まった今、Lululemon　の開発チームは、実験を試みながら最適なソリューションを入手できるようになり、リソースを確保済みのソリューションで妥協する必要がなくなりました。

詳しいストーリーを読む »

「継続的インテグレーションとデプロイパイプラインは自動化され、管理が簡単で、誰にでもわかりやすくあるべきです。AWS はそのすべてを兼ね備えています。オンプレミス環境で今まで得られなかったレベルのシンプルさと透明性を実現しています」。

– Lululemon、プロダクトアーキテクチャディレクター、Sam Keen 氏

HyperTrack は、アプリケーションでライブ位置情報を追跡するためのセルフサービスのクラウドプラットフォームです。HyperTrack では、2015 年後半の設立時に、デベロッパーが新しい機能の構築に使える時間を減らすことなく、自動的にスケールして、予想される成長に対応できるプラットフォームを構築する必要がありました。

HyperTrack では、エンジニアリングが関与しなくても自動的にスケールアップおよびスケールダウンするために、モバイル開発フレームワークとサーバーレスアーキテクチャに AWS Amplify を使用することに決めました。

その結果、同社では、サーバーレスに切り替える前に使用していたアーキテクチャと比べて、30% のコスト削減を実現できました。このコスト削減の大部分は、運用リソースをサーバー管理に投入しなくて済んだことが要因です。HyperTrack では、数百万のイベントを管理しながらも、毎週 40 時間の作業時間を節約しています。

詳しいストーリーを読む »

オーナーシップが根付く文化を育む：管理の低減、モダン Dev+Ops でさらなるイノベーション

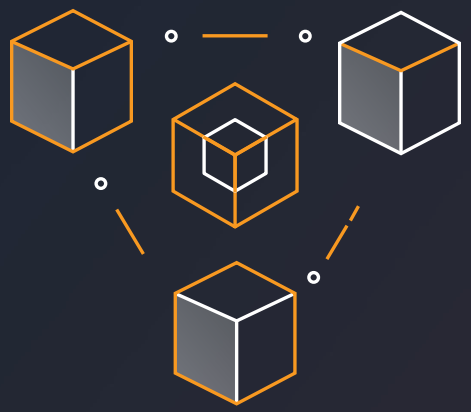
イノベーションの根源にあるのは人材です。したがって、顧客により良い成果を提供するために人材に権限を与えることがモダンアプリケーション開発の出発点になります。AWS では、権限を与えることがチームのあり方に与える影響の大きさを説明するのに、「プロジェクトでなく製品」という概念を使用しています。単純に言うと、製品を構築するチームが製品の動作と保守に責任を負うということです。

責任の所在を明確にすることで、一部でなく製品全体の開発に対してチームが責任感を抱くようになります。

AWS はスケーラビリティに優れたウェブアプリケーションである Amazon. com の構築および運用に 10 年以上携わってきました。その経験から、チームに自己裁量権を与える重要性を学んできました。アプリケーションには、顧客の意見の採用、ロードマップの計画、アプリケーションの開発、運用などのライフサイクルがあります。そのライフサイクル全体を通じてチームに所有者としての権限を付与したところ、チームは所有者として開発を行い、顧客のために新たな成果を届ける責任感を抱くようになりました。自己裁量権を与えると、チームはやる気を出し、創造力を掻き立てられます。また、信頼のおける環境でリスクを恐れない文化が育まれます。

オーナーシップの文化を受け入れることはテクノロジーと本質的には関係ありませんが、モダンアプリケーション開発の最も困難な側面の 1 つに挙げられます。チームに製品所有者としての権限を与えるには、組織の考え方、チームの構造、チームが担当する作業を変える必要があります。

ほとんどの組織では、IT は 2 つのグループに分類されます。それは戦略的で競争力のある武器としての見方と、より一般的には、ビジネスの成長をサポートするためのコストセンターとしての見方があります。



イノベーション文化を育む

1. **まずは顧客の話を聞く** — どのイノベーションも顧客のニーズを探るところからスタートし、顧客を満足させることを最終目標とします。顧客の需要に重点を置くことを常に優先します。
2. **構築作業はプロを雇って任せる** — 顧客が求める製品と機能を構築、リリースするプロセスを阻む障害をすべて取り除きます。イテレーションペースが速いほど、フライホイールの回転も速くなります。
3. **信念を持って構築者をサポート** — イノベーションに口先だけのサポートは不要です。リーダーから営業、サポート担当者まで、企業の全領域でイノベーションを導入、実践しましょう。

