

コンピューティング

EC2  
ECS  
Lambda

ストレージ

EBS  
S3

ネットワーク

VPC  
Route 53  
ELB

データベース

RDS  
DynamoDB

など

実務に役立つ、主要・重要サービスの基礎知識をやさしく学べる

図解

Amazon Web Services

# AWS<sup>(の)</sup>

仕組みとサービスが

# たった1日で よくわかる

インフラの  
知識ゼロから  
大丈夫!  
新入社員、  
非エンジニア  
必読!

感動的に  
わかりやすい!

NRIネットコム株式会社

上野史瑛／小林恭平／尾澤公亮／高梨友之

これから学び始める人の

# 最高のガイドブック

図解

Amazon Web Services

# AWS<sup>(の)</sup>

仕組みとサービスが  
たった1日で  
よくわかる

NRIネットコム株式会社

上野史瑛／小林恭平／尾澤公亮／高梨友之

## 本書に関するお問い合わせ

この度は小社書籍をご購入いただき誠にありがとうございます。小社では本書の内容に関するご質問を受け付けております。本書を読み進めていただきます中でご不明な箇所がございましたらお問い合わせください。なお、ご質問の前に小社 Web サイトで「正誤表」をご確認ください。

最新の正誤情報を下記の Web ページに掲載しております。

<https://isbn2.sbscr.jp/12818/>



上記ページのサポート情報にある「正誤情報」のリンクをクリックしてください。  
なお、正誤情報がない場合、リンクは用意されていません。

### ご質問送付先

ご質問については下記のいずれかの方法をご利用ください。

#### ▶ Web ページより

上記のサポートページ内にある「お問い合わせ」をクリックしていただき、ページ内の「書籍の内容について」をクリックすると、メールフォームが開きます。要綱に従ってご質問をご記入の上、送信してください。

#### ▶ 郵送

郵送の場合は下記までお願ひいたします。

〒 106-0032

東京都港区六本木 2-4-5

SB クリエイティブ 読者サポート係

■本書内に記載されている会社名、商品名、製品名などは一般に各社の登録商標または商標です。本書中では®、™マークは明記しておりません。

■本書の出版にあたっては正確な記述に努めましたが、本書の内容に基づく運用結果について、著者およびSBクリエイティブ株式会社は一切の責任を負いかねますのでご了承ください。

©2022 Ueno Fumiaki / Kobayashi Kyohei / Ozawa Kosuke / Takanashi Tomoyuki

本書の内容は著作権法上の保護を受けています。著作権者・出版権者の文書による許諾を得ずに、本書の一部または全部を無断で複写・複製・転載することは禁じられています。

## はじめに

AWS (Amazon Web Services) や各種のクラウドサービスの利用が広がり、クラウドの活用はビジネスに欠かせなくなっています。

クラウドが持つサービスや機能の種類も多様になり、そのすべてを理解することが難しくなっています。たとえば、AWS では現在 200 以上のサービスが存在します。新入社員のエンジニアや、非エンジニアの方がクラウド関連の会話に参加すると、会話に出てくる単語の意味がわからず苦労する場面も多くあるでしょう。

本書はそんな方々に向けて書かれた内容になっています。

多くの企業で使われている AWSについて、**できるだけ簡単な用語を使って、幅広く**解説しています。AWS の主要なサービスについて、サービス名を聞いたたらその基本機能をすぐに言える程度の理解度を目指して説明しています。図もふんだんに使用し、わかりやすさ重視で書いています。あえて詳細な機能説明には踏み込まず、短時間で幅広く AWS を理解できる内容となっています。

クラウドを活用するには IT の基礎知識も理解し、クラウド以前のシステム構築も知っておく必要があります。そこで、本書では**クラウドを理解するための前提となる知識**についても解説しています。たとえば IP アドレスや CIDR ブロックなど、名前は聞いたことはあるけれどよくわかっていないという方も多いのではないでしょうか。「クラウドを勉強したいけど、そもそも IT の基礎知識がない…」という方も、ぜひ本書で学習を始めてみてください。

筆者自身も、最初は 1、2 個程度の AWS サービスから触り始め、細かい部分はよくわからない状態で学習していました。わからないながらも、画面を操作するだけでサーバーが起動する簡単さに感動したことははっきり覚えています。

現時点でみなさんが AWS について何もわかっていなくてもまったく問題ありません。本書をきっかけに何か AWS、クラウドの好きな部分をまずは 1 つ見つけてもらったら幸いです。その 1 つをきっかけに、ぜひ理解を深めていってください。理解が深まると、日々進化する AWS の変化や新サービスも理解できるようになります。楽しくなることで、AWS の利用や学習がより積極的に進むでしょう。

理解を深めた読者のみなさんが、ビジネスでクラウドを活用し、他の方へクラウドの素晴らしさを共有いただけたら嬉しいです。

# Contents

はじめに .....	3
------------	---



## Amazon Web Services の基礎知識

9

— AWS (Amazon Web Services) とは	
<b>01</b> AWS を知るはじめの一歩 .....	10
— クラウドとオンプレミス	
<b>02</b> クラウド関連の用語を知っておこう .....	12
— AWS の特徴	
<b>03</b> AWS を理解する 6 つのポイント .....	18
— AWS の提供するサービス	
<b>04</b> サービスの分類と代表的なサービス .....	24
— AWS の導入事例	
<b>05</b> 日本国内の導入事例から利用イメージをつかもう .....	27



## Amazon Web Services の始め方

33

— AWS アカウントの作成とログイン	
<b>01</b> アカウントを作成して AWS の利用を始める .....	34
— AWS の操作	
<b>02</b> AWS サービスを操作する 3 つの方法 .....	38
— AWS 利用料の管理	
<b>03</b> AWS サービスの利用料を可視化する .....	42
— リージョンとアベイラビリティゾーン	
<b>04</b> AWS のシステムが構築される場所を選ぶ .....	44



## 3 コンピューティングサービス

49

01	サーバーとは 押さえておきたいサーバーの基礎知識	50
02	Amazon EC2 (Elastic Compute Cloud) とは EC2で仮想サーバーを手軽に作成できる	58
03	Amazon EC2 の外部公開とアクセス制御 仮想サーバーを安全に外部に公開する	64
04	Amazon EC2 Auto Scaling とは 負荷に応じてサーバーを自動的に追加・削除する	67
05	AWS Lambda とは サーバーを持たずにプログラムを実行する	69
06	AWS Lambda の実用的な使い方 サーバーレスの活用法をもっと理解する	74
07	コンテナとは コンテナの仕組みと特徴を理解する	81
08	Amazon ECS とは ECSでコンテナのメリットを享受する	86
09	その他のコンピューティングサービス サーバー知識なしで使える代表的なサービス5選	91



## 4 ストレージサービス

97

01	Amazon S3 (Simple Storage Service) とは Amazon S3はデータの保存場所	98
----	---	----

---

Amazon S3 の基本		
<b>02</b>	Amazon S3 の基本用語と基本操作.....	102
Amazon S3 のライフサイクルとバックアップ		
<b>03</b>	Amazon S3 のデータを適切に保存する.....	105
Amazon S3 の外部公開とアクセス制御		
<b>04</b>	Amazon S3 のデータを安全に公開する.....	109
Amazon EC2 のストレージ (EBS) とバックアップ		
<b>05</b>	仮想サーバーのデータは EBS に保存する.....	113
その他のストレージサービス		
<b>06</b>	データ共有・バックアップ・転送のためのサービス .....	116

---



## ネットワークとコンテンツ配信サービス 121

ネットワークの基礎知識		
<b>01</b>	ネットワークの重要用語を覚えよう.....	122
Amazon VPC とは		
<b>02</b>	Amazon VPC で仮想ネットワークを作る.....	130
Amazon VPC の主要機能①		
<b>03</b>	Amazon VPC の主要機能の使い方.....	134
Amazon VPC の主要機能②		
<b>04</b>	VPC 同士や外部サービス、オンプレミスとの接続 .....	139
Amazon VPC の構成例		
<b>05</b>	他の AWS サービスと組み合わせた構成例.....	143
ELB (Elastic Load Balancing) とは		
<b>06</b>	ELB で負荷分散して可用性を高める .....	146
Amazon Route 53 とは		
<b>07</b>	高性能で手軽に使える DNS サービス.....	149

---

<b>08</b>	Amazon CloudFront とは CloudFront で高速かつ落ちないネット配信を実現	156
-----------	--	-----

<b>09</b>	その他のネットワークとコンテンツ配信サービス 多彩なネットワーク機能を提供するサービス 7 選	161
-----------	--	-----



## 6 データベースサービス

169

---

<b>01</b>	データベースとは 押さえておきたいデータベースの基礎知識	170
-----------	---------------------------------	-----

<b>02</b>	Amazon RDS とは AWS でリレーショナルデータベースを使う	174
-----------	--	-----

<b>03</b>	Amazon Aurora とは Aurora の高機能で便利な特徴を知っておこう	182
-----------	--	-----

<b>04</b>	Amazon DynamoDB とは キーと値の組み合わせでデータを管理する	186
-----------	---	-----

<b>05</b>	Amazon Redshift とは データ分析のために大量データを取り込む	193
-----------	---	-----

<b>06</b>	AWS におけるデータベース移行 データベース移行に役立つサービス 2 選	195
-----------	--	-----

<b>07</b>	その他のデータベースサービス 多種多様なデータベースに対応するサービス 7 選	198
-----------	--	-----



## 7 セキュリティ、アイデンティティサービス

205

---

<b>01</b>	AWS におけるセキュリティ AWS のセキュリティをしっかり理解しよう	206
-----------	---	-----

02	AWS IAM (Identity and Access Management) とは IAM は AWS 利用時のセキュリティの要	209
03	AWS CloudTrail とは すべての操作を記録して証跡として残す	213
04	AWS Config とは 設定履歴や設定内容を自動で管理する	217
05	AWS GuardDuty とは AWS の怪しい操作を検知して不正を防ぐ	221
06	AWS WAF とは Web アプリケーションのセキュリティを強化する	224
07	その他のセキュリティ、アイデンティティサービス システムのセキュリティを強固にするサービス 6 選	227



8	知っておきたいその他のサービス	233
01	データ分析サービス 溜めたデータを次に生かすサービス 6 選	234
02	機械学習サービス お手軽に始める機械学習	240
03	マネジメントサービス システム自体を管理するサービス 4 選	244

Index	253
-------	-----



# Amazon Web Services の基礎知識

はじめに Amazon Web Services (AWS) を利用するうえで前提となる知識を紹介します。そもそもクラウドサービスとは何なのか、AWS では何ができるのか、という観点で解説していきます。

- section  
01 AWS (Amazon Web Services) とは  
AWS を知るはじめの一歩
- section  
02 クラウドとオンプレミス  
クラウド関連の用語を知っておこう
- section  
03 AWS の特徴  
AWS を理解する 6 つのポイント
- section  
04 AWS の提供するサービス  
サービスの分類と代表的なサービス
- section  
05 AWS の導入事例  
日本国内の導入事例から利用イメージをつかもう

# 01 AWS を知るはじめの一歩

**keyword** • クラウドサービス • パブリッククラウド

## Amazon Web Services は代表的なクラウドサービス

Amazon Web Services (以下 AWS) とは、インターネット通販で有名な Amazon.com が運営するクラウドサービスです。

ショッピングサイトやオンラインゲーム、社内の業務システムなど、何らかのシステムを構築する際にはコンピューターやデータベースなどの機能が必要ですが、インターネット経由でそれらの機能を利用できるサービスの総称をクラウドサービス、または単にクラウドと呼びます。

AWS は代表的なクラウドのひとつで、アプリケーションを稼働させるコンピューティング、データベース、ストレージ、モバイル、IoT、機械学習といったさまざまなサービスを提供しています。利用者はこれらのサービスを自由に使用して、あらゆるタイプのシステムを構築できます。

## AWS は代表的なクラウドサービス



## 多くの利用実績がある AWS

AWS のように誰でも利用できるクラウドのことをパブリッククラウドと呼びます。AWS 以外の有名なパブリッククラウドとして、グーグルが運営する **Google Cloud**、マイクロソフトが運営する **Microsoft Azure**<sup>アジュール</sup> があります。米国調査会社のガートナーの調査結果によると、パブリッククラウドの市場シェアは AWS が 40% と、他と比べ圧倒的な実績となっています。

### パブリッククラウドの売上と市場シェア

社名	2020年		2019年		2019年から 2020年の成長率 (%)
	売上 (100万ドル)	市場シェア (%)	売上 (100万ドル)	市場シェア (%)	
Amazon.com	26,201	40.8	20,365	44.6	28.7
マイクロソフト	12,658	19.7	7,950	17.4	59.2
アリババ*	6,117	9.5	4,004	8.8	52.8
グーグル	3,932	6.1	2,367	5.2	66.1
ファーウェイ*	2,672	4.2	882	1.9	202.8
その他	12,706	19.8	10,115	22.1	25.6
合計	64,286	100	45,684	100	40.7

(出典：<https://www.gartner.com/en/newsroom/press-releases/2021-06-28-gartner-says-worldwide-iaas-public-cloud-services-market-grew-40-7-percent-in-2020>)

\*アリババ、ファーウェイは中国の企業です。

日本国内でも多くのシステムや企業で AWS が利用されています。AWS の導入事例ページには、2022 年 1 月現在で約 200 の事例が紹介されています。利用実績が多い分、参考となるドキュメントや事例も多いです。学習用のコンテンツも豊富にあり、自習して AWS の理解を深められます。また、利用者による JAWS-UG (AWS Users Group - Japan) というユーザーグループもあり、勉強会が活発に行われています。読者のみなさんも本書をきっかけに、AWS の学習を積極的に進めていきましょう。

#### ▶ AWS クラウド導入事例

<https://aws.amazon.com/jp/solutions/case-studies/>

#### ▶ JAWS-UG

<https://jaws-ug.jp/>

## 02

クラウド関連の用語を  
知っておこう

**keyword** • オンプレミス • パブリッククラウド • プライベートクラウド • SaaS • PaaS  
• IaaS

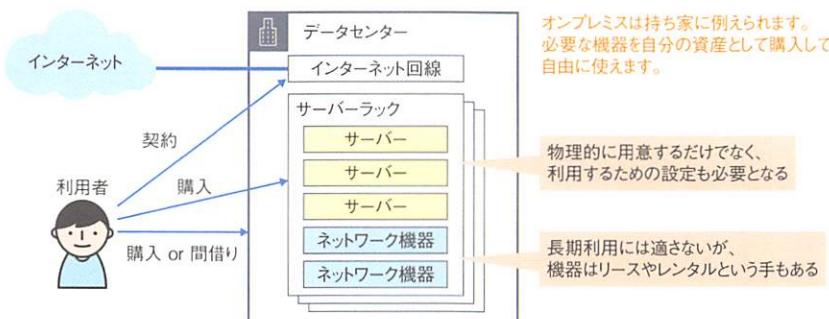
## 「クラウド」ってなんだろう？

AWSについて知っていく前に、「クラウド」そのものの意味や、情報システムに活用するうえでよく使われる関連用語を確認しておきましょう。ここでは、しばしばクラウドの対義語として使われる「オンプレミス」という言葉や関連技術、クラウドの細かな分類についての用語を説明します。

## オンプレミスとは

オンプレミス (on-premise) とは、利用者が管理する施設内にサーバーなどの機器を設置して運用することを指します。AWSのようなクラウドサービスが登場するまでは、情報システムを構築するには自前で機器を準備・設定する必要がありました。具体的には次図に示すようなものが必要となります。サーバーなどを設置する施設は一般的にデータセンターと呼ばれます。

## オンプレミスは自前で機器を準備・設定する



もちろん、購入するだけでなく物理的な設置やケーブルの配線、さらにはサーバーやネットワーク機器の設定が必要となります。こういった作業はベンダーに委託することも可能ですが、いずれにせよ機器の購入費、作業にかかる人件費といった初期投資が高くなりがちです。また、設置や設定にも時間がかかるので、実際に情報システムを構築開始できるようになるまでの準備期間が長いです。

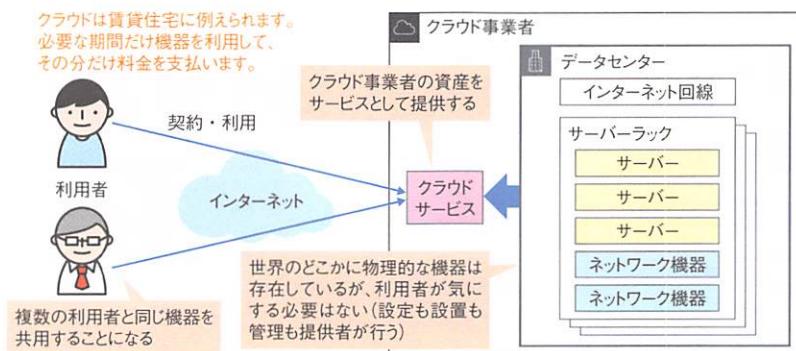
その代わり、機器は利用者が自由に使えることになるので、利用形態に合わせた複雑な構成をとることも可能です。初期投資は高いですが、その後にかかる費用は電気代やインターネット回線利用料といったもののみになるので、全体的なコストから見たランニングコストは比較的低くなります。ただし、故障した機器の修理といった突発的な出費が発生する可能性はあります。

## クラウドとは

**クラウド (cloud)** は、クラウドサービスの提供者がサーバーなどの機器を準備し、そこに構築した仮想サーバーやアプリケーションなどを利用者に提供する代わりに、その利用料を徴収する形態です。物理的なサーバーやネットワークは提供者が準備してくれているので、利用者はすぐに情報システムの構築に着手できます。

なお、「クラウド」は略称で、正式な名称は**クラウドコンピューティング (cloud computing)** といいます。はっきりとした語源は定かではないですが、旧来からネットワークを雲の図形で表すことがあり、その雲の向こう側にあるコンピュータ群というような意味で使われ始めたといわれています。

## クラウドはサービス事業者が機器を準備・設定する



サーバーやアプリケーションがサービスとして提供されるので、オンプレミスと違って利用者は機器の管理を心配する必要はありません。機器が故障しても提供者の責任で修理してくれることになります。

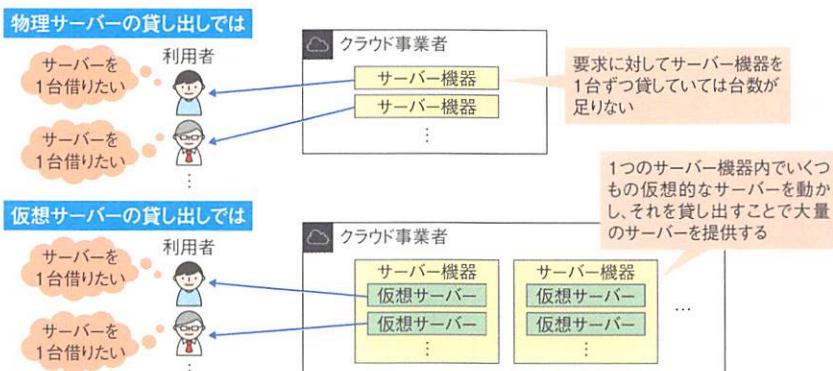
一方、物理的な機器の構成などは提供者が決め、利用者は提供されているサービスに沿って利用することとなるため、システム構成の自由度は下がります。また、長い目で見ると機器を買い切りで入手して使うオンプレミスに比べ、利用料を支払い続けることになるクラウドは総コストが高くなる傾向にあります。しかし、機器の耐用年数や故障の心配なく利用し続けることができるというメリットもあります。

## 仮想化

クラウドサービスでは事業者が用意したサーバー機器の一部を利用者に貸し出しますが、利用者にサーバー機器を1台ずつ貸し出しているわけではありません。**仮想化**という技術を使い、1つのサーバー機器内に**サーバー機器のように動作するプログラム**を複数用意し、その1つひとつを貸し出しています。このサーバー機器のように動作するプログラムは**仮想サーバー**と呼ばれ、それを動かすサーバー機器は**物理サーバー**と呼ばれます。

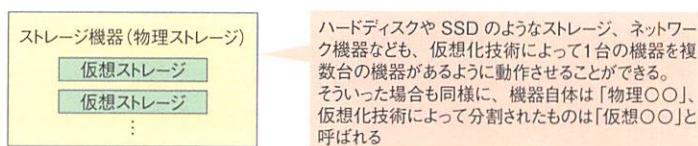
仮想サーバーは、それを動かしているサーバー機器のCPUやメモリといったコンピュータリソースを一部専有し、それぞれが独立したサーバーとして動作します。

### 仮想化技術で1台の機器を複数台のように利用する



また、仮想化ができるのはサーバーだけではありません。ストレージやネットワーク機器なども1台の機器を仮想化によって複数の機器のように扱うことができます。基本的に仮想化によって分割されたものは「仮想〇〇」と呼ばれることを押さえておきましょう。

## いろいろな機器が仮想化できるようになっている

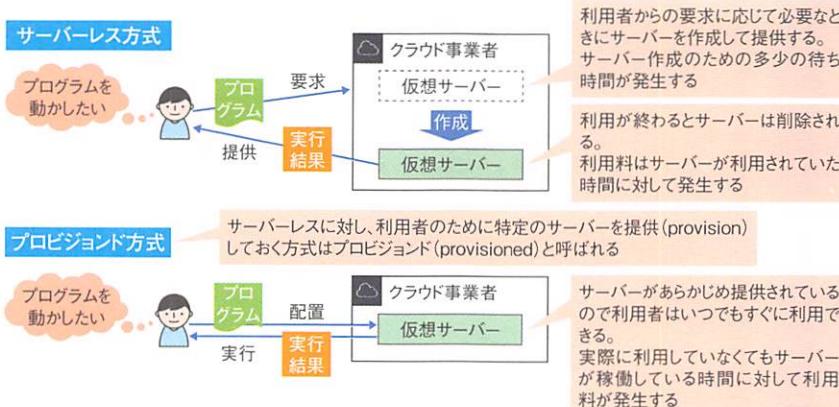


## サーバーレス

クラウドサービスでは**サーバーレス (serverless)** という言葉もよく使われます。直訳のとおり「サーバーを持たない」という意味で、普段からサーバーを稼働させておくのではなく、サービスが利用されるときにだけサーバーを稼働させる方式を指して「サーバーレスなサービス」というように表現します。

クラウドサービスは、一般的にサーバーの提供などによる「コンピュータリソースを利用者に専有させる時間」に対して料金が発生します。サーバーレスなサービスであれば、課金の対象となる時間を最小限にできるというメリットがあります。

## サーバーレスは利用するときだけサーバーを稼働させる



## パブリッククラウドとプライベートクラウド

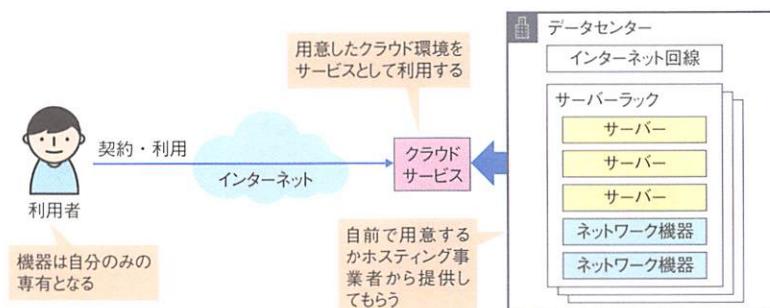
クラウドの形態には、クラウドの動作するサーバーなどの機器を利用者が専有する**プライベートクラウド**と、機器を他の利用者と共有する**パブリッククラウド**というものがあります。AWSはすべての機器をAWS側が管理し、その一部を利用者に貸与するという形であり、利用者の専有できる機器は無いので**パブリッククラウド**にあたります。

もう少し細かく言うと、AWSには専用の機器をユーザーに割り当てるペアメタル、仮想的にユーザーの占有空間を提供するVirtual Private Cloud (VPC)というサービスがありますが、利用が終われば同じ機器を別の利用者に貸すこともあるため、厳密にはそれらも**パブリッククラウド**となります。

プライベートクラウドを利用するには、オンプレミスのように利用者が管理する施設内にクラウド環境を構築するか、利用者専用のクラウド環境を構築し、貸し出してくれるホスティングサービスを利用する方法があります。

とはいっても、VPCのような仮想的な専有空間でもセキュリティは十分に確保できることと、オンプレミスに近い形態である**プライベートクラウド**は初期投資が高くなる傾向にあるため、現在ではクラウドを利用するとなれば基本的に**パブリッククラウド**を選択することが多くなっています。

### プライベートクラウドは機器を専有するクラウドの形態



## 利用形態によるクラウドの分類

ひとことにクラウドと言っても、サービスとして提供されるものの内容によって、いくつかの分類があります。一般的な分類として **SaaS**、**PaaS**、**IaaS** というものがあります。

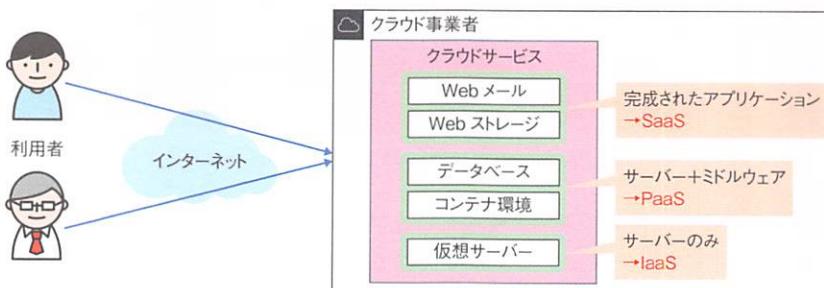
**SaaS** (Software as a Service) は、アプリケーションをサービスとして提供する方法です。具体的なサービス名を挙げると、Gmail、Dropbox、Office 365、Zoom などがあります。普段これらの Web アプリケーションを利用している方も多いと思いますが、これらもクラウドの一種なのです。

**SaaS** はそれだけで利用できるアプリケーションを提供しますが、**PaaS** (Platform as a Service)、**IaaS** (Infrastructure as a Service) はアプリケーションを作るための部品をサービスとして提供する方法です。こちらはアプリケーションを提供する側に向けたサービスであり、利用者はこれらを組み合わせてアプリケーションを提供します。**PaaS** と **IaaS** の違いは、クラウドサービス提供側の管理する範囲です。

**PaaS** の場合はクラウドサービス提供側が OS やミドルウェアまでを管理し、機能のみを利用者に提供します。AWS ではいわゆる **マネージドサービス** と呼ばれるもので、RDS や DynamoDB、Lambda などがこれにあたります。利用者はそれらの機能だけを利用し、メンテナンスは AWS に任せることができます。一方、設定は AWS の提供する範囲でしか変更できません。

**IaaS** はサーバーやネットワーク機能だけが提供され、それらの設定や管理を利用者が行う形態です。AWS では EC2 や VPC、EBS などがこれにあたり、利用者が OS 設定のような細かなレベルで設定を管理できます。

### クラウドが提供するサービスは SaaS、PaaS、IaaS に分類される



## 03

AWS を理解する  
6つのポイント

## keyword

- ・責任共有モデル
- ・リージョン
- ・アベイラビリティゾーン
- ・従量課金
- ・Design for Failure
- ・AWS Well-Architected フレームワーク

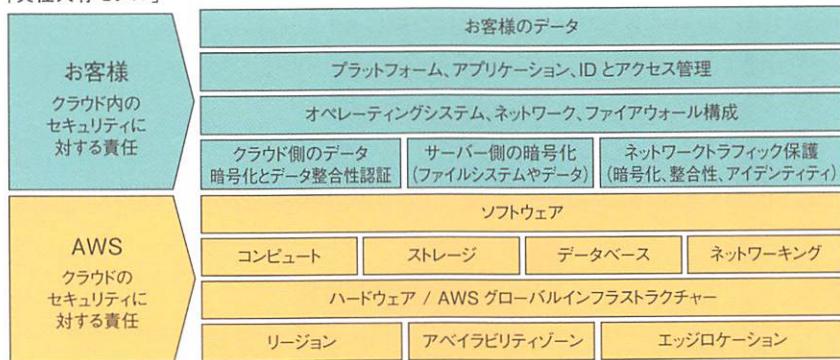
## AWS と利用者で責任を共有する

AWS の特徴を、オンプレミスのシステムとの比較も行いながら見ていきます。

オンプレミスではサーバー機器などハードウェアに故障があった場合は、利用者の責任で利用者が修理、復旧作業を行います。AWS ではハードウェアに故障があった場合、AWS の責任で復旧が行われます。AWS ではこの責任の範囲を「**責任共有モデル**」という形で示しています。

## 責任共有モデルで、AWS 側と利用者側の責任範囲が規定されている

## 「責任共有モデル」

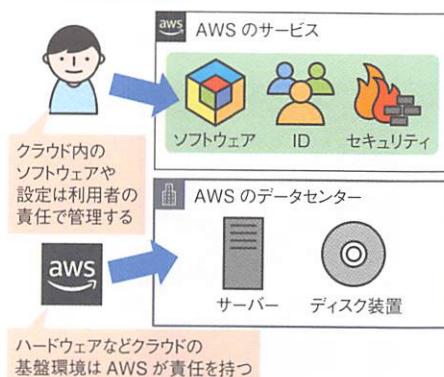


(出典：<https://aws.amazon.com/jp/compliance/shared-responsibility-model/>)

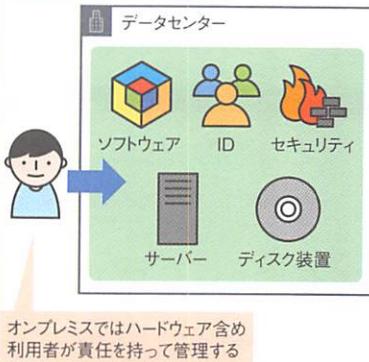
利用者はその部分を除いて運用、管理を行えばよく、負荷の軽減につながります。責任の範囲は利用するサービスによって異なります。本書では、第3章からの各サービスの解説の中で、特に責任範囲を意識しておきたい場合に限って言及しています。

## ハードウェアの管理はクラウドに任せられるので管理負荷が減る

### AWSでの責任範囲



### オンプレミスでの責任範囲

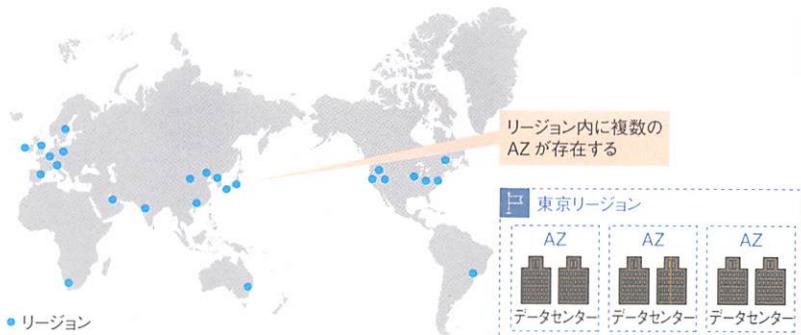


## グローバルなシステムを構築できる

AWSが管理するデータセンターは世界中に存在します。地域ごとにリージョンという単位で分離されており、本書執筆時点では26のリージョンが存在します。日本で通常利用する場合は東京リージョン、大阪リージョンを選択します。

各リージョンには、アベイラビリティゾーン（以下AZ）が複数存在します。AZは1つ以上のデータセンターから構成されます。リージョン内の各AZは独立した場所にあるため、たとえばデータセンター障害のような大規模障害が発生しても、別のAZでサービスを提供できます。

## AWSのデータセンターは世界中の地域に存在している



使用するリージョンは利用時に選択すればよいだけなので、利用者は簡単にグローバルなシステムを構築できます。

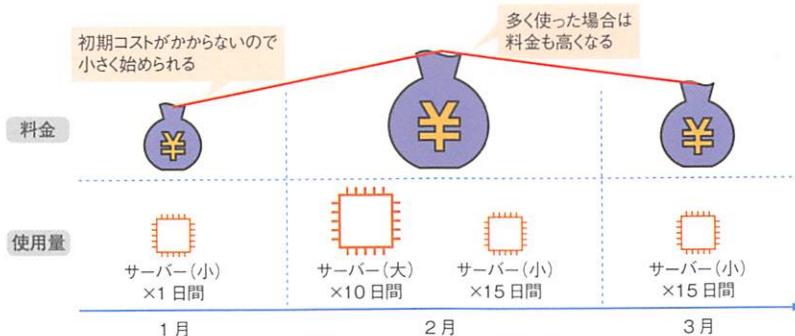
## システムを構築するリージョン（地域）は簡単に選べる



### 使った分だけ利用料を支払う

AWS の利用料金は**従量課金**です。多くのサービスは 1 時間あたりの料金が設定されており、その料金 × 使用時間という形で利用料金が発生します。オンプレミスの場合はハードウェアの購入など初期コストが多く発生しますが、AWS では使った分のコストのため、小規模でスタートする、まずは試してみるといったことが簡単にできます。

### AWS の利用料金はサービスの料金 × 使用時間



具体的な料金の例を仮想サーバーサービスである EC2 で見てみましょう。EC2 の場合、検証用の小さなサーバーを 24 時間 × 30 日間起動していたとすると、次表のような料金が発生します。

### 利用料金例：検証用の小さなサーバーを 24 時間 × 30 日稼働した場合

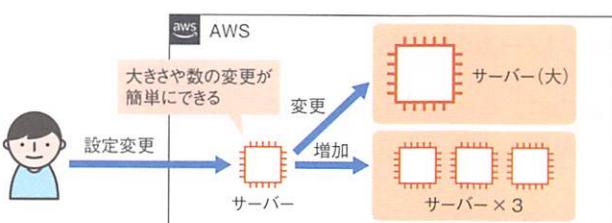
料金タイプ	設定値	料金(単価)	料金(30日間)
サーバー (インスタンス)	タイプ: t3.micro	0.0136USドル ／1時間	9.792USドル
データ容量 (EBS)	容量: 20GB (汎用SSD)	0.096USドル ／1GB	1.92USドル

合計 11.712US ドル、日本円で約 1,288 円となります (1US ドル = 110 円で計算)。サーバーのタイプなど設定値によって変動する点に注意してください。コストの管理は難しい部分もあります。コストの管理方法は 2-3 節 (p.42) で紹介します。

### システムの増減が自由にできる

AWS で作成したサーバーは、大きさ (CPU、メモリ量) や数を簡単に変更できます。構築したサービスの利用者が増えた場合にサーバーを増やすといった、柔軟な対応が可能です。

### サーバーの大きさや数を設定で簡単に変更できる

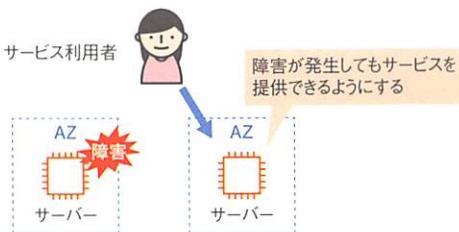


## 障害を前提に考える

AWSでは、**障害は必ず起こる前提で設計する**という考え方があります。障害が発生しない方法を考えるのではなく、障害が発生してもサービスを継続的に提供できるよう考えるという意味です。この考え方を「**Design for Failure**」と呼びます。

パソコンなどの機器を使用していて、動作の不具合や故障を経験したことがあると思います。AWSも同じで、必ず一時的な不具合や故障は発生します。サーバーを配置できるAZが複数存在するため、複数のAZに配置し、1カ所またはAZ全体の障害があった場合もサービス提供を継続できるようにします。システムが継続して稼働できる能力のことを、**可用性**と呼びます。

### 複数のAZにサーバーを配置して障害に備えるのが基本



ストレージサービスのS3のように、AWS側で自動的に複数のAZに配置されたり、自動的にデータ領域の拡張が実施されるサービスもあります。

## 設計のお手本となるフレームワークがある

AWSには「**Well-Architected**」という考え方があります。直訳すると「良く設計された」となりますが、AWS社員や多くの業界での設計・構築の考えが詰め込まれたものです。

このような考え方が、**AWS Well-Architected フレームワーク**という形で用意されています。利用者はこのフレームワークを参考にして、AWSならではのポイントを学ぶことができます。

考え方の分類として、「運用の優秀性」「セキュリティ」「可用性」「パフォーマンス効率」「コスト最適化」「持続可能性」の6種類があり、AWSでは6本の柱と呼

んでいます。「持続可能性」は、2021年12月に追加された新しい柱です。

詳細については、AWSが提供する次の資料に書かれています。

#### ▶ AWS Well-Architected フレームワーク Web サイト

<https://wa.aws.amazon.com/index.ja.html>

#### ▶ AWS Well-Architected フレームワーク 日本語ホワイトペーパー (PDF)

[https://d1.awsstatic.com/whitepapers/ja\\_JP/architecture/AWS\\_Well-Architected\\_Framework.pdf](https://d1.awsstatic.com/whitepapers/ja_JP/architecture/AWS_Well-Architected_Framework.pdf)

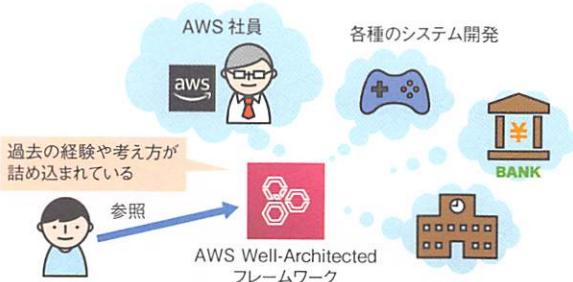
初心者の方が、いきなりこれを読んでから構築を進めるわけではありませんが、大切な考え方なので、こういったお手本があるということは覚えておきましょう。

また、内容が難しい場合、AWSまたはWell-Architectedパートナーと呼ばれるフレームワークに詳しい企業に相談も可能です。

#### ▶ AWS Well-Architected パートナープログラム

<https://aws.amazon.com/jp/partners/programs/well-architected/>

### AWSを利用するに際してお手本となる考え方が始まられている



# 04 サービスの分類と代表的なサービス

**keyword** • AWS の代表的なサービス • マネージドサービス

## AWS サービスの分類

AWS の提供するサービスは日々増え続け、世界のさまざまなニーズに応じたサービス追加や改良が繰り返されています。200種類以上あり、すべてを利用することはないといますが、どういったものが提供されているかだけでも把握しておくことで、より有効に AWS を活用できるでしょう。

AWS のサービスを把握する方法としては、**サービスの特性ごとに分類して理解するのがおすすめ**です。AWS の公開しているドキュメントも以下の URL のように分類されて整理されています。

### ▶ AWS のドキュメント

[https://docs.aws.amazon.com/ja\\_jp/](https://docs.aws.amazon.com/ja_jp/)

本書においても、AWS 公式の分類とはまったく同じではありませんが、サービスの分類ごとに章を分け、同じようなサービスごとに解説する形をとっています。

#### • コンピューティング (→第 3 章)

アプリケーションやミドルウェアを動作させるための環境を提供するサービスです。いわゆる仮想サーバーを提供します。AWS 公式の分類では分かれていますが、本書ではコンテナサービスもコンピューティングサービスと併せて説明します。

#### • ストレージ (→第 4 章)

データを保管するためのサービスです。データ保管といっても、データを整理する役割も含むデータベースサービスは別分類とされており、単にデータを保管するためだけのサービスと考えてください。いわゆる仮想ディスク、仮想ストレージといったものがこの分類にあたります。

### ・ネットワークとコンテンツ配信（→第5章）

各サービスをつなぐネットワーク、ユーザーにコンテンツを提供するための機能を持つサービスです。AWS内部の仮想ネットワーク、オンプレミス環境とAWSをつなぐ専用線、DNS、CDNといったサービスが含まれます。

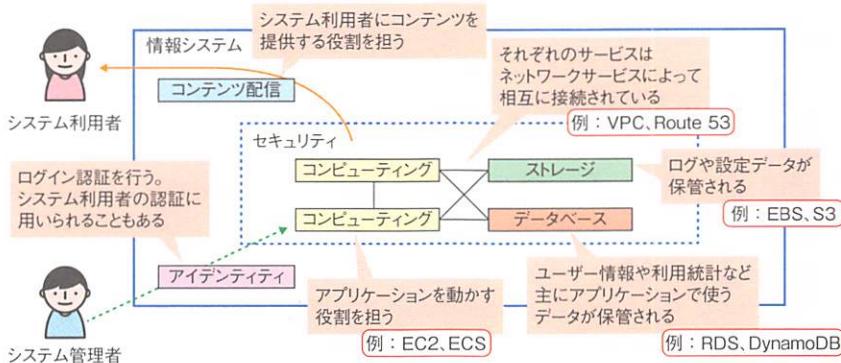
### ・データベース（→第6章）

さまざまな形式のデータを整理して保管し、検索や集計を可能にするデータベースを提供するサービスです。データの整理と柔軟な検索に長けたリレーショナルデータベース（RDB）や、単純な構造のデータを高速に扱うkey-value型データベース、大量データの集計が得意な列指向データベースといったさまざまなデータベースのためのサービスが存在します。また、データベースに格納されるデータはアプリケーションの要となることが多いので、データの保全機能が充実しているのも特徴です。

### ・セキュリティ、アイデンティティ（→第7章）

システムをサイバー攻撃から守るためのサービスも多数提供されています。また、AWS自体のログイン管理やサービスを利用するための認証を行うサービスもさまざまな形式で提供されています。本書ではセキュリティの一環である暗号化サービスについても併せて説明します。

## AWSのサービスの分類は一般的な情報システムの機能に当てはまる



- その他（→第8章）

ここまで紹介したもの以外にも、データ分析や機械学習、システム管理、開発用ツール、さらにはIoTやコールセンター、動画配信、オンライン会議など、非常に幅広い分野のサービスが提供されています。本書ではその中でも比較的よく利用されるものを中心に紹介します。

## 代表的なサービス

前項に掲載した図は一般的な情報システムの構成図です。もちろんシステムごとに構成は異なりますが、基本的にアプリケーションを動作させる部分、データを保管する部分を中心に構成されます。具体的なAWSサービスでいうと、コンピューティングサービスとしては仮想サーバーであるEC2、コンテナサービスであるECSがよく利用され、ストレージにはEBSやS3、データベースにはRDSやDynamoDBがよく使われます。

ネットワークにおいては仮想的なプライベートクラウド環境を実現するVPC、高機能なDNSであるRoute 53、複数のコンピューティングサービスに処理を分散するELBといったものが多くのシステムで利用されているといえるでしょう。

サービスの種類が非常に多いとはいえ、特定の用途に特化したサービスも多いため広く使われているものは限られており、LightsailやBeanstalkのように上述のよく使われるサービスを組み合わせて簡単に使えるようにパッケージ化したサービスもあります。よって、まずは代表的なサービスから理解していくのがAWSを知る近道となります。本書でも代表的なサービスに絞って解説する形式としています。

## マネージドサービス、非マネージドサービス

クラウドサービスではマネージドサービス（Managed Service）と呼ばれるものがあります。これは機能を利用するための管理や運用（システムマネジメント）を事業者が行うものを指し、AWSではRDSやS3など大半のサービスがこれにあたります。さらにデータ量に応じた保存領域の拡張機能を備えたAuroraのように、利用者のメンテナンスがほぼ必要ないサービスは特にフルマネージドサービス（Full Managed Service）とも呼ばれます。

逆にEC2のように利用者が自分でOSレベルの設定管理をしたり、障害対応を行う必要のあるサービスは非マネージドサービスと呼ばれます。

## 05

日本国内の導入事例から  
利用イメージをつかもう**keyword** • AWS の導入事例

## AWS が公開している事例を見てみよう

AWS は世界中ではもちろん、日本国内の企業にも数多くの導入事例があります。AWS の公式サイトに多数の事例が紹介されていますが、ここではその一部を紹介します。

## ▶ AWS クラウド導入事例

<https://aws.amazon.com/jp/solutions/case-studies-jp/>

## ゲーム業界の事例

AWS の代表的な特徴ともいえるリソースの柔軟性、グローバル性が力を発揮するのがゲーム業界です。ゲーム内のイベントによるアクセスの爆発的な増加に対応するための一時的なリソースの強化や、そもそもリソース不足という概念のないサーバーレスサービス、世界各地のユーザーにデータを配信するためのコンテンツ配信サービスなどが活用されています。

・任天堂株式会社／株式会社ディー・エヌ・エー

URL <https://aws.amazon.com/jp/solutions/case-studies/nintendo-dena-2020/>

任天堂株式会社が株式会社ディー・エヌ・エーと協業して配信するゲーム「マリオカート ツアー」ではデータベースに Amazon Aurora (以下 Aurora) が採用されています。2019 年に全世界同時リリースされましたが、リリース時のアクセス増加に耐えるため、予測の 3 倍を処理できる規模のクラスタ (サーバー群) が用意されました。具体的には Aurora のインスタンス数は 1,200 台で、Aurora 全体の 1 秒あたりのクエリ数は最大で約 30 万を記録し、データ量も 1 カ月で 30TB ま

で達しましたが Aurora は安定したレスポンスを返し続けたそうです。フルマネージドかつ簡単にリソースが増やせる AWS の特徴を生かした事例といえるでしょう。

⇒Amazon Auroraについては第6章で解説しています。

#### ・株式会社スクウェア・エニックス

URL <https://aws.amazon.com/jp/solutions/case-studies/square-enix/>

株式会社スクウェア・エニックスの提供する「ドラゴンクエストX」ではゲーム内に写真撮影機能があり、その写真はサーバーに転送され、サーバー上で加工されて保管されます。通常は 1 分間に 2 ~ 300 枚程度の処理数ですが、大みそかのイベントの際には 1 分間に 6,000 枚まで増え、画像処理が終わってユーザーが写真を閲覧できるようになるまでに、長いときは 3 ~ 4 時間もかかることがありました。当初は画像処理サーバーの増設を考えたそうですが、年に数回のイベントのために高価なサーバーを導入するより、AWS Lambda を利用することで理論上無限にスケールでき、かつ利用した分だけの費用増加にとどめられる構成をとることになりました。結果、1 分間に 18,000 枚の画像処理を数秒から 10 数秒で完了できるようになり、費用もオンプレミスで行う場合と比べ 20 分の 1 度程にまで削減されたとのことです。サーバーの他の処理によるリソース逼迫の影響を受けないサーバーレスサービスの特徴と、従量課金制である費用面のメリットが生きた事例です。

⇒AWS Lambdaについては第3章で解説しています。

## ヘルスケア業界の事例

ヘルスケア業界では膨大なデータの解析能力だけでなく、データ自体が患者の要配慮個人情報となるゲノム情報や既往歴であったり、創薬のための機密情報であったりと高度なセキュリティが求められます。AWS ではデータ解析に有用となるハイパフォーマンスな CPU や GPU を搭載したコンピューティングサービス、膨大なデータを保持するストレージサービスが提供されており、Amazon VPC（以下 VPC）や AWS WAF などによるセキュリティ確保が可能なため、ヘルスケア業界にも徐々に導入が進んでいます。

### ・エーザイ株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/eisai/>

エーザイ株式会社の創薬における研究開発には、AWS のハイパフォーマンスコンピューティング (HPC) インスタンスが採用されています。導入が進んだ背景には、VPC によるセキュリティ確保が可能となったことに加え、50TB にもなるオンプレミスのデータを AWS Snowball によってセキュアに移行できたことがあります。オンプレミス環境では数百コアが限界で、他の研究員と利用時間の調整が必要だった解析環境も、AWS 移行後は数千コアレベルのものが用意でき、かつ必要なときに目的に応じた HPC 環境を立ち上げられるようになりました。オンプレミスだと非常に高価となるハイパフォーマンスな環境を必要なときだけ用意し、いらないときは削除してコストを抑えられるという、AWS の有効な使い方ともいえる事例です。

⇒Amazon VPCについては第5章、AWS Snowballについては第4章で解説しています。

### ・シスメックス株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/sysmex/>

シスメックス株式会社が国立研究開発法人国立がん研究センターと共同開発した、がんゲノム医療で利用される「OncoGuide NCC オンコパネルシステム」では、短い開発期間で厚生労働省、総務省、経済産業省の 3 省が定めた 2 つの医療情報システムに関する各ガイドラインに対応するシステムを実現するために AWS が採用されています。要件やルールの変更に合わせてトライ＆エラーしながらの柔軟な開発や、マネージドサービスの活用による運用効率とセキュリティの両立といった AWS のメリットが取り込まれています。今後もニーズに合わせた改良や機能拡張が予定されるシステムとのことで、そういう点でも AWS の拡張性が期待できる事例だといえます。

## 製造業界の事例

製造業界の事例は AWS の公式サイトでも特に多く紹介されています。オンプレミスシステムの AWS 移行や IoT を活用した新サービスの構築、AI やデータ分析といった AWS サービスを利用したシステムの構築など、多岐にわたる事例があります。

・東海理研株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/tokairiken/>

東海理研株式会社は、Amazon Rekognition を採用した顔認証入退室管理プラットフォームの事例を公開しています。この事例の特徴としては、プロジェクトのスタート時点で同社内にクラウドや画像認識のための機械学習の専門家はいなかったことが挙げられます。その状態から AWS のトレーニングや自主学習、セミナーといったサポートを利用しつつ自社のスタッフを教育して開発を進めたとのことです。「AWS は、他のクラウド事業者よりも学習リソースが整っており、テキストやセミナーが豊富です。本気で学ぶ意欲があれば、当社のようにゼロからでもサービスを構築できるのが魅力です」とコメントがあるとおり、サービスの使い方についてのドキュメントだけでなく、深く理解するための資料が充実しているのも AWS の魅力だといえるでしょう。

⇒ AWSの機械学習サービスについては第8章で解説しています。

・ヤマハ発動機株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/yamaha-nri/>

ヤマハ発動機株式会社は、株式会社野村総合研究所の支援を受け、顧客と販売員のやりとりのデータを収集・分析する POS システム構築をわずか 3 ル月で実装しています。クラウドネイティブと呼ばれる「クラウドの利点を徹底的に活用する」という設計思想のもと、AWS のマネージドサービスをフル活用したサーバーレスなシステムが構築されました。GraphQL API を提供する AWS AppSync をはじめ、ユーザー認証を行う Amazon Cognito、データ格納のための Amazon DynamoDB といったマネージドサービスで各機能が構成され、開発量を大幅に抑えています。市場の変化に素早く対応してシステムを整備する必要のある、製造業界ならではの事例です。

⇒ Amazon Cognitoについては第7章、Amazon DynamoDBについては第6章で解説しています。

## 金融業界の事例

金融業界でも AWS は広く活用されています。PCI DSS (Payment Card Industry Data Security Standard) のような高度なセキュリティ基準に対応することも可能で、金融資産を扱う業種であるゆえに、より迅速に顧客ニーズに対応するためのシステム改修が行えるという観点から AWS が選択されています。また、スケーラブルなシステムが構築しやすく、高い信頼性を持つシステムが実現しやすいというのも理由のひとつでしょう。

### ・ PayPay 株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/paypay/>

PayPay 株式会社の提供するサービス「PayPay」は、開発決定から利用開始までをわずか 3 カ月という短期間で実現しました。各社がキャッシュレス決済の展開を始める中、いち早く市場シェアを勝ち取るための戦略であったとのことで、事実 PayPay はキャッシュレス決済サービスの中でも多くのシェアを獲得することに成功したといえるでしょう。さらなるサービスの拡大を視野に入れたアップデートを継続できるのも AWS の利点だといえます。

### ・ 三井住友海上火災保険株式会社

URL <https://aws.amazon.com/jp/solutions/case-studies/ms-ins-nec/>

三井住友海上火災保険株式会社の提供する代理店システム「MS1 Brain」は、AI を利用した顧客一人ひとりに最適化した保険商品やサービスを提案するためのシステムです。蓄積された顧客情報、事故情報、家族構成の変化といったビッグデータを解析し、顧客ニーズの予測分析、提案活動における最適なアクションの提示、AI が導き出した最適な保険プランを説明するパーソナライズド動画、販売計画策定の支援という機能を提供します。これまで KKD (カン・経験・度胸) といわれる人間の感覚のみに頼っていた判断に、AI の導き出す結果を参考情報として加えることで、効率の向上を目指すという考え方です。機密性の高い大量のデータを、セキュリティに配慮しつつ高速に分析する基盤が必要となる、代表的な事例です。

⇒ AWS のデータ分析サービスについては第8章で解説しています。

## column

### AWS の学習方法

本書以外にも AWS を学習するにはさまざまな方法があり、AWS 公式の学習コンテンツも数多く存在します。特に AWS を初めて学ぶ方にとって役立つコンテンツをいくつか紹介します。

- AWS 初心者向け資料

<https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-level-100/>

初心者向けの学習資料、動画がまとめられているページです。AWS 公式のコンテンツが多数掲載されており、すべて無料です。

- AWS 初心者向けハンズオン

<https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/>

実際に AWS 環境を触って学べる、ハンズオン形式の動画セミナーです。実際に AWS 環境を触って勉強したい方におすすめです。かなり丁寧に手順を説明してくれるので、初心者の方でも問題なく構築できるでしょう。AWS アカウントは自分で用意する必要があり、わずかに利用料金が発生する点は注意が必要です。コンテンツの視聴は無料です。

- AWS Skill Builder（トレーニングポータルサイト）

<https://explore.skillbuilder.aws/>

2021 年にリニューアルされた、AWS 公式トレーニングのポータルサイトです。多くのトレーニングコンテンツが無料で提供されており、今後もコンテンツは拡充されていくでしょう。

- 認定試験

<https://aws.amazon.com/jp/certification/>

本書を読み終えて AWS の基礎が理解できた方は、次のステップとして AWS 公式の認定資格取得に向けて勉強してみるとよいでしょう。試験合格という明確な目標があると、学習も進めやすいです。AWS には複数の認定試験が用意されていますが、まずは基礎コースとなっているクラウドプラクティショナー合格を目指しましょう。本書は特に試験対策向けの内容となっているわけではないので、上記に紹介したコンテンツも参考に、試験に向けて学習してみてください。



# Amazon Web Services の始め方

---

AWS の概要がつかめたら、早速利用を始めましょう。この章では AWS 利用で必須となる AWS アカウントの取得や利用料の確認方法など、最初にやるべきことを紹介します。

---

- section  
01 AWS アカウントの作成とログイン  
アカウントを作成して AWS の利用を始める
- section  
02 AWS の操作  
AWS サービスを操作する 3 つの方法
- section  
03 AWS 利用料の管理  
AWS サービスの利用料を可視化する
- section  
04 リージョンとアベイラビリティゾーン  
AWS のシステムが構築される場所を選ぶ

## 01

アカウントを作成して  
AWS の利用を始める

**keyword** • AWS アカウント • AWS のサポートプラン • ルートユーザー • IAM ユーザー

## AWS アカウントの作成

第1章では、AWS サービスの全容と提供されるサービスについて大まかに説明しました。ここでは実際に AWS サービスの利用を始めるための準備について説明します。紙面上で流れを押さえてもらえばよいので、実際に作業していただく必要はありませんが、余裕のある方は取り組んでみてください。

AWS サービスの利用のためには、必ず利用者固有のアカウントへのログインを行う必要があります。このアカウントのことを **AWS アカウント** と呼びます。AWS アカウントには以下の情報を登録します。

- メールアドレス
- 連絡先情報（フルネーム、電話番号、住所）
- クレジットカード情報

上記の情報の準備ができたら早速アカウントを作成していきましょう。アカウントを作成するには、以下の新規登録用の URL にアクセスします。

## ▶ AWS アカウントの新規作成

<https://portal.aws.amazon.com/billing/signup#/start>



- 1 メールアドレス、パスワード、アカウント名の設定
- 2 連絡先情報の設定
- 3 クレジットカード情報の設定
- 4 SMS または音声電話での本人確認
- 5 サポートプランの選択

## アカウントを作成してAWSの利用を始める

前図のような画面が表示されたら、必要情報を入力していきます。図の右側に示した5ステップで行われます。注意点として、登録するメールアドレスは、AWSアカウントごとに固有でなくてはなりません。

また、AWSアカウント作成時には、**そのアカウントで利用可能なAWSサポートプラン**を選択できます。プランは、無料サポートの「ベーシック」、有料でより手厚いサポートが提供される「デベロッパー」「ビジネス」「エンタープライズ」の4種類があります。次表に簡単な比較をまとめましたので、用途によって最適なものを選択してください。AWSから直接技術サポートを受けたい場合は、有料サポートプランを利用する必要があります。

## AWSのサポートプラン。技術サポートを受けたい場合は有料

プラン	主な用途	利用料(月額)	料金の質問	技術的質問	技術問い合わせ方法	ケース応答時間
ベーシック	デフォルトで利用可能	無料	○	×	インスタンスのヘルスチェック 不合格時のみ	-
デベロッパー	学習・テストなど、個人向け	最低： 29 USD	○	○	メール (平日9:00～18:00)	<ul style="list-style-type: none"> <li>一般的なガイダンス：24時間以内</li> <li>システム障害：12時間以内</li> </ul>
ビジネス	本番環境ワークロード利用者	最低： 100 USD	○	○	電話、チャット、メール(24時間)	<ul style="list-style-type: none"> <li>一般的なガイダンス：24時間以内</li> <li>システム障害：12時間以内</li> <li>本番システムの障害：4時間以内</li> <li>本番システムのダウン：1時間以内</li> </ul>
エンタープライズ	ビジネスおよび重要なワークロード利用者	最低： 15,000 USD	○	○	・電話、チャット、メール(24時間) ・担当者への直接連絡	<ul style="list-style-type: none"> <li>一般的なガイダンス：24時間以内</li> <li>システム障害：12時間以内</li> <li>本番システムの障害：4時間以内</li> <li>本番システムのダウン：1時間以内</li> <li>ビジネス／ミッションクリティカルなシステムのダウン：15分以内</li> </ul>

画面の指示に従って登録を完了すると、登録メールアドレスに確認メールが送付されます。これでAWSを使い始める準備ができました。

## AWS アカウントへのログイン

前項で AWS アカウントの作成が完了しました。次は実際にブラウザからアカウントへのログインを行い、実際に AWS サービスへアクセスしてみましょう。

先ほど作成した AWS アカウントを利用するには、ユーザーとしてアカウントにログインする必要があります。初めてのログインには、アカウント作成と同時に自動的に作成される、**ルートユーザー**と呼ばれる特別なユーザーでログインを行います。

### ▶ AWS アカウントへのログイン

<https://console.aws.amazon.com/console/home>

### 初めてのログインはルートユーザーで行う



メールアドレスとパスワードを入力してログインを実行し、AWS マネジメントコンソールへとアクセスできれば初めてのログインは成功です。ここから AWS サービスへアクセスできます。

## ログインすると AWS マネジメントコンソールが表示される



先ほど、ルートユーザーは特別なユーザーであると述べました。通常 AWS アカウント内では、IAM というサービスで作成される IAM ユーザーごとに適切な権限を付与してサービスを利用します（7-2 節（p.209）で紹介）。

ルートユーザーは IAM で付与可能なすべての権限に加えて、AWS アカウント自体の設定変更や解約、契約変更などを行うことができる権限を付与された最上級のユーザーとなります。その AWS アカウントの管理者として、「なんでもできる」ユーザーです。

万一路ユーザーの情報が漏えいした場合、**AWS アカウント上のすべての情報が閲覧可能となるうえ、すべての AWS サービスが自由に操作可能となります。**そのため、ルートユーザーの取り扱いには細心の注意を払わなくてはいけません。

基本的に最初のログイン以降の AWS サービス利用は IAM ユーザーで行い、ルートユーザーは使用しないことが推奨されています。ルートユーザーの使用は、AWS アカウント全体に関わる設定変更（たとえば、アカウント名の変更やサポートプランの変更）など、ルートユーザーでしかできない作業時のみとしましょう。

# 02 AWS サービスを操作する 3つの方法

**keyword** • AWS マネジメントコンソール • AWS CLI • AWS SDK

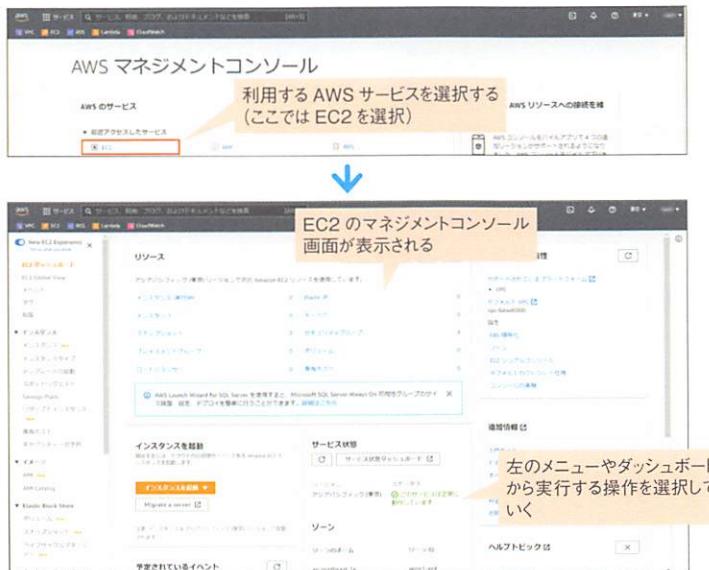
## Web ブラウザで AWS を操作する

AWS サービスの操作は、Web ブラウザをはじめ、さまざまな媒体から行うことができます。本節ではその代表的な例を取り上げます。

最も一般的かつ直感的な操作方法は、Web ブラウザで **AWS マネジメントコンソール** を利用することです。Web サイトを利用するような形で、マウスなどで直感的な AWS 操作が行えます。

