

AWS をはじめよう

mochikoAsTech 著

2018-10-08 版 mochikoAsTech 発行

はじめに

2018年10月mochikoAsTech

この本を手に取ってくださったあなた、こんにちは、あるいははじめまして。「AWSをはじめよう」の筆者、mochikoAsTechです。

AWSは好きですか？それとも好きとか嫌いとか言えるほど、AWSのことをまだよく知らない段階ですか？

本著「AWSをはじめよう」ではAWSでサーバを立てたり、WordPressをインストールしたりして、実際にブラウザで自分のサイトが見られるところまでを手を動かして実践していきます。

ちなみに本著「AWSをはじめよう」(以下AWS本)は、前作「DNSをはじめよう」(以下DNS本)のストーリーの続きとなっていますので、DNS本を読まずにいきなりAWS本から読むと「上中下巻セットなのに中巻からいきなり読んだ」という感じで色々意味が分からずちょっと戸惑うことになります。

読み進んでいくと第2章辺りで「さてここで事前に下茹でしておいたじゃがいもを取り出します」といわれて「は？下茹でとかいつしてたの？！」という状態になりますので、「DNSは興味ないし面倒くさいんだけど・・・」という方もできればDNS本をお読みいただいて、下ごしらえを済ませた状態でAWS本を開いてみてください。きっとその方が美味しくお召し上がりいただけます。第1章はDNS本を読んでいなくても問題ない内容ですので、とりあえずそのまま読んでいただいても構いません。

AWSの普及によって「アプリケーションエンジニアは開発だけやっていればいい、サーバ周りはインフラエンジニアに任せておけばいい」という完全分業の時代が終わり、今まで聖域化されていたインフラやサーバの世界に、アプリケーションエンジニアやフロントエンドエンジニアも気軽に踏み込めるようになってきました。嫌でも踏み込まざるを得ない時代になってきた、ともいえます。

ですがソースコードを書くアプリケーションエンジニアと違って、インフラエンジニアが実際どんなことをしているのかなんて想像もつかない、「サーバを立てろ」といわれても何をどうしたらいいのか分からない、という人も少なくないのではないでしょうか。

でもインフラってやってみると意外と楽しいんです。そして土台であるインフラを学ぶことで、上もののアプリで頑張っていたことがあっさり解決できる、という場面も結構あったりします。

本著は AWS が、そしてサーバやインフラが、怖いものではなくすごく楽しいものなんだよということをかつての私のような初心者へ伝えたくて書いた一冊です。読んで試して「面白かった！」と思ってもらえたなら、そしてインフラを前より少しでも好きになってもらえたなら何より嬉しいです。

想定する読者層

本著は、こんな人に向けて書かれています。

- AWS が何なのかよく分かっていない人
- ブログやポートフォリオサイトを独自ドメインで作ってみたい人
- JavaScript や HTML や CSS なら書けるけどサーバは分からなくて苦手という人
- プログラミングの勉強がしたいけど環境構築でつまづいて嫌になってしまった人
- これからシステムやプログラミングを学ぼうと思っている新人
- ウェブ系で開発や運用をしているアプリケーションエンジニアの人
- インフラやサーバになんとなく苦手意識のある人
- AWS、EC2、RDS、ELB、Auto Scaling、Route53などの単語に興味がある人
- クラウドってなんだろう？ サーバってなんだろう？ という初心者

本著の特徴

本著では前作「DNS をはじめよう」で買ったドメインを使って、実際に WordPress で自分のサイトを作ります。手を動かして AWS でサーバを立てたりネットワークの設定をしたりしながら学べるので理解がしやすく、インフラ初心者でも安心して読み進められる内容です。

また実際にありがちなトラブルをとり上げて、

- 上手くいかないときは原因をどう調べたらいいのか？
- 見つかった問題をどう解決したらいいのか？
- 今後、同様の問題はどうしたら事前に避けられるのか？

を解説するとともに、実際にコマンドを叩いて反復学習するためのドリルもついています。

本著のゴール

本著を読み終わると、あなたはこのような状態になっています。

- WordPress のおしゃれなサイトができあがっている
- 使うも壊すも自由な勉強用の Linux サーバ環境が 1 台手に入る
- オンプレミスとクラウドの違いやメリットデメリットが説明できるようになっている
- 読む前より AWS やサーバや黒い画面が怖くなくなっている

免責事項

本著に記載されている内容は筆者の所属する組織の公式見解ではありません。

また本著はできるだけ正確を期すように努めましたが、筆者が内容を保証するものではありません。よって本著の記載内容に基づいて読者が行った行為、及び読者が被った損害について筆者は何ら責任を負うものではありません。

不正確あるいは誤認と思われる箇所がありましたら、電子版については必要に応じて適宜改訂を行いますので GitHub のイシュー や プルリクエストで筆者までお知らせいただけますと幸いです。

<https://github.com/mochikoAsTech/startAWS>

目次

はじめに	3
想定する読者層	4
本著の特徴	4
本著のゴール	5
免責事項	5
第1章 インフラとサーバってなに？	11
1.1 AWSを理解するには先ずインフラを知ろう	12
1.2 インフラとは？	12
1.3 サーバとは？	13
1.3.1 サーバの姿を見てみよう	17
1.3.2 サーバはデータセンターにいる	19
1.3.3 物理サーバと仮想サーバ	23
1.4 オンプレミスとクラウド	25
1.4.1 オンプレミスとは？	25
1.4.2 クラウドとは？	25
1.4.3 クラウドのメリットとデメリット	26
1.4.4 AWSはAmazonがやっているクラウド	28
1.4.5 パブリッククラウドとプライベートクラウド	28
1.4.6 AWS以外のクラウド	29
第2章 AWSを使い始めたら最初にやること	31
2.1 AWS無料利用枠を使おう	32
2.2 AWSのアカウント作成	33
【コラム】「DNSをはじめよう」はどこで買える？	34
2.3 マネジメントコンソールにサインイン	34
2.3.1 【ドリル】AWSの管理画面はなんて名前？	36

2.4	IAM でユーザの権限管理	37
2.4.1	ルートユーザーの普段使いはやめよう	37
2.4.2	IAM ユーザを作ろう	38
2.4.3	MFA (多要素認証) で不正利用から IAM ユーザーを守る	51
2.4.4	ルートユーザーも MFA を有効にする	64
2.5	リージョンの変更	66
2.6	CroudTrail でいつ誰が何をしたのか記録	69
第 3 章	AWS でウェブサーバを立てよう	73
3.1	事前準備	74
3.1.1	お使いのパソコンが Windows の場合	74
3.1.2	お使いのパソコンが Mac の場合	77
3.2	EC2 でウェブサーバを立てる	77
3.2.1	ステップ 1: Amazon マシンイメージ (AMI)	79
3.2.2	ステップ 2: インスタンスタイプの選択	80
3.2.3	【コラム】T2 系バーストモードの落とし穴	82
3.2.4	ステップ 3: インスタンスの詳細の設定	83
3.2.5	ステップ 4: ストレージの追加	83
3.2.6	ステップ 5: タグの追加	83
3.2.7	ステップ 6: セキュリティグループの設定	83
3.3	SecurityGroup	83
3.4	VPC	83
3.5	EC2	83
3.5.1	請求アラート	83
3.5.2	SSH の鍵認証	83
3.5.3	鍵の変換	83
3.5.4	ElasticIP	83
3.5.5	Bastion	83
第 4 章	サーバのバックアップを取っておこう	85
4.1	スナップショット	85
4.2	AMI	85
第 5 章	ELB でバランスングやサーバの台数を管理しよう	87
5.1	ELB	87

5.2	Auto Scaling	87
5.2.1	スケーリングに使える	87
5.2.2	サーバが 1 台死んでも自動で 1 台立ち上がる	87
第 6 章	DB サーバを立てよう	89
6.1	RDS	89
6.2	Amazon Aurora	89
第 7 章	ネームサーバの設定をしよう	91
7.1	Route53	91
第 8 章	もっと AWS について勉強したい！	93
8.1	公式のオンラインセミナーや資料集	93
8.2	AWS 認定資格のクラウドプラクティショナーを目指してみよう	93
第 9 章	AWS をやめたくなったらすること	95
9.1	無料の 1 年が終わる前にすべきこと	95
9.1.1	【ドリル】サンプル	95
付録 A	本当の Git	97
A.1	Git - ぎゅつと言えないトウインクル	98
あとがき		99
Special Thanks:	99
レビュアー	99
参考書	99
著者紹介		101

第 1 章

インフラとサーバってなに？

この章では AWS とはなにか？ そもそもクラウドとは何か？ サーバとは何か？ という基本を学びます。

1.1 AWS を理解するには先ずインフラを知ろう

AWS とは Amazon Web Services（アマゾン ウェブ サービス）の略で、欲しいものをぽちっとな！ すると翌日には届くあのアマゾンがやっているクラウドです。

「AWS はアマゾンがやっているクラウドです」と言われても、「クラウド」が分からないと結局 AWS が何なのかよく分からぬままですよね。

クラウドって何なのでしょう？

クラウドだけではありません。よくクラウドと一緒に並んでいるサーバやインフラという言葉がありますが、こちらも何だか分かりますか？ IT 系で働いていても、その辺って「なんか・・・ふんわり・・・なんか雲の向こう側にある・・・ウェブサイト作るための何か・・・？」という程度の認識で、クラウドってなに？ とか、サーバってなに？ と聞かれたときに、ちゃんと説明できる人は意外と少ないので私は思います。

なので、先ずは「AWS はアマゾンがやっているクラウド」という文章の意味が分かるよう、インフラ周りから順を追って学んでいきましょう。

1.2 インフラとは？

インフラという言葉は知っていますか？

はじめて聞いたという人も、「なんとなくは分かるけど、説明してと言われたらうーん・・・」な人も、いま自分が考える「インフラ」についての説明をここに書いてみましょう。いきなり正解を聞かれるより、自分で答えを考えて書き出してみてからの方が、正解を聞いたときにきっと自分の中へより染み渡ってくるはずです。

私が考えるインフラとは



のことである

では答え合わせをしてみましょう。

インフラとはサーバやネットワークのことです。

そもそもインフラこと「Infrastructure」は、直訳すると基盤や下部構造といった意味です。ですので「生活インフラ」と言うと一般的には上下水道や道路、そしてインターネットなど、生活に欠かすことの出来ない社会基盤のことを指します。

そして技術用語としては、インフラはシステムやサービスの基盤となる「設備」のことを指します。なので、分かりやすく言うと「インフラとはサーバやネットワークのこと」なのです。

これでもう会社で後輩に「インフラってなんですか？」と聞かれても、堂々と「サーバとかネットワークのことだよ」と答えられますね！

でも後輩に、続けて「え、サーバってなんですか？」と聞かれたらどうしましょう？

1.3 サーバとは？

ウェブ業界で働いている人にとってサーバは身近な存在です。業務上でテストサーバや本番サーバ、ウェブサーバやデータベースサーバなどの単語を見聞きすることはよくあると思いますし、「アクセスが殺到してサーバが落ちた」とか「ソシャゲのアプリを落とそうとしたのにサーバが重くてなかなか落とせない」という会話って割とあると思います。

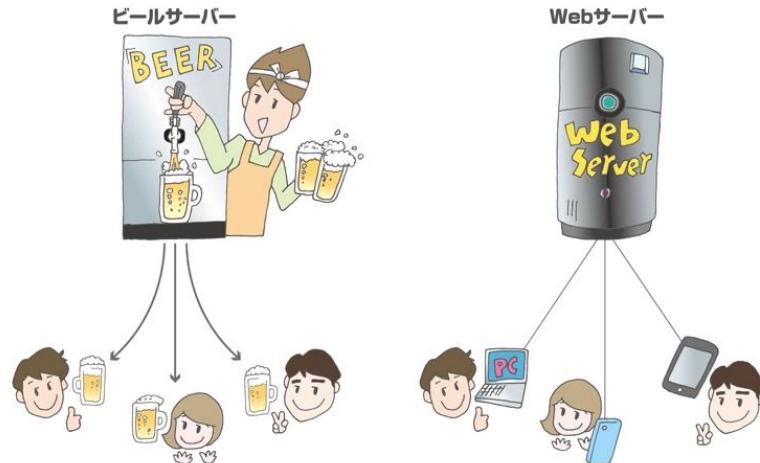
でももし後輩から「サーバってなんですか？」という直球の質問を投げつけられたら、しっかりとホームランで打ち返せますか？

サーバっていったい何なのでしょう？

サーバとはクライアントに対してサービスを提供するものです。でも「サービスを提供するもの」と言われても、ちょっとふんわりした表現すぎて分かりにくいで具体例を出しましょう。

サーバと名の付くもののひとつにビアサーバがあります。

前述の「サーバとはクライアントに対してサービスを提供するものである」という文章を、ビアサーバに当てはめてみましょう。「ビアサーバとは客に対してビールを提供するものである」(図 1.1)、さっきまで分かりにくかった文章もビアサーバを当てはめたら急に分かりやすくなりましたね。



▲図 1.1 ビアサーバもウェブサーバもクライアントに対してサービスを提供するものである

それではビアサーバと同じようにウェブサーバも前述の文章に当てはめてみましょう。「ウェブサーバとは、客に対してウェブページを提供するものである」、こちらも具体的にしたらとても分かりやすいですね。

ビアサーバに対して「ビールをください」というリクエストを投げると、つまりビアサーバのコックを「開」の方へひねると、ビールというレスポンスが返ってきます。ウェブサーバに対して「ウェブページを見せてください」というリクエストを投げれば、ウェブページというレスポンスが返ってきます。

つまりビアサーバもウェブサーバも、その本質は「Server」の直訳である「給仕人」なので、繰り返しになりますがサーバとは**クライアント**に対して**サービスを提供するもの**なのです。

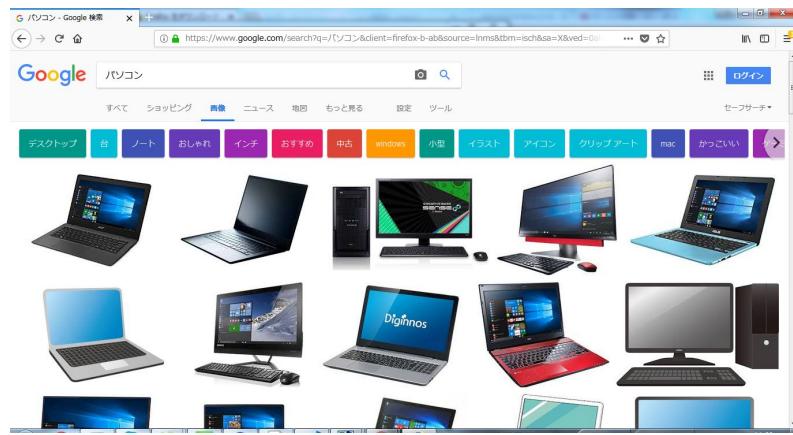
ところでちょっと頭の中でサーバの姿を思い浮かべてみてください。できればイラストじゃなくて実写でお願いします。



▲図 1.2 サーバの姿を実写で思い浮かべて描いてみよう

思い浮かびますか？ あんまり浮かばないですよね。サーバの姿がくっきり思い描ける人の方が少ないのでないかと思います。

でもこれが「パソコンの姿を思い浮かべてください」なら、きっとすぐに浮かんでくるはずです。（図 1.3）



▲図 1.3 パソコンの姿ならすぐに思い浮かぶ

でもサーバの姿は？ と考えると、先ほどとは打って変わってなかなか思い浮かびません。

ではサーバの姿をお見せしたいと思います。

1.3.1 サーバの姿を見てみよう



▲図 1.4 サーバの姿（HPE ProLiant DL360）

じゃじゃーん！ これがサーバの姿です！

これは Hewlett Packard Enterprise (ヒューレット・パッカード エンタープライズ) の HPE ProLiant DL360^{*1}というラックマウント型のサーバです。DL360 は 15 年以上前から愛されているシリーズ^{*2}で、日本でもっとも売れたラックマウント型のサーバと言っても過言ではないかも知れません。1 台につき定価でおおよそ 50 万円以上します。

そして本は本棚に収めるように、サーバはサーバラック（図 1.5）^{*3}という専用の棚に收めことが多いです。

^{*1} HPE ProLiant DL360 <https://www.hpe.com/jp/ja/product-catalog/servers/proliant-servers/pip.hpe-proliant-dl360-gen10-server.1010007891.html>

^{*2} ちなみに 2018 年 8 月時点で発売されているのは HPE ProLiant DL360 Gen10 です。末尾の「Gen10」は世代 (Generation) を表しているので、10 世代目ということです。

^{*3} 写真のサーバラックは HPE Advanced G2 シリーズ <https://www.hpe.com/jp/ja/product-catalog/servers/server-racks/>



▲図 1.5 サーバを収めるためのサーバラック

先ほどの HPE ProLiant DL360 のようなサーバは、このラック（＝棚）にマウントする（＝乗せる）ことができる形状のため「ラックマウント型サーバ」、略してラックサーバと呼ばれています。

ラックマウント型のサーバは 1U（ワンユー）^{*4}・2U・4U のように厚みが異なり、1U のサーバならこのサーバラックの 1 ユニット（1 段）分、2U のサーバなら 2 ユニット（2 段）分を使うことになります。そのためラックマウント型サーバは 1U サーバという名前で呼ばれることがあります。サーバラックは 42U サイズが多く、その名前のとおり 1U サーバを 42 台収めることができます。^{*5}

