株式会社スタイルズの
「誰でもわかるAWS」
ホワイトペーパーシリーズ

AWSサーバーレスで高速、低コストにDXを実現する

新しい技術をサービスに取り入れたいが自社の体制・スキルが弱い、
コンテナ・サーバーレスを活用することでコスト削減したいが、
周囲に詳しいベンダーがいない、そんなお客様へ

aws partner network

aws PARTNER Select Tier Services

Stylez

目次

第一章　サーバーレスって何？

1．サーバーレスとは

近年、パブリッククラウドにおけるITシステム設計の一種として、『サーバーレス』が非常に注目されています。サーバーレスとは、クラウドの利用の一形態です。クラウドの利用形態は、大きく分けて以下の4つが挙げられます。

・IaaS

「Infrastructure as a Service」の略で「イアース/アイアース」と読みます。サーバやネットワークといった、ITシステムを構築するインフラ部分を丸ごと借りる利用形態です。AWSが提供する「Amazon Elastic Compute Cloud(Amazon EC2)」が代表例です。

・PaaS

PaaSは、「パース」と読み「Platform as a Service」の略で、ITシステム開発に必要なOSやミドルウェア、データベースが設定された状態のプラットフォームを利用する形態です。GCPのGoogle App Engineや、AWSのRDS、ElasticBeanstalkといったサービスが代表例です。

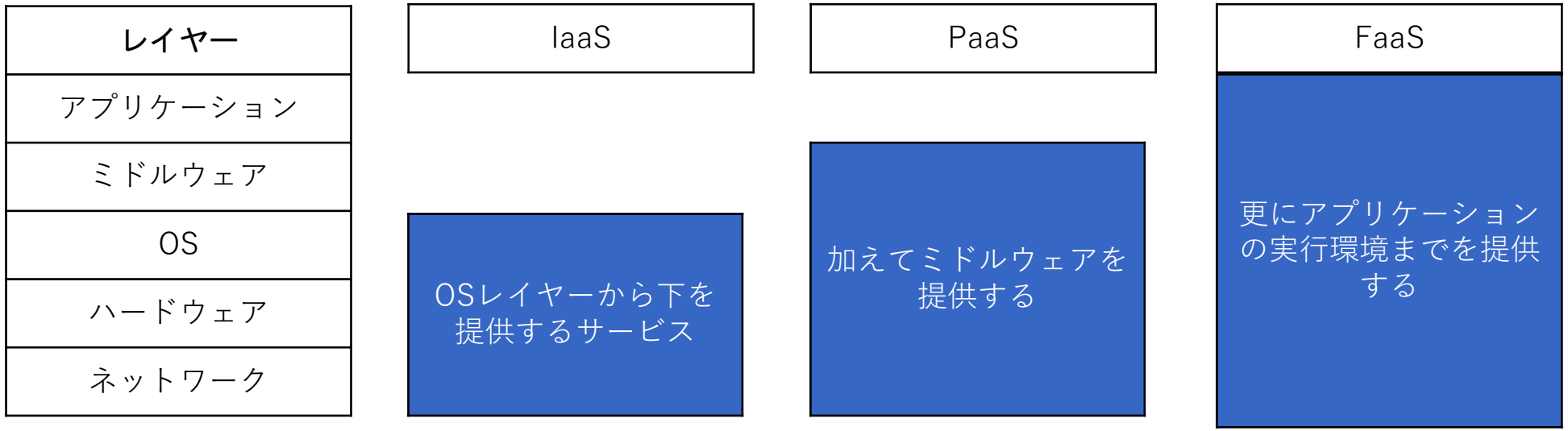
・SaaS

「Software as a Service」の略で「サース/サーズ」と読みます。従来はパッケージとして提供されていたアプリケーションをインターネット上で利用できるサービスで、Microsoft Office 365などが代表例です。

・FaaS

「Function as a Service」の略で「ファース」と読みます。インターネットを通じて、プログラミングで作成した処理の定義・実行のみをおこなうクラウドの利用形態です。AWS LambdaやGCPのCloud functionが該当します。

このうち、クラウドサービス提供側(AWS等)が実際のサーバーを用意し、管理を行うSaaSやFaaSが、『サーバーレス』に該当します。つまり、実際にはアプリケーションが稼働するサーバーが存在するものの、顧客側はクラウドにあるサーバの存在を意識せずに、アプリケーションやデータベースの利用ができるクラウド利用形態のことを『サーバーレス』といいます。。



図：IaaS、PaaS、FaaSの比較

2．サーバーレスの構造や仕組み

先述の通り、サーバーレスは本当に物理的なサーバが存在しない、というわけではなく、クラウドサービス提供者（AWSなど）が物理サーバを用意し、設定、運用管理を行っているから利用者が意識しなくてもいい、ということになります。AWSでいうと、実際は物理サーバ上に小型で起動速度に優れたOSが立ち上がり、Lambdaなどの実行をおこなっている、ということになっています。

※詳細な裏側については、AWSが出しているサーバーレスに関する論文『Firecracker: Lightweight Virtualization for Serverless Applications』に記載があります。

3．サーバーレスのメリット

サーバーレスのメリットを列挙してみると、以下の4つが挙げられます。

・サーバーの運用が不要

一般的にITシステムの運用を行う際には、サーバ（機器自体ハードウェア）が故障していないか、ログにエラーは出力されていないか、ライセンスはきちんと購入されているか、といったことを日々確認する運用が必要です。しかしながら、サーバレスの場合は、これらの作業はすべてAWSやGCPなどのクラウド提供業者が行うため、サーバーの運用に関する様々な業務を削減することができ、人的リソースをアプリケーション開発に注力することができます。

・利用した分だけの課金

サーバーレスは、利用した時間に応じた課金体系です。例えばAWS Lambdaは、実行回数と、Lambda関数のメモリに応じた実行時間に対する課金体系です。このように、使った分だけ課金されるので、コストの効率化を行うことができます。

・可用性（Availability）、回復性（Recoverability）の向上

物理サーバーを利用してITシステムのインフラ設計を行う場合は、データセンターやラックを複数利用するなど、一定の可用性や故障時の回復性を考える必要があります。しかしサーバーレスの各種サービスはクラウドサービス提供者側で非常に高いレベルで可用性や回復性を担保しており、自前で考慮するよりも品質が良い状態で可用性と回復性の担保が可能です。

・柔軟なスケーラビリティ

ITシステムを構築する際には、アクセス数やデータ量など、必要なキャパシティを想定して設計する必要があります。サーバーレスの場合はこれらの作業もクラウドサービス提供者側で自動的に行うため、利用者側での考慮が不要になります。

4．サーバーレスのデメリット

・既存のコードが使えない

Lambdaなどのサーバーレスサービスは、既存のアプリケーションコードをそのまま移行することはできず、ある程度の変換が必要になります。また、Lambda等ではデフォルトで利用できないプログラミング言語も多々存在します。

・ベンダーロックイン

サーバレスアーキテクチャはクラウドベンダーをまたがって開発することが前提となっておらず、一度開発したアプリケーションを他クラウドベンダーへ移行することは推奨されません。

・処理上の成約

AWS Lambda、DynamoDBといったサービスは、何らかの機能制限がある場合があります。たとえば、AWS Lambdaは、15分以上の時間がかかる処理は行えないことになっています。

5．サーバーレスの使いどころ

サーバレスは様々なアプリケーションに利用できます。ただし、処理時間やデータ量に制約があるという点も考慮し、

・小さなデータを高速でやり取りしたい場合

・他システムとの連携を楽に行いたい場合

・小規模にデータ分析を始めたい場合の基盤

のような場合に使うといいでしょう。

観点	メリット	デメリット
ビジネス	・利用した分だけの課金となり、通常はコスト削減につながる。 ・システムの開発速度をスピードアップさせやすくなる。	・ベンダーロックインとなり、他クラウドベンダーへ移行するのは困難。 ・サーバーレス・アーキテクチャで適切に開発／構築できるベンダーはまだ少なく、人材確保が難しい。
開発	・自動でスケールされため、冗長化の考慮／対応などは少なくて済む。 ・疎結合な開発、マイクロサービス等を実現しやすい。	・ベンダーロックインとなり、クラウドベンダーが異なれば、ノウハウを新たに獲得する必要がある。 ・アプリケーションのデバッグの難易度が高い。 ・低レイテンシが求められる場合には注意が必要。
運用	・サーバーの構築やパッチ適用など、構築・運用にかかる手間が不要。 ・障害対応などの負荷が軽減される。	・複数のサービスが関連して動作するため、障害監視などが複雑になり、問題解析も難しくなる。 ・クラウドインフラの障害に大きく影響される。

