

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	29
1.4.6 AWS以外のクラウド	30
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	CloudWatch で請求アラートを設定しよう	37
2.5	IAM でユーザの権限管理	43
2.5.1	ルートユーザーの普段使いはやめよう	43
2.5.2	IAM ユーザを作ろう	44
2.5.3	MFA (多要素認証) で不正利用から IAM ユーザーを守る	57
2.5.4	ルートユーザーも MFA を有効にする	70
2.6	リージョンの変更	72
2.7	CroudTrail でいつ誰が何をしたのか記録	75
第3章	AWS でサーバを立てよう	79
3.1	事前準備	80
3.1.1	お使いのパソコンが Windows の場合	80
3.1.2	お使いのパソコンが Mac の場合	83
3.2	EC2 でサーバを立てる	83
3.2.1	ステップ 1: Amazon マシンイメージ (AMI)	85
3.2.2	ステップ 2: インスタンスタイプの選択	86
	【コラム】T2 系バーストモードの落とし穴	88
3.2.3	ステップ 3: インスタンスの詳細の設定	89
3.2.4	ステップ 4: ストレージの追加	89
3.2.5	ステップ 5: タグの追加	90
3.2.6	ステップ 6: セキュリティグループの設定	91
3.2.7	ステップ 7: インスタンス作成の確認	92
3.2.8	キーペアのダウンロード	93
3.2.9	作成したインスタンスに名前をつける	97
3.2.10	【ドリル】秘密鍵をなくしたらどうなる?	98
	【コラム】サーバは「立てる」もの? 「建てる」もの?	99
3.3	サーバに「入る」とは?	99
3.3.1	SSH とは?	101
3.3.2	パスワード認証と鍵認証	102
3.3.3	接続先となるサーバの IP アドレス	102
3.4	SSH でサーバに入ってみよう	104
3.4.1	お使いのパソコンが Windows の場合	104
3.4.2	お使いのパソコンが Mac の場合	112
3.4.3	サーバをシャットダウンしてみよう	113
3.4.4	再起動しても変わらない ElasticIP をつけよう	117

3.4.5	サーバに入るときに使うドメイン名を作ろう	124
3.5	ターミナルでサーバを操作・設定してみよう	132
3.5.1	ターミナルの基本操作に慣れよう	132
3.5.2	ミドルウェアをインストール	135
3.5.3	OS の基本設定をしておこう	136
3.5.4	ターミナルはなんのためにある？	144
第 4 章	ウェブサーバの設定をしよう	147
4.1	ウェルカムページを見てみよう	148
4.1.1	セキュリティグループで 80 番ポートの穴あけをしよう	149
4.2	自分のサイト用にバーチャルホストを作ろう	151
4.2.1	バーチャルホストとは?	152
4.2.2	バーチャルホストを設定しよう	152
4.2.3	設定ファイルの構文チェック	157
4.2.4	ドキュメントルートを作成	157
4.2.5	index.html を置いてみよう	158
4.2.6	curl でページを確認しよう	160
4.3	Route53 で自分のサイトのドメインを設定しよう	160
4.3.1	www のサブドメインを作ろう	161
4.3.2	ブラウザでページを見てみよう	164
4.3.3	アクセスログとエラーログの大切さ	165
第 5 章	DB サーバを立てよう	167
5.1	RDS で WordPress 用のインスタンスを立てよう	168
5.1.1	パラメータグループを作成	168
5.1.2	RDS インスタンスの作成	168
5.1.3	ウェブサーバから接続確認してみよう	168
第 6 章	WordPress でサイトを作ろう	169
6.1	WordPress のインストール	170
6.2	管理画面にログイン	170
6.2.1	管理画面にダイジェスト認証をかけよう	170
第 7 章	サーバのバックアップを取っておこう	171
7.1	EBS スナップショットと AMI	172
7.2	インスタンスから AMI を作ろう	174

第 8 章 ELB でバランシングやサーバの台数管理をしよう	179
8.1 ELB	180
8.2 Auto Scaling	180
8.2.1 スケーリングに使える	180
8.2.2 サーバが 1 台死んでも自動で 1 台立ち上がる	180
8.3 Route53 で A レコードを変更して Alias にしよう	180
8.3.1 ページを確認してみよう	180
8.4 セキュリティグループの変更	180
8.4.1 ELB 経由の 80 番ポートを許可しよう	180
8.4.2 EIP の 80 番ポートは閉じておこう	180
第 9 章 もっと AWS について勉強したい！	181
9.1 公式のオンラインセミナーと資料集	182
9.2 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	182
9.3 Linux やコマンドも学びたいなら	183
第 10 章 AWS をやめたくなったらすること	185
10.1 無料の 1 年が終わる前にすべきこと	186
10.1.1 AWS アカウントを停止する	186
付録 A 本当の Git	189
A.1 Git - ぎゅつと言えないトワインクル	190
あとがき	191
Special Thanks:	191
レビュー	191
参考書	191
著者紹介	193

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。

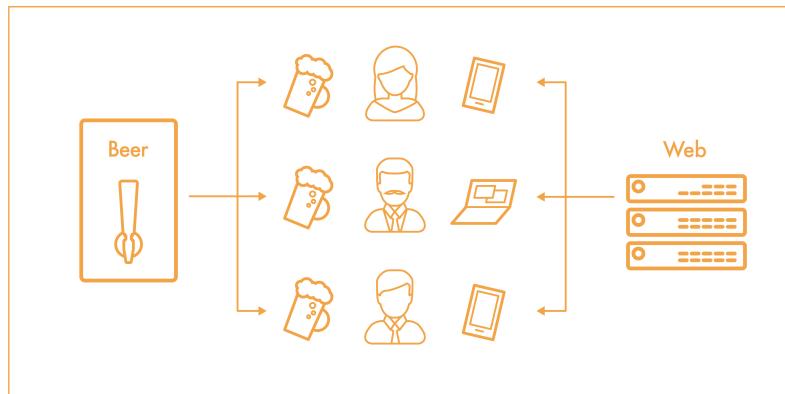
でももし後輩から「サーバってなんですか？」という直球の質問を投げつけられたら、しっかりとホームランで打ち返せますか？

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。

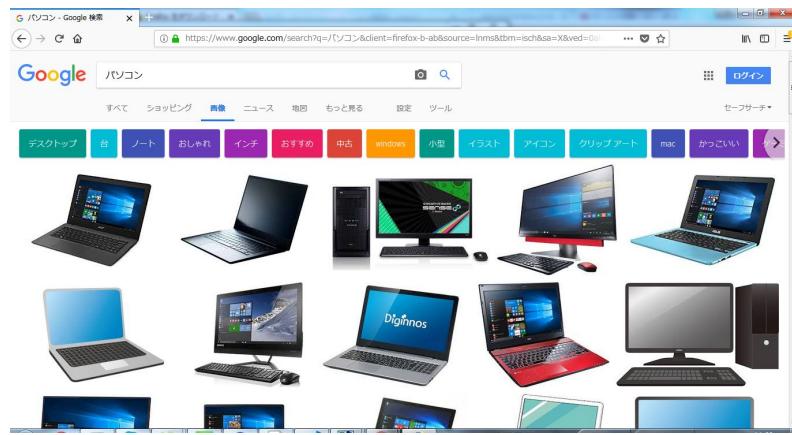
ところでちょっと頭の中でサーバの姿を思い浮かべてみてください。できればイラストじゃなくて実写でお願いします。



▲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？

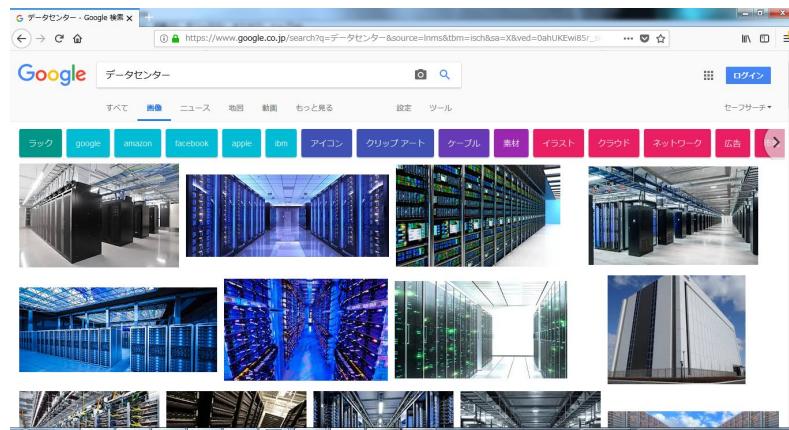
どんなウェブサイトも、世界中のどこかにあるサーバの中で稼動しているはずなのですが、インフラエンジニアでなければサーバを見る機会はなかなかないかも知れません。

^{*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に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なら、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や免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。

^{*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・・・・。

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りってきてサーバがどんどん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、電源容量を増やす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。

^{*12} アニメ映画の「スマーウォーズ」で、冷却用の水が部屋からなくなっこたことでスーパーコンピュータが熱暴走し、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。

*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。

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稟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・管理するのを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。

^{*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。

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分使ったら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つ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なら使い始めのときの初期費用もなく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・・」ということはありません。「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

^{*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。

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）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ってしまっても渋谷のデータ

*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。

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ってフードコート全体がお休みになるとときは、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。こうした障害の際に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ょう。

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？

*19 このように火事や自然災害などが起きても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。

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？

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！」と言いたい人がいました。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、よくよく聞いてみないと分からない、ということです。^{*22}

*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。

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っ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。

*²³ <https://cloud.google.com/>

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

*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と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な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

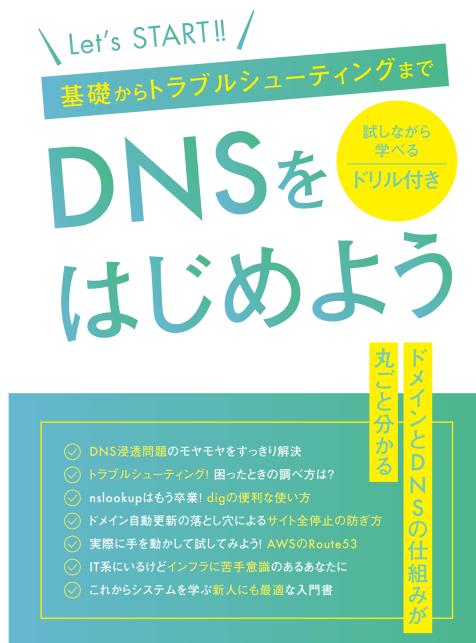
AWS を使うには、先ず AWS アカウントを作成する必要があります。

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。

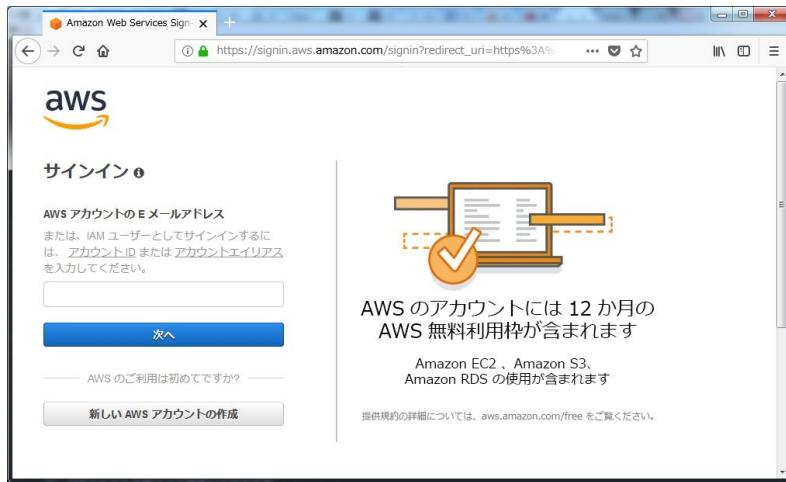
^a <https://mochikoastech.booth.pm/>

^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>

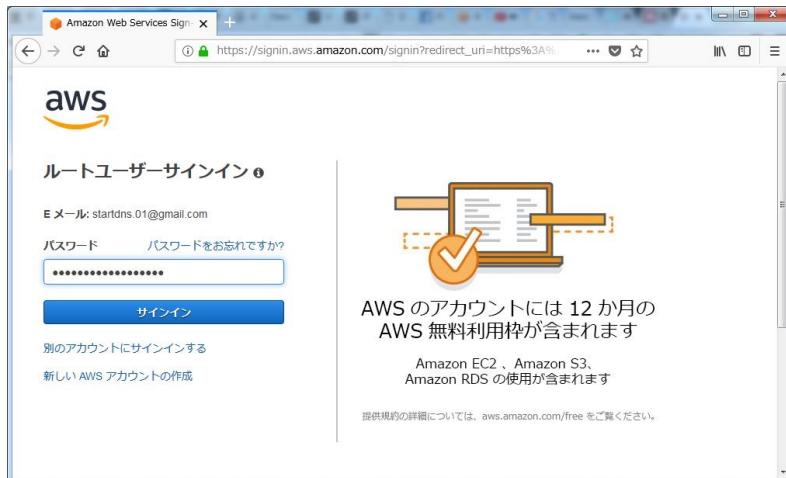
2.3 マネジメントコンソールにサインイン

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。



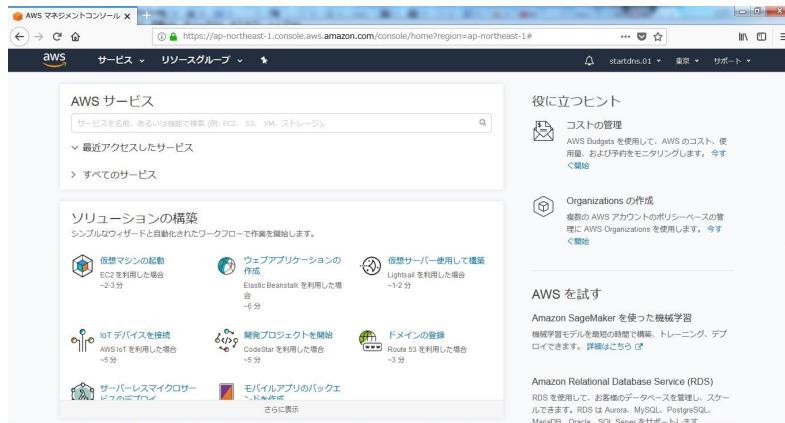
▲図 2.3 マネジメントコンソールのサインイン画面

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 _____

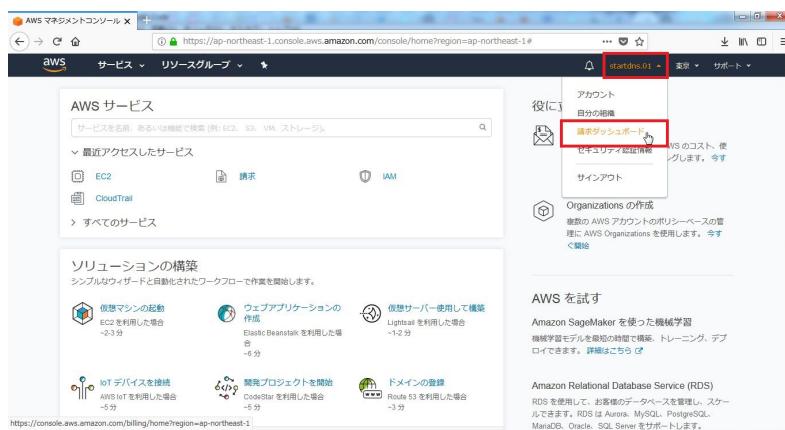
解答

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 CloudWatch で請求アラートを設定しよう

AWS はどのサービスも従量課金です。誤って高スペックで高額なサーバを立ててしまい 1か月後に青ざめる・・・ということにならないよう、利用額が一定金額を超したらメールでアラートを飛ばす設定をしておきましょう。こうしたアラートの設定は AWS の CloudWatch というモニタリングサービスで行います。

右上のルートユーザー名（図 2.6）から「請求ダッシュボード」をクリックしてください。



▲ 図 2.6 ルートユーザー名>請求ダッシュボード

請求ダッシュボード（図 2.7）が表示されたら、左メニューの「設定」をクリックします。



▲図 2.7 請求ダッシュボードで左メニューの「設定」をクリック

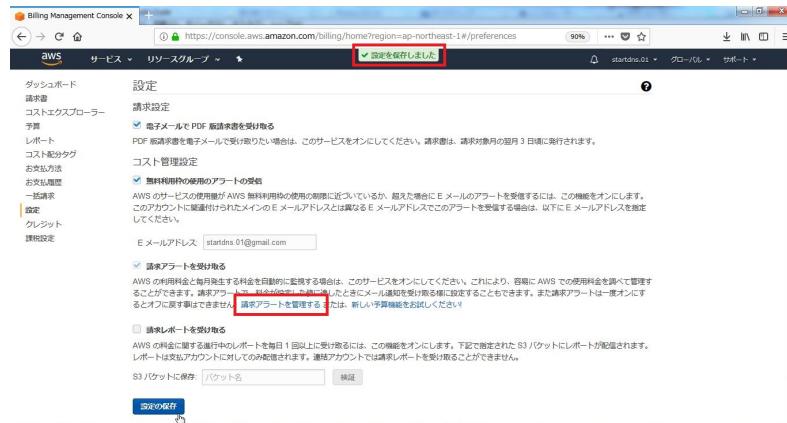
先ず一番上の「電子メールで PDF 版請求書を受け取る」にチェック（図 2.8）を入れます。続いて「無料利用枠の使用のアラートの受信」には元々チェックが入っていると思いますので、その下の「E メールアドレス:」に自分のメールアドレスを記入してください。その下にある「請求アラートを受け取る」にもチェックを入れたら「設定の保存」をクリックします。

The screenshot shows the 'Settings' (設定) page in the AWS Billing Management Console. Several checkboxes are checked: '請求メールで PDF 版請求書を受け取る' (Checkmark), '無料利用枠の使用のアラートの受信' (Checkmark), and '請求アラートを受け取る' (Checkmark). The 'E メールアドレス' field contains 'startdns.01@gmail.com'. At the bottom, the 'Save Settings' (設定の保存) button is highlighted with a red box.

▲図 2.8 チェックを入れてメールアドレスを記入したら「設定の保存」をクリック

上部に「設定を保存しました」と表示（図 2.9）されたら、次は「請求アラートを管理する」リンクをクリックします。

2.4 CloudWatch で請求アラートを設定しよう



▲図 2.9 保存できたら「請求アラートを管理する」をクリック

CloudWatch のダッシュボードが表示（図 2.10）されました。「請求アラームを作成すると」と書かれたリンクをクリックします。



▲図 2.10 「請求アラームを作成すると」と書かれたリンクをクリック

アラームの作成画面（図 2.11）で「超過」という欄に 1USD と書くと、今月の利用料が 1 ドルを超えた時点でアラートメールが来ます。本著の内容で AWS を利用した場合、無料利用枠からはみ出す分が最大でも月 2 ドル程度ですのでここには 2 と書いておきます。「通知の送信先」には自分のメールアドレスを記入してください。



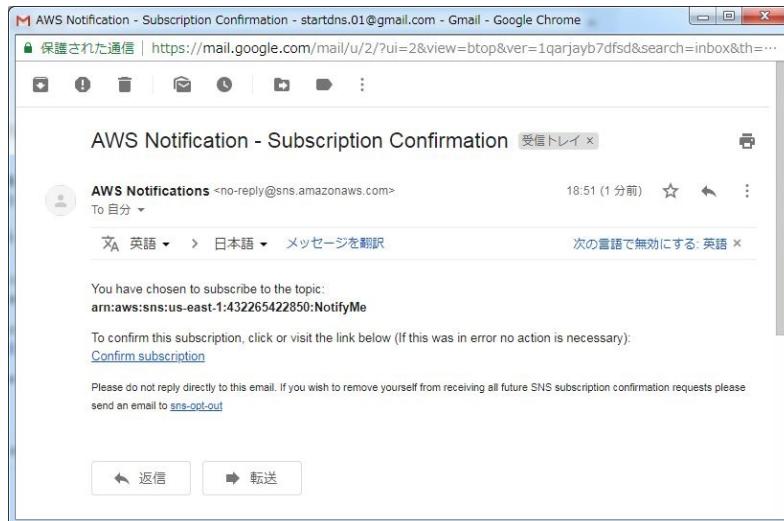
▲図 2.11 金額とメールアドレスを記入したら「アラームの作成」をクリック

「超過」の金額と「通知の送信先」のメールアドレスを記入したら「アラームの作成」をクリックしてください。「新しいメールアドレスの確認」(図 2.12) と表示されます。



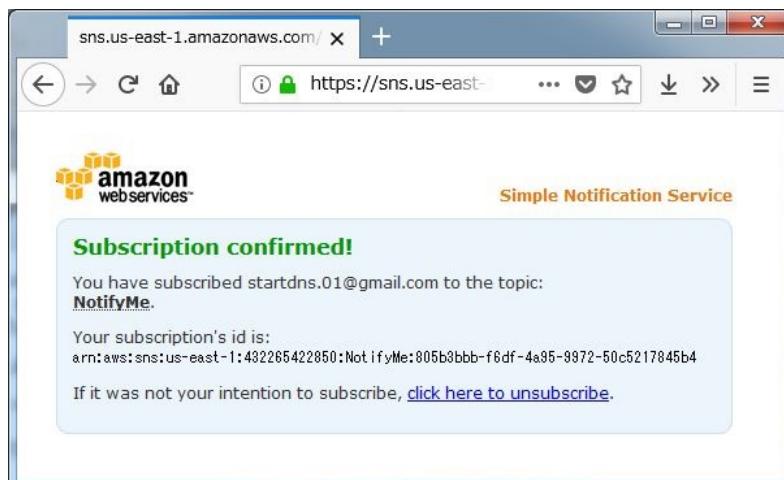
▲図 2.12 通知の送信先として新しいメールアドレスを登録すると確認が行われる

通知の送信先として新しいメールアドレスを登録した場合、「AWS Notification - Subscription Confirmation」という件名でメールアドレスの確認メール(図 2.13)が届きます。メール本文にある「Confirm subscription」というリンクを踏んでください。



▲図 2.13 メール本文中の「Confirm subscription」をクリック

Subscription confirmed!と表示（図 2.14）されたらこの画面は閉じてしまって構いません。



▲図 2.14 Subscription confirmed!と表示されたらこの画面は閉じる

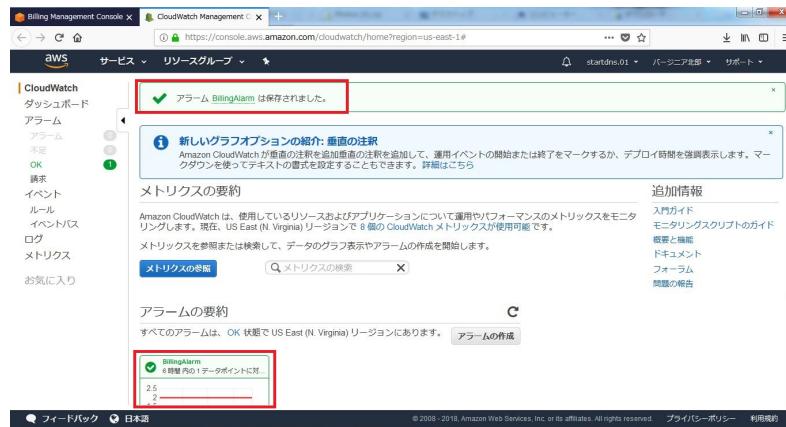
すると元の画面でメールアドレスの横に緑のチェックマークがつく（図 2.15）ので、