AWS マネジメントコンソールは、前節で紹介したようにブラウザで AWS アカウントにログインすることで利用可能になります。

### ブラウザから AWS を操作できる AWS マネジメントコンソール



AWS サービスごとに  
コンソールがデザイン  
されており、一部の  
特殊な設定を除いた  
ほとんどの AWS 機能  
を操作できます。

## コマンドで AWS を操作する

マネジメントコンソールでの操作以外にも、AWS サービスは文字によるコマンドによって操作を行うことができます。コマンドでの操作には、**AWS Command Line Interface (AWS CLI)** のインストールが必要です。Linux、Windows、macOS など、利用する OS ごとにインストーラーやソースコードをダウンロードし、インストールを行います。

ここでは Linux、Windows でのインストール方法について簡単に紹介します。インストール後は、さらに AWS CLI を利用する環境に応じて個々のアクセスキーやリージョンの設定といった準備を行います。

### Linux でのインストール

以下のコマンドを実行します。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip" ————— awscliv2.zip をダウンロード
$ unzip awscliv2.zip ————— awscliv2.zip を展開
$ sudo ./aws/install ————— インストーラーを実行
```

### Windows でのインストール

以下の URL から AWS CLI MSI インストーラーをダウンロードして実行します。

#### ▶ AWS CLI MSI インストーラーのダウンロード

<https://awscli.amazonaws.com/AWSCLIV2.msi>

AWS CLI ではコマンドを実行することで、AWS 上のすべてのリソースに対して AWS 機能の操作を行うことができます。Linux ではコンソール上で、Windows や macOS ではコマンドラインツール上でコマンドを入力します。

次ページの図は新たにセキュリティグループを作成するための AWS CLI コマンドです。基本的にはどのような操作も同様のコマンド構成となっており、操作を実行する AWS サービスとその操作内容に加えて、その操作を実行するために必要な設定事項をコマンドのオプションとして記述します。

## セキュリティグループを作成するための AWS CLI コマンド

```
$ aws ec2 create-security-group --group-name sample-sg --description "Sample sg"
```

CLI コマンド 対象サービス サービスへの操作 操作での設定事項① 操作での設定事項②

このコマンドの場合、「EC2」サービスで「セキュリティグループの名前」と「セキュリティグループの説明」を設定し、「セキュリティグループの作成」を行う、という意味となります。

上で紹介したコマンドはあくまでも一例です。対象の AWS サービスごとに実行できる操作や設定できるオプションがまったく違うことも珍しくありません。AWS CLI を利用する際には、AWS CLI の公式コマンドリファレンスを参照することをおすすめします。

### ▶ AWS CLI の公式コマンドリファレンス

<https://docs.aws.amazon.com/cli/latest/reference/>

マネジメントコンソールでの直感的な操作と比べて少々難易度の高いコマンドでの AWS 操作ですが、この操作方法の利点としては**再現性の高さ**が挙げられます。

ブラウザでの操作はわかりやすい半面、マネジメントコンソールの UI に依存しているともいえます。ボタンの押し間違いなどの簡単な操作ミスや、UI のアップデートにより画面構成や文言が変更されると以前と同様の手順で実行できなくなるなど、特に複数人で開発を行う場合にはこのような問題がネックになってきます。

その点、AWS CLI のコマンド操作であれば、一度コマンドを作成してしまえば、コピー & ペーストでいつでも誰でも同じ結果が期待できるため、AWS 操作の手順化、スクリプト化を簡単にを行うことができ、操作ミスの削減も見込めるようになります。行いたい操作や、スキルの習熟度を考慮し、必要に応じてコンソール操作とコマンド操作を使い分けていくことが大切です。

## プログラムで AWS を操作する

AWS サービスはプログラムからも直接操作を行うことができます。プログラムで AWS を操作するためには、開発言語ごとに **AWS SDK** という開発キットをインストールします。

SDK とは、Software Development Kit の略で、ソフトウェアやサービスを開発する際に特定の機能に必要なプログラムやサンプルコードをまとめたものです。

### AWS SDK を利用すると、自前のプログラムから AWS を操作できる



AWS SDK は現在、以下の 8 種類の開発言語に対応しています。

- JavaScript
- Ruby
- Python
- Java
- PHP
- Go
- .NET
- C++

上記に加えて、React や Angular などのライブラリ・フレームワーク向け、Android や iOS などのモバイル OS 向けの SDK をはじめとして、IoT デバイス向け SDK など、さまざまな開発環境向けの SDK が提供されています。

SDK の主な用途としては、前述のような AWS リソースの操作はもちろん、アプリケーションの一部として AWS リソースを組み込むことや、ストレージサービスへのデータ保存、データベースサービスへのデータ入出力などが挙げられます。

SDK を活用することでアプリケーションと AWS サービスの直接的な接続を実現でき、より複雑で柔軟な AWS との連携を行うことが可能となります。

## 03

## AWS サービスの利用料を可視化する

**keyword** • AWS Billing and Cost Management

## AWS ではコスト管理が超重要

AWS を利用するうえで、必ず考慮しなくてはならないことのひとつが AWS サービスの利用料金です。AWS サービスの料金体系はサービスごとに違いますが、すべてに共通するのは、**従量課金制**であるということです。定額制のサービスとは違い、使えば使った分だけその使用量に応じて、毎月の利用料が請求されます。

当然、システムが巨大化、複雑化し、多くの AWS サービスを利用すればするほど、全体の利用料の管理は困難となります。また、あなたの AWS アカウントが不正に利用され、多額の利用料が請求されるといった可能性もゼロではありません。

本節ではそのような問題を未然に防ぐため、**AWS Billing and Cost Management** を利用したコスト管理の概要を紹介します。

## AWS Billing and Cost Management とは

AWS Billing and Cost Management では、請求・コスト管理のためのいくつかの機能が提供されており、これらを利用してすることで、サービスの使用状況やコスト情報の可視化を行うことができます。



AWS Billing and Cost Management を利用するには、マネジメントコンソールでユーザー アカウントのプルダウンメニューから「請求ダッシュボード」にアクセスします。

ダッシュボード画面のトップには、直近の利用料の概要やサービス別の利用料内訳などがまとめられており、そのアカウントでのサービス利用状況を簡単に把握することができます。また、メニューバーから、より詳細な請求情報やコスト管理機能にアクセスできます。

提供されている主な機能は次表のとおりです。請求情報や利用状況レポートは「Billing」で、コスト状況は「Cost Management」の機能で確認と管理ができます。

## AWS Billing and Cost Management が提供している主な機能

カテゴリ	機能	説明
Billing	請求書	<ul style="list-style-type: none"> <li>各月のアカウント全体の請求、請求のサービスごとの内訳を確認できる</li> <li>請求情報はCSVファイルとしてダウンロードできる</li> </ul>
	Cost & Usage Reports	<ul style="list-style-type: none"> <li>アカウントの請求情報や使用金額をはじめとして、より詳細な製品属性、料金属性などをまとめたレポートを自動作成できる</li> <li>AWSサービスごとの使用状況をレポートとして作成可能</li> </ul>
Cost Management	Cost Explorer	<ul style="list-style-type: none"> <li>サービスごとにコストとリソース使用状況をグラフによって可視化し、分析・予測ができる</li> </ul>
	Budgets	<ul style="list-style-type: none"> <li>利用サービスのリソース状況、コストを監視し、設定した予算を超えた場合アラート発報を行うことができる</li> <li>設定予算ごとの状況確認、管理ができる</li> </ul>

## コスト管理の方針

コスト管理をするうえで押さえておきたいのは、以下の 2 点です。

- Cost Explorer で正しく現状を把握し、方針を立てること
- Budgets で予算の設定を行い、利用状況を監視すること

Cost Explorer は AWS 全体のコスト状況の把握に役立ちます。サービスごとにコストを確認することで、どこにお金がかかっているか実態が明確になるため、サービスコスト削減を含めコスト最適化を図ることができます。

また、Budgets で予算設定を行うことで、利用料が設定した金額を超えた際に検知し、通知を受け取ることができます。このように、予測できないコスト増加への対策を行うことも重要となります。

## 04

# AWS のシステムが構築される場所を選ぶ

**keyword** • リージョン • 東京リージョン • 大阪リージョン • アベイラビリティゾーン  
• AZ 名 • AZ ID

## リージョンは世界中に存在する地域

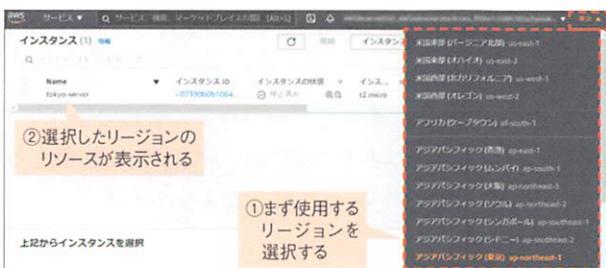
1-3 節 (p.18) の「AWS を理解する 6 つのポイント」でも紹介したとおり、AWS では世界中の各地域にリージョンと呼ばれる地理的に離れた領域が存在し、各リージョン間が AWS のプライベートネットワークで接続されています。たとえば、日本には東京リージョン「ap-northeast-1」と大阪リージョン「ap-northeast-3」という 2 つのリージョンが存在します。また、米国連邦政府、州、各地方自治体、契約業者が使用できる GovCloud (米国) リージョン ("us-gov-west-1", "us-gov-east-1") という特別なリージョンもあります。

世界各地にリージョンがあることにより、次のようなメリットがあります。

- 世界各地のユーザーに使用してもらうグローバルなシステム構築ができ、利用者からの通信遅延を削減できる
- 法的要件で特定の国にシステムを構築しなければいけない場合、特定のリージョンを使用することで要件を満たせる
- 大災害などで特定のリージョンでシステムが使用できなくなった場合、他のリージョンでシステムを稼働できる

AWS をマネジメントコンソール (画面) から利用する場合、まず画面の右上から使用するリージョンを選択します。たとえば東京リージョンを選択して EC2 のインスタンス (サーバー) 一覧を表示した場合、東京リージョン内の EC2 のみが表示されます。つまり、リージョンごとに分離してリソースを管理できます。

## マネジメントコンソールから選ぶだけで各リージョンを使用できる



### 東京リージョンと大阪リージョン

現在は普通に使用できる**大阪リージョン**ですが、通常リージョンとして開設されたのは、2021年3月と、最近の出来事になります。通常リージョンになる前は、**ローカルリージョン**として、一部機能が制限された状態で利用可能でした。東京リージョンが開設されたのが2011年3月のため、**そのちょうど10年後**に大阪リージョンが開設されることになります。国内で複数リージョンが使用できるのは米国と日本のみのため、日本国内のAWS需要の高さがわかります。

日本国内でも、システムやデータは国内に配置したい、ただし、関東地域の大災害には備えておきたいという声も大きくありました。東京リージョンと大阪リージョンは地理的に離れているため、大災害時にも日本国内で業務継続が可能なシステムが構築可能となりました。

なお、現状では東京リージョンのほうが利用できるサービス、機能が多いです。大阪リージョンも続々と機能拡充がされていますが、現時点での日本国内にシステムを構築する場合、まず東京リージョンをメインとして選択するほうがよいでしょう。

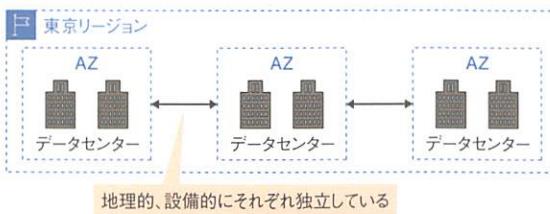
### 国内の2つのリージョンで災害時に備えたシステム構築が可能



## アベイラビリティゾーンはリージョン内の独立した場所

各リージョンに複数存在するアベイラビリティゾーン（以下AZ）は、1つ以上のデータセンターから構成され、それぞれ独立しています。距離的に離れている点（100km以内）も独立していますが、電力源、ネットワークなどの設備面もAZごとに独立して機能しています。つまり、あるAZでの障害は、別のAZに影響を与えないことを意味します。それぞれ独立しつつも、AZ間は高速なネットワークで接続され、通信は暗号化されています。

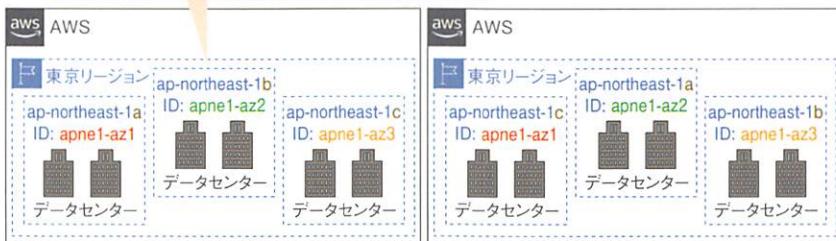
## リージョン内には複数のアベイラビリティゾーン（AZ）がある



AZには、AZ名（名前）とAZ IDが存在します。AZ名は、「ap-northeast-1a」のように、リージョン名+アルファベットの形式となっています。一方、AZ IDは「apn1-az1」のような形式となっています。AZ名とAZ IDは1対1で対応していますが、AWSアカウントごとにAZ名に対応するAZ IDは異なります。AWSリソースが各AZに分散するよう、このような構成となっています。そのため、どこかのAZで障害があった場合は、AZ名ではなくAZ IDを気にする必要があります。

## AZ IDが同じ=地理的に同じ場所のAZ。AZ名はユーザーごとに違う場所のことがある

AWSアカウントごとに、AZ名に紐付くIDは異なる。  
同じID=同じ場所のAZのため、障害時はIDを確認する。  
利用時(リソース作成時)はAZ名で指定する



AWSで何かサービスを使用する場合、基本的にはまずリージョンを選択する<sup>\*</sup>ことになりますが、AZはサービスの中で選択することになります。

\* IAMなど、グローバルサービスと呼ばれる、リージョンを選択しないサービスもあります。

たとえばネットワークサービスのAmazon VPCでサブネットを作成する場合、次図のように作成時にAZを選択します。

## 自システム用のネットワーク作成時にAZを選択する例



AWSでは複数のAZにリソースを配置してシステムの可用性を高めることも多いため、利用者が複数のAZを指定してリソースを作成することが多いです。また、ストレージサービスのAmazon S3では、自動的に3つ以上のAZにデータがコピーされ、利用者側でAZを意識することはありません。

本番稼働システムや重要なデータを保存する場合、基本的にはすべて複数のAZで稼働することを目指してAWSリソースを作成しましょう。前述のAmazon S3のようにAWS側で自動的に複数のAZが利用される場合もあるため、各サービスでリソースがどのように各AZに配置されるか理解することも大切です。

**column**

## システム開発の基本用語を押さえておこう

本書では AWS の利用方法を説明するうえで、システム開発でよく利用される用語を使っています。システム開発の経験がない方には馴染みがない言葉もあると思いますので、ここで簡単に説明します。

### ・インフラ

インフラストラクチャ (infrastructure) の略で、情報システムの動作する基盤となるサーバーやストレージなどの機器、ネットワークケーブルといったハードウェアと OS、ミドルウェアといった一部のソフトウェアを指します。

AWSにおいてはサービスとして提供されない AWS 側の管理する部分を指しますが、システム開発においては実処理を行うアプリケーションの対義語として使われ、EC2 や RDS などの AWS サービスも含まれることができます。

### ・コード

正式には「ソースコード」と呼ばれる、プログラミング言語で書かれたプログラムの設計図です。HTML 文書や CloudFormation などで使われる JSON ファイルも、同様の意味でコードと呼ばれることがあります。

### ・ビルド

ソースコードをコンピューターが実行できる形式に変換することを指します。プログラミング言語によってはビルドを必要とせず、ソースコードをコンピューターがそのまま実行できるものもあります。

### ・デプロイ

アプリケーションを対象のコンピューターに配置し、実行できるように準備することです。

### ・デフォルト

「初期設定」を意味します。「デフォルトで〇〇の動作をする」という文章は、「何も設定をしなければ〇〇の動作をする」と読み換えることができます。

### ・スケール、スケーリング

コンピュータリソースの量を変更することを指します。単体のコンピューターのリソースを増減させることをスケールアップ、スケールダウンと呼び、複数台で動作しているシステムのコンピューターの台数を増減させることをスケールアウト、スケールインと呼びます。

## Chapter 3

# コンピューティング サービス

---

この章ではデータ処理を行うサービスについて紹介します。代表的な EC2 をはじめ、Lambda、ECS といった近年注目が集まるサービス、コンテナについても解説します。

---

- section 01** サーバーとは  
押さえておきたいサーバーの基礎知識
- section 02** Amazon EC2 (Elastic Compute Cloud) とは  
EC2 で仮想サーバーを手軽に作成できる
- section 03** Amazon EC2 の外部公開とアクセス制御  
仮想サーバーを安全に外部に公開する
- section 04** Amazon EC2 Auto Scaling とは  
負荷に応じてサーバーを自動的に追加・削除する
- section 05** AWS Lambda とは  
サーバーを持たずにプログラムを実行する
- section 06** AWS Lambda の実用的な使い方  
サーバーレスの活用法をもっと理解する
- section 07** コンテナとは  
コンテナの仕組みと特徴を理解する
- section 08** Amazon ECS とは  
ECS でコンテナのメリットを享受する
- section 09** その他のコンピューティングサービス  
サーバー知識なしで使える代表的なサービス 5 選

# 01 押さえておきたい サーバーの基礎知識

**keyword** • サーバー • クライアント • Web サーバー • DB サーバー • メールサーバー  
• Linux • Windows Server • サーバーの仮想化

## サーバーとは

AWS にかかわらず、どのようなシステムでも欠かすことのできないのが**システム処理を実行するコンピューター**です。AWS では、仮想サーバーサービスである EC2 やコンテナサービスの ECS など、幅広いコンピューティングサービスが提供されています。

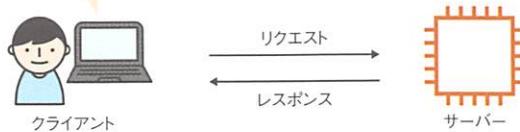
ネットワーク上でデータやサービスの提供を行うコンピューターを**サーバー**、そのサービスを利用するプログラムを**クライアント**といいます。代表的な例は Web サイトと Web ブラウザの関係です。この例だと、Web サイトを提供しているコンピューターが Web サーバー、Web ブラウザがクライアントとなります。

クライアントは、サーバーに対して特定のデータを送信するようにリクエストを行い、サーバーはそのリクエストに応じたレスポンスを返します。Web サイトであれば、特定のページの URL を指定してリクエストを行い、Web サーバーはその URL に対応したデータをレスポンスとして返します。

上記では Web サイトを例としましたが、それに限らずオンラインゲームや SNS、メールなどでも、やりとりする通信規約（プロトコル）やデータは違うものの、それぞれサーバーが存在し、同様の処理が行われています。

## ネットワークを介して何らかのサービスを提供するのがサーバー

クライアントはプログラム。  
人が操作しなくとも動作するものもある



## 代表的なサーバーの種類

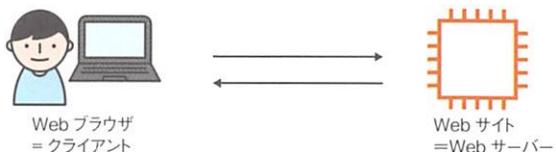
サーバーは、提供するサービスによってさまざまな種類があります。ここでは、代表的なサーバーの種類を紹介していきます。

### Web サーバー

前述したように、サーバーの代表的な例としては **Web サーバー** が挙げられます。Web サーバー上には Web ページ自体の構造を作成する HTML ファイルやデザインを定義する CSS ファイル、サイトに表示される画像ファイルなど、サービス提供に必要なデータが保存されています。Web サイトの構成に必要となるデータの保存と、システムの制御を行うプログラム群を導入したサーバーが Web サーバーと呼ばれます。

AWS 上で Web サーバーを構築する場合は、主に EC2 や ECS で実現することが多くなるでしょう。Web サーバーの詳細な仕組みは後述します。

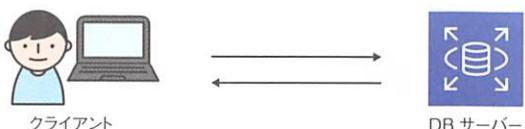
#### Web サーバーは Web ページの表示に必要なデータを提供する



### DB サーバー

システムが取り扱うデータを一元管理する、データベース管理システムが導入されているサーバーのことを、**DB ( DataBase、データベース ) サーバー** といいます。データベースにもさまざまな種類が存在しますが、データ保存や更新、バックアップなどの管理機能、データ検索などを行うデータ処理機能は共通しています。

#### DB サーバーはクライアントから要求されたデータ処理の結果を返す



DB サーバーは、クライアントから特定のデータについて、参照や書き換え、削除などのデータ処理要求をリクエストとして受け付け、それに対しての実行結果をレスポンスとして返します。

AWS では、RDS や DynamoDB などデータベースに特化したサービスが提供されています。EC2 にデータベース管理システムを導入して DB サーバーを構築することも可能ですが、RDS や DynamoDB は AWS 管理のマネージドサービスであったり、チューニングが容易であったり、大きな利点があるため、用途に応じてこれらの DB サービスの中から最適なものを選択して利用することを推奨します。

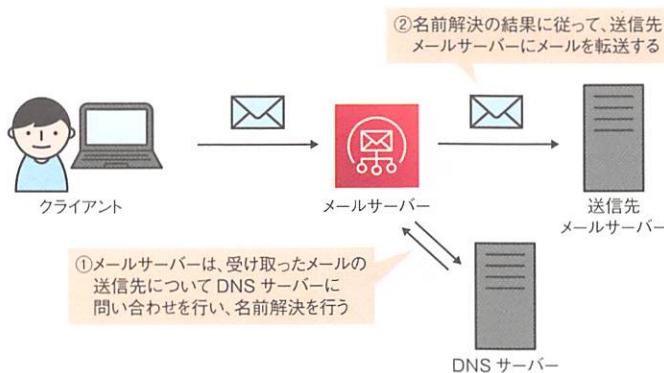
## メールサーバー

SMTP プロトコル、POP3 プロトコルを利用してメール送信や配達、受信を行うサーバーのことを **メールサーバー** といいます。一般的にメールサーバーは、その役割に応じて SMTP サーバーと POP3 サーバーに分類されますが、ここではシステムで利用されることが多い SMTP サーバーによるメール配信について説明します。

SMTP サーバーは、メール送信と配達を行うサーバーです。メール送信者はメールソフトなどを用いてメールを作成し、そのメールを SMTP サーバーに対してリクエストとして送信します。SMTP サーバーはメールを受け取った後、そのメールの宛先について DNS サーバーに問い合わせを実施して送信先を特定します。その後、その送信先にメール送信を行います。

AWS では、EC2 で自らメールサーバーを作成することも可能ですし、AWS マネージドのメール送信サービスである Amazon SES を利用することもできます。

### メールサーバー（SMTP サーバー）はメールの送信と配信を行う



## Web サーバーの仕組み

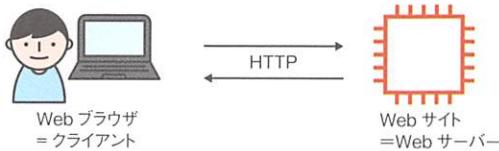
ここでは、Web サーバーの基本的な仕組みを説明していきます。

Web サーバーは利用者側の各 Web ブラウザをクライアント、Web サーバーをサーバーとして、クライアント - サーバー通信を行います。このとき、ブラウザと Web サーバー間では、HTTP や HTTPS というプロトコルを用いてさまざまなデータのやりとりが行われます。クライアントが Web サイトに必要なデータをサーバーに要求し、その要求に応じてサーバーがデータを返すのです。

この際に Web サーバーが扱うデータは多岐にわたりますが、次表に代表的な例を記載します。Web サイトのコンテンツ構成を定義する HTML を中心として、サイト自体のデザインや、そこで実行されるプログラムがサーバー上に存在します。

### Web サーバーが扱う主なデータ

カテゴリ	データ	説明
サイト構成	HTML	<ul style="list-style-type: none"> <li>Hyper Text Markup Language</li> <li>Webサイト自体の構造を定義するコード。段落やリストの構成、画像の使用、リンクの作成など、サイトの骨子となる構成が記述される</li> </ul>
サイトレイアウト・デザイン	CSS	<ul style="list-style-type: none"> <li>Cascading Style Sheets</li> <li>HTMLで定義されたコンテンツに対して、スタイル・デザインの適用やレイアウトの指定を行うことが可能</li> <li>具体的には表示される文字のフォントや色、サイズの指定から始まり、サイト上のアニメーション作成なども行える</li> </ul>
スクリプト (サーバーサイド)	PHP、Ruby など	<ul style="list-style-type: none"> <li>Webサーバー側で処理を行うプログラム。リクエストに応じて、処理結果をクライアントに返す</li> <li>必ずWebサーバーで処理が行われるため、結果がクライアントの環境に依存しない</li> </ul>
スクリプト (クライアントサイド)	JavaScript	<ul style="list-style-type: none"> <li>クライアント(ブラウザ)上のシステムで動作するプログラム</li> <li>クライアントで処理が行われるため、サーバーへの負荷が少ない</li> <li>クライアントの環境によって処理速度や結果が異なる場合がある</li> </ul>
画像	JPEG、GIF、PNG など	<ul style="list-style-type: none"> <li>サイト上で表示されるような画像も、サーバー上に保存されている</li> <li>データ圧縮の方式や機能によって、保存形式が選択される</li> </ul>



## サーバーの OS とは

サーバーをはじめとし、一般的なパソコンにも必ず搭載されているものがオペレーティングシステム（OS）です。OSは、人が機器の管理・制御を行うためのインターフェースやハードウェア管理機能、機器上で動作するソフトウェア群が共通して利用する基本的な機能などを実装したソフトウェアです。

一般的な家庭用パソコンでは主に Windows や macOS、スマートフォンでは Android や iOS が利用されていますが、サーバーとして利用される機器では主に Linux や Windows Server が利用されます。

### Linux

Linux は Windows や macOS と違い、無料かつオープンソースなソフトウェアなので、誰でも自由に開発、配布を行えます。そのため、大元の Linux カーネルと呼ばれるソフトウェアを基本とし、さまざまな企業や団体が独自に開発を加えた OS を Linux ディストリビューションとして提供しています。

次表は代表的な Linux ディストリビューションの一覧です。もし Linux を利用してサーバー構築を行うのであれば、これらのディストリビューションから目的に合ったものを選択するところから構築が始まります。

### 代表的な Linux ディストリビューション

ディストリビューション名	説明
Red Hat Enterprise Linux (RHEL)	<ul style="list-style-type: none"> <li>レッドハットが開発した商用向けのLinuxディストリビューション</li> <li>大規模システムなどのサーバーで多く利用される</li> <li>パッケージ管理システムとしてRPMを利用</li> </ul>
CentOS	<ul style="list-style-type: none"> <li>RHELのクローンOS</li> <li>RHELの商用部分を取り除いたLinuxディストリビューション</li> </ul>
Debian GNU/Linux	<ul style="list-style-type: none"> <li>Debianプロジェクトによって開発されているLinuxディストリビューション</li> <li>パッケージ管理システムとしてdebを利用</li> </ul>
Ubuntu Linux	<ul style="list-style-type: none"> <li>Debianをベースに作られたLinuxディストリビューション</li> <li>主に個人用途でのデスクトップ利用が多い</li> </ul>

上記の代表的なディストリビューションは AWS サービスの EC2 でも利用可能な OS として提供されており、自由に選択を行うことが可能です。

## Windows Server

Windows Server は、Windows を提供しているマイクロソフトからリリースされているサーバー用 OS です。Linux とは違い、利用には OS 自体のライセンスや CAL (Client Access License) と呼ばれるサーバー利用ライセンスの購入が必要となります。

Windows Server の大きな特徴のひとつに、GUI (Graphical User Interface) 操作であり一般的に利用されている Windows と UI や使い方が似ていることが挙げられます。基本的に CUI (Character User Interface) 操作となる Linux ディストリビューションと比べて利用しやすいといえます。

また、マイクロソフト製品であるため、その他のマイクロソフト製品との親和性の高さや AD (Active Directory) との連携、充実したサポートなど、さまざまなメリットがあります。

## OS ごとの比較

Linux と Windows Server の特徴について簡単にまとめると次表のようになります。どちらにも優れている点があるので、費用や利用したい機能を考慮して選択する必要があります。

### Linux と Windows Server の比較

項目	Linux	Windows Server
導入費用	基本的に無料 ※ディストリビューションによってはライセンス料あり	ライセンス料に加え、CAL (Client Access License) 費用が必要となり、高額
必要スペック	比較的低スペックでも動作可能	快適な利用にはそれなりのスペックが必要
インターフェイス	主にコマンドラインインターフェイスでの、コマンドのみによる操作	主にデスクトップなどのマウスで操作可能なインターフェイス
専門知識	コマンドによる操作や設定ファイルによる管理など、前提知識が必要	デスクトップのWindowsと似ているため、利用開始までに必要な知識は比較的少ない
サポート	基本的になし ※有償ディストリビューションはあり	マイクロソフトによるサポート体制あり
その他	ディストリビューションやパッケージを自由に選択でき、要望に沿った機能や特徴を実現できる	Active DirectoryやSQL Serverなど、マイクロソフトのソフトウェアとの親和性が高い

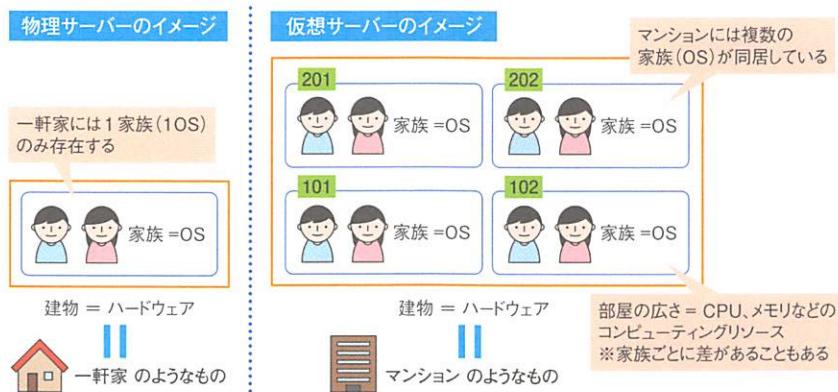
## サーバーの仮想化

通常、サーバーなどのハードウェア1台につき1つのOSが大前提となります。特定のソフトウェアを利用して**サーバーの仮想化**を行うことで、1つのハードウェア上で複数のOSを稼働させることができます。

サーバーの仮想化というと難しく聞こえるかもしれません。身边なものに例えると、**物理サーバーは一軒家、仮想サーバーはマンション**としてイメージしてみてください。この例では、建物=ハードウェア、そこに住む家庭=OSとして考えていきましょう。

一軒家の場合、その建物に住む家庭は基本的に1家庭のみですが、家の中のすべての部屋はその家族内で利用することになります。一方のマンションでは、建物内を1部屋ごとに区切り、その部屋ごとに1家庭が暮らすことができます。1つの建物の中に複数の家庭が暮らしているのです。

### 物理サーバーと仮想サーバー、それぞれのイメージ



仮想サーバーにおいて行われていることはこれと同様で、1つのハードウェア上で複数のOSを同居させることができます。ハードウェアの中で部屋を分割し、その部屋ごとにOSを稼働させるのです。ハードウェアのマンション化を行うのが、サーバーの仮想化です。

上図に記載しているように、仮想化を行ったサーバーではそれぞれの家族ごとに部屋の大きさ = CPUやメモリなどのコンピューティングリソースの規模を決める

必要があります。実際のサーバーでは、OSの種類や稼働させるソフトウェアによってリソースの割り当てを考える必要があります。

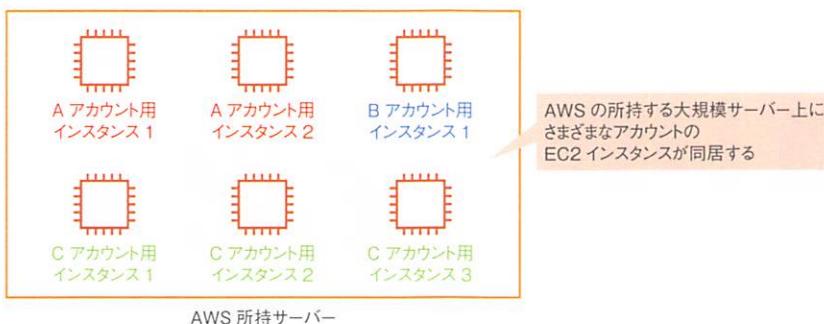
全体でどの程度の数の家族が住めるのかは、建物の大きさ（ハードウェアのリソース）次第で決まります。

## AWSにおける仮想化

AWSにおいても前述のような仮想化技術は多く利用されています。最も代表的なものは **Amazon Elastic Compute Cloud (Amazon EC2)** サービスです。詳細は次節から説明しますが、EC2ではAWSの大規模なサーバーでこの仮想化が行われています。ユーザーは自分の用途に応じてOSの種類やCPU、メモリの大きさ（=部屋の大きさ）を自由に選択して、**インスタンス**（仮想サーバー）を作成することができます。

ユーザーとしてEC2を利用する際に仮想サーバーであることを意識することはあまりありませんが、AWSを支える重要な技術となっています。

## AWSが提供する仮想サーバーサービスがEC2



## 02 EC2 で仮想サーバーを手軽に作成できる

- keyword**
- Amazon マシンイメージ (AMI)
  - インスタンスタイプ
  - AWS System Manager
  - EC2 の購入方法

### Amazon EC2 は仮想サーバーサービス

Amazon Elastic Compute Cloud (以下 EC2) は、仮想サーバーを数分で作成できるサービスです。Linux や Windows など、OS は AWS が用意しているものから選択します。オンプレミスではハードウェアや OS の準備から利用者が行いますが、EC2 では OS があらかじめインストールされた状態から利用を開始します。

### EC2 ではハードウェアや OS は用意された状態から始められる



CPU やメモリなどサーバーのスペックも利用者が自由に選択でき、作成後の変更（拡張）も可能です。データを保存するストレージの容量も利用者が指定し、こちらも容易に変更が可能です。変更や削除が容易に行えるため、**気軽に作成して試してみる**ことができます。複数の AZ に配置して可用性を確保する設定も利用者が行います。

EC2 では、仮想サーバーを **インスタンス** という単位で管理します。利用者はインスタンスタイプ（=仮想サーバーのスペック）を決定し、インスタンスタイプの単価と利用時間に対して従量課金で料金が発生します。サーバー停止中は料金は発生しません。

## 仮想サーバーを作成する

EC2 のインスタンス（仮想サーバー）は、最低限、次の設定値を選択すれば作成可能です。画面から各設定を選択するだけです。

- Amazon マシンイメージ (AMI)
- インスタンスのスペック (インスタンスタイプ)
- 配置するネットワーク
- データを保存するストレージの容量
- アクセス許可設定 (セキュリティグループ)

Amazon マシンイメージ (以下 AMI) は、OS やソフトウェアの設定が入ったテンプレートです。あらかじめ AWS が用意しており、利用者はそこから任意の AMI を選択します。

### あらかじめ用意されている AMI から使いたいものを選択



インスタンスタイプを選択することで、仮想サーバーの性能が決定します。CPU やメモリの容量が併せて表示されるため、必要なタイプを決定します。後から変更することも可能です。

## 仮想サーバーの性能を決める。後からでも変更可能

1 AMI の選択 2. インスタンスタイプの選択 3. インスタンスの設定 4. ストレージの追加 5. タグの追加 6. セキュリティグループの設定 7. 確認

**ステップ 2: インスタンスタイプの選択**

Amazon EC2では、異なるユースケースに合わせて最適化されたさまざまなインスタンスタイプが用意されています。インスタンスとは、アプリケーションを実行できる仮想サーバーです。インスタンスタイプはさまざまなCPU、メモリ、ストレージ、ネットワークキャパシティの組み合わせによって構成されているため、使用するアプリケーションに合わせて適切なリソースの組み合わせを選択できます。インスタンスタイプおよびそれをコンピューティングのニーズに適用する方法に関する詳細は[こちら](#)。

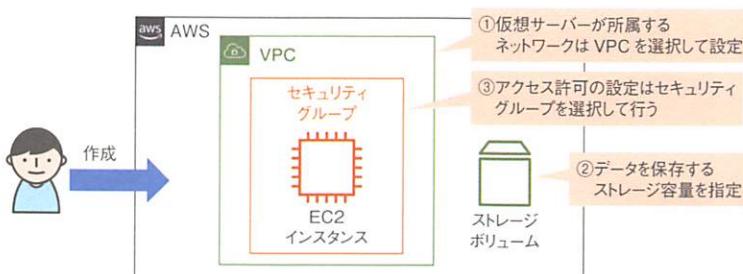
フィルター条件: **すべてのインスタンスマリナー** ▾ **現行世代** ▾ **列の表示/非表示**

現在選択中: t2.micro (- ECU, 1 vCPU, 2.5 GHz, - 1 GiB メモリ EBS のみ)

ファミリー	タイプ	vCPU	メモリ (GiB)	インスタンスストレージ (GB)	EBS 最適化利用	ネットワークパフォーマンス (Mbps)	IPv6 サポート
t2	t2.nano	1	0.5	EBS のみ	-	低から中	はい
<input checked="" type="checkbox"/> t2	<b>t2.micro</b>	<b>1</b>	<b>1</b>	<b>EBS のみ</b>	<b>-</b>	<b>低から中</b>	<b>はい</b>
t2	t2.small	1	2	EBS のみ	-	低から中	はい
t2	t2.medium	2	4	EBS のみ	-	低から中	はい
t2	t2.large	2	8	EBS のみ	-	低から中	はい

その他、配置するネットワークは利用者が作成した VPC を選択し、ストレージ容量 (EBS) を設定して、最後にアクセス許可設定であるセキュリティグループを選択します。VPC、EBS、セキュリティグループの詳細は後ほど説明します。ここでは仮想サーバーの作成に必要な設定をいくつか選択すると理解してください。

## 仮想サーバーを配置するネットワークやストレージ容量などを決める



すべての設定を選択して、インスタンスを作成すると、一覧に「実行中」状態で表示されます。

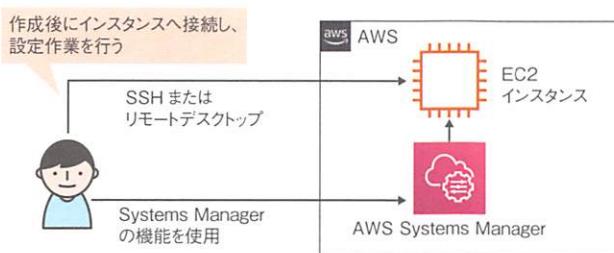
インスタンス (1/2) 情報

Q インスタンスをフィルタリング

Name	InstanceId	インスタンスの状態	インスタンスタイプ	ステータスチェック
<input checked="" type="checkbox"/> ec2-instance	i-0063172ce66...	実行中	t2.micro	○ このチェックに合格

インスタンスの作成後は、LinuxであればSSH、Windowsであればリモートデスクトップなど、管理系の機能を使用して接続できます。AWS Systems ManagerというEC2インスタンスを管理するサービスもあり、それを使用してサーバーに接続することも可能です。

## ネットワーク越しに仮想サーバー（インスタンス）に接続して操作



## インスタンスタイプでサーバー性能を決定する

インスタンスの作成時、仮想サーバーの性能はインスタンスタイプを選ぶことによって決定しました。インスタンスタイプのネーミングは次図のようになっています。

## EC2インスタンスタイプの名前のルール



ファミリー（インスタンスマルチ）には次表のようなものがあります。

## EC2 インスタンスのファミリー

ファミリー	特徴
T	バースト可能な汎用タイプ
M	バランスのとれた汎用タイプ
C	コンピューティング最適化 (vCPU数が多い)
R	メモリ最適化 (メモリ搭載量が多い)
P	高速コンピューティング
I	ストレージ最適化

T 系インスタンスについてはベースラインと呼ばれる決められた CPU 使用率が定義されており、それを超えて（バーストして）利用が可能です。ただし一定量を超えるとベースライン以下の性能になるか、追加課金が発生するため、本番環境など常時使用が見込まれる環境では M 系のインスタンスを選択したほうがよいでしょう。

オンプレミス環境では機器選定のため事前にマシンスペックを決定する必要がありますが、クラウド環境では作成後のスペック変更が容易です。そのため、構築やテストをしながらインスタンスタイプを決定していく、運用開始後も適宜見直すことをおすすめします。

## EC2 のさまざまな購入方法

1・3 節 (p.18) の「AWS を理解する 6 つのポイント」でも紹介したとおり、通常は利用時間とインスタンスタイプに応じて従量課金で料金が発生します。この通常利用タイプの EC2 インスタンスを **オンデマンドインスタンス** と呼びます。

1 年間または 3 年間分の料金を先払いして利用する **リザーブドインスタンス** というものもあります。オンデマンドインスタンスを 1 年間または 3 年間連続で利用するよりも、**最大 72% 割引** となります。この購入方法は、テーマパークの年間パスポートに近いです。毎日通うのであれば、1 年分の通常料金を支払って通うよりも、まとまった料金を前もって支払うことによってお得な購入ができます。

また、2019 年に **Savings Plans** という新しい購入方法も登場しています。これはリザーブドインスタンスと同様、先払い形式、割引料金で購入できる方法ですが、リザーブドインスタンスに比べて柔軟に幅広く適用が可能になっています。たとえ

ばりザーブドインスタンスでは、インスタンスマリーやサイズを指定する必要がありますが、Savings Plans は指定せずに購入が可能で、複数のスマリーやサイズに割り当てることが可能になっています。今後はザーブドインスタンスではなく Savings Plans を購入したほうがよいでしょう。

その他、ザーブドインスタンスよりもさらに安く購入できる方法として、**スポットインスタンス**があります。AWS 上で利用されていないリソースを活用してインスタンスを起動し、**最大 90% の割引価格**で使用できるオプションです。ただし、スポットインスタンスは予期せず AWS 側で停止される可能性があります（停止される場合は 2 分前に通知が行われます）。開発環境や複数台稼働している場合など、停止が許容される場合に有効活用できます。先ほどのテーマパークで例えるならば、1,000 人定員のところに 990 人来場していて、残り 10 名分を大安売りしているイメージです。また、他の顧客が来たら追い出される場合もあります。

## EC2 の利用用途によっては割引購入できる場合もある

購入タイプ	特徴	利用用途
オンデマンド	通常の購入方法。利用状況に応じて料金が変動	利用状況に濃淡があり、使用予定がわからない場合
ザーブド、 Savings Plans	1年間または3年間分を先払い。 オンデマンド比で最大72%割引	本番稼働など、1年以上稼働が続く予定がある場合
スポット	空き部分を使用。予期せぬ停止あり。 オンデマンド比で最大90%割引	開発環境や複数台稼働など、中断が可能な場合

### 他の EC2 の料金

インスタンス料金以外にも、EC2 では **ネットワーク外部へのデータ通信、データを保存するストレージ** (EBS、詳細は 4-5 節 (p.113) で紹介) に対して料金が発生します。

インスタンス料金に比べて安価になる場合が多いですが、多くのデータを保存して外部に通信する場合は気にする必要があります。ここでは詳細は紹介しきれないため、具体的な料金の計算方法が気になる方は AWS の料金ページで確認してみてください。

#### ▶ Amazon EC2 オンデマンド料金

<https://aws.amazon.com/jp/ec2/pricing/on-demand/>

# 03 仮想サーバーを安全に外部に公開する

**keyword** • パブリックサブネット • パブリック IP アドレス • セキュリティグループ

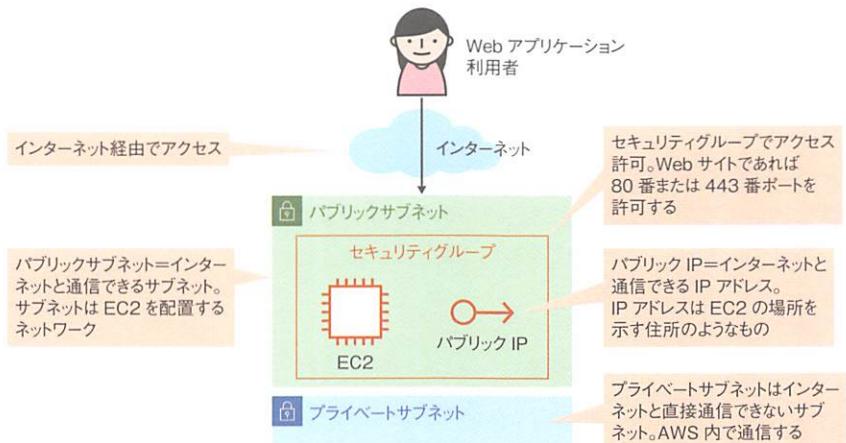
## サーバーの外部公開

EC2 上で構築した Web アプリケーションを公開する場合、いくつか追加で設定が必要となります。公開すると、みなさんが使用しているパソコンやスマートフォンからインターネット経由で Web アプリケーションを確認できるようになります。EC2 とインターネットの通り道を設定してあげる必要があります。なお、本節ではネットワークの用語がいくつか出てきます。出てくる用語がまったくわからないという方は、5-1 節 (p.122) で解説するネットワークの基礎知識を先に読んでください。

EC2 をインターネットに公開するには、次の 3 つの条件を満たす必要があります。

- EC2 をパブリックサブネットに配置する
- パブリック IP アドレスを EC2 に付与する
- セキュリティグループで外部からのアクセスを許可する

## EC2 をインターネットに公開するために必要なこと



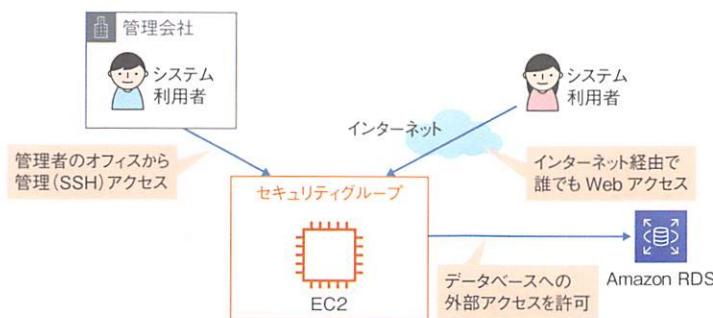
サブネットとパブリック IP は 5-1 節 (p.122) で詳細を紹介します。ここではインターネットと通信するためのネットワークの設定と理解してください。セキュリティグループはアクセス許可の設定です。全世界に公開する場合は、インターネットから EC2 へのアクセスをポート番号指定 (Web サイトであれば 443 番ポートまたは 80 番ポート) で許可します。

## サーバーへのアクセス制御

どこからどこに対してアクセスを許可するといった、EC2 のアクセス制御にはセキュリティグループを使用します。オンプレミスではファイアウォールという機器を使用してシステムへのアクセス制御を行いますが、セキュリティグループは仮想ファイアウォール機能です。

EC2 への通信をインバウンドルール、EC2 から外部への通信をアウトバウンドルールという形式で指定します。どちらも通信を許可するネットワークとポート番号を指定します。ポート番号は使用する通信の種類で決まり、たとえば Linux を操作する SSH は 22 番ポート、Web アクセスを行う HTTP は 80 番ポートとなります。

## EC2 にアクセスしてよい送信元や通信の種類を設定する



以下はインバウンドルールの設定画面および例です。

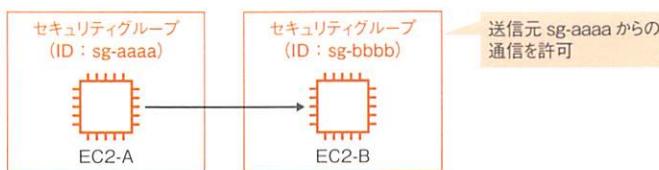
インバウンドルール					
ソース=送信元。 0.0.0.0/0はすべてのネットワーク (インターネット全体)を意味する				インバウンドルールを編集	
タイプ	プロトコル	ポート範囲	ソース	説明 - オプション	
HTTP	TCP	80	0.0.0.0/0	10.0.0.0/16は内部ネットワーク (プライベートなネットワーク)を意味する	
SSH	TCP	22	10.0.0.0/16		

次図はアウトバウンドルールの設定画面で、初期状態では外部へのすべての通信が許可されています。

アウトバウンドルール				
タイプ	プロトコル	ポート範囲	送信先	説明 - オプション
すべてのトラフィック	すべて	すべて	0.0.0.0/0	-

セキュリティグループは ID を持っております。送信元（ソース）や送信先にこの ID を指定できます。たとえば、EC2-A と EC2-B があり、A から B への通信を許可したい場合、B のセキュリティグループのインバウンドルールで送信元に A のセキュリティグループの ID を指定できます。

### あるセキュリティグループからの通信を許可するといった指定も可能



実際に ID を設定すると次図のように表示されます。

インバウンドルール (3)					
タイプ	プロトコル	ポート範囲	ソース	説明	
HTTP	TCP	80	0.0.0.0/0	-	
HTTP	TCP	80	59-0727b9f7c4eaaf4d...	-	
SSH	TCP	22	10.0.0.0/16	-	

セキュリティグループで 1 点注意すべきなのは、設定できるルールは**許可のみ**という点です。特定のネットワークからの通信を**拒否する設定はできません**。セキュリティグループ以外にも、第 5 章で紹介するネットワーク ACL（アクセスコントロールリスト）やその他のアクセス制御機能があるため、拒否設定を行う場合はそちらを検討します。

なお、セキュリティグループは EC2 だけでなく、他のサービスでも使用されます。アクセス制御の基本で、誤った設定で外部公開するとセキュリティ事故にもつながる重要な機能です。ぜひ基本的な使い方は覚えておいてください。

# 04

## 負荷に応じてサーバーを自動的に追加・削除する

**keyword** • Amazon EC2 Auto Scaling

### サーバーの自動追加と削除

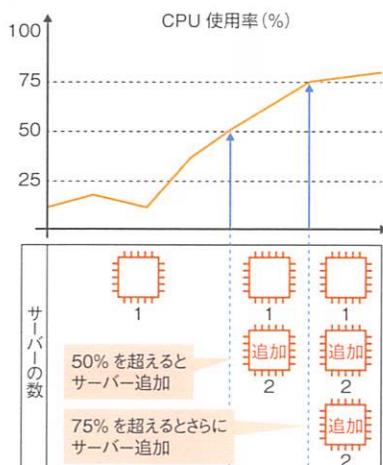
Amazon EC2 Auto Scaling (以下 Auto Scaling) という機能を使用すると、サーバーの複製追加と削除を負荷状況に応じて自動実行できます。サーバーを追加することをスケールアウト、削除することをスケールインと呼びます。

たとえばサーバー負荷の指標として CPU 使用率がありますが、その使用率に応じて次図のようなサーバー追加処理（スケールアウト）が可能です。追加するサーバーは起動テンプレートとして準備し、使用する Amazon マシンイメージ (AMI) の情報と起動に関する設定を登録しておきます。

### CPU 使用率などのルールに応じてサーバーを自動的に追加・削除

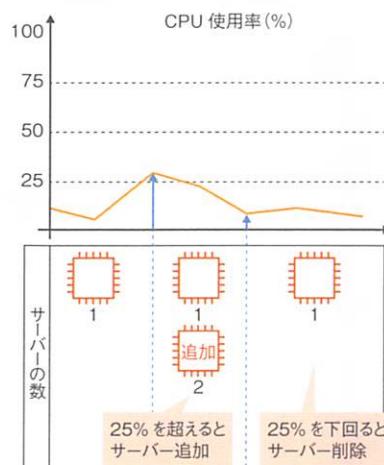
#### インスタンス追加ルール

- CPU 使用率 >50% ⇒ 1 台追加
- CPU 使用率 >75% ⇒ さらに 1 台(計 2 台)追加  
※削除のルールも設定可能



#### (CPU 使用率) 追跡ルール

- CPU 使用率が 25% 以下になるように調整



**ターゲット追跡スケーリング**という、指定した値に近づくようサーバー数を自動設定する機能もあります。Auto Scaling の利用自体は無料で、稼働している EC2 インスタンスの料金が発生します。

今回は CPU 使用率を例として紹介しましたが、ユーザーアクセス数など、アクセス状況に応じた自動追加と削除も設定可能です。

また、土日はアクセス数が少ないなど、システムの利用予定がわかっている場合は、**スケジュールスケーリング**という設定ができます。

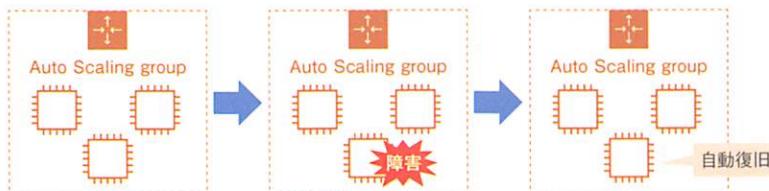
### スケジュールでサーバーを追加・削除することもできる

月～金	土・日	月～金
 1	 1	 1
 2		 2
 3		 3
.....		

平日はサーバー 3 台、  
土日はサーバー 1 台

Auto Scaling は障害に強いという利点もあります。指定された台数を維持するという機能のため、サーバーに障害があって停止した場合は、Auto Scaling の機能によりサーバーが自動で再作成されます。

### Auto Scaling は障害対策としても有効



その他、サーバー数の需要を予測する**予測スケーリング機能**や、インスタンス起動時のテンプレートとなる起動設定の**バージョン管理**など、多くの機能を持っていきます。AWS をパワフルに活用できる機能のひとつですので、ぜひこの機能は覚えておいてください。Auto Scaling の機能は EC2 以外のサービスでも存在します。

# 05 サーバーを持たずに プログラムを実行する

**keyword** • サーバーレスコンピューティング • AWS Lambda • 関数の実行

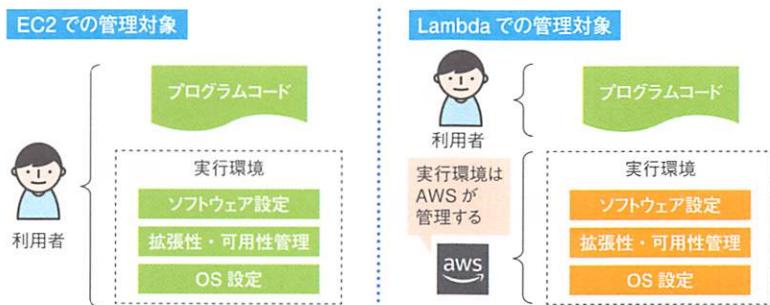
## AWS Lambda はサーバーレスコンピューティングサービス

AWS Lambda (以下 Lambda) はサーバーレスコンピューティングサービスです。OSなどのインフラストラクチャの管理が不要で、利用者はプログラムコードを準備し、Lambda にアップロードするだけで実行できます。

サーバーレスという言葉になっていますが、実際に稼働するサーバーが無いというわけではなく、AWS 側で実行基盤が管理されるため利用者の管理するサーバーが無いということになります。利用者はインフラ部分の管理を AWS 側に任せられるため、ビジネスロジックに関わるコード開発に集中できます。

EC2 と利用イメージを比較すると、次図のようになります。

## EC2 と Lambda の比較。Lambda は実行環境も AWS に任せる



2022 年 1 月現在、Lambda は次のプログラミング言語をサポートしています。

- Node.js\*
- Java
- Python
- PowerShell
- Ruby
- C#
- Go

\* Node.js は JavaScript の実行環境です。

これら以外の言語も、カスタムランタイムという機能を使用して利用可能です。また、2020年12月にはコンテナイメージのサポートも開始し、利用者が作成したコンテナイメージをLambdaへデプロイ（配置）できるようになっています。

Lambdaには次の利点があります。

#### ●セキュリティ

OSなどのインフラストラクチャのパッチ適用など、実行基盤のセキュリティはAWSで管理され、利用者は自分が書いたコードのみ責任を持って開発すればよいため、より簡単にセキュアな実行環境を実現できます。

#### ●コスト

Lambdaではコードが実行される時間1ミリ秒ごとに課金されます。EC2では、たとえ処理が行われていなくても起動している時間分料金が発生しますが、Lambdaでは実行していない時間は料金が発生しないため、コスト削減につながりやすいです。

#### ●可用性

AWSには複数の物理的に独立したアベイラビリティゾーン（AZ）がありますが、LambdaではAWS管理のもと複数のAZにわたって関数が実行されます。たとえば1回目はAZ-Aで実行され、2回目はAZ-Cで実行されるということが起こります。AZ障害時は実行可能なAZで実行されます。利用者が意識しなくても高可用性、耐障害性が保たれています。

#### ●拡張性

Lambdaは複数の処理を受けると、自動的にAWSが管理する処理用のインスタンスが立ち上がり、自動拡張されます。東京リージョンでは同時実行数が最大1,000となっており、この値は申請により引き上げることも可能です。

さて、Lambdaの利点だけ聞くと、EC2は使用せずにLambdaだけ利用したほうがよいと思うかもしれません。しかし実際には、すべてのケースでLambdaを使用することは難しいです。EC2はLambdaに比べて次のような利点を持ちます。

- オンプレミス上のアプリケーションをAWSへ移行する場合、EC2ではOS設定などをそのまま移行しやすい

## サーバーを持たずにプログラムを実行する

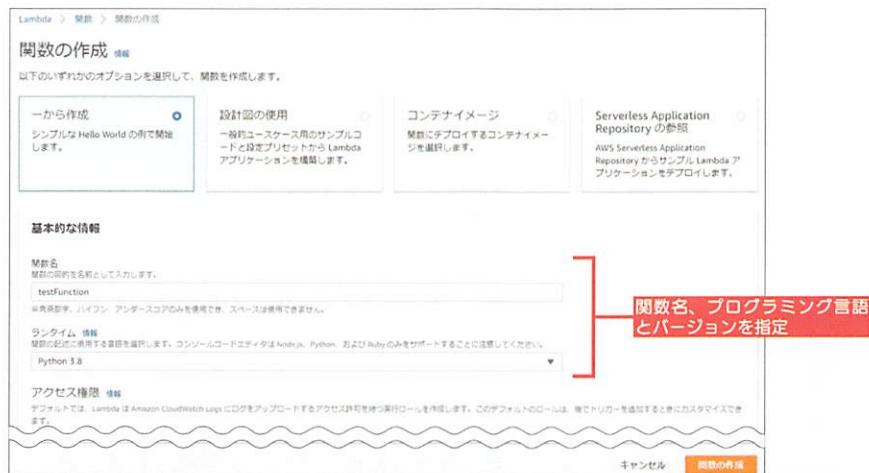
- EC2 はインスタンスタイプや OS、ネットワークなどを自由に設定できるという柔軟性がある
- 大量のトラフィックやアクセスを常時処理する場合、EC2 のほうが安くなる場合もある
- EC2 ではサーバーにプログラムをデプロイ（配置）するという従来の方式で開発を進められる。Lambda には AWS 独自の設定方法や開発方法があるため、初心者には難しい場合もある

稼働するアプリケーションの状況、開発する組織のスキルなど、場面に応じて適切なサービスを選択することが重要です。

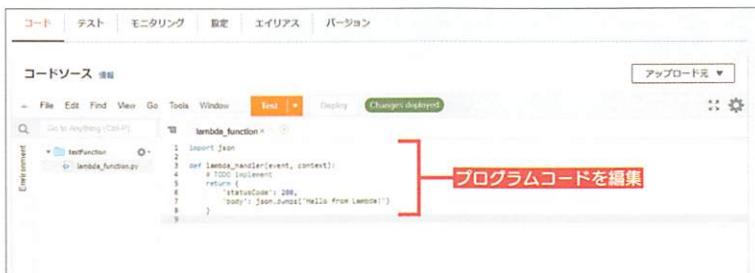
## 関数を作成して実行する

Lambda では、**関数**という単位でプログラムコードを管理し、処理も関数単位で実行します。

関数の作成は複数のツールから可能で、AWS マネジメントコンソールの画面上からも作成できます。「プログラムコードを書くだけで実行できる」というのは具体的にどういうことなのか、Python を例に関数の作成方法を見ていきます。



- 1 Lambda の画面から関数の作成を選ぶと、上の画面が表示されます。任意の関数名を指定して、使用したいランタイム（プログラミング言語とバージョン）を選択します。今回は Python 3.8 を選択します。



- ② 関数を作成すると、画面上にデフォルトのプログラムコードが表示され、そのまま画面上で編集できます。



- ③ 作成した関数は、画面上からテスト実行が可能です。②の画面上に表示されている [Test] ボタンを押下すると③の画面になるので、関数に渡すデータを JSON 形式で指定します。そのまま [作成] ボタンを押してテストイベントを作成します（今回実行するデフォルトのコードではこの JSON データの中身は使用しません）。



- ④ 再度コード編集画面に戻って [Test] ボタンを押下すると関数が実行され、実行結果が表示されます。今回はデフォルトコードのままで、「Hello from Lambda!」という文字列を含むデータを表示する関数を実行しています。

関数を画面上から作成し、プログラムコードを書くだけで実行できました。このように環境の準備をせず簡単に実行できる点がLambdaの魅力のひとつです。なお、画面上でコードの編集ができるのはPythonやNode.jsなど、コンパイルが必要な一部の言語のみとなります。

## column

### JSON とは

ショイソン

JSON (JavaScript Object Notation) は、データを表現するためのデータ記述言語のひとつです。JavaScriptのオブジェクト（データ）の表記方法から由来しており、このような名前になっていますが、JavaScriptに限らず多くのプログラミング言語やAWSの各サービスで利用されます。

JSONは{}の中にキーと値をコロンで区切って記述します。1つの{}の中に複数のキーと値を定義することも可能です。

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

{ }の中にさらに{ }を階層的に含めたり、[ ]を使用して配列の値を入れたりすることもできます。AWSでもよく使用されるデータ表現方法のため、基本的な書き方は覚えておくとよいでしょう。

## 06

サーバーレスの活用法を  
もっと理解する

**keyword** •Lambda の利用例 •Lambda の追加設定とモニタリング •Lambda の利用料金

## AWS Lambda は他のサービスと組み合わせて使用する

前節の関数のテストでは、利用者が指定した JSON 情報をインプットとして関数を手動実行しました。別的方法として、ユーザーのアクセスやデータ連携など、何かをきっかけに自動的に Lambda 関数を実行する場合は、他の AWS サービスと連携します。いくつか例を見ていきます。

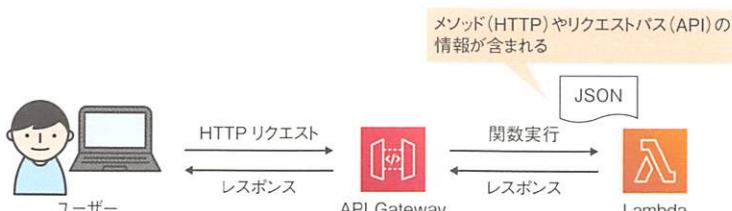
## ユーザーのリクエストに応じて Lambda 関数を実行する

Amazon API Gateway (以下 API Gateway) というサービスと Lambda を組み合わせることで、ユーザーの HTTP リクエストを契機 (トリガー) として Lambda 関数を実行できます。これにより簡単な Web アプリケーションが作成可能です。

API Gateway がユーザーから HTTP リクエストを受け取ると、API Gateway は GET などの HTTP メソッドや、/api などのリクエストパスの情報を含む JSON データを Lambda 関数に渡して実行します。Lambda 関数で応答内容を返すようコードを書いておくことで、API Gateway 経由で応答を返します。Web ブラウザであれば、Web ブラウザの画面上にレスポンス (応答) データが表示されます。

API Gateway もサーバーレスサービスのため、利用者はサーバー管理不要で Web アプリケーションを構築できます。

## API Gateway と組み合わせて Web アプリケーションを構築

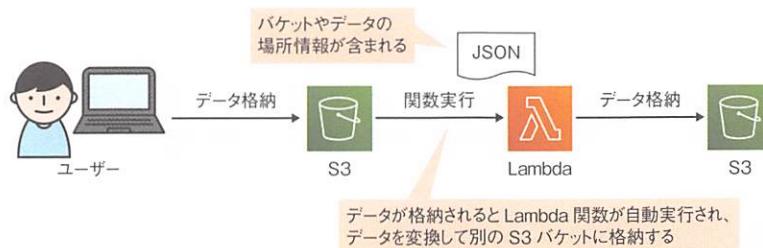


## S3へのデータ格納時に自動処理を行う

Amazon S3（以下S3）バケットにデータが格納されたら、格納されたデータに対してLambdaで自動処理を行うこともできます。たとえばデータに含まれる情報を加工して別のS3バケットに格納するといった処理が可能です。

この場合もS3バケットやデータ情報がJSONとしてLambda関数に渡されるため、関数のコードでその情報をもとにデータを処理します。

## S3と組み合わせて格納されたデータを自動的に加工



## 定期的にLambda関数を実行する

毎日12:00など、決まった時間に処理を実行したい場合は、Amazon EventBridge（以下EventBridge）というサービスとLambdaを組み合わせることで実現できます。EventBridge側にルールという形で日次実行する定義を設定しておき、ターゲットにLambda関数を指定することで定期実行が可能です。

たとえば毎日決まった時間にEC2を停止または起動するといった運用もこの構成で可能です。

## EventBridgeと組み合わせてプログラムを定期実行



ここでは3つの構成を例として紹介しましたが、実際には他のサービスも含むさまざまな構成パターンでLambdaは実行されます。代表的な構成例としてこういった使い方があると理解してください。

## AWS Lambda の追加設定とモニタリング

Lambda の実行環境は基本的に AWS 側で管理され、利用者の細かな設定作業は不要ですが、次のようなパラメータは利用者が設定します。

### ● メモリ容量

関数実行時に使用できるメモリ容量を指定します。デフォルト値は最小の 128MB で、最大 10,240MB まで指定可能です（リージョンにより最大値は一部異なります）。CPU の値は指定できず、指定したメモリ容量に比例して設定されます。

### ● タイムアウト時間

関数が実行できる最大時間であり、実行開始後この時間を経過しても実行されている場合は停止されます。デフォルト値は 3 秒で、1 ~ 900 秒の間で指定が可能です。

### ● 環境変数

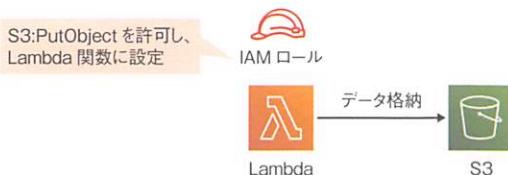
環境変数は、関数に書いたコードの外部で所持できるパラメータ値です。たとえば、開発環境と本番環境で接続するデータベース情報が異なる場合、環境変数に接続情報を保存することで、同じコードで複数環境に対応した関数の管理が可能です。



Lambda は S3 に対してデータを格納したり、EC2 の起動／停止をしたり、他の AWS リソースを操作する目的で使用されることが多いですが、最初から Lambda の関数がすべての AWS リソースに対する操作権限を持っているわけではありません。Lambda 関数に IAM (Identity and Access Management) ロールという形式で権限を付与して設定します。この IAM ロールは利用者が権限を設定して作成します（IAM ロールの詳細は 7-2 節（p.209）で紹介します）。

たとえば、S3 バケットにデータを格納する処理を行う場合は、S3 の PutObject 権限を持つ IAM ロールを作成して、Lambda 関数に設定します。

## Lambda から他の AWS リソースを操作するには権限の設定が必要



なお、前節で関数のテストを紹介した際に作成した Lambda 関数は、特に IAM ロールの作成操作はしていません。IAM ロールを指定しない場合は、Lambda デフォルトの IAM ロールが自動作成されて設定されます。自動で作成される IAM ロールには、Amazon CloudWatch (以下 CloudWatch) というモニタリングサービスにログを送信する権限のみ付与されています。

**不要な権限は設定せず、処理内容に対して必要最低限の権限を設定**することが重要です。

### Lambda 関数の実行状況のモニタリング

AWS のモニタリングサービスである CloudWatch に次のような情報が Lambda から自動で送信されるため、利用者はその状況を確認できます。Lambda の画面からグラフ情報としてそのまま確認可能です。

- 実行回数
- 実行時間（最小／最大／平均）
- エラー数と実行成功率

## CloudWatch で Lambda 関数の実行状況がモニタリングできる



実行回数のような数値情報（メトリクス）だけではなく、関数が送出するログも、CloudWatch のログに自動送信されます。利用者はここから関数実行内容の詳細を確認できます。

## Lambda 関数の実行内容の詳細もログで確認できる

ログイベント

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns ▾

View as text  アクション ▾ Create Metric Filter

イベントをフィルター Clear 1m 30m 1h 12h Custom

メッセージ

現時点では古いイベントはありません。再試行