表：サーバーレスのメリット、デメリット

第二章　AWSのサーバーレス紹介

1．　AWS サーバーレスの代表選手 AWS Lambda とは？

AWSにおけるサーバレスアーキテクチャを考えるうえで一番メジャーなサービスが、AWS Lambda(ラムダ)です。

LambdaはFaaS(Function as a Service：ファース）というサービスの特性をもっています。FaaSは、インターネットを通じてプログラミングで作成した処理を定義・実行するクラウドの利用形態です。

AWS Lambda

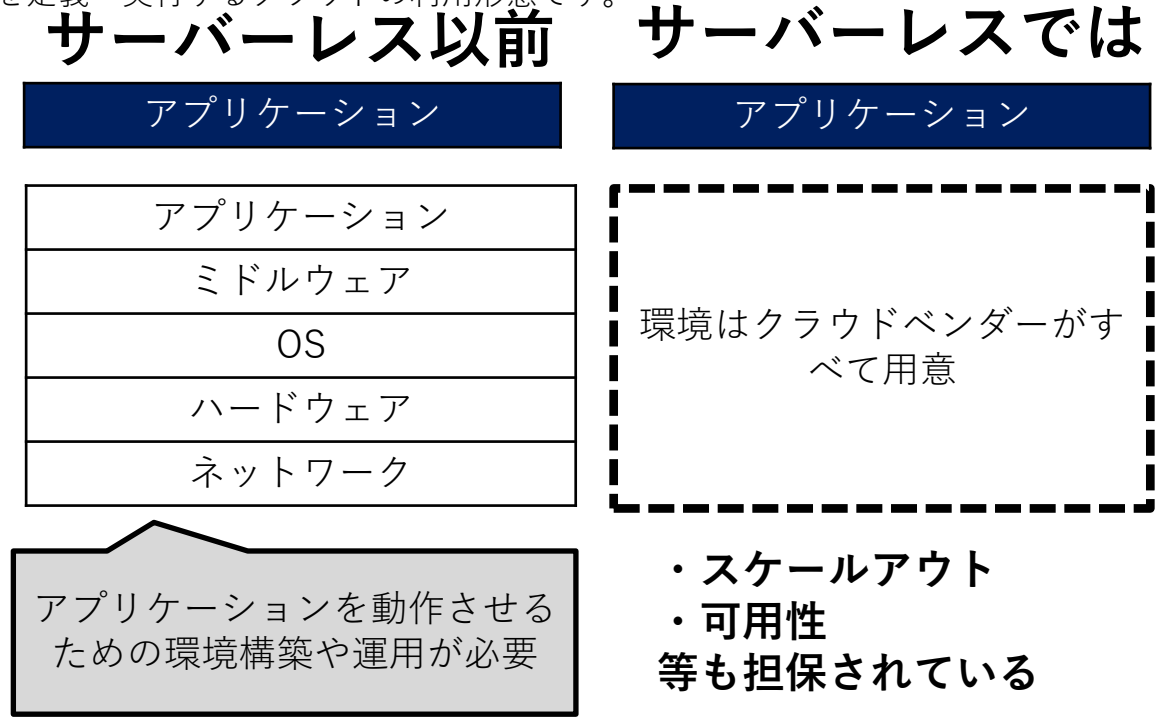
従来のEC2などのIaaSサービスと比べると、以下のような特徴があります。

・Lambdaは、ミドルウェアやパッケージのイ　ンストール・管理が不要

・Lambdaは、サーバ内のデータのバックアップやログのローテーションなど、運用設計や構築作業についても、

・OSやミドルウェアなどの脆弱性の管理についても不要になり、セキュリティ上の考慮点が減る

・可用性やキャパシティプランニングもLambdaは考慮が不要



図：サーバーレスとは

また、Lambdaの最大の特徴は、その安さです。Lambdaは次の項目に関して料金がかかります。

・Lambda関数のリクエスト数（実行回数）
毎月、最初の100万リクエストまで無料、次の100万リクエスト（実行）あたり0.2ドルとなっています。

・Lambda関数のメモリに応じた実行時間
メモリに応じた実行時間では、メモリ1GBあたり、1秒間に0.00001667ドル請求されます。1GBで40万秒実行した分が毎月無料枠として提供されます。

具体的な例を出してみると、128MBのLambdaを1秒間実行した場合は、
128MB/1GB * 0.00001667ドル = 約0.0000000021ドル
となり、利用料金が非常に安いことがわかります。

2．その他のAWSサーバーレスサービス

Lambda以外にも、AWSにはAmazon API Gateway、Amazon SNS、Amazon SQS、AWS Step Functionsなどのサーバーレスサービスがあります。それぞれどのようなサービスなのか、特性やユースケースについて記載していきます。

・Amazon API Gateway
Amazon API Gatewayとは、APIの管理や実行を容易にするサービスです。APIの管理、認証と認可、監視、バージョン管理といった運用に必要な機能もそろっているため、開発が簡略化できます。
したがってユースケースとしては、
・APIを外部に公開したい
・既存のAPIをAWSに移行したい
といった場合に利用すると良いでしょう。

Amazon API Gateway

・Amazon SNS

Amazon SNS（Simple Notification Service）はサーバレスで通知を可能にするサービスです。ユーザやアプリケーションの何らかのアクションをトリガーに、Lambdaの起動やメッセージ通知、SMSなどによるモバイル通知が行えます。ユースケースとしては、

・大量にメッセージ通知を行いたい

・なんらかのアクションをLambdaなどのほかのサービスの実行につなげたい

といった場合に利用すると良いでしょう

Amazon SNS

・Amazon SQS

Amazon SQS（Simple Queue Service）とは、サーバレスのメッセージキューイングサービスです。「メッセージキューイング」は、システム間でデータをやり取りする際に、一時的に溜め込む仕組みのことです。ユースケースとしては、

・アプリケーションに処理時間が長い処理が考えられる場合

・大量のデータ件数を処理する場合

などに利用すると良いでしょう

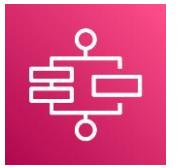
Amazon SQS

・AWS Step Functions

Step functionsとは、ワークフローを設定して各サービスの実行を連携するサービスです。ユースケースとしては、

・順序性があるバッチ処理をサーバレスに行いたい場合

などに利用すると良いでしょう。



AWS Step Functions

3．サーバーレスの制約

サーバーレスサービスは、少なからず何らかの機能制限がある場合があります。たとえば、AWS Lambdaは、15分以上の時間がかかる処理は行えないことになっています。利用の前には、制約について調査しておくといいでしょう。

4．サーバーレスの開発は楽？

・楽になること

サーバレスサービスを利用すると、OSやミドルウェアの設計や開発が不要になるため、インフラ部分を設計する必要がありません。また、サーバレスの特徴として、アプリケーションのコードのみを考えればいいため、運用設計や、実際の運用フェーズにおける負荷も軽減できます。このように、インフラに関する設計や運用工程が大幅に軽減できます。