「アラームの表示」をクリックしてください。



▲図 2.15 緑のチェックマークがついていたら「アラームの表示」をクリック

上部に「アラーム BillingAlarm は保存されました。」と表示（図 2.16）されていることを確認したら、「BillingAlarm のグラフ」をクリックしてください。



▲図 2.16 「BillingAlarm のグラフ」をクリック

BillingAlarm の状態が OK（図 2.17）になっていたら請求アラートの設定は完了です。もしオレンジ色で「データ不足」と表示されていた場合は、右上の更新マークをクリックしてみてください。



▲図 2.17 BillingAlarm の状態が OK になっていたら請求アラートの設定は完了

これで請求額が閾値の 2 ドルを超したら「ALARM: "BillingAlarm"」という件名で請求アラートのメールが届くようになりました。英語のメールが届いたからといってスルーしないように注意してくださいね。

2.5 IAM でユーザの権限管理

2.5.1 ルートユーザーの普段使いはやめよう

さて、皆さんのがいま使っているのは「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。

「ルートユーザってなんでもできるユーザなんでしょ？ 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、量気良いくちばん高

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2.18)



▲図2.18 「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まで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。

2.5.2 IAM ユーザを作ろう

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

^{*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、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？ と思いますよね。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。

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ましたり、特定のプロジェクトル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ましたりします。

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

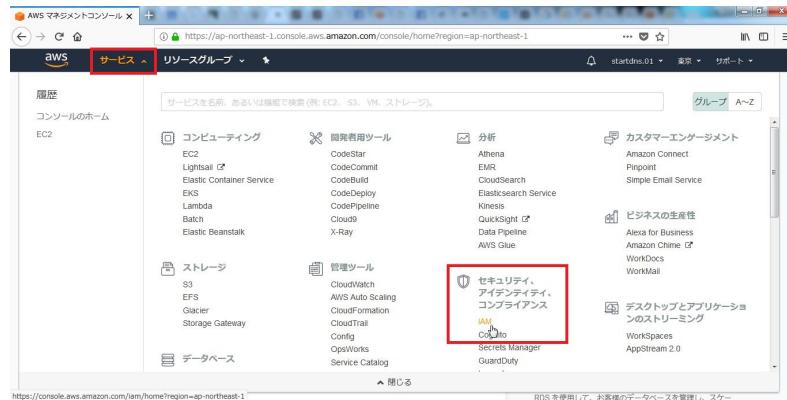
本著では 1 人だけで作業をする想定なので、IAM ユーザーも 1 人しか作りません。ですがもし業務などで複数名でマネジメントコンソールにサインインする場合は、IAM ユーザーは必ず 1 人につき 1 つずつ作成してください。まったく同じ作業をするから開発チーム内の A さんと B さんは同じ IAM ユーザーを共有すればいいのでは？と思われる場合でも、必ず A さん B さんそれぞれに別々の IAM ユーザーを用意することをお勧めします。

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

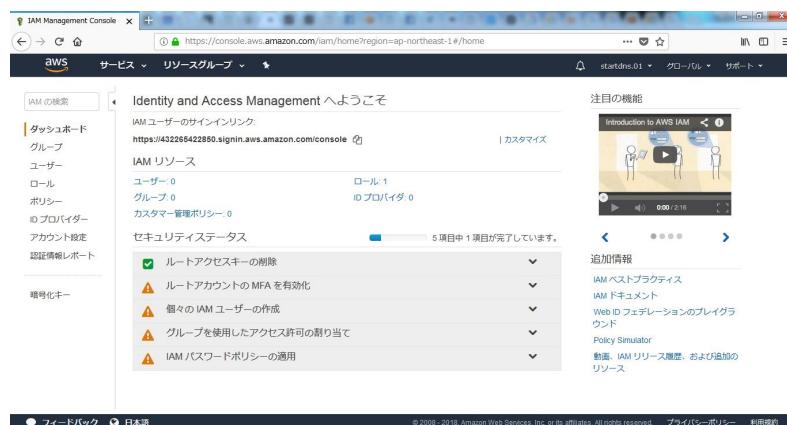
それではマネジメントコンソールの左上にある「サービス」から、「セキュリティ、アイデンティティ、コンプライアンス」の下にある「IAM」（図 2.19）をクリックしてください。

^{*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 2.19 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 2.20）が表示されます。



▲図 2.20 IAM ダッシュボード

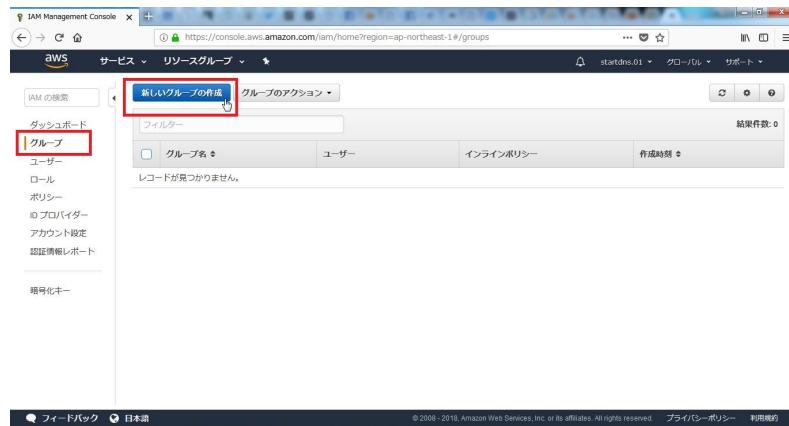
IAM のグループを作成

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って、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 2.21）します。



▲図 2.21 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。

先ずは手順 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 2.22）してください。



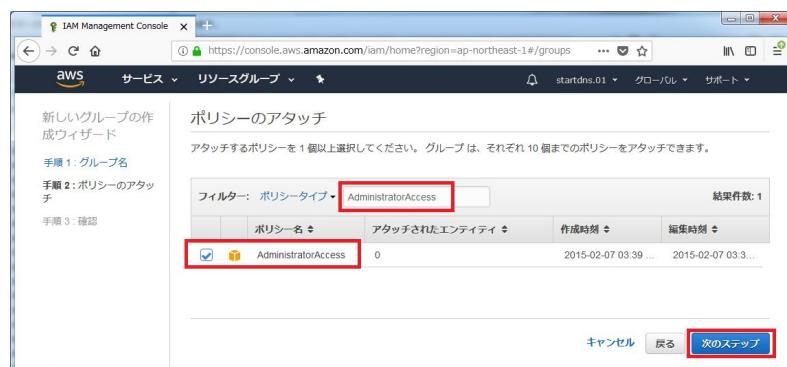
▲図 2.22 グループ名に「start-aws-group」と入力して「次のステップ」をクリック

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

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「AdministratorAccess」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2.23）してください。

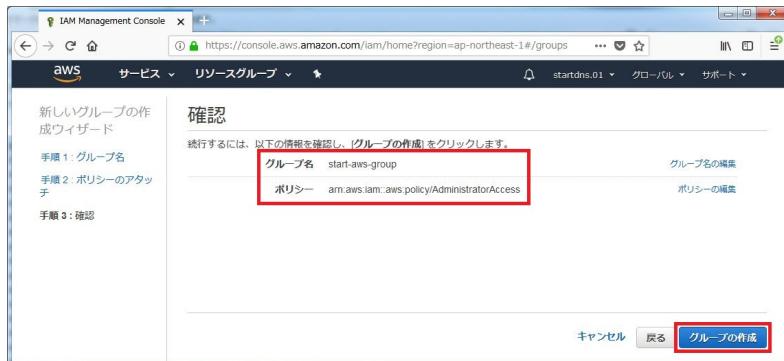


▲図2.23 「AdministratorAccess」にチェックを入れて「次のステップ」をクリック

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2.24）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 2.24 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 2.25）されたらグループの作成は完了です。



▲図 2.25 「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 2.26）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 2.26 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 2.27) です。IAM のユーザー名を入力してください。本著では IAM のユーザー名は「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 2.27 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 2.28）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 2.28 「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 2.29）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 2.29 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 2.30)



▲図 2.30 「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 2.31) とパスワード (図 2.32) 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 2.31 URL の数字をメモしておこう

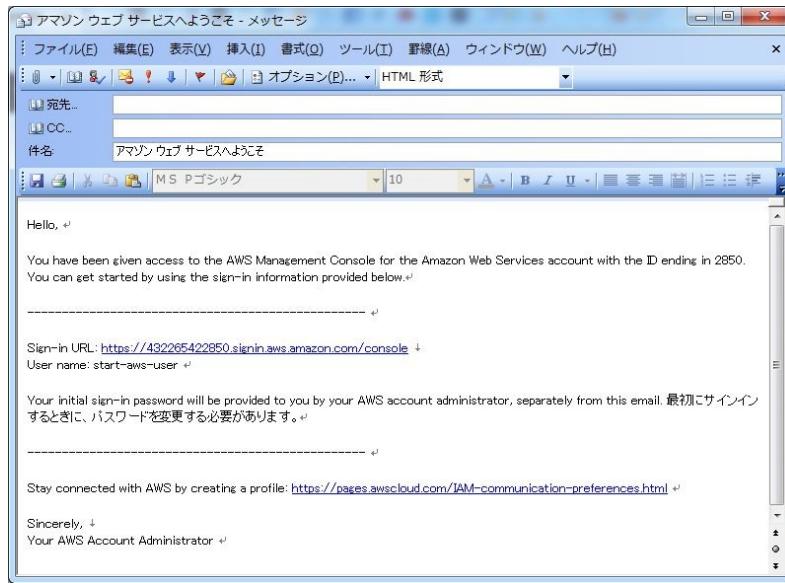


▲図 2.32 パスワードの「表示」をクリックして、パスワードもメモしておこう

▼表 2.1 IAM ユーザの情報

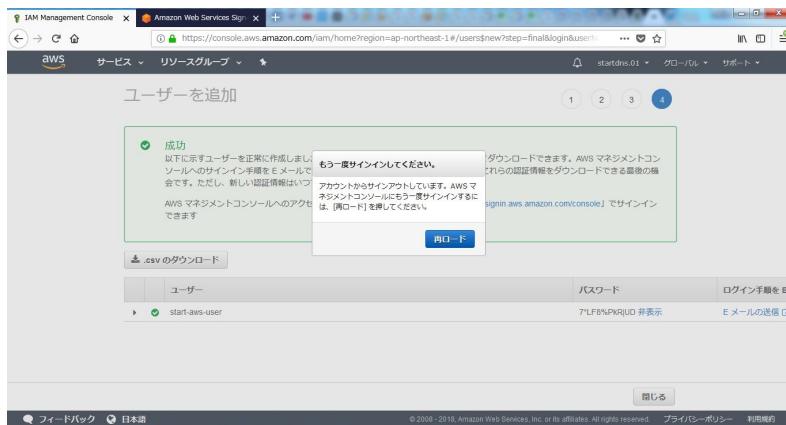
IAM ユーザー情報	例	自分の IAM ユーザー情報
AWS アカウント (URL の数字)	432265422850	
ユーザー名	start-aws-user	
パスワード	7*LF8%Pkr UD	

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 2.33）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。



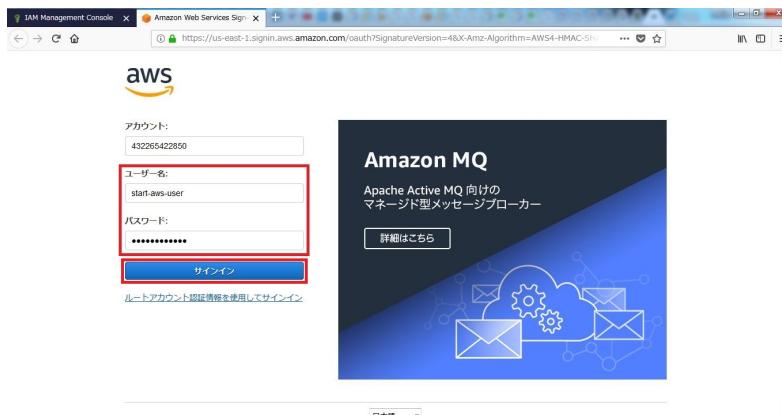
▲図 2.33 サインイン URL や IAM ユーザー名をメールで自分自身宛てに送っておこう

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 2.34）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。



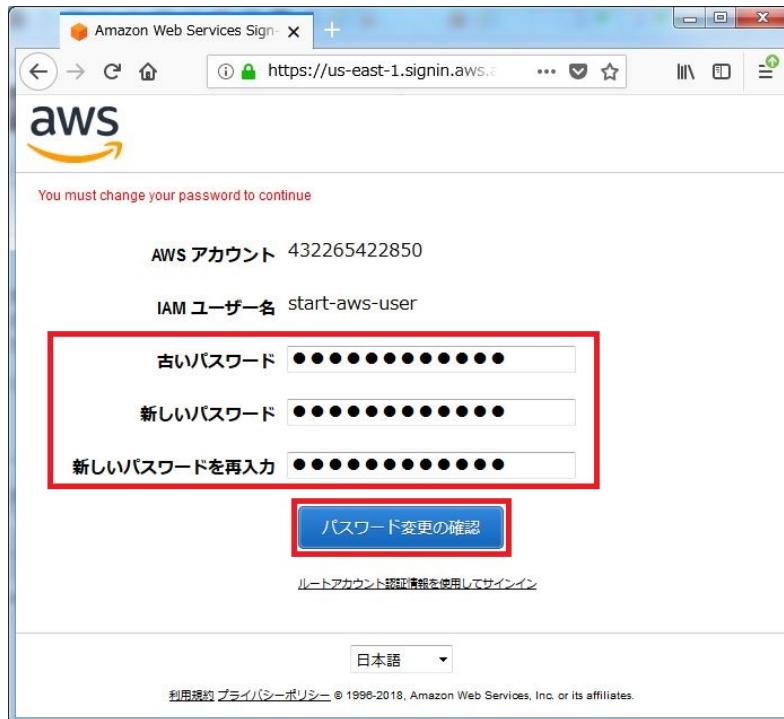
▲図 2.34 ルートユーザーでサインインしていた画面は自動的にサインアウトされる

別タブで開いた IAM ユーザーのサインイン画面（図 2.35）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 2.35 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 2.36）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。



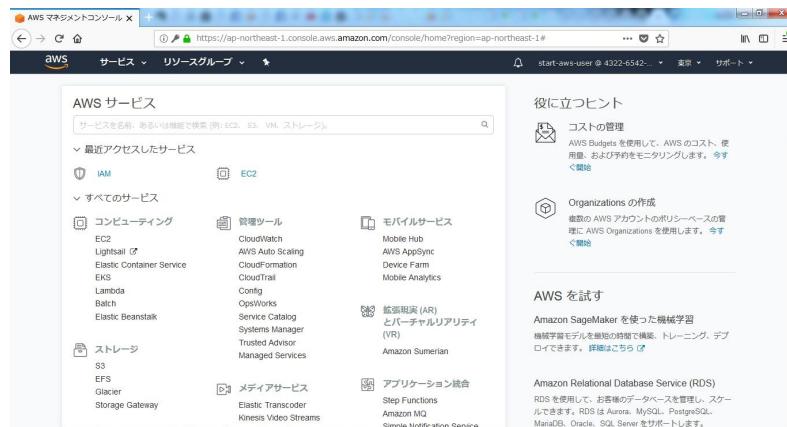
▲図 2.36 新しいパスワードを入力して「パスワード変更の確認」をクリック

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 2.37）が表示されます。皆さんもサインインできましたか？



▲図 2.37 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 2.38）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。



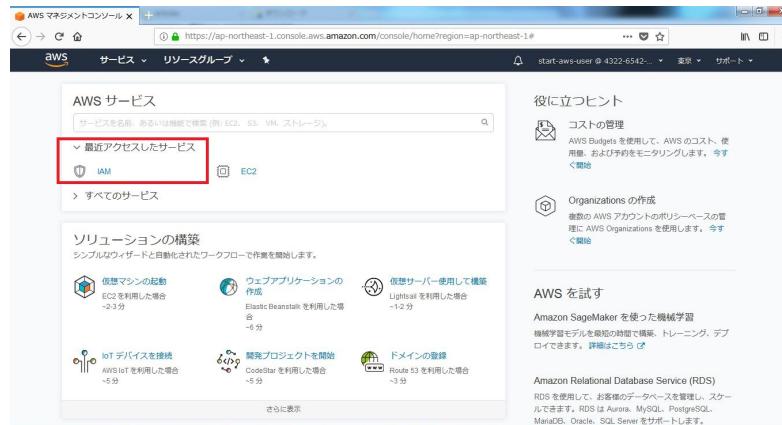
▲図 2.38 IAM ユーザーでサインインしていることを確認

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。

2.5.3 MFA (多要素認証) で不正利用から IAM ユーザーを守る

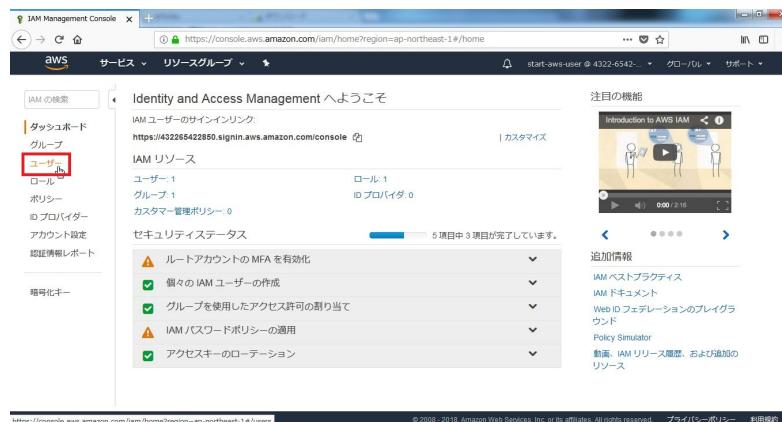
無事に IAM ユーザーでサインインできたら、再び左上にある「サービス」から「IAM」をクリックしてください。もしくは「最近アクセスしたサービス」から「IAM」をクリック

ク（図2.39）でも構いません。



▲図2.39 「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2.40）します。



▲図2.40 左メニューの「ユーザー」をクリック

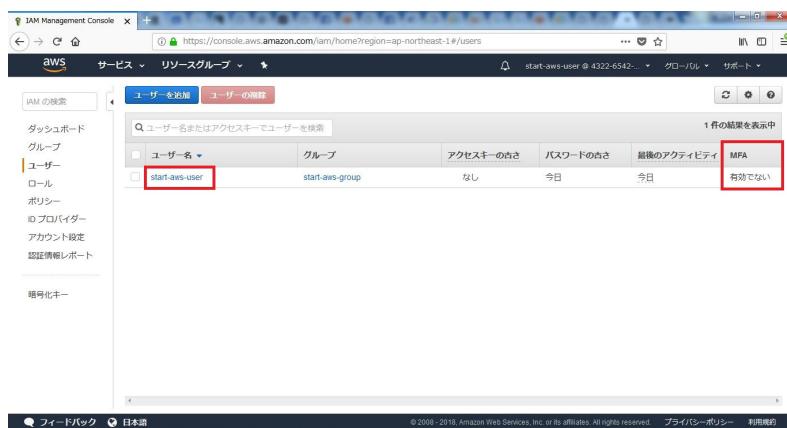
現状は、

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 2.41）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 2.41 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 2.42）します。

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



▲図 2.42 「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 2.43）^{*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 2.43 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 2.44）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 2.44 「仮想 MFA デバイス」を選択して「次のステップ」をクリック

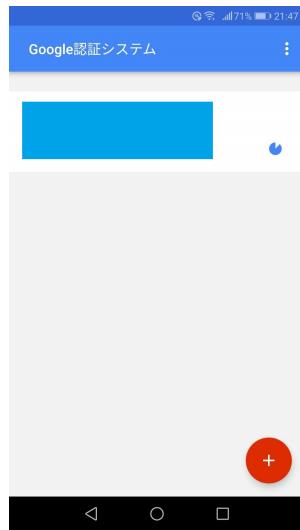
お手元のスマホに「Google 認証システム（Google Authenticator）」をインストール^{*11}したら「次のステップ」をクリック（図 2.45）します。



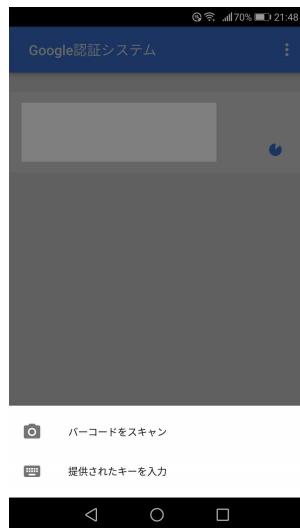
▲図 2.45 スマホに認証アプリをインストールしたら「次のステップ」をクリック

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 2.46）して、「バーコードをスキャン」を選択（図 2.47）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。



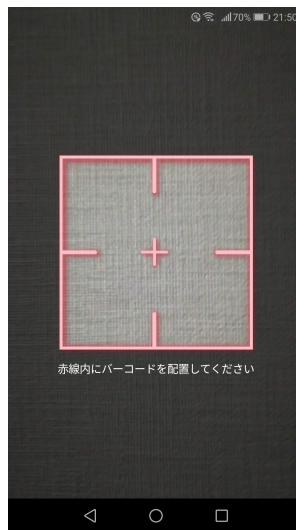
▲図 2.46 「Google 認証システム」のアプリを起動したら右下の赤い+ボタンをタッチ



▲図 2.47 「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 2.48) 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 2.49)がス

マホの赤枠内に収まるようにします。

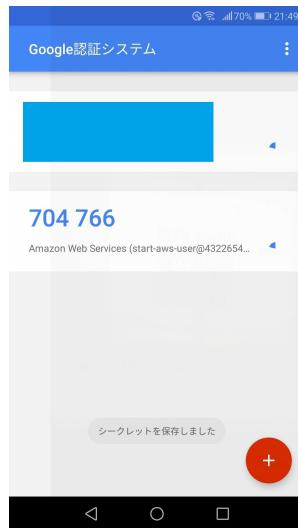


▲図 2.48 「赤線内にバーコードを配置してください」と表示される



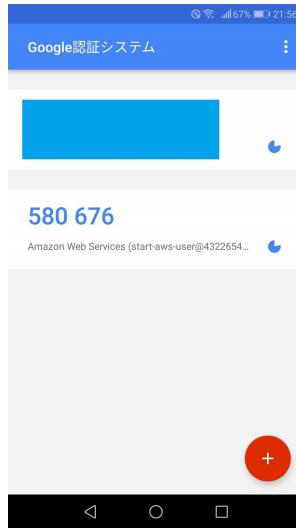
▲図 2.49 マネジメントコンソールに表示されたバーコードをスマホで撮る

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 2.50）されるようになります。



▲図 2.50 「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 2.51）。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 2.52）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。