```
START RequestId: 57407189-e907-4a63-aed1-0f8d7df1331f Version: $LATEST
this is test
END RequestId: 57407189-e907-4a63-aed1-0f8d7df1331f
REPORT RequestId: 57407189-e907-4a63-aed1-0f8d7df1331f Duration: 1.15 ms Billed Duration: 2 ms Memory Size: 128 MB Max
Memory Used: 51 MB Init Duration: 115.06 ms
```

追加設定なしでも CloudWatch を使用してこのようなモニタリングが可能ですが、AWS の他のサービスや外部のサービスを組み合わせて、より詳細なモニタリングを行うことも可能です。

## AWS Lambda の利用料金

Lambda の利用料金は、EC2 のようにインスタンスが稼働した時間に対して発生するという料金体系とは異なり、関数に対するリクエストの数とコードの実行時間に基づいて課金されます。東京リージョンでは次のような料金設定になっています。

### Lambda の料金設定（東京リージョンの場合）

項目	料金
リクエスト	リクエスト100万件あたり 0.20USD
実行時間	(メモリ) GB-秒あたり 0.0000166667USD

GB- 秒という単位が見慣れないかもしれません、これは**メモリ 1GB の関数が 1 秒間実行された場合の料金**ということです。メモリ割り当てが 512MB であれば実行時間あたりの料金は半額になりますし、実行時間が 1 ミリ秒であれば料金も 1,000 分の 1 になります。具体例を見ておきます。

例 「メモリ容量を 1GB に設定した Lambda 関数を、150 万回呼び出し、1 回あたりの平均実行時間が 1 秒の場合」

まずリクエストについては、100 万回の 1.5 倍（150 万回）になるので、0.20USD × 1.5 倍 = **0.30USD** となります。

実行時間は、 $0.0000166667\text{USD} \times 1\text{GB} \times 1\text{秒} \times 150\text{万回} = 25.00\text{USD}$  となります。

リクエスト料金 0.30USD と実行時間料金 25.00USD を合わせて、**合計 25.30USD** となります。

ただし、Lambda には次の**無料利用枠**があります。

- リクエスト 每月 100 万件は無料
- 実行時間 每月 40 万 GB- 秒は無料

これを先ほどの料金に適用すると、次のとおりになります。

- リクエスト

$$0.20\text{USD} \div 100\text{万回} \times (150\text{万回} - 100\text{万回} (\text{無料分})) = 0.10\text{USD}$$

- 実行時間

$0.0000166667\text{USD} \times (150\text{万GB-秒} - 40\text{万GB-秒(無料分)}) = 18.33\text{USD}$

2つを合わせて**合計 18.43USD**になります。

今回は 150 万回と呼び出し回数の多い例を紹介しましたが、検証用途で Lambda を軽く利用する程度では、料金はかかるないと考えてよいでしょう。ただし、プロビジョンドコンカレンシー (Provisioned Concurrency) という同時実行性能の設定を行った場合や、外部へのデータ転送は別途料金が発生するため注意してください。

# 07 コンテナとは

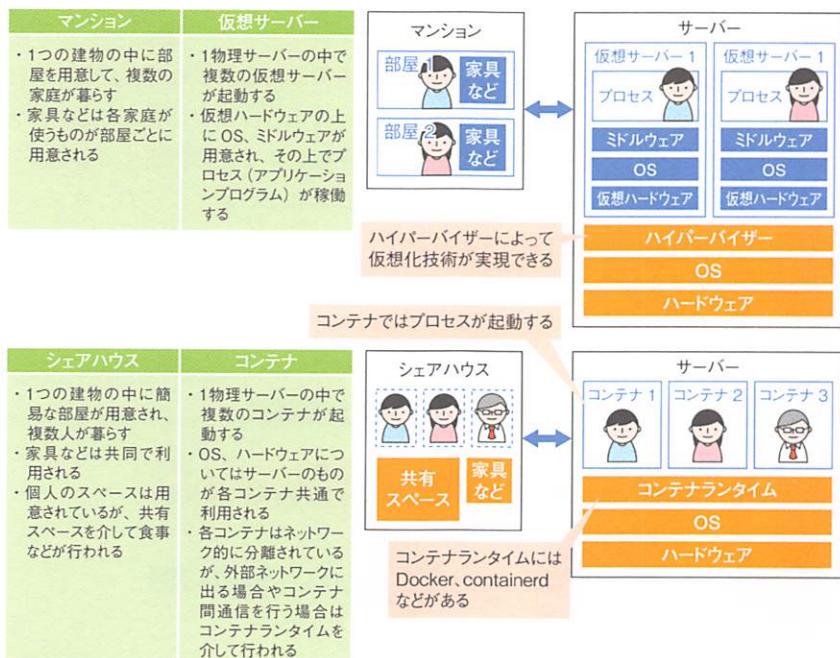
## コンテナの仕組みと特徴を理解する

**keyword** • コンテナ • コンテナイメージ

### コンテナとは

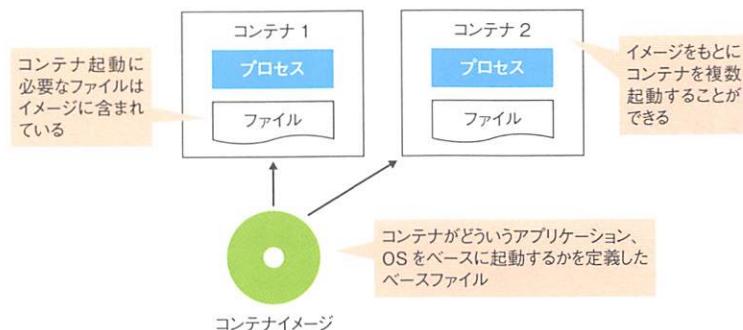
近年ではシステムのアーキテクチャの選択肢として、コンテナが挙げられるケースが増えています。コンテナはよくサーバーの仮想化と比較されることがあります。サーバーを家、サーバー上で稼働するアプリケーションプロセスを家に住んでいる住人に例えると、仮想化、コンテナはそれぞれ次図のようなイメージとなります。

### 仮想サーバーはマンション、コンテナはシェアハウスのようなもの



コンテナを起動する場合、どういったアプリケーションプロセスで起動するかをあらかじめ定義したベースファイルをもとにして起動します。これを、**コンテナイメージ**と呼びます。

## コンテナイメージからコンテナを複数起動できる



## コンテナは軽量で高速

コンテナの大きな特徴として、仮想化と比較して軽量で、より高速であるという点が挙げられます。

コンテナが軽量といわれる理由として、**コンテナ内に含まれるもののが少ない**ことがあります。仮想サーバーが起動する場合は、OS、ミドルウェア、アプリといったものが必要になります。そのため、仮想サーバーを起動する際に指定する仮想サーバーイメージにはこれらのファイル群がパッケージされています。一方、コンテナではアプリプロセスのみが起動します。コンテナイメージも基本的にはアプリを実行するためのファイル群のみがパッケージされるため、仮想サーバーイメージと比べて軽量になります。

## コンテナイメージは仮想サーバーイメージよりも軽量



高速であるという点に関しては、**起動が速い**ことがあります。仮想サーバーは一般的なサーバーと同様にOSが起動してからミドルウェア、アプリという風に順を追って起動していきます。それに対して、コンテナは直接アプリを起動するため、全体として起動が速くなります。また**処理速度についても、OSへのオーバーヘッドがない分、仮想サーバーよりも高速**となります。

## コンテナは起動も処理速度も仮想サーバーより高速

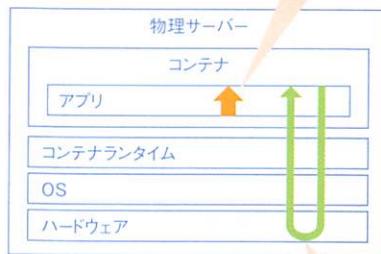
### 仮想サーバーの場合

OS→ミドルウェア→アプリという順番で起動していくため時間がかかる



### コンテナの場合

直接アプリが起動するため、仮想サーバーと比較して起動時間が短い



起動時間や処理時間が速い

ビルド・デプロイにかかる所要時間が短くなるため、効率的な開発が可能となる

## コンテナはデリバリーしやすい

**デリバリーに優れている**という点もコンテナのメリットといえます。ここでのデリバリーとは、「システム環境への反映のしやすさ」と捉えてください。

コンテナは、コンテナイメージを作成しておくと、どのサーバー環境においてもコンテナランタイムさえインストールしておけばコンテナイメージで定義したアプリケーションプロセスを実行できます。これは、ローカルのパソコンで起動していたコンテナをそのまま本番環境や開発環境などさまざまな環境でも同じように起動できるということです。これによってアプリケーションプロセスの再現性が高くなり、複数のシステム環境で動かす場合でも環境による差異を少なくできます。

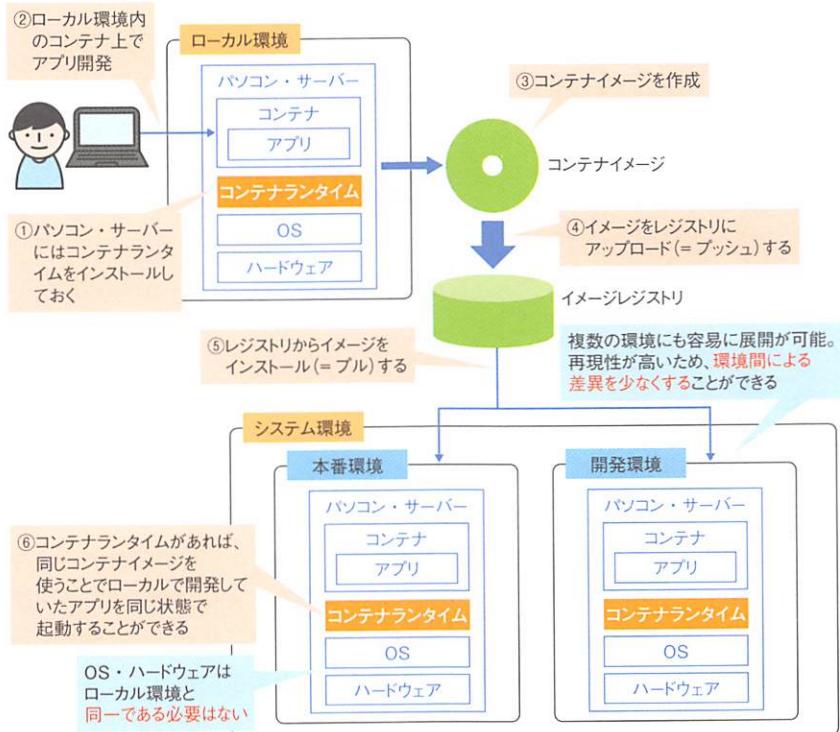
また、コンテナイメージはレジストリと呼ばれるイメージ管理ツールを用いて、イメージごとにレポジトリという単位で管理することができます。レジストリを経由することで、コンテナイメージを複数のシステム環境に対して容易に展開することができます。

このように起動が高速でデリバリーがしやすいことは、開発がスムーズになることがあります。結果的に、開発にかかる時間を短縮できることもメリットです。

一方、コンテナランタイムはサーバー側のOS上で起動するため、同じサーバー上で起動するコンテナは異なるOSを利用することはできません。また、コンテナは基本的に揮発性という性質を持ち、コンテナが削除されるとコンテナ内に保存されているデータも一緒に削除されてしまいます。

そのため、OSに依存したり、処理するデータを保持しなければならない設計になっているアプリケーションは、コンテナの利用には不向きです。

## コンテナイメージは複数のシステム環境に容易に展開できる



## コンテナオーケストレーション

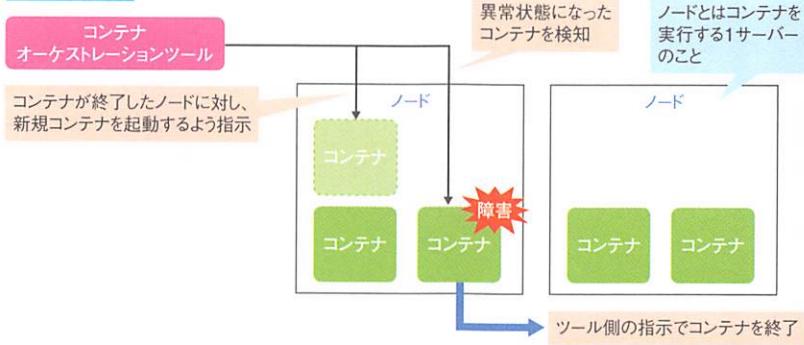
実際にコンテナを用いてシステムを構成するとき、多くの場合はさまざまなプロセスを複数のコンテナで稼働させることになります。この際にコンテナ1つずつを意識して運用しようとかなりの労力がかかります。

たとえばコンテナが異常終了した場合、そのたびに人間が毎回フォローをしていくと大変です。またシステム側への負荷が増えた場合に、コンテナを何台追加で起動するのか、そもそもどの程度負荷が増えた場合にコンテナを追加するのかといった点を毎回考えながら対応していると、運用のスピード感も出なくなります。

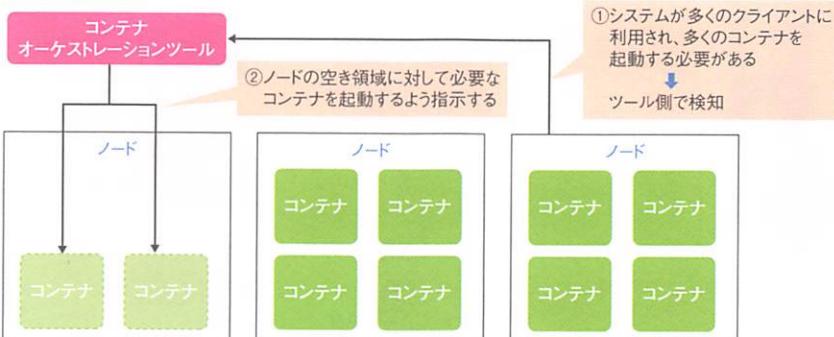
**コンテナオーケストレーションツール**ではこういった管理を自動化することで、運用負荷を軽減することができます。コンテナを用いたシステム開発において、欠かせない存在となっています。

### コンテナはコンテナオーケストレーションツールで管理する

#### 状態監視



#### スケーリング



# Amazon ECS とは

## 08 ECS でコンテナの メリットを享受する

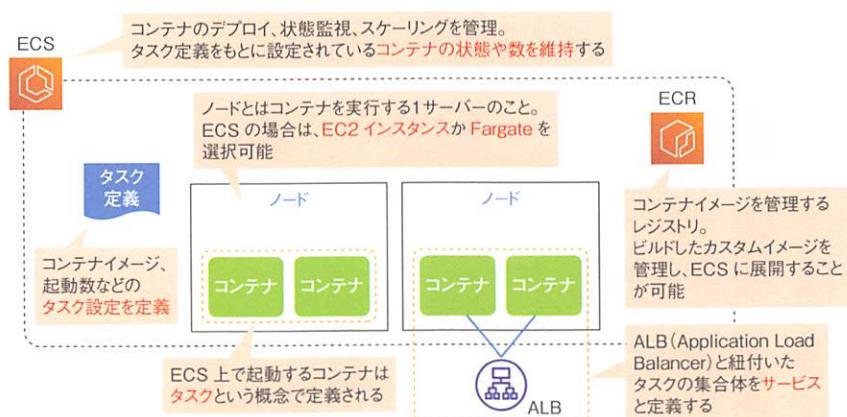
**keyword** • Amazon ECS • Amazon EKS • Amazon ECR • AWS Fargate

### AWS のコンテナオーケストレーションサービス

AWS 上でコンテナシステムを構築する場合には、一般的に **Amazon Elastic Container Service** (以下 ECS) と **Amazon Elastic Kubernetes Service** (以下 EKS) という 2 つのサービスが選択肢として挙げられます。この 2 つの大きな違いは、コンテナオーケストレーションの機能を AWS が担うのか、Kubernetes (後ほど説明) が担うのかです。最初に ECS について説明します。

ECS とは、AWS におけるマネージドなコンテナオーケストレーションサービスです。コンテナオーケストレーションツールはそれ自体の構築や管理に労力がかかりますが、この部分を AWS にすべて任せることができます。また、**Amazon Elastic Container Registry** (以下 ECR) というコンテナイメージレジストリサービスも提供されています。これを使用することで、カスタマイズしたコンテナイメージを AWS 上で管理することができ、ECS への展開も容易になります。

### ECS はマネージドなコンテナオーケストレーションサービス

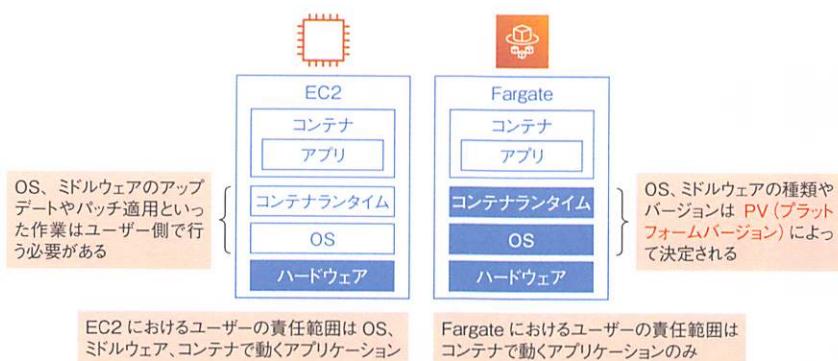


## EC2 と Fargate

ECS を利用してコンテナを起動する場合に、どのプラットフォームで起動するかを 2 種類から選択することができます。種類としては **EC2 と AWS Fargate (以下 Fargate)** があります。EC2 のほうは、EC2 インスタンス内で起動するコンテナランタイム上でコンテナを起動するタイプです。この場合、コンテナランタイムがインストールされているサーバー (=EC2 インスタンス) の管理をユーザー側で行う必要があります。

これに対して、Fargate は AWS 側が管理するサーバー上でコンテナを起動するタイプです。Fargate はコンテナランタイム・OS の部分も含めてサーバーの管理は AWS 側に任せられるため、ユーザー側はこの部分を意識する必要がありません。サーバー内のコンテナランタイム・OS などのバージョンは **プラットフォームバージョン (以下 PV)** で管理されており、Fargate を起動する際にはこの PV を指定する必要があります。PV はおおよそ 1 年単位で更新されており、この点についてはユーザー側でバージョン更新に追従していくという運用が必要になります。

### ECS でコンテナを起動する方法は 2 種類から選べる



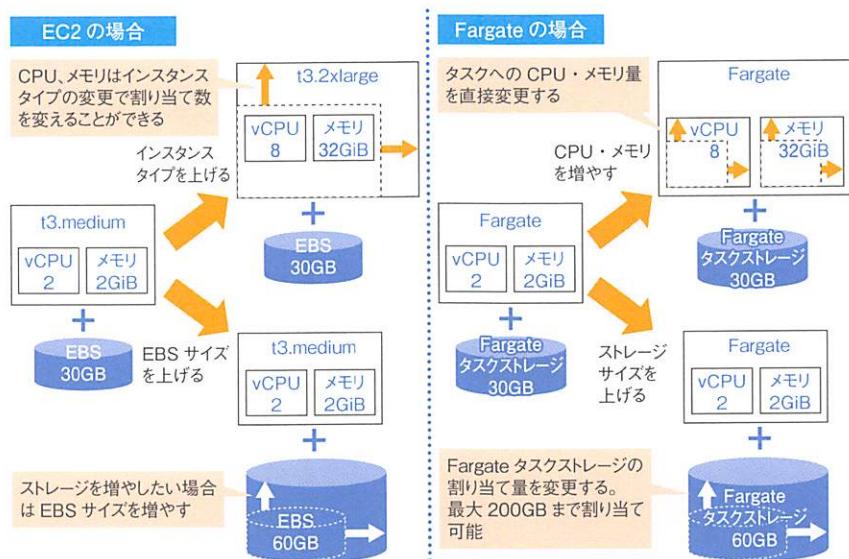
ノード（コンテナを実行するサーバー）側に割り当てるハードウェアのリソース（CPU・メモリ・ストレージ）の割り当て方についてもプラットフォーム種別で異なります。

EC2 の場合、CPU・メモリを増やしたい場合はインスタンスタイプを適切なものに変更し、ストレージを増やしたい場合はストレージ (EBS) のボリュームサイズ

を拡張することで対応できます（EBSについては4-5節（p.113）で説明します）。

これに対してFargateの場合、CPU・メモリに関しては直接割り当てる数量を指定できます。ただし、指定するCPU値に対して割り当て可能なメモリ量の範囲が定められています。ストレージ量はあらかじめFargateタスクストレージとしてタスク起動時に割り当てられており、割り当て量はPVによって異なります。

## コンテナプラットフォームごとのハードウェアリソースの割り当て



## Fargate で割り当て可能な CPU・メモリの対応表

CPU	メモリ
256 (.25vCPU)	512 (0.5GB)、1024 (1GB)、2048 (2GB)
512 (.5vCPU)	1024 (1GB)、2048 (2GB)、3072 (3GB)、4096 (4GB)
1024 (1vCPU)	2048 (2GB)、3072 (3GB)、4096 (4GB)、5120 (5GB)、6144 (6GB)、7168 (7GB)、8192 (8GB)
2048 (2vCPU)	4096 (4GB) ~ 16384 (16GB) (1024 (1GB) 単位で増やすことが可能)
4096 (4vCPU)	8192 (8GB) ~ 30720 (30GB) (1024 (1GB) 単位で増やすことが可能)

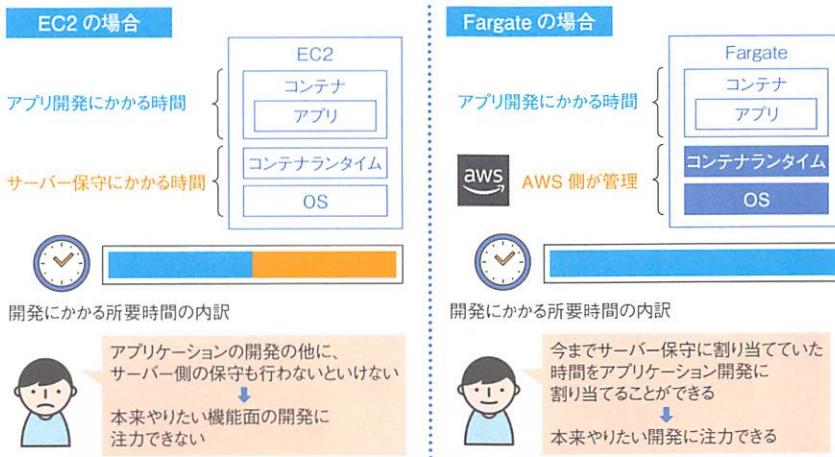
1vCPU = 1024CPU ユニットと換算され、  
128 (0.125vCPU) ~ 10240 (10vCPU)  
までの範囲で割り当て可能

メモリは MiB または  
GB で表現可能

## Fargate のメリット

Fargate の主な特徴は、サーバー側が AWS によって管理されているという点です。プラットフォームに EC2 を用いてコンテナを動かす場合、OS の設定はもちろん、コンテナランタイムのインストールや設定をする必要があり、またセキュリティパッチの適用など、定期的なメンテナンス対応も必要になります。Fargate を利用することで、これらのメンテナンス作業が不要となるため、ここに割り当てていた時間を削減でき、ECS を利用するハードルを大きく下げられます。

### Fargate ではサーバー側の運用の労力を考えなくてよい



ECS の料金面については、プラットフォームが EC2 の場合はインスタンスの利用料のみを支払う料金体系ですが、Fargate の場合はコンテナが利用したリソース量に応じて独自の料金体系が組まれています。Fargate のほうがやや割高ではありますが、サーバー側の管理費用が削減できることにより享受できるメリットを考慮すると、懸念するほどの料金差ではありません。

#### 2vCPU、4GB メモリの場合(東京リージョン)

Fargate	c5.large
0.13332USD (=2×0.05056USD+4×0.00553USD)	0.107USD

#### 2vCPU、16GB メモリの場合(東京リージョン)

Fargate	r5.large
0.1896USD (=2×0.05056USD+16×0.00553USD)	0.152USD

Fargate 料金については下記で計算

$$\text{料金} = \text{タスクに割り当てる vCPU 数} \times 1 \text{ 時間単位の vCPU 実行単価} + \text{タスクに割り当てるメモリ数} \times 1 \text{ 時間単位のメモリ実行単価}$$

## Kubernetes をより使いやすくするサービス

本節の冒頭で紹介したとおり、AWS におけるコンテナオーケストレーションサービスとして ECS の他に、EKS (Elastic Kubernetes Service) があります。AWS 上で **Kubernetes** を利用することができるサービスです。

Kubernetes はグーグルが開発したコンテナオーケストレーションツールです。現在は CNCF (Cloud Native Computing Foundation) と呼ばれる団体からオープンソースソフトウェアとして公開されています。先頭の K と末尾の s の間に 8 文字あるということで「**k8s**」と表記されることが多いです。k8s を動かすうえではいくつかのコンポーネントが必要になります。そのため、k8s の環境を自前で構築しようとすると、これらのコンポーネントをすべて管理する必要が出てきます。k8s はそれ自身の構築やその後の運用に時間やコストがかかることが課題でしたが、EKS を用いることでこれらの課題をクリアしたうえで運用することが可能です。

k8s はそれ自体がプラットフォームとなっているため、システム要件として k8s を使うという要件がなければ、基本的には **AWS でコンテナを動かす = ECS を使う**、という選択になります。しかし、k8s で動くことを前提としたシステムで、かつ AWS 上での稼働を考えている場合は、EKS は必須のサービスといえます。

## EKS は AWS 上で Kubernetes を利用できるサービス



09

→ その他のコンピューティングサービス

# サーバー知識なしで使える 代表的なサービス 5 選

**keyword**

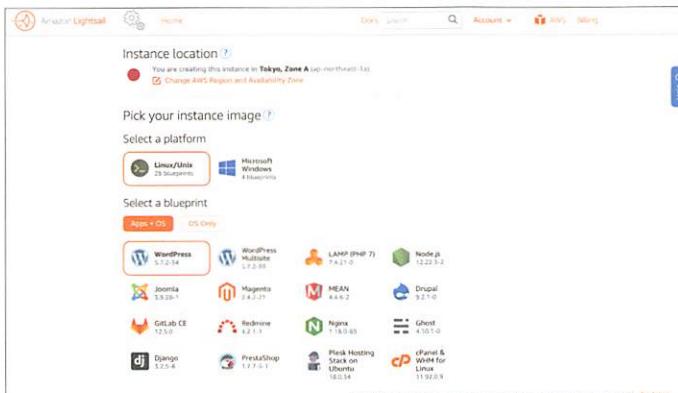
- AWS Lightsail
- AWS Elastic Beanstalk
- AWS App Runner
- AWS Batch
- AWS Outposts

## 用途に合わせて使えるサービスが提供されている

EC2 や ECS、EKS は構築の自由度が高く、これらに習熟していれば実質どんなシステムでも構築できてしまいます。とはいえ、サーバー知識はないけれど用意したアプリケーションを動かしたい、単純な構成のシステムをすぐに構築したいといったニーズもあり、AWS では用途によって適切な構成をすぐに提供するためのコンピューティングサービスも提供されています。

### AWS Lightsail

**AWS Lightsail**（以下 Lightsail）は、一般的によく利用される構成の仮想サーバーを簡単に素早く構築するためのサービスです。次図のようなわかりやすいアイコンと説明で構築画面が表示され、それに沿ってクリックしていくだけで簡単に指定したソフトウェアがインストールされた仮想サーバーが作成できます。作成したインスタンスには Web ブラウザからコンソールアクセス可能です。

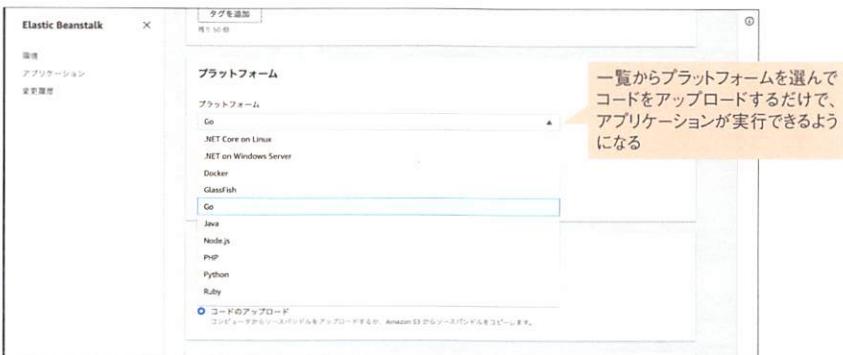


サービス自体はシンプルですが、複数のインスタンスを作成すると自動的にアクセスが負荷分散されるようになったり、SSL/TLS 証明書による通信の暗号化にも対応していたりと基本的な機能はしっかりと用意されています。

また、データベースやコンテナ環境の作成にも対応していることに加え、VPC を通じて他の AWS サービスと連携できるので、簡単な構成のシステムは Lightsail を使えばすぐに構築できるようになっているといえるでしょう。一方、EC2 で構築するときのようにミドルウェアのバージョンを考慮してインストールしたり、ELB を利用して柔軟な負荷分散を設定したりといった自由度は持ちません。Lightsail でサーバーを構築した後、そういった機能を使いたいと思ったときのために簡単に EC2 へ移行できる機能も備わっています。

## AWS Elastic Beanstalk

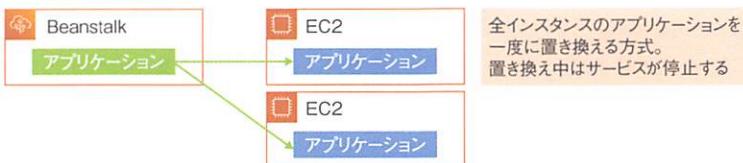
**AWS Elastic Beanstalk**（以下 Beanstalk）は、Java、.NET、PHP、Node.js、Python、Ruby、Go、Docker といった主要な言語、環境で開発されたアプリケーションをデプロイ（配置）するための環境一式を自動的に作成するサービスです。具体的には、次図のような一覧からプラットフォームを指定してアプリケーションのコードをアップロードするだけで、EC2 や RDS、S3 といった AWS リソースが自動的に構築され、アプリケーションがデプロイされます。



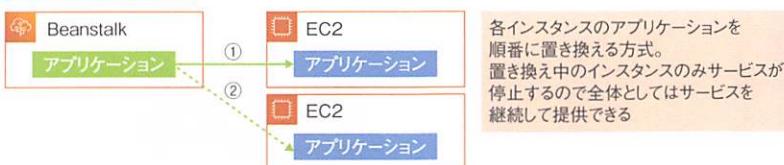
EC2 や RDS といった AWS サービスが使われる所以、自分で一から構築することでも同じ構成が実現できますが、サーバーの知識がなくとも構築ができるという利点があります。また、複数のデプロイ方式に対応しており、簡単に必要に応じたデプロイ方式を選択してデプロイできることも大きな特徴です。

## Beanstalk には 3 種類のデプロイ方式が用意されている

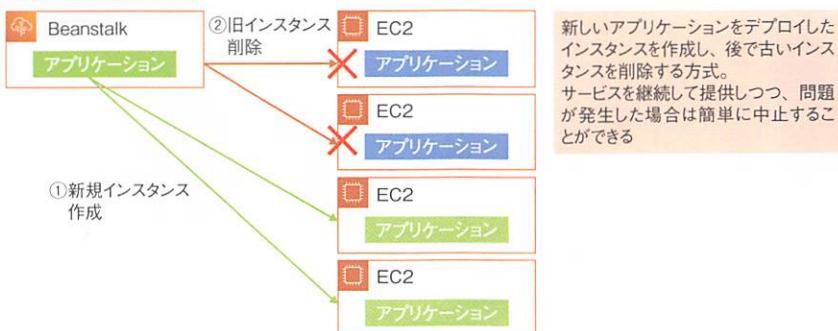
### ALL at once 方式



### ローリング方式



### イミュータブル方式



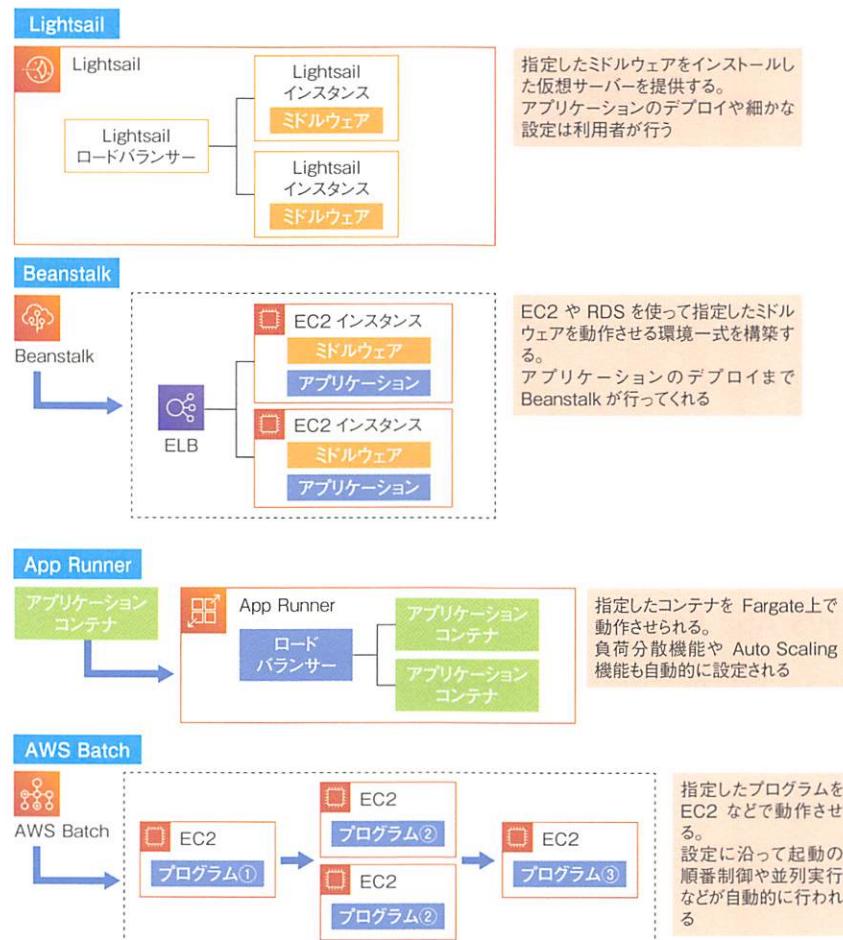
## AWS App Runner

AWS App Runner (以下 App Runner) は、コンテナ化されたアプリケーションを簡単にデプロイするサービスです。使い方も非常に簡単で、コンテナイメージを指定し、サービスポートや Auto Scaling などいくつかの設定を行うだけでコンテナを Fargate へデプロイしてくれます。2021年5月に追加された比較的新しいサービスで、今後にも大きな期待が持てます。

## AWS Batch

**AWS Batch** は、指定したプログラムを EC2 や Fargate で動作させるサービスです。これだけだと Lambda と同じように聞こえてしまいますが、動作させるプログラムの順序管理や、多数のプログラムの並列稼働といった複雑な制御に特化したサービスとなっています。

### Lightsail、Beanstalk、App Runner、AWS Batch の利用イメージ

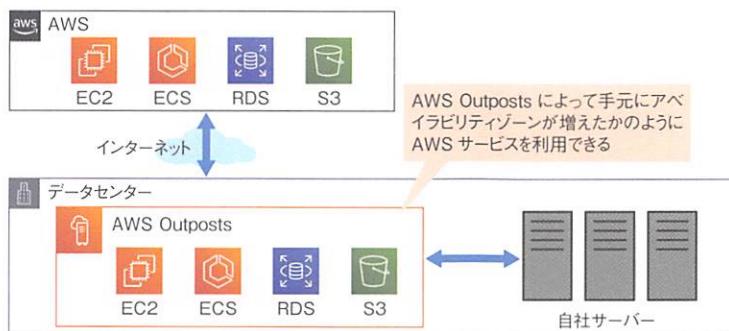


## AWS Outposts

最後はここまでとは少し毛色の違うサービスを紹介します。AWS Outposts は、AWS が物理サーバーを貸し出すサービスで、利用者がオンプレミス環境で AWS と同じサービスを動かすことができるようになります。つまりは AWS をプライベートクラウド化して提供するサービスです。

AWS Outposts はデータセンターに設置できるような一式のサーバー機器のセットで提供され、内部では EC2 や ECS、RDS、S3 といったサービスが動作するように作られています。これをデータセンターに設置し、AWS とインターネットなどで接続することで、データセンターに AWS の一部が出張してきたかのように利用できます。少し難しい表現になりますが、具体的には AWS Outposts を AWS の新たなアベイラビリティゾーンとして使うことができます。

### AWS をプライベートクラウド化して利用できる





## ストレージサービス

ストレージとはデータを格納する場所のことです。AWS の代表的なストレージサービスである S3 およびその特徴と、その他のストレージサービスの基本機能を解説します。

- section 01** Amazon S3 (Simple Storage Service) とは  
Amazon S3 はデータの保存場所
- section 02** Amazon S3 の基本  
Amazon S3 の基本用語と基本操作
- section 03** Amazon S3 のライフサイクルとバックアップ  
Amazon S3 のデータを適切に保存する
- section 04** Amazon S3 の外部公開とアクセス制御  
Amazon S3 のデータを安全に公開する
- section 05** Amazon EC2 のストレージ (EBS) とバックアップ  
仮想サーバーのデータは EBS に保存する
- section 06** その他のストレージサービス  
データ共有・バックアップ・転送のためのサービス

✓ Amazon S3 (Simple Storage Service) とは

# 01 Amazon S3 はデータの保存場所

**keyword** • オブジェクトストレージ • Amazon S3 • S3 の特徴 • S3 の利用料

## S3 はオブジェクトストレージ

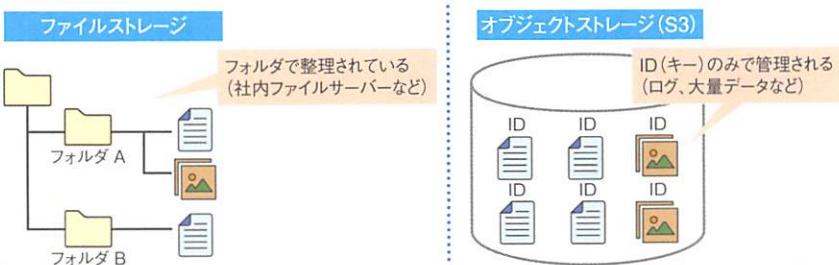
Amazon Simple Storage Service (以下 S3) は、AWS が提供するオブジェクトストレージサービスです。ここでいうストレージとはデータを保存する場所を、オブジェクトとはテキストファイルや音声ファイルといったデータを指します。

オブジェクトストレージとは、従来のファイルストレージのようなフォルダ構造は持たず、オブジェクトキーによりデータを一意に特定してデータの出し入れ、管理を行うストレージです。キーのみでデータを管理するため、シンプルで大容量のデータを保存、管理できます。

## Amazon S3 はデータを格納できるサービス



## オブジェクトストレージは ID (キー) でデータを管理する



社内のドキュメント管理など更新が頻繁にあるものは、現在でもファイルストレージが多く使われています。S3は、動画や画像、ログファイル、バックアップファイル、Webコンテンツなど、AWS上でシステムを構築する際にさまざまな目的で利用されます。

## S3は低コストで高い耐久性を持つ

S3には、次のような特徴があります。

### ●容量無制限

1オブジェクトあたり5TBという制約はあるものの、オブジェクト数やデータ容量の制限はありません。

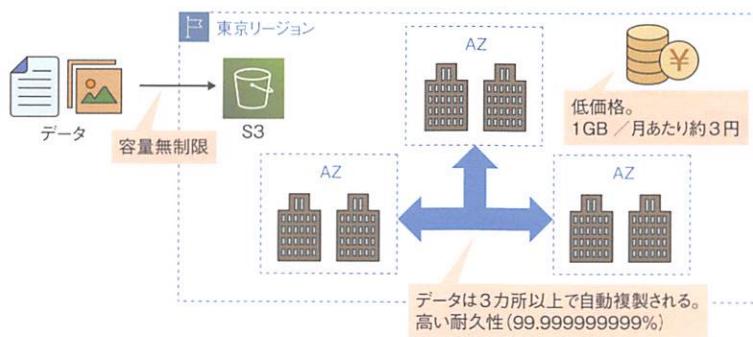
### ●高い耐久性

標準で、3つ以上のAZにデータがコピーされます。コピーすることでデータの耐久性を高めており、AWSでは99.99999999%（9×11、イレブンナイン）という高い耐久性の数値を示しています。

### ●低コスト

東京リージョン、標準のストレージで保存料金は1カ月あたり0.025USD／GBとなっています。後ほど紹介するストレージクラスの変更を行うと、さらに安い料金でデータ保存が可能です。

## S3は容量無制限で、高い耐久性を備え、低コストで使える



料金は使用した分だけ、容量は無制限のため、事前に確保する容量を決めずに気軽に利用を始められます。

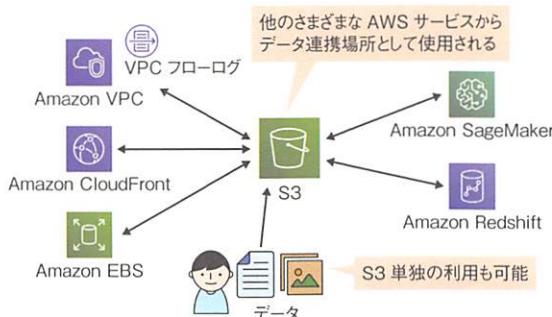
## S3 は他のサービスと連携する

高い拡張性、耐久性、低コストという特性から、S3 は他の AWS サービスからも多く使われます。

たとえば、Amazon VPC というネットワークサービスのログ（VPC フローログ）や、コンテンツ配信ネットワーク（CDN）サービスである Amazon CloudFront のアクセスログなど、各種ログの保存場所として S3 が使用されます。その他にも、EC2 のストレージである EBS のスナップショット保存や機械学習サービスの入力情報の格納場所など、幅広い用途で使用されます。

もちろん、利用者がデータを S3 に直接アップロードするといった形で、S3 単独で利用することも可能です。

## S3 はさまざまなサービスのデータ保存場所としても利用される



## S3 の料金例

S3 の料金は、データ保存容量、データアクセス回数、データ転送量の 3 タイプで発生します。

項目	料金(東京リージョン)
データ保存*1	0.025USD／GB
データアクセス(アップロード)	0.0047USD／1,000回
データアクセス(ダウンロード)	0.00037USD／1,000回
データ転送(S3→インターネット)*2	0.114USD／GB

\*1 最初の50TB／月の料金で、それ以上は少し安くなります。

\*2 最初の100GB分と、インターネット→S3分はすべて無料です。

次のような例を見ていきます。

- ・1月1日～15日で500GBのデータを格納
- ・1月16日～31日で900GBのデータを格納
- ・1月中にアップロードを1,000回、ダウンロードを2,000回実施
- ・1月中にダウンロードしたデータはトータル200GB

このとき、データアクセス料金は保存と転送に比べて極めて安くなるため、ここでは考慮しません(0.01USD以下)。保存と転送で、次のとおり料金が発生します。

#### データ保存：

$$(500\text{GB} \times 15\text{日間} \times 24\text{時間} + 900\text{GB} \times 16\text{日間} \times 24\text{時間}) \\ \div (31\text{日間} \times 24\text{時間}) = 706\text{GB} \quad (\text{1カ月の平均使用量})$$

$$706\text{GB} \times 0.025 = 17.65\text{USD}$$

#### データ転送：

$$(200\text{GB} - 100\text{GB} \text{ (無料分)}) \times 0.114 = 11.4\text{USD}$$

合計：**29.05USD** (1USD110円計算で**約3,196円**)

データ転送料金が、保存料金よりも高くなる場合もあるため、大量のデータをダウンロードする場合は注意が必要です。

料金の詳細については AWS 公式のページも確認してください。

#### ▶ Amazon S3 の料金

<https://aws.amazon.com/jp/s3/pricing/>

# 02 Amazon S3 の 基本用語と基本操作

**keyword** • バケット • オブジェクト • キー • API

## S3 で使用される重要な用語

ここで、S3 を使用するうえで重要な用語を押さえておきましょう。

### ● バケット

オブジェクト（データ）を保存する場所のことです。利用者は作成したバケットにオブジェクトを格納します。バケットに使用する名前は、世界中（全リージョン）で一意である必要があります。つまり、「test-bucket」という名前でバケットを作成したら、他の利用者を含め、全リージョン・全 AWS アカウントで同じ名前のバケットは作成できません。

### ● オブジェクト

バケットに格納されるデータ本体です。バケット内のオブジェクト数は無制限に保存できますが、1 オブジェクトの最大サイズは 5TB までという制約があります。

### ● キー

オブジェクトの格納 URL パスです。バケット名とキー名、オブジェクト名を組み合わせて一意になるよう設定されます。

## バケット、オブジェクト、キーの関係



S3はオブジェクトストレージで、実体としてフォルダは持っていないのですが、「/」という文字列を区切り記号として判断し、マネジメントコンソール上ではあたかもフォルダ構造であるかのようにオブジェクトが表示されます。

## マネジメントコンソール上ではフォルダ構造のように表示される

Amazon S3 > test-bucket-abc-202112 > folder/

「/」があるとフォルダ構造で表示される

S3 URI をコピー

オブジェクト プロパティ

オブジェクト (2)

オブジェクトは、Amazon S3 に保存された基本的なエンティティです。Amazon S3 イベントリ を使用して、バケット内のすべてのオブジェクトのリストを取得できます。他のユーザーが自分のオブジェクトにアクセスできるためには、明示的にアクセス権限を付与する必要があります。詳細は[こちら](#)。

C S3 URI をコピー URL をコピー ダウンロード 開く 編集 アクション フォルダを作成

アップロード

Q ブラウザでオブジェクトを検索

名前	タイプ	最終更新日時	サイズ	ストレージクラス
object1.txt	txt	2021/12/19 10:29:14 PM +09	43.0 B	スタンダード
object2.txt	txt	2021/12/19 10:29:14 PM +09	43.0 B	スタンダード

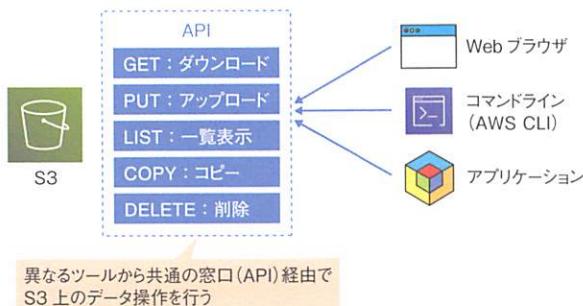
## S3のデータは画面やプログラムから操作できる

S3には、データを登録したり削除したりする API (Application Programming Interface) が用意されており、それを呼び出すことにより S3内のオブジェクトを操作します。APIは操作を受け付ける窓口のようなものであり、共通の窓口を用意して、ブラウザやプログラムなどさまざまな場所からデータの操作ができるようにしています。

次表に、主要な操作をいくつか紹介します。

GET	S3からデータ(オブジェクト)をダウンロードする操作
PUT	S3にオブジェクトをアップロードする操作。新規アップロードも更新(上書き)もこの操作となる。
	1つのPUT操作では最大5GBまでのため、それ以上のデータをアップロードする場合は、 <b>マルチパートアップロード</b> と呼ばれる機能を使用して、複数のパートに分割して最大5TBのデータがアップロードできる
LIST	S3バケット内のオブジェクトを一覧で表示する操作。キーの前方一致でパス指定のフィルタ(データの絞り込み)也可能
COPY	S3内でオブジェクトのコピーを行う操作。異なるバケットやリージョン間でもコピーが可能
DELETE	S3内の任意のオブジェクトを削除する操作

## S3 のデータは API を通してブラウザやプログラムから操作できる



また、マネジメントコンソール上では次図のように表示され、利用者は簡単にデータのダウンロードとアップロードが可能です。アップロードはファイルのドラッグ & ドロップでも可能です。

## 画面から簡単にダウンロードとアップロードができる

The screenshot shows the Amazon S3 console interface. At the top, it displays the path: 'Amazon S3 > test-bucket-202105 > folder/'. On the left, there's a sidebar with 'folder/' and a 'オブジェクト (1)' section. The main area shows a table with one item: 'test.txt' (Type: txt, Size: 7.0 B, Storage Class: スタンダード). A red box highlights the 'ダウンロード' (Download) button in the toolbar above the table. Another red box highlights the 'アップロード' (Upload) button in the same toolbar. Callout boxes provide instructions: '①ダウンロードしたいオブジェクト (データ) を選択' (Select the object you want to download (data)) points to the table row for 'test.txt'; '②クリックするとダウンロードできる' (Click to download) points to the 'ダウンロード' button; and 'ここからデータのアップロードもできる' (You can also upload data from here) points to the 'アップロード' button.

# 03 Amazon S3 のデータを適切に保存する

**keyword** •ストレージクラス •ライフサイクル設定 •S3 Analytics •バージョン管理  
•災害対策 •クロスリージョンレプリケーション

## 使用しないオブジェクトは安く保存する

S3 には、コスト（料金）や可用性の異なる複数のストレージクラスが用意されています。コスト効率を重視したアーカイブ向けストレージである、Amazon S3 Glacier（以下 S3 Glacier）というストレージクラスもあります。

## データの利用頻度やコストなどに応じてストレージクラスを選ぶ

ストレージクラス	特徴	保存AZ	可用性
標準(STANDARD)	デフォルトのストレージクラス	3以上	99.99%
Intelligent-Tiering	アクセス頻度に応じて4つのアクセス層に自動分割してコスト削減する	3以上	99.90%
標準低頻度アクセス(STANDARD-IA)	標準より低コストだが、データの取り出し容量に対し課金が発生	3以上	99.90%
1ゾーン低頻度アクセス(1ZONE-IA)	1AZにのみデータを格納。データ取り出しに課金あり	1	99.50%
S3 Glacier Instant Retrieval	S3 Glacier Instant Retrievalよりさらに保存は低コストだが、取り出しコストが高くなる	3以上	99.90%
S3 Glacier Flexible Retrieval	S3 Glacier Instant Retrievalよりさらに保存は低コストだが、取り出しこストがかかり時間もかかる(数分または数時間)	3以上	99.99%
S3 Glacier Deep Archive	S3 Glacierよりも低コストだが、取り出しに多くの時間(半日～)がかかる	3以上	99.99%

・表に記載のストレージクラスの耐久性はすべて99.99999999%です。

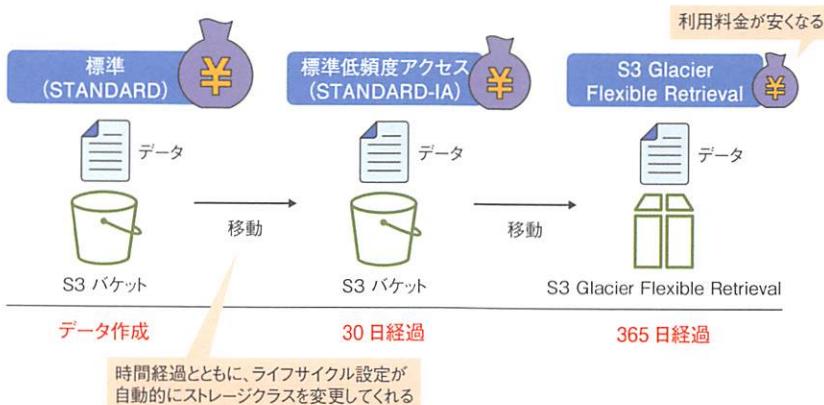
- ・可用性はS3にデータをアップロードしたり、ダウンロードしたりという、**使用できるか**という観点の**数値**となります。また、表に記載の可用性は設計上の値で、保証するものではありません。
- ・ストレージコスト(料金)は標準が一番高く、表の下に行くほど安くなり、S3 Glacier Deep Archiveが最も低成本となります。

データは念のため保存しておきたいが、ほとんど使用することはないといった場

今は、S3 Glacier Deep Archive など安いストレージクラスを選択しましょう。逆に、Web ページで表示するコンテンツなど、いつでも利用できるようにしておくべきデータを保存する場合は、標準（STANDARD）を選択しましょう。利用頻度や求められる可用性、コスト要件に応じて最適なストレージクラスを選定します。

また、S3 には**ライフサイクル設定**という機能があり、自動的にストレージクラスの変更も可能です。たとえば、作成から 1 年経過したオブジェクトは S3 Glacier Flexible Retrieval に移行するといった設定ができます。オブジェクトの削除も可能です。ただし、一度低成本ト（たとえば標準→ STANDARD-IA）のストレージクラスに移行すると、標準クラスには戻せません。

### 利用頻度に応じて自動的にストレージクラスを変更することも可能



ストレージクラスの種類が多く、どれを使用したらよいか最初は迷うかもしれません。S3 Analytics (S3 分析) という機能を使用して、どれだけアクセスがあったかなど、データのアクセス状況を確認できます。S3 Analytics でデータ容量とアクセス状況を確認してからストレージクラスを決定するのもよいでしょう。

### データへのアクセス状況は S3 Analytics で確認できる

#### S3 Analytics の利用イメージ

データ容量 1TB

アクセス量 10GB

データ容量に対してアクセスが少ない（ほとんどアクセスが無い）ので、ストレージクラスを安いものに変更したほうがよい

## バージョン管理と海外へのコピー

S3 上のデータは 3 カ所以上にコピーされるため、AWS が原因でデータが失われるということはあまり無いでしょう。そのため、データセンター障害など AWS 障害に備えたデータのバックアップは基本的に不要です（後述しますが、東京全体の障害などを考慮する場合は必要です）。

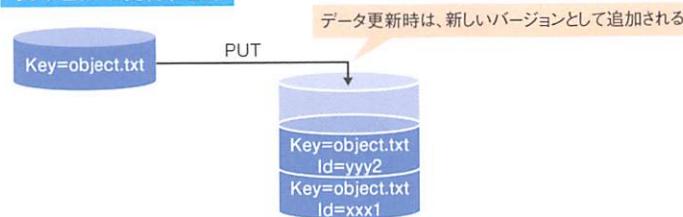
ただし、利用者の操作ミスによりデータを削除してしまったり、意図しない変更をしてしまったりする可能性はあります。S3 にはバージョン管理という機能があり、これを有効にすることでデータの誤操作があっても復旧可能です。

S3 バケットのバージョン管理機能を有効にすると、すべての世代のオブジェクトが履歴情報として保存されます。オブジェクトの更新時（PUT）は、既存のオブジェクトとは異なる新しいバージョン ID が付与されます。バージョン管理の設定は S3 バケット単位となり、オブジェクト単位の設定はできません。

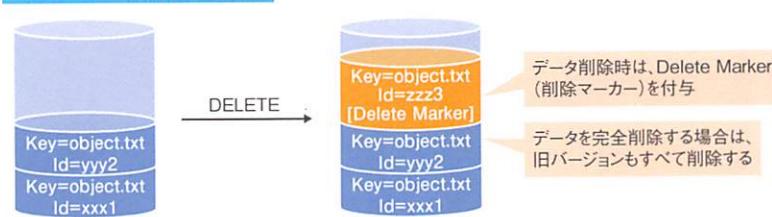
削除操作（DELETE）を行った場合は、オブジェクトが実際に削除されるわけではなく、削除マーカーが付与されます。過去バージョンはそのまま残ります。削除マーカーが付与されたオブジェクトに GET 操作を行うと、404 Not Found エラーが返されます。オブジェクトを完全に削除する場合は、バージョン ID を指定することで特定のバージョンを永久に削除できます。

### S3 バケットのバージョン管理機能が有効なときの動作

#### オブジェクトの更新 (PUT)



#### オブジェクトの削除 (DELETE)

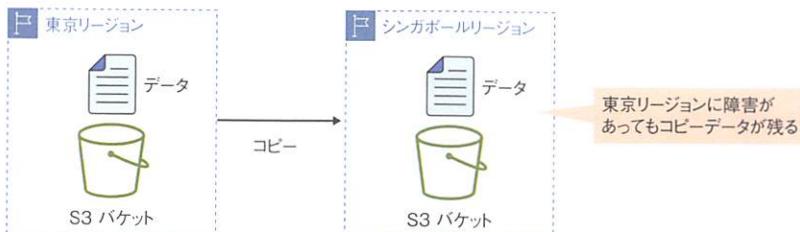


S3 のバージョン管理は、利用者の誤操作による削除や変更も復旧することができる便利な機能ですが、過去バージョンは 1 オブジェクトとして課金されるため追加料金が発生します。前項で紹介したライフサイクル設定を使用して、過去バージョンを一定期間で自動削除するといった運用も可能です。

また、S3 には海外リージョンにデータをコピーする機能もあり、これを使用することでデータの耐久性をさらに高めることができます。データ保存先とは異なるリージョンにコピーする機能をクロスリージョンレプリケーション (CRR) と呼びます。

筆者も実際に経験したことは無いですが、システム構築の現場では、データの消失を防ぐために関東全体が災害にあった場合も想定して設計を行うことがあります。こういった災害に備えることを災害対策 (Disaster Recovery、DR) と呼びますが、S3 では S3 自身の機能でこれを実現できます。

### 海外リージョンへデータをコピーして耐久性を高めることも可能



なお、2021 年に大阪が正式リージョンとして開設されたため、データは国内に保存するという規則がある場合でも、東京 ⇄ 大阪間でデータのコピーを行うことで、日本国内で災害対策が可能となっています。

# 04 Amazon S3 のデータを安全に公開する

**keyword** • S3 バケットのアクセス制御 • Web サイトホスティング機能 • データの暗号化  
• AWS KMS

## S3 バケット内へのアクセス制御

S3 バケットには個人情報などの重要な情報も格納される場合があるため、許可された人が許可されたオブジェクトへ正しくアクセスできるよう、**アクセス制御**を行うことが重要になります。

S3 バケットへのアクセス制御は、次の 3 種類の方法で実装できます。

### ● ユーザーポリシー (IAM ポリシー)

データを操作する側を制御する設定です。「Aさんは S3 バケットのデータ取得のみ可能で、アップロード不可」といった具合に、ユーザーに対して設定できます。

### ● アクセスコントロールリスト (ACL)

オブジェクト（データ）またはバケット単位で設定できるアクセス制御です。「データ A は別の AWS アカウント X にアクセス許可」「データ B はすべてのユーザーにアクセス許可」といった設定が、データごとに細かく設定できます。

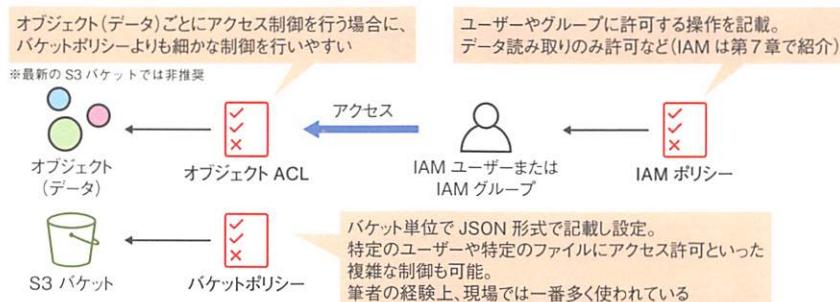
※最新の S3 バケットでは ACL の使用は非推奨です

### ● バケットポリシー

バケット単位で設定するアクセス制御です。バケット単位といつても、「ファイル名が logs で始まるデータにアクセス許可」「IP アドレス XX.XX.XX.XX からアクセス許可」といった複雑な設定が可能です。設定は JSON 形式で記載します。

バケットポリシーや ACL の設定内容次第では、オブジェクトを全世界に公開するパブリックな状態になるため注意が必要です。意図的に公開することもありますが、不要であれば公開は情報漏えいの観点から避けるべきです。S3 には**ブロックパブリックアクセス**という機能があり、意図せず S3 バケットおよびオブジェクトが公開状態になることを防ぐこともできます。

## S3 バケットへのアクセス制御は3種類の方法で行える

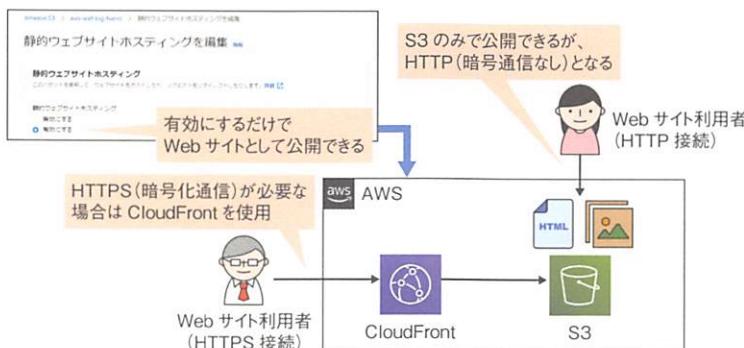


## S3 を使用して Web サイトを公開する

S3 バケットに格納した HTML や CSS、JavaScript などのコンテンツファイルを、Web サイトとして公開できます。S3 の **Web サイトホスティング機能**を有効にすると、オブジェクトが Web コンテンツとして公開されます。有効にすると `[http:// バケット名.s3-website-リージョン名.amazonaws.com]` といった URL でアクセスできます。前述したバケットポリシー や オブジェクト単位の ACL を使用して、特定のユーザーにコンテンツを公開することも可能です。

なお、S3 のみでは暗号化された通信を使用する HTTPS に対応しておらず、HTTP アクセスとなります。インターネットに公開する本番システムでこの機能を利用する場合は、コンテンツ配信ネットワーク (CDN) サービスである Amazon CloudFront を併用して HTTPS アクセスができるようにします。