・楽にならないこと

LambdaやSNS、SQSといった各種サービスの使い方や広範なユースケースを勉強する必要があるため、学習コストとしては高くなってしまいます。

5．サーバーレスで何が変わる？

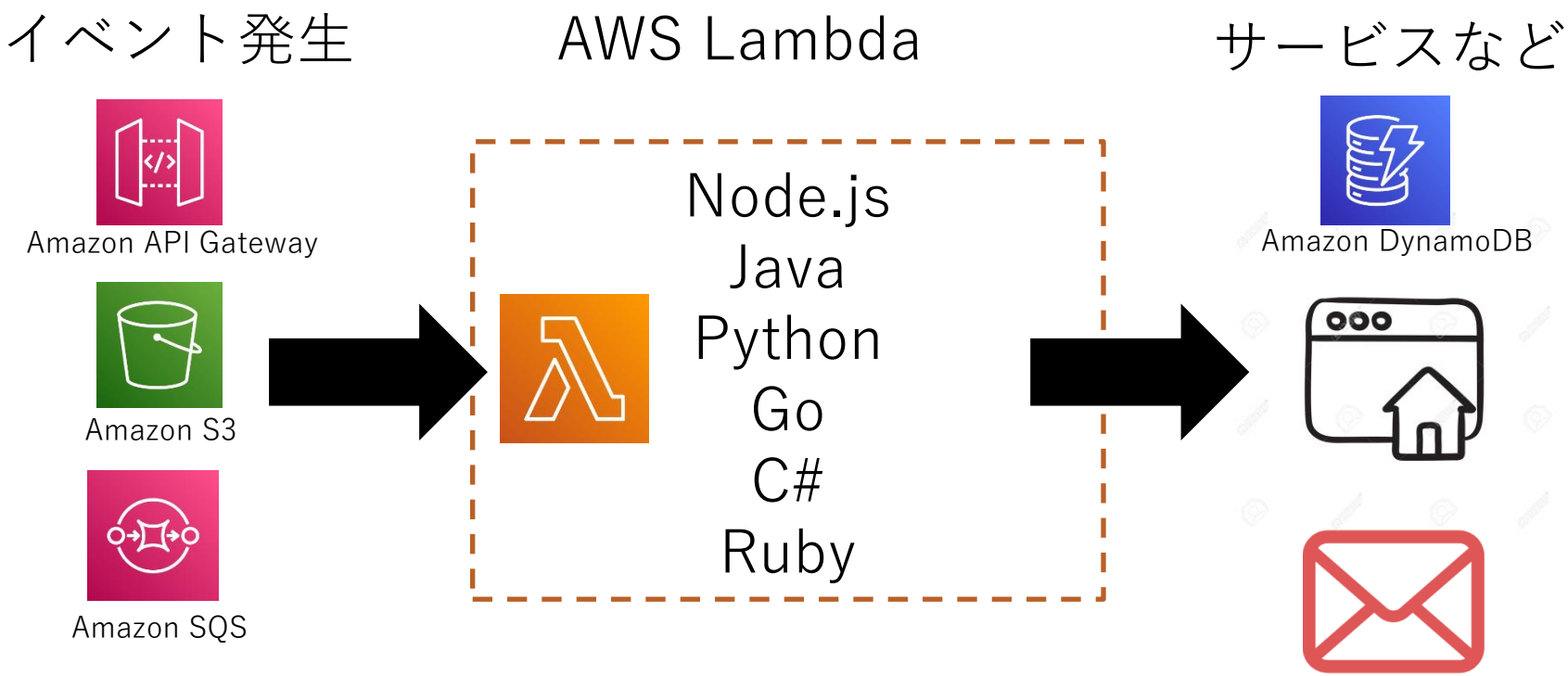
・ビジネス環境の変化に対応しやすくなる。

アプリケーションの開発は早くても数か月、大規模システムの場合は1年もかかってしまう場合もあります。しかし、サーバーレスを用いたアプリケーションは、場合にもよりますが、リリースは最速で数十分で行うこともできます。サーバレスを利用することで高速にアプリケーション開発が可能になるので、ビジネス環境の変化に柔軟に対応できるようになります。

・人的リソースを最適化できる。

サーバーレスではない場合は、データセンターやサーバの契約や管理、サーバの保守といった作業に、多大な人的リソースとコストがかかります。しかしサーバレスを利用することで、契約や管理に向けたコミュニケーションコストや、インフラエンジニアの人員低減が可能になります。その分、ビジネスの改善やアプリケーションの改良に人や時間、お金を集中できるようになります。

サーバレスのサービスを利用する場合、特性を十分に理解する必要があります。しかしながら、Lambdaを代表として、サーバレスは高速かつ低価格で、アプリケーションをリリースするためのQCDを大幅に改善することができます。様々なサービスを組み合わせることで、幅広い範囲のアプリケーションを構築できるため、是非利用してみてください。



図：サーバーレスなアプリケーションの構造

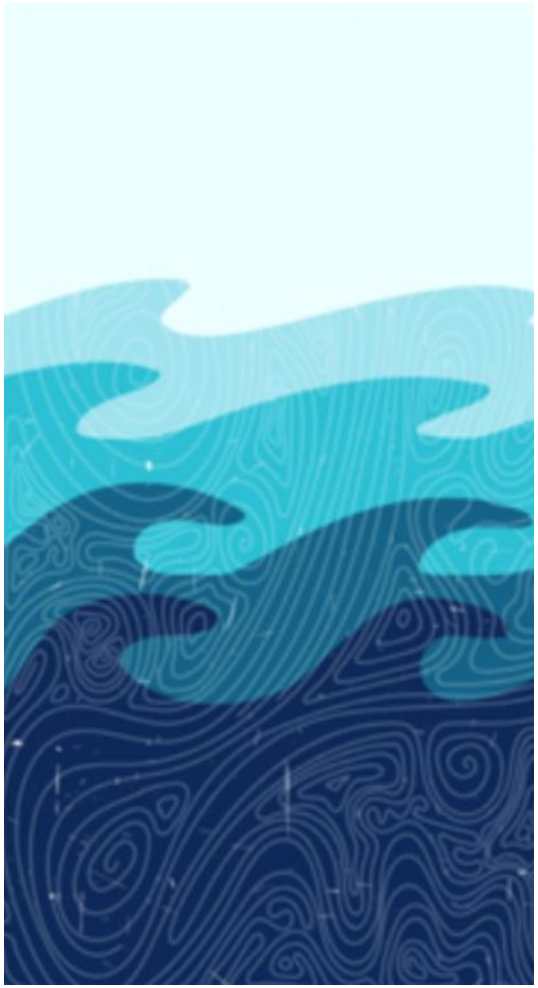
第三章　AWS Lamdaってなに？なにができるの？

1．サーバーレスの意義とAWS Lambda

サーバーレスの意義は、開発の高速化と、運用コストの低減にあります。まず、サーバーレスのサービスは数クリックで利用開始することが可能で、OSやミドルウェアのセットアップ等を行うことなく実施することができます。したがって、従来のアプリケーションよりも開発スピードが大幅に向上します。

また、サーバーレスサービスでは、稼働するサーバーの障害対応や交換といった運用保守作業はすべてAWSやGCPなどのクラウド提供業者が行います。そのため、サーバーの運用に関する様々な業務を削減することができ、人的リソースをアプリケーション開発に集中させることができます。

AWS Lambdaはサーバーレスの代表的なサービスとして、様々な場面で利用されています。



2．AWS Lambdaの詳細

・Lambdaの制約は？

まず、Lambdaは起動時間と、同時実行数に制限があります。起動時間については、1つの実行につき15分までとなっています。そのため、処理時間が15分以上にならないよう処理を分割して並列化する処理設計が必要です。

同時実行数については、Lambdaでは上限値は同一アカウントの同一リージョン（アジアパシフィック（東京)を前提）内で1,000までとなっていますが、上限に達するとそれ以上の関数の呼び出しは制限（これを、スロットリングと呼びます）されます。

・スロットリングを回避するには？
1つが「リトライ処理を組み込む」という方法です。同期処理の場合、スロットリングが発生するとLambda関数は特定のエラーを返却します。このエラーが発生した場合、関数の呼び出し元でリトライ処理を実行します。また、AWS Lambdaの場合、上限緩和の申請をすれば、同時実行数の増加が認められることがあります（ただし、必ず認められるとは限りません）。

・他のサービスとの連携についての考慮点は？
Lambda以外のサーバーレスサービスの制限にも考慮が必要です。例えば、サーバーレスアプリケーションの構築としてよくあるのが、API Gateway + Lambdaという組み合わせです。この場合はAPI Gatewayの制限にも考慮する必要があります。API Gatewayの場合、最大29秒でタイムアウトとなる仕様となっています。このため、29秒以内にLambda関数から値を返却し、以降は非同期で処理を実行するなどの考慮も必要になります。

・使えるプログラミング言語は？
現在、利用できるプログラミング言語は、Java、Ruby、Python、Node.js、.NET、Goとなっています。

・ほかのライブラリを使いたい場合は？
Pythonでは多様なライブラリが用意されています。そのようなライブラリを使いたい場合は、レイヤという機能を使います。レイヤは、複数のLambda関数で外部ライブラリやビジネスロジックを共有できる機能です。使いたいライブラリのファイルなどをZip化し、レイヤにアップロードすることで、関数内で任意のライブラリが利用できるようになります。

・トラブルシューティングはどうする？
Lambda関数のエラーをトラブルシューティングする場合は、Cloudwatchにログを出力させるようにします。それによって、実行時のログが出力されるようになります。また、エラーは発生していないが、スローダウンが発生している、といった性能の検証も必要な場合があります。その場合は、X-rayというAWSのサービスと連携するといいでしょう。

3．AWS Lambda の良いところ

AWS Lambdaのいいところは、先述のサーバレスのメリットと同じように、高速開発が可能であることと、仮想サーバーの運用保守が不要で、大幅な人的コストの改善が見込めるということになります。また、Lambda特有のメリットとしては、AWのサービスとサービスを疎結合につなげることが簡単にできます。他のサービスとの連携については下記のようなケースが考えられます。

・S3のイベント発生（ファイルアップロードなど）をトリガーにデータ処理をおこなう
・DynamoDBのデータ更新に合わせて何らかの処理をおこなう
・API Gatewayに対してユーザーがアクセスしたときに、ユーザー認証を行わせる
・Event Bridgeと組み合わせて、毎日〇〇時に決められた処理を実行する
・StepFunctionと組み合わせて処理フローを定義して起き、バッチ処理のような使い方をする