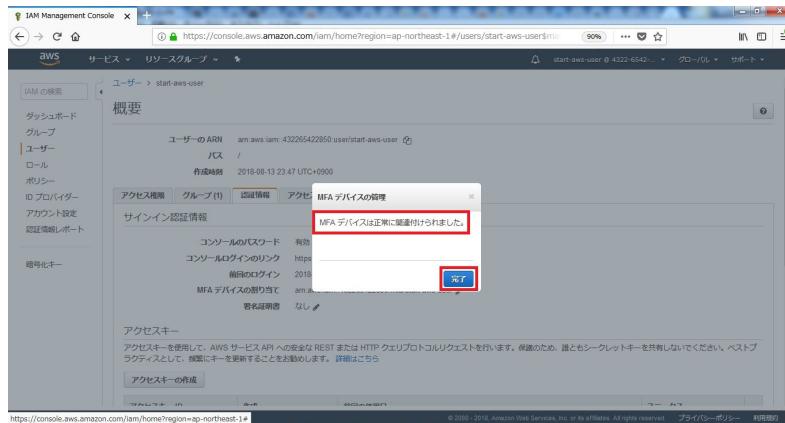


▲図 2.51 認証コードは 30 秒ごとに次々と切り替わる



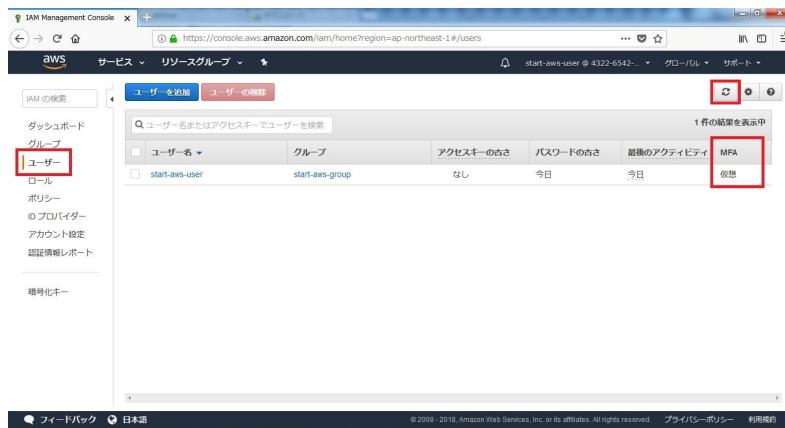
▲図 2.52 「認証コード 1」と「認証コード 2」を入力して「仮想 MFA の有効化」をクリック

「MFA デバイスは正常に関連付けられました。」と表示（図 2.53）されたら「完了」をクリックしてください。



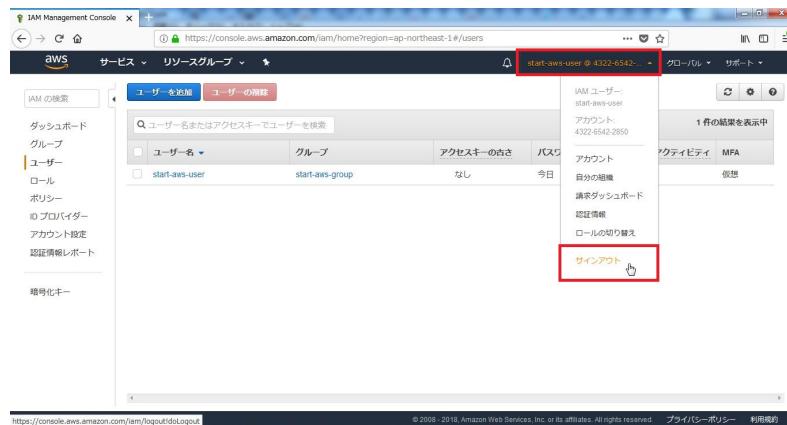
▲図 2.53 「MFA デバイスは正常に関連付けされました。」と表示されたら「完了」をクリック

再び左メニューの「ユーザー」をクリック（図 2.54）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 2.54 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 2.55）します。



▲図 2.55 「サインアウト」をクリック

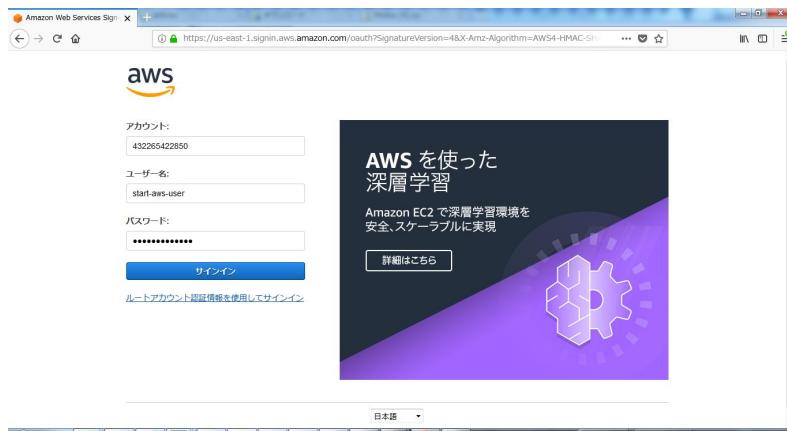
右上の「コンソールへログイン」をクリック（図 2.56）します。



▲図 2.56 「コンソールへログイン」をクリック

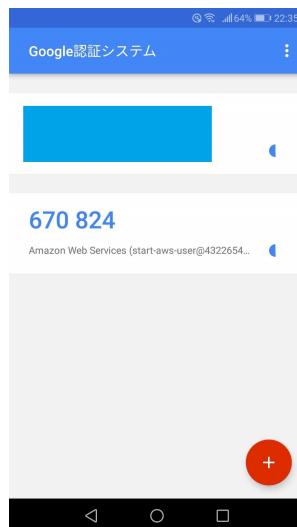
1. アカウント（12桁の数字）
2. ユーザー名
3. パスワード

の3つを入力したら「サインイン」をクリックします。

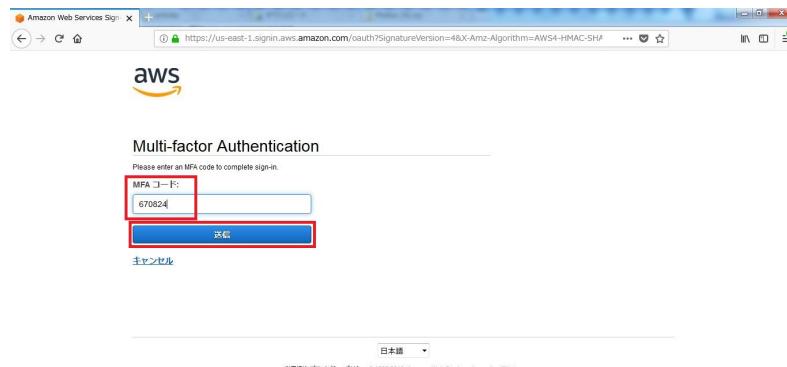


▲図 2.57 「サインイン」をクリック

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 2.58）して、表示されている認証コードを「MFA コード」（図 2.59）のところへ入力したら「送信」をクリックしてください。



▲図 2.58 スマホで「Google 認証システム」を起動

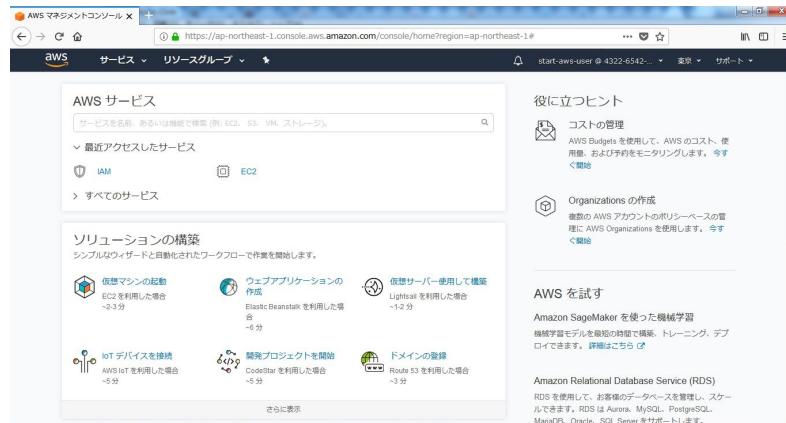


▲図 2.59 認証コードを「MFA コード」に入力して「送信」をクリック

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。

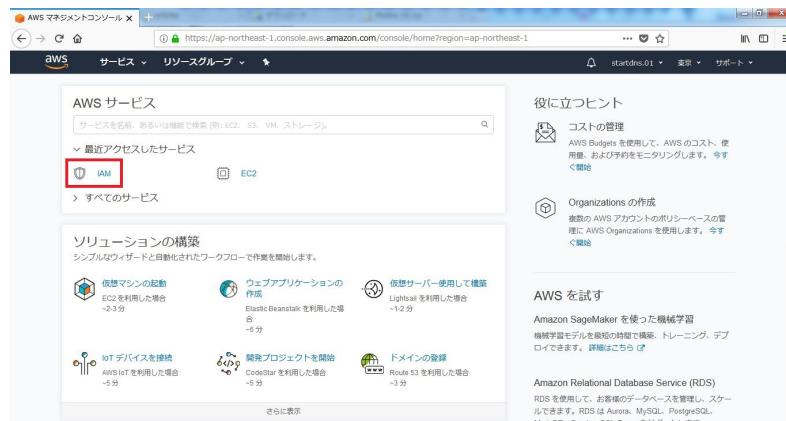


▲図 2.60 MFA を用いたサインインができるようになった！

2.5.4 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 2.61）して IAM のダッシュボードを開きます。



▲図 2.61 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

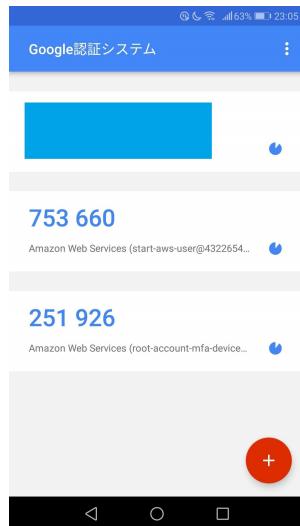
IAM のダッシュボードで「ルートアカウントの MFA を有効化」(図 2.62)*13を開いて「MFA の管理」をクリックします。



▲図 2.62 「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 2.63）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。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。

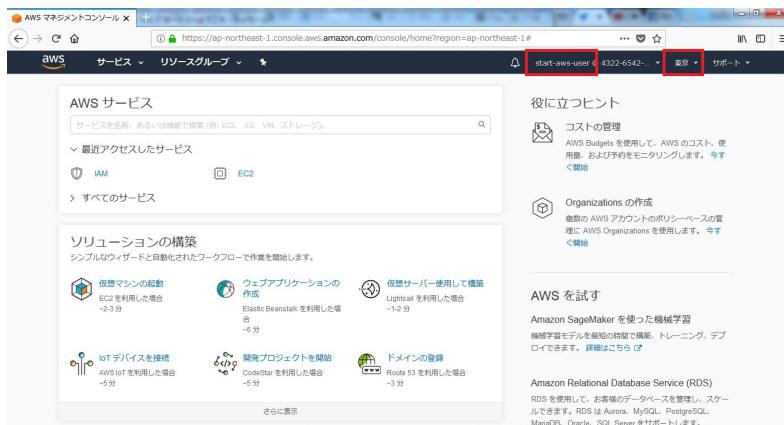


▲図 2.63 ルートアカウント用の認証コード（6桁の数字）も表示されている

ルートアカウントでも MFA が有効にならたら、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ってしまったら、先ほど送っておいたメールに「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。

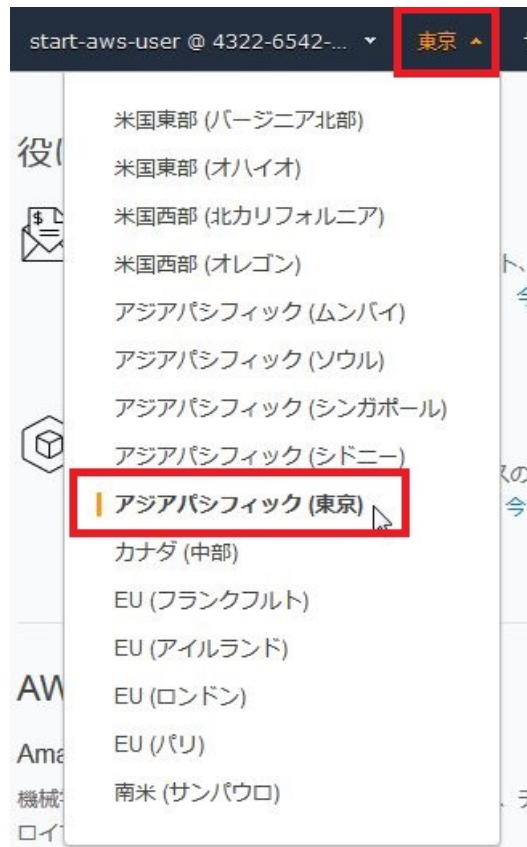
2.6 リージョンの変更

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ってこられましたか？



▲図 2.64 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 2.64）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 2.65）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 2.65 「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

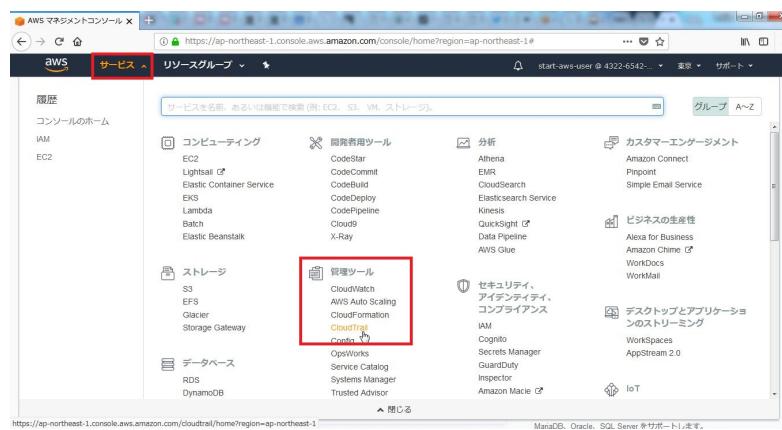
さらにリージョンという地域ごとの区分の下に、さらにアベイラビリティゾーン^{*15}という区分があります。アベイラビリティゾーンの場所は公開されていませんが、たとえば東京リージョンの下に「池袋アベイラビリティゾーン」と「品川アベイラビリティゾーン」がある、というようなイメージです。

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。

2.7 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 2.66）をクリックしてください。



▲図 2.66 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 2.67）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づつ異なります。

*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 sidebar with 'CloudTrail' and three main menu items: 'ダッシュボード' (Dashboard), 'イベント履歴' (Event History) which is highlighted with a red box, and '証跡情報' (Audit Trail Information). The main content area has a title 'CloudTrail へようこそ' (Welcome to CloudTrail) and a sub-section '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、証跡を作成します。証跡により、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 for your AWS account. To keep records of these events, create a trail. Trails allow you to create event streams, trigger alerts, and create event work flows.). Below this is a table titled '最近のイベント' (Recent Events) with the following data:

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). The right side of the dashboard has a sidebar with links like '料金表' (Billing), 'トライアル' (Trial), 'フォーム' (Forms), and 'よくある質問' (FAQ).

▲図 2.67 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 2.68）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。

2.7 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 2.68 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 でサーバを立てよう

この章では実際に AWS の EC2 を使ってサーバを立てます。

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。

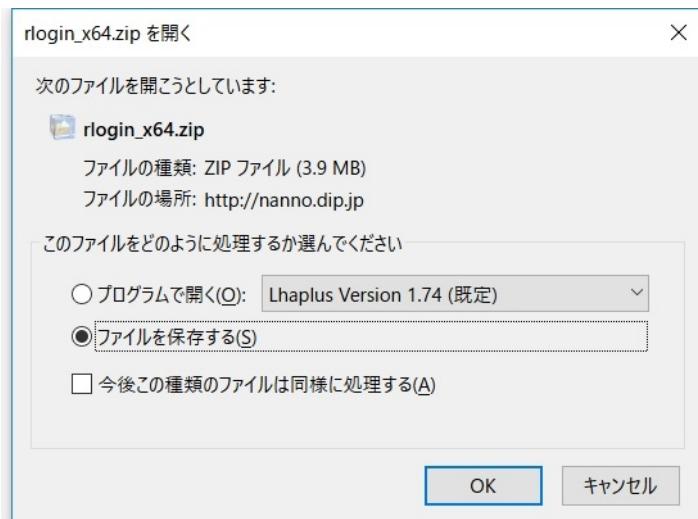
^{*1} PuTTYjp を使う場合、.pem の秘密鍵を PuTTYgen で.ppk に変換する必要があります。他のターミナルソフトに比べると一手間余計にかかるので、特にこだわりがなければ RLogin を使用するのがお勧めです。

^{*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

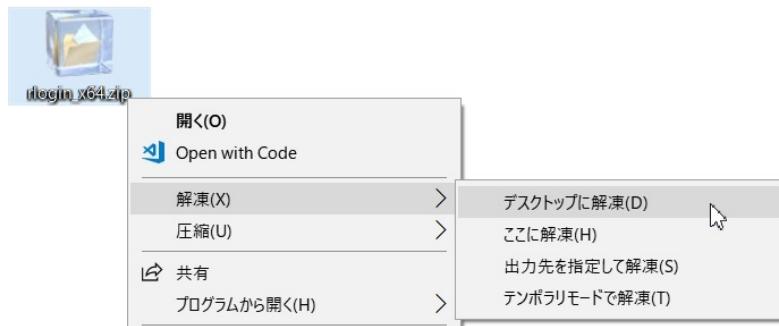
ダウンロードした ZIP ファイルを保存（図 3.3）します。保存場所はどこでも構いませんが、後でどこに置いたか分からなくなりそうな人はデスクトップに保存しておきましょう。



▲図 3.3 「ファイルを保存する」でパソコンに保存

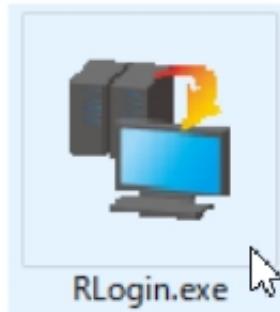
デスクトップの ZIP ファイル (rlogin_x64.zip) を右クリック（図 3.4）して、解凍>デスクトップに解凍^{*3}をクリックします。

^{*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/>



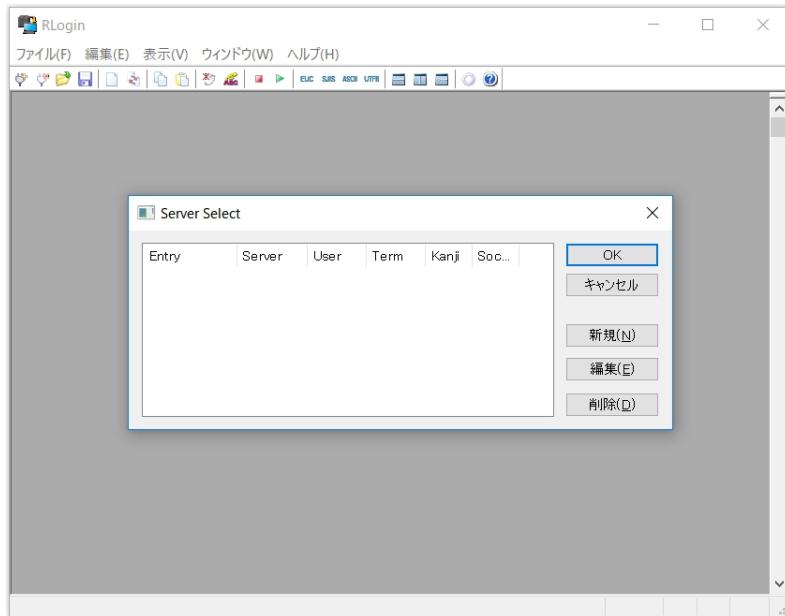
▲図3.4 ZIPファイルを右クリックして解凍>デスクトップに解凍

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。

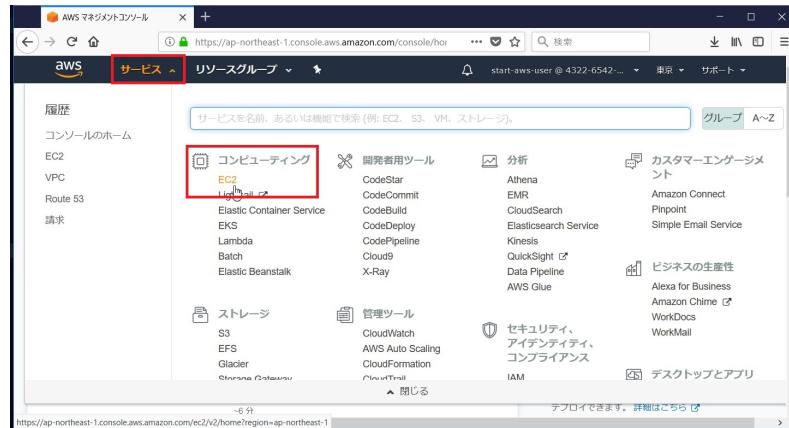
3.2 EC2 でサーバを立てる

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

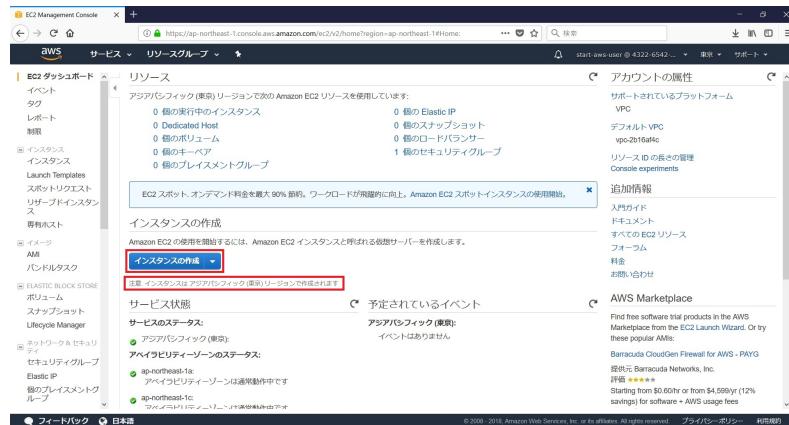
ちなみにAWSではサーバのことを、**インスタンス**と呼びます。ここから先でインスタンスと書いてあつたらサーバのことだと思ってください。