^{*4} 1U の厚みは 1.75 インチ（44.45mm）です。

^{*5} 但しラックに供給される電源の量や放熱の問題もあるため、実際は 42U サイズのラックにサーバ 42 台をぎちぎちに詰めるとは限りません。



▲図 1.6 タワー型サーバとブレードサーバ

「ラックマウント型サーバ」だけでなく、デスクトップパソコンのような形状の「タワー型サーバ」(図 1.6)^{*6}や、シャーシやエンクロージャーと呼ばれる箱の中にサーバを何本も差し込んで使う省スペースな「ブレードサーバ」^{*7}など、サーバには色々な形があります。

そしてこうしたラックマウント型サーバ、タワー型サーバ、ブレードサーバのように、手で触れる実体があるサーバのことを**物理サーバ**といいます。物理的な実体があるから物理サーバです。(この「物理サーバ」という言葉は後でまた出てきますので覚えておいてください)

そもそもですが、人がウェブサイトを作る時には土台となるサーバが必ず必要となります。たとえばてなブログで無料のブログを作ったときでも、ameba owned で無料のホームページを作ったときでも、あなた自身はサーバのことなど気にも留めないと思いますが、どこかしらに必ずそのブログやサイトが乗っかっているサーバは存在しています。

ではそれらのサーバはいったいどこにいるのでしょうか？

1.3.2 サーバはデータセンターにいる

前述のサーバラックや、その中に詰まったラックサーバを実際に見たことはありますか？

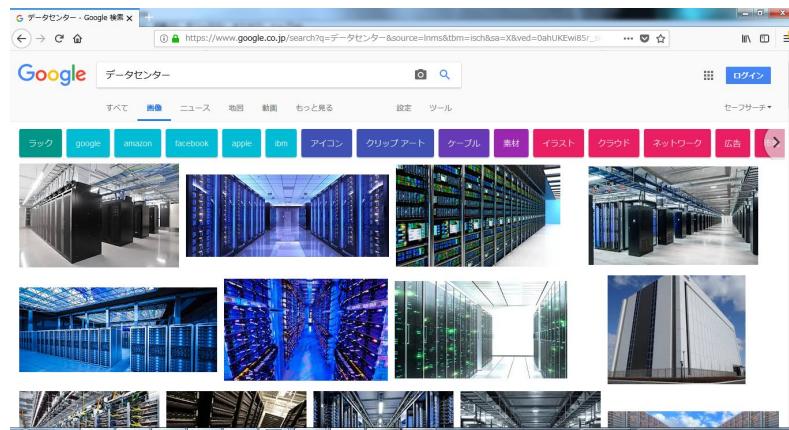
どんなウェブサイトも、世界中のどこかにあるサーバの中で稼動しているはずなのですが、インフラエンジニアでなければサーバを見る機会はなかなかないかも知れません。

^{*6} <https://www.hpe.com/jp/ja/product-catalog/servers/proliant-servers/pip-hpe-proliant-ml110-gen10-server.1010192782.html>

^{*7} <https://www.hpe.com/jp/ja/integrated-systems/bladesystem.html>

サーバはほとんどの場合、データセンターと呼ばれる場所に設置されています。^{*8}

カラオケをするには防音や音響設備の整ったカラオケルームが適しているように、サーバを動かすためのさまざまな設備が整った場所のことをデータセンター、略して DC^{*9}といいます。先ほどのラックサーバがたくさん並んでいますね。（図 1.7）



▲図 1.7 データセンターはサーバのための設備が整っている

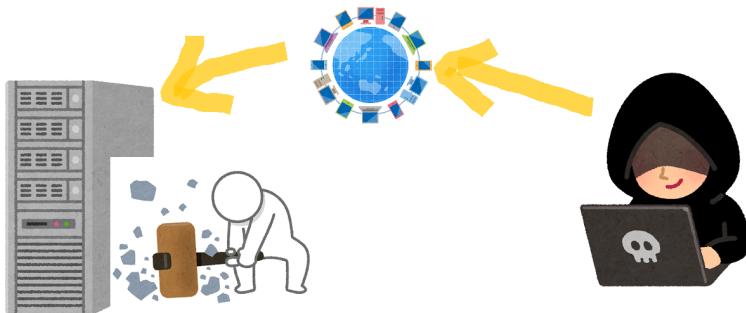
しかし「サーバを動かすための設備」と言われても、「電源取れればそれでいいんじゃないの？ 専用の建物なんか要る？」と思われるかも知れません。サーバを動かすのに適した設備とはどんなものなのでしょう？

〈サーバに適した設備〉 1. 防犯設備

もし悪い誰かが「この企業が気に食わないから商品のウェブサイトを落としてやろう」と思ったとき、あるいは「このネットショップの顧客情報を根こそぎ盗んでやろう」と思ったとき、ネット越しにサイトを攻撃したり侵入したりするだけでなく、そのサイトが動いているサーバのところまで行って物理的に破壊したり、ハードディスクを引っこ抜いて盗んだり、という手段があります。

^{*8} 企業によっては、オフィス内にサーバルームがあってサーバはそこに設置されているかも知れません。趣味で自宅にサーバを置いている人もいますね。

^{*9} データセンター（Data Center）の頭文字を取って DCですが、の中でもインターネット用途向けのデータセンターはインターネットデータセンター、略して iDC と呼ばれたりします。「逆にインターネット用途以外のデータセンターってなに？」となります。メインフレーム（インターネット以前の時代の大型コンピュータ）向けのデータセンターということのようです。



▲図 1.8 攻撃や窃盗はインターネット越しでも直接でもできる

データセンターは後者の「物理的な攻撃や侵入」からサーバを守るための設備を整えています。

堅牢さはデータセンターによって異なりますが、たとえば次のような防犯対策が取られています。

- 所在地を一般に公開しない^{*10}
- 建物自体に侵入経路となる窓がない
- 厳重な警備体制
- 事前予約をした上で顔写真つきの身分証を提示しないと建物に入れない
- エントランスで空港と同じような手荷物チェックや金属探知機チェックがある
- 上着や荷物、携帯電話、カメラなどは持ち込み禁止
- 借りているサーバラックがある階にしかエレベータが止まらない
- サーバルームへの入退室は監視カメラと生体認証で記録
- 入るときと出るときで体重が違うと出られない

入退出時の体重チェックは健康のためではなく、盗んだハードディスクを持ち出せないようにするためです。また「うちは弱小サイトだから誰かのうらみも買わないし、盗まれるような個人情報もないよ」という場合でも、防犯だけでなく天災や熱の対策も必要です。

^{*10} 2018年7月、都内で建築現場の火災が発生した際に「この建物はAWSのデータセンターとして建築していた可能性が高い」というニュースが出ており、断定はしていなかったものの「それは報道していいの？周知の事実になってしまったら、もう一度同じ場所に立てるの無理なのでは・・・？」とちょっと気になりました。

〈サーバに適した設備〉2. 地震・火事対策設備

ウェブサイトはサーバ上で稼動しているため、サーバが止まればもちろんサイトも見られなくなってしまいます。^{*11}

もし地震などの天災があったときにも絶対にサーバを止めないため、データセンターの建物は耐震構造になっていることはもちろん、次のような電力供給対策もされています。

- 変電所から電力を受ける受電設備は複数用意して冗長化している
- もし停電があっても電力が途絶えないよう、電力は複数の変電所から引いている
- 万が一完全に電力が途絶えたら即座にUPS（無停電電源装置）が起動
- さらに数分以内に自家発電機が稼動し、最低数日間は追加の燃料給油なしで稼動可能
- 燃料（重油やジェット燃料）は販売元業者と有事の優先供給の契約をしており、供給が続く限りは自家発電で稼動し続けることが可能

電源がなくなればパソコンも落ちてしまうように、サーバも、その上で稼動しているウェブサイトも、電力が供給されなければ落ちてしまいます。仮にサーバを2台用意して「1台壊れても、もう1台でサイトは見られる！大丈夫！」と安心していても、データセンターの電力そのものが止まってしまえば、どちらのサーバも電源が切れてサイトは見られなくなります。電気は使えて当たり前と思いがちですが、東日本大震災後の輪番停電のように「当たり前が崩れたとき」にも、いつもどおり稼動できる環境がデータセンターには求められているのです。

さらに万が一火事が起きてても、サーバにじゃんじゃん水をかけて消火するわけにはいきません。火は酸素をエネルギーにして燃えるので、多くのデータセンターでは酸素以外のガスで部屋を満たして消火するガス消火設備を備えています。

〈サーバに適した設備〉3. 空調設備

そして一生懸命稼動しているサーバはとても熱くなります。皆さんのパソコンにもファンなどの冷却機構が付いていて、使っていると自分自身の熱を冷まそうとしますよね。サーバも同じで、たくさんのサーバが詰まったサーバラックの裏側には熱い空気がいっぱい吐き出されてきます。

暑い部屋ではサーバが故障したり落ちてしまったりする^{*12}ため、データセンター内の

*11 冗談のようですが「こちらのサーバを停止して削除しますね」「はい、使ってないのでいいです」という会話をした後で、実際にサーバを削除したら「サイトが見られなくなったんですけど！」という連絡が来た、という話も聞きます。サーバがなければサイトは見られない、というのは決して万人にとって当たり前のことではないのです・・・・ないのです・・・。

*12 アニメ映画の「サマーウォーズ」で、冷却用の水が部屋からなくなったことでスーパーコンピュータが熱

サーバルームの空調はとても強く、人間が過ごすにはちょっとつらい寒さです。

このように防犯、地震・火事対策、空調といった設備が整ったデータセンターで、サーバは日々元気に稼動しているのです。

以上、サーバってどこにいるの？ というお話をしました。

1.3.3 物理サーバと仮想サーバ

ところで部屋を借りるとき、「1DK の部屋なら 8 万円、2LDK の部屋なら 12 万円」のように広いほうが家賃は高くなります。データセンターを借りるときもまったく同じで「ラックの 1/2 なら 12 万円、1 ラックまるごとなら 20 万円」のように、ラックサイズによって月額の費用が変わってきます。

そして 2LDK の部屋に 1 人で住むと、1 人で 20 万円負担しなければなりませんが、シェアハウスにして 10 人で住めば 1 人当たりの家賃コストは 2 万円に下がります。サーバラックも同様で、42U のラックに 1U サーバを 1 台しか乗せないより、42 台詰め込んだ方が 1 台当たりのラック使用コストは下がります。

2000 年代前半、インターネットが盛り上がってきてサーバが必要になってきた頃、42U のラックになんとかもっとたくさんのサーバを詰め込めないか？ と試行錯誤した結果、前述の省スペースなブレードサーバや、1U の半分サイズで奥と手前に 2 台収納できる 1U ハーフサイズのサーバなどが台頭してきました。

しかし 1 つのラックに割り当てられた電源の量には上限があるため、42U にぎちぎちに詰め込むと今度は電源が足りなくなってしまいます。^{*13}データセンターによっては、電源容量を増やせるオプションを提供しているところもありますが、それはそれで「10A 増やしたら 3 万円」のように月額費用に跳ね返ってきます。

ラックのスペースは決まっている、使える電源の容量も決まっている。でもそこに置けるサーバの台数を増やしたい！ そこで「物理サーバのサイズを小さくする」とは別のアプローチで生まれてきたのが仮想サーバです。

物理サーバが一軒家だとすれば、仮想サーバはマンション（図 1.9）です。

暴走し、主人公が大事なゲームに負けてしまうシーンを思い出してください。

^{*13} ぎちぎちに詰め込んでなんとか稼動していたものの、ある日データセンターで停電が起きて全台停止。電源はすぐに復旧して自動で全台一斉に起動しようとしたが、サーバは起動時がいちばん電源を食うため、ラックのブレーカーが落ちて再度全台停止。一斉に起動しようとしてはブレーカーが落ちて全台停止、をずっと繰り返していた・・・という怪談を聞いたことがあります。本当に怖い話ですね。



▲図 1.9 物理サーバは一軒家、仮想サーバはマンション

一軒家には基本的に 1 世帯しか住めません^{*14}が、マンションにすれば土地のサイズは同じままで 10 世帯住むことができます。1 台の物理サーバをそのまま使うのではなく、物理サーバ上に何台ものバーチャル（仮想的）なサーバを作ることで、サーバラックのサイズはそのままで論理的なサーバ台数を増やすことができたのです。

このときマンションの建物にあたる物理サーバをホストサーバ、101 号室や 201 号室のような各部屋にあたる仮想サーバをゲストサーバと呼んだりします^{*15}。

前述のとおり物理的な実体があるのが物理サーバですが、その逆で手で触れる物理的な実体がないのが仮想サーバです。手で触れられるのはあくまでホスト OS のサーバであり、ゲスト OS のサーバはその中に仮想的にしか存在しないため、手で触ることはできません。

同じ広さのラックスペースに、今までよりたくさんのサーバが詰め込めるなんて仮想サーバ素晴らしい！ と思いますが、一軒家よりマンションの方が建築コストが高いのと同じで、仮想サーバを立てるにはホストサーバとなる物理サーバのスペックも高くなればならないため、初期投資額がぐっと高くなります。

データセンターで借りるラックスペース代も高いし、物理サーバだって何十万もします。スペースを切り詰めるために仮想サーバにしたいと思っても、ホストサーバとなる物理サーバはスペックが高いのでさらに高額・・・となると中小企業やスタートアップ企業が自社で物理サーバや仮想サーバを所有・管理するのはなかなか大変なことです。

そこで資本力のある会社が大きなホストサーバをたくさん立てて、その上の仮想サーバ（ゲストサーバ）を他の人に貸すような仕組みが生まれました。

長々と物理サーバと仮想サーバについて説明してきましたが、なんとなくゴールがお分

*14 一軒家で 2 世帯同居だってあるでしょ！ という突っ込みは心にしまってください。

*15 ホスト OS、ゲスト OS という呼び方をすることもあります。

かりでしょうか？勘のいい方はもうお気づきのことだと思いますが、ここでようやく「クラウドとは何か？」という話と繋がってきます。

1.4 オンプレミスとクラウド

1.4.1 オンプレミスとは？

昔々は、企業が「そろそろ自社のウェブサイト作りたいなあ・・・だからサーバが必要だ！」と思ったら、「サーバを買う」という選択肢しかありませんでした。

サーバを買うと言っても、お店に行ってぱっと買って持ち帰れる、という訳ではありません。

「どのメーカーのサーバにしよう？HPEかな？それともIBMかな？DELLがいいかな？」と各社の見積もりを販社経由で取り、値引き交渉をして、それでも数十万から数百万するので社内の裏議を通してやっと購入。購入してもすぐ届くわけではなく、数週間待ってやっと届きます。そして届いたらサーバを段ボールから出して、データセンターもしくは自社のサーバルームにあるサーバラックのところまで持つて行って、がっちゃんこと設置。^{*16}

設置できたら今度は同じく自前のネットワーク機器からLANケーブルを繋ぎ、電源も繋ぎます。そしてOSのインストールディスクを用意してサーバにOSをインストールして・・・以下省略しますが、要は「ただ自社のウェブサイトが作りたいだけなのに、サーバを用意するまでがすごく大変だった」ということです。

自分でサーバを買って、何もかも自分で用意しないといけないため、

- 初期投資のサーバ代が高い
- サーバを置くのに適した場所も必要
- 「欲しい！」と思ってから使い始めるまでに時間がかかる

という状況でした。このようにインフラを自前で用意して、自社で所有・管理するのがいわゆるオンプレミスです。

1.4.2 クラウドとは？

これに対してクラウドは、オンプレミスと違ってサーバを買うのではなく、サービスとして「使う」だけです。

クラウドなら「自社のウェブサイト作りたいなあ・・・だからサーバが必要だ！」と

^{*16} 昨今はあまり聞かない単語ですが、サーバを箱から出してセットアップする一連の作業を「キッティング」といいます。

思ったら、ブラウザを開いて、クラウド事業者のサイト上で使いたいサーバのスペックを選んでぽちっとなするだけで、すぐにサーバを立てることができます。しかも Amazon のクラウドこと AWS なら課金も 1 秒単位の従量課金なので、たとえばサーバを 5 分使つたら 5 分ぶんの費用しかかかりません。こんな簡単にサーバを使い始めたりやめたりできるのは、クラウド事業者が物理サーバそのものを提供しているのではなく、性能のいいホストサーバをたくさん用意しておいて、その上に立てた仮想サーバ（ゲストサーバ）を提供しているからです。

オンプレミスはサーバを買って使う、クラウドはサービスとして使うということですね。でもまだちょっとわかりにくいと思うので、お店を例にオンプレミスとクラウドの違いを確認してみましょう。

1.4.3 クラウドのメリットとデメリット

たとえば私が突然ピザ作りに目覚めて、もうインフラエンジニアなんかやってる場合じゃない！ ピザ屋を始めるんだ！^{*17}と思いついたとします。

ピザ屋さんをオープンすべく、土地を買って、そこに店舗となる建物を建てて、電気とガスと水道を通して、床板や壁紙を貼って・・・からやると、お金も場所も時間もたくさん必要です。しかも準備が整ってやっとオープンしたと思ったら、たった 1 か月で資金が足りなくなってしまってお店がつぶれることになったとしても、今度は建物の取り壊しや土地の処分など、止めるときは止めるときでやることがたくさんあります。このように全部自分で買って、自分で所有・管理するオンプレミスだと、「ちょっと気軽にピザ屋さんをやってみよう」はなかなか厳しいことが分かります。