このように、他のAWSサービスと連携して、様々なアプリケーションを高速に開発できる点が、大きな強みです。

4．非同期呼び出し時の重複起動に注意

非同期でLambda関数を実行する場合は重複起動の可能性があります。非同期処理でLambdaがイベントを処理できずにエラーとなった場合、自動的にリトライされるためです。

重複起動しないためには設計時の考慮が必要です。Lambdaが起動する時にDynamoDBにトリガーとなったイベントを保存します。そして、重複起動が発生した場合はDynamoDBに保存されているイベントを確認して後続処理を動かさないなどです。このように設計時に重複処理を発生させないように対応することで防ぐことができます。

種類	説明	トリガー
同期	同期呼び出しの場合は直接Lambdaを 1 回 実行します。Lambdaが実行され、処理が完了してからレスポンスが返ってきます。	APIGateway（デフォルト） Cognito、Alexa、Lex
非同期	非同期呼び出しの場合はLambdaを直接実行するわけではなく、キューイングされた後実行されます。Lambdaの呼び出しに失敗した場合は自動的に2回までリトライされます。キューイングされたタイミングでレスポンスが返ってきます。	APIGateway（ヘッダーでEventを指定した場合） AWSIoT CloudWatch Events、CloudWatch Logs CodeCommit S3、SNS、SES、KinesisFirehose　　等

表：AWS Lambdaの同期・非同期

5．AWS Lambdaは使った分だけの従量課金

従量課金はAWS全体の大きなメリットですが、それでもサーバインスタンス（EC2）を起動していれば（処理が行われていない時間も）課金対象になります。一方、Lambdaならリクエストがあったらその時間だけが課金対象となります。

6．課金単位が変更でより安く利用可能に

Lambdaの課金体系は2020年12月に刷新され、以前より安くなりました。Lambdaの実行時間に関する課金体系が更新され、実行時間が100ミリ秒単位ではなく1ミリ秒単位で課金されるようになりました。従来は、Lambdaの実行時間が50ミリ秒だった場合、100ミリ秒に切り上げて料金請求がなされていましたが、新しい料金体系では、50ミリ秒分の課金しか発生しなくなり、約半額で利用できるようになりました。

第四章　サーバーレスの主要なユースケース

1．サーバーレスのメリット、デメリット

まずは、メリットについて記載してきます。

・サーバーの運用が不要

一般的にITシステムの運用を行う際には、アプリケーションが稼働している物理サーバ、OS、ミドルウェアの運用保守も必要です。具体的には、サーバ（機器自体）が故障していないか確認する、OSやアプリケーションのログにエラーは出力されていないか監視をする、ライセンスを定期的に更新する、といったことを日々行っていく必要があります。しかしながら、サーバーレスの場合はこれらの作業はすべてAWSが行います。可用性や拡張性の設計に関しても、AWSなどのクラウド提供者側で実施するため、運用を行っている中でリソース拡張作業を行う、といったことも不要になります。

・利用した分だけの課金

サーバーレスは、オンプレミスやIaaSなどの他のクラウド利用形態と違い、利用した時間や実行回数に応じて課金がされます。例えば、オンプレミスはサーバーを設置していなくても場所を借りるだけでも利用料金がかかりますし、IaaSの場合は、サーバーの利用契約を3か月単位で結ぶ場合では、3か月の間に使っても使わなくても課金が発生してしまいます。一方でサーバレスは、プログラムを実行した時間に応じた課金体系です。例えばAWS Lambdaは、Lambda関数のリクエスト数（実行回数）と、Lambda関数のメモリに応じた実行時間に対する課金体系です。このように、使った分だけ課金されるので、コスト効率が非常に優れています。

・高速開発が可能

一般的なプリケーションの設定は、アプリケーション自体の設計のほかに、OSやミドルウェアの設計も行います。しかしながら、先述のように、サーバレスの場合はクラウド提供者側でOSやミドルウェアの設計を行うため、利用者側での対応が不要です。したがって、設計工程を大幅に短縮することができ、アプリケーションやITシステムの高速開発が可能です。

次に、サーバーレスのデメリットについて記載していきます。

・学習コストが存在する

サーバーレスのサービスには様々なサービス、設定項目が存在するため、それらに関する学習コストがかかります。たとえば、AWSでサーバレスのサービスを構築する場合、S3、API Gateway、AWS Lambdaといったサービスについて理解をしておかないと、利用料金が高くなってしまったり、思うように性能が出なかったり、十分にサーバレスのメリットを得ることはできません。

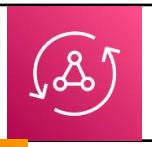
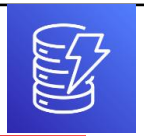
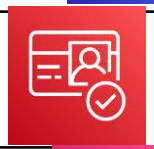
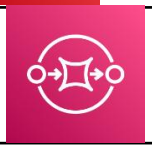
・機能の制約がある

AWS Lambda、DynamoDBといったサービスは、機能の制約があります。たとえば、AWS Lambdaは、15分以上の時間がかかる処理は行えないことになっており、それ以上の時間がかかるような処理には向いていません。このように、各サービスの制約についてはは事前に確認を行うと良いでしょう。

サーバーレスのメリット	サーバーレスのデメリット
・サーバーの運用が不要 ・利用した分だけの課金 ・高速開発が可能	・学習コストが存在する ・機能の制約がある

2．サーバーレスのユースケース

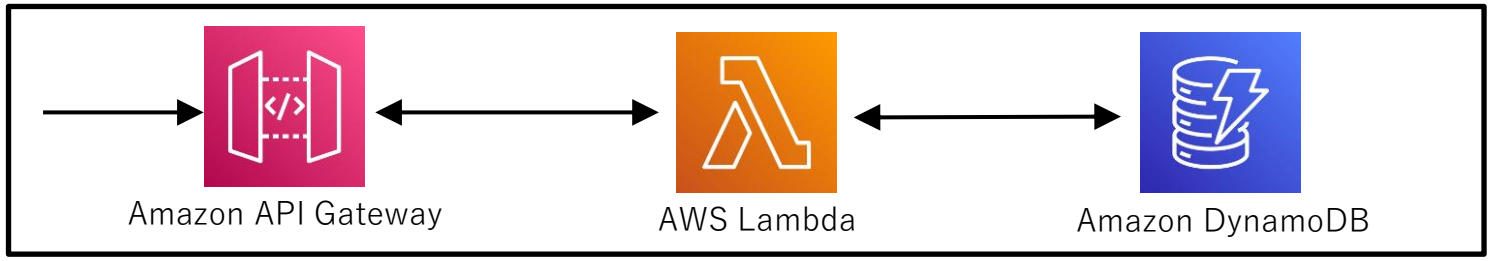
サーバーレスサービスの組み合わせにより、どのようなサービスが構築できるのかを記載していきます。基本的には、AWSのサーバーレスサービスの基本的な部品は次スライドのように利用します。次スライドに記載された、基本的なサービスを組み合わせて、様々なサービスを構築していきます。

ユースケース	利用するサービス
外部との接続 インターフェース	API Gateway  AppSync 
処理ロジック	AWS Lambda  AWS Fargate 
データストリーミング	Kinesis Data Stream  Kinesis Firehose 
データストア	S3 
データベース	DynamoDB  Aurora Serverless  RDS 
認証	Cognito 
アプリ結合 メッセージング	AWS SQS  AWS SNS 

表：サーバーレスのユースケースとサービス一覧

3．ユースケース例：動的Web、モバイルバックエンド

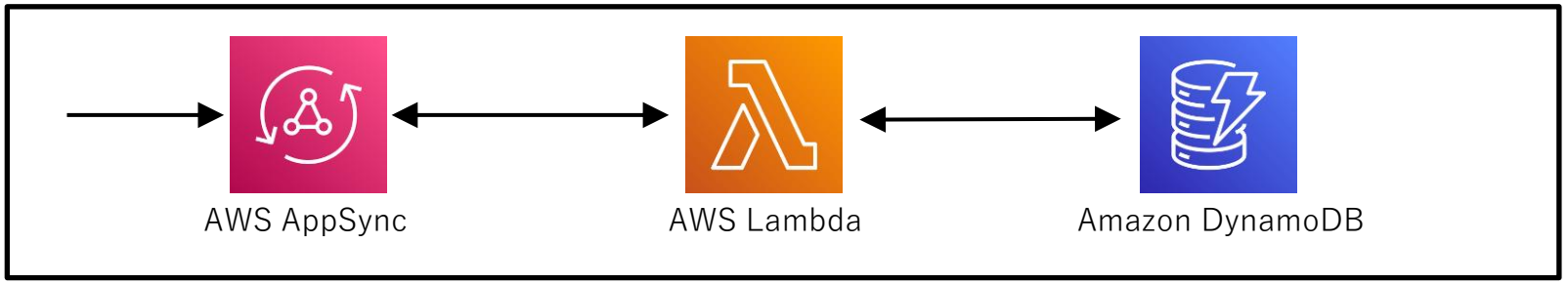
・API公開の典型的なサーバーレス実装です。
・リクエスト／レスポンス型向け（同期モデル）です。
・REST APIを経由してDBの情報を同期的に参照／更新します。
・SPAやモバイルアプリで多様されるパターンです。
・レガシーなWebサーバー/APIサーバーの置き換えにも適用が可能です。
・API Gatewayの統合タイムアウトが30秒であることを考慮し、非同期呼び出しの適用も考慮が必要です。