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。



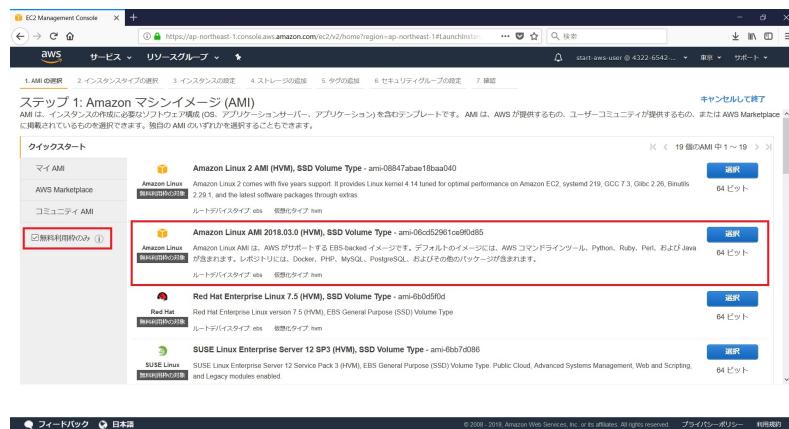
▲図3.8 EC2ダッシュボードで「インスタンスの作成」をクリック

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。

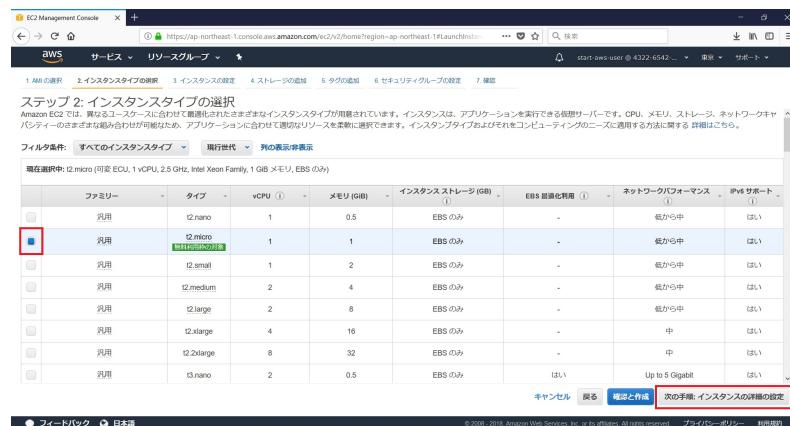
Amazon Linuxには2018年8月時点で

- 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3.10)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 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

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

ステップ 3 のインスタンスの詳細（図 3.11）は、すべてデフォルト設定のままで構いません。そのまま「次の手順: ストレージの追加」をクリックしてください。



▲図 3.11 何も変更せず「次の手順: ストレージの追加」をクリック

3.2.4 ステップ 4: ストレージの追加

ステップ 4 ではサーバにくっつけるストレージの容量を設定（図 3.12）します。ストレージというのはデータを保存しておく場所のことです。皆さんのパソコンにも「ハードディスク」というストレージがついていて、作ったファイルはそこに保存していますよね。

Amazon には EBS^{*9}というストレージサービスがあり、ここでは「EC2 のサーバにくっつける EBS ボリュームの種類やサイズはどうしますか？」と聞かれています。

^{*9} EBS は Amazon Elastic Block Store の略で、EC2 向けのストレージサービスのことです。



▲図 3.12 何も変更せず「次の手順: タグの追加」をクリック

下部に「無料利用枠の対象であるお客様は 30GBまでのEBS汎用(SSD)ストレージまたはマグネティックストレージを取得できます。」と書いてあるとおり、無料利用枠でEBSは最大30GBまで使えますがデフォルトの8GBのままで十分なので変更は不要です。「次の手順: タグの追加」をクリックしてください。

3.2.5 ステップ5: タグの追加

ステップ5ではインスタンスにタグを付けて分類(図3.13)できます。タグにはキーと値があり、たとえばキーが「Environment (環境)」のタグを作って、インスタンスによって値を「production (本番)」や「staging (ステージング)」や「Develop (開発)」にすることで、どれが本番のサーバでどれがステージングのサーバなのか区別できるようになります。



▲図 3.13 タグは作成せず「次の手順: セキュリティグループの設定」をクリック

今回はサーバは 1 台しか立てないので、タグはつけずに先へ進みましょう。「次の手順: セキュリティグループの設定」をクリックしてください。

3.2.6 ステップ 6: セキュリティグループの設定

ステップ 6 ではセキュリティグループの設定（図 3.14）を行います。セキュリティグループというのはいわゆる「ファイアウォール」のことで「自宅からのアクセスは通すけどそれ以外からは通さない」のように特定の通信のみを通してそれ以外は阻止することで、文字通り防火壁となってサーバを守ってくれます。

上部の「セキュリティグループの割り当て」が「新しいセキュリティグループを作成する」になっていることを確認したら、次の 2 つを入力^{*10}してください。

1. セキュリティグループ名 : ec2-security-group
2. 説明 : Allow from anywhere

^{*10} セキュリティグループ名や説明には日本語は使えませんので注意してください。



▲図3.14 セキュリティグループ名と説明を変更したら「確認と作成」をクリック

続いて「ここからのアクセスのみを通す」というルールを確認します。下部に「送信元が0.0.0.0/0のルールを指定すると、すべてのIPアドレスからインスタンスにアクセスすることができます。」という警告が出ているとおり、デフォルトのルールは「SSH^{*11}での接続はどこからでも通す」という設定になっています。^{*12}このままで構いませんので「確認と作成」をクリックしてください。

3.2.7 ステップ7: インスタンス作成の確認

ステップ7は「この内容でインスタンスを作成しますよ？問題ないですか？」という確認の画面（図3.15）です。表示されている内容で問題なければ「作成」をクリックしてください。

^{*11} SSHという単語が突然出てきましたが、SSHについては後述します。

^{*12} サーバにどこからでも接続できてしまうのはよくないので、できればSSHでのアクセス元も制限したいのですが、ここで制限をかけると「出先でモバイルWi-Fiに繋いだとき」などに接続元のIPアドレスが変わってファイアウォールで阻止され、あなたの自身もサーバに接続できなくなってしまいます。今自宅にいて今後も自宅からしかサーバにSSHで接続しない！自宅のIPアドレスは固定だから変わらない！という人だけ、ルールのソースを「マイIP」にして説明に「IP address of my house」と書いておいてください。こうするとあなたの自宅のIPアドレスからしかSSHで接続できなくなります。



▲図 3.15 問題なければ「作成」をクリック

3.2.8 キーペアのダウンロード

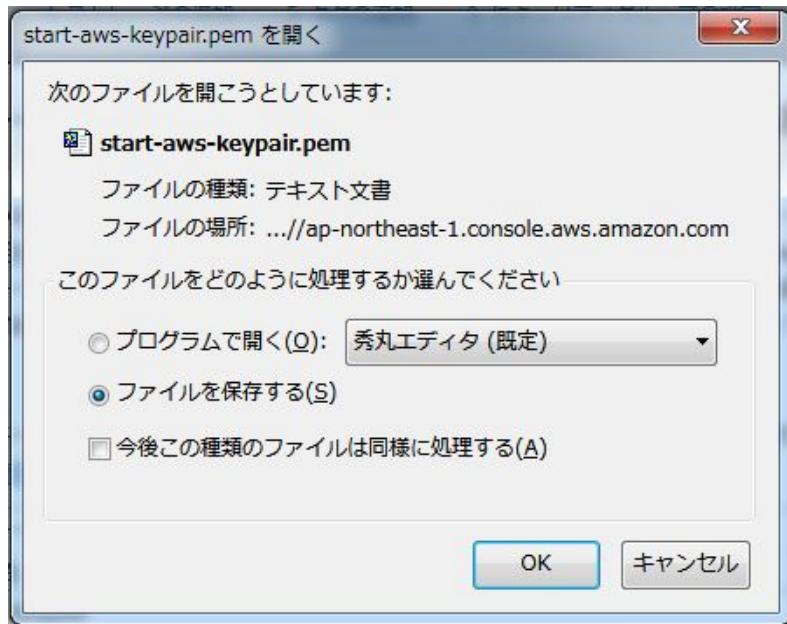
「作成」をクリックすると「既存のキーペアを選択するか、新しいキーペアを作成します。」と表示（図 3.16）されます。「新しいキーペアの作成」を選んでキーペア名を「start-aws-keypair」にしてください。



▲図 3.16 キーペア名を「start-aws-keypair」にして「キーペアのダウンロード」をクリック

キーペアとはサーバに入るための鍵と鍵穴のペアのことです。サーバのドアに鍵穴を設置してパソコンに鍵を保存することで、鍵を持っている人しかサーバに入れなくなりま

す。キーペア名を入力したら「キーペアのダウンロード」をクリック（図3.17）してください。そしてダウンロードしたキーペア（start-aws-keypair.pem）はパソコンのデスクトップなど絶対に忘れない場所に保存（図3.18）しておいてください。



▲図3.17 ダウンロードしたキーペア（start-aws-keypair.pem）を保存



▲図 3.18 デスクトップなど絶対に忘れない場所に保存しておくこと

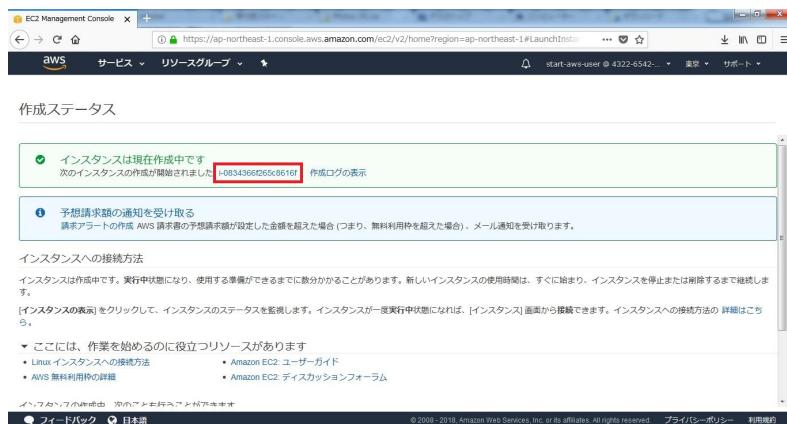
このキーペア (start-aws-keypair.pem) はこれ以降2度とダウンロードできません。家の鍵と同じで、キーペアをなくしてしまうとこの後サーバに入ろうとしたときに「鍵がない！ 入れない！」となります。絶対になくさないでください。

キーペアをパソコンに保存したら「インスタンスの作成」をクリック（図 3.19）してください。



▲図 3.19 キーペアをパソコンに保存したら「インスタンスの作成」をクリック

「インスタンスの作成」をクリックすると、作成ステータスの画面で「インスタンスは現在作成中です」(図 3.20) と表示されます。



▲図 3.20 「インスタンスは現在作成中です」と表示されたらインスタンス ID をクリック

「i-0834366f265c8616f」のようなインスタンス ID をクリックして EC2 ダッシュボード

に戻りましょう。

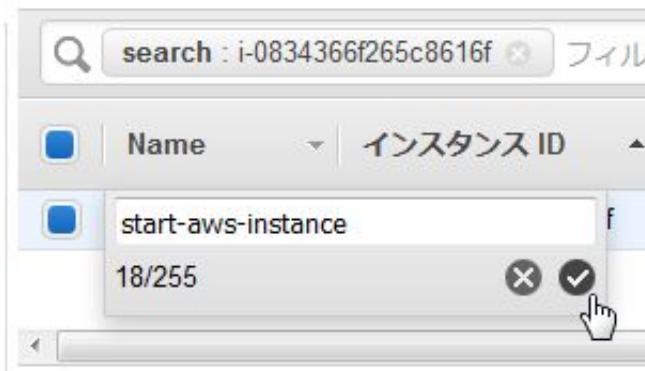
3.2.9 作成したインスタンスに名前をつける

作成したインスタンスが表示（図 3.21）されていますので、このインスタンスに名前を付けておきましょう。Name のところにカーソルを持っていくと鉛筆のマークが表示されますのでクリックしてください。



▲図 3.21 作成したインスタンスの Name のところにある鉛筆マークをクリック

Name に「start-aws-instance」と書いたらチェックボタンを押して（図 3.22）ください。インスタンス名には日本語を使わないことをお勧めします。



▲図 3.22 Name に「start-aws-instance」と書いたらチェックボタンを押す

- Name が「start-aws-instance」になっていること
- インスタンスの状態が「running」になっていること

を確認（図3.23）したらインスタンスの作成は完了です。^{*13}

おめでとうございます！あなたは無事に「サーバを立てた」のです！それでは自分で立てたサーバに入行ってみましょう。



▲図3.23 インスタンスの作成完了

3.2.10 【ドリル】秘密鍵をなくしたらどうなる？

問題

ある日パソコンが壊れてしまい、パソコン上に保存してあったキーペアこと「start-aws-keypair.pem」も消滅してしまいました。キーペアはサーバへSSHでログインするときに必要なのですが、キーペアがなくなってしまったのであなたはとても困っています。どうしたらまたサーバにログインできるようになるのでしょうか？

- A. キーペアをマネジメントコンソールのEC2ダッシュボードから再ダウンロードすればよい
- B. 再ダウンロードはできないのでEC2ダッシュボードからキーペアを再発行すればよい
- C. キーペアをなくしたら二度とサーバにはログインできない

^{*13}もしインスタンスの状態が「running」以外だった場合は、少し待ってから右上の更新マークをクリックしてみてください。

答え _____

解答

正解は C です。キーペアは 1 回きりのダウンロードで再発行もできません。壊れたパソコンからキーペアを持ち出せない場合、そのサーバには二度と SSH でログインできません。残念ですが新しくインスタンスを作り直すしか方法はありません。

【コラム】サーバは「立てる」もの？「建てる」もの？

サーバは「立てる」ものでしょうか？ それとも「建てる」ものでしょうか？

英語だとサーバ構築を「server build」と表現しますし「築く」「建築する」という意味では「建てる」が正しい気もします。

一方、パソコンが起動することをよく「立ち上がる」と言いますし、色々な設定をしてやった我が子のようなサーバがようやく「立った」という意味では、「立てる」でいいような気もします。

個人的には、儀同世津子がパソコンのスイッチをつけて「立ち上がろうとしてるところ」と言ったら、瀬戸千衣は「立ち上がるの？」と眉を寄せた^aので私は「立てる」派です。

^a 森博嗣の「封印再度」（講談社文庫）549 ページより引用。

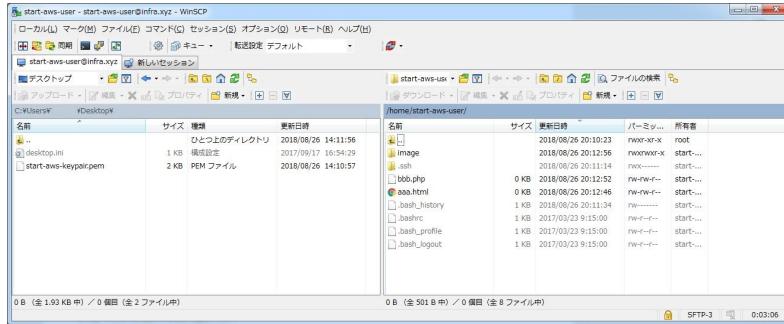
3.3 サーバに「入る」とは？

無事に EC2 でインスタンスが作成できたので、いよいよサーバに入ってみたいのですが・・・ところでサーバに「入る」といわれてもピンときますか？ サーバに「入る」って、いったいどういうことでしょう？

- ファイルをアップするために WinSCP や Cyberduck でサーバに「接続する」
- ログを見るためにターミナルでサーバに「ログインする」

この 2 つはどちらもサーバに「入る」という行為です。

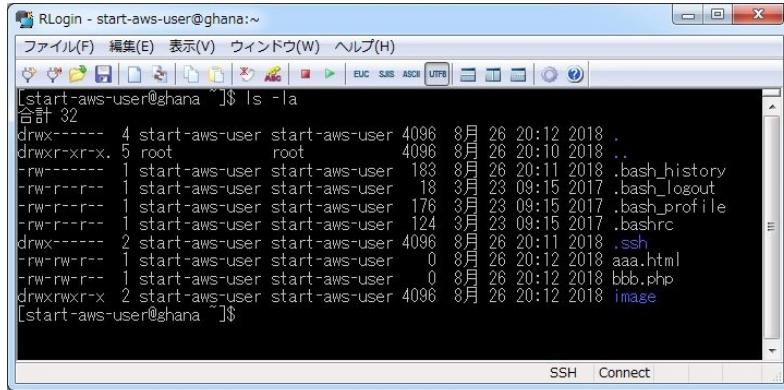
「今までサーバになんか入ったことない！」という方でも、こんな画面（図3.24）を見たことはありませんか？



▲図3.24 WinSCPでサーバに「入る」

これは「FTPクライアント」という種類のソフトで、サーバへ画像やHTMLファイルなどをアップロードするときによく使われます。左側が自分が使っているパソコンで、右側がサーバに「入って」います。

もう1つ、こんな画面（図3.25）も見たことはありませんか？



▲図3.25 ターミナルでサーバに「入る」

こちらは「ターミナル」という種類のソフトで、サーバでログの確認をしたり設定ファイルを書き換えたりするときに使われます。エンジニア以外の人には、いわゆる「黒い画面」と言った方がお馴染みかもしれません。

ところでFTPクライアントの右側とターミナルを見てみてください。どちらも

「image」というフォルダがあって、「aaa.html」と「bbb.php」というファイルがありますよね。実はこの2つ、どちらも同じサーバに入って同じフォルダを見ています。

実際の画面を目にしたことで、少しばかり「入る」という言葉のイメージがつきましたでしょうか？

3.3.1 SSH とは？

EC2 のインスタンスを立てる途中、セキュリティグループが「SSH での接続はどこからでも通す」という設定になっていたのを覚えていますか？

ここでこの「SSH」について少し説明をします。SSH とは Secure Shell の略で、データセンターにあるサーバと自分が使っている目の前のパソコンをセキュアに繋いでくれるサービスのことです。

第1章「インフラとサーバってなに？」で「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」という説明をしました。聞きなれないかも知れませんが、実は「SSH が動いているサーバ」のことを SSH サーバと呼びます。

ビアサーバは客に対してビールを提供するもので、ウェブサーバは客に対してウェブページを提供するものでしたが、SSH サーバはいったい何のサービスを提供してくれるのでしょうか？

SSH は「サーバに入れて」とリクエストされたら、その人が誰なのか確認した上でサーバに入ってくれます。つまり SSH サーバは、サーバへアクセスしてきた客に対して「ネットワークを介してサーバにログインできる」というサービスを提供しているのです。SSH があることによってデータセンターにあるサーバと自分が使っている目の前のパソコンをセキュアに繋ぐことができるのです。

SSH と似ている仕組みに「FTP」^{*14}や「Telnet」^{*15}があります。定番 FTP クライアントである「FFFTP」なら使ったことがある、という方もいるのではないでしょうか。

この FTP や Telnet では通信内容が暗号化されずにそのまま送られるため、通信経路を盗聴すればアカウントやパスワードは丸見えになってしまい、という問題点がありました。自分のパソコンからサーバまでの道のりを通信データが丸裸で流れていってしまうようなイメージです。

それに対して SSH はデータを暗号化した上でやり取りできます。きちんと全身タイツを着て正体が見えない状態で通信データが流れしていくので、たとえ盗聴されても暗号化する前のデータがどんなものだったのかはすぐには分かりません。そのためサーバにファイルをアップロードしたりダウンロードしたりするときには、この SSH の機能を用いた

*14 File Transfer Protocol の略。

*15 Telecommunication network の略。

「SFTP」^{*16}や「SCP」^{*17}というファイル転送の仕組みがよく使われます。このような理由からサーバにはSSHで入ります。

3.3.2 パスワード認証と鍵認証

サーバにSSHで入るときには、そのユーザが本人かどうかを確認する方法として「パスワード認証」や「鍵認証」が用いられることが多いです。

パスワード認証ではユーザ名とパスワードを使ってサーバに入ります。鍵認証の場合はあらかじめサーバに鍵穴を設置しておいて、パソコンの中にある鍵を使ってサーバに入ります。鍵穴と鍵^{*18}のペアなので「キーペア」と呼ばれているのですね。この鍵というのが皆さんのが先ほどダウンロードした「start-aws-keypair.pem」のことなのです。

先ほど作ったインスタンスにはすでに鍵穴は設置されているので、鍵(start-aws-keypair.pem)を使えばサーバに入ることができます。

3.3.3 接続先となるサーバのIPアドレス

サーバに「入る」ということのイメージが付いたところで、接続先となるサーバのIPアドレスを確認してみましょう。

先ほど作成したインスタンスの説明(図3.26)の中にある「IPv4パブリックIP」をメモ(表3.1)してください。

*16 SSH File Transfer Protocolの略。

*17 Secure Copyの略。

*18 鍵穴は「公開鍵」あるいは「パブリックキー」、鍵は「秘密鍵」あるいは「プライベートキー」と呼ばれます。



▲図 3.26 「説明」タブの「IPv4 パブリック IP」をメモ

▼表 3.1 インスタンスの IPv4 パブリック IP をメモ

例	IPv4 パブリック IP
	13.230.112.72

「IPv4 パブリック IP」のところへカーソルを持っていくと「クリップボードにコピー」と表示（図 3.27）されますので、パソコンの中のメモ帳などにもメモしておくと便利です。



▲図 3.27 パソコンの中のメモ帳などにもメモしておく

それではメモした IP アドレスを使ってサーバに入ってみましょう。

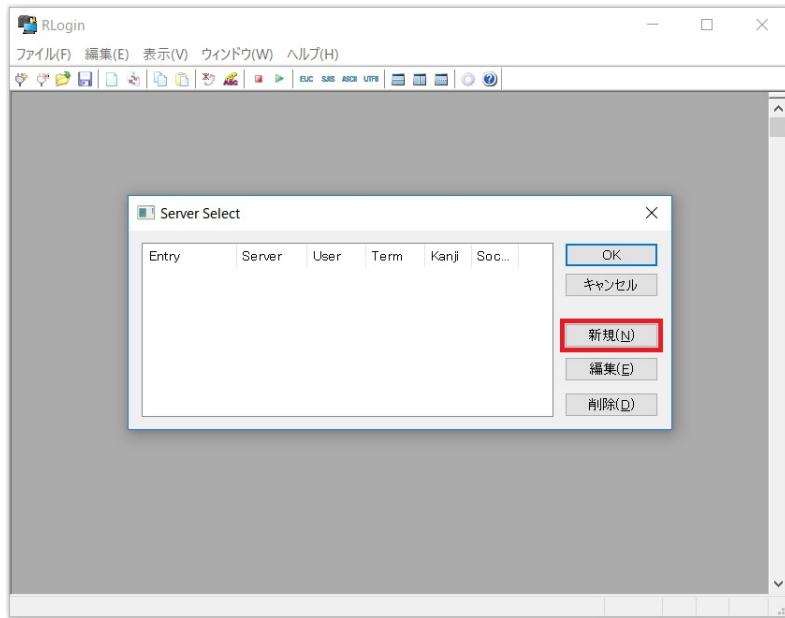
3.4 SSHでサーバに入ってみよう

3.4.1 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、デスクトップの「rlogin_x64」というフォルダの中にある「RLogin.exe」(図3.28)をダブルクリックして RLoginを起動(図3.29)してください。起動したら「新規」をクリックします。