一方クラウドは、フードコートへの出店に似ています。「ピザ屋をはじめたい！ だからフードコートの一区画を借りてやってみよう！」という感じです。

これだと建物はもうあって、電気ガス水道ももう用意されています。フードコート内の一区画を契約して使わせてもらうだけなので、すべて自分で準備するオンプレミスと違ってすぐに始められます。しかも数か月やってみて「もうピザ焼くの飽きたわー！」と思ったら、その区画を借りるのを止めるだけでいいのです。前述のとおり AWS なら使い始めの初期費用もなく 1 秒単位の従量課金なので「ピザ屋さんもうやめたいけど、この先 30 年のローン支払いが残ってるからやめるにやめられない・・」ということもあります。「前月の 25 日までに契約終了を申し出る必要がある」といった制限すらないので、本当にいつでもやめられます。

クラウドならとても簡単に出店できる（つまり簡単にサーバを用意できる）ので、私は本来やりたかった「美味しいピザを焼いて売る」（ウェブサイトを作って自社を宣伝する）

^{*17} そしたら「AWS をはじめよう」の続編として「ピザ屋をはじめよう」という本が書けますね。

という本業に注力できます。

さらに、もしピザ屋さんが大繁盛したら、フードコート内で自店の隣の区画も借りて、お店を広くすることも簡単にできるので、初めから広い区画を借りておく必要もありません。つまり、ウェブサイトへのアクセスが増えてきてサーバのスペックが足りなくなったら、後からサーバを増強したり好きなだけサーバの台数を増やしたりもできるので、最初から高スペックなサーバを借りておく必要がないということです。

クラウドなら初期投資額が少なく、すぐに始められて、すぐにやめられる。よく「クラウドはスマートスタートに向いている」と言われますが、その理由はまさにこういうところにあるのです。

一方でオンプレミスにもメリットはあります。自分が夢見るピザ屋さんのイメージに合わせて好きな広さや造りの建物を設計するところから始めるので、フードコートとは違って自由度がとても高くなります。たとえばクラウドならメニューにあるサーバから選ぶことしかできませんが、オンプレミスなら「CPUは最小限でいいけどメモリとハードディスクはめいっぱい積みたい」といったように、サービスに最適なサーバをこだわって作ることができる、ということです。

またサーバを購入して所有していれば会社の資産となりますし、クラウドの場合はどれだけ長く使ってもサーバは自社のものにはなりません。あくまで借りているだけです。会計の視点から見るとオンプレミスの場合はサーバ代は固定費となるため先々の見通しもつけやすいですが、クラウドの場合は使った分だけの変動費となるため費用の予測はあくまで予測となります。

さらに長い目で見るとフードコートにテナント料を払い続ける方が、土地や建物を買うよりも最終的には高くなるかも知れません。前述のとおり初期投資は少なくて済むのですが、クラウドのいいところは決して「コストが安くなる」ということではありません。実際、AWSは他社の共有レンタルサーバやVPS^{*18}と比べると高額です。

クラウドのよいところは、前述のすぐに始められてすぐにやめられる初期コストの低さ。それからショッピングセンター内でフードコートが入っている南館が火災で倒壊しても、すぐに北館に移ってピザ屋の営業を再開できる、といった可用性です。

この冗長性を自力で確保しようとしたら大変です。ピザ屋さんを常に営業し続けておくために、いつ来るか分からない火災に備えて最初から予備の店舗も確保しておかなければならないとしたら相当なコストがかかります。オンプレミスのサーバなら、ただ自社サイトを作りたいだけなのに、品川と渋谷の2か所でデータセンターを借りて両方に1台ずつサーバを用意しておき、もし品川のデータセンターが火災で使えなくなても渋谷のデータセンターにあるサーバは生きているのでサイトは見られる、という体制にしておくよう

*18 Virtual Private Server の略。先ほど出てきた仮想サーバのことだと思ってください。

な大仰な話^{*19}です。構成次第でこれが可能になるクラウドはすごいですよね。

ここまでクラウドの良さを色々お話ししてきましたが、もちろんデメリットもあります。

もし何かトラブルがあつてフードコート全体がお休みになるときは、問答無用でピザ屋さんもお休みになってしまいます。つまり使っているクラウドで大規模障害が起きたら、一利用者である私たちにできることはなく復旧までひたすら待つしかない、ということです。AWSでも広範囲にわたる障害は定期的に起きています。たとえば2016年には豪雨による電源障害でサーバに接続できなくなる事象が発生^{*20}しました。こうした障害の際もAWSが発表してくれる内容がすべてですので、原因が分かるまで自分で徹底的に調べる、あるいは自力で何とかする、ということはできません。

またフードコートの通路やトイレ、駐車場といった共有スペースは他店舗（ドーナツ屋さんやラーメン屋さんなど）と共有していますので、フードコート内で他のお店が混んでくると、駐車場が満杯になってピザ屋さんに来たかったお客様が入れなかつたり、人波が自分の店の方まで押し寄せてきたりとマイナスな影響も受けます。つまり同じクラウド^{*21}を使っているウェブサイトにアクセスが集中すると、たとえば回線がひっ迫したりして自分のサイトまで繋がりにくくなる、というデメリットがあるということです。

1.4.4 AWSはAmazonがやっているクラウド

たくさんお話ししてきたので、一度おさらいをしましよう。

企業が「自社のウェブサイト作りたいなあ・・・だからサーバが必要だ！」と思ったとき、自分でサーバを買って自分で管理しなければいけないのがオンプレミスで、従量課金ですぐに使って性能や台数の増減も簡単にできるのがクラウドです。

そしてようやく最初の話に戻ると、AWSとはAmazon Web Services（アマゾン ウェブ サービス）の略で、欲しいものをぽちっとな！ すると翌日には届くあのAmazonがやっているクラウドなのです。

AWSがなんなのか、お分かりいただけましたでしょうか？

1.4.5 パブリッククラウドとプライベートクラウド

ところでパブリッククラウドやプライベートクラウド、という言葉は聞いたことがありますか？

^{*19} このように火事や自然災害などが起きたときにサービスが止まらないように備えておく体制をディザスタリカバリ (Disaster Recovery)、略してDRと言います。

^{*20} Amazonクラウドのシドニーリージョン、豪雨による電源障害でEC2などに一部障害。現在は復旧－Publickey <https://www.publickey1.jp/blog/16/amazonec2.html>

^{*21} 具体的には、同じホストサーバを使っている他のゲストサーバ上のサイト。あるいは同じインターネット回線を使っている他のサーバ上のサイト、ということです。

AWSのようなクラウドは、パブリッククラウドと呼ばれることもあります。みんなでホストサーバという資源（リソース）を共有して使うので、「公共の」という意味の「パブリック」が付いています。

クラウドが少しずつ使われるようになった頃に「クラウドって便利そうだけど、みんなで共有するのってちょっと抵抗あるな・・・」と思った人たちを安心させるため、「プライベートクラウド」という言葉が生まれました。このプライベートクラウドとはいったい何なのでしょう？

たとえばオンプレミスの環境で「高スペックな物理サーバを買ってホストサーバにして、その上でゲストサーバ（仮想サーバ）を立てられるようになった」とします。「ホストサーバのスペックが足りる限りという制限はあるものの、好きなときに好きなだけゲストサーバを立てたり、増強したりできるのでこれはもはやクラウド！ プライベートなクラウドだ！」と言いだした人がいました。また「クラウド事業者が提供しているホストサーバを1台まるまる占有する契約をしたぞ！ 自社で物理サーバを所有している訳ではないのでこれはクラウドなんだ。しかも他の人はこのホストサーバ上のゲストサーバを使えないから、プライベートなクラウドだ！」と言う人も現れました。定義は曖昧なのですが、このようにみんなで共有せず、自社だけで専有できるクラウドをプライベートクラウドと呼ぶようです。

こうしたプライベートクラウドだと「初期投資額が少ない」「サーバの性能や台数を後から好きなだけ増強できる」といった、クラウド本来のメリットが享受できないように思えます。

そもそもクラウドは英語で書くと「Cloud」（雲）です。物理的な実体や設置してある場所を意識することなく、インターネットという大きな雲の向こう側にあるサーバを好きなように利用できる環境を「クラウド」と呼んでいたはずなのに、果たしてこのようなプライベートクラウドはクラウドなのでしょうか？

このようにクラウドという言葉はとても曖昧です。結局「クラウド」という言葉の定義がはっきりしていないため、その人が言っている「クラウド」という言葉がなにを指しているのかは、よくよく聞いてみると分からず、ということです。^{*22}

1.4.6 AWS 以外のクラウド

ところでクラウドは AWS 以外にも Google の Google Cloud Platform^{*23}、Microsoft の Azure（アジュール）^{*24}、その他にも国内クラウドとしてさくらインターネットがやつ

^{*22} 実際、オンプレミス環境にある仮想サーバをクラウドサーバと呼んでいるケースも多々あります。

^{*23} <https://cloud.google.com/>

^{*24} <https://azure.microsoft.com/ja-jp/>

ているさくらのクラウド^{*25}、お名前.comと同じGMOグループのGMOクラウド^{*26}などたくさんあります。

その中でもなぜ「AWSがいい」と言われているのでしょうか？

理由は使う人によってそれぞれだと思いますが、私なりの「なぜAWSなのか？」を考えてみました。

2018年時点、クラウド市場ではAWSがシェア33%でトップを独走中^{*27}です。そのため他のクラウドと比べると、AWSなら使ったことがあり対応可能なエンジニアも多いし、何か困ったときに調べて出てくる情報も多い、というのが、私がAWSを選ぶいちばんの理由です。それ以外だと、利益が出た分だけどんどん投資されてサービスが改良されていくため、細かな使い勝手がどんどん良くなっていく^{*28}、というところもポイントです。

クラウドを選ぶ理由、の中でもAWSを選ぶ理由というのは、普遍的な何かがあるわけではなく、本来は使う人やその上で動かすサービスによって異なるはずです。あなたが動かしたいサービスによっては、AWSではなく他のVPSやオンプレミスの方がいいケースだってもちろんあるはずです。これから使ってみて、あなた自身がAWSの良いところを発見できたらいいですね。

*²⁵ <https://cloud.sakura.ad.jp/>

*²⁶ <https://www.gmocloud.com/>

*²⁷ 2018年第1四半期、クラウドインフラ市場でAWSのシェアは揺るがず33%前後、マイクロソフト、Googleが追撃、IBMは苦戦中。Synergy Research - Publickey https://www.publickey1.jp/blog/18/20181aws33googleibmsynergy_research.html

*²⁸ 画面や機能もどんどん変わっていくので、この後出てくる設定画面も、皆さんのが手を動かしてやってみると頃には本著のキャプチャとは違うものになっているかも知れません。AWSのいいところでもあり、本やマニュアルを作つて説明する側にとってはつらいところでもあります。

第 2 章

AWS を使い始めたら最初にやること

この章では AWS の管理画面にサインインして、AWS を使い始めたら最初にやるべき設定を実践していきます。初期設定とかいいから早くサーバ立てたい！ という気持ちだと思いますが、あなたのお財布を守るために最初にしっかりセキュリティを強化しておきましょう。

2.1 AWS 無料利用枠を使おう

AWS を初めて使用する場合、AWS アカウントを作成してから 1 年間は利用料が無料となります。但し、無料利用枠の範囲は決まっており、何をどれだけ使っても無料という訳ではありません。何もかも全部無料だと思ってサーバをバカスカ立てると、あとでクレジットカードにしっかり請求が来ますので注意してください。

どのサービスをどれくらい無料で使えるのか？は「AWS 無料利用枠の詳細 (<https://aws.amazon.com/jp/free/>)」(図 2.1) に「Amazon EC2 は t2.micro インスタンスが月に 750 時間無料」、「Amazon EBS は 30GB 無料」のように細かく書かれていますので、そちらを参照してください。^{*1}



▲図 2.1 AWS 無料利用枠の詳細

なお本著で使用する AWS のサービスは、基本的にこの無料利用枠の範囲内に収まるようになっています。但し、Route53 というネームサーバのサービスなど、一部は無料利用枠の対象外となるため毎月 50 セント～数ドル程度かかりますのでその点はご留意ください。

うっかり多額の請求が来ても筆者が代わりに支払うことはできません^{*2}ので、そうならないよう後ほど「利用金額が〇円を超えたたらメールで知らせる」という請求アラートの設定をしっかりしておきましょう。

^{*1} EC2 ってなに？ EBS ってなに？ は後述します。

^{*2} できませんできません、人間にはこんなこと絶対にできません。

2.2 AWS のアカウント作成

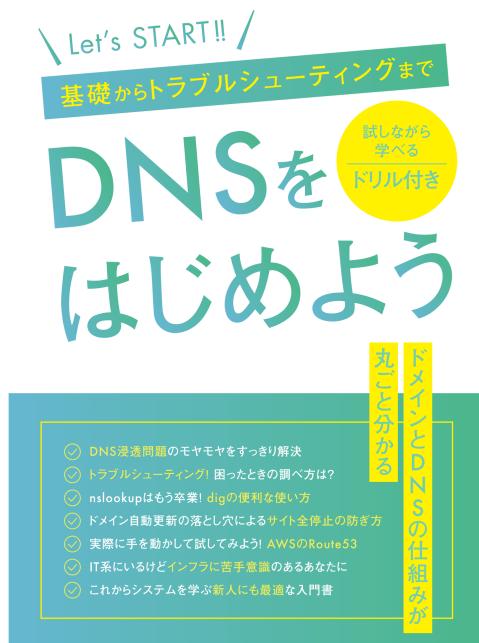
AWS を使うには、先ず AWS アカウントを作成する必要があります。

AWS アカウントの作成は「DNS をはじめよう」(図 2.2)*3 の「第 3 章 AWS のネームサーバ (Route53) を使ってみよう」で済ませていますので、本著でもその AWS アカウントを引き続き使用していきます。

まだ AWS アカウントを持っていない！ 作っていない！ という人は、先に「DNS をはじめよう」で、

1. ドメインを買う
2. AWS のアカウントを作る
3. ネームサーバとして AWS の Route53 を使う

という 3 つのステップを踏んでから、この先へ進むようにしてください。



▲図 2.2 「DNS をはじめよう」(1,000 円) は BOOTH で好評発売中

*3 <https://mochikoastech.booth.pm/>

【コラム】「DNS をはじめよう」はどこで買える？

「AWS をはじめよう」の前作である「DNS をはじめよう」（通称 DNS 本）は書籍版、PDF ダウンロード版とともに BOOTH^aで購入できます。

BOOTH はピクシブ株式会社^bが運営している同人誌の通販及びダウンロード販売サイトで、書籍版を購入すると 1~2 営業日以内に BOOTH 倉庫からネコポスで本が送られてきます。PDF 版なら購入後すぐにダウンロードして読むことができます。技術書典で頒布されている同人誌の多くは BOOTH でも購入できますので、気になる方は「技術書典」のタグで検索^cしてみることをお勧めします。

本といえば Amazon なので「Amazon で売ってくれないかな？」と思われる方も多いと思うのですが、そもそも Amazon では同人誌が販売できないため、Amazon で売るためには先ずは ISBN コード（商業誌の裏表紙にあるバーコードとその下の番号）を頑張って取らねばなりません。そこに労力を割くよりは、いい本を書く方向で頑張ろうと思いますのでどうぞご理解ください。

ちなみに「DNS をはじめよう」は、ただの DNS 好きである筆者が DNS へのあふれんばかりの愛を早口で詰め込んだ本ですが、技術書典 4 当日に 750 冊、その後もダウンロード販売で売れ続けて累計 1300 冊以上（2018 年 8 月現在）という驚きの頒布数となりました。手にとって、買って、読んでくださった皆さん、ありがとうございます。