ユースケース図:動的Web、モバイルバックエンド

4．ユースケース例：リアルタイムモバイル・モバイルオフライン処理

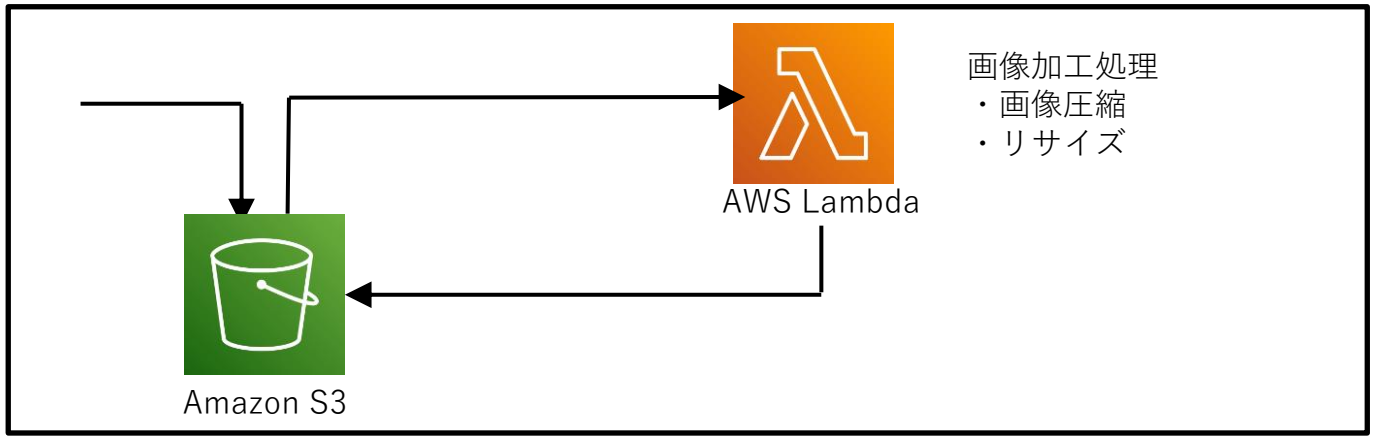
・リアルタイム通信要件や非接続状態（オフライン）でも動作する要件があるモバイル向きです。
・AWS Appsyncはリアルタイム機能とオフライン機能を備えたフルマネージドなGraphQLサービスです。
・フロントエンドには、AWS Amplifyの適用も可能です。
・GraphQLのモデリングによってAPIを設計します。
・データソースとしては、AWS Lambda、Amazon DynamoDB、Amazon Aurora Severless等を利用可能です。



ユースケース図:リアルタイムモバイル・モバイルオフライン処理

5．ユースケース例：シンプルなデータ加工（画像の例）

・データの投入をトリガーとしてファイル情報を引き渡して、処理を起動します。
・Amazon S3からAWS Lambdaを非同期に呼び出します。
・Amazon S3のイベント発行はAt Leaset Onceなので、起動漏れはありませんが、重複して呼び出された場合の考慮をしておく必要があります。

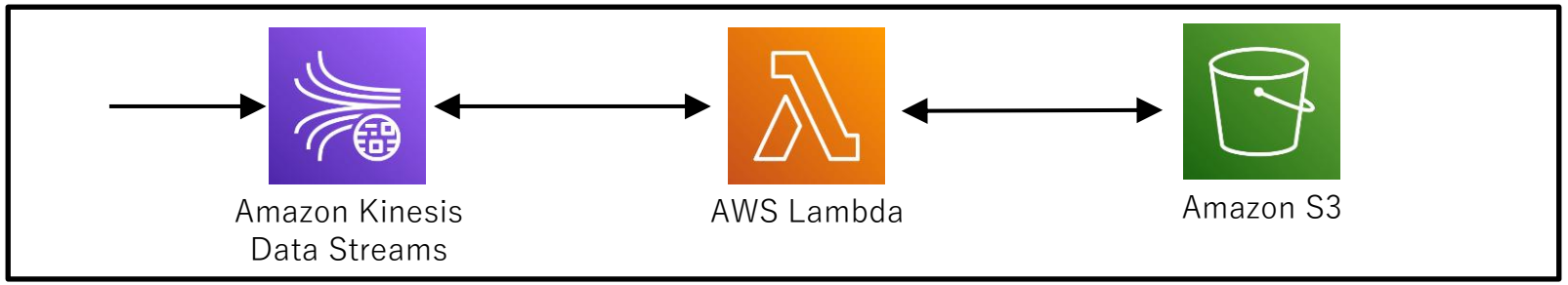


ユースケース図:
リアルタイムモバイル
・モバイル オフライン
処理

6．ユースケース例：流入データの連続処理

・Kinesisに連続して流入するデータを定期的に受信して、データ加工を行って格納します。

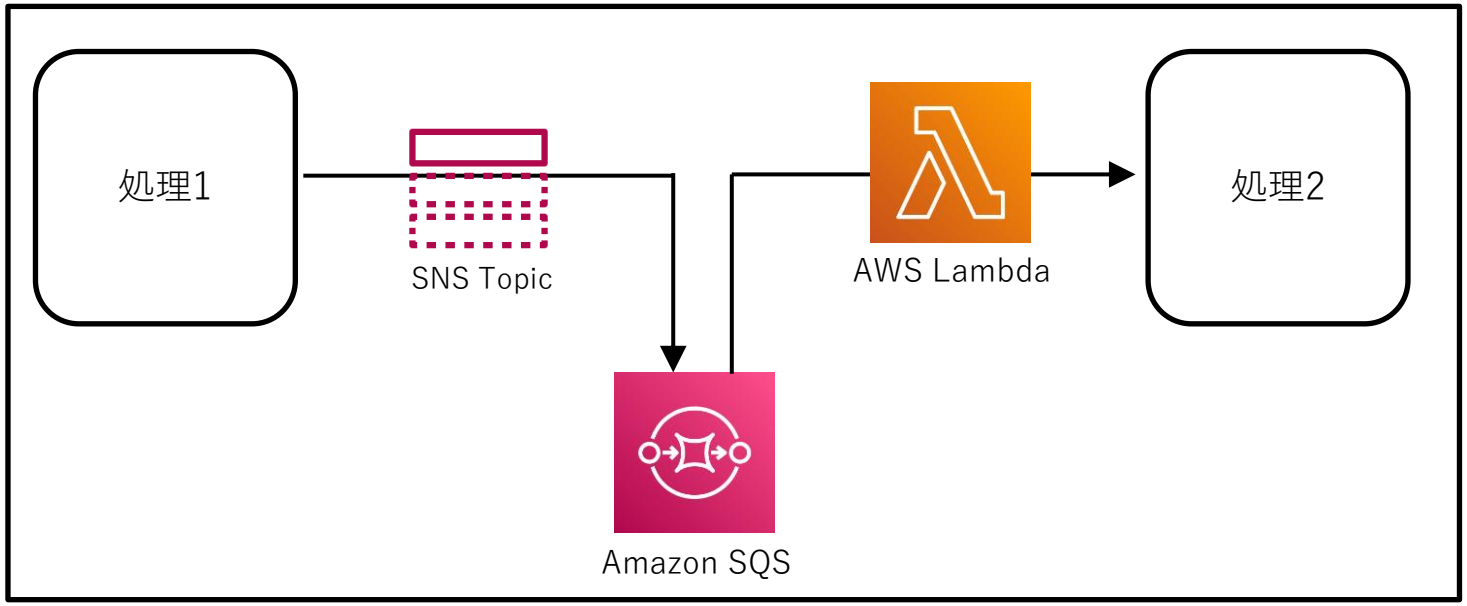
・ストリーミングデータを変換してAmazon S3やAmazon Redshift、Amazon Elasticsearchなどに保存するユースケースであれ　ば、Amazon Kinesis　Data Firehoseの利用も検討するといいでしょう。