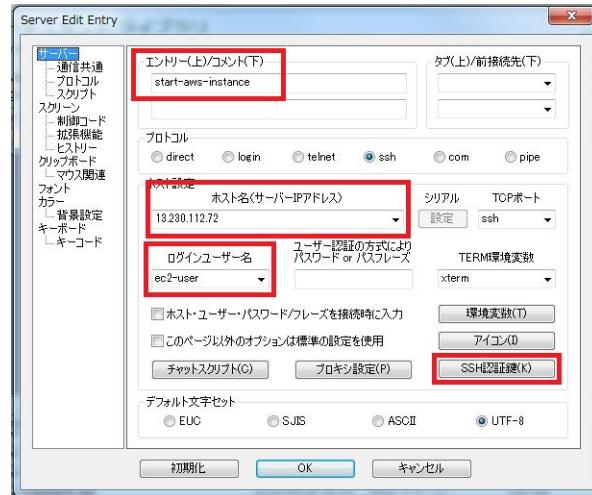


▲図3.28 RLogin.exeをダブルクリック



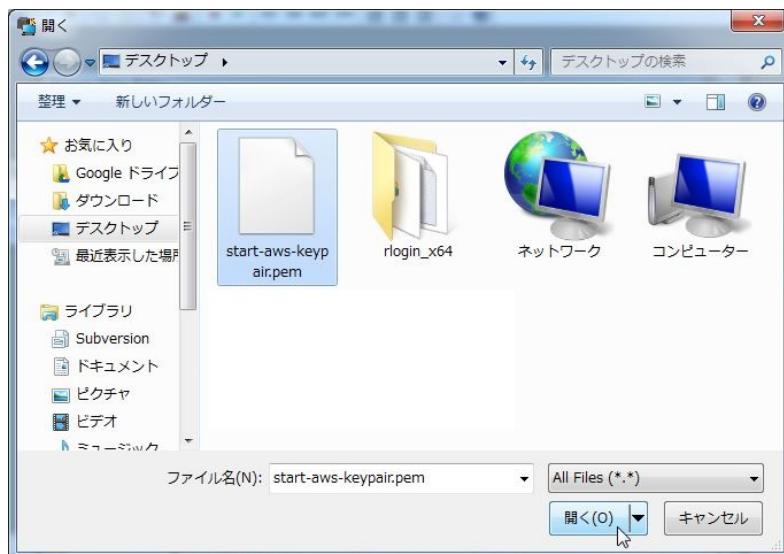
▲図 3.29 RLogin が起動したら「新規」をクリック

初めに「エントリー（上）/コメント（下）」に「start-aws-instance」と入力します。続いて「ホスト名（サーバー IP アドレス）」に先ほどメモした「IPv4 パブリック IP」を入力（図 3.30）します。「ログインユーザー名」には「ec2-user」と入力してください。ec2-user というのはインスタンスを作成すると最初から存在しているユーザです。



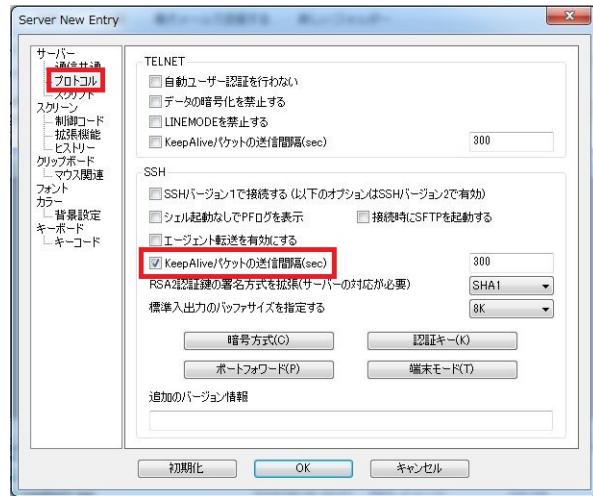
▲図 3.30 「ホスト名（サーバー IP アドレス）」と「ログインユーザー名」を入力

続いて「SSH 認証鍵」をクリック（図 3.31）して、デスクトップなど絶対に忘れない場所に保存しておいた「start-aws-keypair.pem」を選択したら「開く」をクリックします。



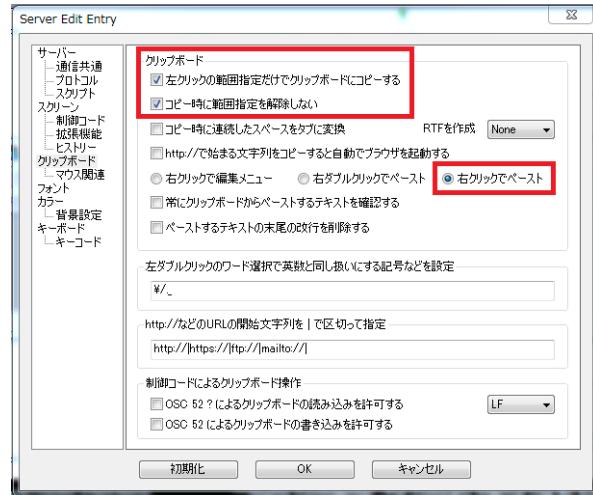
▲図 3.31 「SSH 認証鍵」をクリックして「start-aws-keypair.pem」を選択

次に左メニューで「プロトコル」を選択（図 3.32）したら、「KeepAlive パケットの送信間隔(sec)」にチェックを入れておきます。これを設定しておくとターミナルをしばらく放っておいても接続が勝手に切れません。



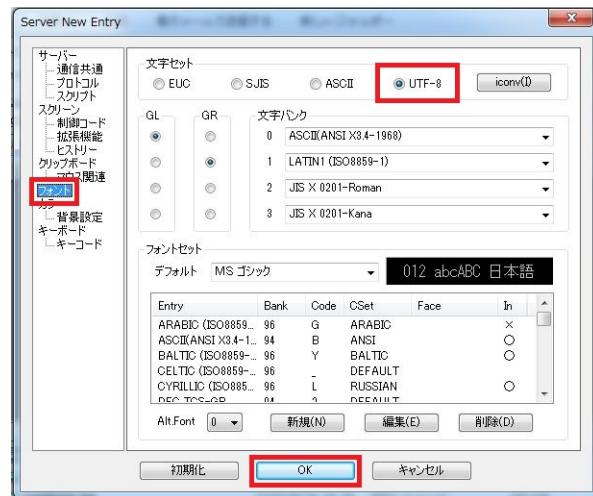
▲図 3.32 「KeepAlive パケットの送信間隔(sec)」にチェックを入れる

続いて左メニューで「クリップボード」を選択（図 3.33）したら、「左クリックの範囲指定だけでクリップボードにコピーする」と「コピー時に範囲指定を解除しない」にチェックを入れて「右クリックでペースト」を選択します。



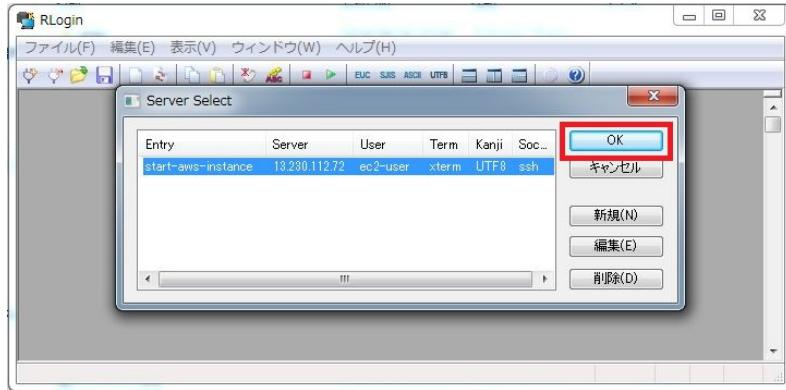
▲図 3.33 右クリックや左クリックの設定

次に左メニューで「フォント」を選択（図 3.34）したら、文字セットを「UTF-8」に変更します。すべて設定できたら「OK」をクリックしてください。



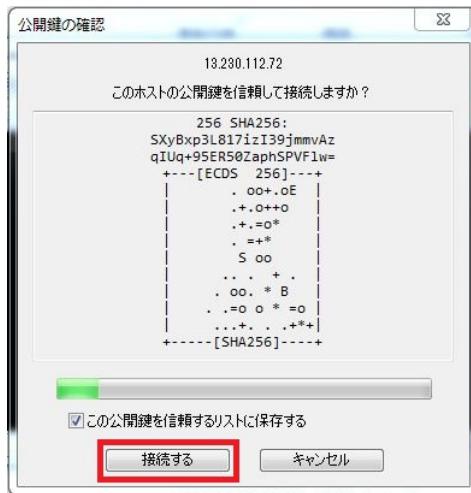
▲図 3.34 文字セットを「UTF-8」に変更

設定が保存できたら「OK」をクリック（図 3.35）してください。



▲図 3.35 設定が保存できたら「OK」をクリック

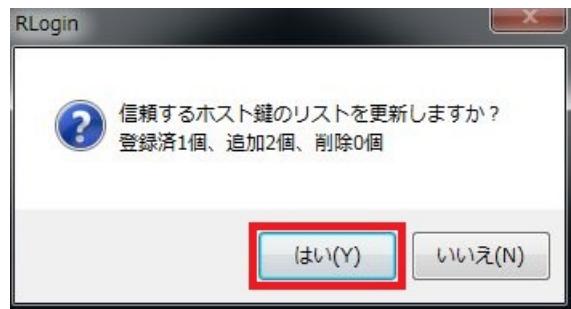
すると初回のみ、この「公開鍵の確認」が表示（図 3.36）されます。これは「初めて入るサーバだけど信頼していいですか？本当に接続しますか？」と聞かれているので、「接続する」をクリックしてください。サーバにはそれぞれフィンガープリントという固有の指紋があるため、下部の「この公開鍵を信頼するリストに保存する」にチェックが入っていれば RLogin が覚えていてくれて、次回以降は「これは前に信頼していいって言われたサーバだ！」と判断してそのまま接続させてくれます。



▲図 3.36 「公開鍵の確認」が表示されたら「接続する」をクリック

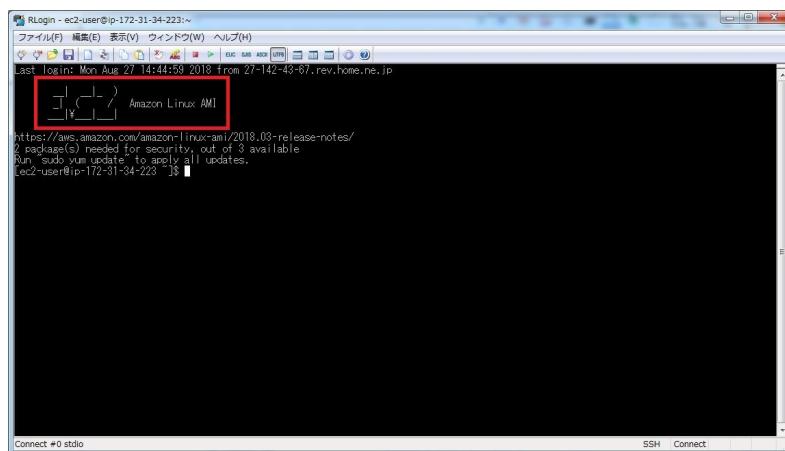
続いて「信頼するホスト鍵のリストを更新しますか？」と聞かれたら「はい」をクリッ

ク（図3.37）してください。



▲図3.37 「信頼するホスト鍵のリストを更新しますか？」と表示されたら「はい」をクリック

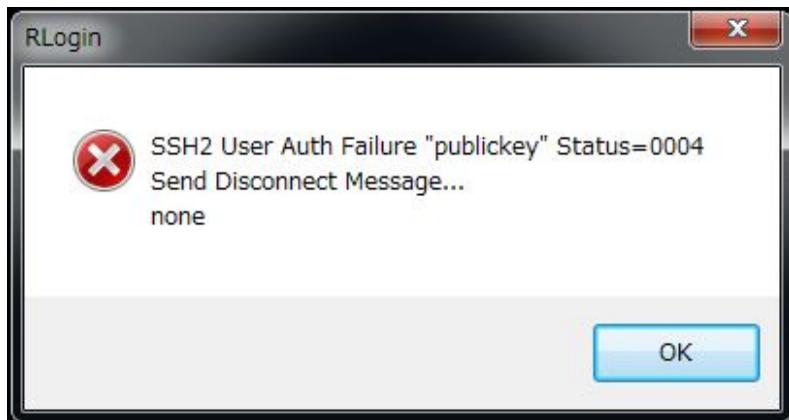
「Amazon Linux AMI」と表示（図3.38）されたら無事サーバに入っています。おめでとうございます！



▲図3.38 「EC2」というアスキーアートが表示されたら成功！

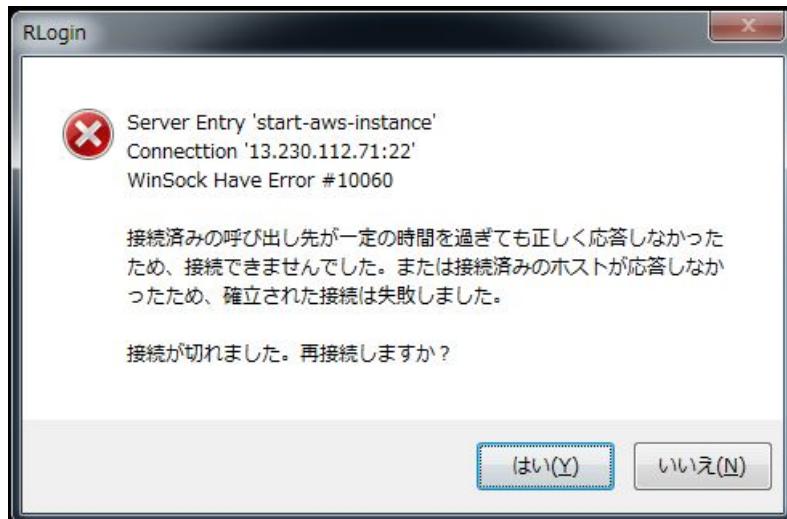
もし「Amazon Linux AMI」と表示されず、代わりに「SSH2 User Auth Failure "publickey" Status=0004 Send Disconnect Message... none」というようなエラーメッセージが表示（図3.39）されてしまったら、これは「鍵がない人は入れないよ！」とお断りされている状態です。恐らく「SSH認証鍵」をクリックして「start-aws-keypair.pem」を選択する作業を忘れているものと思われますので「SSH認証鍵」の設定を確認してみてく

ださい。



▲図 3.39 このエラーが表示されたら「SSH 認証鍵」の設定を確認しよう

あるいは「接続済みの呼び出し先が一定の時間を過ぎても正しく応答しなかったため、接続できませんでした。」というエラーメッセージが表示（図 3.40）されてしまった場合は、「ホスト名（サーバー IP アドレス）」に書いた「IPv4 パブリック IP」が間違っているものと思われます。「ホスト名（サーバー IP アドレス）」の IP アドレスを確認してみてください。



▲図 3.40 このエラーが表示されたら「ホスト名（サーバー IP アドレス）」の IP アドレスを確認しよう

3.4.2 使いのパソコンが Mac の場合

Macを使っている方は、ターミナルを起動してください。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。

そして開いたターミナルで次の文字を入力してReturnキーを押します。これはサーバに入るときに使う鍵をオーナー以外が使えないよう、chmodというコマンドで読み書き権限を厳しくしています。この作業は最初の1回だけで構いません。

```
chmod 600 start-aws-keypair.pem
```

続いてターミナルで次の文字を入力したら再びReturnキーを押します。「IPv4 パブリック IP」の部分は先ほどメモした「IPv4 パブリック IP」に書き換えてください。*-i*オプションは「サーバにはこの鍵を使って入ります」という意味ですので、もし「start-aws-keypair.pem」を保存した場所がデスクトップ^{*19}以外の場合は適宜書き換えてくだ

^{*19} ちなみに～（チルダ）はホームディレクトリを表すので「~/Desktop/start-aws-keypair.pem」は「/Users/<ユーザ名>/Desktop/start-aws-keypair.pem」と同じ意味です。

さい。

```
ssh ec2-user@IPv4 パブリック IP -i ~/Desktop/start-aws-keypair.pem
```

初回のみ次のようなメッセージが表示されますが、これは「初めてに入るサーバだけど信頼していいですか？本当に接続しますか？」と聞かれていますので、「yes」と打ってReturnキーを押してください。するとMacはちゃんとこのサーバのことを覚えてくれて、次回以降は「これは前に信頼していいって言われたサーバだ！」と判断してそのまま接続させてくれます。

```
Are you sure you want to continue connecting (yes/no)?
```

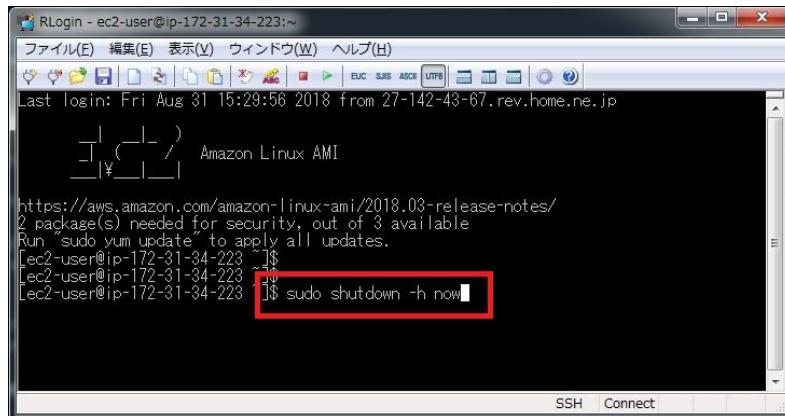
「Amazon Linux AMI」と表示されたら無事サーバに入っています。おめでとうございます！

3.4.3 サーバをシャットダウンしてみよう

折角サーバにログインできたので早速コマンド（命令）を打ってみましょう。できれば普段は絶対に打つ機会のないような・・・そうだ！

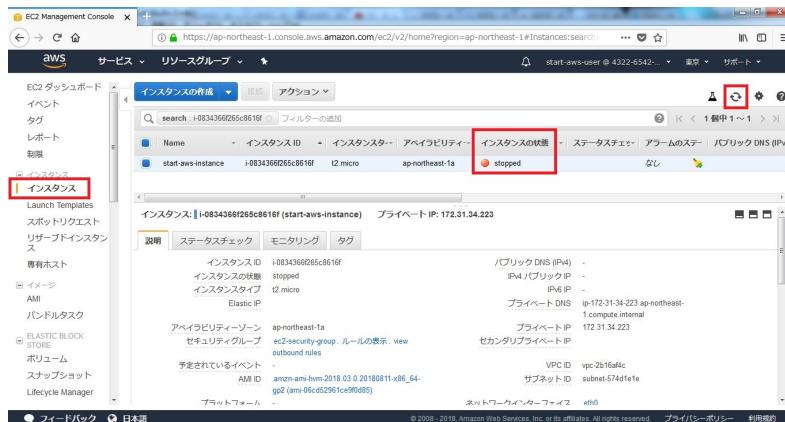
```
sudo shutdown -h now
```

と入力したらEnterキー（もしくはReturnキー）を押してください。（図3.41）これはサーバをシャットダウンするコマンドです。シャットダウンが気軽に試せるのは勉強用のインスタンスならではですね。



▲図 3.41 サーバをシャットダウンしてみよう

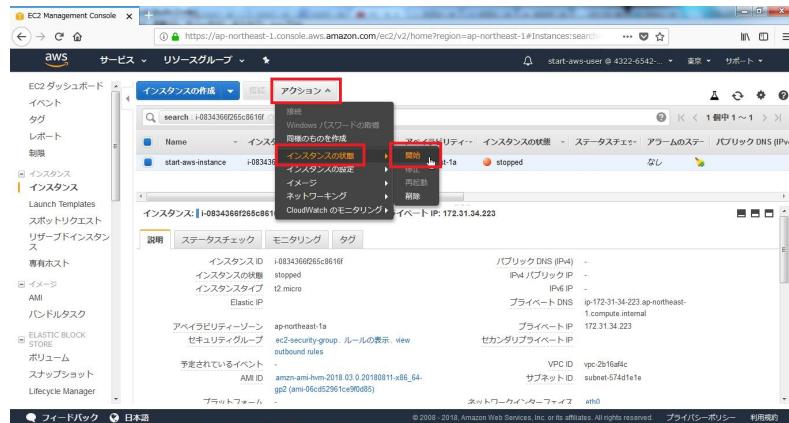
Enterキーを押したらサーバがシャットダウンされてSSHの接続も切れてしましました。それではちゃんとシャットダウンできたのか、EC2のダッシュボードからも確認してみましょう。EC2ダッシュボードで左メニューの「インスタンス」を選択（図 3.42）したら、右上の更新マークをクリックしてください。シャットダウンしたのでインスタンスの状態が「stopped」になっています。



▲図 3.42 EC2ダッシュボード>左メニューの「インスタンス」>更新マークをクリック

それでは止まってしまったサーバを再起動しましょう。「アクション」から「インスタンスの状態」で「開始」をクリック（図 3.43）してください。

3.4 SSH でサーバに入ってみよう



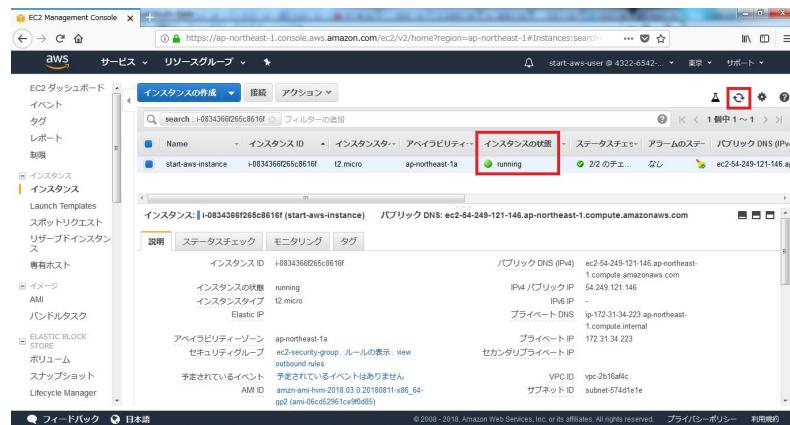
▲図 3.43 アクション>インスタンスの状態>開始をクリック

「これらのインスタンスを開始してよろしいですか?」(図 3.44)と確認が出るので「開始する」をクリックしてください。



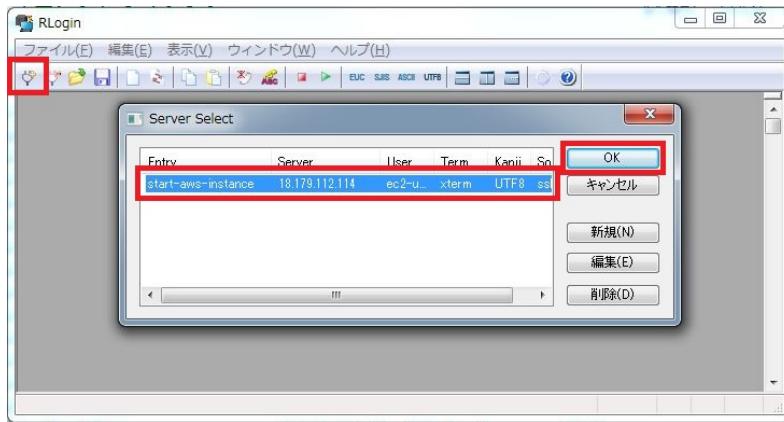
▲図 3.44 アクション>インスタンスの状態>開始をクリック

インスタンスの状態が「pending」になつたら「running」になるまで、何度か右上の更新マークをクリック (図 3.45) してみてください。



▲図3.45 インスタンスの状態が「running」になるまで右上の更新マークをクリック

インスタンスの状態が「running」になったら再びサーバに入ってみましょう。Windowsの方は RLogin で左のアイコンをクリック（図3.46）します。先ほどの「start-aws-instance」を選択したら「OK」をクリックしてください。