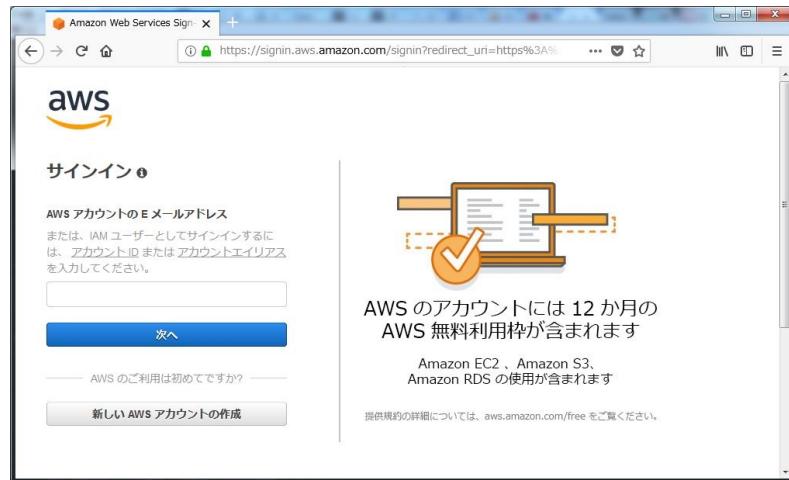
^a <https://mochikoastech.booth.pm/>

^b イラストを投稿できる SNS、pixiv でお馴染み。<https://www.pixiv.net/>

^c <https://booth.pm/ja/search/技術書典>

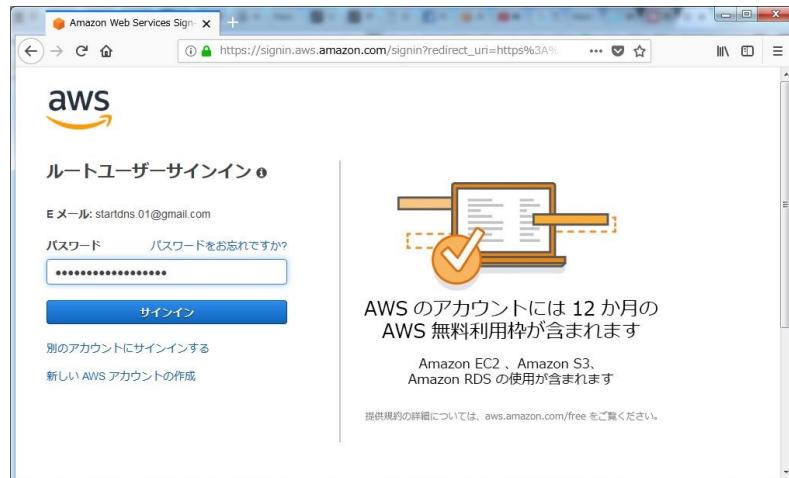
2.3 マネジメントコンソールにサインイン

それでは早速、AWS のサインイン画面 (<https://console.aws.amazon.com/>) を開いて（図 2.3）、マネジメントコンソールにサインインしましょう。サインインという言葉には馴染みがないかも知れませんが、「ログイン」と同じ意味です。



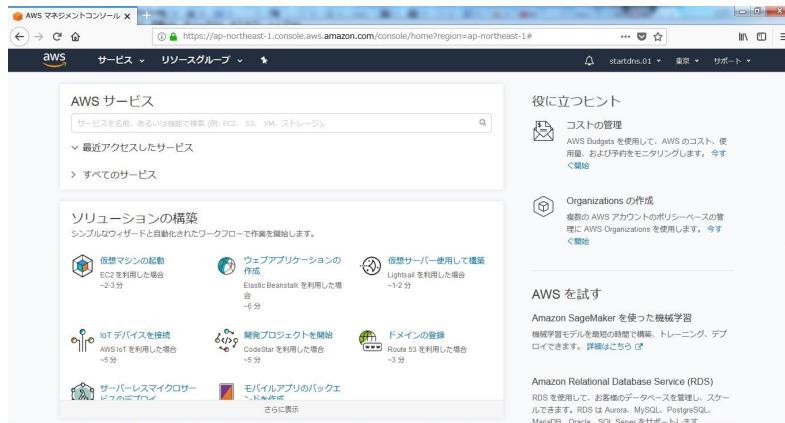
▲図 2.3 マネジメントコンソールのサインイン画面

先ずは AWS アカウントの E メールアドレスを入力して「次へ」、続いてルートユーザー サインインの画面（図 2.4）でパスワードを入力して「サインイン」ボタンを押します。



▲図 2.4 E メールアドレスを入力後、パスワードを入力してサインイン

無事にサインインできたら、マネジメントコンソール（図 2.5）が表示されます。皆さんもサインインできましたか？



▲図 2.5 マネジメントコンソール (AWS の管理画面)

このマネジメントコンソールが、AWS の管理画面となります。これからサーバを立てたりする作業は、すべてこの画面で行っていきます。

2.3.1 【ドリル】AWS の管理画面はなんて名前？

問題

AWS の管理画面はなんと呼ばれているでしょう？

- A. コントロールパネル
- B. マネジメントコンソール
- C. クラウドコンソール

答え _____

解答

正解は B のマネジメントコンソールです。マネジメントコンソールにはサインイン画面 (<https://console.aws.amazon.com/>) からサインインします。AWS を使うときは、このマネジメントコンソールで色々な操作をしますので、名前を覚えておいてください。

2.4 IAM でユーザの権限管理

2.4.1 ルートユーザーの普段使いはやめよう

さて、皆さんがあなたのアカウントに使ったのは「ルートユーザー」と呼ばれるユーザです。実はルートユーザーは全権を持っていてなんでもできるユーザなので、普段からこのユーザを使って色々な操作をするのはお勧めしません。

「ルートユーザーってなんでもできるユーザなんですよ？ 大は小を兼ねるっていうし、便利なんだからそれ使えばいいじゃない」と思われるかも知れませんが、最寄のスーパーまで晩御飯の買い物に行くだけなのに、1,000万円と利用上限額なしのクレジットカードをアタッシュケースに詰めて持っていく人は居ないですよね？ 子供の財布なら1,000円くらい、自分の財布なら20,000円くらい、のように使う人によって使える金額を制御しておくことで、財布を落としたり盗まれたりしたときのダメージを少なくしておくのは、誰しも無意識にやっているセキュリティ対策だと思います。

AWSのユーザにはクレジットカードを紐付けているのですから、お財布もルートユーザーも等しく扱いには気をつけなければいけません。たとえば悪い人があなたのルートユーザーのEメールアドレスとパスワードを盗んで、こっそりマネジメントコンソールにサインインしたとします。ルートユーザーならなんでもできるので、景気良くいちばん高いサーバ^{*4}を100台立てた^{*5}としましょう。その場合、1日で180万円かかるので、1ヶ月後には5,400万円の請求^{*6}があなたのところへやってきます。（図2.6）

^{*4} EC2のd2.8xlargeというサーバは、東京リージョンだと1時間あたり6.752ドル。日本円にすると1日でおおよそ18,000円です。もちろん大量に立てられないように台数制限はありますが、ルートユーザーならその制限を緩和するリクエストを出すことだって可能です。

^{*5} そんなにいっぱいサーバを立ててどうするの？と思いませんよね。なんと悪い人たちは他人のアカウントで高スペックのサーバを大量に立てて、ビットコインを生み出すためのマイニング（採掘）をするのです。

^{*6} 不正利用による請求であっても本来は契約者に支払い義務がありますが、ネット上の体験記を見ると支払いを免除あるいは返金してもらえることが多いようです。もし心当たりのない多額の請求が来たら落ち着いてサポートに問い合わせてみましょう。



▲図 2.6 「AWS 不正利用」で検索すると不正利用による請求で青ざめた体験記がたくさん

このようにルートユーザーだとなんでもできてしまうため、権限が必要ないときにはルートユーザーを使うのは大変危険です。繰り返しになりますが、1,000万円とクレジットカードが詰まったアタッシュケースを持って、うつきうきでスーパー・マーケットに行くのは危ないのでやめましょう。

2.4.2 IAM ユーザを作ろう

という訳でルートユーザではなく、必要な作業ができる権限だけを持った自分用の「IAM ユーザ」というユーザを作つて普段はそちらを使いましょう。

IAM ってなに？

IAM は Identity and Access Management の略で、AWS の利用を安全に管理するためのサービスです。

ある程度の規模の会社であれば、1つの鍵をみんなで使いまわしたりせずひとりひとりに ID カードが付与されていますよね。オフィスを安全に利用するため、ID カードを使うことで社内の人間しかオフィスフロアへ入れないようになっていたり、特定のプロジェクトチームにはそこのプロジェクトメンバーしか入れないようになっていたりします。

IAM も同じで AWS を利用する人を限定したり、サービスによって使える人を絞ったりすることができます。

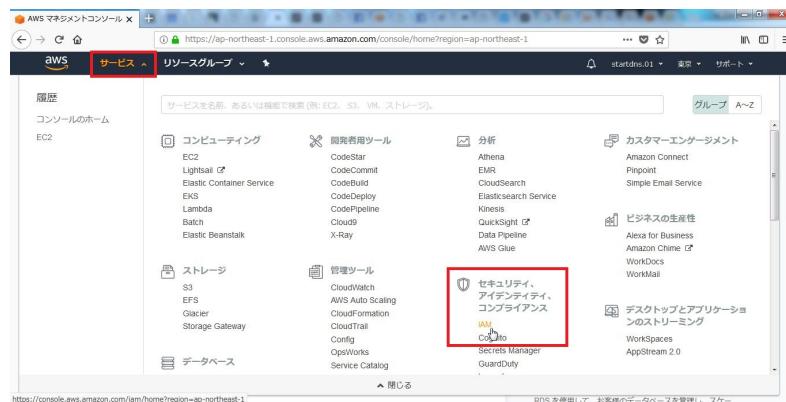
IAM ユーザーは 1 人に 1 つ

本著では 1 人だけで作業をする想定なので、IAM ユーザーも 1 人しか作りません。ですがもし業務などで複数名でマネジメントコンソールにサインインする場合は、IAM ユーザーは必ず 1 人につき 1 つずつ作成してください。まったく同じ作業をするから開発チーム内の A さんと B さんは同じ IAM ユーザーを共有すればいいのでは？ と思われる場合でも、必ず A さん B さんそれぞれに別々の IAM ユーザーを用意することをお勧めします。

なぜならば AWS では「いつ・どの IAM ユーザーが・なにをしたのか」をすべて記録していて、後から調べることができるのですが、仮に A さんと B さんが 1 つの IAM ユーザーを共用していた場合、何か重大なトラブルが起きたとき^{*7}に「結局、誰がやらかしたのか？」を人単位で追いかけることができなくなってしまうからです。

IAM ダッシュボード

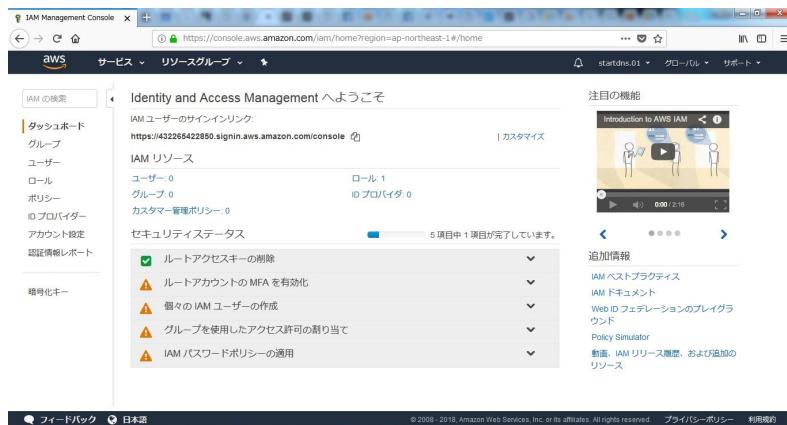
それではマネジメントコンソールの左上にある「サービス」から、「セキュリティ、アイデンティティ、コンプライアンス」の下にある「IAM」（図 2.7）をクリックしてください。



▲図 2.7 サービス>セキュリティ、アイデンティティ、コンプライアンス>IAM

「IAM」をクリックすると、IAM のダッシュボード（図 2.8）が表示されます。

^{*7} 想像もしたくないですが、たとえば誰かがすべてのサーバをバックアップ含めてすべてきれいに削除してしまった、とか。IAM ユーザーを共用していると、使っていた人全員に疑いがかかつてしまします。



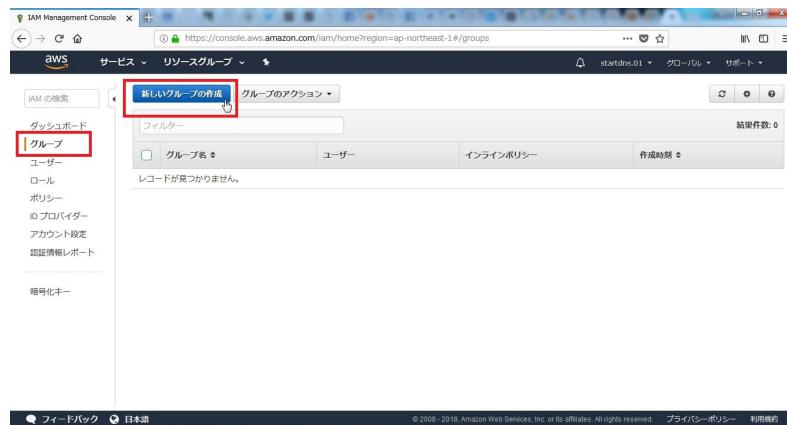
▲図 2.8 IAM ダッシュボード

IAM のグループを作成

では先ず IAM ダッシュボードの左メニューから「グループ」を選んでください。

IAM ではグループを作成して、そのグループに対して権限を設定し、個々の IAM ユーザはグループに所属させることでアクセス権限を管理できます。たとえば前述のような「開発チームの A さんと B さんにはまったく同じ権限を付与したい」という場合に、先に developers というグループを作つて、developers グループに権限を付与しておけば、A さん B さんの IAM ユーザは developers グループに所属させるだけで必要な権限を渡すことができます。

今はまだ IAM にグループが 1 つもないため、先ずはグループを作りましょう。左上の「新しいグループの作成」をクリック（図 2.9）します。



▲図 2.9 左メニューの「グループ」>「新しいグループの作成」をクリック

ここから 3 つの手順で新しいグループを作成していきます。

まずは手順 1 の「グループ名」です。本著では IAM のグループは「start-aws-group」にします。グループ名の欄に「start-aws-group」と入力して、右下の「次のステップ」をクリック（図 2.10）してください。



▲図 2.10 グループ名に「start-aws-group」と入力して「次のステップ」をクリック

続いて手順 2 の「ポリシーのアタッチ」でグループに対してポリシーを紐付けます。なんだかものすごくたくさん並んでいますが、それぞれ「どのサービスでどんな操作を許可する」というポリシー（方針）ですので、そこから必要なポリシーを選択してグループにアタッチ（紐付け）していきます。

たくさんあるのでここでは 2 つだけ紹介します。先ほどのルートユーザーと同様に何でもできる 1 番権限の強いポリシーが「AdministratorAccess」です。そして「Adminis-

tratorAccess」から IAM に関する権限だけを引いたのが「PowerUserAccess」という、2番目に権限の強いポリシーです。

本来は必要最小限の権限だけを付与するべきですが、今回は細かな設定はせずにこの1番強力な「AdministratorAccess」というポリシーを「start-aws-group」にアタッチします。^{*8} フィルターに「AdministratorAccess」と入力して、下に表示された「AdministratorAccess」にチェックを入れたら、右下の「次のステップ」をクリック（図2.11）してください。

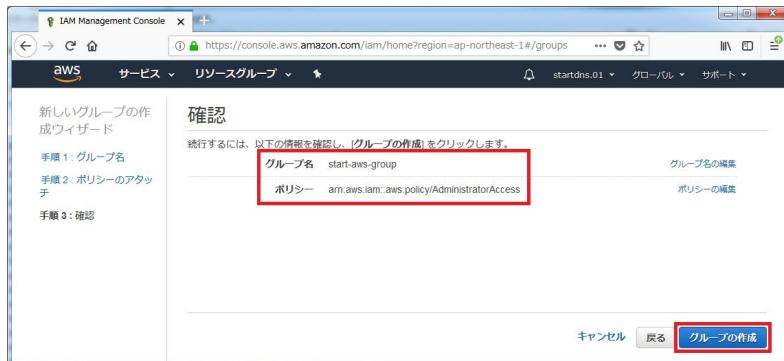


▲図2.11 「AdministratorAccess」にチェックを入れて「次のステップ」をクリック

最後に手順3の「確認」です。グループ名とポリシーを確認したら、右下の「グループの作成」をクリック（図2.12）します。

- グループ名 : start-aws-group
- ポリシー : arn:aws:iam::aws:policy/AdministratorAccess

^{*8} え、大金の詰まったアタッシュケース持って買い物に行っちゃうの？！と思ったあなたは正しいです。ですがポリシーを細かにアタッチしていく作業をすべて解説するのは紙面の都合上難しいため、ここでは「普段はルートユーザーではなく IAM ユーザーを使うべき」「本来は IAM ユーザーには必要最小限の権限だけを付与すべき」ということだけ覚えておいて先へ進みましょう。



▲図 2.12 グループ名とポリシーを確認して右下の「グループの作成」をクリック

IAM のグループ一覧に、今作ったばかりの「start-aws-group」というグループが表示（図 2.13）されたらグループの作成は完了です。



▲図 2.13 「start-aws-group」というグループが作成できた

IAM のユーザを作成

IAM のグループが作成できたので、続いて IAM ユーザーを作成しましょう。左メニューから「ユーザー」を選ぶと、ユーザーの一覧画面（図 2.14）が表示されます。まだ IAM ユーザーが 1 つも存在しないため、「IAM ユーザーが存在しません。」と表示されています。それでは上部の「ユーザーを追加」をクリックしてください。



▲図 2.14 左メニューの「ユーザー」>「ユーザーを追加」をクリック

ここから 3 つのステップで IAM ユーザーを追加していきます。

まずはステップ 1 (図 2.15) です。IAM のユーザー名を入力してください。本著では IAM のユーザ名は「start-aws-user」にします。IAM のユーザー名はこの後もサインイン時に使用しますので、もし別のユーザー名にした場合は、しっかりメモしておいてください。