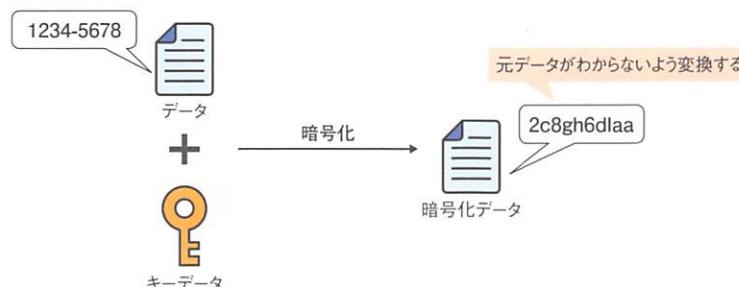
## S3 バケットを Web サイトとして公開できる



## 保存されるデータの暗号化

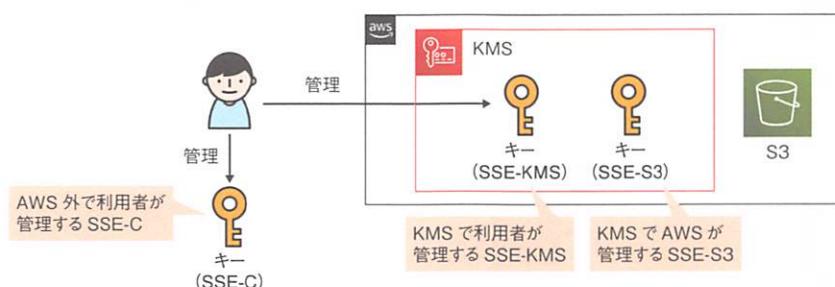
**データの暗号化**とは、データとキーデータ（鍵データ）を組み合わせてデータ内容を他人から見られないようにする仕組みです。キーデータがなければデータが読み取れない状態にしておき、キーデータを安全に管理することでデータを守ります。

### データとキーデータを組み合わせて暗号化データにする



S3では、バケットのデフォルトの暗号化機能を有効にすると、オブジェクト格納時にAWS側で自動的に暗号化が行われます。暗号化用のキーが必要になりますが、AWSが提供するキー管理機能の **AWS Key Management Service**（以下KMS）で作成したキーや、利用者が管理しているキーなど全3種類から選択して使用できます。KMSのキーを使用する場合は、利用者は暗号化の処理を意識せずに利用が可能なため、より簡単にオブジェクトを安全に保つことができます。

### AWSでは3種類の暗号化用のキーが使用できる



暗号化の設定は、S3 の画面上から簡単に行えます。利用者が管理するキー (SSE-C) を使用する場合は、事前にキーを準備する必要があります。

## 暗号化の設定はマネジメントコンソールから簡単に行える

**デフォルトの暗号化**

このバケットに保存された新しいオブジェクトを自動的に暗号化します。[詳細](#)

サーバー側の暗号化

- 無効にする バケットに格納する際にデータを自動的に暗号化できる
- 有効にする

暗号化キータイプ

お客様が用意した暗号化キー (SSL-C) を使用してオブジェクトをアップロードするには、AWS CLI、AWS SDK、または Amazon S3 REST API を使用します。

- Amazon S3 キー (SSE-S3)  
Amazon S3 が作成、管理、使用する暗号化キー。[詳細](#)
- AWS Key Management Service キー (SSE-KMS)  
AWS Key Management Service (AWS KMS) で保護されている暗号化キー。[詳細](#)

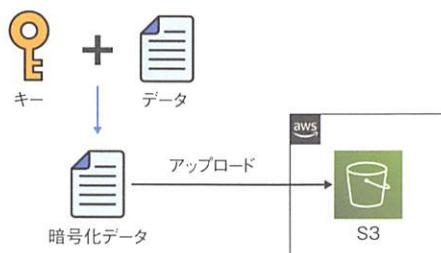
AWS KMS キー

- AWS 管理キー (aws/s3)  
`arn:aws:kms:ap-northeast-1:035015041922:alias/aws/s3`
- AWS KMS キーから選択する
- AWS KMS キー ARN を入力する

暗号化に使用するキーを選択する

S3 で提供されるのはオブジェクト格納時の暗号化ですが、S3 へ送信するときに暗号化をしたい場合は、あらかじめデータをアプリケーション側で暗号化して転送します。クライアントサイド暗号化とも呼ばれます。クライアントサイド暗号化で使用するキーは、ユーザー独自のキー、KMS 上のキーのどちらも使用可能です。なお、S3 の API へのアクセスは基本的に HTTPS で行われるため、通信全体としては暗号化されています。

## アップロード時のデータを守るにはあらかじめ暗号化しておく



通信時もデータを暗号化したい場合は  
あらかじめ暗号化しておく

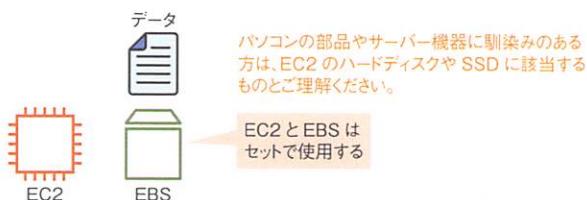
# 05 仮想サーバーのデータは EBS に保存する

**keyword** • Amazon EBS • ボリュームタイプ • スナップショット • EBS の利用料

## サーバーのデータを保存する EBS

Amazon Elastic Block Store (以下 EBS) は、EC2 とセットで使用するストレージサービスです。EC2 とセットで使用するため、EC2 上で稼働するアプリケーションのデータやログ、設定情報などを保存する目的で使用します。

### EBS は EC2 (仮想サーバー) のデータの保存場所



EC2 インスタンスの作成時にはデフォルトで 1 つの EBS がアタッチ（接続）され、複数アタッチすることも可能です。

アプリケーションによって、求められるデータの書き込みおよび読み込み速度が異なることがあります、EBS では対応する速度に応じた複数のボリュームタイプが用意されています。EBS の書込／読込性能は、IOPS (Input/Output Per Second) と呼ばれる 1 秒あたりの書込／読込回数を単位として性能値を決定します。

ボリュームタイプ	概要
汎用SSD (gp2, gp3)	バランスの取れた汎用タイプで、一般的な用途に使用
プロビジョンドIOPS SSD (io1, io2, io2 Block Express)	ストレージパフォーマンス (IO) が必要な場合に使用。必要な IOPS を利用者が指定する
スループット最適化HDD (st1)	低コストの磁気ストレージ
Cold HDD (sc1)	st1 よりさらに低コストの磁気ストレージ。アクセス頻度が低い場合に使用

特別な性能要件が無い場合は、デフォルトの汎用 SSD を選べばよいでしょう。高性能なアプリケーションが稼働する場合はプロビジョンド IOPS SSD、できるだけ安価にしたい場合はスループット最適化 HDD か Cold HDD が選択肢となります。

## サーバーのバックアップとイメージ管理

EBS には重要なデータが含まれることもありますが、故障するとデータが無くなってしまいます。筆者自身は AWS 側のシステム障害が原因でデータが無くなつた経験はありませんが、可能性としては考えられます。また、人による誤操作によりデータが無くなる可能性もあります。

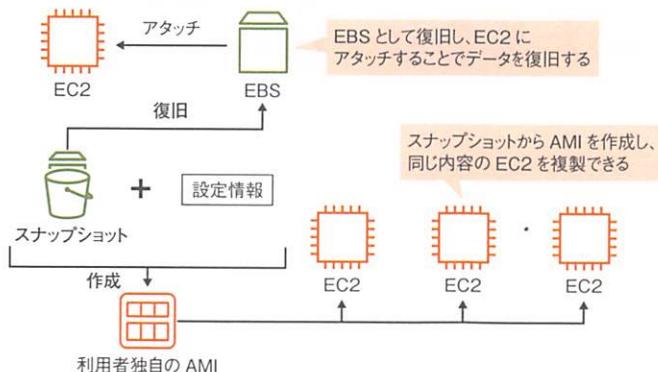
こういったデータの消失には**バックアップ**を取得して備える必要がありますが、EBS には**スナップショット**というデータのバックアップ機能があります。スナップショットには取得した時点のデータや設定情報など、EBS に保存していたすべての情報が含まれます。

### 故障に備えて EBS のバックアップを取得しておく



取得したスナップショットを復旧する場合は、一度 EBS ボリュームとして復旧し、それを EC2 に再びアタッチすることで利用できます。また、スナップショットといくつかの設定情報（アーキテクチャ情報など）を組み合わせて、**利用者専用の AMI** も作成できます（AMI については p.59 参照）。3-2 節の仮想サーバー作成の例では AWS 管理の AMI を使用しましたが、利用者が独自に設定した AMI を使用して仮想サーバーの複製も簡単にできます。

## スナップショットからデータの復旧やサーバーの複製が行える



### EBS の料金例

3-2 節 (p.58) では EC2 について、停止中は料金が発生しないと紹介しました。しかし、EC2 とセットで使用される EBS は、容量を確保している時点で料金が発生し、停止して料金を抑えるといったことはできません。そのため、EC2 は停止していても、データを保存している EBS には料金が発生すると理解しましょう。

東京リージョンでボリュームタイプ gp3 (汎用 SSD) を選択した場合、1GB／月あたり **0.096USD** の料金が発生します。つまり、100GB の EBS が 1 カ月存在した場合、**9.6USD** (1USD110 円計算で**約 1,056 円**) となります。月の使用期間で計算されるため、たとえば 15 日間利用の場合は半額程度の料金となります。

データ保存量だけでなく、IOPS (読み書き回数) やデータ処理容量にも料金が発生しますが、データ保存と比べ安い料金となっています。ただし、大量の読み書きを行う場合は注意が必要です。

ボリュームタイプによっても料金の計算方法は異なります。詳細が気になる方は AWS の公式ページも確認してください。

#### ▶ Amazon EBS の料金

<https://aws.amazon.com/jp/ebs/pricing/>

# 06

## データ共有・バックアップ・転送のためのサービス

keyword

- Amazon EFS
- Amazon FSx
- AWS Storage Gateway
- AWS Transfer Family
- AWS Backup
- AWS DataSync
- AWS Snow Family

### S3 と EBS が基本。場合により有用なサービスもある

ここまで紹介してきたように、AWS のストレージといえば S3 と EBS が基本です。使い方の自由度が高く可用性も高いため、実際その 2 つで事足りることが多いです。

しかし、これら以外にも、場合によってはより有用な選択肢となりえる共有ストレージのサービスや、ストレージ管理のための便利なサービスがいくつか存在します。ここではその中から代表的なサービスを紹介します。

### Amazon Elastic File System

EBS は 1 つの EC2 インスタンスにアタッチして利用する非常に高速なストレージです。一方、S3 は多くのクライアントからの同時利用が可能なストレージですが、HTTPS を利用した API でアクセスする仕組みとなっており転送速度があまり速くありません。

そこで、比較的高速にデータを転送できる NFS (Network File System) というプロトコルを用いた、複数の EC2 インスタンスからの同時利用が可能なストレージとして、**Amazon Elastic File System** (以下 EFS) が提供されています。

EC2 インスタンスと書きましたが、NFS は Linux で利用されるプロトコルであり、Linux であればオンプレミスのサーバーからでも利用できます。逆に、EC2 インスタンスであっても Windows には対応していません。

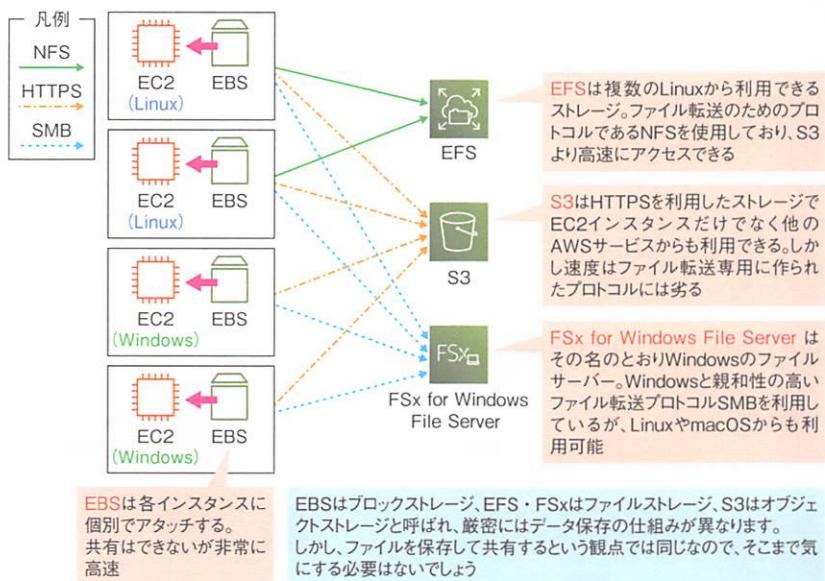
### Amazon FSx

Amazon FSx (以下 FSx) はファイルサーバーを構築するためのサービスです。

主に Windows で利用されるファイル共有プロトコルである SMB (Server Message Block) を利用する **Amazon FSx for Windows File Server** と、大規模なクラスターコンピューティング・スーパーコンピューターなどで使われる Lustre という高性能なファイルシステムを利用する **Amazon FSx for Lustre** の 2 種類が提供されています。

SMB は Windows だけでなく Linux や macOS も対応しているので、Linux 以外で共有ファイルサーバーを利用したい場合は Amazon FSx for Windows File Server が選択肢に挙がるでしょう。

## S3、EBS、EFS、FSx、それぞれの特徴



なお、課金体系が異なるので単純な比較は難しいのですが、課金のメインの対象である「保管されるデータ量」をもとに費用面を比較すると、S3 の安さは圧倒的です。普段利用しないバックアップや巨大なデータは基本的に S3 に置くことが推奨されていますが、その理由のひとつがこの安さであるといえます。

### 保管されるデータ量 1GBあたりの月額（東京リージョン）

S3 : 0.025USD、EBS : 0.096USD、EFS : 0.08 USD、FSx : 0.156USD

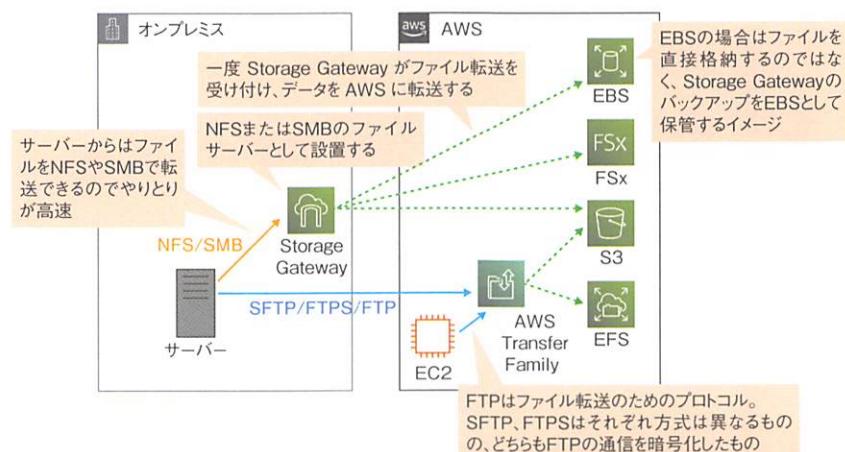
## AWS Storage Gateway

**AWS Storage Gateway** (以下 **Storage Gateway**) は、オンプレミス上にサーバー機器もしくは仮想サーバーとして設置することで、オンプレミスの機器から受け取ったデータを AWS 上の S3、FSx、EBS とやりとりするサービスです。次項の図で示すようなイメージとなります。オンプレミスの機器からは Storage Gateway は NFS や SMB で接続するストレージのように扱えるので、直接 AWS サービスとデータをやりとりするよりも高速に処理が行えます。

## AWS Transfer Family

**AWS Transfer Family** は、S3 および EFS と HTTP や NFS ではなく、SFTP (Secure File Transfer Protocol)、FTPS (File Transfer Protocol over SSL)、FTP (File Transfer Protocol) といった FTP 系のプロトコルで通信するためのサーバーを構築するサービスです。FTP は古くから利用されているファイル転送プロトコルで、既存のシステムで使われていることが多いです。そういったシステムを AWS に移行する際に、アプリケーション内の FTP を利用する作りはそのままに、転送先を可用性の高い S3 や EFS に変更することができます。

### Storage Gateway、Transfer Family の特徴



## AWS Backup

**AWS Backup** は、その名のとおり AWS 上の EBS、EFS、FSx といったストレージ、RDS や DynamoDB といったデータベースのデータをバックアップするためのサービスです。

バックアップのルールを設定しておくだけで、それに沿ったバックアップが自動的に実行されます。バックアップは S3 に保管されるため、非常に可用性、耐久性が高いです。

## AWS DataSync

**AWS DataSync** (以下 DataSync) は、オンプレミスと AWS、もしくは AWS のストレージサービス間でデータの転送を行うためのサービスです。S3、EFS、FSx といったストレージサービスと、オンプレミスのサーバーにインストールされた DataSync Agent の間で簡単にデータの転送が行えます。最大 10Gbps (秒間 10G ビットのデータ転送速度) を実現でき、通信の暗号化機能、転送したデータの整合性チェック機能も備えているので、安全かつ高速にデータを転送できます。

## AWS Snow Family

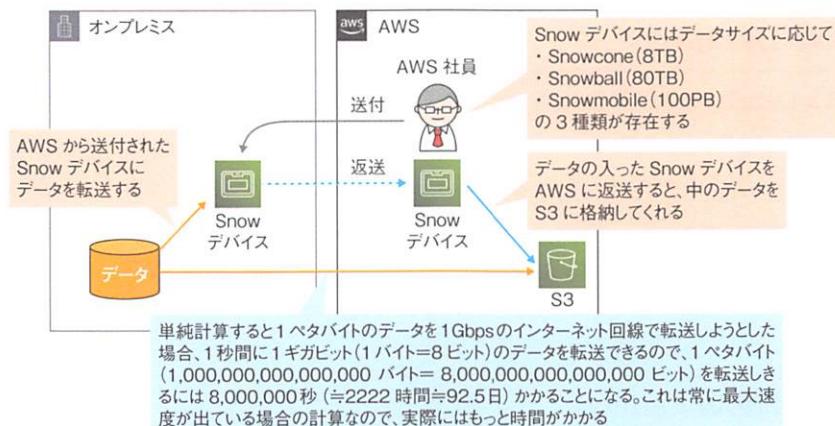
オンプレミスのデータを AWS に転送する場合、少量のデータであれば問題ありませんが、テラバイト (TB) クラスのデータを転送しようとすると膨大な時間がかかります。たとえば 100TB のデータを 1Gbps のインターネット回線で AWS に転送する場合、常時 1Gbps の速度が出たとしても約 10 日かかってしまいます。さらに、インターネット回線は契約上 1Gbps だとしても多数の人と共に用する回線である以上、常に最高速度が出るわけではありません。50 ~ 80% の速度が出れば一般的に安定した回線といわれますが、50% の速度で常時転送できたとすると 100TB のデータ転送には約 20 日かかる計算になります。こういったデータ転送ができるだけ速く行うためのサービスが **AWS Snow Family** です。

これは、物理的なストレージを AWS から借りて、そこにデータを格納して AWS に返送することで、受け取った AWS 側でデータをデータセンターから直接 AWS 内に転送してくれるサービスです。ストレージの送付のための日数がかかりますが、最大で 100PB (ペタバイト) のデータをわずか数週間で転送できます (1PB =

1024TB)。

AWS Snow Family には 3 種類の容量のストレージが用意されており、用途に応じて選択できます。最大容量の **AWS Snowmobile** では、全長 14m のトレーラーに大量に積み込まれたストレージに 100PB のデータを格納して輸送できます。もちろん大切なデータなので、GPS による追跡や 24 時間無休の監視カメラなどセキュリティ面も万全です。

### 大容量データを移す際には物理ストレージを使うサービスもある





Chapter  
5

# ネットワークと コンテンツ配信サービス

AWS を扱ううえでは欠かせない VPC をはじめとして、スケーラブルで便利なネットワークサービスについて、ネットワークの基礎知識も踏まえたうえで解説します。

- section 01** ネットワークの基礎知識  
ネットワークの重要用語を覚えよう
- section 02** Amazon VPC とは  
Amazon VPC で仮想ネットワークを作る
- section 03** Amazon VPC の主要機能①  
Amazon VPC の主要機能の使い方
- section 04** Amazon VPC の主要機能②  
VPC 同士や外部サービス、オンプレミスとの接続
- section 05** Amazon VPC の構成例  
他の AWS サービスと組み合わせた構成例
- section 06** ELB (Elastic Load Balancing) とは  
ELB で負荷分散して可用性を高める
- section 07** Amazon Route 53 とは  
高性能で手軽に使える DNS サービス
- section 08** Amazon CloudFront とは  
CloudFront で高速かつ落ちないネット配信を実現
- section 09** その他のネットワークとコンテンツ配信サービス  
多彩なネットワーク機能を提供するサービス 7 選

## 01

## ネットワークの重要用語を覚えよう

**keyword** •IP アドレス •CIDR •ファイアウォール •負荷分散 •ルーティング •DNS

## IP アドレスはネットワーク上の住所

普段パソコンやスマートフォンで Web サイトを閲覧するとき、画面上に Web ページが表示されますが、実際は任意の場所に存在する「Web サイトを提供するためのサーバー」（以下 Web サーバー）にアクセスして取得した Web ページを画面上に表示しています。

Web サーバーはいったいどこに存在するのでしょうか。それは、とあるデータセンター（サーバーを設置する施設）かもしれませんし、AWS のようなクラウド上に存在しているかもしれません。では、そのサーバーにアクセスするためにはどうすればよいのでしょうか。

実は、ネットワーク上にも住所が存在します。それが **IP アドレス**です。Web サイトにアクセスする際には、この IP アドレスをもとに住所を特定することで、そこに存在する Web サーバーにアクセスできます。

## ネットワーク上の場所には IP アドレスでアクセスする

## ○○公園に移動する場合

○○公園に行きたい



○○公園



住所：XX 市 YY 丁目 ZZ 番地

## ○○の Web サイトを閲覧する場合

○○の Web サイトが見たい



「XX.YY.ZZ.XYZ」にある○○Web サイトにアクセス



○○Web サイト



実体はデータセンターやクラウド上に存在する Web サーバー

IP アドレス：XX.YY.ZZ.XYZ

Web サーバーだけでなく、パソコンやスマートフォンなどのネットワークに接続できる機器には、IP アドレスが割り振られています。

一般的には IPv4 という規格が使われます。IPv4 は次図のような形式です。

## IP (v4) アドレスは 4 つの数字をピリオドで区切って表される

1つの数字は 0 ~ 255 までの  
256 (=2<sup>8</sup>) 個の範囲をとる

192 . 168 . 1 . 1

IPv4 は 4 つの数字が  
ピリオドで区切られた形式



2<sup>8</sup> 個の数字が 4つある組み合わせなので、  
 $2^{8 \times 4} = 2^{32}$  個の IPv4 アドレスが存在する

なお、IPv4 の他に IPv6 という規格も存在します。IPv4 のアドレスが  $2^{32}$  個 (= 約 43 億個) 存在するのに対し、IPv6 のアドレスは  $2^{128}$  個 (= 約 340 潤個) 存在します。

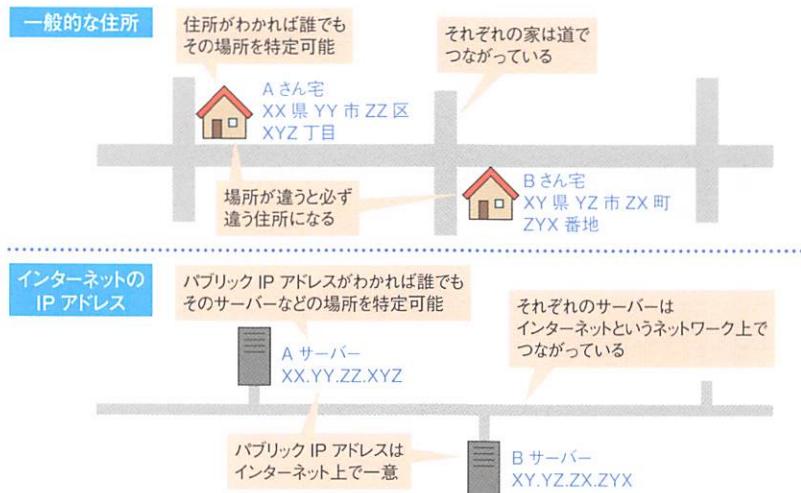
近年、インターネット上において IPv4 だけでは住所を割り振れないほどにサーバーやインターネットに接続する機器が増えてきたため、これに対応できるように IPv6 が注目されています。ただ、IPv6 はまだ普及しきっておらず、現状では IPv4 を使うことがほとんどですので、ここではこういった IP アドレスもあるということを押さえておければ問題ありません。

## パブリック IP アドレスとプライベート IP アドレス

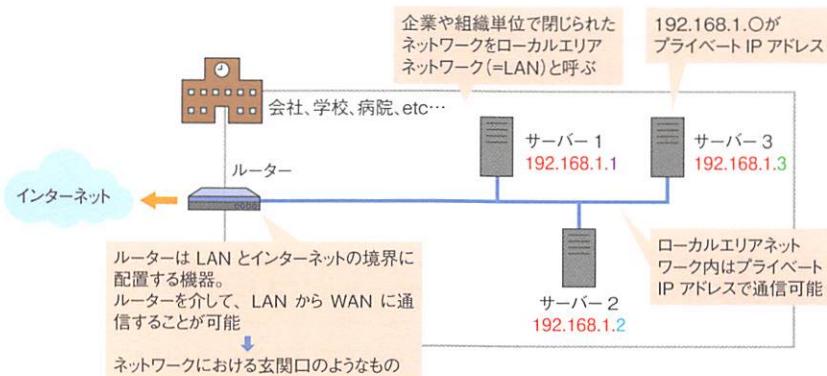
前項では、IP アドレスはネットワーク上の住所を表すとお伝えしました。もう少し詳しくいうと、IP アドレスは大きく 2 種類に分けられます。そのうちのひとつが、日本中あるいは世界中で、「そのアドレスはインターネット上のここ！」と特定することが可能となるアドレスです。それを **パブリック IP アドレス** といいます。別名として、グローバル IP アドレスと呼ばれることもあります。

もうひとつは、**プライベート IP アドレス** といいます。世界中で識別可能なパブリック IP アドレスに対して、プライベート IP アドレスは閉じられたネットワーク (= ローカルエリアネットワーク (LAN)) 内でのみ識別可能な IP アドレスです。

## パブリック IP アドレスは世界中からアクセス可能



## プライベート IP アドレスは LAN 内でのみアクセス可能



プライベート IP アドレスとして利用できるのは、下記に示す範囲です。

- 10.0.0.0 ~ 10.255.255.255 (10.0.0.0/8)
- 172.16.0.0 ~ 172.31.255.255 (172.16.0.0/12)
- 192.168.0.0 ~ 192.168.255.255 (192.168.0.0/16)

この範囲以外をパブリック IP アドレスとして利用可能です。

## CIDR ブロックで IP アドレスの範囲を決める

次図のように、IP アドレスを用いてネットワークの範囲を表すことができます。これを、**CIDR ブロック**と呼びます。

IP アドレスを表す数字列は、大きく **ネットワーク部** と **ホスト部** に分けることができます。ネットワーク部は文字どおりネットワークを表す部分で、ホスト部はそのネットワーク内におけるホスト、つまりネットワーク内で接続可能なサーバーなどを表す部分です。

次図における「/16」の部分は**サブネットマスク**とも呼ばれており、IP アドレスを用いてネットワークの範囲を表す場合はほぼこの表記が使われます。

### IP アドレスの後に / (スラッシュ) と数字でネットワークの範囲を表す

[172.31.0.0/16] は Amazon VPC のデフォルト VPC の CIDR ブロック

この部分が「サブネットマスク」を表す。ここでは 10 進数表記

172 . 31 . 0 . 0 /16

2 進数表記 10101100 . 00011111 . 00000000 . 00000000

サブネットマスク 11111111 . 11111111 . 00000000 . 00000000  
(2 進数表記)

/16 の場合は上位 16 ビットがネットワーク部となる。  
2 進数表記で左から 16 番目までの  
1 になっている部分がそれにあたる

0 になっている箇所がホスト部にある。  
/16 の場合は下位 16 ビットが該当する。  
この場合、アドレス総数は  $2^{16}=65,536$  個となる



このうち、172.31.0.0（ホスト部がすべて 0 のアドレス）は**ネットワークアドレス**、172.31.255.255（ホスト部がすべて 1 のアドレス）は**ブロードキャストアドレス**という特別なアドレスで、ネットワーク内のサーバーなどに割り当てることはできない。どのネットワークにおいてもこの 2つのアドレスが存在するため、割り当て可能なアドレスの個数は、「（ネットワーク内のアドレス総数）-2」個となる

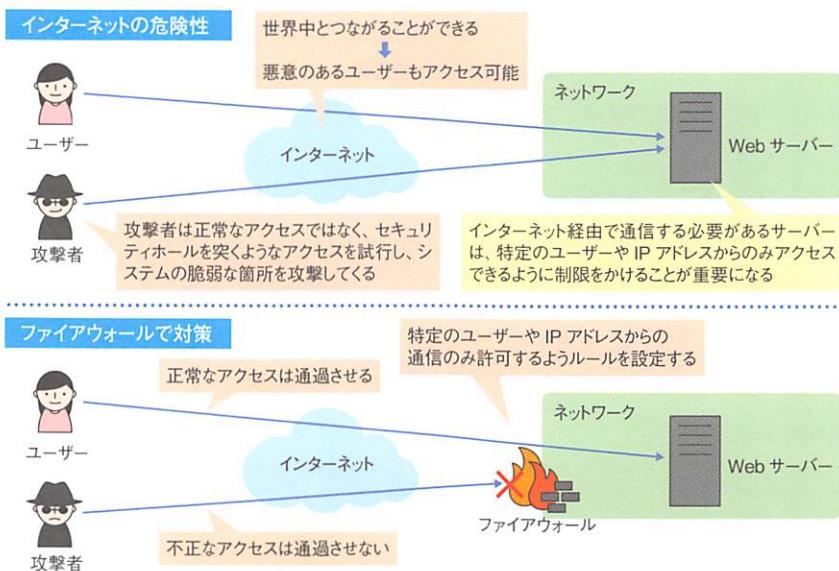
## ファイアウォールで許可された通信だけ通す

私たちは普段、インターネットを経由することで世界中のさまざまな Web サイトやサービスにアクセスできていますが、世界中とつながるということは、悪意のあるユーザー（＝攻撃者）からの不正なアクセスの危険にさらされることもあります。そういった不正なアクセスによるシステムへの侵入を防ぐために用いられるのが**ファイアウォール**です。

ファイアウォールというのは役割名のようなもので、具体的にはサーバーとして設置されてたりソフトウェアとしてインストールされてたりとさまざまです。

たとえば、普段使っているパソコンにインストールされているOSにも標準でファイアウォール機能が付いています。AWSにおいても、セキュリティグループやネットワークACLといったファイアウォール機能を備えたサービスが存在します（どちらも後述します）。

## ファイアウォールで不正な通信、許可のない通信を遮断する



## 負荷分散で複数のサーバーにアクセスを振り分ける

Webサーバーを用いたシステム構成を考えるうえでは、**負荷分散**は欠かせません。Webシステムは閲覧するユーザーが増えれば増えるほど、アクセス数が上昇し、サーバー側に与える処理負荷も増大します。

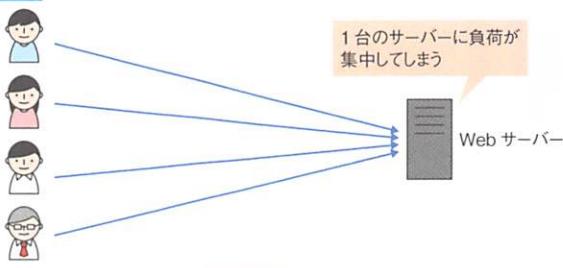
Webサーバーが1台しかない場合、このような負荷がサーバーに集中することでのCPUやメモリなどの使用量が増大し、処理速度の低下や、サーバーがダウン（停止）するリスクがあります。Webシステムが正常に閲覧できなくなってしまい、ユーザーに大きな影響を与えます。

このような事態を避けるためには、Webサーバーを冗長化、つまり複数台にし、負荷分散を行う必要があります。負荷分散をする際には一般的に**ロードバランサー**と呼ばれる負荷分散装置を用います。

## 1台のサーバーにアクセスが集中しないように負荷分散する

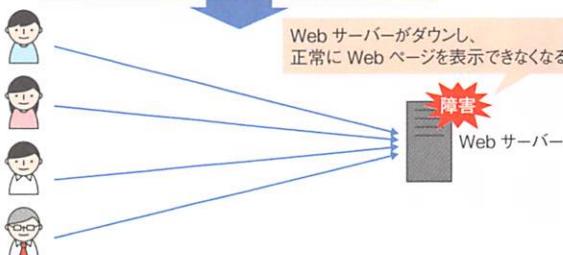
### Webサーバーが1台のみの場合

人気のあるWebサイトほど、閲覧するユーザー数も多い



サーバーが耐えられる負荷を超えると…

ユーザーが必要な情報にアクセスできなくなるため、安定してサービスを提供するためには、継続してWebシステムが稼働できるようにする必要がある



### Webサーバーを冗長化

Webサーバーの前段でアクセスを受け付けて負荷を分散する

Webサーバー1  
Webサーバー2  
Webサーバー3  
Webサーバー4

同じアクセス数の場合、複数台で処理を受け付けるほうが1台あたりの負荷が軽減される

ユーザーは引き続きWebページを閲覧可能

サーバーが1台ダウンした場合



Webサーバー1  
Webサーバー2  
Webサーバー3  
Webサーバー4

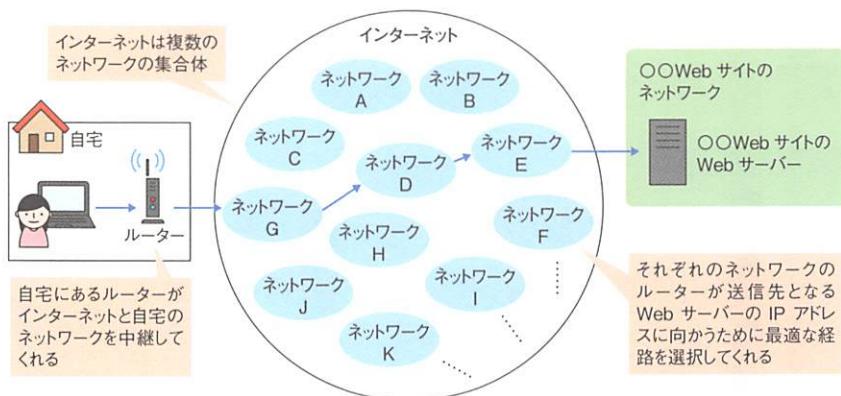
1台のサーバーがダウンしても他のサーバーが稼働し続けるため、システム全体で見ると安定してサービスを提供できているように見える

## ルーティングとルートテーブル

IP アドレスが住所だとすると、その住所に向かうためにはそこまでの道順を知っておく必要があります。ですが、私たちは普段インターネットを使う際にそのような道順を意識しません。代わりに **ルーター** (router) が最適な道順を導き出してくれています。ルーターとは、異なるネットワーク間の境目をつないでくれる中継器です。そのルーターが目的の IP アドレスに向けての道順を決めるのを **ルーティング** (routing) と呼びます。

インターネット経由でアクセスする際には複数のルーターを経由しながら目的のサイトまでたどり着く形になるので、どのようにルーターを経由すれば効率よくたどり着けるかという最適な経路選択が必要になります。

### 目的の IP アドレスに向けての道順を決めるのがルーティング



各ルーターはそれぞれが所有する経路情報をもとに、送信先 IP アドレスに向けて進むべきネットワークを決定しています。この経路情報を **ルーティングテーブル** と呼びます。ルーティングテーブルはルーターが所有するネットワーク上の地図のようなものです。

AWSにおいては、Amazon VPC の **ルートテーブル** と呼ばれる機能がそれにあたります。ルートテーブルでは、インターネットや各 AWS サービスに対してどのエンドポイントを経由するかという経路情報を管理することができます。詳しい内容は次節から解説していきます。

## AWS のルートテーブルの例

送信先 IP アドレス (CIDR ブロック) に対応するターゲット (転送先) を定義する

ルートテーブルにはどのアドレス (CIDR ブロック) と通信するにはどのターゲットにデータを渡すかが書かれています

送信先	ターゲット
172.31.0.0/16	local
0.0.0.0/0	igw-xxxxxxxxxxxxxx

0.0.0.0/0 は全ネットワーク向けを表し、これによって 1 行目の 172.31.0.0/16 以外の通信をここで定義するターゲット (インターネットゲートウェイ) に振り分ける

デフォルト VPC の場合、VPC CIDR アドレスの範囲はすべて local 通信となる

インターネットへ通信する場合はインターネットゲートウェイ (igw) をターゲットとする

## DNS はドメイン名から IP アドレスを調べる仕組み

私たち普段、「https://○○.com」のような URL を用いて Web サイトにアクセスしています。この「○○.com」の部分を **ドメイン** と呼びます。ドメインはドットでつなげた文字列で構成されており、企業名や団体名を表現していることがほとんどです。ドメインはその文字列に表される企業や団体などが提供するサイトの住所を定義するものになります。

このドメイン名に対して IP アドレスが紐付けられていて、その IP アドレスにアクセスしている形になります。しかし、普段私たちはドメイン名にどのような IP アドレスが紐付けられているかを意識することはありません。これは DNS (Domain Name System) が私たちの代わりにドメイン名に紐付く IP アドレスを探し当ててくれているためです。このような DNS の仕組みを **名前解決** といいます。

## DNS はドメイン名に対応する IP アドレスを探す電話帳のようなもの



# Amazon VPC とは

## 02 Amazon VPC で仮想ネットワークを作る

keyword •Amazon VPC •CIDR ブロック •サブネット

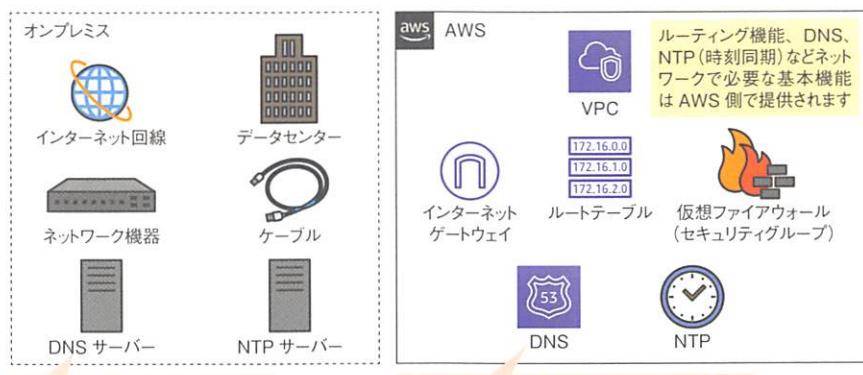
### Amazon VPC は仮想ネットワーク

Amazon Virtual Private Cloud (以下 VPC) は、AWS 上に作成できるプライベート仮想ネットワーク空間です。このネットワーク内に EC2 などの AWS リソースを配置して利用します。

1 つの VPC を論理的なまとまりとして分離でき、複数の VPC 間の接続も可能です。インターネットに公開するパブリックな VPC や、VPN (仮想プライベートネットワーク) などを使用して接続するプライベートな VPC が構築できます。

オンプレミスでネットワーク環境を構築する場合、データセンター、ネットワーク機器やサーバー機器、インターネット回線など用意すべきものが多く、準備期間と初期コストがかかります。一方、AWS で VPC をネットワーク環境として準備する場合、いくつかの操作を行えば数分でネットワークを構築できます。

### Amazon VPC でクラウド上に数分でネットワークが作れる



VPCを作成するときには **CIDR ブロック** (IP アドレスの範囲) を指定し、指定した CIDR ブロックのネットワークを確保します。たとえば 「10.0.0.0/16」と指定した場合は 65,536 個の IP アドレス、「10.0.0.0/28」と指定した場合は 16 個の IP アドレスが使用できます。どうしてそうなるかは、次図のように IP アドレスを 2 進数で表現したもので確認してください。

## CIDR が /16、/28 の場合に使用できる IP アドレスの数

10.0.0.0/16	10.0.0.0/28
00001010 00000000 00000000 00000000 ネットワークアドレス (/16) $2^{16}$ =65,536 個	00001010 00000000 00000000 00000000 ネットワークアドレス (/28) $2^4$ =16 個

前節で「パブリック IP アドレスとプライベート IP アドレス」を紹介しましたが、VPC では通常、**プライベート IP アドレスを指定**します。これは AWS の推奨です。任意のパブリック IP の範囲で CIDR ブロックを指定することができますが、もし外部のパブリック IP と重複した場合に通信できなくなる可能性があるため、基本的にはプライベート IP アドレス空間を CIDR ブロックに指定したほうがよいでしょう。

プライベート IP アドレスの範囲であれば、基本的には自由に指定して OK ですが、オンプレミス環境や他の VPC など、外部のネットワークとの接続を検討している場合は、**接続するネットワークと VPC の CIDR ブロックが重複しないように注意**します。重複すると直接接続できません。

また、CIDR ブロックの指定によって確保できるホストアドレスの数が変わりますが、**ホストアドレスは余裕をもってなるべく多く確保**しておく必要があります。通常、最初から最終的に必要となる IP アドレスの総数を把握することは難しいで

## 接続するネットワークと VPC の CIDR ブロックが重複しないようにする



CIDR ブロックが同じ場合は直接接続できない

以下から /16 以下の範囲で指定を推奨  
 • 10.0.0.0 ~ 10.255.255.255 (10.0.0.0/8)  
 • 172.16.0.0 ~ 172.31.255.255 (172.16.0.0/12)  
 • 192.168.0.0 ~ 192.168.255.255 (192.168.0.0/16)

す。後々リソースの拡張のために IP アドレスが追加で必要になる場合もあります。また、AWS が自動的に使用する IP アドレスもあり、VPC に設定した CIDR ブロックの IP アドレスがすべて使えるわけではないためです。

## VPC とサブネットの作成

VPC は、EC2 や他の AWS サービスと同じく、AWS マネジメントコンソール (GUI) または API を使用してコマンドやプログラム経由で作成が可能です。マネジメントコンソールから作成する場合は、画面から以下の項目を指定します。

- VPC 名 (Name タグ)
- CIDR ブロック
- IPv6 の設定
- テナンシー (専用ハードウェアを使用するかどうか)

EC2 よりも設定項目は少なく、すぐに仮想ネットワークを作成できます。IPv6 の設定はデフォルトで無効であり、必要に応じて設定を行います。テナンシーは、ライセンスやセキュリティ要件でハードウェアを専有したい場合のみ、「専有オプション」を指定します。

## VPC はマネジメントコンソールから簡単に作成できる

**VPC を作成** 情報

VPC は、Amazon EC2 インスタンスなどの AWS オブジェクトによって使用される AWS クラウドの分離された部分です。

**VPC の設定**

名前タグ - オプション  
「Name」というキーと、指定した値を使用してタグを作成します。

my-vpc

IPv4 CIDR ブロック 情報  
10.0.0.0/16

IPv6 CIDR ブロック 情報  
 IPv6 CIDR ブロックなし  
 Amazon 提供の IPv6 CIDR ブロック  
 IPv6 CIDR 所有 (ユーザー所有)

IPv6、テナンシーは必要に応じて設定する  
(通常あまり使用しない)

テナント 情報

デフォルト

なお、VPCだけでは EC2などのリソースをネットワーク内に作成できません。VPCの中に、さらに細かいネットワークの単位であるサブネットを作成する必要があります。

あります。

サブネットは、1つのAZに所属する必要があり、複数のAZにまたがることはできません。つまり、複数AZにリソースを配置して可用性を高めたい場合は、サブネットも併せて複数作成する必要があります。

サブネットにもCIDRブロックを指定しますが、作成するVPCのCIDRブロックの範囲内でCIDRブロックを指定する必要があります。たとえばVPCのCIDRブロックが10.0.0.0/16である場合、10.0.0.0/24などのCIDRブロックを指定します。

## VPCの中にサブネットを作成して、そこにEC2などを作成していく



# 03 Amazon VPC の主要機能の使い方

**keyword**

- ルートテーブル
- インターネットゲートウェイ
- パブリックサブネット
- プライベートサブネット
- パブリック IP
- Elastic IP
- NAT ゲートウェイ
- ネットワーク ACL

## VPC に経路情報を設定してインターネットと通信する

前節で説明したサブネットのように、VPC 内には複数の機能が存在し、それらの機能が相互に連携を行います。主要な機能を紹介しながら、VPC の使い方について説明します。

### ○ルートテーブル

ルートテーブルは、ネットワークの経路情報です。オンプレミス環境ではルーターなどのネットワーク機器にルーティングテーブルを設定しますが、AWS では VPC 内にルートテーブルを作成し、サブネットごとに使用するルートテーブルを指定します。VPC を流れるパケットは、このルートテーブルの情報をもとに経路（どこに通信するか）を決定します。VPC 作成時にデフォルトで 1 つのルートテーブルが作成されます。デフォルト状態では VPC 内の経路情報しかないとため、VPC 外へ通信はできません。

たとえば送信先 0.0.0.0/0（全ネットワーク）に対し、次項で説明するインターネットゲートウェイをルートとして登録すると、インターネットゲートウェイを経由してインターネットへ通信できるようになります。

## サブネットごとにルートテーブルを作成して通信経路を設定する



## ○インターネットゲートウェイ

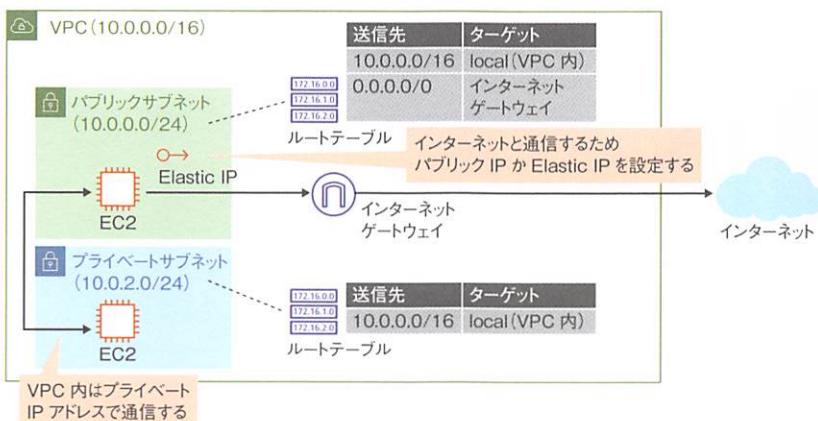
インターネットゲートウェイは、サブネット内にある EC2 などのリソースが、インターネットと通信できるようにするために使用する機能です。インターネットゲートウェイを作成し、サブネットのルートテーブルに設定することで、インターネットから VPC 内へ通信、および VPC 内からインターネットへ通信できるようになります。

インターネットゲートウェイへのルートが設定されたサブネットを **パブリックサブネット** と呼びます。逆に、インターネットゲートウェイ経由でインターネットと通信できないサブネットを **プライベートサブネット** と呼びます。

EC2 の場合は、**パブリック IP** または **Elastic IP** を付与することで、インターネットと EC2 が通信できるようになります。パブリック IP、Elastic IP は **いずれも EC2 に設定できるパブリック IP（グローバル IP）アドレス** です。パブリック IP は自動設定される IP アドレスで、再起動の都度変更になります。Elastic IP は静的 IP アドレスとも呼ばれ、利用者が確保して永続的に使用できる IP アドレスです。ファイアウォールなどで IP アドレスを固定して通信を許可する場合は Elastic IP を使用します。

パブリック IP は無料で使用できますが、Elastic IP は未使用の場合はわずかに料金が発生します (0.005USD / 1 時間)。起動中の EC2 などで使用している場合は無料となります。

## インターネットに公開するにはインターネットゲートウェイを使用



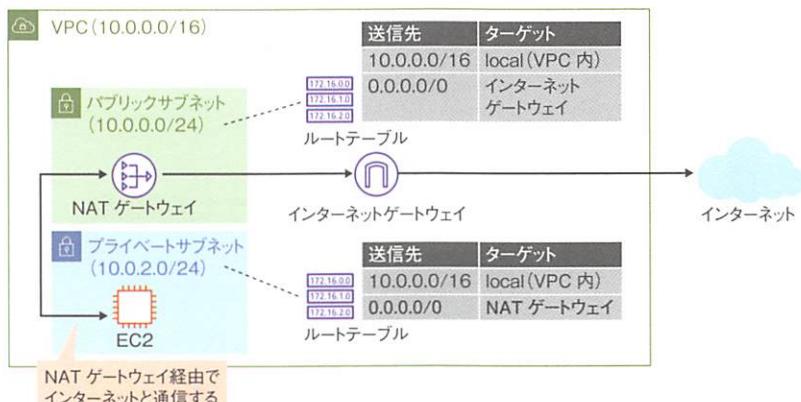
## ○ NAT ゲートウェイ

プライベートサブネットにある EC2 などのリソースは、通常インターネットと疎通ができません。ただし、インターネット上にあるソフトウェアパッケージのダウンロードなど、VPC 内からインターネットへ通信を行いたい場合があります。このために使用できるのが **NAT ゲートウェイ**です。インターネットゲートウェイとは異なり、インターネットから VPC 内へは通信できないため、通信が一方通行となり、より安全に AWS 外部と通信が行えます。

NAT とは **Network Address Translation** の略で、**プライベート IP アドレスをパブリック IP アドレスに変換すること**を意味します。NAT ゲートウェイでは、プライベートサブネットの IP アドレスを NAT ゲートウェイのパブリック IP アドレスに変換することでインターネットと通信が可能になります。

なお、NAT ゲートウェイを使用して通信を行うためには、外部通信を行うサブネットのルートテーブルに経路情報（ルート）を登録する必要があります。

### 内部からインターネットを使いたいときは NAT ゲートウェイを使用



## VPC のアクセス制御と通信ログ確認

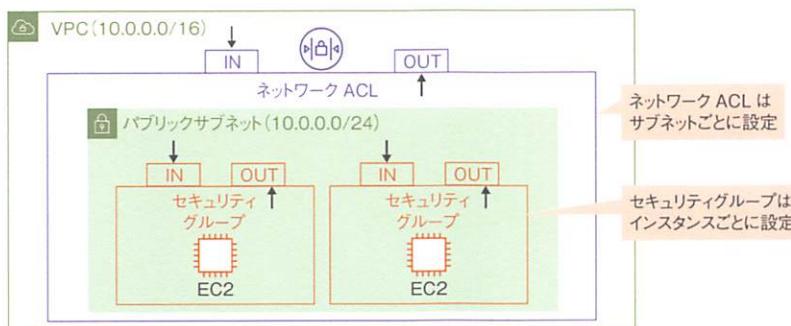
VPC には、サブネット単位でアクセス制御を設定できる **ネットワーク ACL** という機能があります。3-3 節 (p.64) でも紹介した **セキュリティグループ** と組み合わせて、アクセス制御の設定が可能です。セキュリティグループとの比較は次表のとおりです。

### ネットワーク ACL とセキュリティグループの比較

	ネットワークACL	セキュリティグループ
設定単位	サブネット単位で設定	インスタンス(リソース)単位で設定
許可／拒否の設定	ルールの許可と拒否が可能	ルールの許可のみ可能 (許可しないものはすべて拒否)
ステートレス／フル	ステートレス (行き戻り両方の許可が必要)	ステートフル (行きのみの許可でOK)
ルールの優先順位	番号を指定して優先順位を決定	すべてのルールを評価

デフォルト状態では、ネットワーク ACL ではすべての通信が許可されます。サブネット全体でネットワークアクセスの許可や拒否を行いたい場合は、セキュリティグループで追加の設定を行うことで、より安全なネットワークを構築できます。

### 2つのアクセス制御機能で安全なネットワークを構築する



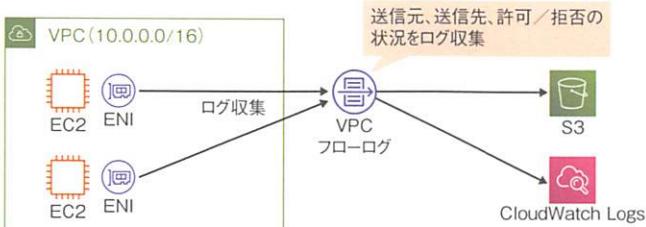
ネットワーク ACL やセキュリティグループで許可された通信や、拒否された通信の状況は、**VPC フローログ** という VPC 内の IP トラフィック状況をログとして保存

できる機能を使用して確認できます。

保存先は、AWS の監視サービスである CloudWatch Logs (p.246 で紹介) か、S3 を指定できます。CloudWatch Logs に保存した場合は、ログ内容に応じてメール通知を飛ばすといった監視も可能です。料金は S3 のほうが安いです。

EC2 などの VPC 内のリソースは、IP アドレスごとにネットワークインターフェイス (ENI、Elastic Network Interface) を持ち、ログはネットワークインターフェイス単位で出力されます。EC2 だけでなく、Elastic Load Balancing や RDS、Redshift など、VPC 上で稼働するサービスのログがすべて出力されます。送信先／送信元の IP アドレスやポート、送信の許可／拒否などの情報が含まれます。

## VPC 内のリソースの通信ログを保存しておく



# 04 VPC 同士や外部サービス、オンプレミスとの接続

**keyword**

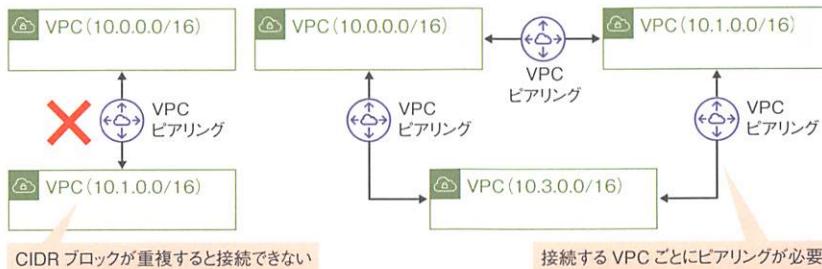
- VPC ピアリング • AWS Transit Gateway • VPC エンドポイント • AWS PrivateLink
- AWS Site-to-Site VPN • AWS Client VPN • AWS Direct Connect

## VPC と外部ネットワークを接続する

他の VPC や、VPC 外の AWS サービスに接続を行う機能を紹介します。1 つ目は **VPC ピアリング**です。

VPC ピアリングを使用して、異なる 2 つの VPC を接続して互いに通信できるようにします。ピアリングは 2 つの VPC 間で行うため、たとえば 3 つの VPC 間で通信を行う場合はそれぞれピアリングの設定を行う必要があります。接続する VPC の CIDR ブロックは重複できないため注意が必要です。

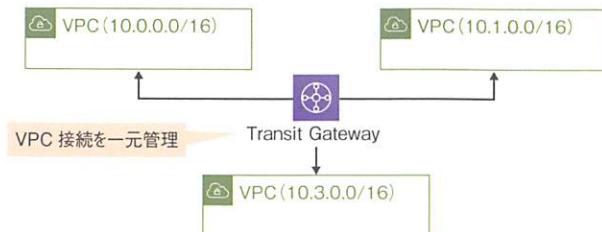
### VPC ピアリングで異なる VPC 間で通信できる



2 つ目は **AWS Transit Gateway** (以下 **Transit Gateway**) です。Transit Gateway を使用すると、VPC の接続を 1 カ所で中央ハブとして管理できます。VPN などオンプレミス環境との接続にも使用できるため、**AWS のネットワーク接続を一元管理**できます。

料金は VPC ピアリングを使用するよりも高くなるため注意が必要です。とはいえ、より多くの VPC を接続する場合は Transit Gateway を使用して一元管理をしたほうがよいでしょう。

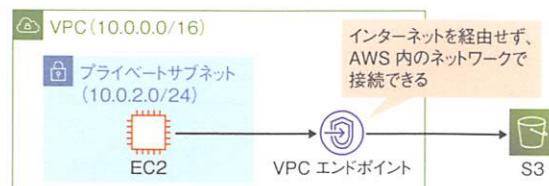
## VPC 間の接続が多くなってきたら Transit Gateway で一元管理



## VPC と VPC 外部の AWS サービスを接続する

S3 など、VPC 外部で動作する AWS サービスは、通常インターネット経由で通信を行います。そうではなく、VPC 内のサービスと S3 を AWS 内のプライベートなネットワーク経由で接続するために **VPC エンドポイント** が使用できます。VPC エンドポイントを設定した VPC は、VPC エンドポイントを経由して S3 などの VPC 外のサービスへアクセスできます。