ユースケース図:流入データの連続処理

7．ユースケース例：イベント駆動の業務処理連携

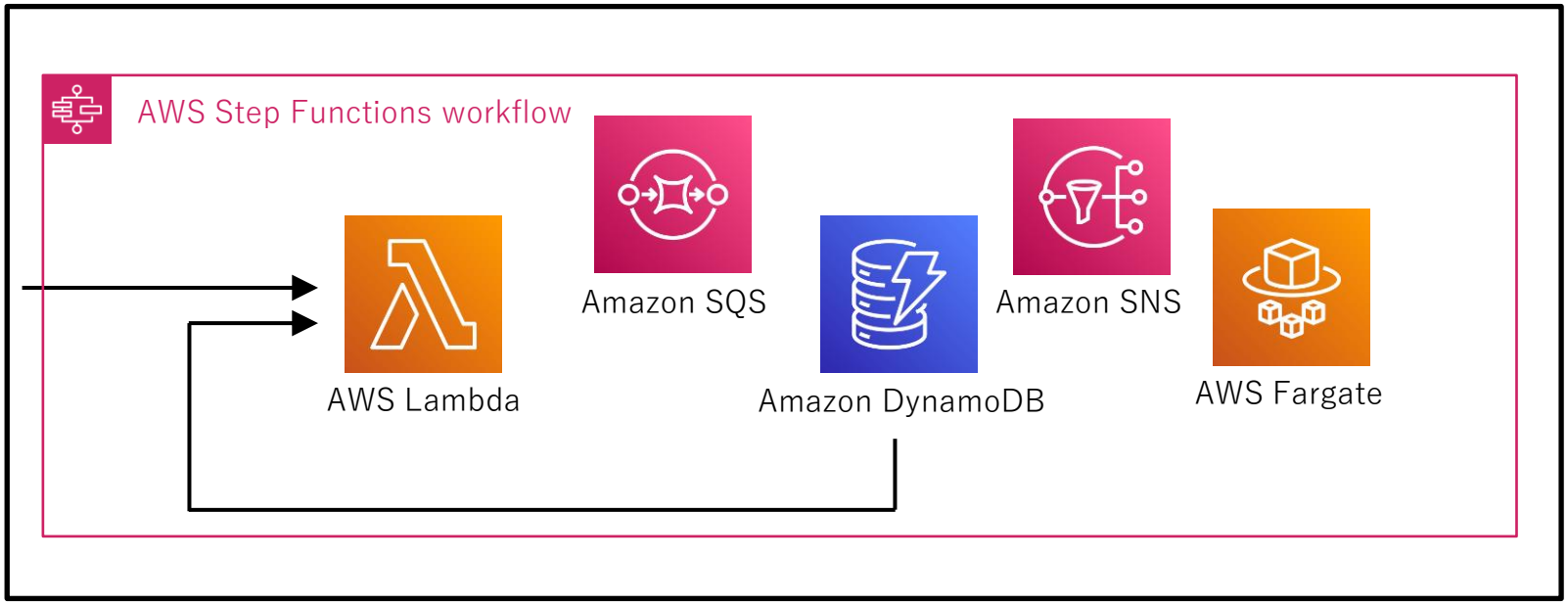
・次の処理のためのタスクをキュー（またはS3）にPushして、次の処理に非同期で連携します。
・一つのトピックに対して、複数の処理を個別に実行することも可能です。
・順序性が大事なビジネスロジックの場合は、Amazon SQSのFIFOキューと組み合わせての実行も可能です。



ユースケース図:
イベント駆動の業務
処理連携

8．アプリケーションフロー処理

・一連の処理フローを可視化して、エラー処理のフロー管理としても利用可能です。
・AWS Step Functionsによってリトライや例外処理を宣言的に設定することが可能です。
・AWS Step FunctionsからAWS Step Functionsへの呼び出しや、他の処理の完了を待機してから再開することも可能です。



ユースケース図:アプリケーションフロー処理

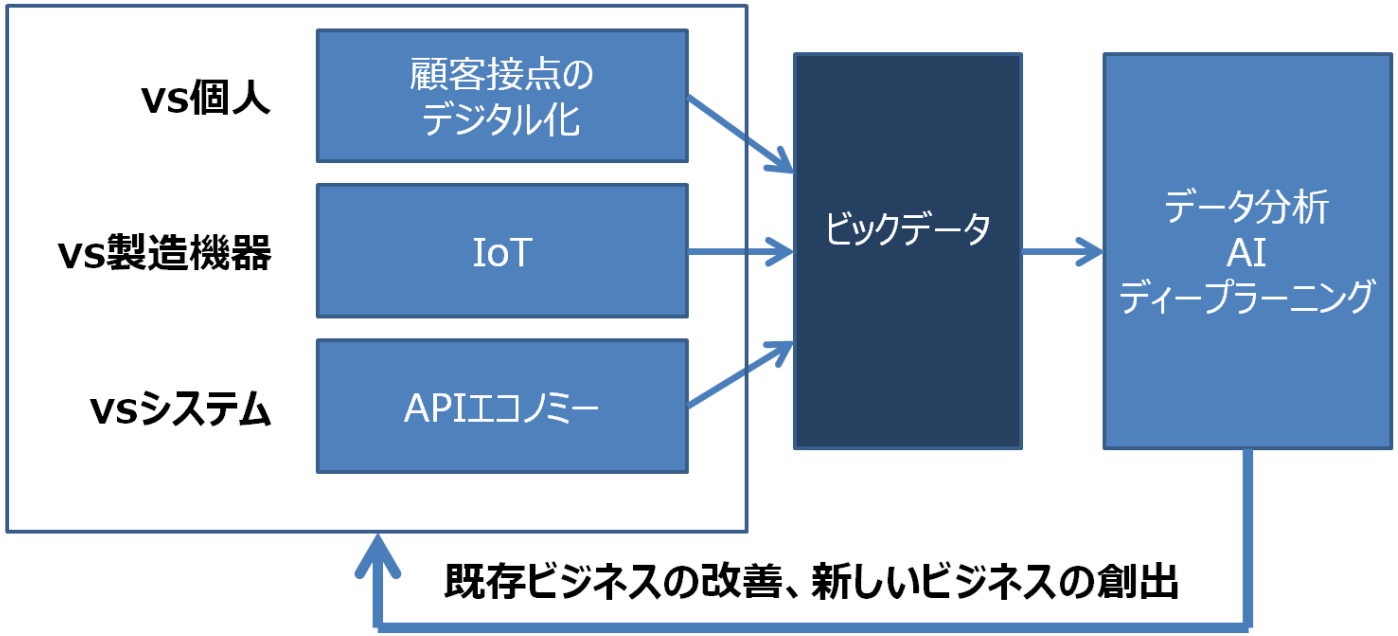
9．その他のユースケース

紹介したような典型的な処理以外にもサーバーレスは様々な機能を有しています。例えば、Lambda、API Gateway、S3、DynamoDBを利用すれば、簡単なWebアプリケーションの構築が可能になります。Amazon connectを利用すれば、サーバーレスでコールセンターの設立が可能です。このように、AWSの様々なサービスを組み合わせることで、高速かつ幅広くITシステムの構築が可能になることが、サーバーレスの大きな特徴です。