続いてこの IAM ユーザーで AWS にアクセスする方法を選択します。AWS でたとえば「サーバを立てる」「サーバを削除する」などの操作をするには、

1. プログラムから AWS の API を叩いて操作する方法
2. ブラウザでマネジメントコンソールを開いて画面上で操作する方法

の 2 つがあります。本著ではマネジメントコンソールからしか操作しないため、「アクセスの種類」は「AWS マネジメントコンソールへのアクセス」にのみチェックを入れてください。

ユーザー名を入力して、アクセスの種類を選択したら「次のステップ: アクセス権限」をクリックします。



▲図 2.15 ユーザー名を start-aws-user にして、AWS マネジメントコンソールへのアクセスにチェック

続いてステップ 2（図 2.16）です。先に作っておいた「start-dns-group」というグループに、ユーザーを追加します。「start-dns-group」にチェックを入れたら、「次のステップ: 確認」をクリックしてください。



▲図 2.16 「start-aws-group」にチェックを入れて、「次のステップ: 確認」をクリック

最後にステップ 3 です。次の 4 つを確認して、問題なければ「ユーザーの作成」をクリック（図 2.17）してください。

1. ユーザー名が「start-aws-user」であること

2. AWS アクセスの種類が「AWS マネジメントコンソールへのアクセス - パスワードを使用」であること
3. グループが「start-aws-group」であること
4. 管理ポリシーが「IAMUserChangePassword」であること



▲図 2.17 確認して問題なければ「ユーザーの作成」をクリック

「成功」と表示されたら IAM ユーザーの作成は完了です。(図 2.18)



▲図 2.18 「成功」と表示されたら IAM ユーザーの作成完了

この画面で表示される URL の数字 (図 2.19) とパスワード (図 2.20) は、この後サイ

シainするときに使用しますので必ずメモ（表 2.1）しておいてください。



▲図 2.19 URL の数字をメモしておこう

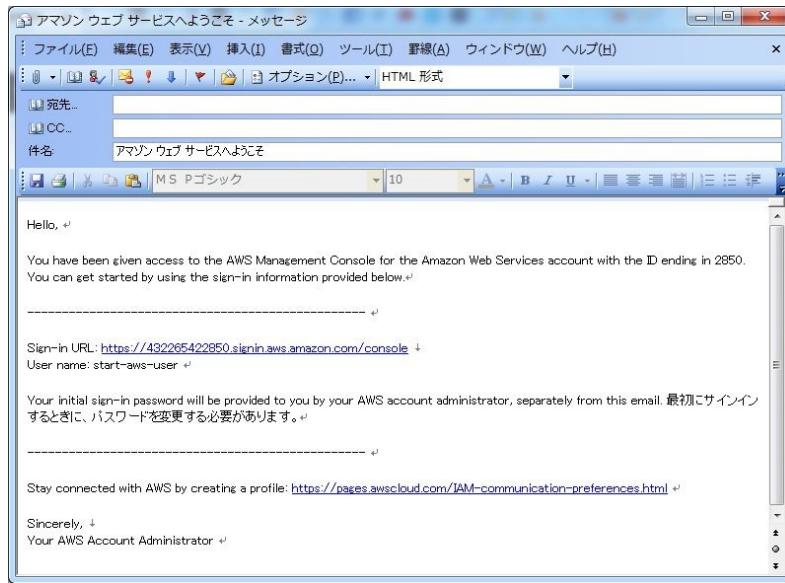


▲図 2.20 パスワードの「表示」をクリックして、パスワードもメモしておこう

▼表 2.1 IAM ユーザの情報

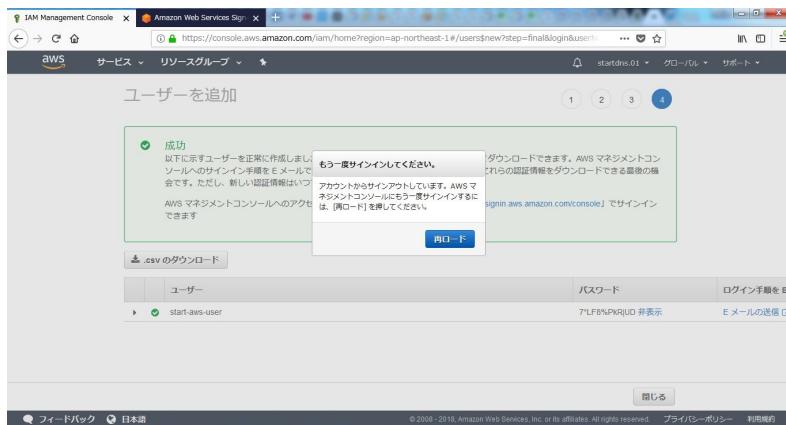
IAM ユーザー情報	例	自分の IAM ユーザー情報
AWS アカウント (URL の数字)	432265422850	
ユーザー名	start-aws-user	
パスワード	7*LF8%Pkr UD	

メモができたら、続けて右下の「E メールの送信」をクリック（図 2.21）します。サインイン URL や IAM のユーザー名は忘れやすいので、メールでも自分自身宛てに送っておくことをお勧めします。



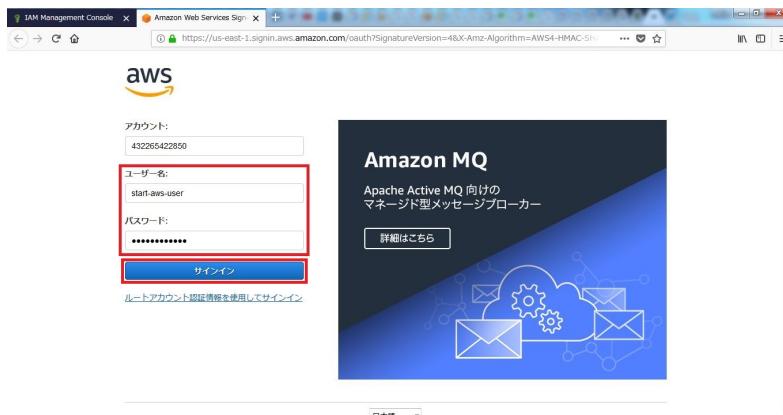
▲図 2.21 サインイン URL や IAM ユーザー名をメールで自分自身宛てに送っておこう

しっかりメモをしてメールも送ったら、「成功」と表示された画面で「https://*****.signin.aws.amazon.com/console」と書いてある URL をクリックします。するとブラウザの別タブで IAM ユーザーのサインイン画面が開いて、自動的に元のタブのルートユーザーはサインアウト（図 2.22）させられます。サインアウトしてしまったタブは閉じてしまって構いません。



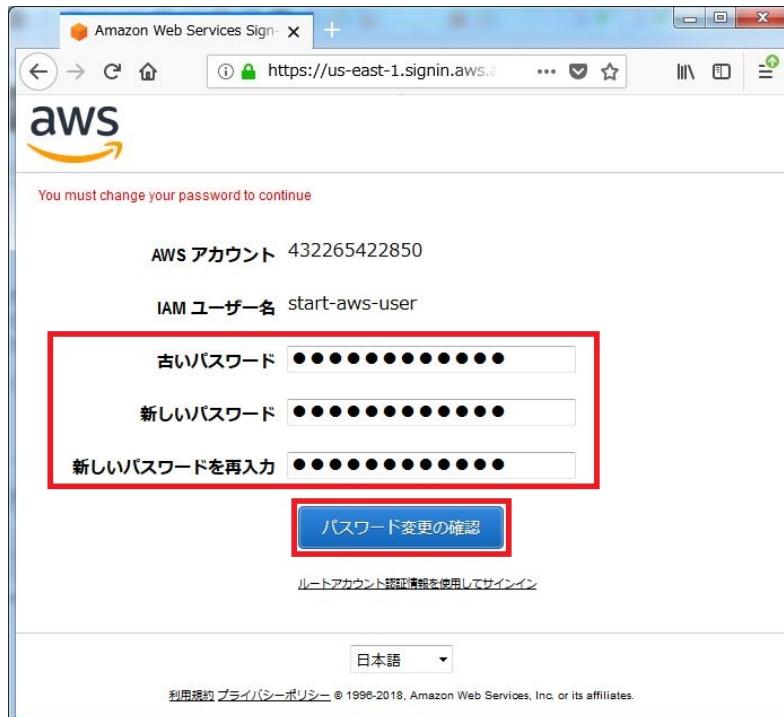
▲図 2.22 ルートユーザーでサインインしていた画面は自動的にサインアウトされる

別タブで開いた IAM ユーザーのサインイン画面（図 2.23）で、先ほどメモしたユーザー名とパスワードを入力して「サインイン」をクリックします。



▲図 2.23 ユーザー名とパスワードを入力して「サインイン」をクリック

サインインすると、新しいパスワードの設定画面（図 2.24）が表示されます。「古いパスワード」に先ほどメモしたパスワードを、「新しいパスワード」には今新たに自分で考えたパスワードを入力してください。すべて入力したら「パスワード変更の確認」をクリックします。



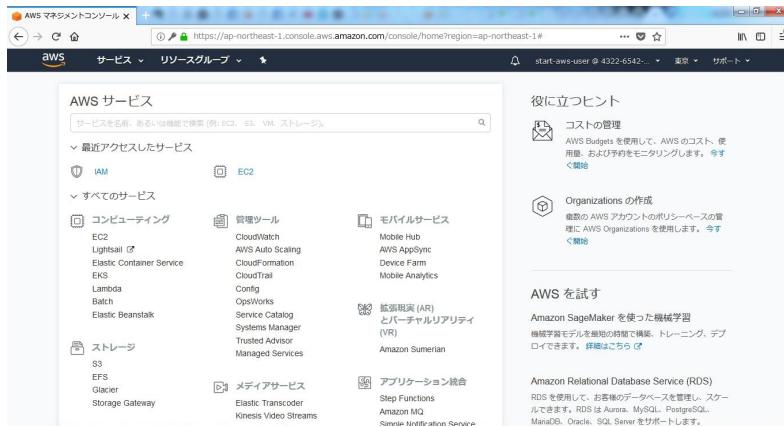
▲図 2.24 新しいパスワードを入力して「パスワード変更の確認」をクリック

ここで再び AWS アカウント、ユーザー名、新しいパスワードをメモ（表 2.2）しておきましょう。

▼表 2.2 IAM ユーザー情報

IAM ユーザー情報	例	自分の IAM ユーザー情報
AWS アカウント (URL の数字)	432265422850	
ユーザー名	start-aws-user	
新しいパスワード	自作のパスワード	

IAM ユーザーで無事にサインインできたら、マネジメントコンソール（図 2.25）が表示されます。皆さんもサインインできましたか？



▲図 2.25 IAM ユーザーでサインインできた！

右上の IAM ユーザー名をクリック（図 2.26）すると、IAM ユーザー名と AWS アカウントが表示されます。ここでルートユーザーではなく IAM ユーザーでサインインしていることが確認できます。



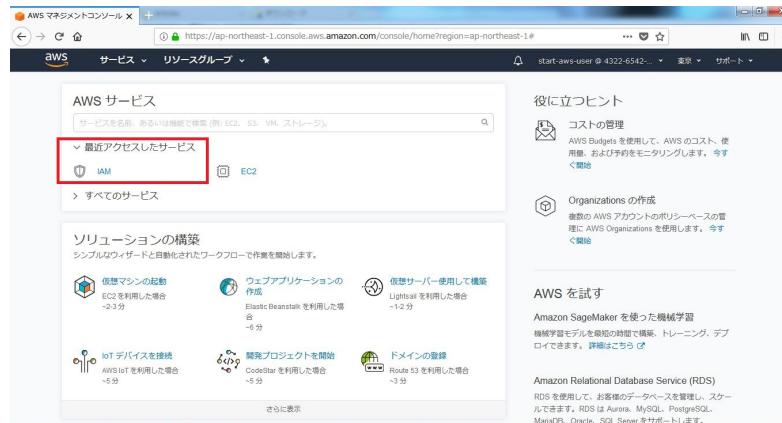
▲図 2.26 IAM ユーザーでサインインしていることを確認

これでルートユーザーではなく、IAM ユーザーでマネジメントコンソールにサインインできるようになりました。

2.4.3 MFA (多要素認証) で不正利用から IAM ユーザーを守る

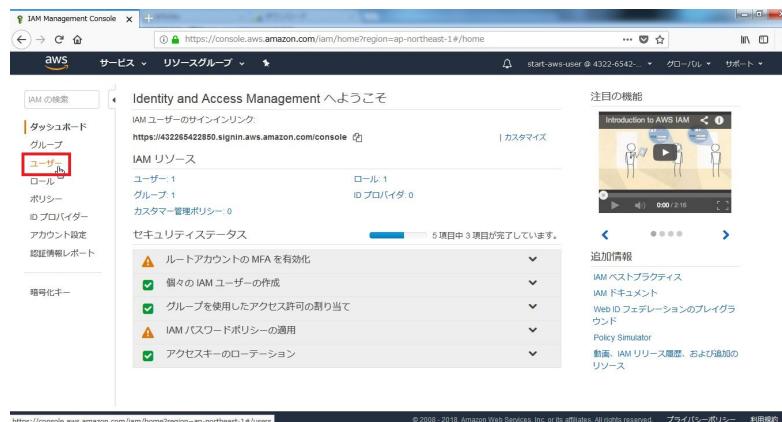
無事に IAM ユーザーでサインインできたら、再び左上にある「サービス」から「IAM」をクリックしてください。もしくは「最近アクセスしたサービス」から「IAM」をクリック

ク（図2.27）でも構いません。



▲図2.27 「最近アクセスしたサービス」から「IAM」をクリック

IAM のダッシュボードが表示されたら、左メニューの「ユーザー」をクリック（図2.28）します。



▲図2.28 左メニューの「ユーザー」をクリック

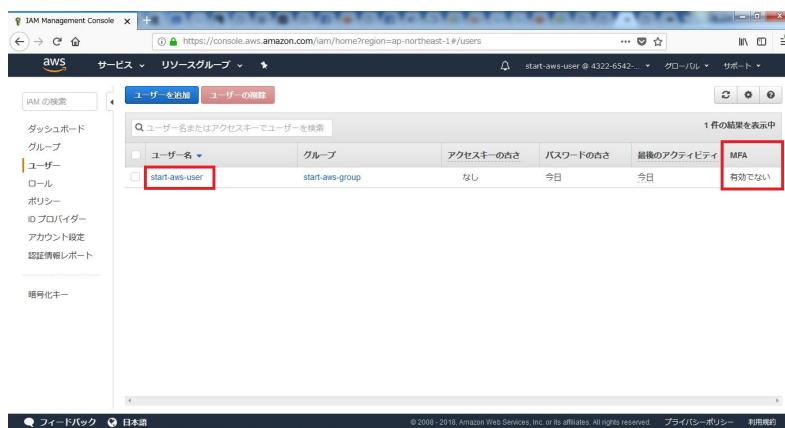
現状は、

1. AWS アカウント（12桁の数字）
2. ユーザー名

3. パスワード

の 3 つがあれば、IAM ユーザーでサインインできてしまいます。ですが MFA（多要素認証）^{*9}を有効にすると「AWS アカウントとユーザー名とパスワード」に加えて、ユーザーが持っているスマホの認証アプリなど別の要素を使って本人か確認することになるため、より安全にアカウントを管理できます。

今はまだ MFA が有効になっていない（図 2.29）ため、有効にしてみましょう。ユーザー名の「start-aws-user」をクリックします。



▲図 2.29 ユーザー名の「start-aws-user」をクリック

「認証情報」タブの「MFA デバイスの割り当て いいえ」の横にあるエンビツマークをクリック（図 2.30）します。

^{*9} AWS Multi-Factor Authentication の略。

The screenshot shows the IAM Management Console with the URL <https://console.aws.amazon.com/iam/home?region=ap-northeast-1#/users/start-aws-user?section=sec>. The left sidebar shows navigation options like IAM Home, Dashboard, Groups, Users, Roles, Policies, and AWS Providers. The main area is titled '概要' (Overview) for the user 'start-aws-user'. It displays the ARN as 'arn:aws:iam:432265422850:user/start-aws-user', a status of 'パス /' (Path /), and a creation date of '2018-08-13 23:47 UTC+0900'. Below this, there are tabs for 'アクセス権限' (Access Permissions), 'グループ (1)' (Groups), '認証情報' (Authentication Information) (which is selected and highlighted with a red box), and 'アクセスアドバイザー' (Access Advisor). Under '認証情報', it shows 'コンソールのパスワード' (Console Password) as '有效' (Enabled) and 'パスワードの管理' (Password Management). It also lists a 'コンソールログインのリンク' (Console Login Link) at <https://432265422850.sigin.aws.amazon.com/console> and a '削除のログイン' (Delete Login) date of '2018-08-14 21:16 UTC+0900'. A red box highlights the 'MFA デバイスの割り当て' (Assign MFA Device) section, which contains two options: 'いいえ' (No) and 'はい' (Yes). The 'No' option is selected. At the bottom, there's a note about using access keys for AWS service API calls and a note about shared secret keys.