## S3 など VPC 外のサービスとインターネットを経由せずに接続する

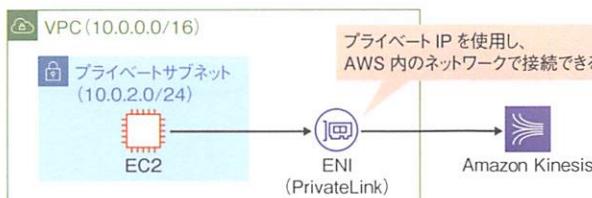


S3 と DynamoDB で使用する VPC エンドポイントは **ゲートウェイエンドポイント** と呼ばれ、AWS 内の通信にはなりますがパブリック IP を指定して通信を行います。

それ以外のサービスのエンドポイントは **インターフェイスエンドポイント** と呼ばれます。こちらは **AWS PrivateLink** という機能を使用して、サブネットにサービス接続用の ENI (ネットワークインターフェイス) を作成してプライベート IP で通信を行います。

ゲートウェイエンドポイントは S3 と DynamoDB のみですが、インターフェイスエンドポイントは多くの AWS サービスに対応しています。

## Private Link は多くのサービスとプライベート IP アドレスを使用して接続する



2種類のエンドポイントが出てきて少し混乱しますが、いずれも VPC 外の AWS サービスとプライベート（AWS 内）で通信を行うための機能です。

## VPC とオンプレミスのネットワークを接続する

VPC は AWS 内のネットワークやサービスだけでなく、オンプレミスのネットワークとも接続が可能です。いくつか方法を紹介します。

まずは、AWS Site-to-Site VPN（以下 Site-to-Site VPN）です。オンプレミス環境のネットワークと VPC 間で VPN 接続を行う機能です。VPN とは、Virtual Private Network の略で、仮想的なプライベートネットワークを使用して通信する技術です。一度 VPN 接続すると、外部のネットワークとプライベート IP アドレスで通信ができます。

オンプレミス環境で用意したルーター機器に AWS 接続用の設定を行い、AWS 側も接続の設定を行うことでプライベートに接続が行えます。プライベートと言いつつも、通信はインターネットを経由します。IPsec VPN という技術を使用し、インターネット上に仮想的な専用ネットワークを作る形になります。

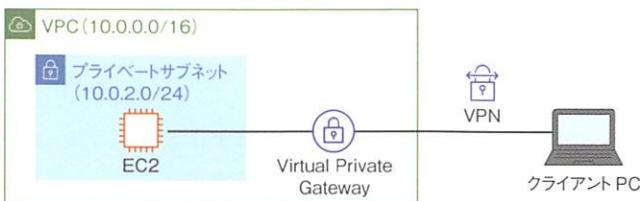
## Site-to-Site VPN で AWS とオンプレミス環境を接続



オンプレミス環境のクライアント端末（パソコン）と VPN 接続を行う、AWS

**Client VPN (以下 Client VPN)** という機能もあります。オンプレミス環境全体ではなく、特定の端末とプライベートに接続したい場合はこちらがおすすめです。

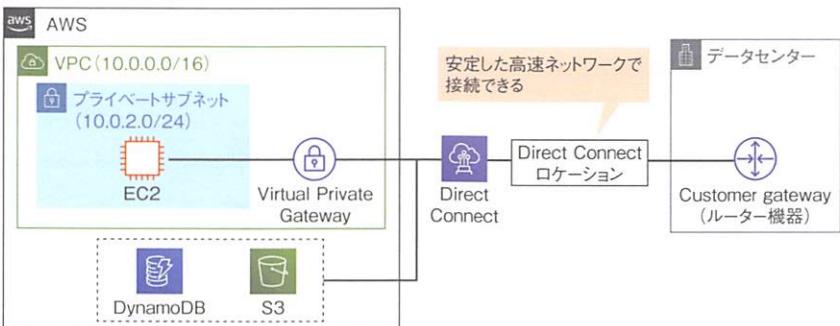
### Client VPN はオンプレミスの特定の端末とだけ接続



その他、AWS とオンプレミス環境を専用線で接続するための機能として、**AWS Direct Connect (以下 Direct Connect)** があります。VPC 以外の AWS サービスとも専用線接続が可能です。AWS のデータセンターと直接接続するわけではなく、Direct Connect ロケーションと呼ばれる既存の接続ポイントを経由して接続します。

VPN 接続と比べ、高速かつ高品質なネットワークを使用できます。ただし、利用コストは高くなり、サービスの利用を申請してから利用できるまでの準備期間も長くなります。

### Direct Connect は高速で高品質な専用線接続サービス。コストは高い



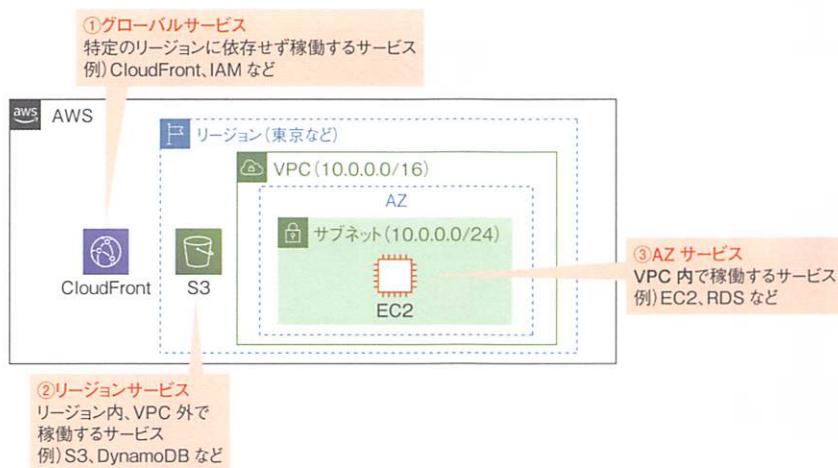
# 05 他の AWS サービスと組み合わせた構成例

**keyword** • AZ サービス • リージョンサービス • グローバルサービス

## VPC の構成例

VPC の主要な関連機能を理解できたところで、他の AWS サービスとも組み合わせた構成例をいくつか紹介します。構成例を見る前に、**AWS のサービスは稼働する場所によって次図のとおり 3 種類に分けられる**ことを理解しておきましょう。

### AWS サービスは稼働する場所によって 3 種類に分けられる



AZ サービスは VPC 内、その他のサービスは VPC 外で稼働します。それでは、具体的な構成例を見ていきましょう。

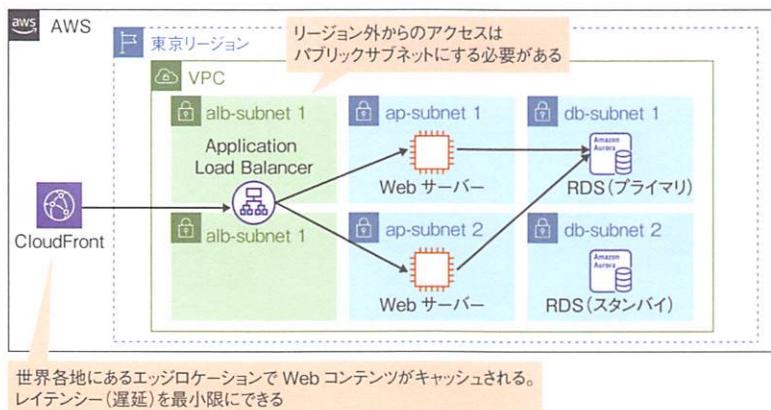
## 構成例 1 : Web + DB + Amazon CloudFront

オンプレミス環境の Web サーバー、DB サーバー環境を AWS へ移行する際によく見られる構成です。

1 つのリージョン（東京）にプライベートなネットワークとして VPC を作成し、その中に Web サーバー（EC2）や DB サーバー（RDS）といったリソースを配置します。サーバーの役割ごとにサブネットを作成し、マルチ AZ 対応（可用性向上）のため役割ごとに 2 つのサブネットを用意しています。VPC 外部から通信を受ける Application Load Balancer（ALB）のみパブリックサブネットに配置しています。他の AWS サービスを VPC 内で利用する場合も、**役割ごとに 2 つまたは 3 つのサブネットを用意する**というパターンが多いです。

また、VPC 外に CloudFront（5-8 節（p.156）で紹介）を利用してすることで、Web コンテンツをキャッシュして利用者がコンテンツを早く表示できるよう工夫しています。

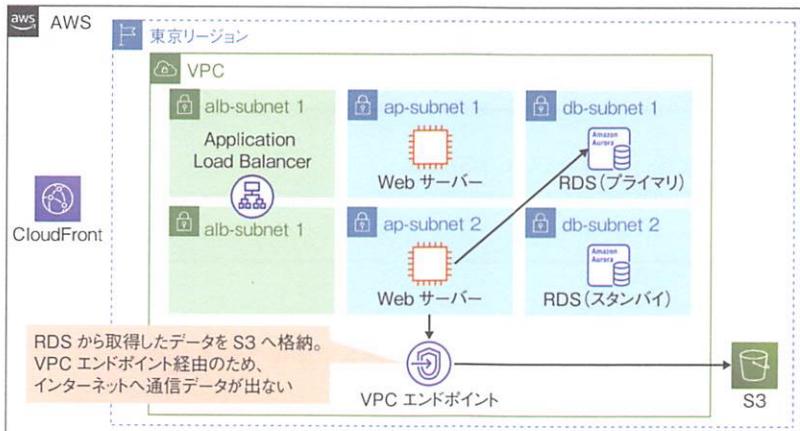
### オンプレミスの Web サーバー、DB サーバー移行時によく見られる構成



## 構成例 2 : VPC エンドポイントを使用した S3 接続

構成例 1 に、VPC エンドポイントと S3 バケットを追加しました。データベース（RDS）から取得したデータを S3 に出力したり、S3 から取得したデータをデータベースにインポートしたりする処理を想定しています。

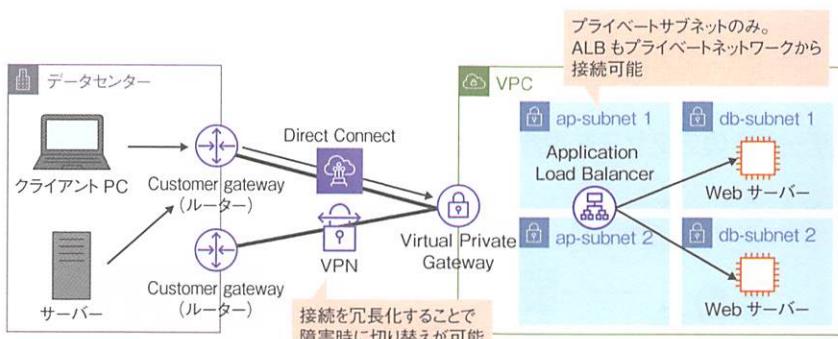
## 1つ前の構成にデータの保存先としてS3を追加



### 構成例 3：オンプレミス環境からプライベート接続

Direct Connect と Site-to-Site VPN の両方を用いたオンプレミス環境とのプライベート接続です。オンプレミス環境専用のプライベートネットワークとなります。VPC とオンプレミス環境のみ通信が可能で、インターネットと疎通をしない（できない）構成にしているのが大きなポイントです。

### AWS をオンプレミス環境専用のプライベートネットワークに利用



# 06

ELB (Elastic Load Balancing) とは

## ELB で負荷分散して可用性を高める

keyword • ELB • ALB • NLB • CLB • GWLB

### ELB とは

Elastic Load Balancing (以下 ELB) は、AWS で提供されるロードバランサー サービスです。ロードバランサーの主な機能は、アプリケーションへのトラフィック（ネットワークを流れるデータ）の負荷分散を行うことです。また、それ以外にもトラフィックを送る先のサーバー（EC2 などのインスタンス）が止まらずに動いているか定期的に確認し、可用性の向上を図ることもできます。

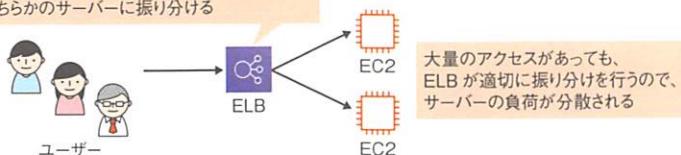
ここでは、ELB が提供する基本的な機能について紹介します。

#### ○負荷分散

トラフィックの転送先として複数のターゲットを指定することで、それらのターゲットにユーザーからのアクセスを自動的に振り分けることができます。こうすることで大量のユーザーからのアクセスが起こった場合でも、複数台のターゲットに負荷を分散できます。また、各ターゲットを違う AZ に配置できるため、地理的に離すことができ、災害発生時の可用性の向上も実現できます。

#### ユーザーからのアクセスを複数台のサーバーに振り分ける

複数の EC2 などを振り分け先(ターゲット)として登録し、ユーザーをどちらかのサーバーに振り分ける

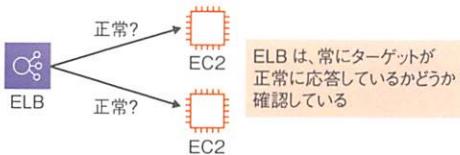


さらに、ELB 自体は負荷状況に応じて自動的にスケールするため、利用者は ELB の性能低下を気にする必要はありません。

## ○ターゲットモニタリング

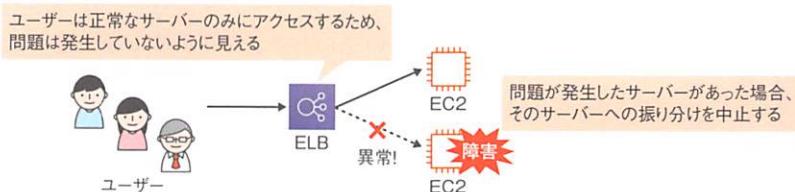
ELB は常にターゲット（トラフィックの転送先）に対する接続を監視し、ターゲットの状態のモニタリングやヘルスチェックを行っています。これにより、リクエスト追跡や CloudWatch メトリクス（p.246 で紹介）の取得を可能としており、利用者はいつでもこれを確認できます。

通信を振り分ける先に異常がないか確認する



モニタリングによって異常な動作を検知した場合は、自動的に対象ターゲットの切り離しを行い、安定稼働を維持できます。

異常を検知したら、そちらには通信を振り分けない



### ○セキュリティ機能

ELB には、セキュリティグループをはじめとした AWS の基本的なセキュリティサービスを適用することができます。ただし、次ページの表で紹介する Network Load Balancer (NLB) はセキュリティグループが設定できません。

また、ELB に SSL/TLS サーバー証明書を設定することで、通信の SSL/TLS 暗号化を実現できます。

## ELB の種類

ELB は、その用途に応じて次表の 4 種類のロードバランサーが提供されています。

## ELB はロードバランサーサービスの総称で、中身は 4 種類ある

名前	説明
Application Load Balancer (ALB)	<ul style="list-style-type: none"> <li>HTTP と HTTPS の負荷分散を行うことができる</li> <li>レイヤー 7 (アプリケーション層) で動作しているため、マイクロサービスやコンテナなど、さまざまなアプリケーションにも対応可能</li> </ul>
Network Load Balancer (NLB)	<ul style="list-style-type: none"> <li>レイヤー 4 (トранSPORT 層) で動作しているため、HTTP/HTTPS 以外の TCP、UDP、TLS の負荷分散も可能</li> <li>数百万リクエストの大規模なトラフィックでも、高速なパフォーマンスを提供可能</li> </ul>
Classic Load Balancer (CLB)	<ul style="list-style-type: none"> <li>ALB、NLB のサービス提供以前から提供されている旧タイプのロードバランサー</li> <li>レイヤー 4 (トランSPORT 層) とレイヤー 7 (アプリケーション層) で動作</li> <li>古いアーキテクチャを利用する必要があるなど、特別なケース以外は、基本的に ALB や NLB の利用が推奨される</li> </ul>
Gateway Load Balancer (GWLB)	<ul style="list-style-type: none"> <li>AWS 上で提供されるサードパーティのセキュリティ製品などのデプロイ、管理を行うことができる</li> <li>レイヤー 3 (ネットワーク層) で動作</li> <li>従来、NLB や VPC ピアリング、Transit Gateway で実現していたアーキテクチャをよりシンプルに実装可能</li> </ul>

基本的には、要件に応じて ALB もしくは NLB のどちらかを選択することになるはずです。一般的に Web サイトや Web システムの負荷分散には ALB、ALB では実現できない細かい制御や HTTP/HTTPS 以外のプロトコルを利用する場合は NLB を選択します。

### ELB の利用料

ELB の利用料は、1 台につき 1 時間あたりのロードバランサー使用料（固定）とロードバランサー キャパシティ ユニット (LCU) 使用料（変動）を合算したものとなります。

LCU は、1 秒あたりの新たな接続数や 1 分あたりのアクティブ接続数など、いくつかの要素から使用料が計算され、その中から最も使用料の高いものが請求されます。課金される要素は ELB の種類ごとに変わってきます。ただ、多くの場合、請求額の大半は時間あたりの使用料が占めることとなり、LCU を意識することは多くありません。アクセス数が膨大となる大規模なシステムや、巨大なデータのやりとりが行われるような場合は、LCU コストについても考慮する必要があります。

# 07 高性能で手軽に使える DNS サービス

**keyword** • Amazon Route 53 • DNS レコード • ルーティングポリシー • ヘルスチェック

## Route 53 とは

Amazon Route 53 (以下 Route 53) は、AWS の提供する DNS サービスです。フルマネージドで提供されているため、オンプレミスで構築される DNS サービスのようなサーバー自体の維持運用や作業は不要となり、すべて AWS マネジメントコンソールから設定、管理を行えます。グローバルサービスであるため、すべてのリージョンで共通して利用可能です。

管理はホストゾーン単位で行われます。ホストゾーンとは、ドメインとそのサブドメインのトラフィックルーティング（後述）の方法についての情報を保持する箱です。ドメイン名を登録したときに、同名のホストゾーンが自動生成されます。

### 登録したドメインごとの設定 = ホストゾーン

Route 53

ホストゾーン「xxx.com」

- ドメイン : xxx.com
- ルーティング方法 : シンプル

ホストゾーン「yyy.com」

- ドメイン : yyy.com
- サブドメイン : sub.yyy.com
- ルーティング方法 : シンプル

登録したドメインごとにホストゾーンが  
作成され、いろいろな設定が管理される

Route 53 の大きな特徴は以下です。

#### • 高い可用性

Route 53 の SLA (サービス品質保証) は 100% と定義されており、他の AWS サービスと比べても高い可用性が実現されています。これは、Route 53 のサービスを提供する基盤システムが全世界に冗長化されていることに由来します。

### • 高いコストパフォーマンス

主な提供サービスであるドメイン登録は、最初の 25 のホストゾーンは 1 つにつき 0.5USD／月程度で利用できます。その他の機能でもトラフィック量による課金は発生しますが、利用料に占める割合は小さいため、通常はあまり意識する必要はありません。

## Route 53 の機能

Route 53 では、大きく 3 種類の機能を利用することができます。各機能の詳細は、この後に紹介していきます。

- ドメイン登録機能
- ドメインへのトラフィックルーティング
- リソースのヘルスチェック

## ドメイン登録機能

Route 53 の中でも主要なサービスとなるのが、任意のホストゾーンを作成し、**ドメイン登録**を行う機能です。「xxx.com」のような任意のドメイン名を、Route 53 上の**DNS レコード**として登録できます。登録可能なドメインは一部制約があるので注意が必要です。

DNS レコードとは、その DNS が管理しているドメインと、どの IP アドレスやその他の情報が紐付いているかが記載してあるデータのことです。DNS はこの DNS レコードに基づいて問い合わせ元に IP アドレスを返答します。

## DNS レコードを DNS サーバーに設定して名前解決を可能にする



AWSでの新規ドメイン登録は有料となります。ドメインは「.com」や「.jp」などTLD（トップレベルドメイン）を選択する必要がありますが、このTLDの種類に応じて登録料などのドメイン利用料金が変化します。登録する際は必ずAWS公式ドキュメントを確認してください。

- ▶ Amazon Route 53 料金表 - 「ドメイン名」から「TLD別の最新料金表」を参照  
<https://aws.amazon.com/jp/route53/pricing/>

Route 53で作成可能な一般的なDNSレコードのタイプを次表に示します。対象のドメイン名を使って、これらのレコードタイプの中から目的に合ったレコードを作成していくことになります。

## Route 53で作成可能な、一般的なDNSレコードのタイプ

レコードタイプ	説明
Aレコード	IPv4アドレス用のレコード
AAAAレコード	IPv6アドレス用のレコード
CAAレコード	ドメインまたはサブドメインの証明書認証機関(CA)を指定できるレコード
CNAMEレコード	あるホスト名に対する別名を定義するレコード
DSレコード	サブドメインでDNSSECを利用する際に使うレコード
MXレコード	該当ドメイン宛てのメールサーバーのホスト名を定義するレコード
NAPTRレコード	動的委任発見システム(DDDS)アプリケーションで使用されるレコード
NSレコード	ホストゾーンのネームサーバーのサーバー名を定義するレコード
PTRレコード	IPアドレスを対応するドメイン名にマッピングするレコード
SOAレコード	Start of Authority(SOA)レコード。 ドメインおよび対応するRoute 53のホストゾーンを定義する
SPFレコード	メールの送信者の身元を確認するために利用されるレコード ※現在はSPFレコードの代わりにTXTレコードの作成が推奨される
SRVレコード	ホスト名に加えて負荷分散優先順位、重み、サービスポート番号の3種類の情報を付与できるレコード
TXTレコード	ホスト名に関連付けるテキスト情報を定義するレコード
エイリアスレコード	AWSアカウント内の特定のサービスで作成されたリソースやエンドポイントを登録するレコード。AWSのみで利用可能な特殊なレコード

次図がDNSレコード登録画面です。レコード名や名前解決される値の入力に加えて、レコードタイプなどを選択していくことで簡単にレコードを作成できます。ルーティングポリシーについては、次項で詳しく解説します。



## ドメインへのトラフィックルーティング

Route 53 では、作成した DNS レコードごとにルーティングポリシーを設定することができます。ここでいうルーティングは、ルーターの経路選択とは違い、ドメイン名に対する IP アドレスをどのように返すか、という意味です。ルーティングポリシーの設定によって、名前解決（ドメイン名に対応する IP アドレスを探すこと）の際のルーティングの挙動を細かく制御できます。

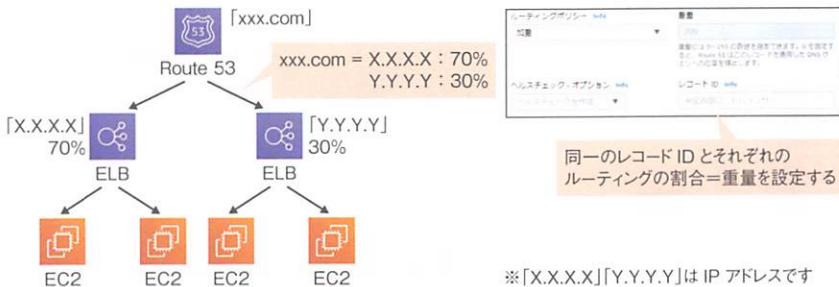
### ○シンプルルーティング

一般的な DNS と同様のルーティングを行います。**1つのドメインに対して1つのIPアドレスのみ**が紐付きます。



### ○加重ルーティング

ルーティング先を複数登録し、それぞれにトラフィックをどの程度割り振るかを**0～255の値 (= 重量)**として指定します。Route 53 は重量によって、どの IP アドレスを返すか決定します。



## ●位置情報ルーティング

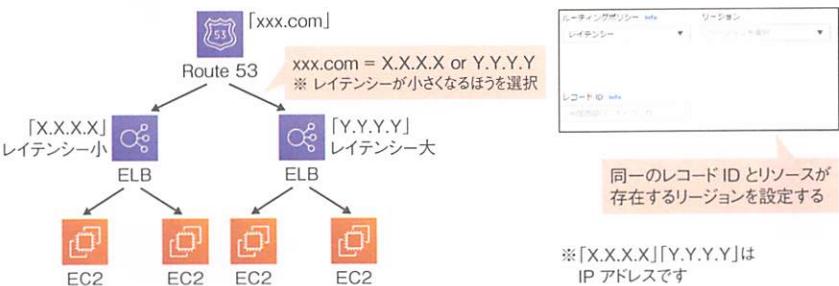
問い合わせ元の位置情報によって、どの IP アドレスを返すか決定します。

例：東京から「xxx.com」へのアクセスなので、「X.X.X.X」を返す



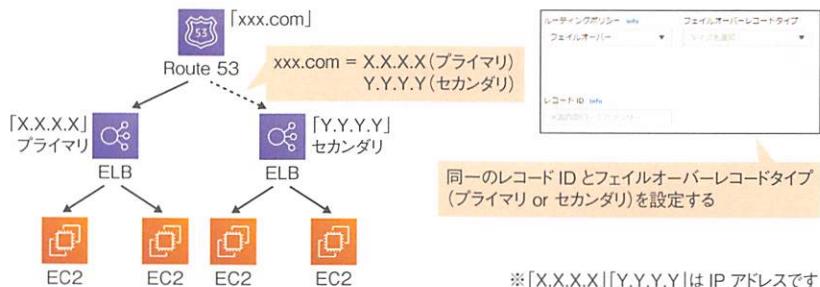
## ●レイテンシールーティング

常にレイテンシー（データ処理の遅延時間）が最小となるリソースの IP アドレスを優先的に返します。



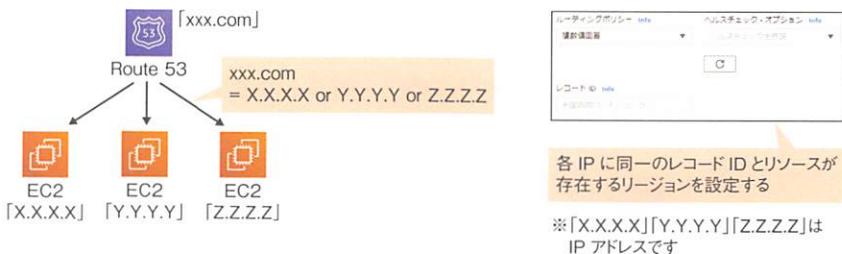
## ○フェイルオーバールーティング

通常はプライマリの IP ヘルテイントしますが、**プライマリに障害が発生した場合はセカンダリヘルテイント**を行います。障害発生の有無は、ヘルスチェック機能を利用して判断されます。



## ○複数値回答ルーティング

複数値回答を設定することで、Route 53 が問い合わせに対して、常に複数の IP アドレスをランダムに返すようにできます。これにより、ユーザーはいくつかのサーバーにランダムに割り振られることになり、負荷分散として機能します。ヘルスチェックが失敗しているサーバーへの割り振りは行われないので、ELB のように機能させることができます。



## リソースのヘルスチェック

これまで何度も紹介しましたが、Route 53 には**ヘルスチェック機能**が存在します。これをを利用して、対象が正常に稼働しているかどうかを確認できます。

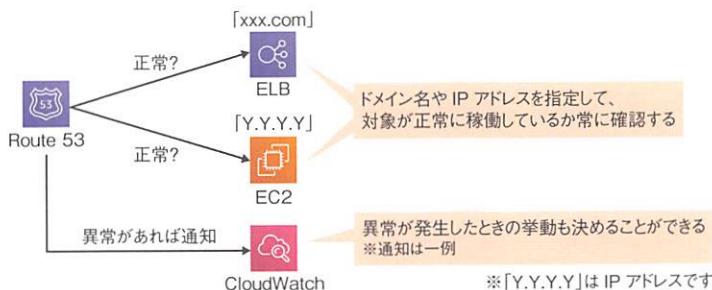
ヘルスチェック対象の正常性判断の基準として、ルーティング先となる Web サー

バーやメールサーバーからの応答結果に加えて、他のサービスのヘルスチェックやCloudWatchアラームを選択できます。

そうして作成したヘルスチェックをもとに、自動でのルーティング変更や失敗時の通知発報などの細かな動作を設定ていきます。

ヘルスチェック対象	説明
エンドポイント	<ul style="list-style-type: none"> <li>エンドポイントとして、IPアドレスもしくはドメイン名を指定し、その応答で稼働を確認する</li> <li>対象エンドポイント、プロトコル、ポート、対象パスを指定する</li> <li>高度な設定項目として、リクエスト間隔やヘルスチェック元のリレーションを選択できる</li> </ul>
他のヘルスチェックのステータス (算出されたヘルスチェック)	<ul style="list-style-type: none"> <li>指定した他のヘルスチェックの結果から正常性を判断する</li> <li>複数のヘルスチェックを対象に、その正常数を基準とする</li> </ul>
CloudWatchアラームの状態	<ul style="list-style-type: none"> <li>作成済みのCloudWatchアラームの状態を確認し、正常性を判断する</li> </ul>

### 対象 IP アドレス（またはドメイン名）に対して高度なヘルスチェックが可能



ヘルスチェックは1つにつき、AWSエンドポイントは0.50USD／月、AWS外部のエンドポイントは0.75USD／月の料金が発生します。また、高度な設定1項目につき1.00USD／月かかるため、必要となるヘルスチェックは要件をよく検討しておかないと、思いがけないコストが発生する可能性があります。

# 08 CloudFront で高速かつ落ちないネット配信を実現

**keyword** • Amazon CloudFront

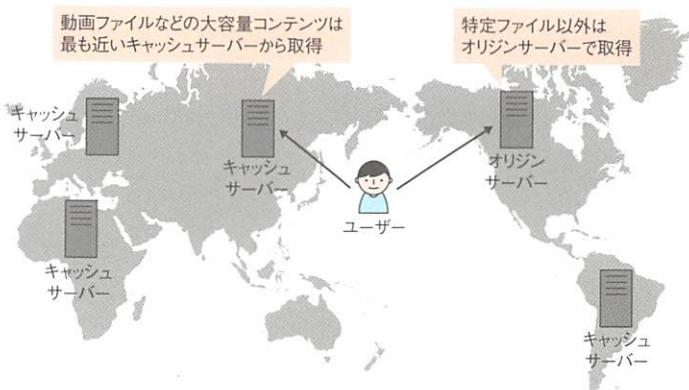
## CloudFront とは

Amazon CloudFront（以下 CloudFront）は、AWS の提供するコンテンツデリバリーネットワーク（CDN）サービスです。CDN とは、動画ファイルをはじめとした大容量デジタルコンテンツをインターネット上で効率的にユーザーに配信するためのネットワークを指します。

CDN はデータ本体を格納しているサーバーを **オリジンサーバー** として、その元データのキャッシュ（コピー）を世界中に存在するキャッシュ専用のサーバー、**キャッシュサーバー** に保存します。CDN を導入している場合、該当データにアクセスするユーザーは、自動的に自分に一番近いキャッシュサーバーのデータを取得するようになるため、取得が高速化されます。考え方は、Route 53 のルーティングポリシーのひとつであった位置情報ルーティングと同じです。

CloudFront はキャッシュサーバーを世界中のエッジロケーションに配置し、AWS 保有のネットワークを通して AWS リージョンへ機能を提供しています。エッ

## CDN の基本的な仕組み。大容量データの効率的な配信に役立つ



ジロケーションは、リージョン別の中間層を含めると 47 カ国 90 以上の都市に 275 以上存在し、世界規模の CDN が実現されています。

## CloudFront 利用のメリット

CloudFront の主なメリットとして、以下が挙げられます。

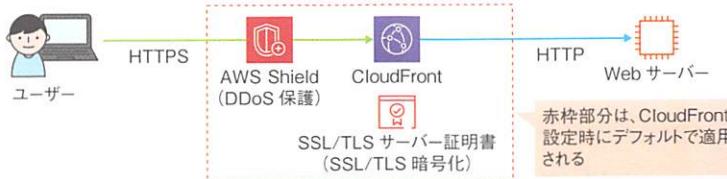
- 大容量コンテンツの高速配信

前項で説明した CDN の仕組みによって、CloudFront は AWS のグローバルインフラストラクチャとそれによる高速処理を活用できるため、動画やオンラインゲームなどの大容量コンテンツを効率的に全世界のユーザーへ配信できます。

- セキュリティの向上

CloudFront を導入することで、導入したサービスは自動的に通信の SSL/TLS 暗号化が行われます。CloudFront はデフォルトで自動生成されたドメイン名が割り当てられており、そのドメイン名に対応した証明書が自動で設定されます。また、AWS Shield (p.229 で紹介) という DDoS 保護サービスも無料かつ自動で適用され、DDoS 攻撃への対策も行えます。

上記のセキュリティ対策は CloudFront が追加料金なし、かつ自動で実施するため、CloudFront を導入するだけで簡単にセキュリティの向上を図ることができます。



- 可用性の向上

CloudFront は全世界にエッジロケーションを持ち、対象データのキャッシュをそこに保存しています。コンテンツ配信がエッジロケーションで行われることになるため、オリジンサーバーへの負荷が分散され、結果的に可用性向上につながります。

さらにオリジンサーバーでの障害発生時の動作も設定できるため、より柔軟なサービス提供を実現できます。

## CloudFront の設定

それでは、実際の CloudFront のマネジメントコンソールにてどのような機能が利用できるか確認していきます。

### ○ディストリビューション

CloudFront では、オリジンサーバーに配置したファイルに関する情報を **ディストリビューション** という単位で扱います。EC2 でいうインスタンスのようなイメージです。ディストリビューション 1つに対して 1つの CloudFront ドメインが割り当てられ、そこにさまざまな設定を行っていきます。ユーザーをこの専用ドメインにアクセスさせることによって、エッジロケーション経由の通信を実現させます。

ディストリビューションの「一般」設定では、アクセスログ出力設定をはじめとしたディストリビューションの基本的な設定を行います。

特筆すべきは **AWS WAF** (7-6 節 (p.224) で紹介) の設定です。AWS Shield と違って追加で利用料金がかかりますが、CloudFront の前段として AWS WAF という AWS マネージドの Web アプリケーションファイアウォールを設定し、IP 制限や脆弱性対策を行うことで、よりセキュリティの向上を図ることができます。



## ○「オリジン」設定

「オリジン」設定では、コンテンツの元データが格納されるオリジンとなるリソースについて設定を行います。ドメイン単位での設定となり、AWS リソースは S3 や ELBなどを対象とできます。自分で管理しているドメインも指定可能です。CloudFront がオリジンにアクセスする際に付与するヘッダー情報も設定できます。



## ○その他の設定

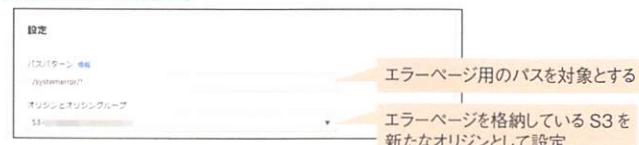
以上の 2 項目が CloudFront を利用するうえで必須の設定ですが、その他にも次表のような詳細設定を行うことで、性能の改善や可用性の向上を図ることができます。

設定項目	説明
ビヘイビア	<ul style="list-style-type: none"> <li>CloudFrontで利用可能とするプロトコルやHTTPメソッドの設定、特に重要なキャッシング設定が行える</li> <li>「*.mp4」のようなWebサービスで利用されるパスパターン単位で制御する</li> <li>キャッシング設定は、キャッシングポリシーとして詳細な設定が行える</li> </ul>
エラーページ	<ul style="list-style-type: none"> <li>オリジンでの障害発生時の動作を設定できる</li> <li>特定のHTTPエラーコードごとに、それが返されたときにどのようなレスポンスをユーザーに返すか、を定義する</li> <li>「ビヘイビア」設定と組み合わせて、エラー発生時は別のオリジンに転送する、といった細かな制御も実現可能</li> </ul>
地理的制限	<ul style="list-style-type: none"> <li>特定の国からのアクセスのみ許可する設定や、逆に特定の国からのアクセスを拒否する設定が行える</li> </ul>
キャッシング削除	<ul style="list-style-type: none"> <li>CloudFront上のキャッシングを手動で削除する操作を実行できる</li> </ul>

たとえば、「エラーページ」設定と「ビヘイビア」設定を組み合わせることで、オリジンサーバーでのエラー発生時に専用のエラーページを表示させるといった、Web サイトではよく見られる作りを実現できます。

## 専用のエラーページを表示する仕組みも簡単に実現できる

### ビヘイビアの設定



### 実際の動き



## CloudFront の利用料

CloudFront も他の多くのサービスと同様に従量課金制です。通信が行われる地域によって単価は微妙に変わりますが、基本的に CloudFront の仕組みが実際に利用されたデータ転送量と CloudFront へのリクエスト数に従って料金が決定されます。

従量課金ではありますが、データ転送量は月 1TB まで、HTTPS リクエスト数は月 1,000 万リクエストまでは無料で利用可能です。無料枠を超えた場合は、日本からの利用であれば、データ転送量 1GBあたり 0.114USD、HTTPS リクエスト 1 万件あたり 0.012USD の料金が発生します。無料枠はありますが、大容量コンテンツでなくともアクセス数次第では想像以上の料金がかかることがあるため、注意が必要です。

CloudFront では、独自に使用状況レポートが提供されています。利用状況を把握することで、直近の利用料をある程度予測できます。

A screenshot of the CloudFront usage report interface. It shows a table with columns for '日付範囲' (Last 30 days), '請求リージョン' (All locations), '粒度' (Daily), and '次' (Next). There are also buttons for 'すべてのディストリビューション' (All distributions) and 'CSV をダウンロード' (Download CSV).

## 09

# 多彩なネットワーク機能を提供するサービス7選

**keyword** • AWS Direct Connect • AWS VPN • AWS Transit Gateway • AWS PrivateLink  
• Amazon API Gateway • AWS Global Accelerator • AWS Ground Station

## AWSとの接続方法は多彩

AWSへは基本的にインターネットを介して接続することになりますが、用途によってはオンプレミス環境と連携するなどの理由で、ネットワークの安定性・安全性が求められることがあります。5-4節（p.139）でそのためのサービスをいくつか紹介しましたが、ここであらためて接続方法のまとめと、それぞれのサービスの詳細について説明します。また、本節の後半では、構築したシステムを外部から利用させる際に便利なサービスを紹介します。

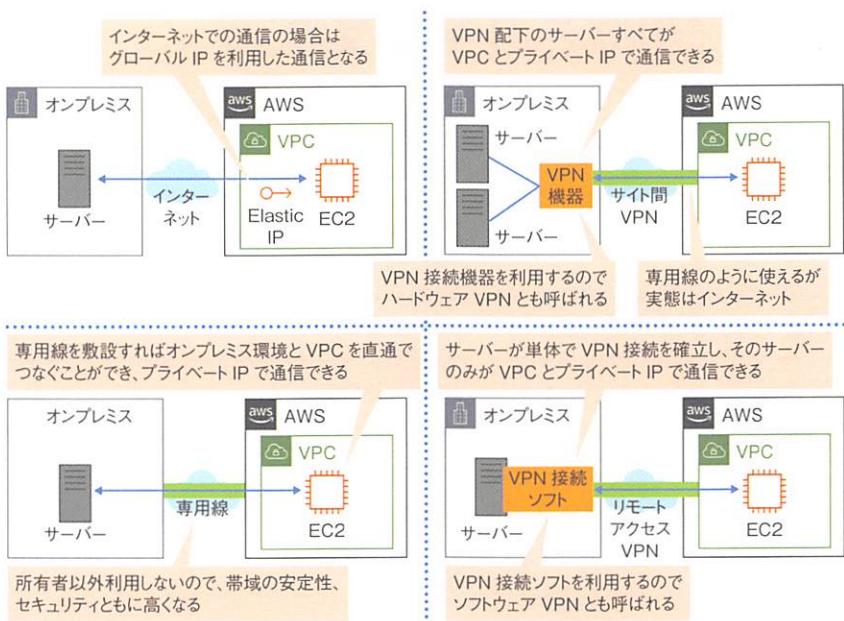
## 専用線とVPN

オンプレミス環境とAWSに作成したVPCを接続する場合、インターネットのほかに専用線とVPN（Virtual Private Network）という選択肢があります。

専用線とは、拠点間に物理的に敷設された専用の通信回線です。不特定多数が共用するインターネット回線と異なり、その回線所有者の独占利用となるので通信の安全性は高く、他者の通信による回線の逼迫もないため安定した通信速度が見込めます。一方、導入のための費用は高額になります。

VPNはインターネットを仮想的に専用線のように扱う技術です。通信を行う機器間で通信の暗号化と経路の制御を行うことで、お互いの拠点間を専用線でつないでいるかのように通信できるようになります。次図に示すとおり、サイト間VPNとリモートアクセスVPNという分類があり、用途によって使い分けられます。物理的な経路としてはインターネットなので、通信量増加による回線逼迫などの影響は受けますが、比較的安価に利用することができます。

## オンプレミスと AWS の接続には主に 4 つの方法がある

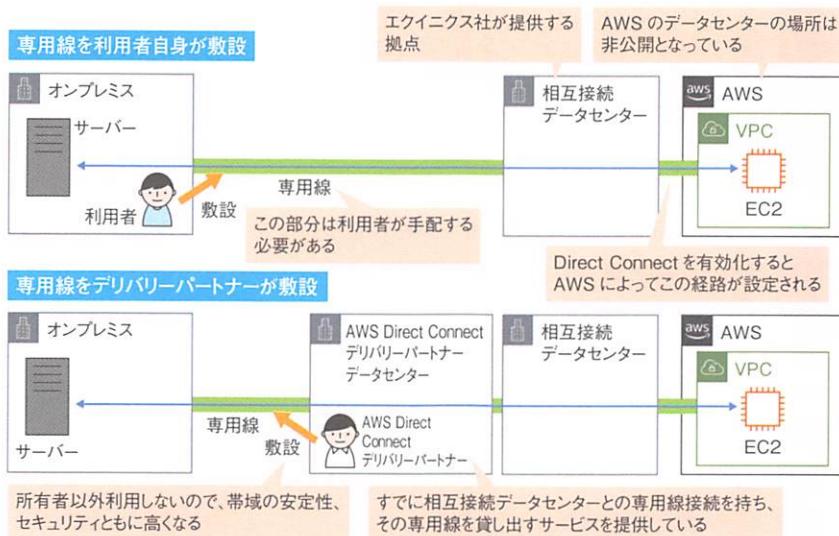


### AWS Direct Connect

AWS Direct Connect (以下 Direct Connect) は、AWS とオンプレミス環境との間に専用線を引くためのサービスです。AWS のデータセンターの場所は非公開となっているので、実際には AWS の指定する接続拠点（相互接続データセンター）までの専用線を利用者が用意することで、AWS がそこから利用者のアカウントの環境に通信をつなげてくれます。なお、Direct Connect は DX と略されることもあります。

専用線は利用者が通信事業者と契約して独自に敷設することができますが、AWS Direct Connect デリバリー・パートナーに認定されている企業が敷設済みの AWS の接続拠点までの専用線を利用させてくれるサービスもあります。後者の場合は、利用者のデータセンターまでの接続を AWS Direct Connect デリバリー・パートナーに任せることもできるため、より手軽に Direct Connect を利用できます。

## Direct Connect は専用線を敷設するサービス



## AWS VPN

AWS には VPN を利用して接続することも可能です。先に説明したサイト間 VPN、リモートアクセス VPN の両方をサポートしており、それぞれに対応するサービスが提供されています。

### • AWS Site-to-Site VPN

AWS Site-to-Site VPN（以下 Site-to-Site VPN）は、名前のとおりサイト間 VPN を構成するためのサービスです。Site-to-Site VPN を設定することで AWS 側に仮想プライベートゲートウェイと呼ばれる VPN 接続先を作成し、オンプレミスに設置した VPN 機能を持つルーター機器と VPN 接続を確立します。そのルーター機器を介したオンプレミス環境からの通信は、VPN 接続を通じて AWS に到達するようになるので、オンプレミス環境と AWS の VPC が相互にプライベート IP アドレスで通信できるようになります。またかも専用線でつながっているかのような構成にできます。Site-to-Site VPN では IPsec VPN という技術が使われています。

- AWS Client VPN

AWS Client VPN（以下 Client VPN）は、リモートアクセス VPN を用いて AWS に接続するためのサービスです。Client VPN エンドポイントと呼ばれる VPN 接続先を作成し、接続元パソコンにインストールされた VPN 接続ソフトウェアから VPN 接続を確立し、そのパソコンが AWS の VPC へプライベート IP アドレスで通信できるようにします。Client VPN では SSL-VPN という技術が使われています。

利用される技術は異なるものの、両方とも同じようにオンプレミス環境から AWS へ仮想的な専用線接続を確立するサービスです。拠点ごと AWS と VPN 接続したい場合は Site-to-Site VPN、特定の端末だけ一時的に VPN 接続したい場合は Client VPN を選択します。あくまでインターネット接続を専用線のように見せかけているだけなので、インターネット接続と同様に他の利用者による通信回線の逼迫などで通信速度の低下などが発生する可能性があります。ただ、すでに利用しているインターネット回線がそのまま流用できるため、専用回線が必要となる Direct Connect に比べコストが大幅に抑えられます。

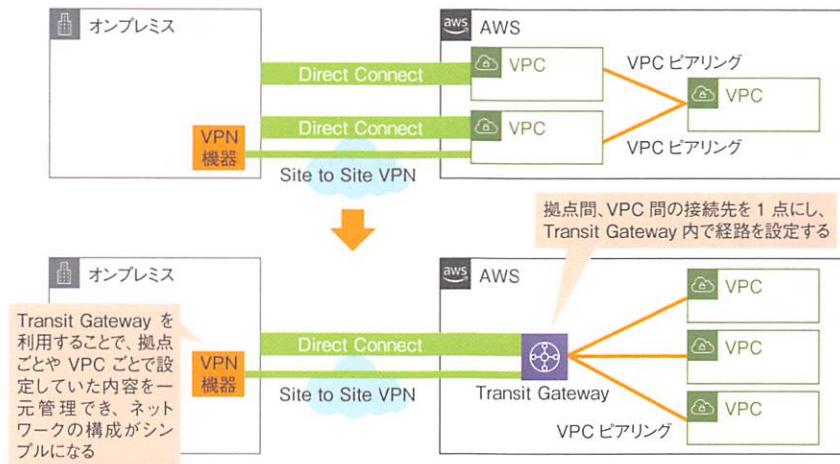
Site-to-Site VPN は Direct Connect と併用することも可能で、「通常時は Direct Connect を利用して通信し、Direct Connect の異常時（回線障害など）には事前に接続を確立しておいた Site-to-Site VPN を使う経路に切り替える」という使い方も可能です。

## AWS Transit Gateway

p.139 でも触ましたが、VPC の接続をまとめたためのサービスとして、AWS Transit Gateway（以下 Transit Gateway）があります。ここまでで説明したとおり、オンプレミスと AWS をつなぐ通信サービスにはさまざまな種類が用意されており、さらに VPC 同士を接続する構成をとることも多くなっています。Transit Gateway を利用することで、次図のように VPC への接続をすっきりさせることができます。

Transit Gateway を通る通信 1GBあたり 0.02USD、接続する VPC ごとに 1 時間あたり 0.07USD の料金が発生するため、それぞれ単独で VPC と接続する構成に比べ料金が高くなりますが、図のように 1 カ所で接続を管理できるため全体の構成を把握しやすく、運用の煩雑さが大幅に改善されます。接続が複雑になるシステムでは導入する価値が高いといえるでしょう。

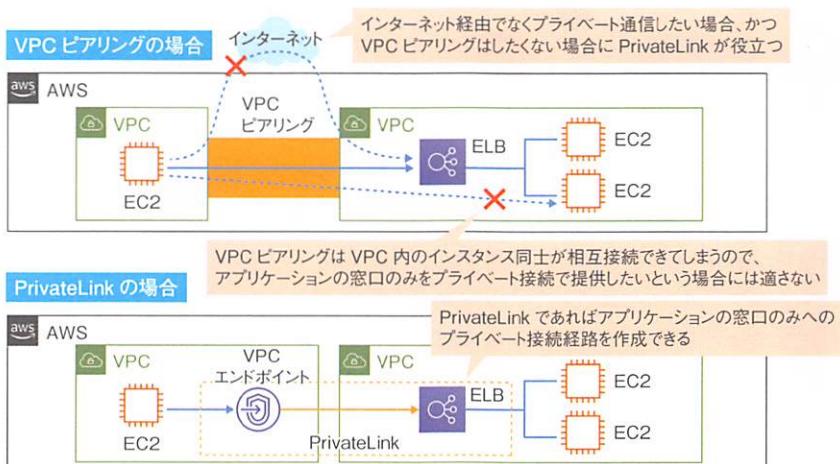
## Transit Gateway はネットワーク間の接続を一元管理するサービス



## AWS PrivateLink

AWS PrivateLink (以下 PrivateLink) は、VPC エンドポイントを作成するサービスです。p.140 でも説明しましたが、PrivateLink を利用することで AWS 上に構成されたシステムの提供するサービスを、インターネットを経由することなく別の

## PrivateLink は他の VPC 上のサービスとのプライベートな接続を提供

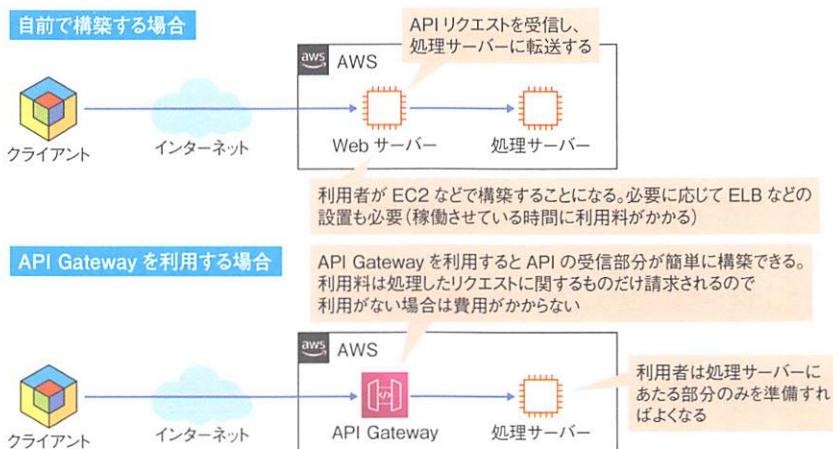


VPC から利用できるようになります。VPC 同士の通信だけなら VPC ピアリングでも行えますが、PrivateLink であれば前ページの図のように接続先を制限した構成が可能となります。

## Amazon API Gateway

Amazon API Gateway (以下 API Gateway) は、Web API を作成するサービスです。API とは Application Programming Interface の略で、アプリケーション同士が通信するための窓口を意味します。API Gateway を使えば、利用者はデータの処理を行う部分を開発するだけで、すぐに Web 上に API サービスを公開できます。

### Web API を提供する用途では API Gateway が便利



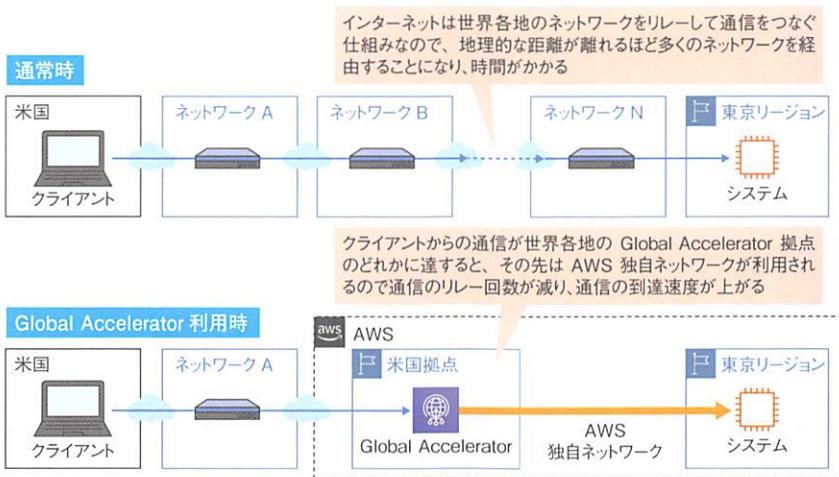
API Gateway を使わずに Web 上に API サービスを公開する場合は、自分で Web サーバーを構築し、Web 上に API リクエストを受ける窓口を作る必要があります。この場合は、構築の手間に加え、Web サーバーを動かすためのコストがかかってしまいます。API Gateway は処理したリクエスト数とデータ転送量に対してのみ利用料が発生するようになっており、さらに認証機能を備えていたり、エラー率や処理時間の監視機能を備えていたりと多機能であるため、API を作成するという点では非常に優秀なサービスといえます。

## AWS Global Accelerator

**AWS Global Accelerator**（以下 Global Accelerator）は、世界中に広がる AWS のネットワーク網を利用してすることで、クライアントがより高速に AWS 内のシステムにアクセスできるようにするサービスです。AWS によると最大 60% ものトラフィック性能が向上するといわれています。世界中に入り口のあるインターネット上の高速道路というとイメージしやすいかもしれません。CloudFront と同様に、世界中に拠点を持つ AWS ならではのサービスといえるでしょう。

また、Global Accelerator を利用するとシステムへアクセスするための固定 IP アドレスが発行されるため、システムを利用する IP アドレスを固定化したいときに採用されることもあります。

### Global Accelerator は国境を越える通信を高速化するサービス



## AWS Ground Station

ネットワークとは少し毛色が違いますが、AWS は人工衛星との通信も提供しています。AWS Ground Station というサービスで、人工衛星の利用予約と AWS の所有する基地局との通信ができるようになります。

衛星から取得できるデータは主に地球の観測データであり、地球の写真画像や地

形、天候情報といったものが挙げられます。利用には米国のアマチュア無線ライセンスである FCC ライセンスが必要であるなど手軽に利用できるとはいえませんが、AWS の事業範囲の広さと可能性を示すサービスです。



# データベースサービス

情報システムにおいて最も重要なものはデータです。データを扱うことに特化し、扱うデータの特性ごとに独自の進化を遂げたデータベースサービスについて解説します。

- section  
01** データベースとは  
押さえておきたいデータベースの基礎知識
- section  
02** Amazon RDS とは  
AWS でリレーショナルデータベースを使う
- section  
03** Amazon Aurora とは  
Aurora の高機能で便利な特徴を知っておこう
- section  
04** Amazon DynamoDB とは  
キーと値の組み合わせでデータを管理する
- section  
05** Amazon Redshift とは  
データ分析のために大量データを取り込む
- section  
06** AWS におけるデータベース移行  
データベース移行に役立つサービス 2 選
- section  
07** その他のデータベースサービス  
多種多様なデータベースに対応するサービス 7 選

## 01

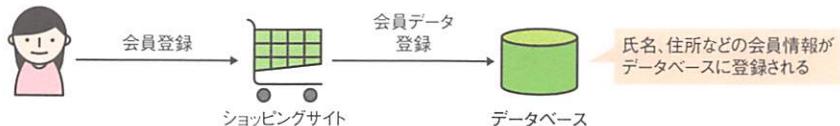
押さえておきたい  
データベースの基礎知識

- keyword**
- ・データベース
  - ・データベース管理システム（DBMS）
  - ・リレーショナルデータベース
  - ・NoSQL データベース
  - ・SQL
  - ・ACID 特性
  - ・トランザクション

## データベースはデータの集合体

本章では AWS の主要なデータベースサービスを紹介します。そもそも **データベース** とは何でしょうか。データベースとは、検索や登録が容易にできるよう **整理されたデータの集まり** です。みなさんも Amazon などのネットショッピングを利用する際、氏名や住所などを登録していますね。これらの登録情報も何らかのデータベースに登録されて管理されています。

## データベースにデータを登録してアプリケーションから利用する



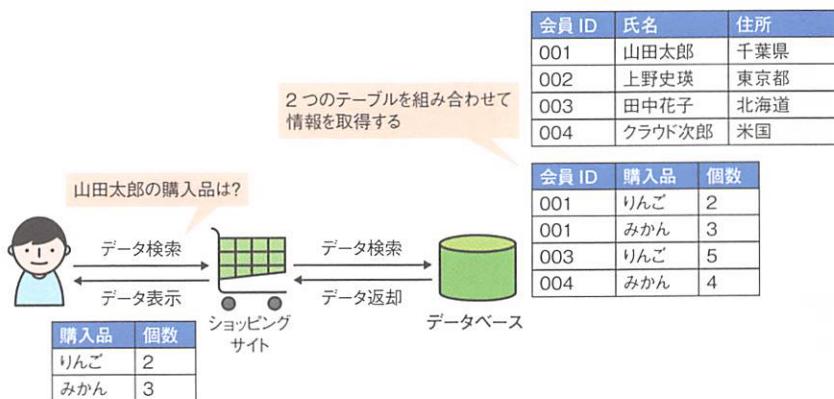
データベースは外部のアプリケーションなどから更新、参照、削除などの操作を受けて適切にデータを操作する必要があります。こういった操作を受けてデータを管理するシステムを **データベース管理システム (DBMS、 DataBase Management System)** と呼びます。

現状、システムで最も多く使われるデータベースが、**リレーショナルデータベース (RDB)** と呼ばれる表形式のデータベースです。次のように表形式で管理される会員情報はリレーショナルデータベースの一例です。

会員ID	氏名	住所
001	山田太郎	千葉県
002	上野史瑛	東京都
003	田中花子	北海道
004	クラウド次郎	米国

1つの表を「テーブル」という単位で管理します。リレーションナル（関係）の言葉どおり、通常は複数のテーブルが関係性を持って管理されます。たとえば、会員テーブルと購入テーブルを別々に管理しておき、購入情報の管理を行う場合はそれらのテーブルを組み合わせて利用します。

## 複数のテーブルを組み合わせてデータを取り出せる



リレーションナルデータベース以外にも AWS には「キー」と「バリュー（値）」という 2 つの要素を組み合わせ、キー・バリュー型で管理する **NoSQL データベース** のサービスも存在します。本章の各サービスのページで、各データベースの特徴も紹介するので、データベースごとの違いも併せて勉強していきましょう。

## SQL で操作するリレーションナルデータベース

リレーションナルデータベースにあるデータに対し、通常は追加、検索、更新、削除の操作を行いますが、これらのデータ操作は **SQL** (Structured Query Language) という言語を使用して行います。それぞれのデータ操作について、次の命令文を使用できます。

- **SELECT** (データの検索)
- **INSERT** (データの追加)
- **UPDATE** (データの更新)
- **DELETE** (データの削除)

たとえば、先ほどの会員情報テーブルから氏名情報をすべて取る SQL 文は、次のようにになります。