第五章　DXのためのサーバーレス開発

1．ソフトウェア開発能力こそが競争力の源泉

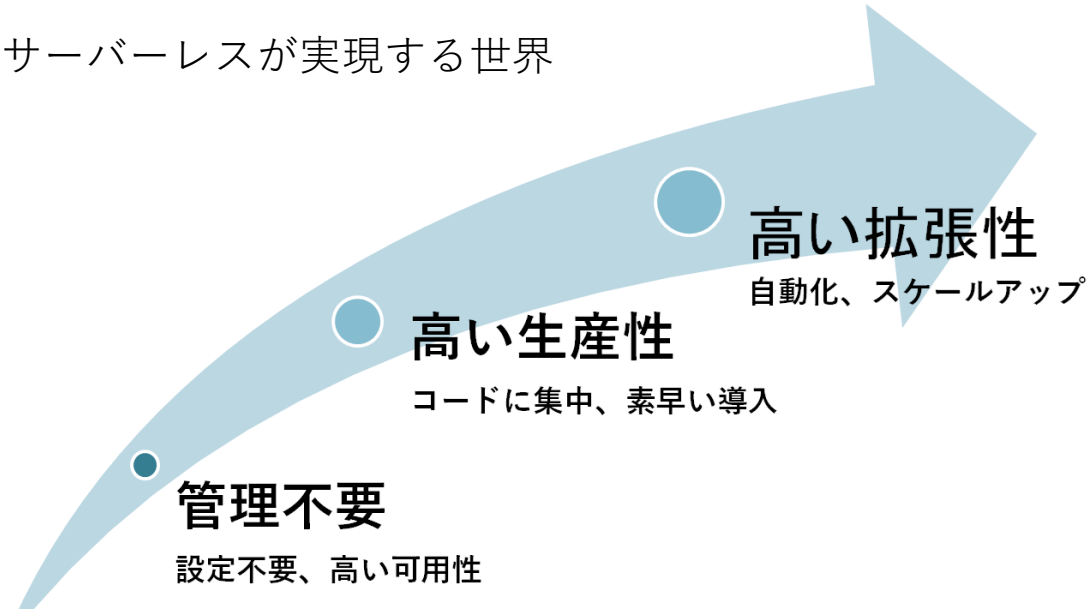
デジタルトランスフォーメーション（DX）の必要性が叫ばれ始めてから、4～5年が経過しています。企業にとっての顧客接点がスマートフォンに移行したのを契機に、デジタルマーケティングを課題にとらえるB2C（企業対個人）型の企業が急増し、一方、B2B（企業間）型の企業は、IoT（モノのインターネット）やAI（人工知能）、ビッグデータ、API（アプリケーションプログラミングインタフェース）といったデジタル技術をテコに、新しいビジネスの創出を喫緊の課題ととらえ始めています。DXとは、こうしたデジタルマーケティングや、IoT、AI、APIといったデジタル技術を使ってデータをリアルタイムに収集し、そのデータを素早く解析することで、ビジネスの改善や新しいビジネスの創出にフィードバックしていく流れ全体を指すと考えればよいでしょう。そのすべての過程で必要になるのが、ソフトウェアの開発であり活用です。高度なソフトウェアを素早く開発し、リリースする能力こそが、デジタル化時代に成長できる企業が求める競争力の源泉なのです。



2．サーバーレスが実現する世界

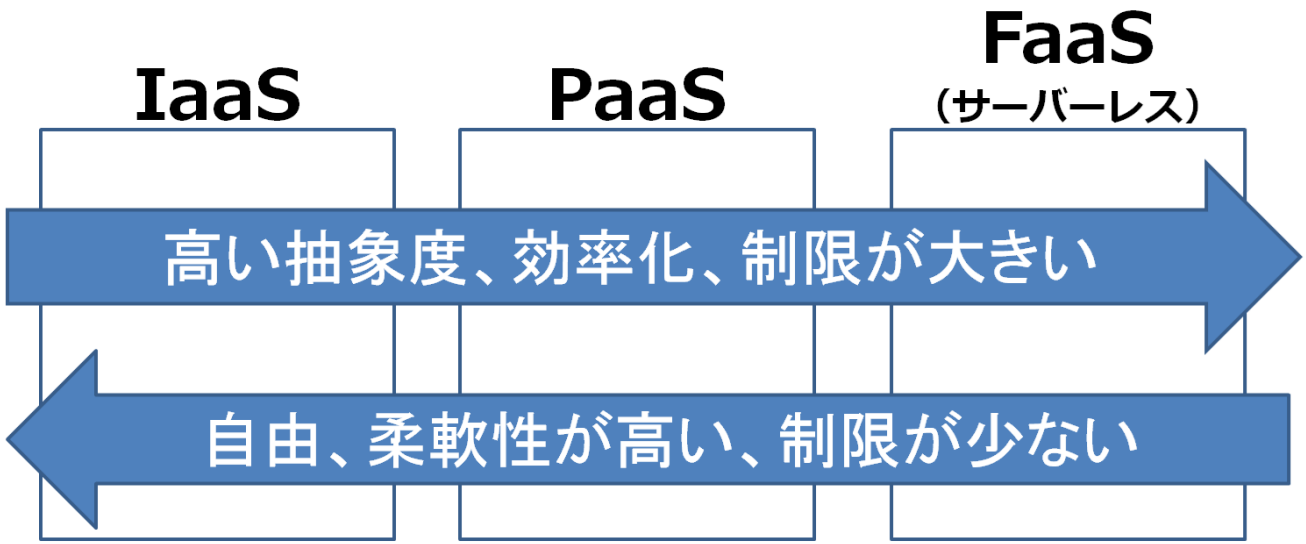
デジタルトランスフォーメーション（DX）のソフトウェアの構築方法として、今、サーバーレスが注目されています。サーバーレスとは、アマゾン ウェブ サービス（AWS）を代表とするクラウドベンダーが提供するサービスで、サーバーの構築や保守などの面倒な管理をすることなく、プログラムを実行できる仕組みです。当たり前の話ですが、これまではちょっとしたプログラムでも、サーバーを構築しなければなりませんでした。しかし、このサーバーレスの仕組みを使うと、プログラムさえ作成してしまえばすぐに実行することができます。
そのため、サーバーの性能設計やメンテナンスは全く必要がありません。さらに、可用性と耐障害性が組み込まれており、開発者はプログラムコードにだけ集中することで、高い生産性を実現できます。

図：サーバーレスが実現する世界



3．IaaSからFaaSまで

サーバーレスは、イベントドリブン型（イベントの発生をきっかけとして処理が開始されるプログラム方式）の動作やステートレス（状態を持たないこと、入力の内容によってのみ出力が決定される方式）が前提となり、設計上の制限が大きいため、それだけでシステムのすべてを構築できるわけではありませんが、APIやマイクロサービスといった実装のためには非常に生産性のよい方式です。そして、それ以外のアプリ（ユーザインターフェース、ステートフルである機能）は従来のIaaS上で動作させるのが自然な構成です。そのため、サーバーレス（FaaS）と従来型のサーバー（IaaS）は対立する概念ではなく、目的によって使い分けるものです。
加えて、サーバーレスのデメリットとして、ベンダーロックインの問題を大きく取り上げるケースが多いのですが、クラウド上で効率的にシステムを開発しようとすればFaaS(サーバーレス)の活用は避けて通れません。それほど、FaaS(サーバーレス)は実装のメリットの大きなサービスです。



図：IaaSからFaaS（サーバーレス）までの抽象度と自由度の相反

4．サーバーレス＝クラウドネイティブな開発のメリット

これまでは、 Webサービスや、高度なモバイルアプリケーションを提供するためには、以下の３つのプロセスが必要でした。

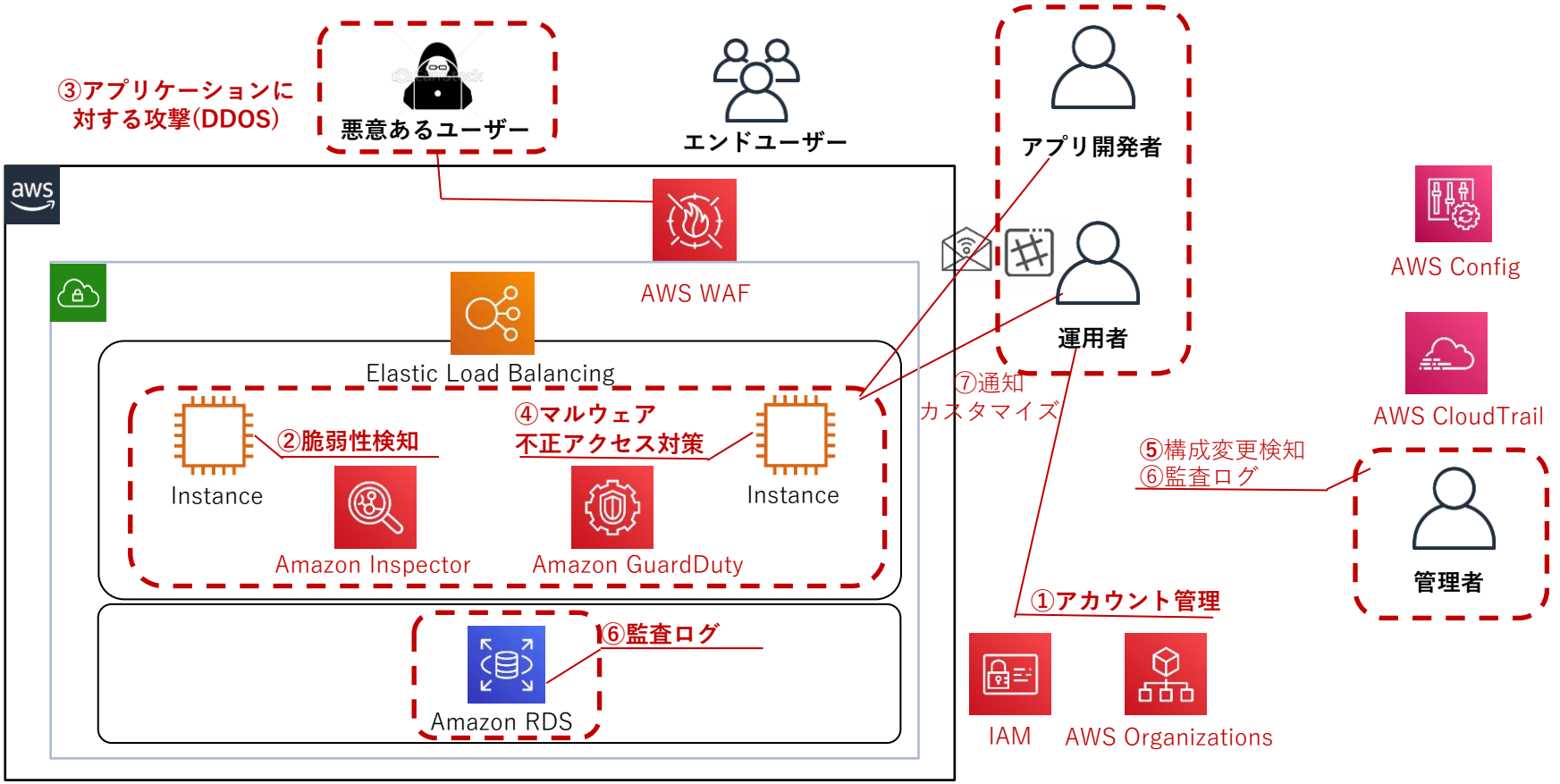
1）アプリケーション（プログラム）を開発する
2）サーバー負荷を予測して、サーバ環境を構築する
3）監視や負荷分散など設計して運用する