▲図 2.30 「MFA デバイスの割り当て いいえ」の横にあるエンピツマークをクリック

多要素認証をするときは、キーホルダーやカードの形をした専用の MFA デバイス（図 2.31）^{*10}を買って使うか、もしくは「Google 認証システム（Google Authenticator）」というスマホの認証アプリを擬似的な MFA デバイスとして使用します。



▲図 2.31 キーホルダータイプの MFA デバイス

言葉で説明しても分かりにくいと思うので実際にやってみましょう。有効にする MFA デバイスタイプは「仮想 MFA デバイス」を選択（図 2.32）して「次のステップ」をクリックしてください。

^{*10} <https://www.amazon.co.jp/dp/B019THYZGE>



▲図 2.32 「仮想 MFA デバイス」を選択して「次のステップ」をクリック

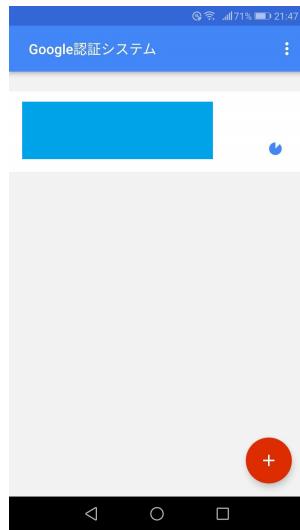
お手元のスマホに「Google 認証システム（Google Authenticator）」をインストール^{*11}したら「次のステップ」をクリック（図 2.33）します。



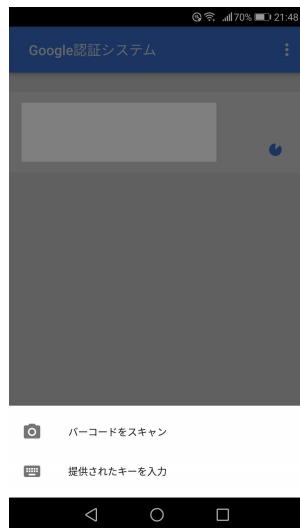
▲図 2.33 スマホに認証アプリをインストールしたら「次のステップ」をクリック

続いてスマホで「Google 認証システム」のアプリを起動したら右下の赤い+ボタンをタッチ（図 2.34）して、「バーコードをスキャン」を選択（図 2.35）します。

^{*11} Android なら <https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2>、iPhone なら <https://itunes.apple.com/jp/app/google-authenticator/id388497605> からインストールできます。Android の場合は元々入っているかも知れません。



▲図 2.34 「Google 認証システム」のアプリを起動したら右下の赤い+ボタンをタッチ



▲図 2.35 「バーコードをスキャン」を選択

「Google 認証システム」のアプリで「赤線内にバーコードを配置してください」(図 2.36) と表示されたら、マネジメントコンソールに表示されたバーコード(図 2.37) がス

マホの赤枠内に収まるようにします。

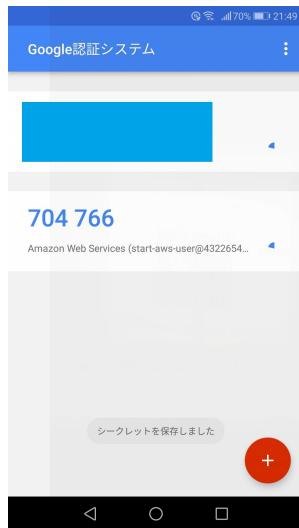


▲図 2.36 「赤線内にバーコードを配置してください」と表示される



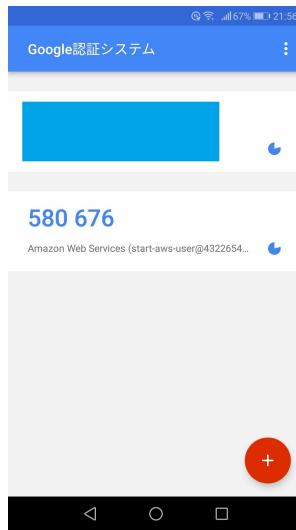
▲図 2.37 マネジメントコンソールに表示されたバーコードをスマホで撮る

すると「Google 認証システム」のアプリで「start-aws-user」用の認証コード（6桁の数字）が表示（図 2.38）されるようになります。



▲図 2.38 「start-aws-user」用の認証コード（6桁の数字）が表示されるようになる

この認証コードは 30 秒ごとに次々と切り替わっていきます（図 2.39）。表示されている認証コードをマネジメントコンソールの「認証コード 1」に入力（図 2.40）したら切り替わるのを待って、次の認証コードを「認証コード 2」に入力します。どちらも入力できたら「仮想 MFA の有効化」をクリックしましょう。

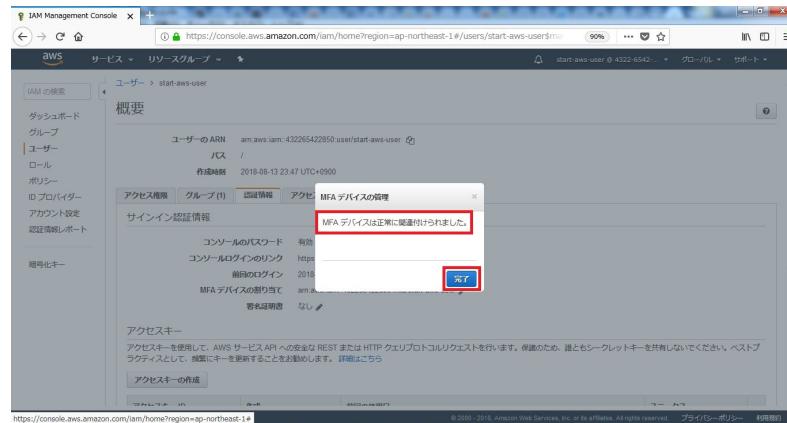


▲図 2.39 認証コードは 30 秒ごとに次々と切り替わる



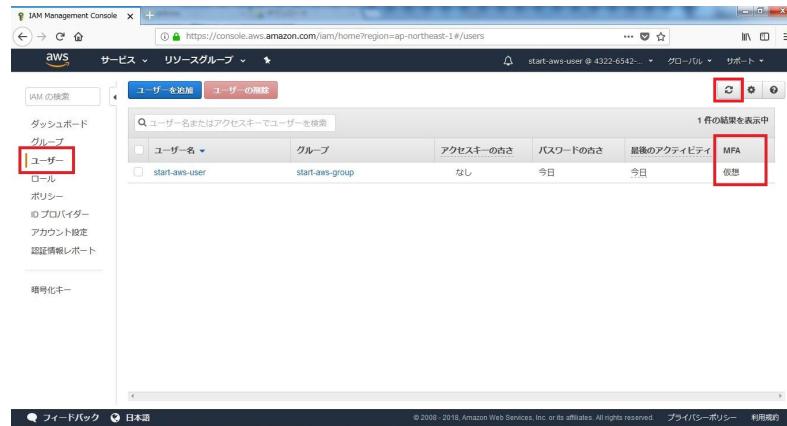
▲図 2.40 「認証コード 1」と「認証コード 2」を入力して「仮想 MFA の有効化」をクリック

「MFA デバイスは正常に関連付けられました。」と表示（図 2.41）されたら「完了」をクリックしてください。



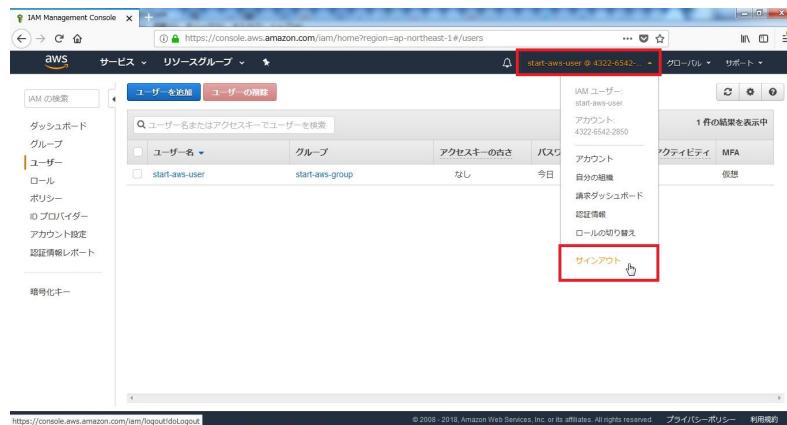
▲図 2.41 「MFA デバイスは正常に関連付けされました。」と表示されたら「完了」をクリック

再び左メニューの「ユーザー」をクリック（図 2.42）してから、右上の更新マークをクリックします。先ほどは「有効でない」になっていた MFA が「仮想」に変わっていたら MFA の設定は完了です。



▲図 2.42 MFA が「有効でない」から「仮想」に変わっていたら MFA の設定完了

それでは MFA を使ったサインインを試してみましょう。右上の IAM ユーザー名から「サインアウト」をクリック（図 2.43）します。



▲図 2.43 「サインアウト」をクリック

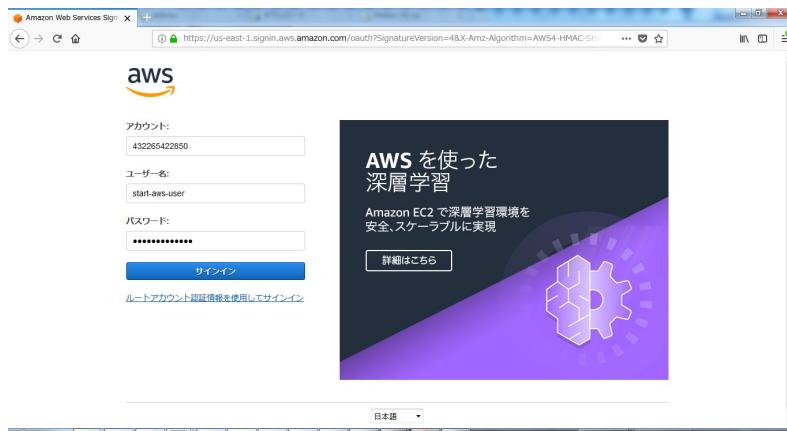
右上の「コンソールへログイン」をクリック（図 2.44）します。



▲図 2.44 「コンソールへログイン」をクリック

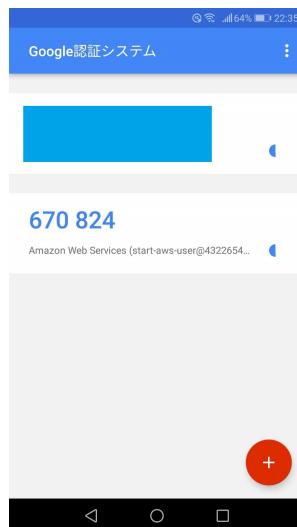
1. アカウント（12桁の数字）
2. ユーザー名
3. パスワード

の3つを入力したら「サインイン」をクリックします。

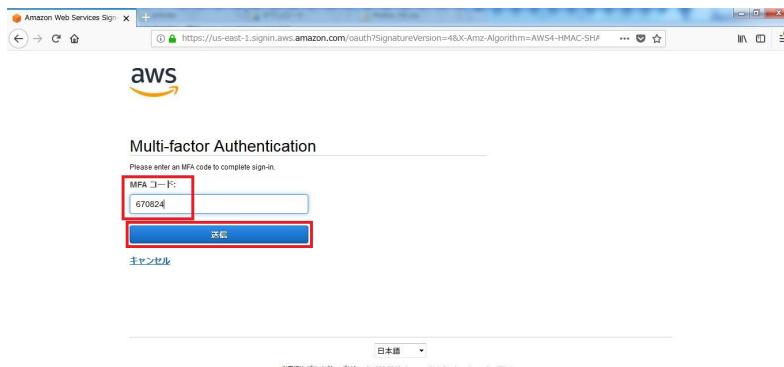


▲図 2.45 「サインイン」をクリック

今までではこれだけでサインインできていましたが、MFA（多要素認証）が有効になったことで、さらに認証コードも確認されるようになりました。スマホで「Google 認証システム」のアプリを起動（図 2.46）して、表示されている認証コードを「MFA コード」（図 2.47）のところへ入力したら「送信」をクリックしてください。



▲図 2.46 スマホで「Google 認証システム」を起動

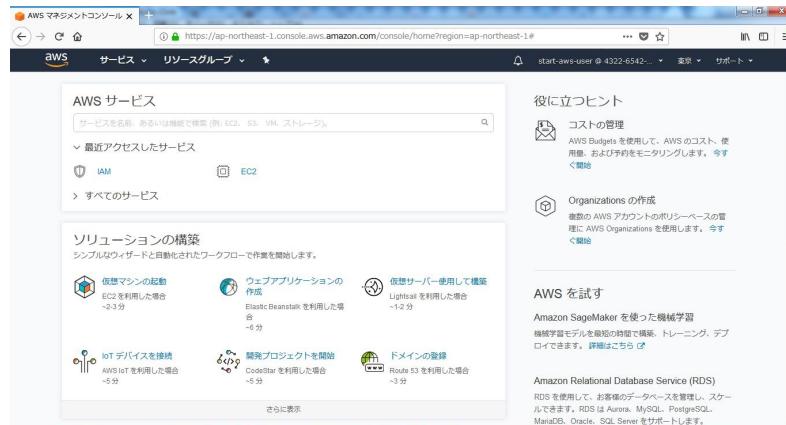


▲図 2.47 認証コードを「MFA コード」に入力して「送信」をクリック

マネジメントコンソールが表示されたら MFA を用いたサインインは成功です！今後、サインインするときには必ず「Google 認証システム」のアプリが必要になるため、もし IAM ユーザのパスワードが盗まれてしまっても、それだけではサインインできないので安心ですね。^{*12}

機種変更前に MFA の削除を忘れてしまい、MFA が有効な状態で新しい端末にて設定を行おうとすると、当然のことながらログイン時に MFA コードを求められた時に旧端末から呼び出しを行えないと元も子も無くなってしまうので、注意が必要です。

^{*12} スマホを機種変更する際は、MFA が有効なまま旧スマホを処分してしまうとサインイン時に MFA コードが確認できずマネジメントコンソールへ入れなくなってしまいます。機種変更前に一時的に MFA を無効にしておくか、新しいスマホを MFA デバイスとして登録してから旧スマホを処分するようにしましょう。

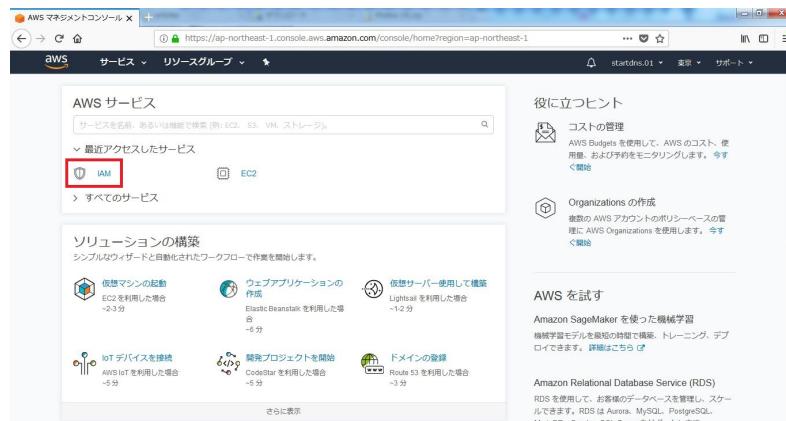


▲図 2.48 MFA を用いたサインインができるようになった！

2.4.4 ルートユーザーも MFA を有効にする

これで IAM ユーザーの「start-aws-user」は MFA が有効になりましたが、ルートユーザーはまだ MFA が有効になっていません。

いったん「start-aws-user」でサインアウトしたら、今度はルートユーザーでサインインしなおしてください。そしてマネジメントコンソールの「最近アクセスしたサービス」から「IAM」をクリック（図 2.49）して IAM のダッシュボードを開きます。



▲図 2.49 ルートユーザーで「最近アクセスしたサービス」から「IAM」をクリック

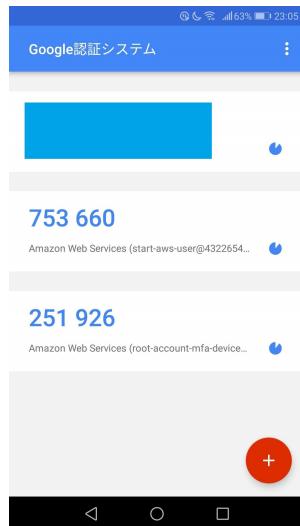
IAM のダッシュボードで「ルートアカウントの MFA を有効化」(図 2.50)*13を開いて「MFA の管理」をクリックします。



▲図 2.50 「ルートアカウントの MFA を有効化」を開いて「MFA の管理」をクリック

ここからは先ほどと同じ手順でルートユーザーでも仮想 MFA を有効にして、サインイン時は「Google 認証システム」を使うようにしておいてください。設定完了すると、Google 認証システムでもルートユーザー用の認証コード（図 2.51）が表示されるようになるはずです。

*13 突然「ルートアカウント」という言葉が出てきましたがルートアカウントとルートユーザーは同じものですね。他にもサインインとログインで揺れでていたりと、AWS では表記揺れはままあることです。日本語の翻訳がおかしいところもあるので、マネジメントコンソールの言語を英語にした方がいっそ分かりやすいかも知れません。英語が苦手な方は隨時脳内補完しながら頑張りましょう。