```
SELECT 氏名 FROM 会員情報テーブル
```

**結果**

氏名
山田太郎
上野史瑛
田中花子
クラウド次郎

今回の例では列名とテーブル名を日本語で記載していますが、通常システムでは name や accounts といったアルファベットが使用されます。

WHERE 句を指定して、条件指定でデータを取得することもできます。たとえば会員 ID が 003 の行の氏名情報を取る場合の SQL 文は、次のようになります。

```
SELECT 氏名 FROM 会員情報テーブル
```

```
WHERE 会員 ID = '003'
```

**結果**

氏名
田中花子

列名（表の「会員 ID」「氏名」「住所」の部分）は「\*」という文字を指定して、すべての列の情報を取得することも可能です。

```
SELECT * FROM 会員情報テーブル
```

```
WHERE 会員 ID = '003'
```

**結果**

会員ID	氏名	住所
003	田中花子	北海道

データの作成や更新、削除も同様に SQL を使用して実行できます。

## データの不整合を無くすための ACID 特性

リレーションナルデータベースが持つ特徴のひとつに、ACID というものがあります。これは原子性 (Atomicity)、一貫性 (Consistency)、独立性 (Isolation)、耐久性 (Durability) の頭文字を取ったものです。データベースに行う処理のひとまとまりをトランザクションと呼びますが、このトランザクション単位で、ACID 特性を持つことになります。

たとえば、銀行口座の預金引き出しについて考えてみてください。システム的には次の 2 つの処理になります。

- (1) 指定した金額を引き出し
- (2) 預金残高データを更新

(1) の処理のみで中断してしまうと、預金残高が正しくなくなるため、(1)、(2) の両方とも実行されるか、いずれも実行されないようにする必要があります。このように、データや処理の不整合を起こさないために ACID 特性が使用されます。4 つの特性の意味は、それぞれ次のとおりです。

### ● 原子性 (Atomicity)

トランザクションは完全に実行されるか、まったく実行されないかのどちらかであるという特性です。

### ● 一貫性 (Consistency)

決められたデータベースのルール条件を満たすという特性です。たとえば預金残高が 1 万円しかないのに 3 万円引き出すことはできません。

### ● 独立性 (Isolation)

トランザクションを単独実行した場合でも、同時に複数実行した場合でも同じ結果である必要があるという特性です。トランザクション間で影響を与えない（独立している）という意味です。

### ● 耐久性 (Durability)

ハードウェアなどの障害があっても、完了したトランザクションの結果は失われてはいけないという特性です。

## 02 AWS でリレーショナルデータベースを使う

**keyword** • Amazon RDS • RDS インスタンスタイプ • スタンバイレプリカ • リードレプリカ  
• RDS のセキュリティ設定 • メンテナンスウィンドウ • バックアップウィンドウ

### RDS はリレーショナルデータベースサービス

Amazon Relational Database Service (以下 RDS) は、リレーショナルデータベースを提供するサービスです。

OS やデータベースエンジンの管理は AWS 側で行われ、利用者は数分でデータベースを起動してアクセスできます。EC2 同様、従量課金となっており、RDS インスタンスが起動している時間に対して利用料が発生します。

データベースエンジンとは、データベースに対しデータを登録、更新、削除するための処理を管理するソフトウェアです。2022 年 1 月現在、RDS では次の 6 つのデータベースエンジンをサポートしています。

- MySQL
- SQL Server
- MariaDB
- PostgreSQL
- Oracle
- Amazon Aurora

このうち Amazon Aurora は、クラウド向けに AWS が構築したデータベースです。MySQL および PostgreSQL と互換性があります。システム構造や機能が他のデータベースエンジンと異なる部分が多いため、Amazon Aurora の特徴は 6-3 節 (p.182) で詳しく紹介します。

EC2 やオൺプレミスでデータベースを構築する場合、利用者自身が OS 設定やデータベースエンジンのインストール、設定を行う必要があります。それなりに時間がかかります。RDS を使用する場合は、あらかじめデータベースエンジンが設定された状態で作成されるので、すぐにデータベースを利用できます。

RDS でデータベースを作成するには AWS マネジメントコンソール (画面) を操作するか、API を使用してコマンドやプログラムから行えます。マネジメントコンソールから作成する場合は、次図のように画面から必要な情報を選択すれば OK です。

## 必要な情報を入力するだけでデータベースがすぐに作成できる

データベース作成方法を選択

標準作成  
可視性、セキュリティ、バックアップ、メンテナンスといったすべての設定オプションを設定します。

簡単に作成  
推奨されるベストプラクティス設定を使用します。一部の設定オプションは、データベースの作成後に変更できます。

**設定**

エンジンのタイプ

Amazon Aurora



MySQL



MariaDB



PostgreSQL



Oracle



Microsoft SQL Server



DB インスタンスサイズ

本番稼働用

db.r5g.xlarge  
4 vCPUs  
32 GiB RAM  
500 GiB  
1.209 USD/時間

開発/テスト

db.r5g.large  
2 vCPUs  
16 GiB RAM  
100 GiB  
0.274 USD/時間

無料利用枠

db.t2.micro  
1 vCPUs  
1 GiB RAM  
20 GiB  
0.030 USD/時間

DB インスタンス識別子

DB インスタンスの名前を入力します。この名前は、AWS アカウントが現在の AWS リージョンで所有しているすべての DB インスタンスにおいて一意である必要があります。

database-1

次のような情報を入力することになります。

- データベースエンジン、バージョン
- DB インスタンス識別子（名前）
- マスターアカウント名、パスワード（ログインに使用）
- DB インスタンスクラス（インスタンスタイプ、マシン性能）
- ストレージタイプ設定（容量や性能を決定）
- マルチ AZ 配置
- ネットワーク設定（配置する Amazon VPC など）

必要情報を入力して作成すると、データベースが起動して接続可能になります。接続して SQL を実行することで、テーブルやデータの作成が行えます。

## データベースの性能を決定するインスタンスとストレージタイプ

RDS で作成するデータベースが稼働するサーバーとストレージは、それぞれタイプを選択することによって性能が変動します。第3章で紹介した EC2 と同じく、RDS で稼働するサーバーのことを **インスタンス** と呼びます。

### ○ RDS インスタンスタイプ

インスタンスの性能（CPU、メモリ）は、インスタンスタイプを選択することにより決定します。インスタンスタイプのネーミングは、次図のようになっています。

#### RDS インスタンスタイプの名前のルール



最初が「db」に固定である以外は、EC2 と同様のネーミングルールになっています。利用料は従量課金となっており、時間あたりの単価はインスタンスタイプにより異なります。高スペックなインスタンスタイプほど高価となります。

### ○ストレージタイプ

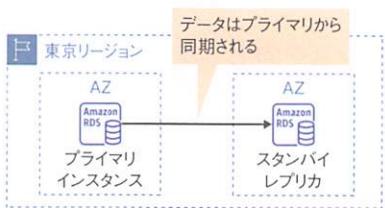
4-5節 (p.113) で紹介した EBS と同じく、RDS でもデータを格納する場所であるストレージの設定を行います。データベースはデータを格納することが主な目的となるため、ストレージの選定がより重要になってきます。ストレージタイプを次の3種類から選択し、容量を指定します。

ボリュームタイプ	概要
汎用SSD	高性能で容量により読み書き性能が決定
プロビジョンドIOPS SSD	高性能かつ指定された読み書き性能を確保
マグネットิกHDD	低コスト

## 可用性向上のスタンバイレプリカと読み込み性能向上のリードレプリカ

RDSでは、メインでデータ処理を行う**プライマリインスタンス**の他に、**スタンバイレプリカ**を別のAZ（アベイラビリティゾーン）に配置することでマルチAZな高可用性を実現できます。**高可用性**とは、システムが停止することなく稼働できる能力のことを指しますが、RDSではこれをスタンバイレプリカにより実現しています。この構成を**マルチAZ配置**と呼びます。

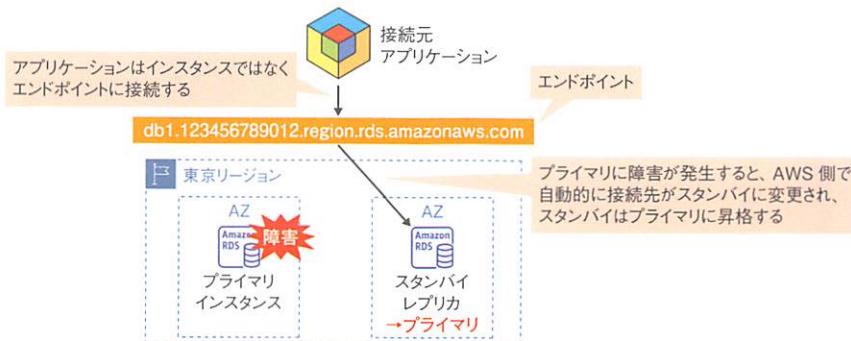
### 停止させたくないシステムはマルチAZ配置にする



プライマリインスタンスやAZに障害があった場合、スタンバイレプリカに切り替えることでデータベース接続を継続できます。この切り替えのことを**フェイルオーバー**と呼びます。

アプリケーションからRDSへ接続する際は、**エンドポイント**というDNS形式の、WebページのURLのような接続情報を使用します。プライマリインスタンスに障害が発生した場合、このエンドポイントの向き先が自動的にスタンバイレプリカに切り替えられます。

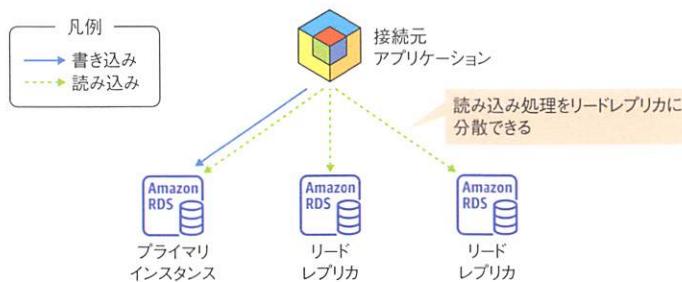
### 障害発生時にはスタンバイレプリカに自動的に切り替わる



もうひとつ、RDS には、読み込み専用として利用できる、**リードレプリカ**という機能があります。リードレプリカはプライマリインスタンスから複製されたインスタンスです。読み込み処理について、インスタンスの数を増やすスケールアウト方式でパフォーマンスが向上できます。

なお、リードレプリカはあくまで読み込み専用のため、書き込み性能は向上できません。書き込み性能については、インスタンスサイズを変更してスケールアップさせることによりパフォーマンスを向上させます。先に紹介したスタンバイレプリカは、可用性の向上が目的であり、通常処理は行わないインスタンスのため、性能向上には役立ちません。

### リードレプリカで読み込み処理のパフォーマンスを向上

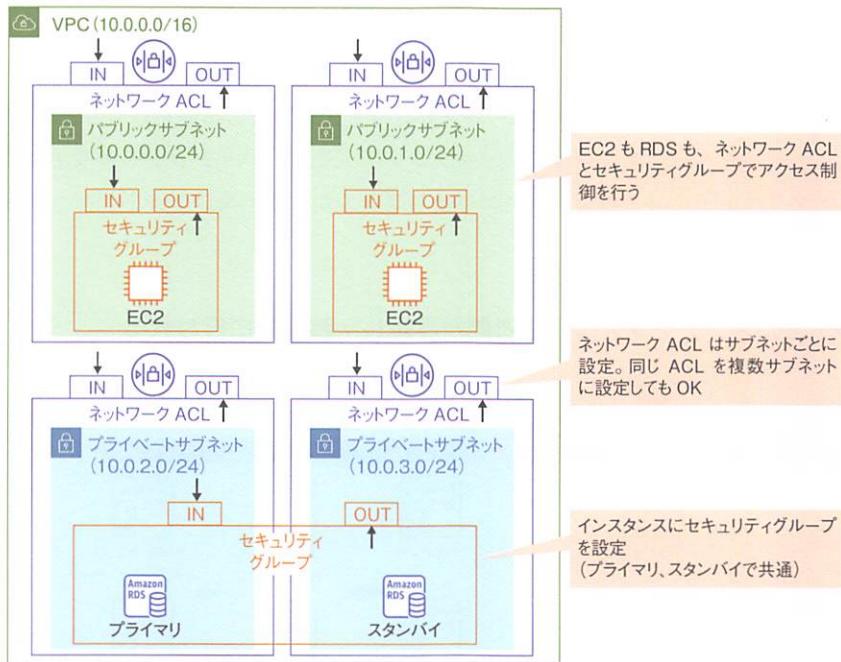


## RDS のセキュリティ設定とバックアップ

### ○ RDS のアクセス制御

RDS は、EC2 と同じく、第 5 章で紹介した VPC 内に作成します。RDS インスタンスを作成するには、少なくとも 2 つのサブネットが必要です。これらのサブネットはリージョン内の異なる AZ を指定する必要があります。データベースにどこからアクセスできるかという設定は、EC2 と同じくセキュリティグループとネットワーク ACL で設定します。

## RDSはVPC内に作成したうえでアクセス制御の設定を行う



通常はデータベースにアクセスするのはVPC内に構築したアプリケーション（EC2などで稼働）のみの場合が多いため、アプリケーションからのみデータベースに接続できるようセキュリティグループやネットワークACLを設定します。

### ○データベースエンジンのアップデート

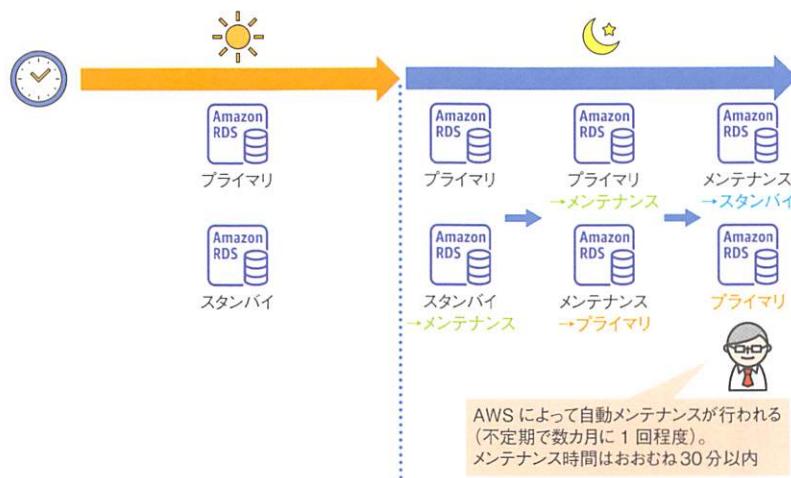
アクセス制御と併せて大事になるのが、データベースエンジンのアップデート（更新）です。データベースエンジンやデータベースエンジンが稼働するOSが古くなると、脆弱性（セキュリティ上の欠陥）が出てくることもあるため、定期的にバージョンアップを行う必要があります。

こういったバージョンアップはAWS側で実施されますが、更新作業はメンテナンスウィンドウという、利用者側が設定した時間帯に実施されます。いくつかのメンテナンス作業では、RDSインスタンスが一時的にオフライン（利用不可）となります。そのため、データベースの使用率が最も低い時間帯を設定するとよいでしょう。

う。たとえば日中にアプリケーションの利用が多く、データベースの使用率が高くなる場合は、利用の少ない夜間の時間帯を設定します。

マルチ AZ 配置についていた場合、スタンバイレプリカ→プライマリインスタンスと順にメンテナンスが実行されます。データベースへのアクセスのないインスタンスをメンテナンスするため、データベースがオフラインとなる影響を軽減できます。ただし、データベースエンジンのアップグレード（新しいバージョンへの変更）はプライマリインスタンスおよびスタンバイレプリカが両方同時にアップグレードされます。

### AWS 側のメンテナンスは利用者側が設定した時間帯に行われる



### ○データのバックアップ

データを守るという意味では、データが壊れたり消失したりした場合に復旧できる必要があります。ソフトウェアや AWS の障害でデータが壊れる可能性もあれば、利用者の誤操作によってデータが壊れる可能性もあります。そういう事態に復旧できるよう、データのバックアップを行います。

RDS には**自動バックアップ機能**があり、これを有効にすることで利用者が指定した日数分の**スナップショット**が自動保持されます。保持日数は 0 ~ 35 の範囲で指定でき、0 の場合は自動バックアップが無効となります。初回のスナップショットは、すべてのデータをバックアップしますが、2 回目以降のスナップショットは差

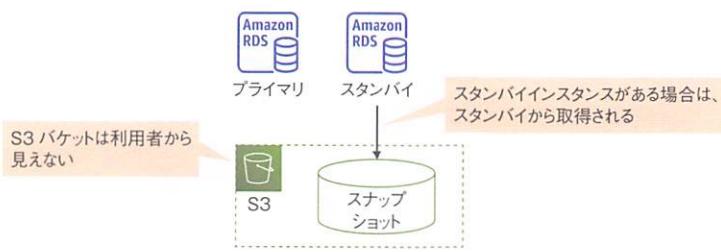
分（変更された）データのみ取得します。基本的な考え方は p.114 で紹介した EBS のスナップショットと同様です。

スナップショットの取得処理は、利用者が指定した**バックアップウィンドウ**の時間帯で実行されます（メンテナンスウィンドウとは別の時間帯を指定します）。利用者の好きなタイミングで、手動スナップショットの取得も可能です。

スナップショットのファイルは S3 バケット上に格納されるため、ファイルの耐久性は非常に高いです。S3 に保存されますが、利用者側から S3 バケットのファイルとして見えるわけではありません。仕組み上、S3 が使用されているとご理解ください。

スナップショットはバックアップされたデータ量に対して料金が発生します。通常使用する RDS のストレージ料金に比べ、10 分の 1 程度の料金となっています。

## RDS の自動バックアップ機能で、指定した日数分のデータを取得



スナップショットは、新しい RDS インスタンスを作成するという形で復元でき、既存の RDS インスタンスへは復元できません。

**ポイントインタイムリカバリ** という機能を使用して、特定の時間（最短で 5 分前、保持期間内）を指定して復元することもできます。こちらはデータをずっとバックアップとして残しておくよりも、特定時間に戻すという用途で使用されます。スナップショットとデータベースの変更ログを組み合わせて使用することで、この仕組みを実現しています。

# Amazon Aurora とは

## 03 Aurora の高機能で便利な特徴を知っておこう

keyword • Amazon Aurora • DB クラスター • クォーラムモデル • Aurora レプリカ

### Aurora は AWS に最適化されたデータベース

Amazon Aurora (以下 Aurora) は、クラウド向けに AWS が構築したデータベースです。MySQL と PostgreSQL に互換性があるため、アプリケーションからは通常の MySQL や PostgreSQL と同様の方法で接続できます。

インスタンスタイプを決定して VPC 内に作成するという基本的な部分は、これまで紹介した他のデータベースエンジンと同様です。構造の違いや Aurora 独自の機能がいくつかあるため、それについて紹介します。

まずは Aurora の構造について紹介します。Aurora は DB クラスターという単位で管理され、処理を行う 1 つ以上の DB インスタンスと、データを管理する クラスター ボリューム で構成されます。クラスター内のインスタンスでストレージが共用されているのが特徴です。Aurora 以外のデータベースエンジンでは、EBS がインスタンスにアタッチされて、インスタンスごとにデータの管理が行われます。

クラスター ボリューム は、3 つの AZ に 2 個ずつ、計 6 個のコピー が作成され、高い耐障害性を持ちます。書き込み処理は並列に行われるため、このコピーが原因で書き込み処理が遅くなるといったことはありません。

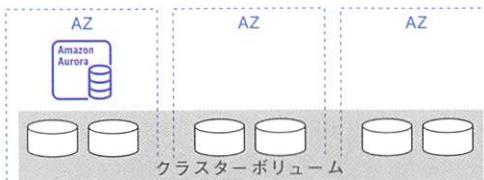
### Aurora の構造は他のデータベースエンジンと大きく異なる

Aurora 以外



Aurora 以外のエンジンは  
1インスタンスごとに  
1つの EBS (ストレージ)

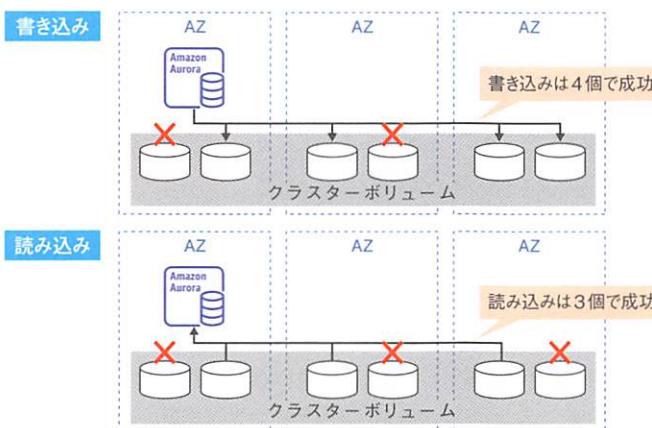
Aurora



Aurora ではクラスター内  
のインスタンスで  
ストレージが共用され、  
6つのコピーを持つ

計6個のコピーになりますが、Auroraではクォーラムモデルというレプリケーション管理（レプリケーションとは複製を作ることの意味）の仕組みが使われております、書き込み処理は6個中4個、読み込み処理は6個中3個成功したらクライアント側に成功と返します。

## 6個のコピーをクォーラムモデルで管理する



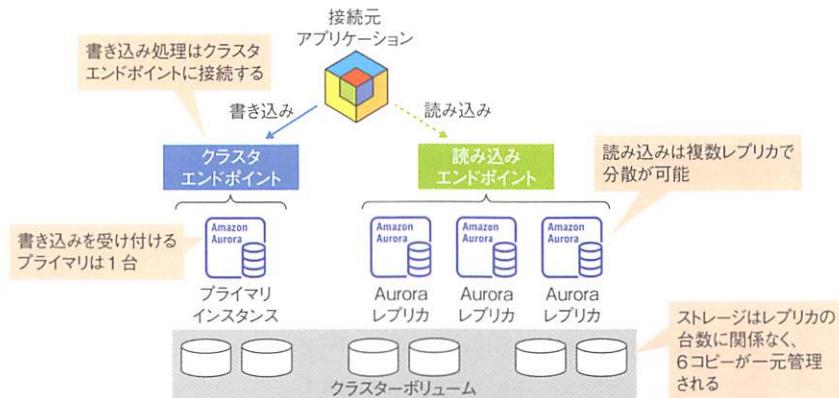
## AWSに最適化されたAuroraの機能

Auroraには、構造の違い以外にも Aurora 独自の機能がいくつか存在します。

最初に紹介するのが Aurora レプリカです。Aurora 以外のデータベースエンジンでは、可用性向上のためのスタンバイレプリカと、読み込み性能向上のためのリードレプリカをそれぞれ用意する必要がありました。Aurora レプリカは、**1つで可用性向上と読み込み性能向上の両方の役割を持ちます**。つまり、通常時は読み込み処理を受けつつ、プライマリインスタンスに障害があった場合に、切り替え（フェイエルオーバー）可能なレプリカとなります。

複数の Aurora レプリカを作成して、読み込み処理を分散できます。アプリケーションから見ると、**読み込みエンドポイント**の URL を指定することで、各レプリカに分散したアクセスが可能です。アプリケーションから見たアクセス先は 1 つになります。

## Aurora レプリカで可用性向上と読み込み性能向上を実現する



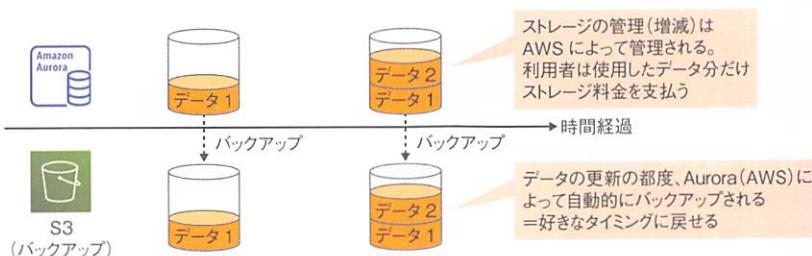
Aurora レプリカ以外にも、ストレージについて Aurora が有利な点があります。Aurora 以外のデータベースエンジンでは、ディスク性能要件に合わせてストレージタイプと確保する容量（100GB など）を利用者が決定します。Aurora では**ストレージタイプおよび容量の指定は必要ありません**。これらの管理は AWS 側で行われます。

Aurora のクラスター ボリューム（ストレージ）は、データ量に応じて自動的に増加し、最大 128TB まで増やすことができます。利用者は空き容量やストレージタイプの選定で悩む必要はありません。

バックアップについても、Aurora 独自の自動バックアップ機能があります。このバックアップは設定した保持期間分、**継続的かつ増分的に**取得されます。つまり、保持期間中の任意の時点へ復元できることになります。保持期間は 1 ~ 35 日で指定でき、S3 に保存されます。期間を 0（無効）にはできません。デフォルトの保持期間は 1 日となっています。他のデータベースエンジンではスナップショット + 変更ログという形でポイントインタイムリカバリを実現していますが、Aurora は独自の機能かつ自動的にバックアップが行われる（無効にできない）というのがポイントです。

他のデータベースエンジンと同じくスナップショット機能もあり、35 日を超えてバックアップを保管したい場合はスナップショット機能を使用して保存できます。

## Auroraのストレージ管理と自動バックアップ



Auroraは、その他にも次のような独自の機能を多く持っています。AWSを学び始めた段階でいきなりすべて覚えるのは難しいため、最初は「Aurora独自の便利な機能が多くある」くらいで覚えておくとよいでしょう。

- 複数リージョンにまたがるグローバルデータベース
- 2つのプライマリインスタンスを持つマルチマスタークラスター
- インスタンスのサイズ指定が不要な Aurora Serverless
- S3 と連携したファイルエクスポート、インポート機能

RDSでデータベースを作成する際、データベースエンジンの選択に迷うことが多いです。まずはAWSに最適化されたAuroraの利用を検討し、移行前の環境理由などでAuroraが使えない場合は他のデータベースエンジンといった具合で選ぶといででしょう。

# Amazon DynamoDB とは

## 04 キーと値の組み合わせでデータを管理する

**keyword** • key-value 型データベース • Amazon DynamoDB • キャパシティユニット  
• DynamoDB の利用

### key-value 型データベースとは

Amazon DynamoDB (以下 DynamoDB) は、key-value 型のデータを格納するデータベースです。シンプルなデータ構造となるため複雑な検索はできませんが、高速なデータの取り出しが可能です。

key-value 型とは、名前のとおり、キーの名前とキーに対する値からなるデータです。キーと値がセットとなっていればよいだけなので、データの形としては自由度が非常に高いですが、6-1 節 (p.170) で説明したリレーションナルデータベース (RDB) のように表の形に整理されていないので集計や検索はとても苦手です。

### リレーションナルデータベースと key-value 型のデータベースの違い

**RDB** 表形式に整形して格納 ➔ 検索や集計が得意

社員番号	氏名	所属部	入社年度
A0011	山田三郎	営業部	2003
B0021	佐藤花子	技術部	2010
B0120	田中武男	技術部	1999
C0013	鈴木隆	総務部	2010

2010 年度入社は何人?

技術部所属は誰と誰?

**key-value** キーと値のセットで格納 ➔ 検索や集計は苦手



プライマリキー (値が各レコードを一意に示す属性)

プライマリキー以外の内容はバラバラでもよい

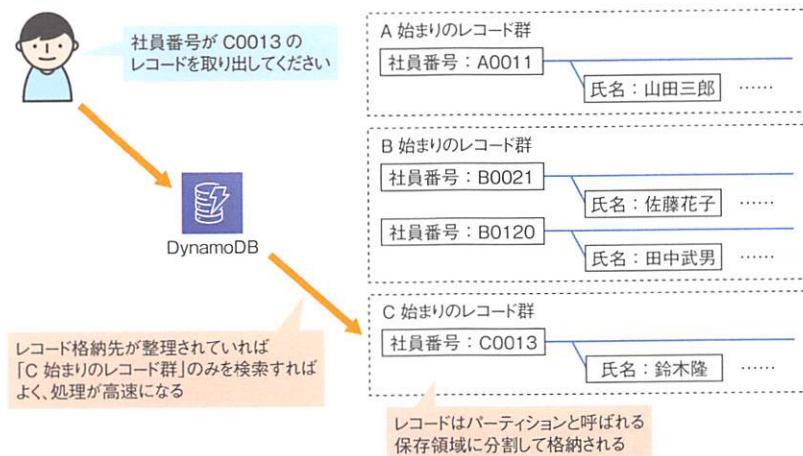
key-value 型のデータベースでは、一連のデータを「レコード」と呼び、レコードの各項目（キーと値のセット）は「属性」と呼ばれます。前ページの図のようにそれぞれのレコードが一意に特定できる属性（プライマリキーといいます）さえ持っていれば、他の属性はどういったものを持っていてもかまいません。各レコードは基本的にプライマリキーでのみ検索することが可能です。

## DynamoDB とは

DynamoDB は、AWS 独自の構成により、前図で見たような key-value 型のデータを 0.0 何秒というレベルで高速に扱うことのできるデータベースです。

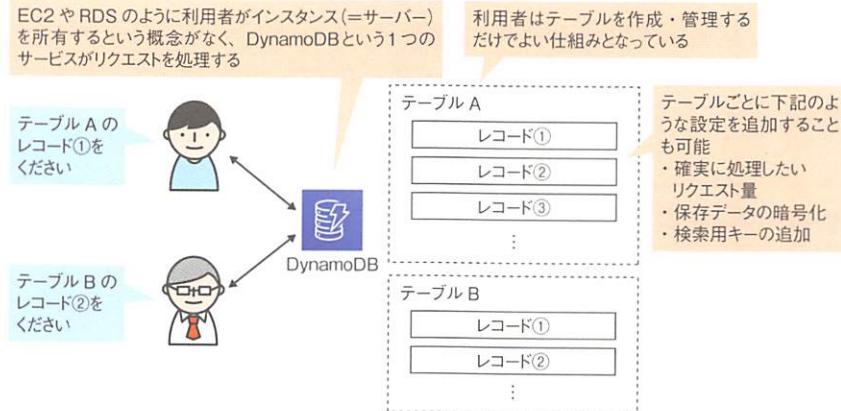
key-value 型のデータは基本的にプライマリキーのみで検索してデータを取り出すので、DynamoDB は内部的にプライマリキーでの検索が行いやすいようにデータを整理して保存します。

## DynamoDB は key-value 型のデータを高速に扱える



また、DynamoDB にはサーバーレスであるという特徴もあります。利用者はレコードの集合体である「テーブル」を作成するだけで、テーブルの操作リクエストはすべて DynamoDB が対応してくれるため、利用者がサーバーの管理を意識する必要はありません。テーブル作成時に最低限、テーブル名とプライマリキーを指定するだけで利用を開始できるのも DynamoDB の魅力です。

## DynamoDB はインスタンスを作成せずに使える（サーバーレス）



基本的な設定はデフォルトとして用意されており、最低限必要な設定は次図のものだけです。

**DynamoDB テーブルの作成**

チュートリアル ?

DynamoDB は、スキーマなしのデータベースであるため、テーブル名とプライマリキー以外は必要ありません。テーブルのプライマリキーは、項目を一意に識別し、データをパーティション分割し、さらに各パーティション内のデータをソートする 1~2 個の属性で構成されています。

テーブル名:  ⓘ

プライマリキー: パーティションキー  
 ⓘ  
 文字列 ⚙ ⓘ

ソートキーの追加

**テーブル設定**

デフォルト設定は、最も手軽にテーブルの使用を開始する手段です。これらのデフォルト設定は、今すぐまたはテーブルの作成後に変更できます。

デフォルト設定の使用

- セカンダリインデックスなし。
- キャパシティーセットは 5 つの読み込みと 5 つの書き込みにプロビジョニングされています。
- SNS トピック「dynamodb」を使用した上限しきい値 80% の基本アラーム。
- デフォルトの暗号化タイプによる保管時の暗号化。

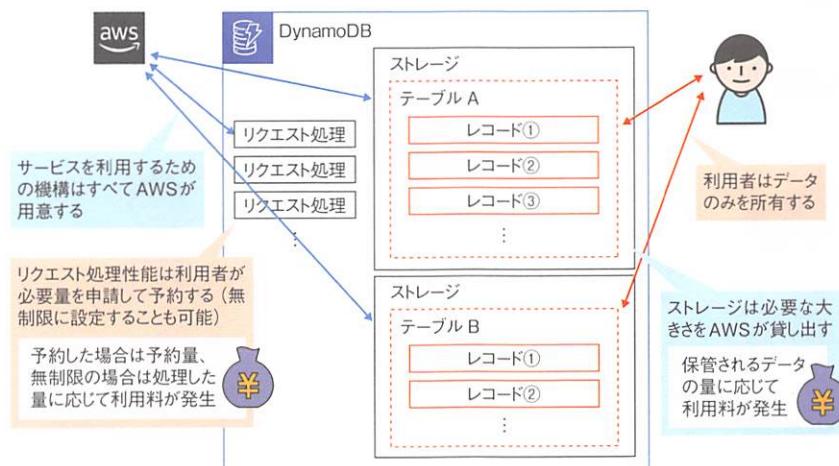
## DynamoDB の利用料

先に説明したとおり、DynamoDB はサーバーレスなサービスです。利用者はデータの集合である「テーブル」のみを所有する形となります。データの保存場所となるストレージはテーブルのデータ量に応じて必要なサイズを割り当てられ、操作リ

クエストは都度 DynamoDB が対応してくれます。

ではどのように利用料がかかるかというと、リクエストを処理してもらう量が予約制となっており、その量に対して料金がかかるようになっています。また、データのデータ保管料もデータサイズに応じてかかります。

リクエストの処理量とテーブルデータの保管に料金がかかる



リクエストの処理量を予約すると書きましたが、具体的には「1秒間に何回、読み込み／書き込みのリクエストを受け付けるか」を指定します。DynamoDBは予約量を超えるリクエストに対してはエラーを返しますが、Auto Scalingを利用することで実際のリクエスト数に応じて、指定した幅の中で予約量を自動的に変更することもできます。

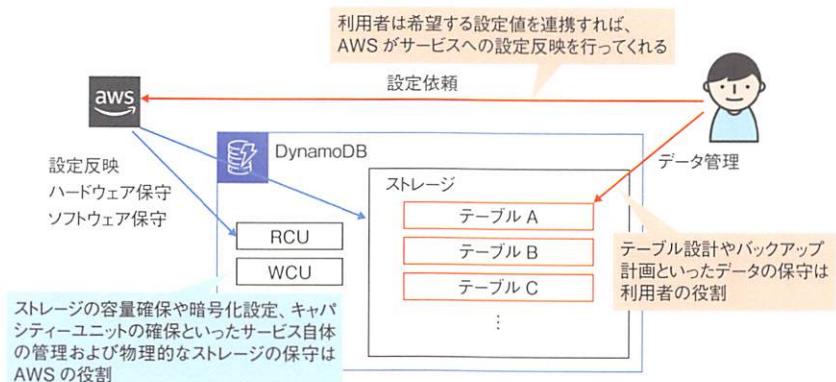
また、最初から予約量を指定せず、発生したリクエストをすべて処理するようにして、実際に処理されたリクエスト数に応じて料金を支払うこともできます。

処理量はキャパシティーユニットと呼ばれ、RCU (Read Capacity Unit)、WCU (Write Capacity Unit) と表記されます。処理量を予約する方式をプロビジョニング済みキャパシティーモード、予約しない方式をオンデマンドキャパシティーモードと呼びます。リクエスト数の予測ができない場合はオンデマンドキャパシティーモードが便利ですが、リクエスト量が多いと予想以上に利用料が跳ね上がることもあるので注意が必要です。

## DynamoDB の保守

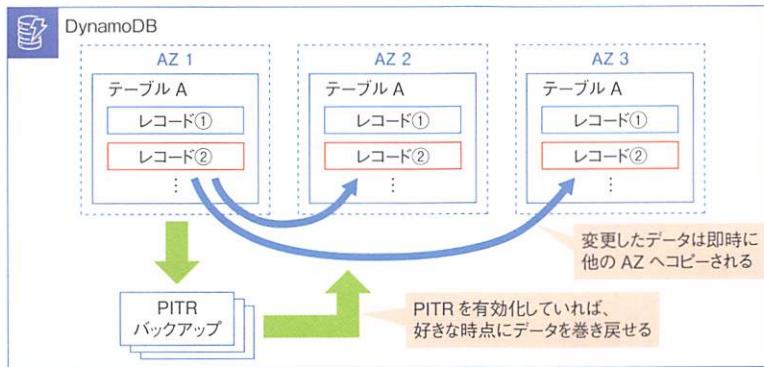
DynamoDBにおいて、利用者の所有物はデータ（テーブル）のみとなります。データを利用するための仕組み（リクエストの受け口やストレージ）はAWSから提供され、利用者のキャパシティーユニットやデータの暗号化の指定に応じて動作します。そのため、利用者はテーブルの構造の検討やバックアップ計画といった、データのメンテナンスのみを考えればよいようになっています。

### DynamoDB の利用者はテーブルのことだけ考えればよい



DynamoDBのデータはAWS内の仕組みにより、冗長化されて保管されます。具体的には、3つのAZ（アベイラビリティゾーン）ヘリアルタイムで複製され、どこかのAZで障害が発生してもデータは無事かつサービスも継続できるようになっています。また、バックアップについても**ポイントインタイムリカバリ (PITR)**というリアルタイムのバックアップ機能があります。PITRを有効化しておくことで、過去35日間の任意の時点にテーブルのデータを戻せるようになり、不意のデータ損失に対応できます。また、任意の時点で手動でバックアップを取得することも可能です。手動で取得したバックアップは無期限で保管することができるので、特定の時点のテーブルの内容を保存したいときに便利です。

## DynamoDB のデータが損失しないための仕組み



DynamoDB は API での操作のほか、マネジメントコンソールからでもデータを表示したり、編集したりすることができます、容易にデータの保守が行えるようになっています。バックアップの取得・管理も数クリックで行えます。

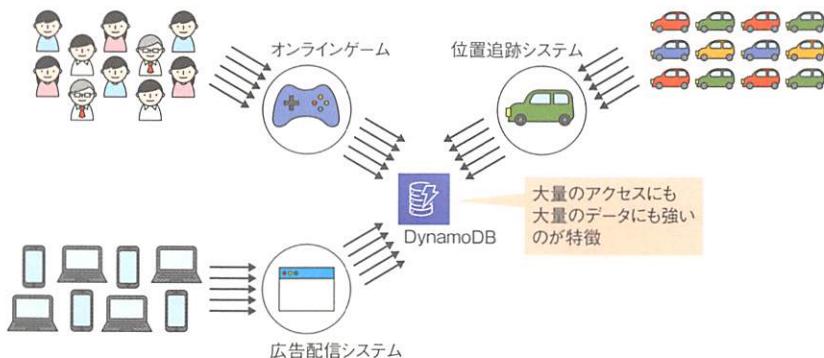
## DynamoDB の利用シーン

ここまで説明したような特徴から、DynamoDB は、1 日に 10 兆件以上、毎秒 2,000 万件以上のリクエストをそれぞれ 0.0 何秒という速さで処理できるとされています。その強みを生かして、データのやりとりが非常に多く、かつユーザーへの

高速な応答が必要となるモバイル、Web、ゲーム、広告、IoTなどのアプリケーションに利用されています。

具体的には、IoT技術を利用した車両の位置追跡システム、ゲームのランキングシステム（リーダーボード）、Web広告のクリック数管理といったものが挙げられます。

### 大量のアクセスを高速で処理する必要のあるアプリケーション向き



# 05 データ分析のために 大量データを取り込む

**keyword** • Amazon Redshift

## Amazon Redshift は分析用データベース

Amazon Redshift (以下 Redshift) は、データの分析に使用するデータウェアハウスサービスです。ウェアハウスは倉庫という意味で、大量のデータを格納しておく用途で使用されるため、データウェアハウスと呼ばれます。

S3 や RDS などの他の場所にある大量のデータを Redshift に取り込み、データエンジニアやデータサイエンティストと呼ばれるデータ分析を行う担当者が、SQL やビジネスインテリジェンス (BI) ツールを使用してデータにアクセスします。

Redshift や多くの分析用データベースは、列指向ストレージという構造になっています。具体的な例を見てきましょう。たとえば次のような学生のデータがあったときに、平均点を分析して算出したいとします。

学生番号	氏名	点数	性別	出身地
001	山田太郎	90	男	千葉県
002	上野史瑛	86	男	東京都
003	田中花子	77	女	北海道
004	クラウド次郎	91	男	米国

注目すべきは学生ごとの行ではなく、点数という列であることがわかります。このように通常、分析は列単位で行うため、列ごとにデータを保存しておくと分析がやりやすくパフォーマンス効率が良くなります。

RDS と同様、データへのアクセスは SQL を使用します。

## 分析用のデータは列ごとに保存しておくと効率が良くなる

データを分析する場合(平均算出など)、  
列単位で処理が行われる

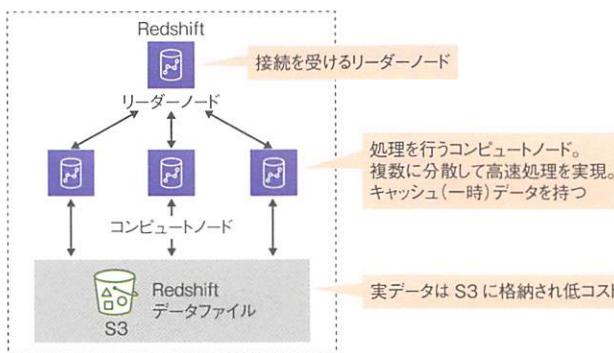
学生番号	氏名	点数	性別	出身地
0001	山田太郎	90	男	千葉県
0002	上野史瑛	86	男	東京都
0003	田中花子	77	女	北海道
0004	クラウド次郎	91	男	米国



Redshift は、SQL 接続を受け付けるリーダーノードと、ストレージおよび SQL 文の実行を行うコンピュートノードの 2 種類のノードから構成されます。コンピュートノードは複数存在し、処理を分散して高速に実行します。

最新の Redshift では RA3 というノードタイプが使用されますが、この構造ではキャッシュ（一時）データをコンピュートノードに持ち、実データは S3 バケットに保存されます。この S3 バケットは Redshift が管理するもので利用者からは見えません。S3 は低コストという特徴を持つため、キャッシュデータで高速な分析を可能としながら低コストを併せて実現しています。

## Redshift の構成。高速処理と低コストを同時に実現する



Redshift は分析およびデータの読み込みに特化したデータベースのため、SQL の INSERT 文や UPDATE 文を実行して 1 件ずつデータを挿入することは推奨されません。COPY コマンドを使用して、S3 バケットから複数のテキストデータを並列でアップロードすることが推奨されています。S3 の他にも、DynamoDB、EC2 などの外部サーバーからの COPY アップロードにも対応しています。

# 06 データベース移行に役立つサービス 2 選

**keyword** • データベース移行 • AWS Database Migration Service  
• AWS Schema Conversion Tool

## データベース移行の重要性

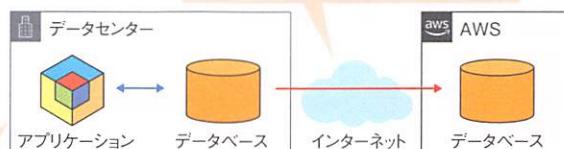
機器の老朽化や処理性能向上のため、アプリケーションをオンプレミス環境から AWS へ移行したり、利用するデータベースエンジンを異なるものに変更したりすることは珍しいことではありません。アプリケーションを AWS や新しいデータベースエンジンに対応するよう改修できたとしても、現在稼働しているサービスのデータベースをどうやって新しいデータベースに移すのかということが大きな課題となってきます。

アプリケーションの移行において、データベースの移行は最大の難関であり、移行時のダウンタイム（サービス中断時間）を短くするカギとなります。

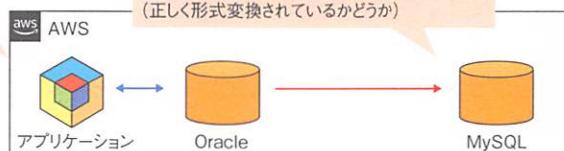
## データベース移行にはさまざまな困難がある

- どちらの場合も、
- ・データの量が多いほど転送に時間がかかる
- ・移行中はアプリケーションからの更新を止めないと移行先のデータとの不整合が発生する

オンプレミスからの移行の場合は、  
ネットワークの速度・安定性の確保が必要  
(全データがちゃんと転送できているかどうか)



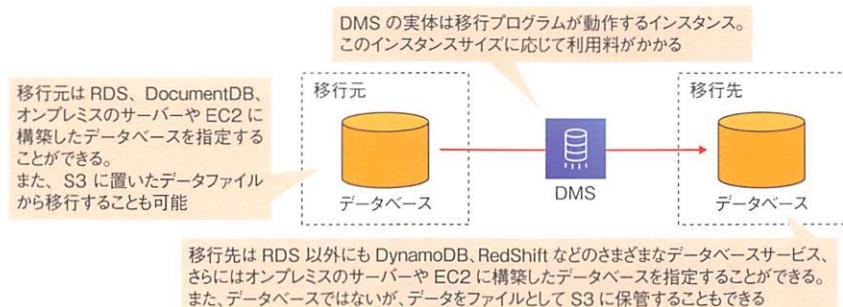
異なるデータベースエンジンへの移行の場合は、  
互換性や網羅性の確認が必要  
(正しく形式変換されているかどうか)



## AWS Database Migration Service

**AWS Database Migration Service (以下 DMS)** は、その名のとおりデータベースを移行するためのサービスです。現行のデータベースから新しいデータベースへデータを移行する役割を担います。単純にデータを移行するだけではなく、データの変換を行いつつ移行したり、移行中に発生した移行元データベースの更新に追従してデータを移行したり、さらには複数のデータベースを1つのデータベースに集約して移行することもできます。また、移行後に正しく移行できたかを検証する機能もあります。

### DMS はデータベースの移行のためのサービス



## AWS Schema Conversion Tool

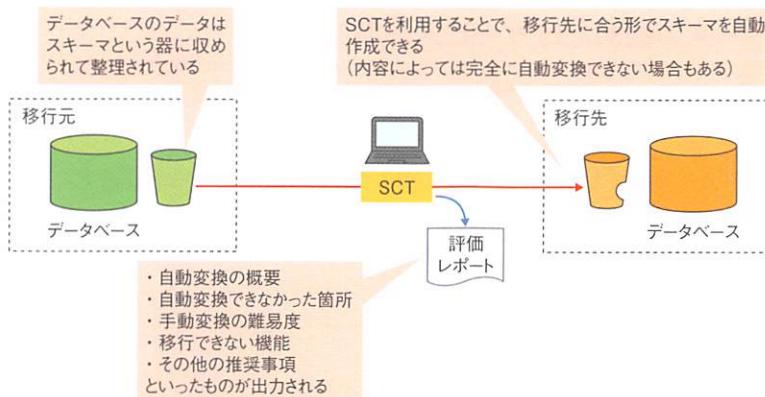
データベースにおけるデータ構造を **スキーマ** といいます。Oracle や MySQL などデータベースエンジンの種類ごとにスキーマの形式は異なり、スキーマは **データベースの設計図**とも呼ばれます。「データベースエンジンごとにデータの入れ物の形が違う」と表現するとわかりやすいかもしれません。

そのため、異なるデータベースエンジンへデータを移行する際は、データ自体の移行の前に、まずスキーマを移行元の形式から移行先の形式へ変換する（つまり、移行先の入れ物を移行元データの形に合うように作り変える）という作業が必要となります。このスキーマの変換を行うためのツールが **AWS Schema Conversion Tool (以下 SCT)** です。なお、SCT は AWS のサービスではなく、パソコンにインストールして使うアプリケーションです。Windows のほか、macOS や Linux に対応しており、SCT を移行元と移行先のデータベースに接続させることで自動的にス

キーマの変換を行うことができます。

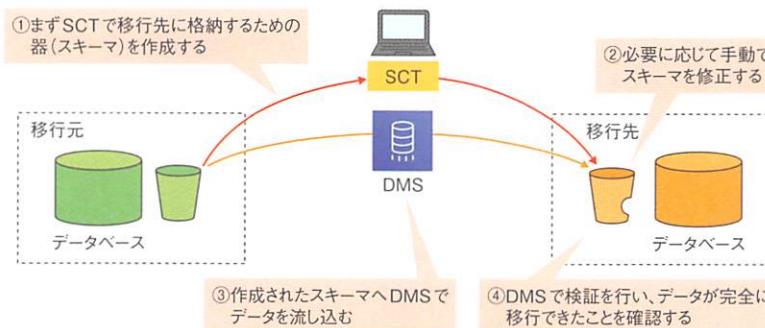
また、機能のひとつとして、スキーマ変換における評価レポートを作成することができます。必ずしも自動的にすべてのスキーマが変換できるとは限らないので、評価レポートには自動変換できないために手動変換が必要な箇所やその労力予測、データベースの機能として移行先で実現できない機能といった観点から判断した移行可能性の評価が提示されます。

## SCT はデータベースのスキーマ変換のためのアプリケーション



「Oracle から MySQL へ移行する」といった異なるデータベースエンジンへの移行は、次図のように SCT と DMS を併用して行うことになります。

## 異なるデータベースエンジンへの移行時は SCT と DMS を併用する



## 07

多種多様なデータベースに  
対応するサービス7選

## keyword

- Amazon ElastiCache
- Amazon MemoryDB
- Amazon DocumentDB
- Amazon Neptune
- Amazon Quantum Ledger Database
- Amazon Managed Blockchain
- Amazon Timestream

## データベースの種類だけサービスがある？

データベースとひとくちに言ってもデータの構造は多種多様で、それに合わせてデータベースの種類も多数存在します。リレーションナルデータベース（RDB）以外のデータベースは一般にまとめて **NoSQL (Not only SQL)** とも呼ばれます。

AWS ではそれらさまざまな種類のデータベースに対応できるよう、多くの種類のデータベースサービスを提供しています。

## Amazon ElastiCache

**Amazon ElastiCache**（以下 ElastiCache）は、**インメモリデータストア**を提供するサービスです。インメモリデータストアとは、サーバーの主記憶装置（メモリ）にデータを格納するデータベースの一種で、SSD や HDD といった補助記憶装置にデータを格納するよりも高速にデータを読み書きする仕組みです。「データ読み込みを高速化するため、よく使うデータを一次保存する技術」であるキャッシュに利用されることが多く、**インメモリキャッシュ**とも呼ばれます。

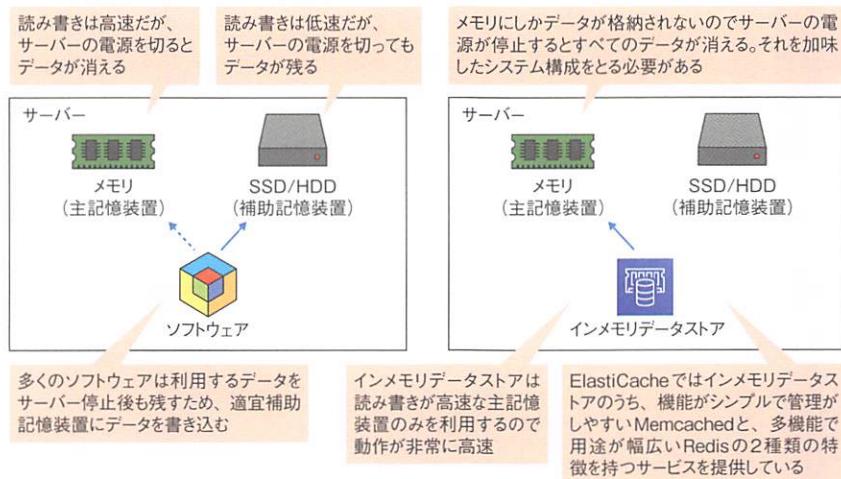
先に書いたように非常に高速で、ElastiCache は 0.001 秒未満の速さで読み込みおよび書き込みリクエストに対応できるようになっています。ただし、メモリ上のデータはサーバーを停止するとすべて消えてしまうため、データが消えてもよいような一時的なデータ保存先として利用されることを想定しています。

現在はインメモリデータストアとして広く利用されている Memcached と Redis というソフトウェアと互換性のある、**ElastiCache for Memcached** と **ElastiCache for Redis** の 2 種類のサービスが提供されており、どちらも 6-4 節（p.186）で説明した key-value 型のデータベースを提供します。

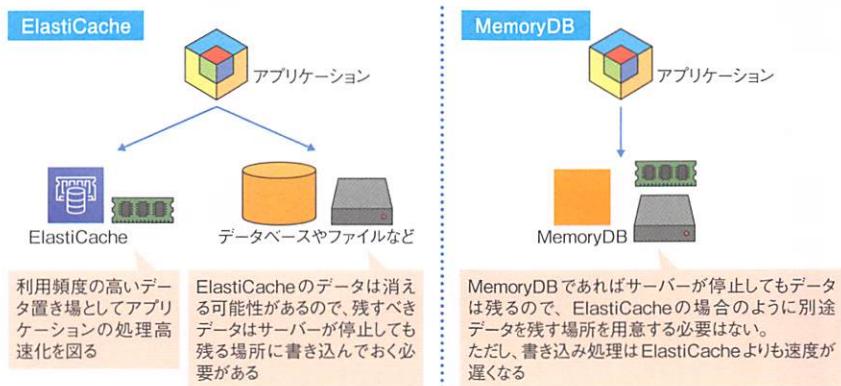
## Amazon MemoryDB

2021年8月に発表された新しいサービスです。ElastiCacheは高速である代わりにデータが消える可能性があるという特徴を持っていますが、ElastiCacheに採用されているRedisはもともと、データをメモリだけでなく補助記憶装置にも書き出すことで、データが消えてもすぐに復旧できる機能を備えているソフトウェアでした。そこで、データの永続性を確保しつつ、高速な読み書きを実現するサービスとしてAmazon MemoryDB for Redisが提供されるようになりました。

### インメモリデータストアはメモリにだけデータを格納する



### ElastiCacheとMemoryDB それぞれの特徴



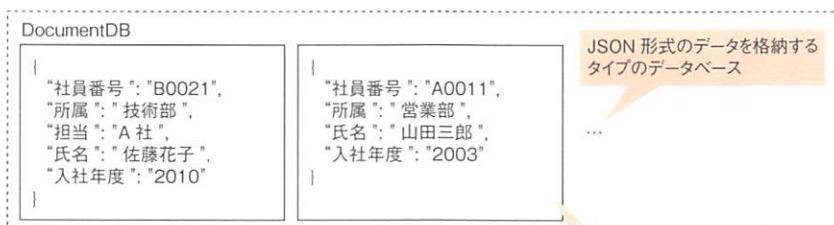
なお、データを保管するための処理が多くなるため、読み込みリクエストに対しては ElastiCache 同様 0.001 秒未満の速さで対応できますが、書き込みリクエストに対しては ElastiCache より少し遅く 0.00 数秒を要します。

## Amazon DocumentDB

Amazon DocumentDB（以下 DocumentDB）は、ドキュメント指向データベースを提供するサービスです。DocumentDB は、MongoDB というソフトウェアとの互換性がある作りになっています。ドキュメント指向と呼ばれるとおり、文書データ（ドキュメント）を格納することに長けたデータベースで、データを JSON 形式で保管し、その内容を検索することができます。

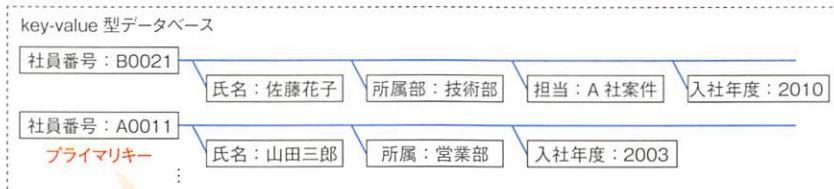
JSON 形式も key と value の組み合わせで表現されるデータであるため key-value 型データベースに似ているのですが、key-value 型データベースはプライマリキーに指定したキーからレコードを探すのに対し、ドキュメント指向データベースは JSON 形式のデータのどの要素からでもレコードを検索できるのが特徴です。

### DocumentDB はドキュメント指向データベース



ドキュメント指向データベースではドキュメントのどの要素からでも検索できる

社員番号が○○のデータは? 所属が××で入社年度が△△のデータは?



key-value 型データベースではプライマリキーでしか検索できない

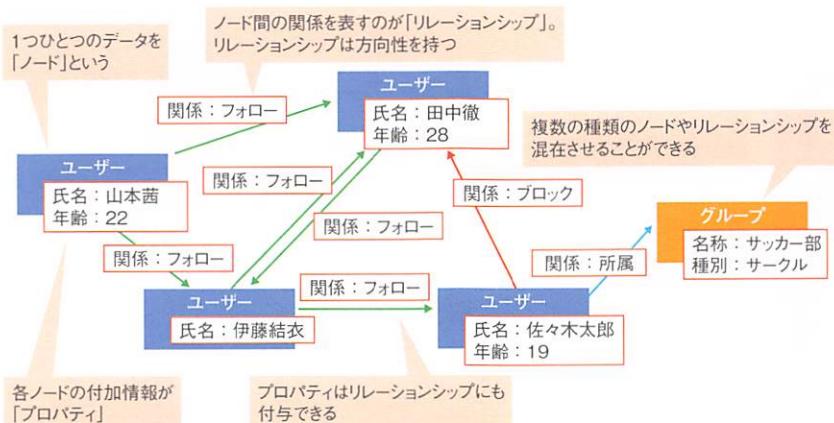
社員番号が○○のデータは?

## Amazon Neptune

Amazon Neptune (以下 Neptune) は、グラフデータベース（グラフ構造を持ったデータを格納するためのデータベース）を提供するサービスです。グラフ構造とは、データ同士の関係性を表すことに特化した構造で、1つひとつのデータを表す「ノード」と、関係を表す「リレーションシップ」、ノードおよびリレーションシップの内容を表す「プロパティ」の3つの要素からなります。

具体的な例としてはSNSのユーザーの関係性を表すような場合が挙げられます。誰と誰がフォローし合っていて、誰がブロックされているかというような関係性の情報は、グラフ構造を利用することで簡単に扱えるようになります。

### Neptuneはグラフ構造を持ったデータを格納するのに適している



## Amazon Quantum Ledger Database

Amazon Quantum Ledger Database (以下 QLDB) は、台帳データベースを提供するサービスです。名前に台帳と付いているとおり、ジャーナルと呼ばれるログデータに記録されるデータの変更履歴を次々と記録していくことでデータを管理します。ジャーナルは追記のみが可能で、変更や削除ができない仕組みになっており、信頼性の高い作りになっています。

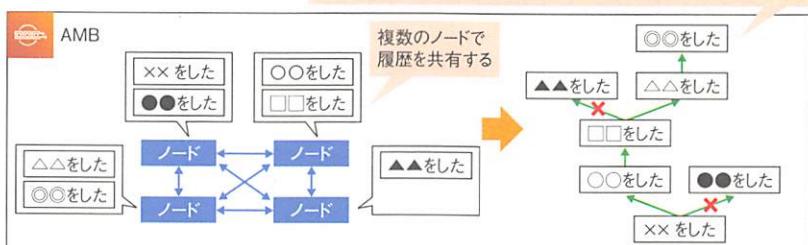
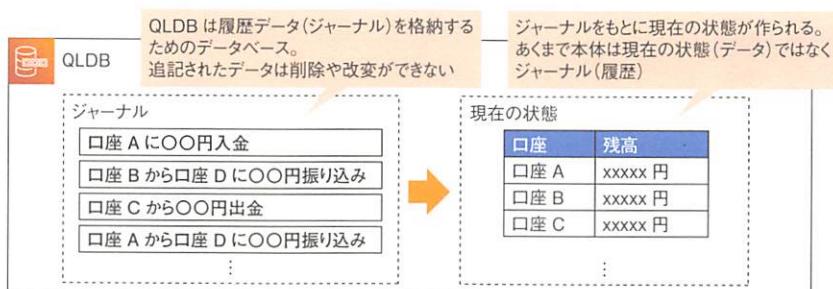
現在のデータそのものだけでなくデータの変遷の履歴が重要となる、銀行取引や製造履歴管理、保険料請求処理といったシステムが用途として考えられます。

## Amazon Managed Blockchain

Amazon Managed Blockchain (以下 AMB) は、ブロックチェーンネットワークを構築するサービスです。ブロックチェーンとは分散型台帳技術の一種で、データの変更履歴を記録する台帳を、ネットワークに属する複数のサーバーで協力して管理する仕組みです。台帳を記録するデータベースとして先に説明した QLDB と似ているのでここで説明していますが、一元的にデータを集約して管理する QLDB とは異なり、データを複数のサーバーに分散させて保持する技術なので、正確にはデータベースではありません。

「1カ所でデータを集中管理する必要性の少ない履歴」を管理する用途に向いており、例としては個人間の金融取引や貿易における業者間の輸出入履歴の管理といったものが挙げられます。有名なものではビットコインなどの仮想通貨に利用されています。

### 台帳データベースの QLDB と、ブロックチェーンを構築する AMB



## Amazon Timestream

**Amazon Timestream** は、時系列データベースを提供するサービスです。センサーなどから送られてくる時間ごとに変化する値を次々と記録することに適したデータベースで、RDB の最大 1,000 倍の速度で、1 日あたり数兆ものイベントを処理できるといわれています。また、時系列データの分析を想定して作られており、データの平滑化、近似、補間機能などを利用した分析を行うことも可能です。

### Timestream はセンサーデータなどの処理に適したデータベース

2つの温湿度計の時系列データを格納する場合の例

ID	時刻	温度	湿度
0001	2021/09/28 23:00	24°C	69%
0002	2021/09/28 23:00	22°C	63%
0001	2021/09/28 23:01	23°C	69%
0002	2021/09/28 23:01	23°C	64%
	:		

RDB であればこのように機器ごと、時刻ごとにデータを格納する形になる

時系列データベースでは、このように各時刻につき 1 つの値が対応するように構成され、特定のデータの時系列の変化が追いややすくなる。

また、指定時間幅のデータの平均値や最大値の算出、値の合計など、時系列データを扱うための機能が備わっている



ID	時刻	温度
0001 : 温度	2021/09/28 23:00	24°C
	2021/09/28 23:01	23°C
	:	
0001 : 湿度	2021/09/28 23:00	69%
	2021/09/28 23:01	69%
	:	
0002 : 温度	2021/09/28 23:00	22°C
	2021/09/28 23:01	23°C
	:	
0002 : 湿度	2021/09/28 23:00	63%
	2021/09/28 23:01	64%
	:	

**column**

## 本書で詳細説明していない AWS サービスについて

本書に名前が登場する AWS サービスの中には、一部、本文中で詳細説明をしていないものがあります。簡単ではありますが、ここで概要を説明します。

- **Amazon Simple Email Service (Amazon SES)**

メールを送信するためのサービスです。大量のメール配信が可能となる性能を備えているだけでなく、スパムメールと判定されずに送信先に届くように最適化された送信プロセスなど効率的にメールを配信する機能を持っています。

- **Amazon EventBridge**

アプリケーションや AWS サービスが発生させるイベントを別のアプリケーションや AWS サービスに連携させるサービスです。また、時間指定などにより Amazon EventBridge 自体がイベントを発生させることもできます。

- **Amazon Simple Notification Service (Amazon SNS)**

メッセージ送信を行うサービスで、アプリケーション同士のメッセージ送信や対人向けの SMS 送信、モバイルプッシュ通知、E メール送信を行うことができます。

- **AWS Marketplace**

AWS を使った製品を提供するサードパーティーのソフトウェア、データ、サービスを検索、購入、デプロイ、管理するためのカタログサービスです。利用者は AWS Marketplace を通してサードパーティーの AWS 製品を購入し、利用することができます。

- **エッジロケーション**

サービス名ではありませんが、CloudFront や Route 53 のようなグローバルに広く展開されるサービスで利用される拠点を示します。リージョンやアベイラビリティゾーンとは別のデータセンターとして設置されています。



# セキュリティ、 アイデンティティサービス

セキュリティについても多種多様なサービスがあります。これらのサービスについてそれぞれの役割を知り、AWS 上のシステムをより安全に守るための方法を学びます。

- section  
**01** AWS におけるセキュリティ  
AWS のセキュリティをしっかりと理解しよう
- section  
**02** AWS Identity and Access Management  
IAM は AWS 利用時のセキュリティの要
- section  
**03** AWS CloudTrail とは  
すべての操作を記録して証跡として残す
- section  
**04** AWS Config とは  
設定履歴や設定内容を自動で管理する
- section  
**05** AWS GuardDuty とは  
AWS の怪しい操作を検知して不正を防ぐ
- section  
**06** AWS WAF とは  
Web アプリケーションのセキュリティを強化する
- section  
**07** その他のセキュリティ、アイデンティティサービス  
システムのセキュリティを強固にするサービス 6 選

# 01 AWS のセキュリティを しっかり理解しよう

**keyword** • 情報セキュリティの 3 要素 • AWS におけるセキュリティの考え方

## 情報セキュリティの 3 要素

システムにおける情報セキュリティとは、保持している情報（データやシステム基盤）を安全に保つ考え方を指します。安全に情報を守るために、情報セキュリティの 3 要素、機密性・完全性・可用性が必要とされています。

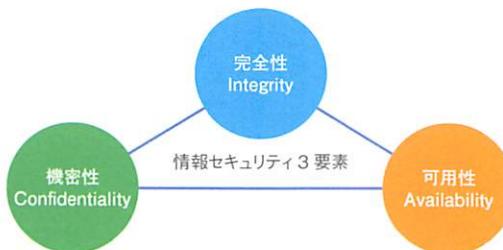
機密性は、**情報が漏えいしないよう管理すること**を意味します。必要なときに必要な人のみが情報を閲覧できるよう、不正アクセス対策を行いながら適切な管理を行います。

完全性は、**情報が正確・最新で欠けていないことを**意味します。情報の変更管理を正しく行い、改ざんなどの不正操作から守る必要があります。

可用性は、**情報を使いたいときに使えるようにしておくこと**を意味します。システム障害などで情報が必要なときに使用できなければ、情報が役立ちません。データが壊れた場合は速やかにバックアップから復旧する必要があります。

まとめると、管理している情報を、適切な人のみが（機密性）、正しい状態で（完全性）、必要なときに（可用性）使用できる必要があるということです。この 3 要素を守るために対策をいろいろ行っていくことになります。

## 機密性・完全性・可用性を守るのが情報セキュリティの考え方



## AWSではいろいろなサービスを組み合わせて情報を守る

AWS 上に構築したシステム、データを安全に管理するにはどうすればよいでしょうか。

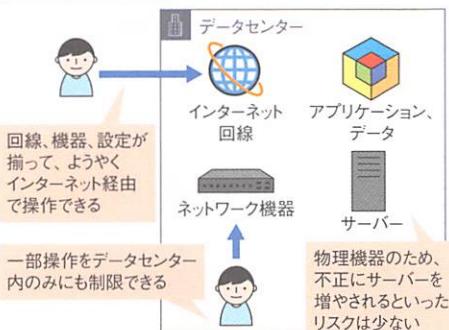
オンプレミスの場合、データセンターにサーバー機器を設置して、設定を行い、インターネット回線を敷設して、やっとインターネット経由でシステムが操作できるようになります。また、勝手にデータセンターに入られて、新たに大きなシステムを勝手に作られるといった心配はないでしょう。

AWS（クラウド）の場合はどうでしょうか。AWS アカウントを作成した直後に、管理者ユーザー（ルートユーザー）でログインして操作を行います。つまり、このユーザー情報が第三者に漏れると、AWS アカウント内で任意の操作を第三者が実行できるようになり、不正アクセスされてしまいます。アカウント作成直後であれば機密情報は無いですが、不正に EC2 などのリソースを大量に作成するといったことは可能です。そのため、誰が、どの操作ができるのかといった設定は慎重に行う必要があります。

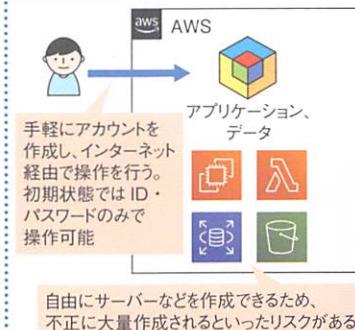
この誰が、どの操作を、という管理は認証・認可の管理になりますが、AWS では IAM というサービスで設定を行います。IAM の詳細は次節で説明します。

## AWSのセキュリティは AWSアカウントの権限を管理することが重要

### オンプレミスの場合



### AWSの場合

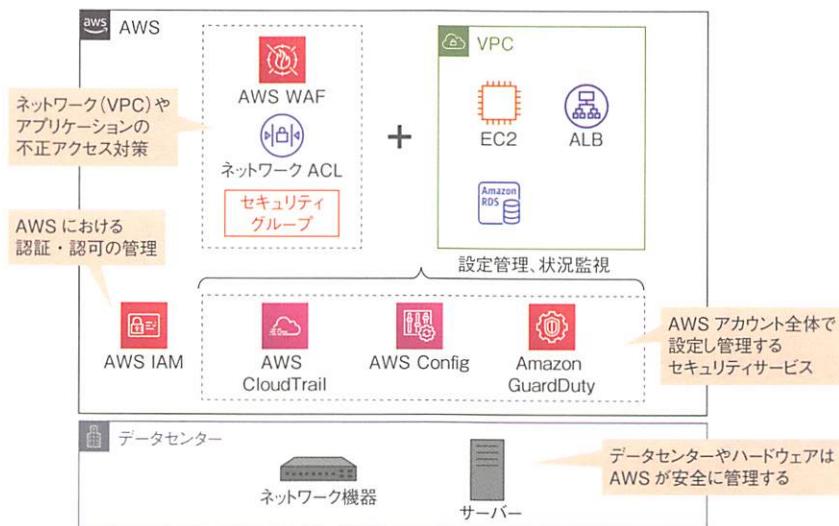


ここまで説明を読むと、AWS のほうがセキュリティ的に危険と思われるかもしれません、決してそんなことはありません。AWS も正しくセキュリティサービス

を活用して管理を行えば、オンプレミス同等またはそれ以上に安全な管理が可能です。また、データセンターやハードウェアの管理は AWS によって安全に管理されています。利用者側がオンプレミス環境で AWS と同等のセキュリティを確保しながら、複数の場所の管理を行うのはとても大変で難しいです。

AWS にはセキュリティサービスや機能が多くあり、初学者には難しいところがあるのも事実です。本章で重要なセキュリティサービスを一通り紹介するので、そこから AWS の安全な使い方を学んでいってください。

## たくさんのセキュリティサービスを一通り、ざっくり学びましょう



## 02 IAM は AWS 利用時のセキュリティの要

**keyword** •AWS IAM •IAM ユーザー •IAM グループ •IAM ポリシー •IAM ロール

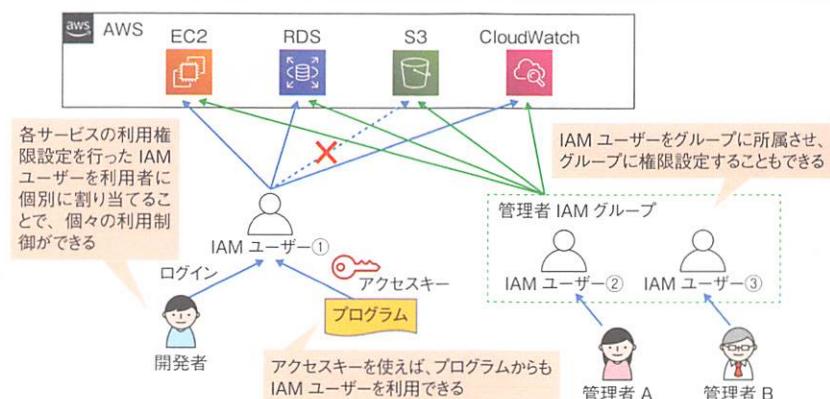
### AWS Identity and Access Management とは

AWS Identity and Access Management (以下 IAM) は、AWS サービスを利用する際の権限管理を行うサービスです。とてもざっくり説明すると、AWS を使うためのユーザーを作り、そのユーザーに「どのサービスのどの機能を利用できるか」という権限を設定することができます。ただ、権限をユーザーではなくサービスに与えたり、一時的な権限を付与したりと、単純な権限管理にとどまらないのが IAM の特徴であり、難しいところです。AWS サービスの操作制御はすべて、このサービスによって管理されます。設定内容は非常に複雑になりますが、概要さえ理解しておけばおおまかな動きやできることを把握することができるです。

### IAM ユーザー、IAM グループ

利用者が AWS を操作するために使うのが IAM ユーザーです。パスワードなどの

### IAM ユーザーを作成して、サービスの利用権限を割り当てる



認証情報を使ってログイン（認証）すると、IAM ユーザーに割り当てられた権限が利用できる（認可）ようになります。

IAM ユーザーはマネジメントコンソールでの操作に使うほか、**アクセスキー**という認証情報を発行してプログラムなどに権限を使わせることも可能です。同じ権限を持つ IAM ユーザーをまとめて管理したいときは、IAM ユーザーを **IAM グループ**というグループに所属させ、IAM ユーザー個別ではなく IAM グループに対して権限を付けることもできます。

## IAM ポリシー

AWS サービスに対する権限設定として、「どのサービスのどういった機能に対して何の操作ができるか」を定義したものが **IAM ポリシー**です。複数のサービスに対する権限設定をひとまとめにして名前を付けて定義できるため、同じ権限設定を使い回すことも可能です。また、1 つの IAM ユーザー、IAM グループ、IAM ロールに対して複数の IAM ポリシーを紐付けることができます。

AWS は日々機能追加やサービス追加が行われているため、人の手で新しい機能やサービスの権限設定を追加していくのは大変です。そこで、AWS が用意する **マネージドポリシー**というものが存在します。マネージドポリシーには「すべてのサービスを使用可能」や「すべてのサービスの参照のみ可能」、「S3 のすべての機能を利用可能」といったものが用意されており、サービスや機能が追加されたときは AWS 側で自動的に権限を追加してくれるため便利です。

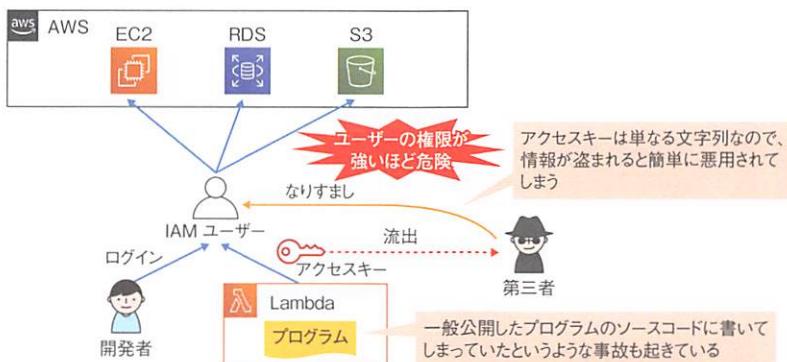
ただし、細かな権限設定をしたい場合は、利用者が独自に IAM ポリシーを作成する必要があります。利用者が作成した IAM ポリシーは、マネージドポリシーに対して **カスタムポリシー**と呼ばれます。AWS のサービスや機能は非常に多く、カスタムポリシーの設定も複雑になりがちですが、次図のように AWS マネジメントコンソールでは項目を選択していくことで IAM ポリシーが作成できる機能も提供されています。



## IAM ロール

IAM ユーザーは利用者に割り当てられますが、「Lambda から EC2 を起動する」というように「AWS サービスが AWS サービスを操作する」という使い方をしたい場合は、**アクセスキー**をプログラム内で利用することで IAM ユーザーに割り当てた権限を使わせることができます。しかし、アクセスキーは固定の文字列であるため、悪意のある第三者に内容が漏れると悪用されるリスクがあります。

### アクセスキーで IAM ユーザーの権限を使わせることの危険性

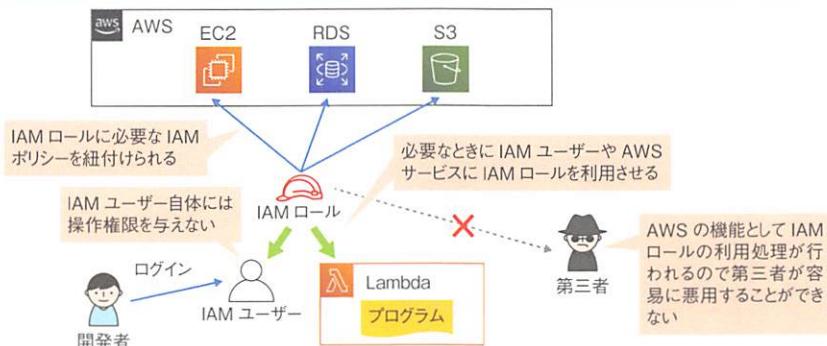


そのため、AWS サービスに権限を与えるときは **IAM ロール** を使うことが推奨されています。

IAM ロールとは、名前のとおり役割を与える機能で、AWS サービスや IAM ユーザーに一時的に IAM ポリシーを付けることができます。IAM ロールを表す AWS の公式アイコンにはヘルメットのデザインが使われているのですが、必要なときに IAM ポリシーを帽子のように対象にかぶせて権限を与えると考えるとわかりやすいかもしれません。

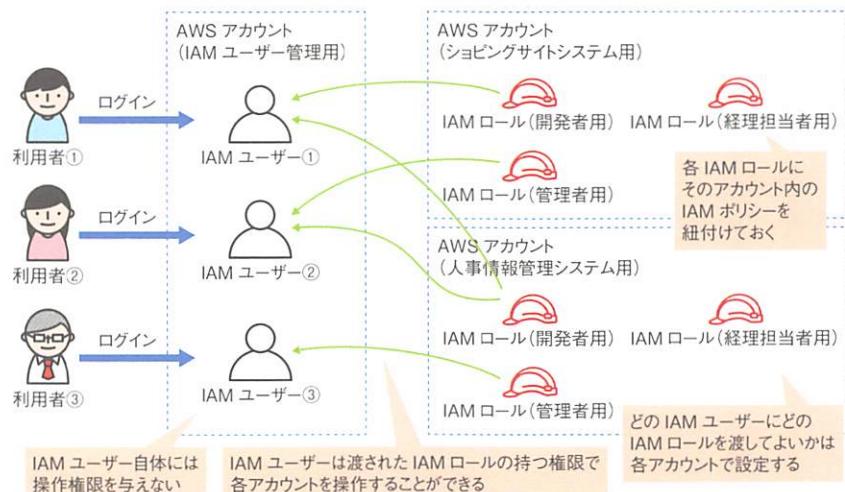
IAM ユーザーのアクセスキーのように「何らかの情報を知っていれば利用できる」というものではなく、IAM ロールは事前に「どの IAM ユーザー、AWS サービスに利用させる」ということを設定しておき、利用要求時にその設定に基づいて利用のための認証が行われるので、第三者に簡単に悪用されることもありません。

## IAM ロールで必要なときにのみ権限を与えるようにする



AWS ではシステムを構築する際、環境の分離、管理の分離、課金の分離といった観点から、構築するシステムごとに AWS アカウントを分離することが推奨されています（p.223 のコラムも参照）。AWS アカウントには IAM ユーザーを利用してログインしますが、AWS アカウントが多数あると各 AWS アカウントに利用者個人個人の IAM ユーザーを作成して管理するのは大変です。そこで、各 AWS アカウントには利用者の役割ごとの IAM ロールのみを作成しておき、利用するたびに都度 IAM ロールを利用させることで、利用者に適切な権限での操作を提供するようにします。この仕組みを **スイッチロール** といい、この機能を利用することで複数アカウント管理における IAM ユーザー管理および権限管理が非常にやりやすくなります。

## システムごとの AWS アカウントの IAM ロールを利用時に割り当てる



# AWS CloudTrail とは

## 03 すべての操作を記録して証跡として残す

keyword • AWS CloudTrail • Amazon CloudWatch Logs

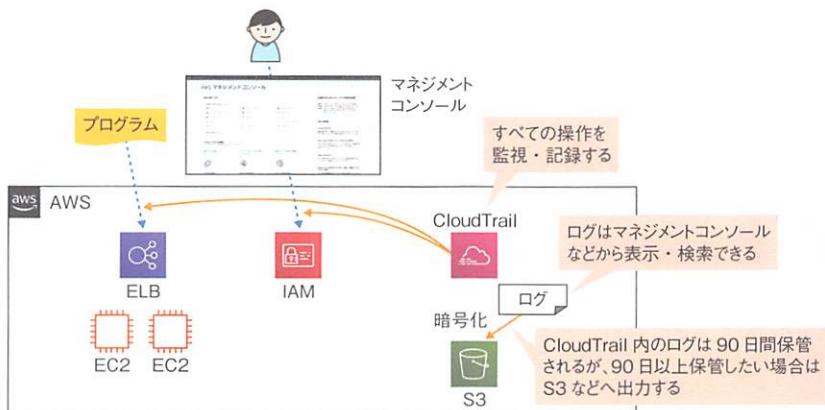
### AWS CloudTrail とは

AWS の利用状況を管理するうえで、「誰が、いつ、何に対して、何をしたか」ということを記録しておくことは非常に重要です。AWS CloudTrail（以下 CloudTrail）は AWS アカウントの作成時点から自動ですべての操作を記録するサービスです。

マネジメントコンソールやプログラムからの操作、AWS サービスにより実行されるアクションすべてを記録し蓄積します。これらの情報から AWS に対して誰がどういった操作をしたのかということを詳しく追うことができ、利用者による不正な操作やプログラムの想定外の動作による思わぬ変更などの調査に役立ちます。

ログは自動的に 90 日間保管されますが、設定により S3 などへログをファイルとして保管することも可能です。そうすることで 90 日を超えてログを保管することができます。S3 などへ保管するログファイルは「証跡（Trail）」と呼ばれ、暗号化されて保管されます。また、保管後のログの改ざんを防ぐ機能もあり、何か問題が

### AWS アカウント作成時からすべての操作が記録されている



発生したときの当時の操作記録を証明するための重要な記録となります。CloudTrail はいわば AWS における監視カメラといえるでしょう。

## 記録される操作

「すべての操作を記録する」と書きましたが、具体的には AWS の機能であるマネジメントコンソールや API といったものを通した操作が記録の対象となります。それに該当しない操作、たとえば SSH やリモートデスクトップを利用して EC2 インスタンスへログインしてから行う操作のようなものは記録することができません。具体的には、次のような操作が記録されます。

### ① 管理イベント

マネジメントコンソールへのログイン、EC2 インスタンスや S3 バケット、Lambda 関数など AWS リソースの自体の作成、変更、削除といった操作

### ② データイベント

S3 バケット内のデータ操作（作成、編集、削除）、Lambda 関数の実行といった AWS リソースへの操作

### ③ インサイトイベント

AWS アカウントの操作において、普段とは異なる異常な操作

（普段の CloudTrail のログから典型的な使用パターンを学習し、そこから逸脱したパターンの操作を検出する機能）

なお、デフォルトではこのうち①の管理イベントのみが記録されるようになっています。②、③のデータイベント、インサイトイベントは設定を有効にすることで記録されるようになります。

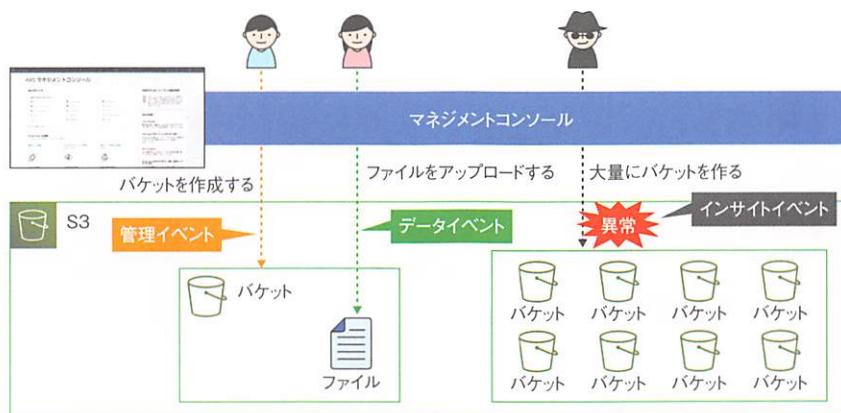
管理イベントは CloudTrail 内に保管される 90 日分のログと 1 つ目の証跡の出力に対しては料金がかかりません。2 つ目以降の証跡の出力の際には、発生したイベントの数に応じて料金が発生します。具体的には東京リージョンで 100,000 管理イベントあたり 2.00USD の料金がかかります。

データイベントは CloudTrail 内には保管されず証跡としてのみ出力されます。管理イベント同様、発生したイベント数に対して料金が発生し、東京リージョンでは

100,000 データイベントあたり 0.10USD となります。

インサイトイベントは検出された異常な操作のみが証跡として出力されます。これらは分析対象のイベント数に応じた課金となります。東京リージョンでは分析対象となった 100,000 イベントあたり 0.35USD がかかります。

## 記録される操作。管理イベント以外は設定が必要



## 証跡の出力

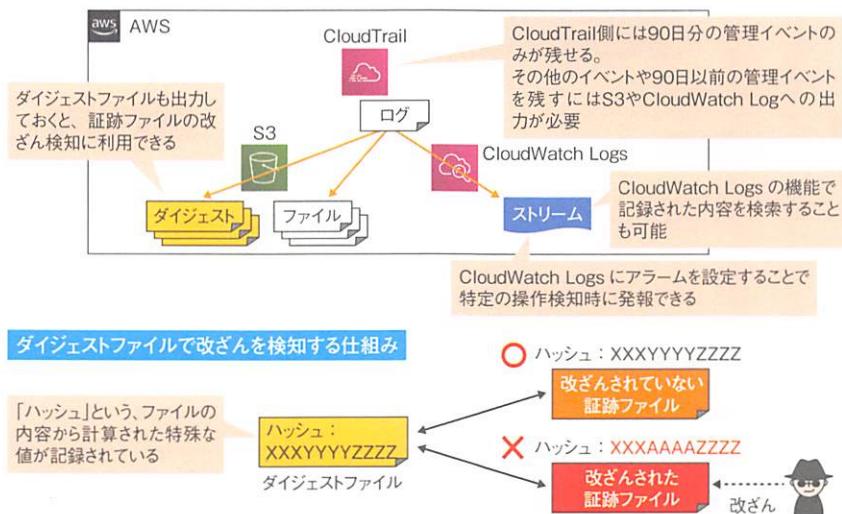
CloudTrail はログを「証跡」として、S3 および CloudWatch Logs に出力することができます。どちらのサービスも保管のための料金は発生しますが、出力されたログを無期限で保存できます。

S3 に保管する場合は、ファイルとして保管されることとなります。監視カメラに例えるなら、撮りためた動画ファイルを大容量のレコーダーに蓄積していくような状態になります。操作履歴を確認するときは、それらのファイルを見ることになります。また、いったん保管されたファイルが悪意のある利用者に改ざんされるというリスクを想定して、CloudTrail が保存したファイルの整合性チェックファイル（ダイジェストファイル）を出力することも可能です。

CloudWatch Logs に保管する場合は、「ストリーム」という形式で保管されるになります。ストリームとは、ログの内容が時系列順に記録されていくものです。監視カメラに例えるなら、撮影された動画がモニターに流され続けているような状

態です。操作履歴を確認するときは、ほぼリアルタイムで流れてくるストリームの内容を確認したり、過去に記録されたストリームの内容を CloudWatch Logs の管理画面上で検索したりすることができます。さらに CloudWatch Logs には、ログの内容をチェックし、特定の文字列が見つかったときにイベントを発生させる機能があります。この機能を利用することで、特定の操作が発生したときに通知を行うことが可能です。

## ログ（証跡）を 90 日を超えて残しておく



## 04

## 設定履歴や設定内容を自動で管理する

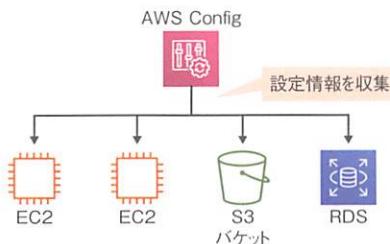
**keyword**

- AWS Config
- Amazon Config ルール
- Amazon EventBridge
- Amazon SNS
- Automation

## 設定履歴を残す AWS Config

AWS Config は EC2 インスタンスやセキュリティグループなど、AWS 上のリソースの設定情報とその変更履歴を残してくれるサービスです。画面上から AWS Config を有効にするだけで、自動的にサポートされる AWS リソースの設定情報が収集され、履歴管理されるようになります。デフォルトでは過去 7 年分の情報が保存されます。

## AWS リソースの設定情報と変更履歴を残しておける



次図は、セキュリティグループの変更内容を表示したものです。いつ、どのような設定だったのか、画面上から確認できます。

The screenshot shows the AWS Config console interface. On the left, there's a navigation pane with "設定履歴" (Change History) selected. The main area displays a table with columns for "ファイル" (File), "リソース" (Resource), and "詳細" (Details). One row is highlighted, showing a security group named "sg-01234567890abcdef". The "詳細" column for this row shows a JSON-like diff output:

```

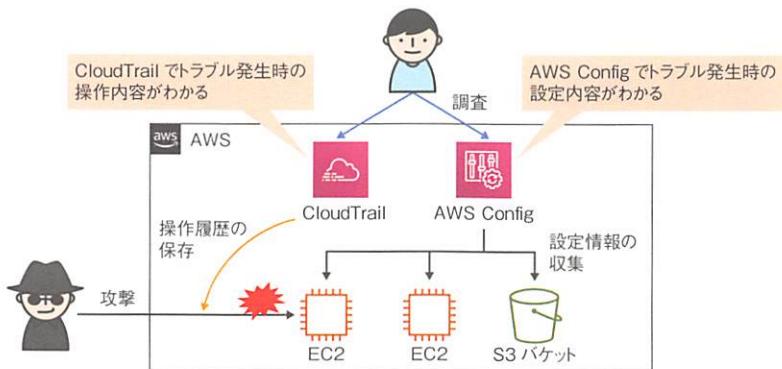
{
  "changes": [
    {
      "changeType": "addObject",
      "resource": "sg-01234567890abcdef",
      "configuration": {
        "description": "新規セキュリティグループ"
      }
    },
    {
      "changeType": "removeObject",
      "resource": "sg-01234567890abcdef",
      "configuration": {
        "description": "初期状態"
      }
    }
  ]
}
  
```

A callout box points to this JSON output with the text "セキュリティグループの設定情報がわかる" (You can see the configuration information of the security group).

前節の CloudTrail では、誰が、どの操作を行ったのかという操作ベースで履歴情報が残りますが、AWS Config は **AWS リソースの設定内容ベース** で履歴情報が残ります。両方とも重要な履歴情報となるため、AWS アカウントを作成したら基本的には有効にしておきたいサービスです。

こういった履歴情報を残しておくことで、トラブルシューティングに役立ちます。クラウドにおける不正アクセスなどのトラブルは、設定変更をきっかけに発生することが多いため、トラブル発生原因の特定や影響調査に大きく役立ちます。

## CloudTrail と AWS Config をトラブルシューティングに役立てる



**高度なクエリ** という機能もあり、これを使用することで特定のリソース数の確認も可能です。たとえば以下のクエリ（SQL 文）では、インスタンスタイプ別の EC2 の数を確認できます。