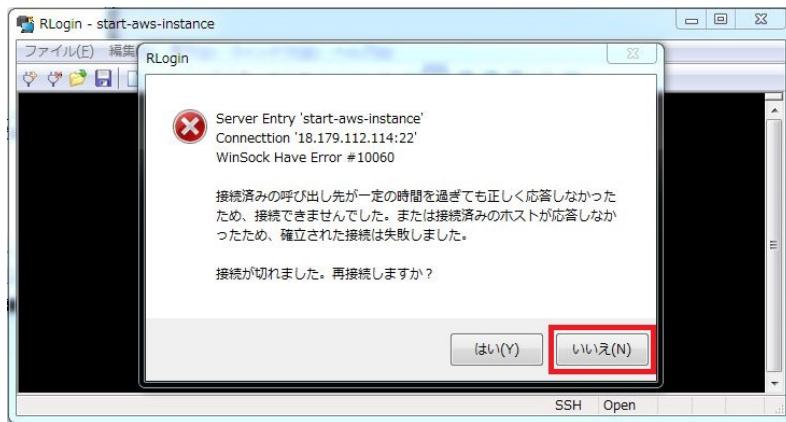


▲図3.46 「start-aws-instance」を選択したら「OK」をクリック

Macの方はターミナルで先ほどとまったく同じコマンド（命令）を実行してみてください。キーボードで「↑」を押すと直前に打ったコマンドが出てきますので、その状態で Returnキーを押してください。

```
ssh ec2-user@IPv4 パブリック IP -i ~/Desktop/start-aws-keypair.pem
```

シャットダウンしてから起動しただけで何の設定も変えてないので、すぐサーバに接続できると思ったのですが・・・暫く真っ黒い画面が続いた後、「接続済みの呼び出し先が一定の時間を過ぎても正しく応答しなかったため、接続できませんでした。」というエラー(図 3.47) が出て接続できなくなってしまいました。「いいえ」をクリックしてエラー画面を閉じましょう。



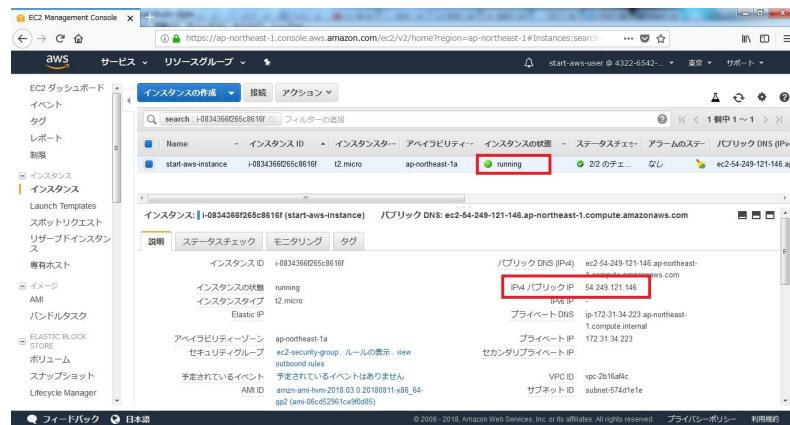
▲図 3.47 シャットダウンして起動したらサーバに繋がらなくなった！

いわゆる「何もしてないのに壊れた！」状態ですが、いったい何が起きたのでしょうか？

3.4.4 再起動しても変わらない ElasticIP をつけよう

急にサーバに入れなくなった原因を探るべく、再び EC2 ダッシュボードでインスタンスの状態を見てみましょう。(図 3.48) エラーメッセージには「応答しなかった」とあります。インスタンスはちゃんと起動しています。

では「IPv4 パブリック IP」に注目してみてください。何か気づきませんか？



▲図3.48 「IPv4 パブリック IP」に注目してみよう

先ほど自分でメモした「IPv4 パブリック IP」と、今表示されている「IPv4 パブリック IP」を比較（表3.2）してみてください。

▼表3.2 インスタンスのIPv4 パブリック IPを比較

シャットダウン前のIPv4 パブリック IP	現在のIPv4 パブリック IP
13.230.112.72	54.249.121.146

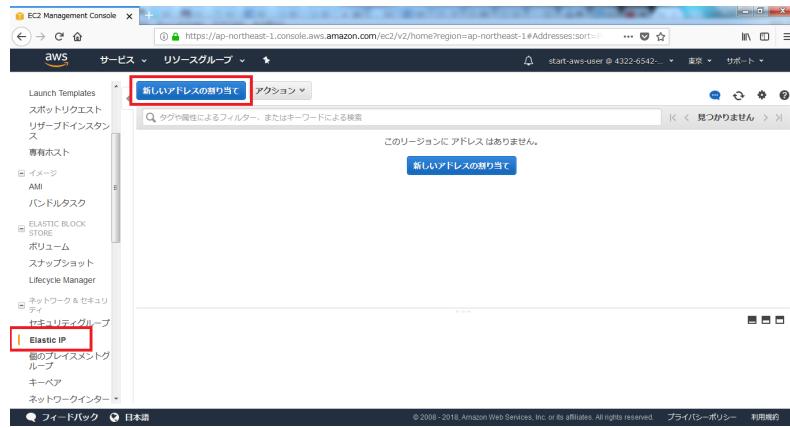
そうなんです。シャットダウンする前と、起動しなおした今の「IPv4 パブリック IP」を比較すると、まったく別々のIPアドレスになっているのです。

実はEC2においてIPアドレスは動的なものです。つまりインスタンスが停止されると、そのインスタンスに紐づいていたIPアドレスは解放されてしまい、インスタンスが再起動されるとまた新しいIPアドレスが紐づけられる、というのがEC2におけるIPアドレスの仕様^{*20}なのです。会社にたとえると、毎日19時に仕事を終えてオフィスのプレーカーを落として帰ると、翌日出社して電気をつけたときには代表電話番号が昨日とは違うものに変わっている、というような感じです。毎日「弊社の電話番号が変わりまして。今日の番号はこれです」とクライアントへ連絡するのは嫌です。

このままだとシャットダウンして起動するたびに「サーバに繋がらなくなった！」と大騒ぎして、EC2ダッシュボードで新しいIPアドレスを確認する破目になります。そうならないようElasticIPという静的な（＝勝手に変わらない）IPアドレスのサービスでサーバに固定のIPアドレスをつけましょう。

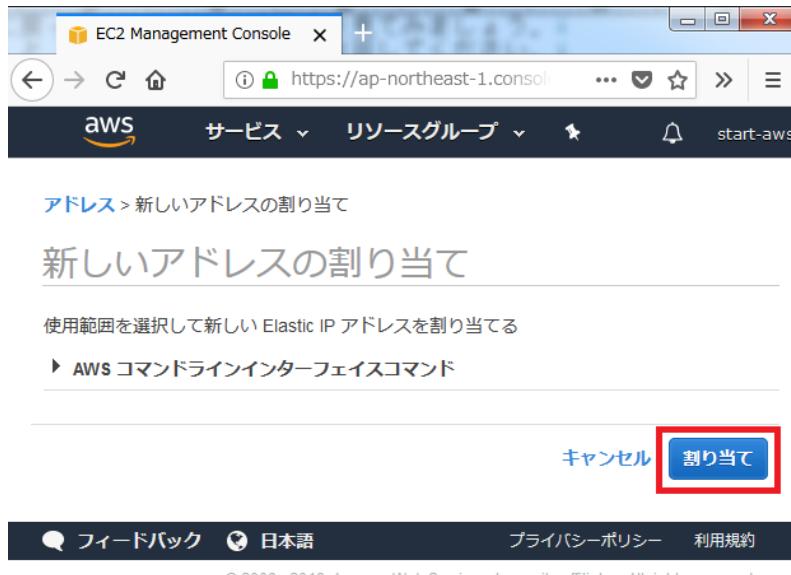
*20 ちなみに再起動であれば「IPv4 パブリック IP」は変わりません。シャットダウンして起動しなおしたときだけ変わってしまいます。

EC2 ダッシュボードの左メニューで「ネットワーク & セキュリティ」の中にある「Elastic IP」をクリック（図 3.49）してください。まだ Elastic IP がひとつもないため、「このリージョンにアドレスはありません。」と表示されます。「新しいアドレスの割り当て」をクリックします。

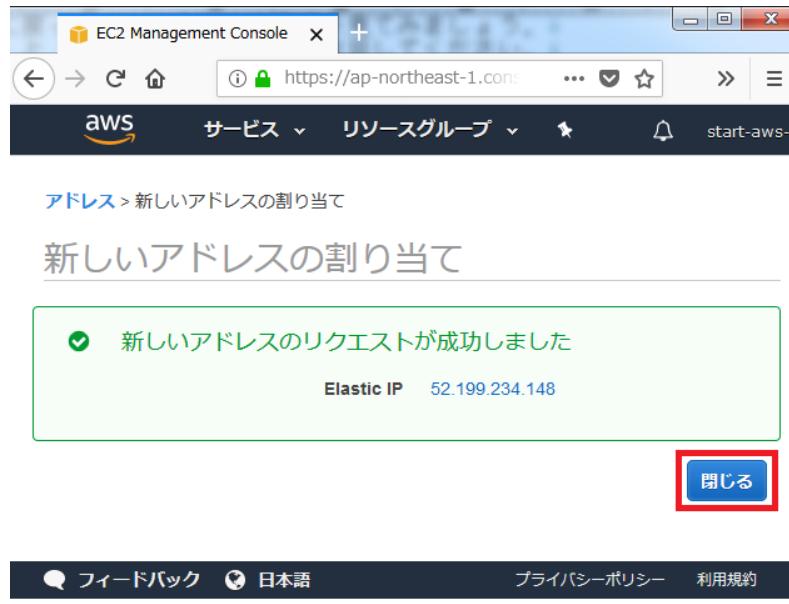


▲図 3.49 「Elastic IP」で「新しいアドレスの割り当て」をクリック

「割り当て」をクリック（図 3.50）して、「新しいアドレスのリクエストが成功しました」と表示（図 3.51）されたら「閉じる」をクリックします。



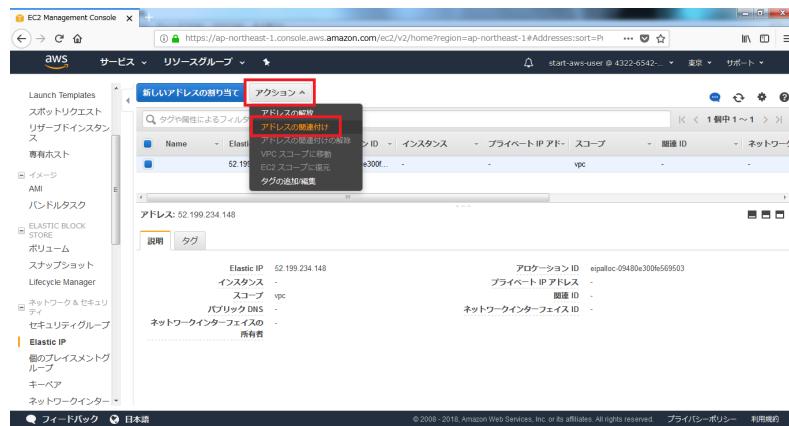
▲図 3.50 「割り当て」をクリック



© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

▲図 3.51 「閉じる」をクリック

今はまだ Elastic IP を手に入れただけでインスタンスにはくっついていないので紐づける作業をしましょう。アクションから「アドレスの関連付け」をクリック（図 3.52）してください。



▲図 3.52 アクションから「アドレスの関連付け」をクリック

リソースタイプは「インスタンス」のままで構いません。インスタンスは「インスタンスを選択します」と書かれた箇所をクリックすると、下に先ほど作った「start-aws-instance」が表示されるのでクリック（図3.53）します。



▲図3.53 「start-aws-instance」をクリック

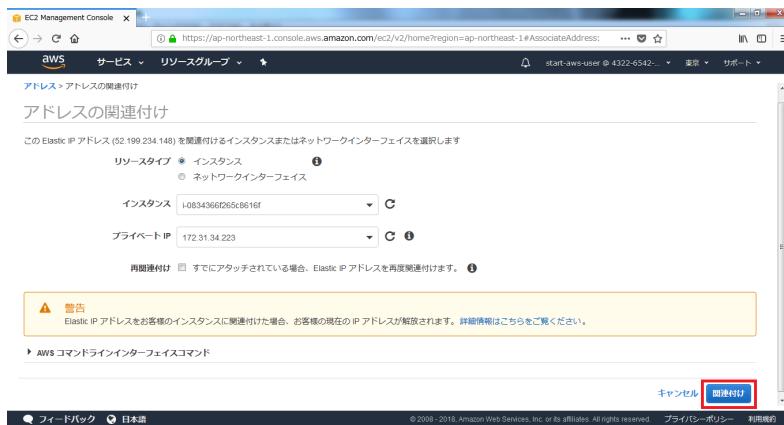
プライベートIPは「プライベートIPの選択」と書かれた箇所をクリックすると、「start-aws-instance」のIPアドレスが表示されるのでクリック（図3.54）します。



▲図3.54 「start-aws-instance」のIPアドレスをクリック

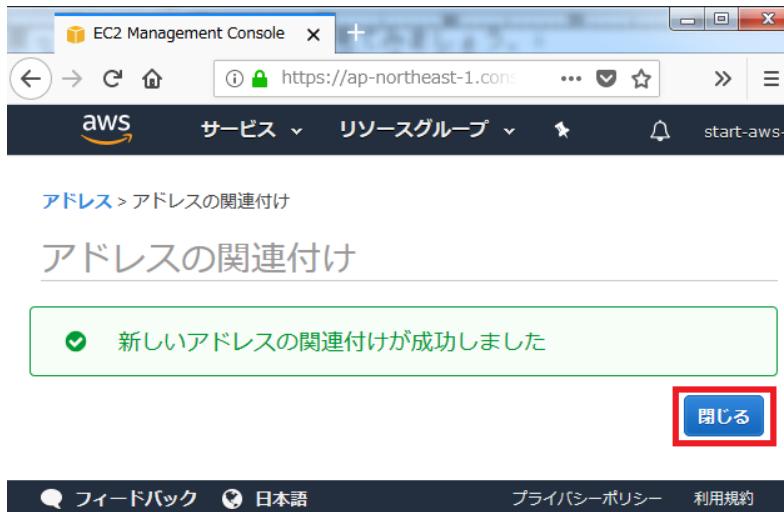
再関連付けにチェックは入れません。この状態（図3.55）で「関連付け」をクリックしてください。

3.4 SSH でサーバに入ってみよう



▲図 3.55 「関連付け」をクリック

「新しいアドレスの関連付けが成功しました」と表示（図 3.56）されたら「閉じる」をクリックします。



▲図 3.56 「閉じる」をクリック

Elastic IP をしっかりメモ（表 3.3）しておきましょう。「Elastic IP」のところへカーソルを持っていくと「クリップボードにコピー」と表示（図 3.57）されますので、パソコン

ンの中のメモ帳などにもメモしておくと便利です。



▲図 3.57 「閉じる」をクリック

▼表 3.3 Elastic IP をメモ

例	Elastic IP
52.199.234.148	

これでインスタンスに「シャットダウン＆起動しても勝手に変わらない IP アドレス」がついたので、今後、RLogin やターミナルでサーバに接続するとき^{*21}はこの IP アドレスをずっと使えます。

でも折角つけた ElasticIP ですが、この「52.199.234.148」のような数字の羅列を覚えたり、SSH でログインするたびに打ち込むのはちょっと面倒ですよね。

皆さんも普段電話をかけるときに、電話番号をいちいち打つのは面倒なのでアドレス帳に名前と電話番号を登録していますよね。それと同じようにネームサーバにドメイン名と IP の紐づけを登録しておけば、ドメイン名を使ってサーバに入れるようになります。

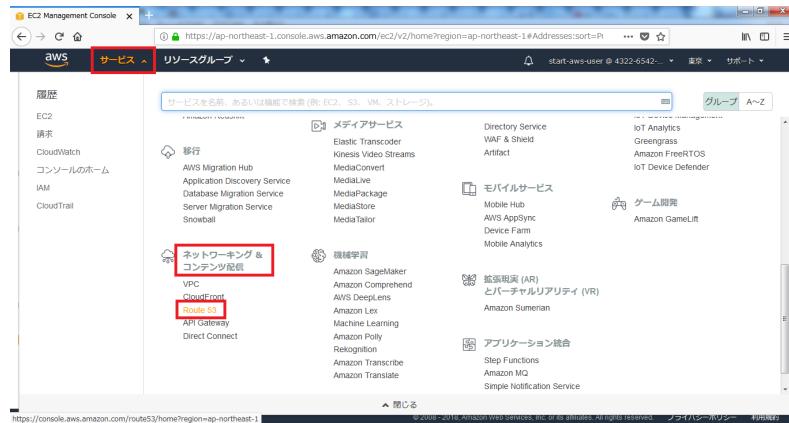
皆さんは「DNS をはじめよう」で買った自分のドメインを持っているので、早速ドメイン名と IP アドレスの組み合わせをネームサーバに登録してみましょう。ドメイン名の設定は Route53 で行います。

3.4.5 サーバに入るときに使うドメイン名を作ろう

マネジメントコンソールの左上にある「サービス」から、「ネットワーキング＆コンテンツ配信」の下にある「Route53」（図 3.58）をクリックしてください。

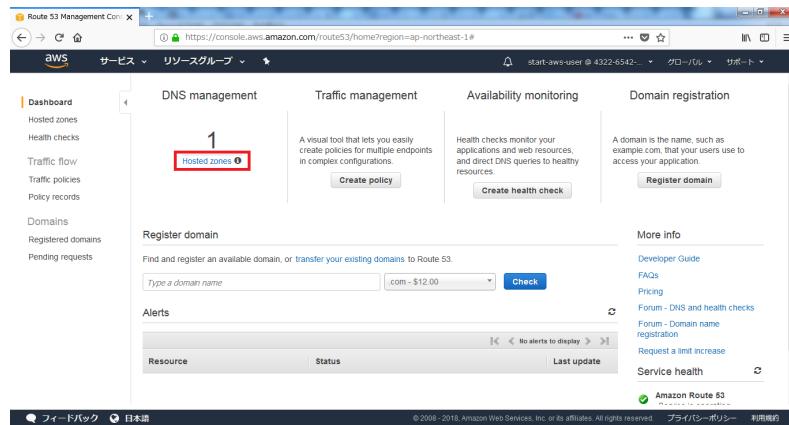
^{*21} 今回はサーバが 1 台だけなので直接ログインしていますが、通常は踏み台サーバ（英語だと Bastion host と呼ばれることが多い）を用意して外部からは踏み台サーバにしかログインできないようにしておき、個々のウェブサーバや DB サーバには踏み台を経由しないと入れない、という構成にすることが多いです。サーバは大切な箱入り娘なので、怖いお父さん（踏み台サーバ）を介さないと話しかけられないようにしてすることで安全性を高めているのです。

3.4 SSH でサーバに入ってみよう



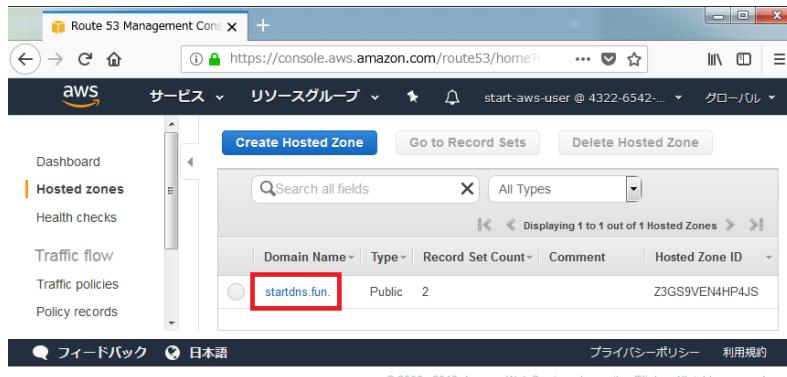
▲図 3.58 サービス>ネットワーキング & コンテンツ配信>Route53

Route53 ダッシュボードを開いたら DNS management の「Hosted zones」をクリック（図 3.59）します。



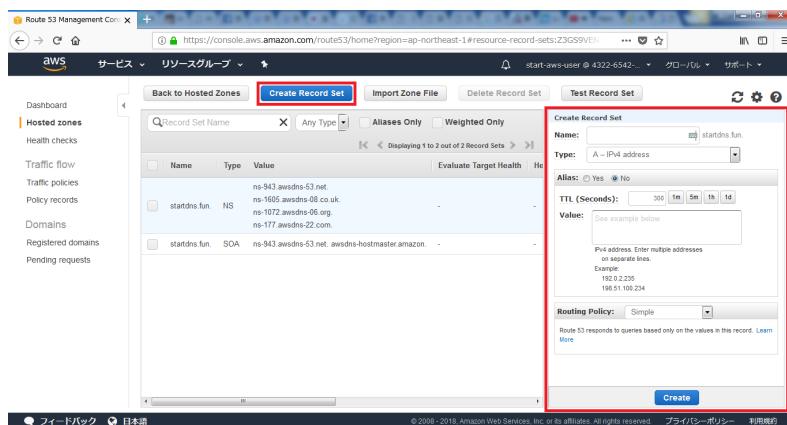
▲図 3.59 「Hosted zones」をクリック

Domain Name の自分のドメイン名（私の場合は startdns.fun）をクリック（図 3.60）します。



▲図 3.60 自分のドメイン名をクリック

新しくリソースレコード^{*22}を作りたいので「Create Record Set」をクリック（図 3.61）してください。すると右側にリソースレコードの情報を入力するフォームが出てきます。



▲図 3.61 「Create Record Set」をクリック

Nameには「login」、Valueには先ほどメモしたElastic IPを入力（図 3.62）してください。それ以外の箇所は何も変更せずそのまま構いません。入力できたら「Create」をクリックします。

^{*22} リソースレコードってなんだっけ？と思ったら「DNSをはじめよう」の「2.4 リソースレコード」を参照。

Create Record Set

Name: .startdns.fun.