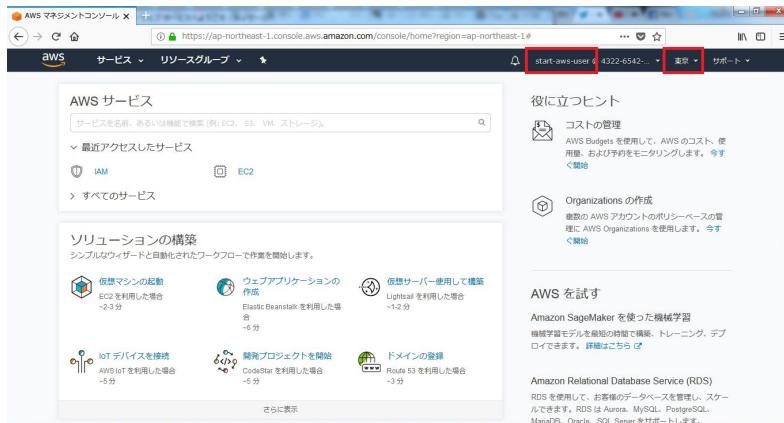


▲図 2.51 ルートアカウント用の認証コード（6桁の数字）も表示されている

ルートアカウントでも MFA が有効になつたら、再び「start-aws-user」でサインインしなおして先へ進みましょう。もしどこの URL からサインインするのか分からなくなくなつたら、先ほど送つておいたメールに「Sign-in URL: https://*****.signin.aws.amazon.com/console」という URL があるはずですので、そこをクリックしてサインインしましょう。MFA コードを聞かれたらスマホの Google 認証システムを起動して、表示されている 6 桁の数字を入力します。

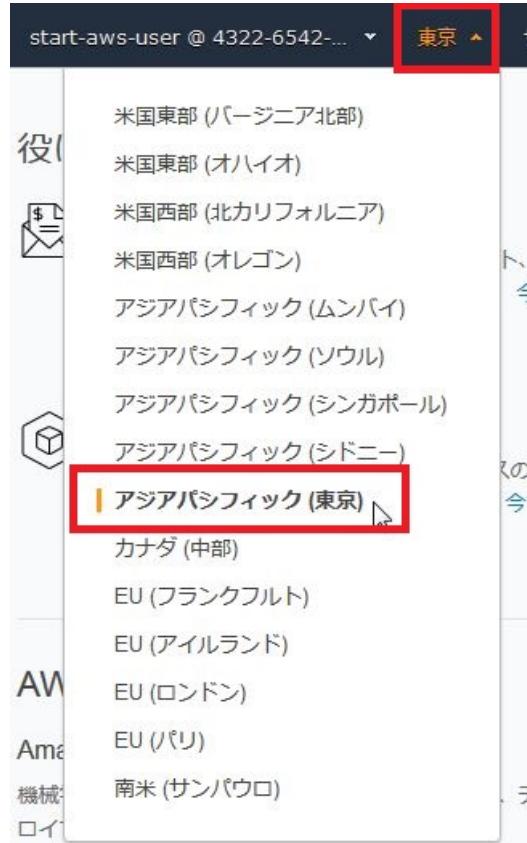
2.5 リージョンの変更

再び「start-aws-user」でサインインしなおして、マネジメントコンソールへ戻つてこられましたか？



▲図 2.52 右上に「start-aws-user」「東京」と表示されていたら OK

マネジメントコンソールの右上（図 2.52）に表示されているユーザー名が「start-aws-user」であること、そしてその右側に表示されているリージョンが「東京」であることを確認してください。ここがもし「東京」以外になっていたら、クリックして「アジアパシフィック（東京）」を選択（図 2.53）してください。選択後、右上が「東京」に変わったらリージョンの変更は完了です。



▲図 2.53 「東京」以外のときはクリックして「アジアパシフィック(東京)」を選択

AWS はバージニア、カナダ、ロンドン、シンガポール、東京など世界の各地域にデータセンターを所有しており、第 1 章「インフラとサーバってなに？」で詳しくお話ししたようにサーバはそのデータセンターの中にいます。

右上に表示されている「リージョン」とはその各地域の中でどこを使うか？を指定するものです。ウェブサイトにアクセスするとき、パソコンのある場所からサーバまで物理的に距離が遠いと、それだけ通信にも時間がかかるて応答時間も遅くなりますので、日本国内向けにウェブサイトを開設する場合は基本的にこの「東京」リージョンを選びましょう。ただし AWS のサービスによって、まだ東京リージョンが使えないものもあります。その場合は次点として「米国東部（バージニア北部）」を選択してください。¹⁴

*14 「米国東部（バージニア北部）」は AWS で最初に作られたリージョンで、新しいサービスはまずここに

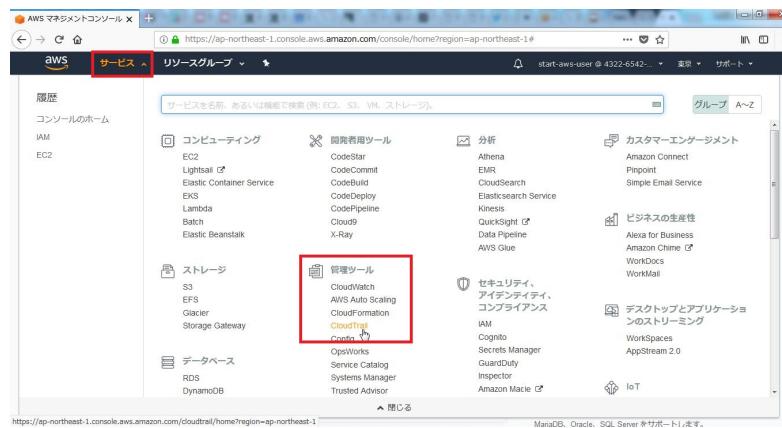
さらにリージョンという地域ごとの区分の下に、さらにアベイラビリティゾーン^{*15}という区分があります。アベイラビリティゾーンの場所は公開されていませんが、たとえば東京リージョンの下に「池袋アベイラビリティゾーン」と「品川アベイラビリティゾーン」がある、というようなイメージです。

それぞれのアベイラビリティゾーンは異なる場所に存在し、異なる変電所から電力を供給しています。そのため、もし東京リージョン内でどこかのアベイラビリティゾーンが局地的な災害に遭ったとしても、他のアベイラビリティゾーンは問題なく稼動しているので東京リージョン全体が止まってしまうことはありません。

この後、サーバを立てる際に誤って「東京」以外のリージョンでサーバを立てないように注意をしてください。

2.6 CloudTrail でいつ誰が何をしたのか記録

リージョンの確認が済んだら、続いて CloudTrail（クラウドトレール）を設定へ進みましょう。マネジメントコンソールの左上にある「サービス」から、「管理ツール」の下にある「CloudTrail」（図 2.54）をクリックしてください。



▲図 2.54 サービス>管理ツール> CloudTrail

「CloudTrail」をクリックすると、CloudTrail のダッシュボード（図 2.55）が表示されます。CloudTrail は AWS のマネジメントコンソールや、プログラムを通じて行われた

リージョンから提供開始されることが多いためです。ちなみに同じサービスでもリージョンによって料金は少しずつ異なります。

*15 アベイラビリティゾーンはよく AZ と略されます。

API呼び出しの履歴を保存してくれるサービス・・・と書くと難しいですが、要は「いつ誰が何をしたのか」を記録しておいてくれるサービスです。

The screenshot shows the CloudTrail Management Console dashboard. On the left, there's a navigation menu with 'CloudTrail' selected. Under 'CloudTrail', the 'イベント履歴' (Event History) option is highlighted with a red box. The main content area is titled 'CloudTrail へようこそ' (Welcome to CloudTrail). It says 'CloudTrail を使用すると、AWS アカウントのイベントを表示できます。これらのイベントの記録を保持するには、ルールを作成します。詳細により、イベントストリックスを作成し、アラートをトリガーして、イベントワークフローを作成することもできます。詳細はごちら' (When you use CloudTrail, you can view events in your AWS account. To keep records of these events, create rules. Depending on the details, you can create event streams, trigger alerts, and create event work flows.). Below this is a '最新情報' (Latest Information) section with a 'すべての更新を表示' (View all updates) link. The central part is titled '詳細はごちら' (See details) and has links for '料金表' (Billing), 'ドキュメント' (Documentation), 'フォーム' (Forms), and 'よくある質問' (FAQ). A table titled 'イベント時間' (Event Time) lists recent events:

イベント時間	ユーザー名	イベント名	リソースタイプ
2018-08-19, 12:03:08 AM	start-aws-user	DescribeConfigurationRecorder...	
2018-08-19, 12:03:08 AM	start-aws-user	DescribeConfigurationRecorders	
2018-08-19, 12:03:08 AM	start-aws-user	LookupEvents	
2018-08-19, 12:03:00 AM	start-aws-user	DescribeConfigurationRecorder...	
2018-08-19, 12:03:00 AM	start-aws-user	DescribeConfigurationRecorders	

At the bottom of the table, there's a link 'すべてのイベントを表示' (View all events).

▲図 2.55 CloudTrail ダッシュボード

CloudTrail ダッシュボードの左メニューにある「イベント履歴」をクリックして、イベントの一覧を確認してみましょう。CloudTrail はデフォルトで有効になっているため、今まで行ってきた「サインイン」や「IAM ユーザーの作成」などもすべてイベントとして記録されています。たとえばフィルターを「ユーザー名」にして「start-aws-user」で検索（図 2.56）してみると、いつサインインしたのか、いつ MFA（多要素認証）のチェックをしたのか、などが確認できます。^{*16}

^{*16} 表示されているイベント時間は UTC ですので日本時間とはずれています。

2.6 CloudTrail でいつ誰が何をしたのか記録

The screenshot shows the AWS CloudTrail Management Console interface. In the top navigation bar, 'CloudTrail' is selected under 'AWS' services. On the left sidebar, 'イベント履歴' (Event History) is selected. The main content area displays a table of events. A red box highlights the 'ユーザー名' (User Name) column, which shows 'start-aws-user' for all listed events. The table includes columns for 'イベント時間' (Event Time), 'ユーザー名' (User Name), 'イベント名' (Event Name), 'リソースタイプ' (Resource Type), and 'リソース名' (Resource Name). The events listed are:

イベント時間	ユーザー名	イベント名	リソースタイプ	リソース名
2018-08-18, 11:59:23 PM	start-aws-user	DescribeTrails		
2018-08-18, 11:17:08 PM	start-aws-user	ConsoleLogin		
2018-08-18, 11:17:02 PM	start-aws-user	CheckMfa		
2018-08-18, 11:16:56 PM	start-aws-user	ConsoleLogin		
2018-08-18, 11:16:15 PM	start-aws-user	CheckMfa		

▲図 2.56 start-aws-user がいつ何をしたのかすべて表示される

このように CloudTrail では何も設定しなくても、デフォルトで過去 90 日間^{*17}のイベントが無料で記録されています。今後もし「消した覚えがないのにサーバが跡形もなく消え去っている！」というようなことがあったら、先ず CloudTrail を開いていつ誰がサーバを削除したのか確認してみましょう。

これで「AWS を使い始めたら最初にやるべきこと」はひととおり完了しました。準備万端！ それでは次章でサーバを立てていきましょう！

*17 業務で使うので 90 日よりももっと前の記録も取っておきたい、という場合は「証跡（しょうせき）の作成」を行ってください。ただし証跡はさかのぼって保存はされないため、何か問題があつてから「100 日前のイベントが見たい！」と思つても見られません。

第 3 章

AWS でウェブサーバを立てよう

この章では実際に AWS の EC2 を使ってウェブサーバを立てます。

3.1 事前準備

3.1.1 お使いのパソコンが Windows の場合

Windows のパソコンを使っている方は、サーバを立てる前に「ターミナル」と呼ばれる黒い画面のソフトをインストールしておきましょう。サーバに接続するときにこのターミナルを使うのですが、ターミナルのソフトには色々な種類があります。

- RLogin (<http://nanno.dip.jp/softlib/man/rlogin/>)
- Poderosa (<https://ja.poderosa-terminal.com/>)
- Tera Term (<https://ja.osdn.net/projects/ttssh2/>)
- PuTTYjp (<http://hp.vector.co.jp/authors/VA024651/PuTTYkj.html>)^{*1}



▲図 3.1 RLogin

本著ではいちばん上の RLogin (図 3.1) を使って説明していきますので、特にこだわりがない場合は RLogin を使うことをお勧めします。RLogin の「実行プログラム (64bit)^{*2}」(図 3.2) の URL、http://nanno.dip.jp/softlib/program/rlogin_x64.zip をクリックしてください。

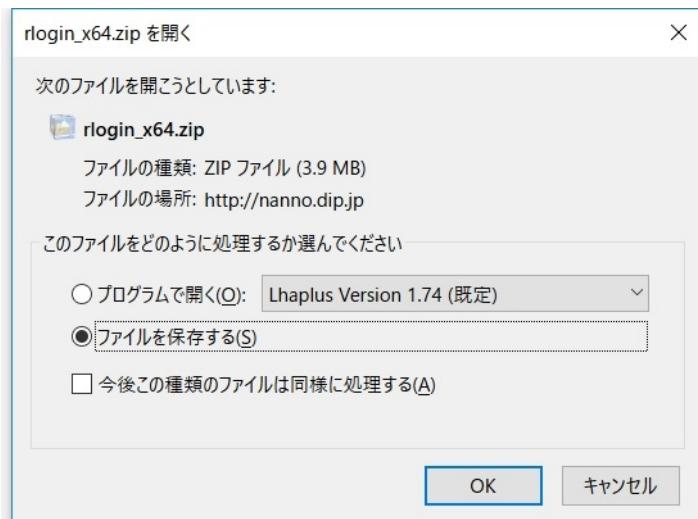
^{*1} PuTTYjp を使う場合、.pem の秘密鍵を PuTTYgen で.ppk に変換する必要があります。他のターミナルソフトに比べると一手間余計にかかるので、特にこだわりがなければ RLogin を使用するのがお勧めです。

^{*2} もしパソコンの Windows が 32bit 版だった場合は「実行プログラム (32bit)」の URL をクリックしてください。

GitHubからダウンロード	https://github.com/kmiya-culti/RLogin/releases/	過去30日間のアクセス数
実行プログラム(32bit)	http://nanno.dip.jp/softlib/program/rlogin.zip	Windows XP以降(32bit) 1001
実行プログラム(64bit)	http://nanno.dip.jp/softlib/program/rlogin_x64.zip	Windows XP以降(64bit) 5337
ソースファイル(GitHub)	http://nanno.dip.jp/softlib/source/rlogin.zip	Visual Studio 2010 191

▲図 3.2 「実行プログラム (64bit)」の URL をクリックしてダウンロード

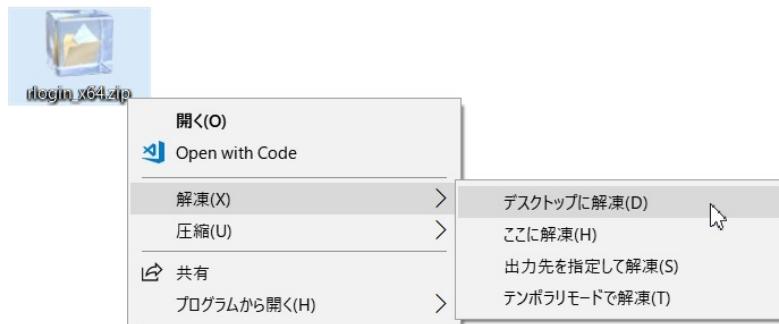
ダウンロードした ZIP ファイルを保存（図 3.3）します。保存場所はどこでも構いませんが、後でどこに置いたか分からなくなりそうな人はデスクトップに保存しておきましょう。



▲図 3.3 「ファイルを保存する」でパソコンに保存

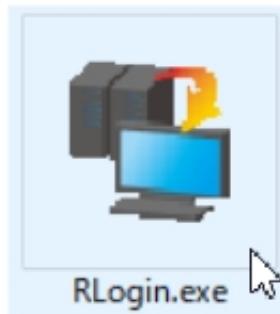
デスクトップの ZIP ファイル (rlogin_x64.zip) を右クリック（図 3.4）して、解凍>デスクトップに解凍^{*3}をクリックします。

^{*3} 右クリックしても「解凍」が見当たらないときは、圧縮・解凍の定番ソフトである Lhaplus をインストールしましょう。 <https://forest.watch.impress.co.jp/library/software/lhaplus/>



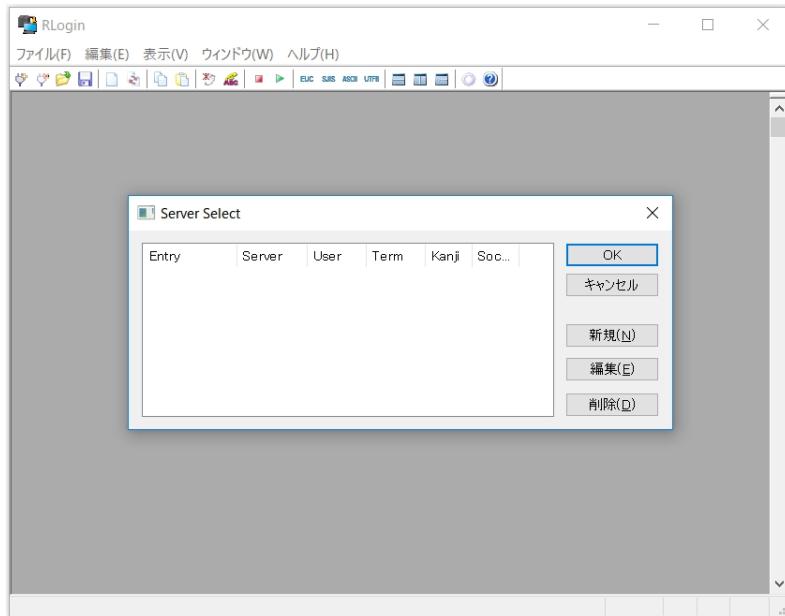
▲図3.4 ZIPファイルを右クリックして解凍>デスクトップに解凍

解凍したら、デスクトップにできた「rlogin_x64」というフォルダの中にある「RLogin.exe」^{*4}（図3.5）をダブルクリックすればRLoginが起動（図3.6）します。