```
SELECT configuration.instanceType, COUNT(*)
WHERE resourceType = 'AWS::EC2::Instance'
GROUP BY configuration.instanceType
```

AWS Config の料金は 1 つの設定項目あたり 0.003USD となっており、安い料金ではありますが、リソース数が増えると利用料も増えていくため注意が必要です。特定のリソースタイプを指定して履歴情報を残すことも可能なので、料金が気になる方はリソースタイプの指定を検討してもよいでしょう。

## 設定内容をチェックする AWS Config ルール

AWS Config ルールは、AWS Config の一機能であり、AWS 内の各設定がルールに準拠しているかチェックできます。たとえば、以下のようなルール（チェック項目）が設定できます。

- EBS が暗号化されているか
- CloudTrail が有効になっているか
- S3 バケットがパブリック読み書き可能になっていないか
- セキュリティグループで SSH ポート（22 番）がパブリック公開されていないか

S3 バケットに関するルールであれば、AWS Config ルールの画面上に AWS アカウント内の S3 バケットが一覧で表示され、それぞれルールに準拠か非準拠かが表示されます。

対象のリソースを選択  
対象範囲内のリソースは、このルールが適用されているリソースとそのコンプライアンス状況を表示します。

リソース ID	リソースタイプ	リソースのコンプライアンス状況	アクションステータス
aws-s3-bucket-12345678901234567890	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567891	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567892	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567893	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567894	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567895	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567896	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567897	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567898	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567899	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567890	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567891	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567892	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567893	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567894	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567895	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567896	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567897	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567898	S3 Bucket	準拠	評議なし
aws-s3-bucket-12345678901234567899	S3 Bucket	準拠	評議なし

公開されているバケットがあれば非準拠と表示される

ルールにはあらかじめ AWS から用意されているものが多くあり、これをマネジドルールと呼びます。利用者独自のルールも作成でき、これをカスタムルールと呼びます。まずは簡単に設定できるマネージドルールを検討するとよいでしょう。

ルールタイプの選択

AWSによって管理されるルールの追加  
ニーズに合わせて以下のルールをカスタマイズします。

カスタムルールの作成  
カスタムルールを作成し、AWS Config に追加します。各カスタムルールを AWS Lambda 関数に発達します。この関数は、SNS リスナーがリードした

AWS マネージド型ルール

名前	ラベル	説明
cloud-trail-log-file-validation-enabled	CloudTrail, Periodic	Checks whether AWS CloudTrail creates a signed digest file with logs. AWS recommends that the file validation must be enabled on all trails. The rule is noncompliant if the validation is not enabled.
cloudformation-stack-drift-detection-check	CloudFormation	Checks whether your CloudFormation stacks' actual configuration differs, or has drifted, from its expected configuration.

マネージドルールは、用意されているものから選択して作成できる

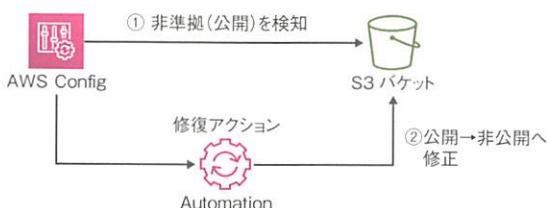
AWS Config ルールで非準拠のリソースが検知された場合に通知することも可能です。Amazon EventBridge、Amazon SNS というサービスを組み合わせることで、利用者へメール通知ができます。本書で Amazon EventBridge、Amazon SNS の詳細は紹介していませんが、ここでは AWS Config ルールと他のサービスを組み合わせて通知ができると理解していただければ大丈夫です。

### ルールに準拠しない使われ方をしていたらメールで通知



非準拠となったリソースを自動的に修正してくれる自動修復機能も AWS Config ルールには存在します。修復アクションという形で AWS から修復内容がいくつか用意されているので、利用者はそこから修復内容を選択すれば OK です。たとえば公開状態になった S3 バケットを、自動的に非公開状態に修正するといった運用が可能です。修復アクションは、AWS Systems Manager というサービスの Automation という機能が利用されています。

### ルールに準拠しない使われ方を自動的に修正することも可能



AWS Config ルールは、設定履歴を残す AWS Config とは別に料金が発生します。作成したルールが評価されるたび、1 評価ごとに 0.001USD\* 発生します。多くのルールを作ると想定外の料金が発生することもあるため、注意しましょう。

\* 東京リージョン、最初の100,000件のルール評価まで。これ以上は1評価の金額が安くなります。

# AWS GuardDuty とは

## 05 AWS の怪しい操作を検知して不正を防ぐ

**keyword** •Amazon GuardDuty •Amazon EventBridge •Amazon SNS  
•サンプルイベント

### AWS 内の脅威を検出する Amazon GuardDuty

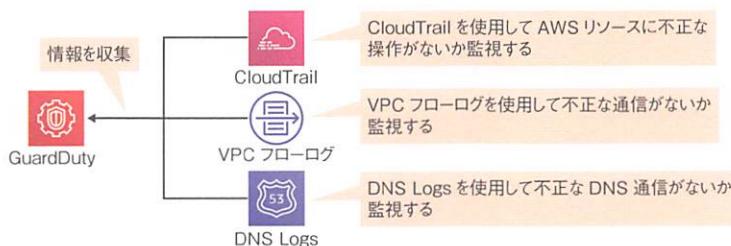
Amazon GuardDuty (以下 GuardDuty) は、AWS 上で発生する不正な動作や悪意のある操作などの脅威を検出するサービスです。サービスをワンクリックで有効化でき、すぐに AWS アカウント内のセキュリティ状況を分析できます。AWS アカウントを開設したらすぐに有効化するとよいでしょう。

検知の仕組みは AWS 側で管理されるため、利用者側で検知の作り込みは不要で、検知した内容を確認すれば OK です。たとえば、以下のようない情報が検知されます。

- ルートユーザーの使用
- IAM アクセスキーの大量利用
- EC2 が DoS 攻撃（サービス拒否攻撃）の踏み台にされている可能性あり

GuardDuty の検知は、CloudTrail、VPC フローログ、DNS Logs の 3 種類の情報とともに行われています。この 3 つに含まれない、EC2 上のアプリケーションや Lambda 関数などの脅威イベントは検知できません。

### GuardDuty は脅威の検知に 3 種類のサービスを利用している



「ルートユーザーの使用」といった具体的な操作を検知するものもあれば、「IAMユーザーがいつもと異なる操作をしている」といった、**環境状況を機械学習で判断して異常と検知するイベント**もあります。

GuardDuty を有効にすることで、脅威の検出状況を画面上で確認できます。ただし、これだけではリアルタイムな検知を行うことができません。利用者へリアルタイムな通知を行う場合は、Amazon EventBridge、Amazon SNS を組み合わせて設定します。この手法は前節の AWS Config で紹介した設定方法と同様です。

## 脅威を検知したらメールで通知



GuardDuty がどんなイベントを検知するのか、検知したらどのように表示されるのか、通知内容はどうなるのか、といった具体的な情報は有効化しただけではわかりません。GuardDuty には、検知内容を確認するための**サンプルイベント**を発行する機能があるので、どういったものが検知されるのか確認したい場合は一度発行してみるとよいでしょう。

## サンプルイベントを発行して検知させると、通知内容を確認できる

GuardDuty > 検出結果 表示中: 74 / 74 12 30 32

結果	情報	アクション
<input type="checkbox"/> 結果の抑制	<input type="checkbox"/> 情報	
保存済みのルール 保存済みのルールがありません		
<input type="button" value="C"/> <input style="float:right" type="button" value="アクション"/>		
最近 ▾ フィルタの追加 サンプルイベント		
検索タイプ	リソース	最終... ▾ カウント ▾
<span style="color:red;">△</span> [例] Backdoor:EC2/DenialOfService.UnusualProtocol <span style="color:yellow;">□</span> [例] Behavior:EC2/NetworkPortUnusual <span style="color:blue;">○</span> [例] Stealth IAMUser/PasswordPolicyChange <span style="color:red;">△</span> [例] Impact:EC2/MaliciousDomainRequest.Reputation <span style="color:blue;">○</span> [例] Policy:S3/AccountBlockPublicAccessDisabled <span style="color:yellow;">□</span> [例] Recon:IAMUser/TorIPCaller <span style="color:red;">△</span> [例] Discovery S3/MaliciousIPCaller	Instance: i-99999999 Instance: i-99999999 GeneratedFindingUserName: GeneratedFindingA Instance: i-99999999 GeneratedFindingUserName: GeneratedFindingA GeneratedFindingUserName: GeneratedFindingA	数秒前 1 数秒前 1 数秒前 1 数秒前 1 数秒前 1 数秒前 1

GuardDuty は、検知に利用するサービスごとに料金が設定されています。

項目	料金(東京リージョン)
CloudTrail 管理イベント	4.72USD／100万イベント
CloudTrail S3 データイベント <sup>*1</sup>	1.04USD／100万イベント
VPCフローログ、DNSログ <sup>*2</sup>	1.18USD／GB

\*1 最初の5億イベントの料金。それより大きくなるとイベントあたりの料金が安くなる

\*2 最初の500GBの料金。それより大きくなると1GBあたりの料金が安くなる

(参考) Amazon GuardDutyの料金

<https://aws.amazon.com/jp/guardduty/pricing/>

料金はイベント数やログの発生量によりますが、通常 GuardDuty の料金が高くなつて問題になるケースはありません。積極的に有効にして活用ていきましょう。

## column

### 複数 AWS アカウントの利用

本書では、AWS アカウントは 1 つ用意して、1 つの AWS アカウント内でシステムを構築する前提で説明しています。しかし、最近では 1 つのシステムで複数の AWS アカウントを使用することも多くなっています。

2-3 節 (p.42) で紹介した利用料の管理や、7-2 節 (p.209) で紹介した IAM (権限) の管理は、基本的に AWS アカウント単位で行います。ただ、1 つのシステムであっても 24 時間 365 日サービスを提供する本番環境と、検証時のみ使用する検証環境では、利用料も権限の管理も扱いが異なることが多いです。たとえば、個人情報がある本番環境は、決まった管理者のみアクセスさせたい（権限を分けたい）場合もあります。そんな場合に、AWS アカウントを本番環境と検証環境で分けることにより、権限や利用料の管理を分けることができます。

2-1 節 (p.34) では、AWS アカウント作成時にクレジットカード情報が必要と説明しましたが、AWS Organizations というサービスを使用すると、メールアドレスと任意の AWS アカウント名のみで AWS アカウントを新規作成できます。また、第 7 章で紹介した各セキュリティサービスを一括で複数アカウントに設定する機能も持っています。本書では詳しくは説明しませんが、複数アカウントの利用が一般的になってきていて、複数アカウントを使用する場合は AWS Organizations を使用することが多いということだけ覚えておいてください。

# 06 Web アプリケーションのセキュリティを強化する

## keyword

• Web アプリケーションファイアウォール • AWS WAF • ウェブ ACL

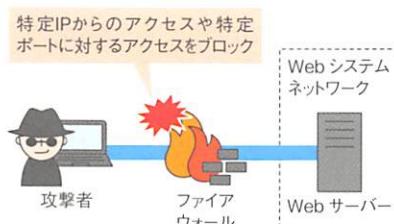
## Web アプリケーションをセキュリティの脅威から守る

Web システムをセキュリティの脅威から守るツールとして、一般的にはファイアウォールがあります。ファイアウォールは特定の IP アドレスからのアクセスやサーバー側の特定ポートに対してのアクセスを防御するものです。これにより、サーバーやネットワークへの不正なアクセスをブロックすることができます。ファイアウォールはその名のとおり「防火壁」を意味します。これは火事のもととなる火種(=不正アクセス)からシステムを防御する役割を持つためにこのように呼ばれています。ファイアウォールは基本的にサーバーの前段に配置され、サーバー側へアクセスが到達する前に処理をします。

同じくファイアウォールと名の付くものに、**Web アプリケーションファイアウォール (以下 WAF)** があります。WAF は **Web アプリケーションの脆弱性を突**

**ファイアウォールと WAF の違い。WAF は Web アプリに特化している**

### ファイアウォール



一般的には保護対象のシステムがあるネットワークとインターネットとの境界に配置される

### WAF

OS インジェクションや SQL インジェクション、クロスサイトスクリプティング(XSS)などの悪意あるコマンドを実行しようとするリクエストを検知しブロックする



一般的にファイアウォールより背後かつ Web システムの前段に配置される

### WAF のメリット

IP やポートの制限で防ぎきれない攻撃や、グローバルに利用される (IP 制限ができない) サービスを提供する Web システムへの攻撃に対して防御することができる

いた攻撃に対して動作し、不正なリクエストを検知・無効化します。ファイアウォールが攻撃者からの道筋を閉ざすような動きをするのに対し、WAFはクライアントからのリクエストを走査し、攻撃と思われるものをピックアップしていくような動きをします。

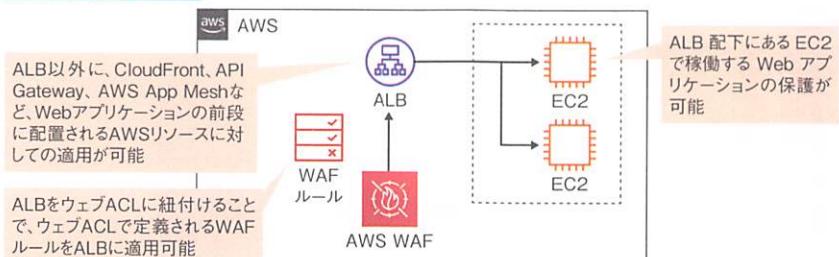
## AWSのWebシステムに手軽に導入できるAWS WAF

AWS WAFはWAF機能を提供するAWSのマネージドサービスです。AWS WAFの登場により、AWS上のWebシステムにおいてより容易にセキュリティ対策を導入できるようになりました。

仕組みとしては、**ウェブアクセスコントロールリスト（ウェブACL）**と呼ばれるリソースで、IP制限や不正なコマンドの検出など、リクエストに対して制御したいルールを管理し、これを対象のAWSサービスに適用することで配置できます。たとえばWebアプリケーションをEC2上で実行する場合は、その前段に配置されるALB（p.148）に適用することで、ALB配下にあるEC2を保護することができます。その他にもCDNサービスであるCloudFrontやAPI Gateway、AWS AppSyncに対しても適用が可能です。

### AWS WAFで通信の制御ルールを作成し、AWSのサービスに適用する

#### ALBに適用する場合



AWS WAFはインフラ部分がAWS側で管理されているため、WAF自体のインフラ部分の管理やメンテナンスは不要です。そのためユーザー側は、どういったWAFルールを適用するのかといったセキュリティ対策に注力できます。また、AWSリソースに適用する際も、簡単な設定変更をするだけで反映されるため、デプロイがしやすいというメリットもあります。

WAF ルールについては、IP 制限や簡単な正規表現によるフィルタリングに加えて、AWS 側で管理されるマネージドルールを利用することもできます。これには、SQL インジェクションやクロスサイトスクリプティングなどの代表的な攻撃手法に対する WAF ルールも含まれています。また、AWS Marketplace（マーケットプレイス）上で販売されるサードパーティの WAF ルールも利用可能なため、これを用いることにより柔軟な WAF ルールの設定が可能です。

料金体系としては、利用するウェブ ACL やルール数、実際に AWS WAF で処理されたリクエスト数によって料金が発生するため、コストパフォーマンスにも優れています。

具体的に、1カ月で 100 万リクエストが想定される AWS 上の Web システムを例に考えてみます。基本的に ALB や CloudFront などの 1 リソースに対して、1 つのウェブ ACL が割り当てられるイメージですので、今回は Web システムで利用する 1 つの ALB に割り当てられる 1 つのウェブ ACL の月額利用料金について考えます。たとえば、AWS 管理のマネージドルールグループのみで 5 つのルールグループを利用した場合は、次図のような料金になります。

#### 例：AWS 管理のマネージドルールグループを 5 つ利用する場合（東京リージョン）

ウェブ ACL の料金 =  $1 \times 5 \text{ USD} = 5 \text{ USD}$   
(1 ウェブ ACL の月額利用料が 5 USD(時間按分))

ルールの料金 =  $5 \times 1 \text{ USD} = 5 \text{ USD}$   
(AWS 管理のマネージドルールグループは 1 ルールグループで 1 USD(時間按分))

リクエストの料金 =  $0.6 \text{ USD}$   
(100 万リクエスト / 月ごとに 0.6 USD)

合計 =  $5 + 5 + 0.6$   
= **10.6 USD**

ウェブ ACL、ルールの料金については時間単位での按分になるため、Web システム側が利用されていない時間帯がある場合は、この金額よりも安くなる可能性がある

独自のカスタムルールや、マーケットプレイスに展開されるサードパーティが提供するマネージドルールを適用する場合は、別途追加料金が発生する

AWS 管理のマネージドルールグループには、基本的な攻撃手法に対する防御ルールがすでに用意されています。さらに踏み込んだセキュリティ対策を行う場合はサードパーティツールなどを検討しますが、基本的なセキュリティ対策を素早く展開したい場合にはこれだけでも幅広くカバーすることが可能です。

# 07 システムのセキュリティを強固にするサービス 6 選

**keyword** • AWS Network Firewall • Amazon Inspector • AWS Shield  
 • AWS Security Hub • Amazon Cognito • AWS Directory Service

## クラウドに求められるのはセキュリティ

システムをクラウドに構築するとなった場合、システムの責任者が真っ先に気にしがちなことはセキュリティの強固さです。オンプレミスと異なり、構成は AWS に委ねられているため、どういったセキュリティが実現されているかが見えづらいのが主な理由ですが、AWS はそれに応えるため多彩なセキュリティサービスを提供しています。

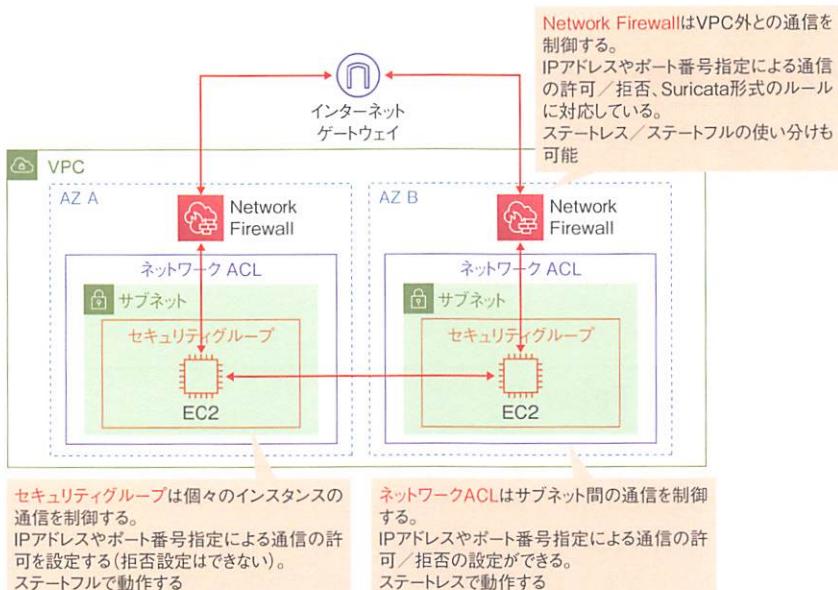
### AWS Network Firewall

AWS 内に構築するシステムは基本的に、VPC の中で動作することになります。VPC のアクセス制御は p.137 で説明したセキュリティグループとネットワーク ACL が基本的に利用されるのですが、セキュリティグループはルールの許可のみが設定可能、ネットワーク ACL はルールの許可／拒否両方が指定可能なもののストレスであるという特徴があり、一般的に利用されるファイアウォールとは性質が異なっています。特にオンプレミス環境でファイアウォールを使ってアクセス制御を行っていたシステムを移行する場合、現行のルールをセキュリティグループとネットワーク ACL の形式に変換するか、AWS Marketplace で提供されているファイアウォール製品を利用して EC2 インスタンスとしてファイアウォールを構築する必要がありました。

**AWS Network Firewall** (以下 Network Firewall) は、2020 年 11 月にリリースされた比較的新しいサービスで、VPC をまたぐ通信に対し、ファイアウォールの役割を果たします。一般的なファイアウォールと同じような IP アドレスやポート番号指定による通信の許可／拒否のほか、ドメイン指定や Suricata というオープンソースの IPS (侵入防止システム) 互換のルール形式にも対応するなど、柔軟な通

信制御を行うことができます。対象のルールをステートレス／ステートフルのどちらで操作させるかということや、通信の許可／拒否以外に「通信は許可するがアラートを上げる」といった設定もできるため、あらゆる要件に対応できるようになっています。さらにマネージドサービスであり、通信量に応じて自動でスケールするため非常に可用性が高くなっています。

## VPC のアクセス制御に利用する 3 つのサービスの位置付け



## Amazon Inspector

マネージドサービスの充実してきた AWSにおいて、EC2 は利用者がインスタンスの脆弱性管理を行う必要のある数少ないサービスとなっています。そのぶん EC2 は利用者にとって構成の自由度が高い環境となっているため、複雑な仕組みのシステムに用いられる傾向にありますが、数が多くなるほどインスタンスの脆弱性管理を行うのが大変になります。

Amazon Inspector (以下 Inspector) は、そういった脆弱性管理を行うためのサービスです。Inspector にはあらかじめ評価のためのルールが用意されており、

## システムのセキュリティを強固にするサービス6選

利用者が選択した内容についての評価を指定されたスケジュールに従って自動的に行います。具体的には、「一般に公開されているソフトウェア脆弱性情報に合致するものがないか」や「AWSのセキュリティのベストプラクティスを満たしているか」といった内容の評価ルールが用意されています。

## 自動的にEC2インスタンスの脆弱性評価を行う

The screenshot shows the 'Amazon Inspector - 結果' (Results) page. It displays a table of findings for four EC2 instances. The columns are '严重度' (Severity), 'Date', '結果' (Result), 'ターゲット' (Target), 'テンプレート' (Template), and 'ルールパッケージ' (Rule Package). All findings are marked as 'High'. The last update was on 2021/11/15 04:11:43. The table shows the following data:

严重度	Date	結果	ターゲット	テンプレート	ルールパッケージ
High	11/25/2016	Instance [REDACTED] is vulnerable to CVE-2014-0224	Inspector test		Common Vulnerabilities and Exposures-1.1
High	11/25/2018	Instance [REDACTED] is vulnerable to CVE-2018-5195	Inspector test		Common Vulnerabilities and Exposures-1.1
High	11/25/2016	Instance [REDACTED] is vulnerable to CVE-2015-7039	Inspector test		Common Vulnerabilities and Exposures-1.1
High	11/25/2018	Instance [REDACTED] is vulnerable to CVE-2016-6692	Inspector test		Common Vulnerabilities and Exposures-1.1

利用者は定期的に出力される評価レポートを確認し、発見された脆弱性に対して都度対応を行うだけになり、運用の手間を大幅に減らすことができます。また、ソフトウェアのアップデートなど簡単な処理で解消できる脆弱性に関しては、特定の処理を自動実行できる AWS Systems Manager を組み合わせることで、対策実行までを自動化することもできます。

## AWS Shield

**AWS Shield** は気付かずにお世話になっているサービスです。

インターネットに公開されたシステムはサイバー攻撃を受ける可能性を避けることはできません。AWS Shield はサイバー攻撃の中でも単純かつ効果の高い **DDoS 攻撃**を防ぐサービスで、インターネットに面するサービスのすべてで自動的に有効化されています。

DDoS 攻撃とは Distributed Denial of Service 攻撃の略で、多数の攻撃者がシステムに大量のアクセスを行って負荷をかけることでサービスの停止を狙う攻撃です。しかし、AWS のサービスによっては処理したリクエスト量に応じて利用料が増え、リクエストの増加や負荷の上昇に反応した Auto Scaling によってリソースが増えることでも利用料が増えるため、**DDoS 攻撃を受けることで利用料が急増してしまう**ことがあります。特にこういった利用料を増加させ、利用者に経済的損失を与えることを目的とした攻撃は **EDoS (Economic DoS) 攻撃**とも呼ばれます。そ

のため、AWSにおいてDDoS攻撃は優先して防ぐべき攻撃であるといえます。

AWS ShieldにはStandardとAdvancedの2種類のプランがあり、Standardは無料で自動的に有効化されます。Standardでは一般的なDDoS攻撃をリアルタイムで検知して自動的に遮断しますが、より高度なDDoS攻撃に対応したい場合はAdvancedを利用することになります。Advancedを有効化することでAWSのDDoS対策専門チームのサポートを受けることができ、対策も任せてしまうことができます。とはいってもAdvancedの料金はかなり高額なので、大規模かつ攻撃を受けやすいようなサービスでないかぎりはほとんど利用することはないでしょう。

## AWS Security Hub

AWS Security Hub（以下Security Hub）は、AWSアカウント全体に対して、セキュリティのベストプラクティスのチェックを行うサービスです。PCI DSS（Payment Card Industry Data Security Standard）というクレジットカード情報セキュリティの国際統一基準や、CIS（Center for Internet Security）という団体が定めたAWSアカウントの基本的なセキュリティのベストプラクティスであるCIS AWS Foundations Benchmarkといった基準に従っているかのチェックのほか、次図に示すようなAWSが定める推奨設定がなされているかをチェックすることも可能です。

### AWSが推奨するセキュリティ設定がなされているかチェックできる

セキュリティコア	状態	件数
セキュリティコア	成功	15
セキュリティコア	失敗	0
セキュリティコア	不明	0
セキュリティコア	データなし	0

セキュリティコア	状態	件数
セキュリティコア	成功	15
セキュリティコア	失敗	0
セキュリティコア	不明	0
セキュリティコア	データなし	0

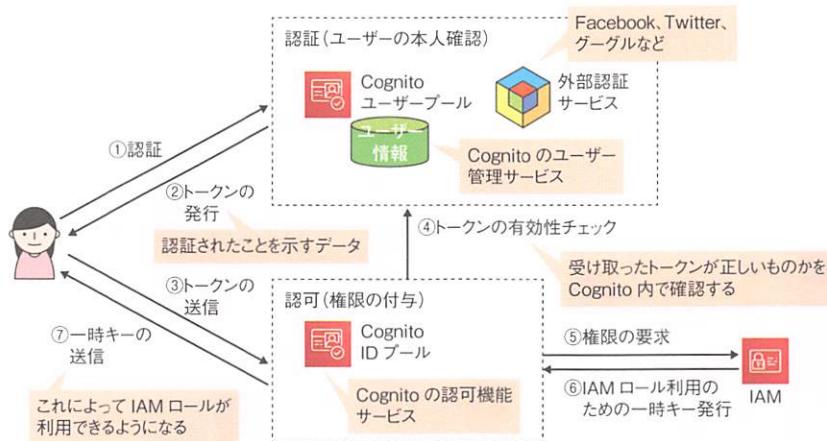
## Amazon Cognito

**Amazon Cognito**（以下 Cognito）は、Web アプリ／モバイルアプリのユーザー認証・認可を行うためのサービスです。ユーザー認証というと IAM が思い浮かぶかもしれません、IAM の対象は AWS のサービスを利用するユーザーで、Cognito の対象は AWS に構築したシステムを利用するユーザーとなります。

**ユーザー認証機能**では、ID とパスワードによる認証のほか、SMS や MFA アプリを利用した多要素認証、電話番号やメールアドレスの有効性確認、パスワード紛失時のパスワード変更機能といった機能も提供されています。

認証されたユーザーに権限を与える機能が**認可機能**です。Cognito は認証されたユーザーに IAM ロールの権限を利用させることができます。Cognito がユーザーに権限を与えるときの認証方法には、先に説明した Cognito が持つユーザー認証機能のほか、Facebook や Twitter、グーグルなどの提供する認証サービスを利用するともできます。そのため、Twitter アカウントと連携して AWS 上に構築したシステムを利用させるといったことも可能となります。

### Cognito で AWS に構築したシステムのユーザー認証・認可を行える



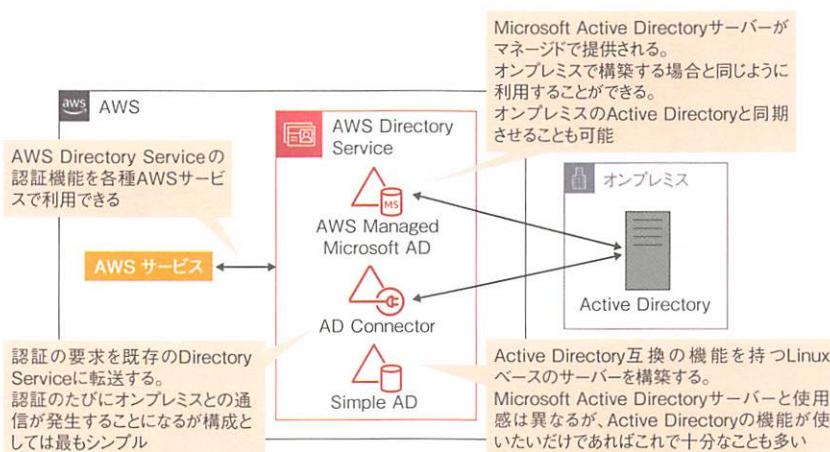
## AWS Directory Service

多くの企業や組織のシステムにおいて利用されているユーザー管理システムは、マイクロソフトの Active Directory でしょう。Active Directory に組織のメンバーのユーザーを登録しておくことで、社内システムの認証に利用したり、コンピューターやプリンターなどの認証に利用したりと、さまざまな IT 資産への認証に使うことができます。

**AWS Directory Service** は、AWS 上で Active Directory を利用するためのサービスです。AWS 上にマネージド型の Active Directory を構築する **AWS Managed Microsoft AD**、Samba 4 Active Directory Compatible Server という Active Directory 互換の機能を提供する Linux 系のサーバーをマネージド型で提供する **Simple AD**、既存の Active Directory を AWS サービスから利用するための中継を行う **AD Connector** の 3 種類のサービスが用意されています。

AWS Directory Service を利用することで、AWS に構築したシステムでも使い慣れた Active Directory でのユーザー管理を行えるほか、オンプレミス環境などにある既存の Active Directory をそのまま AWS で利用することもできます。さらには Active Directory のユーザーに IAM ロールを紐付けることもできるため、AWS を利用する際の認証を IAM ユーザーでなく Active Directory に登録されたユーザーで行うことも可能です。

### AWS 上でマイクロソフトの Active Directory を利用できる





Chapter  
8

# 知りておきたい その他のサービス

---

AWS が今後特に力を入れていくと見られるデータ分析や機械学習のためのサービス、システムを運用するには避けて通れないマネジメントのためのサービスを紹介します。

---

- section  
**01** データ分析サービス  
溜めたデータを次に生かすサービス 6 選
- section  
**02** 機械学習サービス  
お手軽に始める機械学習
- section  
**03** マネジメントサービス  
システム自体を管理するサービス 4 選

## 01

溜めたデータを次に生かす  
サービス6選

## keyword

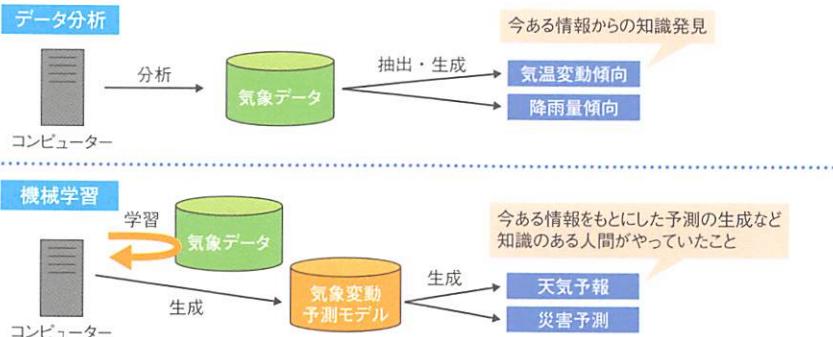
- データレイク
- データウェアハウス
- Amazon Athena
- AWS Glue
- Amazon OpenSearch Service
- Amazon EMR
- Amazon QuickSight
- Amazon Kinesis

## データ分析とは

AWS のストレージサービス、データベースサービスには、大量のデータを高い可用性で保管することができます。また、大量に流れてくるデータを高速で処理するためのサービスも用意されています。こういったサービスを使って集めたデータを分析し、そこから有用な情報を得るのがデータ分析サービスの役割です。

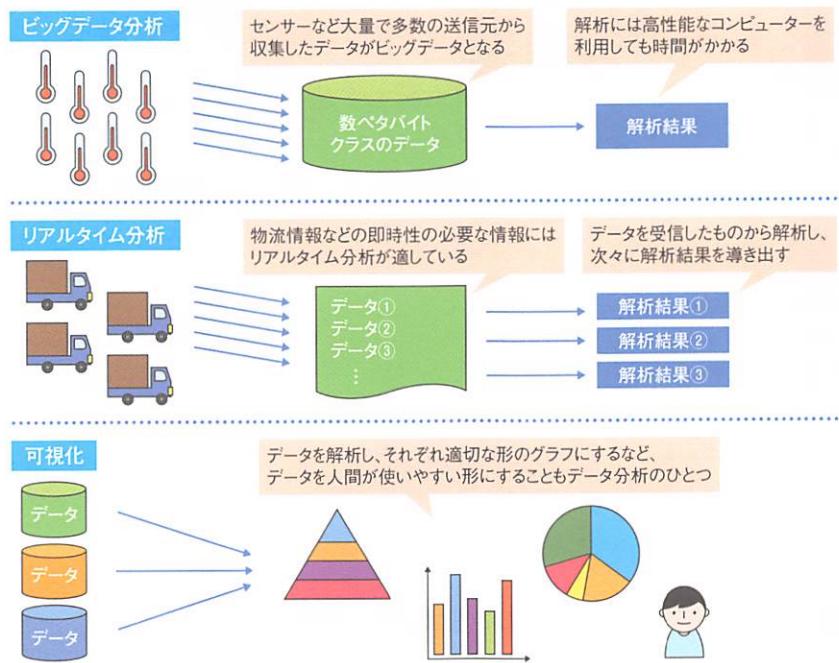
データ分析とは、**新たな知識発見や意思決定のサポートを行うことを目的に、蓄積されたデータから規則性や傾向を探し出すこと**です。機械学習と混同されることは多いですが、コンピューターが傾向を「学習」して新しい定義を作り出すのは異なり、既存のデータに埋もれている規則性や傾向を「発見」するのがデータ分析の主な目的です。とはいえ、近年は「データ分析で発見した内容を機械学習の技術によってコンピューターに学習させる」というように、両者を組み合わせて活用する動きも活発になってきており、両者の境界線は薄くなっています。

## データ分析は知識を発見する。機械学習は定義を作り出す



ひとくちにデータ分析と言っても、分析の対象データはさまざまです。おそらく、データ分析として有名なのは「ビッグデータ」という言葉でしょう。これはその名のとおり、非常に大規模なデータ（具体的には数ペタバイトクラス（1PB = 10万GB））のデータを指します。しかし、このような巨大なデータの分析だけでなく、リアルタイムで流れてくるデータを素早く分析することや、データを整理して可視化することもデータ分析のひとつです。

## ビジネスで活用される 3 種類のデータ分析

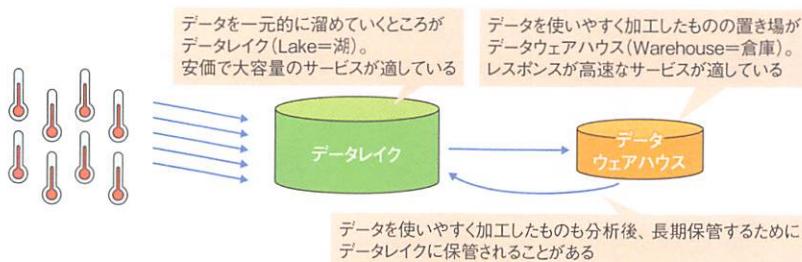


## データの保管

分析に使うデータを溜めていく場所はデータレイクと呼ばれます。送られてくるデータを加工せずにそのまま蓄積し、利用する際に加工してから使う方式がとられることが多いため、データレイクには大容量でデータサイズあたりの利用料が安いサービスが適しています。AWSでは基本的にS3が利用されます。なお、送られてきた未加工のデータは一般的に生データ(Raw Data)と呼ばれます。

生データはそのまま分析対象になることもあります、分析方法に適した形式に加工してデータベースなどに格納したうえで使われることが多いです。そういった加工済みデータを格納しておく場所を**データウェアハウス**と呼びます。データウェアハウスには集計処理が得意なデータベースである Redshift (p.193) がよく採用されます。

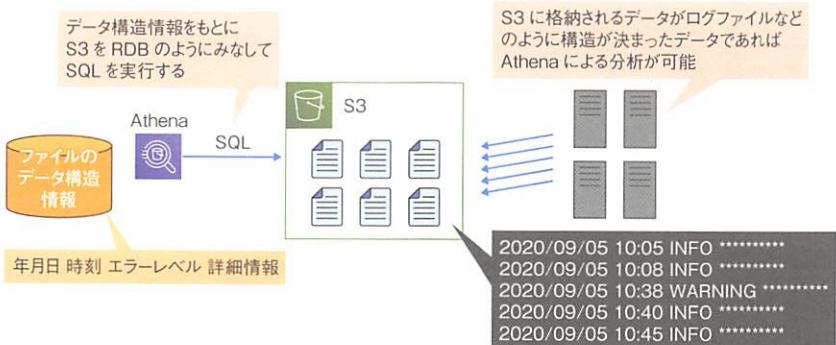
### 未加工のデータと加工済みのデータをそれぞれ格納しておく



### Amazon Athena

Amazon Athena (以下 Athena) は、S3 のデータを SQL で分析できるサービスです。通常、SQL はデータベース (RDB) に対して実行するのですが、S3 に格納されたファイルのデータ構造を Athena にあらかじめ登録しておくことで、S3 がデータベースであるかのように SQL を実行できます。S3 に蓄積されるデータの構造が決まっている場合であれば、手軽に集計や検索を行えます。

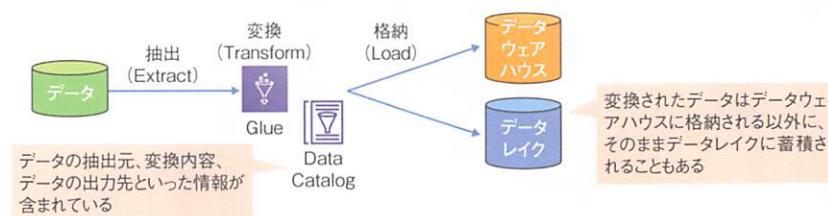
### S3 をデータベースとみなして SQL を実行し、集計・検索



## AWS Glue

データ分析を行う際は、データレイクからデータを取り出し、データ分析に利用しやすい形に変換したり、データウェアハウスに移行したりしてから利用することが多いです。AWS Glue (以下 Glue) はそういった ETL 处理 (Extract / Transform / Load) を簡単に自動化するサービスです。

### データの抽出・変換・格納を自動化



## Amazon OpenSearch Service

Amazon OpenSearch Service は、Elasticsearch というオープンソースソフトウェアをベースに AWS が開発を進める OpenSearch という全文検索エンジンを簡単に AWS 環境に展開できるサービスです。全文検索という名のとおり、対象のデータを格納することで自動的に全内容を検索できるようにしてくれます。さらに、データ可視化ツールである OpenSearch Dashboards というソフトウェアも同梱されており、Web 画面上でデータの検索・集計を行ったり、結果をグラフ化して表示したりすることができます。

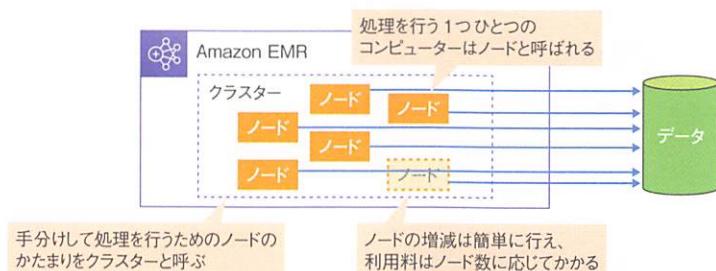
### データの全文検索とグラフ化



## Amazon EMR

ビッグデータ分析は対象となるデータの膨大さから、単独のコンピューターで処理できるものではありません。複数のコンピューターで処理を手分けして分析を行う技術として、Apache Spark や Apache Hadoop などといった仕組みが開発されており、これらは**分散フレームワーク**と呼ばれています。Amazon EMR はこのような分散フレームワークを AWS 上で動かすためのサービスです。

### ビッグデータ解析のための分散フレームワークを動かす



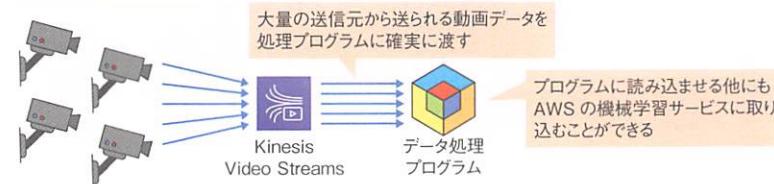
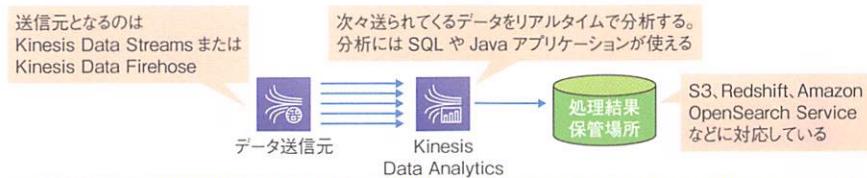
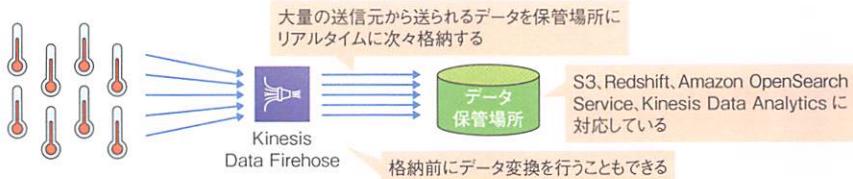
## Amazon QuickSight

Amazon QuickSight は、登録されたデータを解析し、グラフ化するなど、データの可視化を提供するサービスです。単なるデータの可視化だけでなく、機械学習によるデータの特徴発見や、今後のデータ変動の予測という機能を持っています。さらには、「前年比売上高の伸びはどれくらいですか?」というように自然な文章での質問に対し、答えとなるグラフを表示する自然言語クエリという機能も持っています。自然言語クエリは、2022年1月現在、英語にのみ対応しています。

## Amazon Kinesis

リアルタイム分析を行うには、データの送信元から送られてきたデータを素早く処理できる状態に整理する必要があります。Amazon Kinesis はリアルタイムでデータを処理するための4つのサービスからなり、まとめて Kinesis Family と呼ばれます。

## Amazon Kinesis でリアルタイムのデータ処理を行う



## 02 お手軽に始める機械学習

**keyword** • Amazon SageMaker • Amazon Lex • Amazon Polly  
• Amazon Transcribe • Amazon Translate • Amazon Rekognition

### 機械学習とは

機械学習とは、その名のとおり機械（コンピューター）が多数のデータを取り込み、そこから何らかの学習をする（傾向などの知見を蓄積する）ことによって得た結果を用いて、何らかのタスクをこなすことです。少し難しい言い方をしましたが、機械学習には大きく分けて次の3つの種類があります。

#### ● 教師あり学習

何らかの正解があることに関して解答例（教師データと呼ばれます）をコンピューターに学習させ、コンピューターが人間と同じような判断ができるようにする手法です。対象の分野としては、教師データとなる状態と結果のセットが多数用意でき、結果が普遍的な（もしくはそれに近い）文字認識、不良品判別といったものが挙げられます。

#### ● 教師なし学習

明確な正解がないことに関する多数のデータをコンピューターが取り込み、そこから特徴の発見やグループ分けをすることで新たな知見を見出させる手法です。対象の分野としては、購買データから売上の多い顧客層を探したり一緒に売れる商品の組み合わせを発見したりといった潜在ニーズの発見、大量の画像から似た画像の特徴を洗い出して画像検索に利用するといったものが挙げられ、結果が未知のものに有効な手法です。

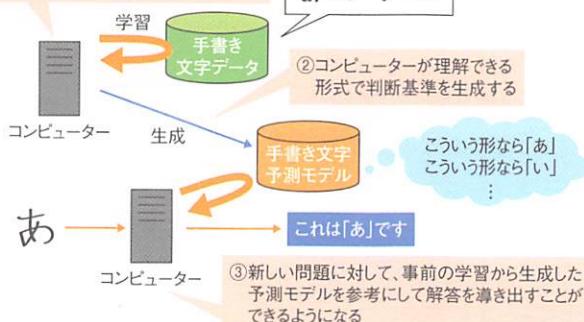
#### ● 強化学習

状況に得点を付け、どういった行動をとれば最大の得点を得られるかを、何度も試行することで導き出す手法です。状況が明確で、かつ行動によって得られた結果を評価しやすいものに向いており、囲碁やチェスのAIに使われたものが有名です。

## 機械学習の手法は大きく3種類

### 教師あり学習

- ①多数の問題と解答の組み合わせから法則を学習する

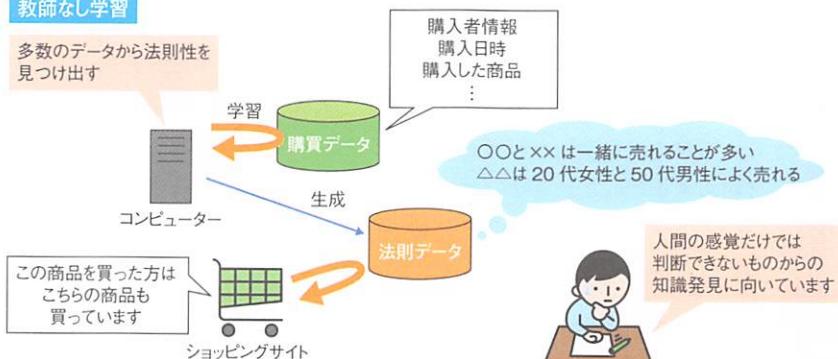


コンピューターが人間と同じような判断ができるようになります



### 教師なし学習

- 多数のデータから法則性を見つけ出す



人間の感覚だけでは判断できないもののからの知識発見に向いています



### 強化学習

- 事前に「どういう結果が出たら何点」というルールを与えておく



人間が考えるよりも高速に多数のパターンを検討することができるため、より優れた方法を発見できる可能性があります



ここに打つと黒が○枚増えるから△点  
ここに打つと黒が□枚増えるから×点

また、機械学習分野で近年よく聞くようになった言葉で、**ディープラーニング**（深層学習）というものがあります。ディープラーニングとは、こういった学習のプロセスをコンピューターが何度も行うことで自律的に学習の精度を高めていく手法です。

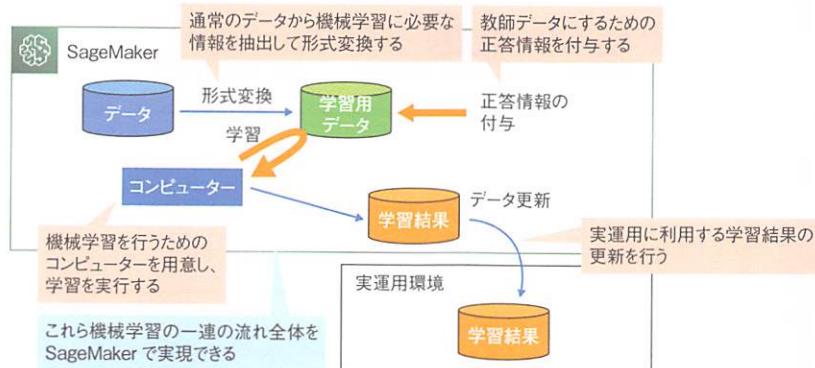
最も普及しているディープラーニングの手法として、人間の脳を模した**ニューラルネットワークの多層化**があります。ニューラルネットワークは多数の脳細胞（ニューロン）の情報伝達を数学的に表した機械学習のモデルで、情報伝達の段階が階層的になっています。この階層を深くすることで、より精度の高い学習結果が得られるという理論がディープラーニングの起源であり、コンピューターの能力向上やインターネットの発展による学習データの流通により2010年代に普及しました。これによりディープラーニングの有用性が注目され、現在ではさまざまな手法のディープラーニングが研究されています。

機械学習には複雑な計算を繰り返し高速に行うためのコンピューターが必要であり、また多数存在する学習モデルへの知識が必要とされていたため、始めるには少々ハードルが高いところがありました。しかし、AWSは機械学習を利用するために必要な一連の要素をまとめてサービス化することで機械学習の知識なしに利用可能とし、さらに複雑な機械学習を行うためのコンピューターも必要なときに必要なだけ使える仕組みを提供しています。これらのサービスによって機械学習はより身近になったといえるでしょう。

## Amazon SageMaker

Amazon SageMaker（以下SageMaker）は、機械学習を実行するための環境を提供するサービスです。多数の機能があり、データの収集や学習に使える形式への変換、教師データの作成といったデータの準備から、学習の実行、得られた学習結果の実運用環境への反映まで、機械学習における一連の作業をすべてまかなえるようになっています。また、機械学習の分野では多数のアルゴリズムが研究・開発されていますが、主要なものについてはあらかじめコンテナイメージとして構築されたものが提供されています。

## 機械学習に必要な機能一式がまとめて提供されている



## その他の機械学習サービス

SageMaker は、利用者が機械学習を始めるために必要な機能一式を提供するサービスですが、それ以外にも AWS が開発した機械学習システムを利用者に提供するサービスが目的別に数多く存在します。

AWS の提供する機械学習サービスはどんどん増えており、入力したデータから未来のデータ推移を予測する、オンライン詐欺を検出する、製品の欠陥を検出する、プログラムのコードをレビューするなど、多種多様な事柄に機械学習を適用することができるようになっています。

## 音声や画像・映像などを扱う機械学習サービス

サービス名称	概要
Amazon Lex	Amazon Alexa に採用されているディープラーニング技術と同じ技術を利用して、音声やテキストを使用した対話型インターフェイスを構築できる
Amazon Polly	ディープラーニングによって自然に聞こえるような音声合成を行うことで、文章をリアルな音声に変換する
Amazon Transcribe	さまざまな音声データを学習しており、音声を迅速かつ正確にテキスト変換することができる
Amazon Translate	ディープラーニングを利用した翻訳サービス
Amazon Rekognition	画像や映像から顔検索など特定の画像の検出をすることができる。指定した物体の検出以外にも、該当の画像が暴力や性的なものなど不適切な内容でないかという分類を行うことも可能

# 03 システム自体を管理するサービス4選

**keyword** • AWS CloudFormation • Amazon CloudWatch • AWS Systems Manager  
• Amazon Code シリーズ

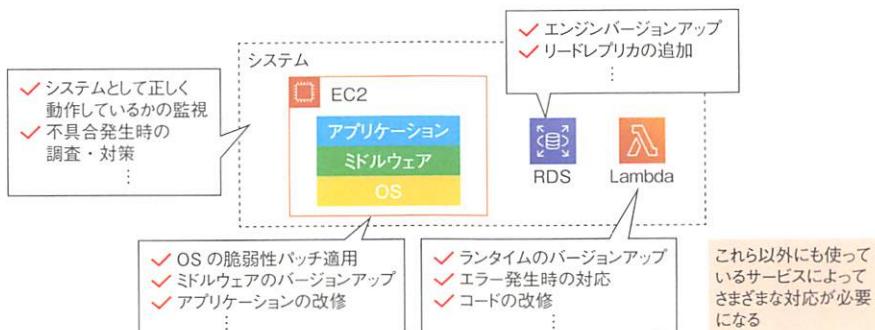
## システムのマネジメントとは

システムを構築し、サービスを提供するという一連の流れを考えると、システムを構築している時間よりも、サービス開始後に機能改善やトラブル対応といったシステムの維持活動を行っている時間のほうが圧倒的に長くなります。これらをいかにやりやすくするかということは、システムを長く運営するうえで大切なポイントです。

システムは、サービスを開始した後も継続して機能改善やトラブル対応、セキュリティ対応といったシステムを維持していくための作業が必要となります。こういった作業を**システムマネジメント**（日本語だと運用管理）といいます。

具体的な内容を挙げてみると、アプリケーションに不具合が見つかったときの改修や新機能追加といったアプリケーションの変更や、その実運用環境（本番環境）への反映（デプロイ）、リソースの追加や構成変更、システムの動作に異常が発生していないかの監視、利用しているソフトウェアにセキュリティ脆弱性が発見されたときの修正パッチ適用やバージョンアップなど、多種多様なものがあります。

## 正しく動作しているかの監視や、問題あるソフトの更新などを行う



## AWS CloudFormation

AWS CloudFormation（以下 CloudFormation）は、AWS リソースを JSON または YAML と呼ばれる書式で記載されたファイル（テンプレートと呼ばれます）で管理し、それを読み込ませることでテンプレートの内容に沿ったリソースを構築することができるサービスです。

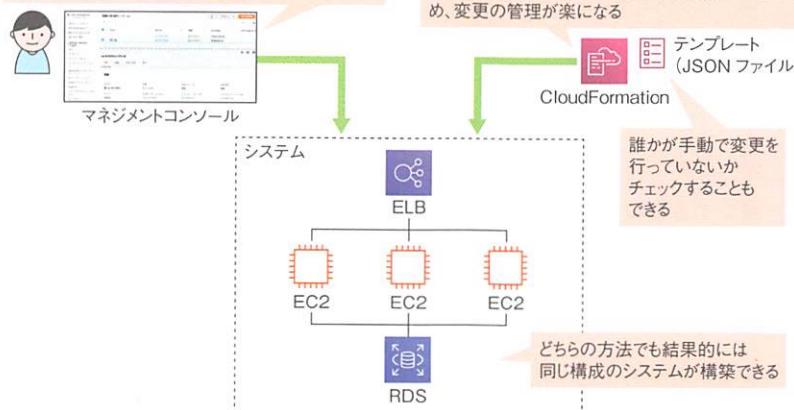
マネジメントコンソール上の数クリックできることをわざわざファイル化して AWS に読み込ませるというのは一見遠回りのように見えます。しかし、テンプレートにしておくことで設定変更前に変更内容を確認することができ、再度同じ構成に戻すことや、同じ構成のリソース一式をもう 1 セット作ることが非常に簡単になります。また、テンプレートの変更履歴を残しておけば、容易にシステムの構成変更履歴を把握することができます。このように AWS の構成をテンプレート化しておくことが、システムマネジメントをやりやすくするひとつの手法となります。

なお、CloudFormation で作成した AWS リソースはマネジメントコンソールなどから作成したものと変わらないので、手動で設定変更することも可能です。そうした場合、テンプレートとの差異が発生してしまうので、CloudFormation ではこうしたテンプレートとの差分（ドリフトと呼ばれます）が発生していないかをチェックする機能も提供されています。

### システムリソースの構成をテンプレート化しておく

マネジメントコンソールは手軽にリソースを作成できるため、試しながら構築したい場合に適している。ただし、テスト用途などで同じ構成のシステムをもう 1 つ作りたいとなった場合は手間がかかる

JSON や XML のファイルを作成するのには手間がかかるが、一度作れば同じ構成のシステムが簡単に作れ、作り直しも容易。ファイルを更新すれば一部だけの変更もできるため、変更の管理が楽になる



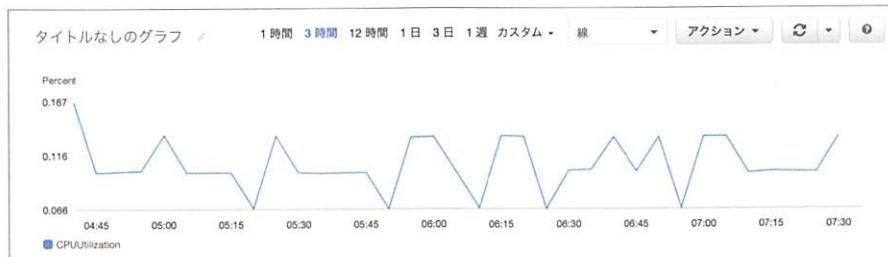
## Amazon CloudWatch

Amazon CloudWatch（以下 CloudWatch）は、AWS サービスのメトリクス、ログを監視し、可視化するサービスです。

メトリクスとは AWS サービスから登録されるデータの集合で、具体的には EC2 や RDS のインスタンスの CPU 使用率や S3 のアクセス受信数といった各サービスの稼働状況を示すデータが挙げられ、これらはサービスを使用していると自動で登録されていきます。何も設定しなくとも自動で登録されるメトリクスは、一般的に標準メトリクスと呼ばれています。メトリクスは、CloudWatch エージェントというソフトウェアや API を利用して、独自のデータを登録することもできます。独自に登録されたメトリクスはカスタムメトリクスと呼ばれ、標準メトリクスと同じように CloudWatch 上で可視化することができます。

登録されたメトリクスは、さまざまな時間範囲や条件を指定してマネジメントコンソール上でグラフ化することができます。

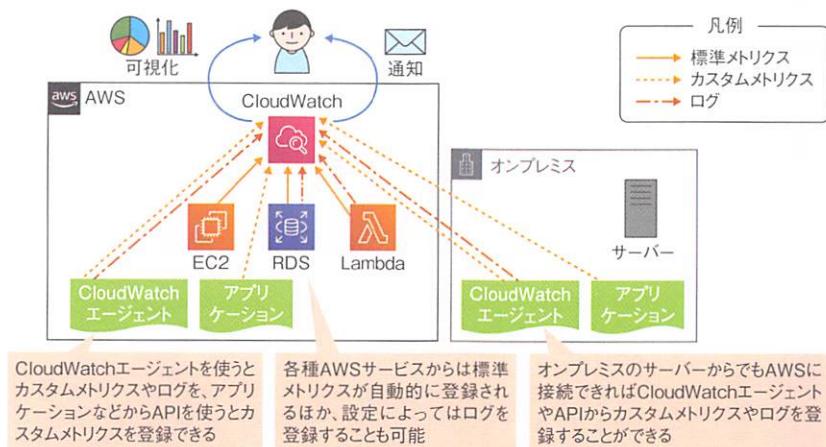
### EC2 インスタンスの CPU 使用率をグラフ化した例



CloudWatch にはアラームという機能があり、特定のメトリクスの値が指定したしきい値を超えたときに利用者に通知したり、指定したアクションを起こしたりできます。アラームを利用してすることでシステムの異常を利用者が検知することができ、うまく使えば異常発生時にシステムの自動復旧を行えます。

ログの監視をする機能は CloudWatch Logs と呼ばれます。ログを CloudWatch Logs に転送することで、マネジメントコンソール上でログの内容を一元的に管理でき、特定の文字列を含むログが output された場合にアクションを起こすこともできます。AWS のサービスが output するログを登録することができるほか、API や CloudWatch エージェントを利用することで任意のログを登録することも可能です。

## サービスの稼働状況を可視化したり、しきい値を超えたら通知したりする



## AWS Systems Manager

AWS Systems Manager (以下 Systems Manager) は、旧称を Amazon EC2 Simple Systems Manager といい、EC2 の管理を行うためのサービスです。旧称から略称を SSM とされており、名称が変更された今も略称は引き継がれて SSM と呼ばれています。具体的には **Linux および Windows のサーバーを管理する機能が多数集まったサービス** であり、AWS と通信できる環境下にあればオンプレミスのサーバーも管理することができます。

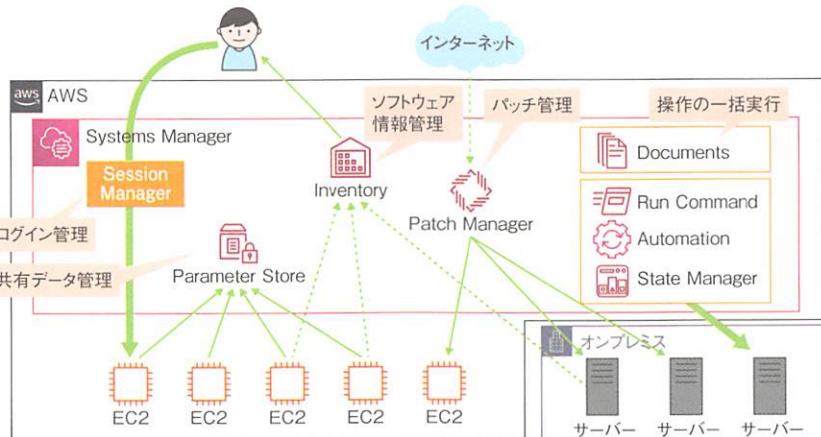
具体的な機能としては次表のようなものがあり、一部の機能の利用にはサーバーに SSM エージェントというソフトウェアをインストールする必要があります。多数のサーバーの一斉操作や、ソフトウェアのインストール状況の一覧化など、多数のサーバーを管理する際に便利な機能が揃っています。

## Systems Manager が提供する主な機能

機能名	概要
Session Manager	セキュリティグループの通信許可設定なしで管理下のサーバーにログインできる機能。EC2インスタンスの場合は、インターネットに接続されていない状態（プライベートサブネットに所属している状態）でもログイン可能。さらに操作ログを保存することもできる

機能名	概要
Inventory	管理下のサーバーにインストールされているソフトウェアを一覧表示する。バージョンやインストール日時なども確認できるため、いつソフトウェアが追加されたかや、アップデートせず古いバージョンのソフトウェアを使い続けているサーバーがないかを簡単に確認できる
Patch Manager	OSやミドルウェアのパッチ適用を管理する機能。「重要度が高いセキュリティパッチはすぐに適用する」「バグ修正パッチはパッチリリースの7日後に適用する」などの条件を設定しておくことで、自動的にパッチ適用を行うことができる
Parameter Store	複数のサーバーで共用したい情報を格納しておくことができる機能。最大8KBの文字列を保存することができ、機密性の高いデータは暗号化して格納することも可能
Documents	下で説明するRun Command、Automation、State Managerで実行するための操作内容を保管する機能。利用者が自分で作成するほか、AWS提供のものを利用することもできる
Run Command、Automation、State Manager	上で説明するDocumentsに指定した操作を実行する機能。単に1つのDocumentsを実行するRun Command、複数のDocumentsを順に実行するAutomation、定期的にDocumentsを実行することでサーバー状態を一定の状態に保つState Managerといった用途別にサービスが用意されている

## サーバーを管理する機能が多数提供されている

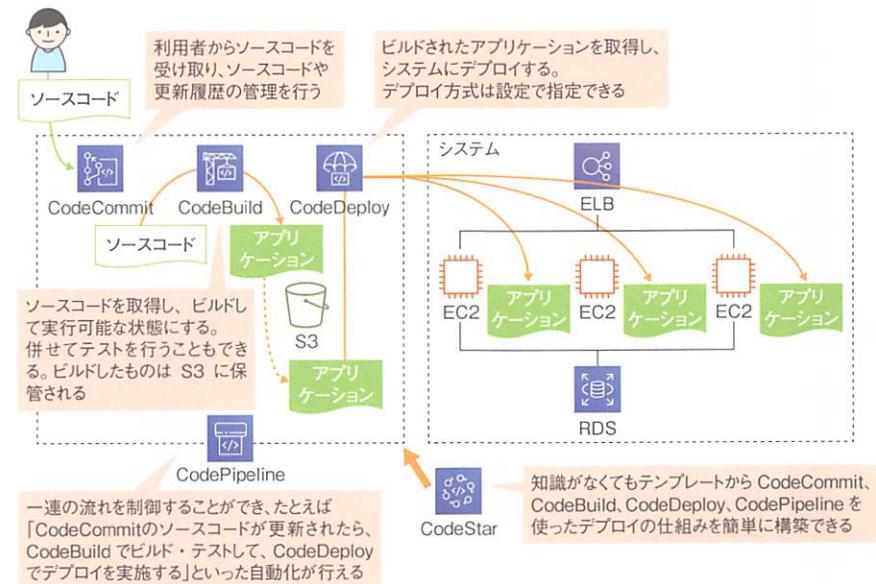


## Amazon Code シリーズ

アプリケーションのソースコードを管理するためのサービスの一式が **Amazon Code シリーズ**です。ソースコードを管理する Git リポジトリサービスである

CodeCommit、ソースコードをビルドしたりテストしたりする CodeBuild、ビルドしたアプリケーションをデプロイする CodeDeploy、ソースコードの更新からビルド、デプロイまでの一連の流れを管理する CodePipeline、これらをテンプレートから一括構築できる CodeStar といったサービスが提供されています。

## ソースコードを管理するサービス群



## マネジメントサービスの有効利用

AWS を用いることで、大量のサーバーを使ってシステムを構築すること、一時的にサーバーを用意して使うことが容易になってきています。また、近年のアプリケーションは更新頻度を高め、常に新しいサービスを提供することが求められることが多くなっています。その代償として、システムマネジメントにおける管理対象は多くなり、また作業頻度も増えがちです。そういう負担を抑えるためにマネジメントサービスをうまく使い、作業の自動化やサーバーの一元管理を進めていくことが安定したシステムの維持に必要となってきます。

**column**

## AWS の利用料について理解を深めよう

### ・ AWS の利用料は何に対して発生するのか

AWS アカウントを自分で作成して使ってみようとしたとき、まず不安になるのが、「どれだけお金がかかるのか?」ではないでしょうか。AWS は使った分だけお金がかかる従量課金制ですが、サービスによって料金のかかり方は異なります。しかし、基本的な課金の法則はあるので、具体的に「何をどう使った分に料金が発生するのか」を押さえておくことでこういった不安は少なくなるでしょう。

なお各サービスの課金体系は、ここで紹介するパターンのうち 1 つだけでなく複数のパターンを併せ持つことが多くなっています。

### ・ リソースの確保

まず、基本的な考え方として、AWS のリソース（CPU、メモリ、ディスク領域）を利用者が専有している時間分、料金が発生することになります。

つまり、EC2 や RDS のようなリソースの量を指定したインスタンスを作成するタイプのサービスは、インスタンスが稼働している間は実際に利用していないても利用料が発生し続けることになります。こういったサービスを利用する場合は、利用していない時間帯はインスタンスを停止しておくなど、リソースを確保し続けないようにすることで利用料を節約することが可能です。

一方、Lambda や Fargate のようなサーバーレスサービスの場合は、利用されるときだけリソースが確保されます。利用料の考え方はこちらも同じで、リソースが確保されていた時間分の料金が発生するため、実際にサービスを利用している時間分のみ利用料がかかることになります。

### ・ データの保管

AWS 上に保管されるデータの量に対しても利用料が発生します。AWS のデータ保管先には、あらかじめ保存できる容量を決めて確保しておくタイプと、データ量に応じて自動的に容量が確保されるタイプの 2 種類があります。

EBS や Aurora 以外の RDS のストレージは前者のタイプで、確保されている領域のサイズに対して利用料がかかります。たとえば 100GB の容量を確保して 1GB のデータしか格納していない場合も 100GB 分の料金が発生します。

データというものは基本的にシステムを運用していくにつれて増加していくものです。古くて不要になったデータを削除することでもデータの保管にかかる料金を節約することは可能ですが、難しい場合はデータ保管先を変更することも利用料の節約に有効です。とりわけ S3 は安価なストレージであり、データ分析などのために大量に用意されるデータは一度 S3 に保存しておき、必要な分だけを分析サービスなどに都度取り込んで利用するという形が推奨されています。

忘れられがちですが、AWS の機能により取得されるバックアップに対しても保管のための料金がかかります。こちらも「1GBあたりいくら」という形の課金体系になっている場合が多く、古いバックアップデータを削除せずに次々とバックアップを取得し続けると、思わぬところで利用料がかかってしまうため注意が必要です。

#### ・処理量

Route 53 や CloudFront のようなネットワーク、コンテンツ配信サービスではリクエストの処理数や処理したデータ量に対して料金が発生します。同様に CloudTrail や AWS Config のようなセキュリティやマネジメントのためのサービスについても、処理したイベントの数、実行したチェックの数などに対しての課金となります。コンピューティングサービスやストレージサービスのような AWS のリソースを一定の時間確保して動作するタイプでないサービスは、こちらの課金体系であることが多いです。

また、リクエストを受けることで動作する Lambda や S3 のようなサービスは、この課金体系も併せ持っていることが多いです。

#### ・データ転送

AWS を利用するうえで盲点となりやすいのが、データ転送にかかる料金です。Site to Site VPN や Direct Connect のようなネットワークサービスはもちろんですが、EC2 や S3 からインターネットへ出していく通信、AZ やリージョンをまたぐ通信、VPC ピアリングの通信に対しても料金がかかります。なお、インターネットからアクセスされる際に入ってくる通信や、同一 AZ 間の通信に対しての料金はかからないようになっています。

大量のデータをインターネットに配信するシステムでは、この料金が予想外に高くなることもあるので注意が必要です。

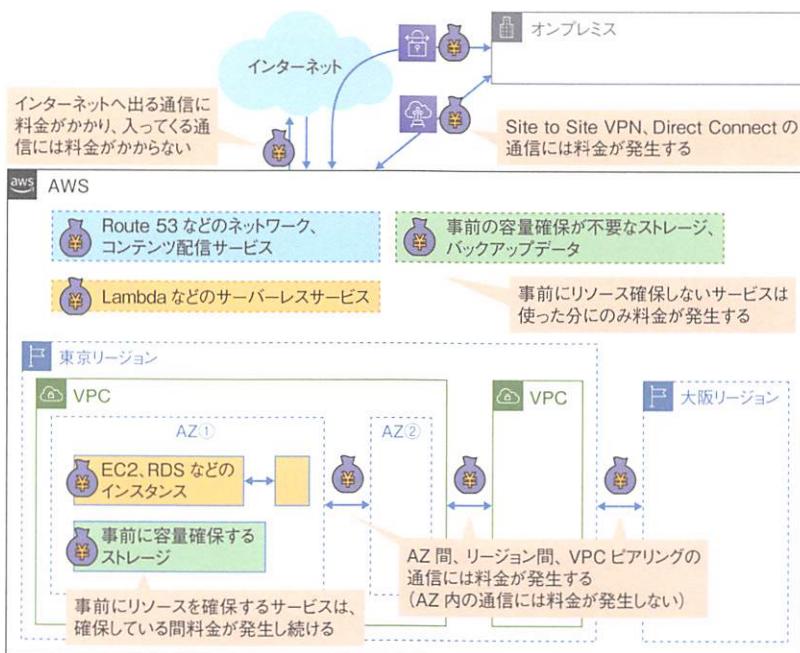
#### ・無料利用枠

AWS には、実験的にサービスを利用したい場合を考慮した、無料利用枠というものがあります。大きく 3 つの種類があり、AWS アカウントを作成してから 12 カ月間のみ使えるもの、対象のサービスを初めて使うときから一定の期間使えるもの、各サービスで常に一定の利用料まで無料とされているものがあります。

たとえば、EC2 はアカウントを作成してから 12 カ月間は Linux の t2.micro または t3.micro インスタンスが毎月 750 時間分無料となります。つまり、1 年間は対象の EC2 インスタンスを無料で起動させ続けられる計算になります。EBS 利用料やデータ転送量はかかるので完全無料とはいきませんが、このように AWS に慣れるため、気軽に試用できる仕組みが用意されています。

AWS をより深く理解するには実際にサービスを利用することも大切です。多彩なサービスが無料で試せるようになっているので、ぜひ無料利用枠を活用してみてください (<https://aws.amazon.com/jp/free/>)。

## AWS のサービス利用料についての基本的な考え方



# Index

## 数字

0.0.0.0/0 ..... 134

## A

ACID特性 ..... 173

Active Directory ..... 232

AD Connector ..... 232

ALB (Application Load Balancer) ..... 148

Amazon API Gateway ..... 74, 166

Amazon Athena ..... 236

Amazon Aurora ..... 174, 182

Amazon CloudFront ..... 156

  ～の設定 ..... 158

  ～の料金 ..... 160

Amazon CloudWatch ..... 77, 246

Amazon CodeBuild ..... 249

Amazon CodeCommit ..... 249

Amazon CodeDeploy ..... 249

Amazon CodePipeline ..... 249

Amazon CodeStar ..... 249

Amazon Cognito ..... 231

Amazon DocumentDB ..... 200

Amazon DynamoDB ..... 186

  ～の保守 ..... 190

  ～の料金 ..... 188

Amazon EBS (Elastic Block Store) ..... 60, 113

  ～の料金 ..... 115

Amazon EC2 (Elastic Compute Cloud) ..... 57, 58

  ～のアクセス制御 ..... 65

  ～の外部公開 ..... 64

  ～の購入方法 ..... 62

～の料金 ..... 63

Amazon EC2 Auto Scaling ..... 67

Amazon ECR (Elastic Container Registry) ..... 86

Amazon ECS (Elastic Container Service) ..... 86

  ～の料金 ..... 89

Amazon EFS (Elastic File System) ..... 116

Amazon EKS (Elastic Kubernetes Service) ..... 90

Amazon ElastiCache ..... 198

Amazon EMR ..... 238

Amazon EventBridge ..... 75, 204, 220, 222

Amazon FSx for Lustre ..... 117

Amazon FSx for Windows File Server ..... 117

Amazon GuardDuty ..... 221

Amazon Inspector ..... 228

Amazon Kinesis ..... 238

Amazon Lex ..... 243

Amazon Managed Blockchain ..... 202

Amazon MemoryDB for Redis ..... 199

Amazon Neptune ..... 201

Amazon OpenSearch Service ..... 237

Amazon Polly ..... 243

Amazon QLDB (Quantum Ledger Database) ..... 201

Amazon QuickSight ..... 238

Amazon RDS (Relational Database Service) ..... 174

  ～のアクセス制御 ..... 178

  ～のデータのバックアップ ..... 180

Amazon Redshift ..... 193

Amazon Rekognition .....	243
Amazon Route 53 .....	149
Amazon S3 (Simple Storage Service) .....	98
～のAPI .....	103
～の料金 .....	100
Amazon S3 Glacier .....	105
Amazon SageMaker .....	242
Amazon SES (Simple Email Service) .....	52, 204
Amazon SNS (Simple Notification Service) .....	204, 220, 222
Amazon Timestream .....	203
Amazon Transcribe .....	243
Amazon Translate .....	243
Amazon VPC (Virtual Private Cloud) .....	60, 130
～のアクセス制御 .....	137
～の構成例 .....	143
～の通信ログ確認 .....	137
Amazon VPCとサブネットの作成 .....	132
Amazon Web Services .....	10
～の学習方法 .....	32
～のサポートプラン .....	35
～の導入事例 .....	11
～の利用料 .....	250
Amazonマシンイメージ(AMI) .....	59, 114
ap-northeast-1 .....	44
ap-northeast-3 .....	44
Auroraレプリカ .....	183
Automation .....	220
AWS App Runner .....	93, 94
AWS Backup .....	119
AWS Batch .....	94
AWS Billing and Cost Management .....	42
AWS CLI (Command Line Interface) .....	39
AWS CLIコマンド .....	40
AWS Client VPN .....	142, 164
AWS CloudFormation .....	245
AWS CloudTrail .....	213
AWS Config .....	217
AWS Configルール .....	219
AWS DataSync .....	119
AWS Direct Connect .....	142, 162
AWS Directory Service .....	232
AWS DMS (Database Migration Service) .....	196
AWS Elastic Beanstalk .....	92, 94
AWS Fargate .....	87
AWS Global Accelerator .....	167
AWS Glue .....	237
AWS Ground Station .....	167
AWS IAM (Identity and Access Management) .....	209
AWS KMS (Key Management Service) .....	111
AWS Lambda .....	69
～の料金 .....	79
～の利用例 .....	74
AWS Lightsail .....	91, 94
AWS Managed Microsoft AD .....	232
AWS Marketplace .....	204, 226
AWS Network Firewall .....	227
AWS Organizations .....	223
AWS Outposts .....	95
AWS PrivateLink .....	140, 165
AWS SCT (Schema Conversion Tool) .....	196
AWS SDK (Software Development Kit) .....	41
AWS Security Hub .....	230
AWS Shield .....	157, 229
AWS Site-to-Site VPN .....	141, 163
AWS Snow Family .....	119
AWS Snowmobile .....	120
AWS Storage Gateway .....	118
AWS Systems Manager .....	61, 220, 247
AWS Transfer Family .....	118
AWS Transit Gateway .....	139, 164
AWS WAF .....	158, 225

AWS Well-Architectedパートナープログ	
ラム	23
AWS Well-Architectedフレームワーク	22
AWSアカウント	34
～の複数利用	223
～へのログイン	36
AWSマネジメントコンソール	37, 38
AZ (Availability Zone)	19, 46
AZ ID	46
AZサービス	143
AZ名	46

## B・C・D

Budgets	43
CDN (Contents Delivery Network)	156
CentOS	54
CIDRブロック	125, 131
CLB (Classic Load Balancer)	148
CloudWatch Logs	246
Cost Explorer	43
CPU使用率	67
CSS	53
DBサーバー	51
DBMS ( DataBase Management System)	170
DDoS攻撃	229
Debian GNU/Linux	54
Design for Failure	22
DNS (Domain Name System)	129, 149
DNSレコード	150
～のタイプ	151

## E・F・G

EDoS攻撃	229
Elastic IP	135
ElastiCache for Memcached	198
ElastiCache for Redis	198
ELB (Elastic Load Balancing)	146
～の料金	148

ENI (Elastic Network Interface)	138
FTP (File Transfer Protocol)	118
GWLB (Gateway Load Balancer)	148

## H・I・J

HTML	53
HTTP	53
HTTPS	53
IaaS	17
IAMグループ	210
IAMポリシー	109, 210
IAMユーザー	37, 209
IAMロール	76, 211
IOPS (Input/Output Per Second)	113
IPv4	123
IPv6	123
IPアドレス	122
JAWS-UG	11
JSON	73

## K・L・N

k8s	90
key-value型データベース	186
Kubernetes	90
Linuxディストリビューション	54
NAT (Network Address Translation)	136
NATゲートウェイ	136
NFS (Network File System)	116
NLB (Network Load Balancer)	148
NoSQLデータベース	171, 198

## O・P・R・S

OS	54
PaaS	17
RCU (Read Capacity Unit)	189
RDB (Relational DataBase)	170
Red Hat Enterprise Linux (RHEL)	54
S3 Analytics	106
SaaS	17

Savings Plans	62
Simple AD	232
SMB (Server Message Block)	117
SMTPサーバー	52
SQL (Structured Query Language)	
	171
SSH	61

## U・V・W

Ubuntu Linux	54
VPCエンドポイント	140
VPCピアリング	139
VPCフローログ	137
VPN (Virtual Private Network)	
	141, 161
WAF (Web Application Firewall)	224
WCU (Write Capacity Unit)	189
Webアプリケーションファイアウォール	
	224
Webサーバー	51, 53
Webサイトホスティング機能 [S3]	110
Windows Server	55

## あ

アウトバウンドルール	65
アクセスキー	210, 211
アクセスコントロールリスト (ACL)	109
アベイラビリティゾーン	19, 46
位置情報ルーティング	153
インサイトイベント [CloudTrail]	214
インスタンス	57, 58, 176
インスタンスタイプ [EC2]	59
～の名前のルール [EC2]	61
インスタンスタイプ [RDS]	176
～の名前のルール [RDS]	176
インスタンスのファミリー [EC2]	62
インターネットゲートウェイ	135
インターフェイスエンドポイント	140
インバウンドルール	65
インメモリキャッシュ	198

インメモリデータストア	198
ウェブ ACL	225
エッジロケーション	156, 204
エンドポイント [RDS]	177
大阪リージョン	44, 45
オブジェクト [S3]	102
オブジェクトストレージ	98
オペレーティングシステム	54
オリジンサーバー	156
オンデマンドインスタンス	62
オンプレミス	12

## か

加重ルーティング	152
仮想化	14
仮想サーバー	14, 56
～の作成	59
仮想サーバーイメージ	82
仮想ネットワーク	130
可用性	22, 206
環境変数	76
関数 [Lambda]	
～の実行状況	77
完全性	206
管理イベント [CloudTrail]	214
キー [S3]	102
機械学習	240
機密性	206
キャッシュサーバー	156
キャパシティーユニット [DynamoDB]	
～	189
強化学習	240
教師あり学習	240
教師なし学習	240
クオーラムモデル	183
クライアント	50
クライアントサイド暗号化	112
クラウド	10, 13
クラウドコンピューティング	13
クラウドサービス	10

グラフデータベース	201
グローバルIPアドレス	123
グローバルサービス	143
クロスリージョンレプリケーション	108
ゲートウェイエンドポイント	140
コード	48
高度なクエリ	218
コスト管理	43
コンテナ	81
コンテナイメージ	82
コンテナオーケストレーションツール	85
コンテナプラットフォーム	87
コンテナランタイム	81, 83
コンテンツデリバリーネットワーク	156
スケジュールスケーリング	68
スタンバイレプリカ	177
ストレージ	98
ストレージクラス [S3]	105
ストレージタイプ [RDS]	176
スナップショット [EBS]	114
スナップショット [RDS]	180
スポットインスタンス	63
責任共有モデル	18
セキュリティグループ	
	60, 64, 65, 137, 228
～のID	66
専用線	161

さ

サーバー	50
～のOS	54
～の仮想化	56
～のバックアップとイメージ管理	114
サーバーレス	15, 69
災害対策	108
サイト間VPN	161
サブネット	132
サブネットマスク	125
時系列データベース	203
システムマネジメント	244
従量課金	20, 42
証跡	213
～の出力	215
情報セキュリティ	206
シンプルルーティング	152
スイッチロール	212
スキーマ	196
スクリプト(クライアントサイド)	53
スクリプト(サーバーサイド)	53
スケール	48
スケールアウト	67
スケールイン	67

た

ターゲット追跡スケーリング	68
ターゲットモニタリング	147
台帳データベース	201
ディープラーニング	242
ディストリビューション [CloudFront]	
	158
データイベント [CloudTrail]	214
データウェアハウス	193, 236
データ可視化	235, 237
データセンター	12
データの暗号化	111
データ分析	234
データベース	170
データベース移行	195
データベースエンジン	174
～のアップデート	179
データベース管理システム	170
データレイク	235
デフォルト	48
デプロイ	48
デリバリー	83
東京リージョン	44, 45
ドキュメント指向データベース	200
ドメイン	129
ドメイン登録機能	150
トランザクション	173

## な

名前解決	129
認可	207
認証	207
ネットワーク ACL	137, 228
ネットワークアドレス	125
ネットワークインターフェイス	138
ネットワーク部	125

## は

バージョン管理 [S3]	107
バケット [S3]	102
～のアクセス制御	109
バケットポリシー	109
バックアップウィンドウ	181
パブリックIP	135
パブリックIPアドレス	64, 123
パブリッククラウド	11, 16
パブリックサブネット	64, 135
ビッグデータ分析	235, 238
非マネージドサービス	26
ビルド	48
ファイアウォール	125
フェイルオーバー	177
フェイルオーバールーティング	154
負荷分散	126, 146
複数回答ルーティング	154
物理サーバー	14, 56
プライベートIPアドレス	123
プライベートクラウド	16, 95
プライベートサブネット	135
プライマリインスタンス	177
フルマネージドサービス	26
プロードキャストアドレス	125
プロセス	81
ブロックチェーン	202
ブロックパブリックアクセス	109
プロビジョンド	15
分散フレームワーク	238
ヘルスチェック機能 [Route 53]	154

ポイントインタイムリカバリ	181, 190
ホストゾーン	149
ホスト部	125
ボリュームタイプ [EBS]	113

## ま

マネージドサービス	17, 26
マルチAZ配置	177
マルチパートアップロード	103
メールサーバー	52
メンテナスウィンドウ	179

## や

ユーザーポリシー	109
予測スケーリング	68

## ら

ライフサイクル設定 [S3]	106
リアルタイム分析	235, 238
リージョン	19, 44
リージョンサービス	143
リードレプリカ	178
リザーブドインスタンス	62
リモートアクセスVPN	161
リモートデスクトップ	61
リレーションナルデータベース	170, 174
ルーター	128
ルーティング	128
ルーティングテーブル	128
ルーティングポリシー [Route 53]	152
ルートテーブル	128, 134
ルートユーザー	36
レイテンシールーティング	153
レジストリ [コンテナ]	84
列指向ストレージ	193
レポジトリ [コンテナ]	84
ロードバランサー	126, 146

## 著者紹介

### 上野 史瑛(うえの ふみあき)

NRIネットコム株式会社でAWSやGCPを中心とした多くのクラウド案件を担当。AWS認定試験は12区分をすべて取得している。業務外でもAWSに関するブログ執筆やイベント登壇を行っている。その活動がAWSにも認められ、2020年/2021年にAPN Ambassadors、APN AWS Top Engineers、APN ALL AWS Certifications Engineersに選出されている。

### 小林 慎平(こばやし きょうへい)

NRIネットコム株式会社入社後、アプリ開発からシステム基盤の構築・運用業務に幅広く携わる。現在はAWSで展開される複数の大規模システムの運用・改善を担当している。資格取得にも力を入れており、IPAの情報処理技術者試験全13区分、AWS認定全12区分を取得済み。2021年にはAPN AWS Top Engineers、APN ALL AWS Certifications Engineersに選出された。

### 尾澤 公亮(おざわ こうすけ)

NRIネットコム株式会社に入社後、基幹システムにおけるP2VやAWS移行案件を通じて、オンプレミス・クラウド双方での開発業務を経験。その後、Webシステムやコンテナベースシステムの設計・開発・運用業務を経て、現在はAWSを中心とした複数のシステム運用・開発を担当している。

AWS認定はアソシエイト全種・CLF・SAPの5区分を取得済み。

### 高梨 友之(たかなし ともゆき)

NRIネットコム株式会社入社後、AWSリプレース案件やオンプレミス運用案件など、クラウド・オンプレミス問わず幅広い業務を実施。現在はAWSでのWebシステム開発を中心に、多くの案件で開発と運用を担当している。

AWS認定はSAA、SAPを取得済み。

■本書のサポートページ

<https://isbn2.sbcr.jp/12818/>



本書をお読みいただいたご感想を上記URLからお寄せください。

本書に関するサポート情報やお問い合わせ受付フォームも掲載しておりますので、  
あわせてご利用ください。

す　かい　　アマゾン　　ウェブ　　サービス　　し　く  
**図解 Amazon Web Servicesの仕組みとサービスが  
たった1日でよくわかる**

2022年2月14日 初版第1刷発行

2022年5月25日 初版第3刷発行

著 者 ..... NRIネットコム株式会社 上野 史瑛・小林 基平・尾澤 公亮・萬梨 友之

発行者 ..... 小川 淳

発行所 ..... SBクリエイティブ株式会社

〒106-0032 東京都港区六本木2-4-5

<https://www.sbcr.jp/>

印 刷 ..... 株式会社シナノ

カバーデザイン ..... 上坊 菜々子

制 作 ..... クニメディア株式会社

企画・編集 ..... 友保 健太

落丁本、乱丁本は小社営業部(03-5549-1201)にてお取り替えいたします。  
定価はカバーに記載されております。

ISBN978-4-8156-1281-8

C0055 ¥2000E



9784815612818

定価 2,200円  
(本体 2,000円 + 税10%)



1920055020008

E1-12 ネットワーク・サーバ

## Contents

- 
- Chapter 1** Amazon Web Services の基礎知識
  - Chapter 2** Amazon Web Services の始め方
  - Chapter 3** コンピューティングサービス
  - Chapter 4** ストレージサービス
  - Chapter 5** ネットワークとコンテンツ配信サービス
  - Chapter 6** データベースサービス
  - Chapter 7** セキュリティ、アイデンティティサービス
  - Chapter 8** 知っておきたいその他のサービス