しかし、クラウドベンダーのマネージドサービスを利用してシステムを開発すると、「プログラムを開発する」以外の部分をクラウドベンダーにお任せできるようになるのです。 また、大規模のサービスであればあるほど、これまでは、「対障害性」「スケールすること」を考慮して、アーキテクチャーを設計する必要がありました。　この「対障害性」「スケールすること」こそが、マネージドサービスの特徴であり、開発スピードを削減できる部分なのです。

我々スタイルズは、エンタープライズのシステムを開発してきたメンバーによって提供しており、そこで培った品質要求レベルで、クラウドネイティブなシステム開発・運用を行っています。

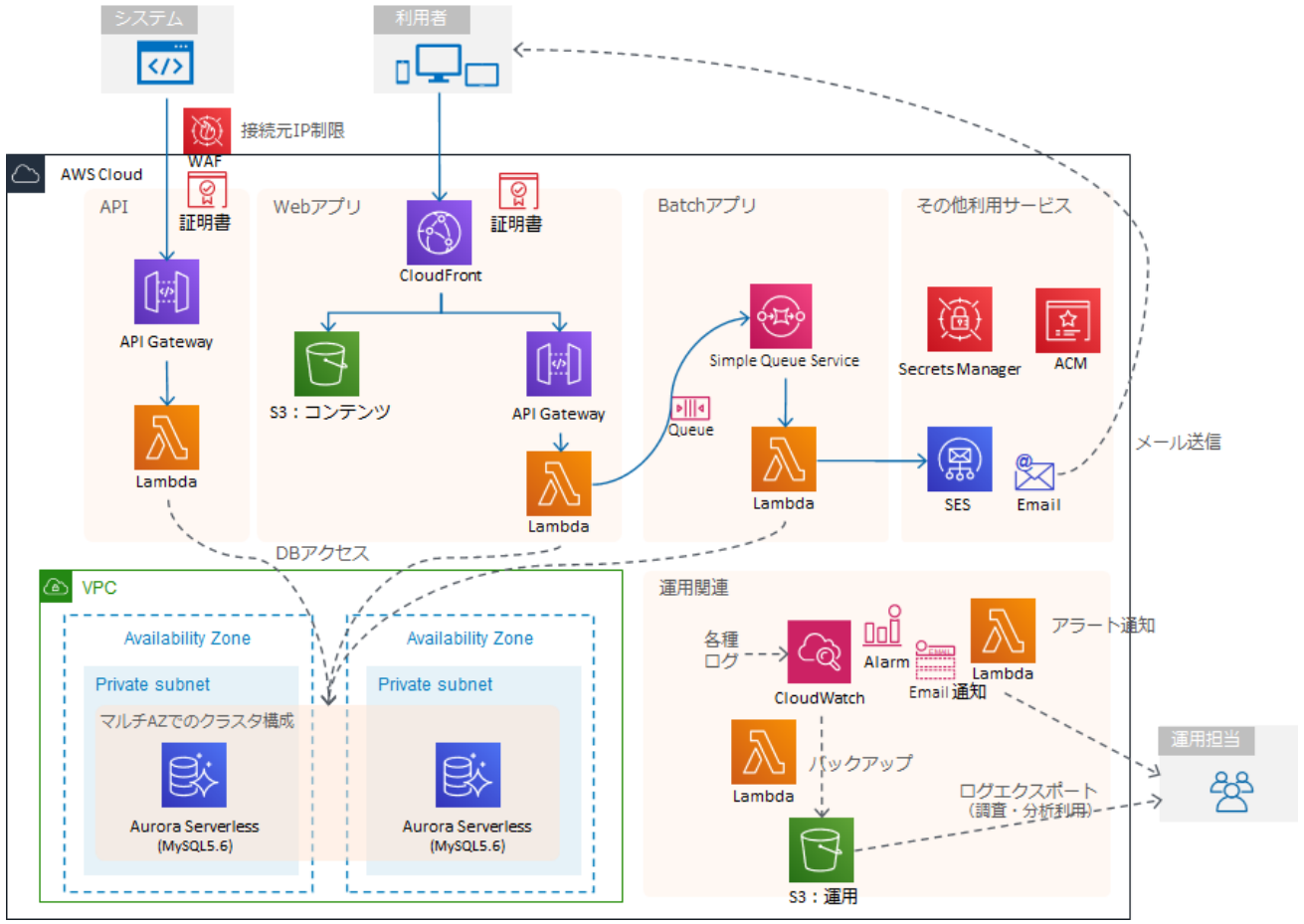
図：サーバーレス＝クラウドネイティブな開発のメリット

マネージドサービスを活用したセキュリティ監視



導入先	某自動車メーカー等多数実績あり
説明	コネクテッドカーシステムの構築に伴い、AWSマネージドサービスを利用し、様々な脅威に対する施策、監査ログの出力・ログ集約といった管理運用面に対する施策を実施。また、これらを運用体制に合わせた通知方式・通知内容の形にカスタマイズを実現。 導入後のセキュリティレポート報告やカスタマイズ対応範囲の保守を実施中。

スポーツメディア向け 認証基盤開発



導入先	某スポーツ関連団体
説明	メディア向けに提供している複数システムにおいて、利用者がすべて異なるアカウントを利用しているため状況にあったため、認証およびユーザ管理をサーバレス構成にて実現。メディア向けのポータルサイト、関連システム向けのAPIを開発。

スタイルズのAWS関連サービス

AWS導入・移行サービス	・自社の基幹システムやWebサービスをクラウドへ移行したいがどのように進めていけば良いのか悩まれているお客様に向けたベストプラクティスをご提供します。
AWSセキュリティサービス	・データ漏洩、不正アクセス、マルウェア感染、DDoS攻撃、改ざんなどのセキュリティインシデント対策から、可視化、自動復旧対処の仕組み開発まで、AWSのセキュリティ対策をすべてご影響可能です。
AWS上のサーバーレス開発	・ECSを利用したコンテナ構成やサーバーレス構成により、オートスケール（自動拡張）やオートヒーリング（自動復旧）など、クラウド利用のメリットを最大限に活用したシステムをご提案します。
AWS監視、運用・保守サービス	・AWS、Zabbixのパートナー企業として多くのお客様に監視サービスを提供させていただいた実績と長年培ったノウハウをベースにお客様環境に合わせた最適な監視運用をご提案をします。
AWSコンテナ構築運用サービス	・Amazon ECS（AWSが提供するコンテナサービス）をベースにお客様のシステムをコンテナ化して運用するまでのご支援をさせていただくソリューションです。
AWS DevOps導入サービス	・インフラのコード化、さらにソースコードのビルドやテスト、および、アプリケーションのデリバリーまで、開発工程全体を自動化するCI/CDの仕組みの構築をAWSの環境上での実現をサポートします。

お問合せ、ご相談は株式会社スタイルズ https://www.stylez.co.jp まで

実績豊富なエンジニア集団の技術と開発ツールで
「開発期間/コスト削減」「品質向上」を実現

株式会社スタイルズ

https://www.stylez.co.jp

東京都千代田区神田小川町1-2　風雲堂ビル6階

AWSパートナーの一員として、CloudShift（クラウドシフト）というブランドにて、AWS導入の支援を提供しております。

CloudShift