▲図3.5 RLogin.exeをダブルクリック

^{*4} フォルダの中にRLoginはあるけどRLogin.exeなんて見当たらない・・・という場合、ファイルの拡張子が非表示になっています。この後も拡張子を含めてファイル名を確認する場面ができますので、「拡張子表示」でGoogle検索して拡張子が表示されるように設定変更しておきましょう。



▲図 3.6 RLogin が起動した

これで RLogin のインストールは完了です。起動した RLogin はいったん閉じてしまつて構いません。また後で使いますので、デスクトップの「rlogin_x64」フォルダとその中にある「RLogin.exe」をごみ箱へ捨てないように注意してください。

3.1.2 お使いのパソコンが Mac の場合

Mac を使っている方は、最初から「ターミナル」というソフトがインストールされていますのでそちらを利用しましょう。ターミナルがどこにあるのか分からぬときは、Mac の画面で右上にある虫眼鏡のマークをクリックして、Spotlight で「ターミナル」と検索すれば起動できます。

以上で事前準備は完了です。お待たせしました。いよいよサーバを立てましょう。

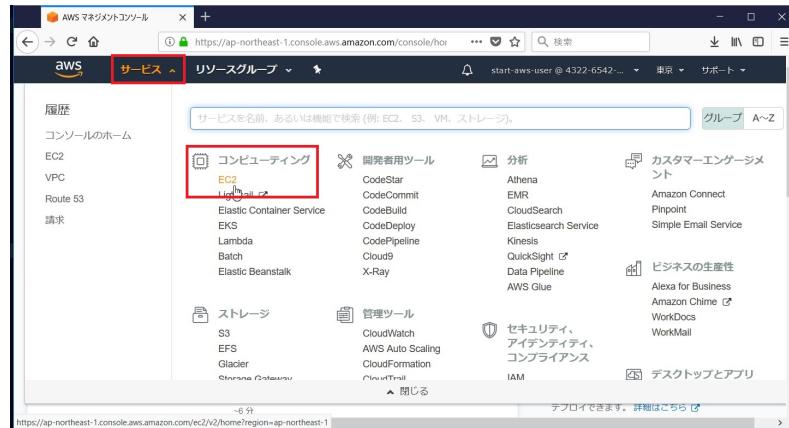
3.2 EC2 でウェブサーバを立てる

AWS には Route53 のようなネームサーバをはじめとして色々なサービスがありますが、サーバは Amazon Elastic Compute Cloud の略で「EC2」(イーシーツー) と呼ばれています。

第3章 AWSでウェブサーバを立てよう

ちなみにAWSではサーバのことを、**インスタンス**と呼びます。ここから先でインスタンスと書いてあつたらサーバのことだと思ってください。

それではマネジメントコンソールの左上にある「サービス」から、「コンピューティング」の下にある「EC2」(図3.7)をクリックしてください。



▲図3.7 サービス>コンピューティング>EC2

「EC2」をクリックすると、EC2のダッシュボード(図3.8)が表示されます。「注意: インスタンスはアジアパシフィック(東京)リージョンで作成されます」と書いてあることを確認したら、中央にある「インスタンスの作成」をクリックしてください。



▲図3.8 EC2ダッシュボードで「インスタンスの作成」をクリック

ここから 7 つのステップでインスタンスを作成していきます。

3.2.1 ステップ 1: Amazon マシンイメージ (AMI)

はじめに Amazon マシンイメージ、略して AMI を選択します。AMI はこれから作るサーバのもととなるテンプレートのようなものです。

左側の「無料利用枠のみ」にチェックを入れる（図 3.9）と、無料利用枠以外の AMI は選択できなくなります。うつかり Windows Server のような有料 AMI を選択しないようチェックを入れておきましょう。



▲ 図 3.9 「無料利用枠のみ」にチェックを入れて「Amazon Linux AMI」を選択

パソコンには OS という基本ソフトが入っていて、Word や Excel、Chrome といったソフトはその OS の上で動いています。皆さんのパソコンにも「Windows 10」や「Mac OS X Lion」などの OS が入っていますよね。

そしてパソコンと同じようにサーバにも「Linux」や「Windows Server」といったサーバ用の OS があります。サーバを立てるときには Linux を選択することが多いのですが、この Linux の中にもさらに「RHEL (Red Hat Enterprise Linux)」や「CentOS」、「Ubuntu」などいろいろなディストリビューション（種類）があります。

今回は上から 2 つめにある「Amazon Linux」の AMI を選択します。Amazon Linux なら AWS のツールがあらかじめ入っており、Amazon による OS のサポートも受けられる^{*5}ため、AWS でサーバを立てるときは OS は Amazon Linux にすることをお勧めし

^{*5} Amazon による OS のサポートというのとは「手取り足取り教えてくれる」ということではなく、たとえば「バグや脆弱性が発見されたときに修正バージョンを出してくれる」というものです。サポートの期限

ます。Amazon LinuxはRed Hat系のディストリビューションですので、Red HatやCentOSのサーバを使ったことがある方なら違和感なく使えると思います。

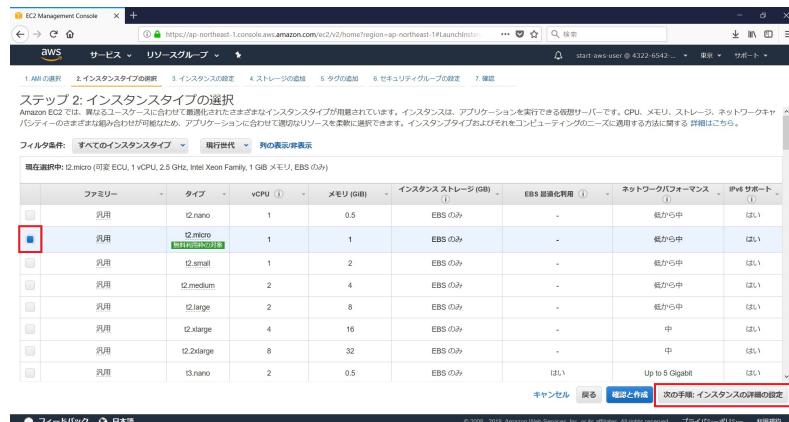
Amazon Linuxには2018年8月時点で

- Amazon Linux
- Amazon Linux 2

の2種類があります。Amazon LinuxはRHEL6系なのでCentOS6、Amazon Linux2はRHEL7系なのでCentOS7と使い勝手はほぼ同じです。本著ではAmazon Linuxを選択します。^{*6}

3.2.2 ステップ2: インスタンスタイプの選択

ステップ2ではインスタンスタイプを選択(図3.9)します。テスト環境にはT2インスタンス、データベースやキャッシュを処理させるならM5インスタンス、というように用途別にたくさん用意されているので、その中から適したスペックのインスタンスタイプを選びます。



▲図3.10 「無料利用枠のみ」にチェックを入れて「Amazon Linux AMI」を選択

インスタンスタイプの接頭辞になっている「T2」や「M4」はインスタンスマリードと呼ばれるグループを表しており、Tが開発・検証用、Mが汎用、CがCPU重視、Rは

はAmazon Linuxが2020年6月30日まで、Amazon Linux2が2023年6月30日まで、と公式に発表されています。<https://aws.amazon.com/jp/amazon-linux-2/faqs/>

*6 ちなみにこの後インストールするApacheというミドルウェアはAmazon Linuxだと2.2系、Amazon Linux2だと2.4系になります。

メモリ重視、のように特徴ごとに分かれています。文字の後ろの 2 や 4 といった数字は世代を表しているので、T2 なら開発・検証向けのグループで 2 世代目ということですね。T は 3 世代目となる T3 もリリースされたので、2018 年 8 月時点では T2 と T3 がどちらも選択できる状態になっています。

インスタンスファミリーの後ろにある nano、micro、small、medium、large、xlarge などは CPU やメモリといったスペックの大きさを表します。t2.small なら CPU^{*7}が 1 コアでメモリが 2GiB、t2.medium なら CPU は 2 コアでメモリが 4GiB というように、大きくなるにつれて段々 CPU やメモリが増えています。^{*8}

今回は個人で「ちょっと WordPress のサイトを作ってみよう！」という用途なので高スペックは必要ありません。無料利用枠の対象となっている t2.micro を選択して「次の手順: インスタンスの詳細の設定」をクリックします。

^{*7} インスタンスタイプの表では CPU が「vCPU」と書かれていると思います。仮想サーバ内にある仮想の CPU のことを Virtual CPU、略して vCPU といいます。

^{*8} S → M → L と段々量が増えていくなんてマクドナルドのポテトと同じですね。

【コラム】T2系バーストモードの落とし穴

T2系のインスタンスタイプには落とし穴があるので注意が必要です。

たとえば先ほど選択したt2.microというインスタンスタイプはCPUが1コアと書いてありますが、実際はベースラインという「普段はここまで使っていいよ」というラインがあり、サイトにあまりアクセスが来ておらずサーバがヒマなとき、vCPUの使用率がこのベースラインを下回っていれば「CPUクレジット」というものがちやりんちやりんと貯まっていきます。^aCPUクレジットはt2.microなら1時間あたり6ずつ溜まっていって最大で144まで蓄積できます。

前述のベースラインですがt2.microだとなんとたったの10%です。普段はvCPUを1コアの1/10しか使えない、ということです。そしてサイトにアクセスがわーっと殺到してCPU使用率がベースラインの10%を超えるとバーストモードになり、バーストモードの間だけはvCPUが100%使えるようになります。

バーストモードの間は今まで貯めておいたCPUクレジットを使います。1クレジットにつき1分間バーストできるので、t2.microなら最大で144分しかバーストできません。つまりCPU使用率が10%を超える状態が2時間半続いたら、その時点でバーストが終了して強制的にまたvCPUが10%までしか使えなくなってしまうのです。

アクセスが殺到していたからvCPUを10%以上使っていたわけで、それが急に10%までしか使えなくなってしまったらどうなるのでしょうか？バーストが終了した瞬間にサーバは過負荷となり、場合によってはサイトが応答できなくなってしまいます。

つまり今までオンプレミスでCPUが1コアのサーバを使っていてそれがちょうどよかったからといって、AWSでt2.microを選ぶとベースラインが10%なのでvCPUは実質0.1しかないため、AWSに引越しした途端サイトが落ちまくって「同じスペックを選んだはずなのにどうして？！」という事態になる可能性があるということです。

CPUクレジットが尽きたら追加課金でバーストモードを延長できるT2 Unlimitedというオプションもありますが、Unlimitedで延々課金されるくらいならもう少し上のインスタンスタイプを選んでベースラインを超えないようにしましょう。ちなみに3世代目のT3系はこのUnlimitedがデフォルトで有効になっているので「なーんだ、t3.microでも結構アクセスさばけるじゃん！」と思っていると、実はUnlimitedで延々課金されていた！となる初心者殺しな仕様といえます。

^a https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/t2-credits-baseline-concepts.html

3.2.3 ステップ 3: インスタンスの詳細の設定

3.2.4 ステップ 4: ストレージの追加

3.2.5 ステップ 5: タグの追加

3.2.6 ステップ 6: セキュリティグループの設定

3.2.7 ステップ 7: インスタンス作成の確認

3.3 SecurityGroup

3.4 VPC

3.5 EC2

3.5.1 請求アラート

3.5.2 SSH の鍵認証

3.5.3 鍵の変換

3.5.4 ElasticIP

3.5.5 Bastion

第 4 章

サーバのバックアップを取っておこう

4.1 スナップショット

4.2 AMI

第 5 章

ELB でバランスシングやサーバの台数を管理しよう

5.1 ELB

5.2 Auto Scaling

5.2.1 スケーリングに使える

5.2.2 サーバが 1 台死んでも自動で 1 台立ち上がる

第 6 章

DB サーバを立てよう

6.1 RDS

6.2 Amazon Aurora

第 7 章

ネームサーバの設定をしよう

7.1 Route53

第 8 章

もっと AWS について勉強したい！

8.1 公式のオンラインセミナーや資料集

AWS についてもっと勉強したい！ という場合は、ネットに繋がればどこからでも参加できる「AWS Black Belt Online Seminar」というオンラインセミナーを受けてみましょう。

<https://aws.amazon.com/jp/about-aws/events/webinars/>

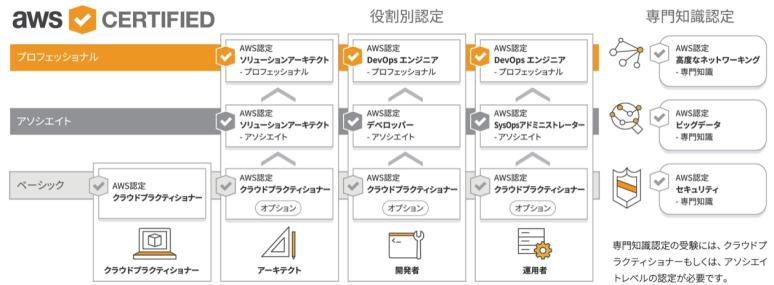
過去に開催されたオンラインセミナーの資料は「AWS クラウドサービス活用資料集」で公開されています。EC2 や ELB などのサービス別に資料が用意されていますので、そちらを読んでみるのもお勧めです。

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

8.2 AWS 認定資格のクラウドプラクティショナーを目指してみよう

AWS には公式の認定資格（図 3.11）^{*1}がいくつかあるのですが、その中で最初に挑戦しやすい入門者向けの認定資格は「クラウドプラクティショナー」です。

^{*1} AWS 認定より引用。 <https://aws.amazon.com/jp/certification/>



▲図 8.1 AWS 認定資格のロードマップ

認定資格に挑戦することで AWS の主要なサービスやセキュリティの基本、料金体系などをまんべんなく学ぶことができます。本著を読んで「もっと AWS について勉強したいな！」と思ったらチャレンジしてみてはいかがでしょうか？

第 9 章

AWS をやめたくなったらすること

9.1 無料の 1 年が終わる前にすべきこと

9.1.1 【ドリル】サンプル

問題

問題問題

- A. Route53
- B. お名前.com

答え _____

解答

正解は B です。

付録 A

本当の Git

またしても何を言っているのかわからないと思いますが、「Git 用語だけでアイドルソングを作って架空のアイドルに歌わせたい」そんな気持ちで作詞をしてしまいました。大切な事が厳しい時ほど才能が花開いてしまう傾向にある、と言い訳をしたいのですが、本著を書くにあたって最初に着手したのがこの付録だったということは、GitHub でコミットログを見れば一目瞭然です。それでは聞いてください。

A.1 Git - ぎゅっと言えないトウインクル

いま何してる？ リモートのあなた

ガマンできずに フェッチして

髪型変えたの 知ったの

あなたへの気持ち 切なくて

思わずスタッショ したまま ずっと埃つもってる

下駄箱に入れた プルリクエスト

変わっていくわたしを ちゃんとプルして抱きしめて

分かれてしまった ふたりのプランチ

コミットログ読めば あの日の気持ちも分かるはず

たくさんのライバル わたしだけをチェリーピックして

きっとわたし あなたのクローン

フォークしたあの日から ずっとあなたを見つめてる

ステージに上がったら もうコミット逃げられない

ためらわないので オリジンにプッシュ

リバートしたって 過去がなくなるわけじゃない

ただ逆の気持ちで 打ち消しただけ

別々に歩んだ ふたりの過去も

リベースすれば ひとつになれるわ

ねえ いますぐ抱きしめて

ぎゅっと言えない トウインクル

あとがき

2018年10月
mochikoAsTech

Special Thanks:

- プロッコリー好きな茶色い猫に捧ぐ

レビュアー

-

参考書

-

著者紹介

mochiko / @mochikoAsTech

Web 制作会社のシステムエンジニア。モバイルサイトのエンジニア、SIer とソーシャルゲームの広報を経て、2013 年よりサーバホスティングサービスの構築と運用を担当。現在は再び Web アプリケーションエンジニアとしてシステム開発に従事。「分からぬ気持ち」に寄り添える技術者になれるように日々奮闘中。

- <https://mochikoastech.booth.pm/>
- <https://twitter.com/mochikoAsTech>

Hikaru Wakamatsu

表紙デザインを担当。「DNS をはじめよう」の名付け親。

Shinya Nagashio

挿絵デザインを担当。

AWS をはじめよう

2018年10月8日 技術書典5版 v1.0.0

著者 mochikoAsTech

デザイン Hikaru Wakamatsu / Shinya Nagashio

発行所 mochikoAsTech

印刷所 日光企画

(C) 2018 mochikoAsTech