Type: A – IPv4 address

Alias: Yes No

TTL (Seconds): 300

Value:

IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234

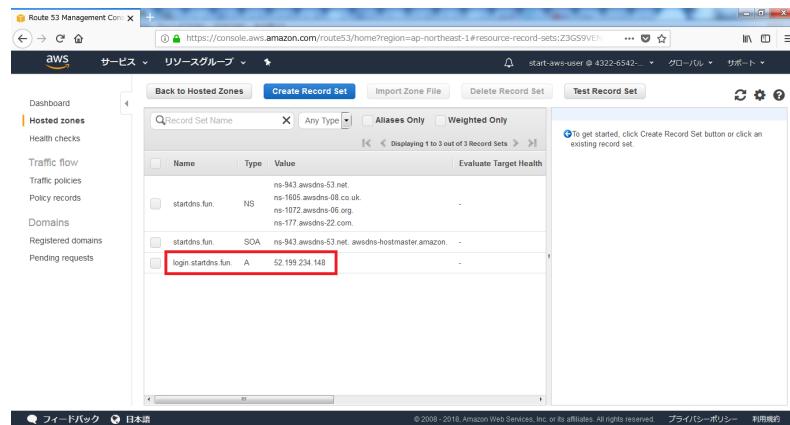
Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. [Learn More](#)

Create

▲図 3.62 Name には「login」、Value には先ほどメモした Elastic IP を入力

これで「login. 自分のドメイン名」(図 3.63) という A レコードが作成できました。

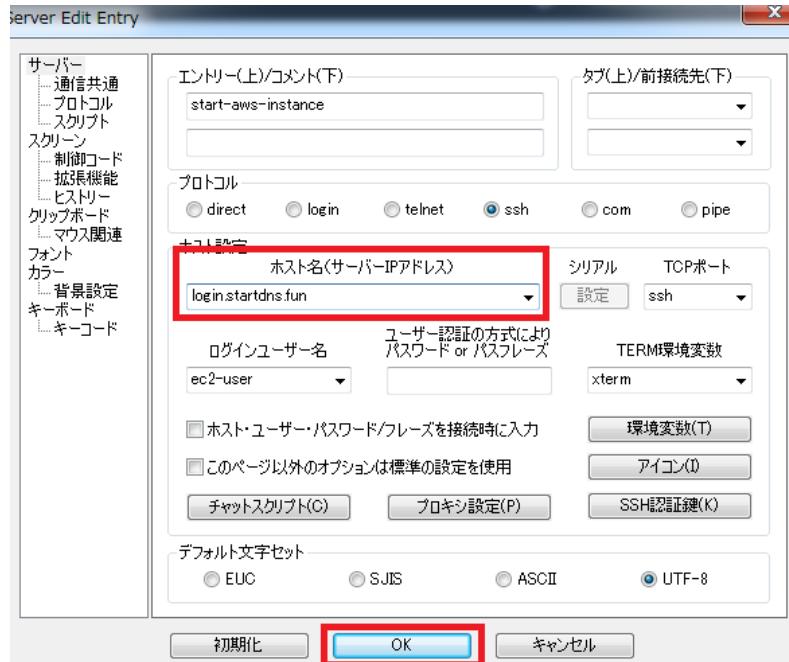


▲図3.63 「login. 自分のドメイン名」という A レコードができた

ではこのドメイン名を使って再びサーバにログインしてみましょう。Windowsの方は RLogin で左のアイコンをクリック（図 3.64）してから「編集」をクリックしてください。「ホスト名（サーバー IP アドレス）」を先ほど作ったドメイン名の「login. 自分のドメイン名」に変更（図 3.65）したら OK をクリックします。

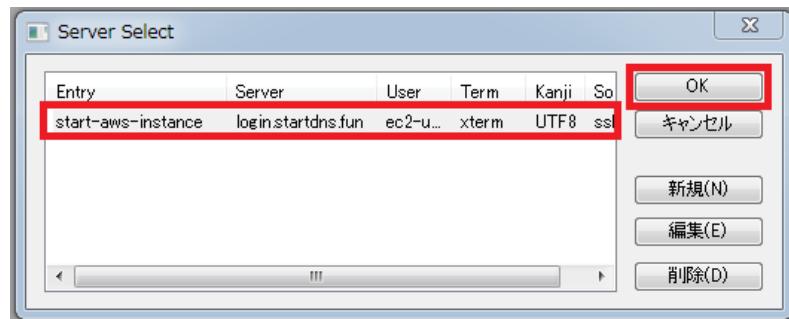


▲図3.64 「編集」をクリック



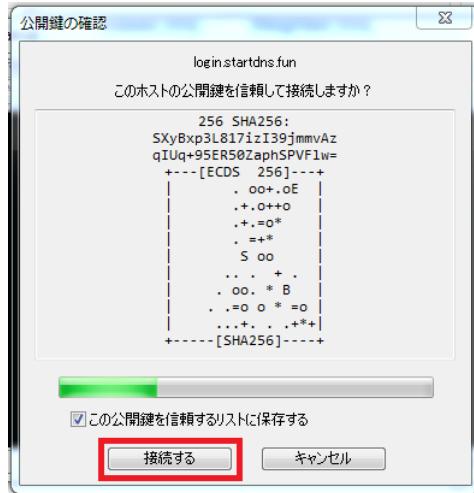
▲図 3.65 「ホスト名 (サーバー IP アドレス)」を「login. 自分のドメイン名」にして OK をクリック

「start-aws-instance」を選択（図 3.66）したら「OK」をクリックしてください。



▲図 3.66 「start-aws-instance」を選択したら「OK」をクリック

この名前でサーバに接続するのは初めてなので、また「公開鍵の確認」が表示（図 3.67）されますが「接続する」をクリックしてください。



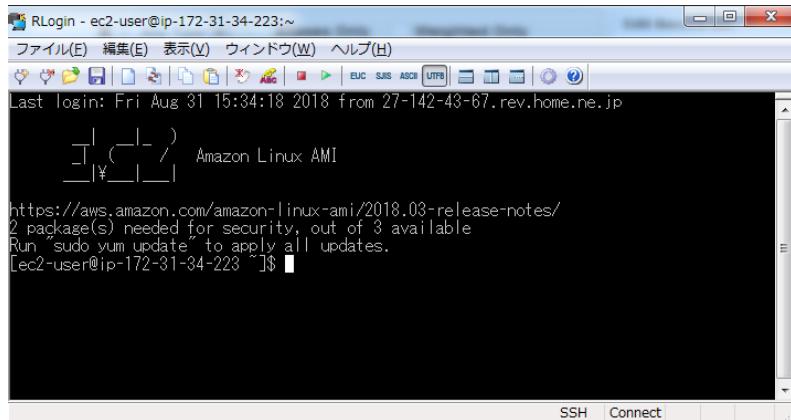
▲図3.67 「公開鍵の確認」が表示されたら「接続する」をクリック

続いて「信頼するホスト鍵のリストを更新しますか？」と聞かれたら「はい」をクリック（図3.68）してください。



▲図3.68 「信頼するホスト鍵のリストを更新しますか？」と表示されたら「はい」をクリック

「Amazon Linux AMI」と表示（図3.69）されましたか？ 無事にドメイン名を使ってサーバに入れました。Windowsの皆さん、おめでとうございます！



▲図 3.69 ドメイン名を使ってサーバに入れた！

Mac の方はターミナルを起動して次のコマンドを実行してください。「キーボードで「↑」を押すと直前に打ったコマンドが出てきますので、アットマークの後ろを「login. 自分のドメイン名」に変更して Return キーを押してください。

```
ssh ec2-user@login. 自分のドメイン名 -i ~/Desktop/start-aws-keypair.pem
```

こちらもこの名前でサーバに接続するのは初めてなので次のようなメッセージが表示されます。「yes」と打って Return キーを押してください。

```
Are you sure you want to continue connecting (yes/no)?
```

「Amazon Linux AMI」と表示されたら、無事にドメイン名を使ってサーバに入れました。Mac の皆さんもおめでとうございます！

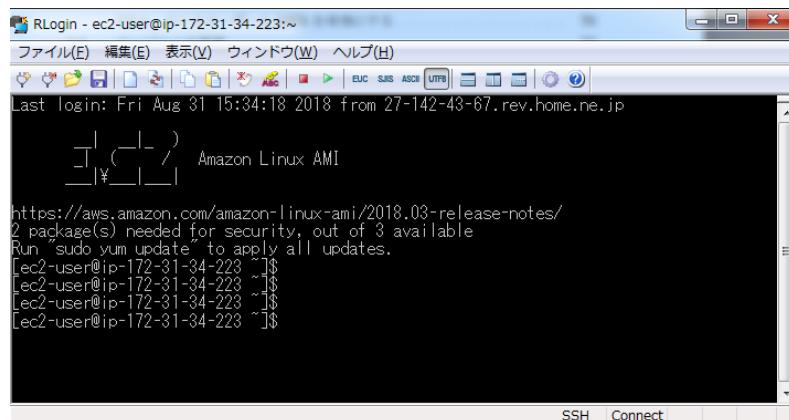
今後は今やったのと同じやり方をそのまま繰り返せばサーバにログインできます。ドメイン名というとどうしても「ブラウザで入力してサイトを見るときに使うもの」というイメージがありますが、「名前から IP アドレスが引けるもの」なのでこういう使い方もできるのです。

3.5 ターミナルでサーバを操作・設定してみよう

3.5.1 ターミナルの基本操作に慣れよう

プロンプト

では黒い画面で何回か Enter キー（あるいは Return キー）を押してみましょう。（図 3.70）普通に改行されますよね。



▲図 3.70 Enter キーを押すと改行されて、プロンプトが常に表示されている

このとき左側にずっと出ている次のような表示は「プロンプト」といって、ログインしているユーザ名やサーバの名前などが表示されています。

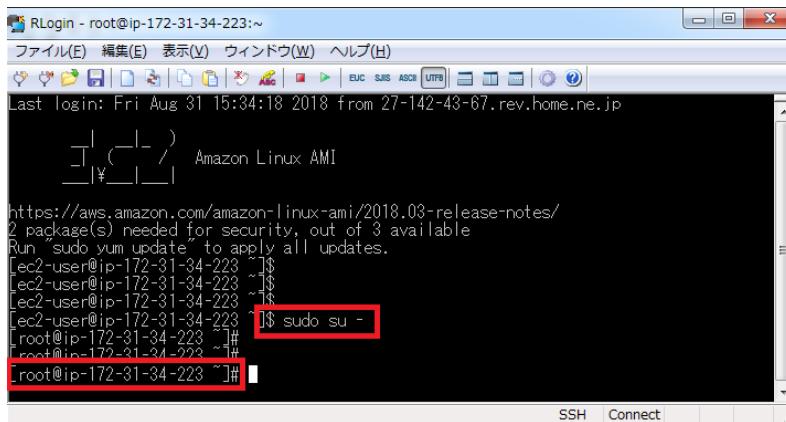
```
[ec2-user@ip-172-31-34-223 ~]$
```

プロンプトを見ると今は「ec2-user」という一般ユーザであることが分かります。これからサーバに色々な設定をしたいのですが、一般ユーザだと権限がないので「root」という全権限をもったユーザになりましょう。

```
$ sudo su -
```

と書いて Enter キーを押すと root になれます。（図 3.71）また何回か Enter キーを押して改行してみましょう。いちばん左側に出ているプロンプトが次のように変化しまし

たか？



▲図 3.71 sudo su -を書いて Enter キーを押すと root になれる

[root@ip-172-31-34-223 ~]#

ユーザ名が「ec2-user」から「root」に変わりました。それからいちばん右の部分も「\$」から「#」に変わっています。プロンプトは一般ユーザだと「\$」、全権を持っている root だと「#」という表示です。今後は「このコマンドを root で実行してください」のように実行ユーザを詳しく書くことはしませんので、書いてあるプロンプトが「\$」だったら ec2-user のような一般ユーザで実行、「#」だったら root で実行するんだ、と思ってください。ターミナル上で「\$」や「#」を自分で入力する必要はありません。

コマンドは失敗したときだけエラーを吐く

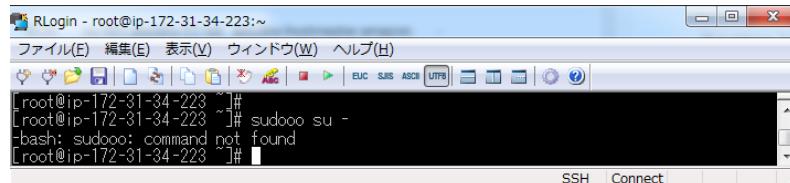
前述の「`sudo su -`」のようなものを「コマンド」と呼びます。コマンドとはサーバに対して「あれをして」「これをして」と頼む命令のようなものです。

サーバでコマンドを打った場合、基本的に上手くいったときは何も言わないで失敗したときだけエラーを吐きます。ですのでコマンドを打った時に何も表示されなくても不安にならなくて大丈夫です。

先ほどの「私を root にして！」という命令である「sudo su -」も、上手くいってちゃんと root になれたのでメッセージは一切出ていないですね。これが「sudo」を打ち間違えて、こんな風に実行するとどうなるでしょう？

```
$ sudooo su -
```

「sudooo なんてコマンドは見つからなかったよ」というエラーメッセージ（図 3.72）が表示されました。

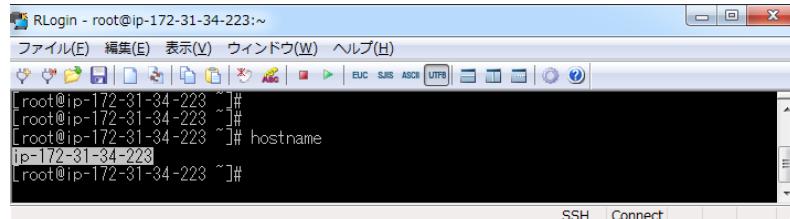


▲図 3.72 「sudooo: command not found」というエラーが表示された

このように何か失敗したときだけエラーが出ます。英語でエラーが出るとそれだけでパニックになってしまいますが、落ち着いてゆっくり読めば「sudooo: command not found・・・ああ、sudooo っていうコマンドが見つかりませんでした、って書いてある」と判読できると思います。エラーが出たら声に出してゆっくり読んでみましょう。

コピー&ペーストするには？

ターミナルで表示されている内容をコピーしたいときは、コピーしたい部分をマウスで選択するだけです。（図 3.73）選択してから Ctrl+C を押す必要はありません。



▲図 3.73 マウスで選択するだけでコピーできる

コピーした内容をターミナルへペーストしたいときは右クリックするだけです。Ctrl+C は使えないで注意してください。

ターミナルを閉じたいとき

もう今日の勉強は終わり！ サーバとの接続を切ってターミナルを閉じたい、というときは exit (イグジット) というコマンドを叩きます。

```
# exit
```

root になっているときに exit を叩くと ec2-user に戻れます。そして ec2-user で再び exit を叩くと、サーバの接続を切ってターミナルを閉じることができます。

```
$ exit
```

exit をせずに右上の赤い×を押してウィンドウを閉じるのは、電話を切るときに通話オフのボタンを押さずに電話線を引っこ抜くような乱暴な切り方なのでお勧めしません。

3.5.2 ミドルウェアをインストール

それでは必要なミドルウェアをインストールしていきましょう。最初に root になっておいてください。インストールするときは yum (ヤム) というコマンドを使います。

```
$ sudo su -
```

先ずは yum で色々アップデートしておきましょう。Windows アップデートみたいなものです。最後に「Complete!」と表示されたら問題なく完了しています。ちなみに-y オプションは YES を意味するオプションです。-y オプションをつけないで実行すると「これとこれを更新するけどいい？ ダウンロードサイズとインストールサイズはこれくらいだよ」という確認が表示されて、y と入力して Enter キーを押さないと更新されません。

```
# yum update -y
```

続いて「DNS をはじめよう」で出てきた whois コマンドと、DB 接続時に必要な mysql クライアントを入れておきます。こちらも最後に「Complete!」と表示されれば OK です。

```
# yum install -y jwhois mysql
```

さらに PHP と Apache も入れます。PHP は Apache のモジュール（部品）なので、PHP7 をインストールしようとすると yum が気を利かせて Apache2.4^{*23}も入れてくれます。

^{*23} サーバに入っている PHP や Apache のバージョンがいくつなのか？ という情報は大切です。今後、あ

```
# yum install -y php72 php72-mbstring php72-mysqlnd
```

Apacheの正式名称は「Apache HTTP Server」です。ちょっと分かりにくいかも知れませんが、パソコンにMicrosoft Excelをインストールしたら「表計算というサービスが提供できるパソコン」になるのと同じで、サーバにこのApacheをインストールすると「リクエストに対してウェブページを返すサービスが提供できるサーバ」、つまりウェブサーバになります。今回はApacheを入れましたがウェブサーバのミドルウェアは他にも色々な種類があります。

インストールが終わったので、サーバを起動した際にApacheが自動で立ち上がってく るよう、自動起動の設定もオンにしておきましょう。

自動起動の管理対象にhttpd(Apacheのこと)を追加した上で、自動起動をオンにします。

```
# chkconfig --add httpd  
# chkconfig httpd on
```

自動起動の設定ができたか確認してみましょう。

```
# chkconfig --list httpd  
httpd           0:off    1:off    2:on     3:on     4:on     5:on     6:off
```

Linuxにはグラフィカルモードやシングルユーザモードなどランレベルと呼ばれるいくつかのモードがあります。この0から6の数字はランレベルごとの自動起動設定を表しており、現状のランレベルは3なので3のところがonになっていれば大丈夫です。

3.5.3 OSの基本設定をしておこう

date(データ)コマンドでサーバの時間を確認してみましょう。日本はいま18:14なのですが、サーバの時間は9:14でずれてしまっています。

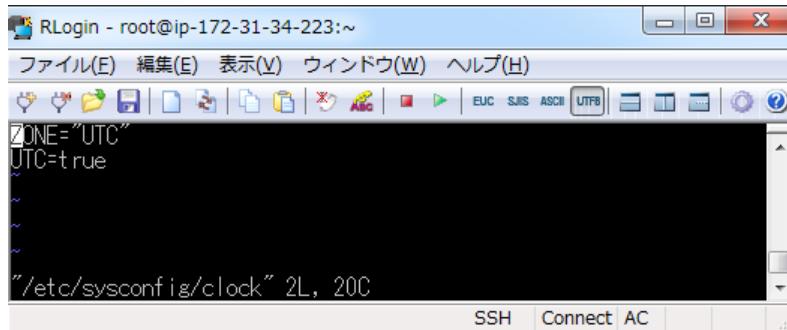
```
# date  
Sat Sep 1 09:14:44 UTC 2018
```

なたがPHPのソースコードやApacheの設定ファイルを書こうと思って調べたとき、PHP5の関数やApache2.2の設定方法を参考にしてしまうと、PHP7やApache2.4の環境では上手く動かない可能性があります。

日本時間になるようタイムゾーンの変更を行いましょう。

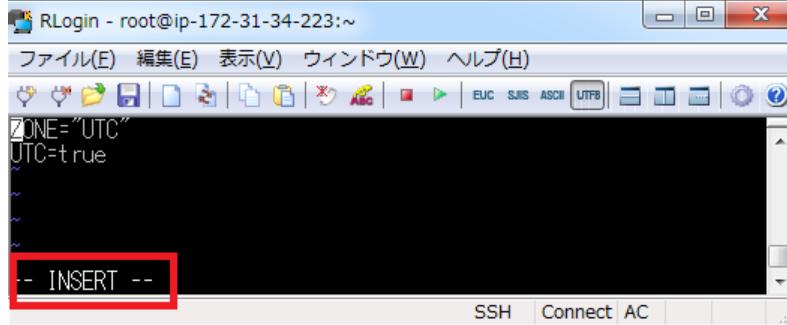
```
# vi /etc/sysconfig/clock
```

vi（ブイアイ）はテキストファイルを編集するためのコマンドです。vi コマンドでファイルを開くと、最初は次のような「閲覧モード」の画面（図 3.74）が表示されます。閲覧モードは「見るだけ」なので編集ができません。



▲図 3.74 vi コマンドでファイルを開いた

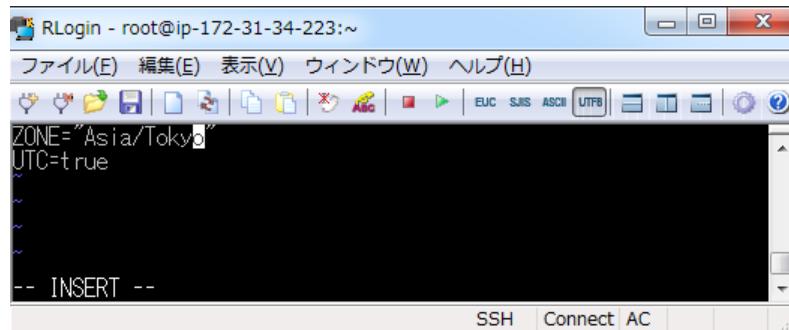
この状態で i（アイ）を押すと「編集モード」*24に変わります。（図 3.75）左下に「-- INSERT --」と表示されていたら「編集モード」です。



▲図 3.75 i（アイ）を押すと「-- INSERT --」と表示される「編集モード」になった

*24 ここでは初心者の方でも直感的に分かるよう「閲覧モード」「編集モード」と呼んでいますが、正しくは「ノーマルモード」「インサートモード」です。

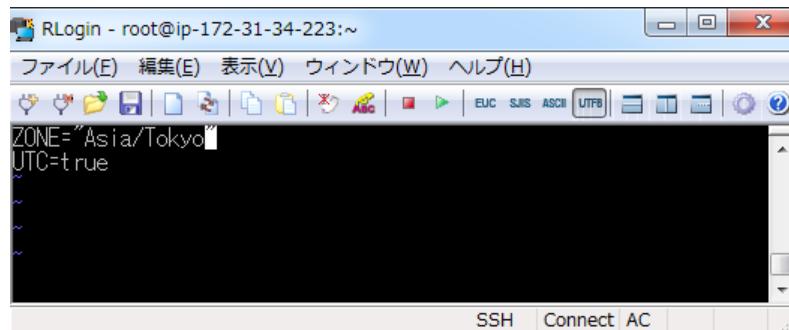
「編集モード」になるとファイルが編集できるようになります。それでは「ZONE="UTC"」を「ZONE="Asia/Tokyo"」(図 3.76) に書き換えてください。



A screenshot of a terminal window titled "RLogin - root@ip-172-31-34-223:~". The window has a menu bar with "ファイル(F)", "編集(E)", "表示(V)", "ウィンドウ(W)", and "ヘルプ(H)". Below the menu is a toolbar with icons for file operations like Open, Save, Copy, Paste, and Find. The status bar at the bottom shows "SSH Connect AC". The main text area contains the command "ZONE='Asia/Tokyo'" followed by "UTC=true" on a new line. The text "~~ INSERT ~~" is visible at the bottom left of the text area, indicating edit mode.

▲図 3.76 「ZONE="UTC"」を「ZONE="Asia/Tokyo"」に書き換える

「編集モード」のままだと保存ができないので書き終わったら ESC キーを押します。すると左下の「-- INSERT --」が消えて再び「閲覧モード」になります。(図 3.77)

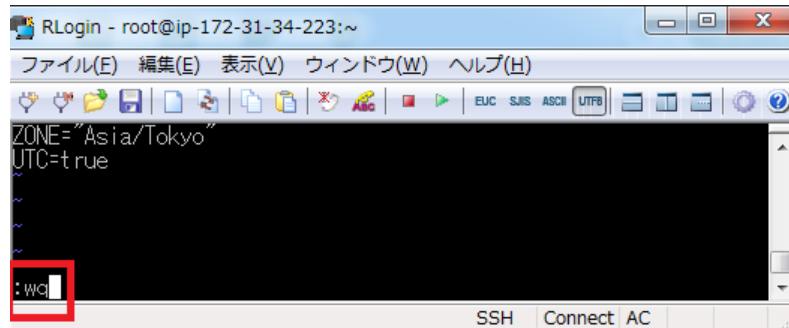


A screenshot of a terminal window titled "RLogin - root@ip-172-31-34-223:~". The window has a menu bar with "ファイル(F)", "編集(E)", "表示(V)", "ウィンドウ(W)", and "ヘルプ(H)". Below the menu is a toolbar with icons for file operations like Open, Save, Copy, Paste, and Find. The status bar at the bottom shows "SSH Connect AC". The main text area contains the command "ZONE='Asia/Tokyo'" followed by "UTC=true" on a new line. The text "~~ INSERT ~~" is no longer visible at the bottom left of the text area, indicating edit mode has been exited.