管理業務を減らしてイノベーションを加速

モダンアプリケーションによって、迅速なイノベーションが可能になり、競合他社との差別化を図れます。スピードと俊敏性を売りにしたサービス、プラクティス、戦略を導入することで、通常のビジネスから顧客価値を高めて差別化するアクティビティにリソースをシフトすることができます。多くの実験に取り組み、より迅速にアイデアをリリースに発展させることも可能です。また、構築のプロが管理業務よりも構築作業により多くの時間を費やす環境を育むこともできます。モダンアプリケーションは、Amazon 自身を含む企業が迅速かつ俊敏にイノベーションを実現するための手法です。

なぜ AWS でモダンアプリケーションを構築するのか？

市場への迅速な投入	構築およびリリースサイクルを高速化し、運用上の間接諸経費を軽減することにより、デベロッパーは新しい機能を迅速に構築できます。テストとリリースのプロセスを自動化することでエラー率を低減し、製品をより早く市場に投入できるようになります。	**実績を見る：** **Urbanbase では AWS を使用して 20 倍の速さでサービスを立ち上げ (英語)**
イノベーションを加速	モジュール式のアーキテクチャを使用することで、個々のアプリケーションコンポーネントへの変更を迅速に行うことができ、アプリケーション全体へのリスクが低くなるため、チームはより頻繁に新しいアイデアを試すことができます。	**実績を見る：** **iRobot では AWS Lambda および AWS IoT プラットフォームを使用してロボット掃除機 Roomba を管理**
信頼性の向上	開発ライフサイクルの各段階のテスト手順とモニタリングを自動化することで、モダンアプリケーションはデプロイ時からすでに信頼性の高いものとなっています。またあらゆる問題をリアルタイムで評価し、対応することができます。	**実績を見る：** **Siemens では、お客様管理システムのアラートが 90% 減少し、インフラストラクチャコストが 85% 削減 (英語)**
TCO の向上	従量課金制モデルにより、モダンアプリケーションの過剰なプロビジョニングやアイドル状態にあるリソースへの支払いコストを削減できます。また、インフラストラクチャ管理の負荷を軽減することで、メンテナンスコストを減らすこともできます。	**実績を見る：** **AWS Lambda を使用してアプリケーションのメンテナンスを最大 80% 削減**

アプリケーションのモダナイゼーションプロセスをスタートする

マネージドコンテナサービスへのリプラットフォーム

クラウドで実行されているコンテナ化されたアプリケーションの 80% が AWS で実行されています*。クラウドにおける Kubernetes の全ワークロードの 84% が AWS で実行されています*

リソース (英語)

Amazon ECS workshop、
Amazon EKS workshop、
AWS AppRunner workshop

推奨トレーニング (クラスルーム) (英語)

Running Containers on Amazon Elastic Kubernetes Service (Amazon EKS)

推奨トレーニング (オンライン)

Amazon Elastic Container Service (ECS) Primer

サーバーレス技術とツールを使用した新しいモダンアプリケーションの構築

モダンアプリケーションの構築のためにサーバーレスファースト戦略を導入することで、メンテナンス時間を最大 80%、開発時間を約 70%削減 **

リソース (英語)

Innovator Island — サーバーレスウェブアプリケーション開発のワークショップ。サーバーレスウェブアプリケーションを構築する動画チュートリアル。

推奨トレーニング (クラスルーム)

Advanced Developing on AWS

推奨トレーニング (オンライン)

Architecting Serverless Solutions

モダン Dev+Ops モデルへの変換

高度に進化した DevOps プラクティスを使用したチームの 60% が、セキュリティの脆弱性を 1 日未満で完全に修復 #

リソース

Amazon Builder’s Library、
AWS DevOps サービス

推奨トレーニング (クラスルーム)

DevOps Engineering on AWS

推奨トレーニング (オンライン) (英語)

Getting Started with DevOps

AWS でのモダンアプリケーション構築についての詳細はこちら →

モダンアプリケーション開発のベストプラクティスの導入について問い合わせる →

モダナイゼーションを加速するため AWS パートナーとつながる →

aws modern apps

* 出典 : Nucleus Research
** 出典 : Deloitte
# 出典 : 2020 DORA state of DevOps Report