▲図 3.77 ESC を押すと左下の「-- INSERT --」が消えて再び「閲覧モード」になる

「閲覧モード」に戻ったら「:wq」^{*25}と入力して Enter キーを押せば保存されます。(図 3.78)

*25 write (書き込んで)、抜ける (quit) という命令なので wq です。



▲図 3.78 「:wq」と入力して Enter キーを押せば保存される

色々やっているうちになんだか誤が分からなくなってしまって「今の全部なかったことにしたい！ 取り合えず vi からいったん抜けたい！」と思ったときは、ESC キーを押して「:q!」^{*26}と入力して Enter キーを押すと変更を保存せずに抜けることができます。

編集できたら cat (キャット) コマンド^{*27}でファイルの中身を確認してみましょう。

```
# cat /etc/sysconfig/clock
ZONE="Asia/Tokyo"
UTC=true
```

さらに ln コマンドでシンボリックリンクを作ります。シンボリックリンクは Windows でいうところのショートカットみたいなものですね。

```
# ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
```

ちなみに入力しているパス (path) は入力途中でタブを押すと自動的に補完されるので、全部手打ちしなくても大丈夫です。

```
たとえば
# ln -sf /usr/sh
まで打ってから Tab キーを押すと
```

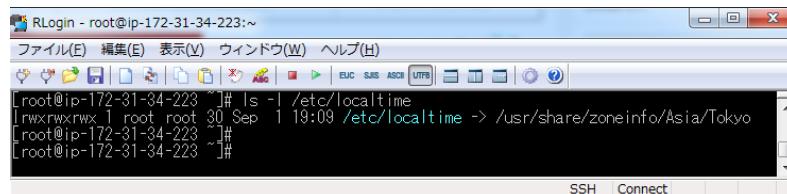
^{*26} 保存せずに強制終了 (quit!) という命令なので q!です。

^{*27} cat は猫ではなく「conCATenate files and print on the standard output」の略だそうです。私はいつも「猫がファイルの中身を全部出して見せてくれてるんだ」と考えることでちょっと幸せになります。

```
# ln -sf /usr/share/
のように自動補完されます
```

シンボリックリンクが生成されたか ls コマンド^{*28}で確認してみましょう。(図 3.79)

```
# ls -l /etc/localtime
```



▲図 3.79 「:wq」と入力して Enter キーを押せば保存される

続いて locale (言語) の設定をします。今は言語の設定が英語になっているのでエラーメッセージなども英語で表示されますが、分かりやすくするため言語設定を日本語に変更しましょう。

```
# vi /etc/sysconfig/i18n
```

先ほどと同じ vi コマンドでファイルを開くと、最初は「閲覧モード」になっています。
i (アイ) で「編集モード」にして「en_US」の部分を「ja_JP」に書き換えてください。

```
# vi /etc/sysconfig/i18n
LANG=en_US.UTF-8
↓
LANG=ja_JP.UTF-8
```

書き終わったら ESC キーを押して「閲覧モード」に戻り「:wq」で保存します。編集で
きたら cat (キャット) コマンド^{*29}でファイルの中身を確認してみましょう。

*28 ls は list の略で、名前のとおりファイルを一覧表示してくれるコマンドです。-l は long の略で「詳細を表示」というオプションです。

*29 cat は猫ではなく「conCATenate files and print on the standard output」の略だそうです。私はいつも「猫がファイルの中身を全部出して見せてくれてるんだ」と考えることでちょっと幸せになります。

```
# cat /etc/sysconfig/i18n  
LANG=ja_JP.UTF-8
```

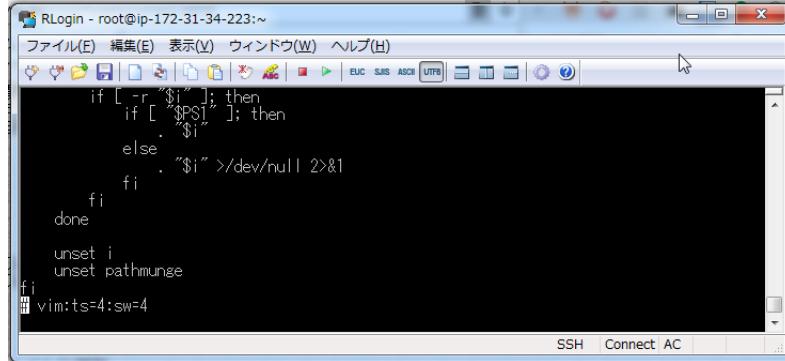
最後に history の設定を行います。history コマンドを叩くと今まで自分が実行したコマンドの履歴が見られます。

```
# history  
1 sudooo su -  
2 yum install -y jwhois mysql  
3 yum install -y php72 php72-mbstring php72-mysqlnd  
4 chkconfig --add httpd  
5 chkconfig httpd on  
6 chkconfig --list httpd  
7 date  
8 vi /etc/sysconfig/clock  
9 cat /etc/sysconfig/clock  
10 ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime  
11 ls -l /etc/localtime  
12 vi /etc/sysconfig/i18n  
13 cat /etc/sysconfig/i18n
```

とても便利なのですが、このままだと「そのコマンドをいつ実行したのか?」という日時が分かりません。また最大 1000 件までしか保存されないためそれ以上前の履歴を追うことができません。設定を変更して最大で 2000 件まで保存されて、日時も表示されるようにしましょう。

```
# vi /etc/bashrc
```

先ほどと同じ vi コマンドでファイルを開くと、最初は「閲覧モード」になっています。「閲覧モード」のままで shift+g を押してファイルの最終行へ移動（図 3.80）してください。



▲図 3.80 shift+g で最終行に移動した

最終行に移動したら o (オー) で下に 1 行追加して「編集モード」になり、次の 2 行を追記してください。

```
HISTTIMEFORMAT='%F %T'  
HISTFILESIZE=2000
```

書き終わったら ESC キーを押して「閲覧モード」に戻り「:wq」で保存します。編集でたら tail (テイル) コマンド^{*30}でファイルの中身を確認してみましょう。

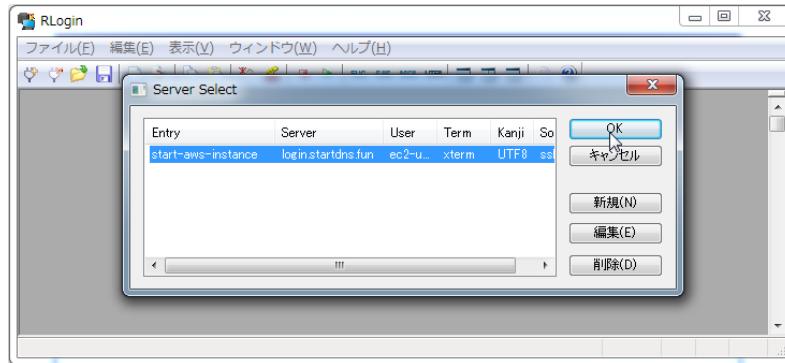
```
# tail -2 /etc/bashrc  
HISTTIMEFORMAT='%F %T'  
HISTFILESIZE=2000
```

以上で OS の基本設定は終了です。変更した設定を有効にするため reboot (リブート) コマンドでサーバを再起動しておきましょう。

```
# reboot
```

SSH の接続も切れてしまいますが、割とすぐに再起動しますので再度 RLogin やターミナルで接続(図 3.81)してみてください。今度はさつきと同じ設定でそのまま接続できるはずです。

^{*30} tail コマンドは名前のとおりファイルの尻尾、つまり末尾を表示するコマンド。引数で「-2」と指定すれば末尾から 2 行、「-10」と指定すれば末尾から 10 行が表示される。



▲図 3.81 さっきと同じ設定で接続してみよう

接続できたら date コマンドでサーバの時間を確認してみましょう。サーバのタイムゾーンが東京になって時間のずれはなくなり、言語も日本語に変わったことで次のように表示されるはずです。

```
$ date
2018年 9月 1日 土曜日 19:58:14 JST
```

続いて root になって history コマンドを叩いてみましょう。root になったときのメッセージも日本語に変わっていますね。

```
$ sudo su -
最終ログイン: 2018/09/01 (土) 20:00:12 JST 日時 pts/0

# history
1 2018-09-01 20:01:41 sudo su -
2 2018-09-01 20:01:41 ip addr
3 2018-09-01 20:01:41 hostname
4 2018-09-01 20:01:41 chkconfig --add httpd
5 2018-09-01 20:01:41 chkconfig httpd on
6 2018-09-01 20:01:41 chkconfig --list httpd
7 2018-09-01 20:01:41 date
8 2018-09-01 20:01:41 vi /etc/sysconfig/clock
9 2018-09-01 20:01:41 cat /etc/sysconfig/clock
10 2018-09-01 20:01:41 ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
11 2018-09-01 20:01:41 ls -l /etc/localtime
12 2018-09-01 20:01:41 vi /etc/sysconfig/i18n
13 2018-09-01 20:01:41 cat /etc/sysconfig/i18n
14 2018-09-01 20:01:41 vi /etc/bashrc
15 2018-09-01 20:01:41 tail -2 /etc/bashrc
16 2018-09-01 20:01:41 reboot
17 2018-09-01 20:02:58 date
```

```
18 2018-09-01 20:03:01 history
```

設定変更して reboot するまでは日時が記録されていなかったためすべて同じ日時になっていますが、reboot 後はちゃんと実行した日時が表示されています。ちなみに history は「exit」するときに履歴が書き込まれるため、前述の「exit」で抜けずに赤い×を押してウインドウを閉じたりするとその分は記録されずに消えてしまいます。

以上で「サーバを立てる」という作業はおしまいです。

3.5.4 ターミナルはなんのためにある？

ターミナルで yum や vi を叩いてサーバの設定を色々してきましたが、ここで「結局、ターミナルって何なの？」という振り返りをしておきましょう。

ターミナルはサーバを操作するための画面です。

皆さんがパソコン使うときはモニタに表示された画面を見ながらキーボードとマウスを使って「フォルダを開いて先週作ったWordファイルを探す」とか「Wordファイルを開いて今週の報告書を書く」というような操作を操作をすると思います。フォルダを開くときは「ダブルクリック」をして、書いた内容を保存するときは「上書き保存する」ボタンを押しますよね。

サーバも同じです。サーバを使うときは「ターミナル」という操作するための画面を開いて、ディレクトリ^{*31}を開いて移動するときはダブルクリックの代わりに cd^{*32} というコマンドを叩いて移動しますし、ディレクトリの中を見るときはもダブルクリックの代わりに ls コマンドを叩いて見ます。

皆さんがいま使っている Windows や Mac といった「パソコン」だったらマウスやキーボードを使ってアイコンやボタンを見ながら操作できますが、サーバは基本的にこの真っ黒な「ターミナル」で文字を打って操作します。パソコンのときはダブルクリックやボタンを押す、という形で伝えていた命令がすべてコマンドに置き換わっていると思ってください。

パソコンもないのにマウスやキーボードだけあっても意味が無いように、ターミナルもそれ単体では何もできません。操作対象であるサーバがあつて初めて役に立つ道具なのです。

ちなみにターミナルは背景の色も文字の色も好きに変えられます。どうしても「黒い画面怖い！」という感覚が抜けない人は、ピンクとかオレンジとか好きな色にしてみましょ

*³¹ Linux ではフォルダのことをディレクトリと呼びます。

*³² change directory の略。

う。³³

まとめるとターミナルとはサーバを操作するための画面で、操作するときにはコマンドという命令を使います。

³³ Mac のターミナルはそもそも黒じゃなくて白ですね。

第4章

ウェブサーバの設定をしよう

この章ではウェブサーバの設定を行います。Apache でバーチャルホストを作つて WordPress のサイトを表示してみましょう。

4.1 ウェルカムページを見てみよう

第3章「AWSでサーバを立てよう」でサーバにApacheをインストールしました。インストールしただけでもまだ何の設定もしていませんが、ウェルカムページと呼ばれるデフォルトのページが見られるはずです。しかしブラウザで「<http://login.自分のドメイン/>」を開いてウェルカムページを見ようとしたところ、「ページ読み込みエラー 接続がタイムアウトしました」「このサイトにアクセスできません。login.自分のドメインから応答時間が長すぎます。」といったエラーメッセージが表示（図4.1）されてページを見るすることはできませんでした。



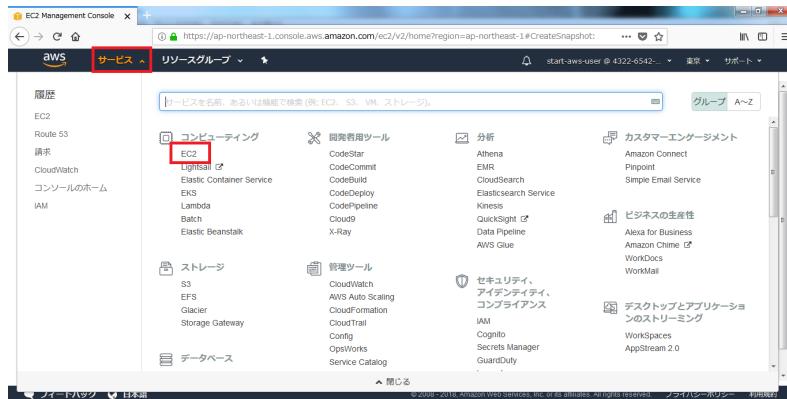
▲図4.1 <http://login.自分のドメイン/> が見られなかった

これはインスタンスの手前に居るセキュリティグループ^{*1}が「SSH（ポート番号22番）の通信しか通さない」という設定になっていて、HTTP（ポート番号80番）のリクエストを遮断していることが原因です。ポート番号とは、セキュリティグループという防火壁やサーバという家についているドアのようなものだと思ってください。同じサーバを訪問するときでもSSHは22番のドアを通るし、HTTPは80番のドアを通るし、HTTPSは443番のドアを通ります。

^{*1} セキュリティグループはいわゆる「ファイアウォール」のことです。セキュリティグループってどんなものだったっけ？ という方は第3章「AWSでサーバを立てよう」でインスタンスを作るとき「ステップ6」で設定したことを思い出してください。

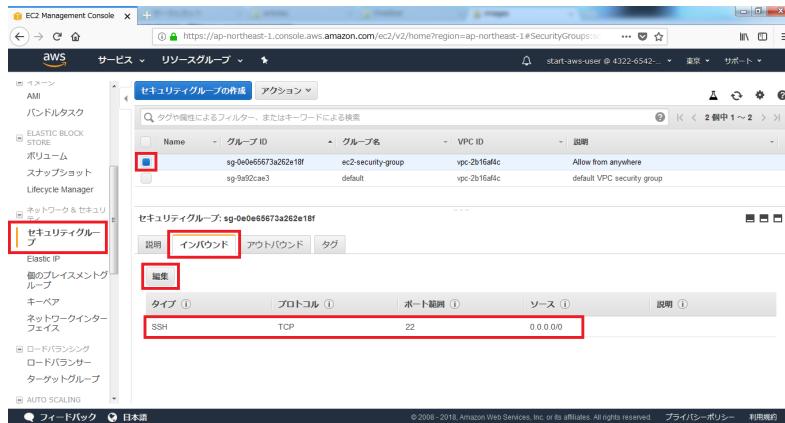
4.1.1 セキュリティグループで 80 番ポートの穴あけをしよう

ウェルカムページが見られるように、セキュリティグループで「HTTP（ポート番号 80 番）」の穴あけをしましょう。マネジメントコンソールの左上にある「サービス」から、「コンピューティング」の下にある「EC2」（図 4.2）をクリックしてください。



▲図 4.2 サービス>コンピューティング> EC2

「EC2」をクリックすると、EC2 ダッシュボード（図 4.3）が表示されます。左メニューの「セキュリティグループ」をクリックしてください。「ec2-security-group」をクリックして「インバウンド」タブを見ると、現状は通信を許可する対象に「SSH（ポート番号 22 番）」しか含まれていません。「インバウンド」タブの「編集」をクリックします。



▲図 4.3 EC2 ダッシュボードで「セキュリティグループ」をクリック

「ルールの追加」をクリックしたらタイプは「HTTP」を選択（図 4.4）します。ソースが「カスタム」になり「0.0.0.0/0, ::/0」と入力されるので、カンマから後ろを消して「0.0.0.0/0」にしてください。入力できたら「保存」をクリックします。



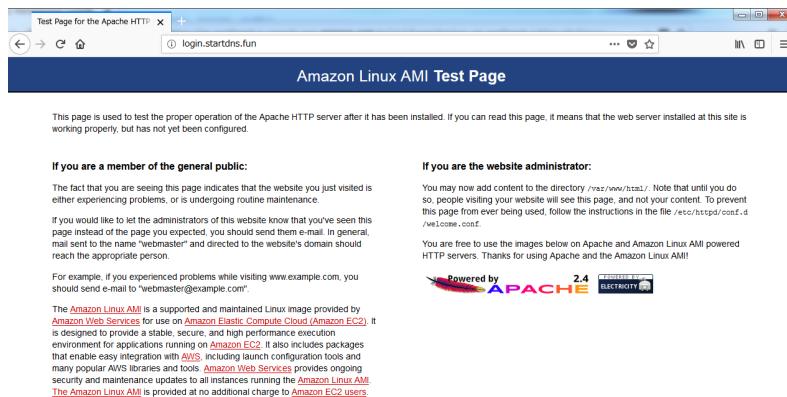
▲図 4.4 タイプは「HTTP」、ソースは「0.0.0.0/0」にして「保存」をクリック

「インバウンド」タブで HTTP が追加（図 4.5）されていたら穴あけ作業は完了です。



▲図 4.5 「インバウンド」タブで HTTP が追加されていたら穴あけ完了

もう一度ブラウザで「`http://login. 自分のドメイン/`」を開いてみましょう。「Amazon Linux AMI Test Page」と書かれたウェルカムページが表示（図 4.6）されるはずです。



▲図 4.6 Amazon Linux AMI Test Page が表示された

4.2 自分のサイト用にバーチャルホストを作ろう

ここからは自分のサイトの「バーチャルホスト」を作ります。

4.2.1 バーチャルホストとは？

「バーチャルホスト」という言葉を聞いたことはありますか？早く作業をしたいと思うので、ちょっと正確さは欠きますが分かりやすさ最優先で端的に説明します。

まずバーチャルホストの「ホスト」というのはサーバのことです。実は昔は1つのホストで1つのウェブサイトしか動かすことができませんでした。1つウェブサイトを作るたびに、1台のホスト、つまりサーバを立てなければならなかった、ということです。

AWSなら1台立てるのもあつという間ですが、第1章「インフラとサーバってなに？」で詳しく説明したように昔は「サーバを立てる」のにたくさんのお金や時間や労力が必要だったため、小さいウェブサイトを1つ作るたびに1台ホストを用意する、というのはなかなか難しいことでした。

そこで1台のホストの中にバーチャルなホストを何個も用意^{*2}して、ウェブサイトがいくつも同居できる機能が生まれました。それこそが「バーチャルホスト」です。

今はApacheにこの「バーチャルホスト」という機能があるので、1台のサーバ上で2つ以上のウェブサイトを同居させることができます。

4.2.2 バーチャルホストを設定しよう

それでは早速、自分のドメイン（筆者ならstartdns.fun）のサイト用にバーチャルホストを作りましょう。^{*3}

Windowsの方はRLoginを起動して「start-aws-instance」に接続してください。Macの方はターミナルで次のコマンドを実行してください。

```
ssh ec2-user@login. 自分のドメイン名 -i ~/Desktop/start-aws-keypair.pem
```

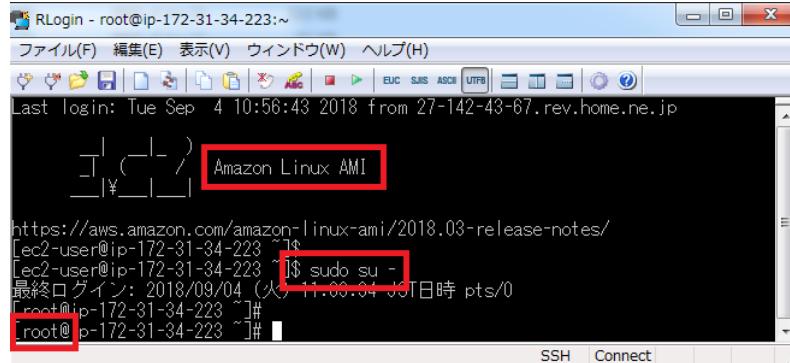
「Amazon Linux AMI」と表示されたら「sudo su -」^{*4}でrootになります。（図4.7）

^{*2} マンションの建物（ホストサーバ）の中に101号室や102号室などの各戸（仮想サーバ）があり、その中にはそれぞれ子供部屋や書斎や居間などの部屋（バーチャルホスト）がある、という例えなら分かりやすいでしょうか。

^{*3} サーバ1台にウェブサイト1つだけであればバーチャルホストにする必要はありませんが、バーチャルホストの作り方を覚えておけば後で「サブドメインで別のサイトを作りたい」と思ったときに役立つはずです。

^{*4} sudoは「他のユーザとしてコマンドを実行する」ためのコマンドで、suは「他のユーザになる」ためのコマンドです。「他のユーザ= root」の場合はユーザ名を書かなくてもいいので省略していますが、省略せずに書くと「sudo -u root su - root」（rootとして「rootになる」というコマンドを実行する）ということです。ちなみに勘違いされることが多いですがsuは「Super User」ではなく「Substitute User（ユーザーを代用する）」の略です。

```
$ sudo su -
```



▲図 4.7 Amazon Linux AMI と表示されたら root になろう

Apache の大本となる設定ファイルは「/etc/httpd/conf/httpd.conf」です。350 行以上あるので tail コマンドを使って最後の 5 行だけ確認してみましょう。

```
# tail -5 /etc/httpd/conf/httpd.conf
#
# Supplemental configuration
#
# Load config files in the "/etc/httpd/conf.d" directory, if any.
IncludeOptional conf.d/*.conf
```

大本の設定ファイルの中で、さらに「conf.d」というディレクトリに中ににある「*.conf」というファイルをインクルード（取り込み）^{*5}していることが分かります。でもいきなり「conf.d/*.conf」と書かれても「3 丁目」とだけ書かれた住所のようなもので、どこの「conf.d/*.conf」を指しているのかよく分かりませんよね。「conf.d」より上のディレクトリはどこで指定しているのか、grep という検索のコマンドで探してみましょう。

```
# grep "ServerRoot" /etc/httpd/conf/httpd.conf
ServerRoot "/etc/httpd"
```

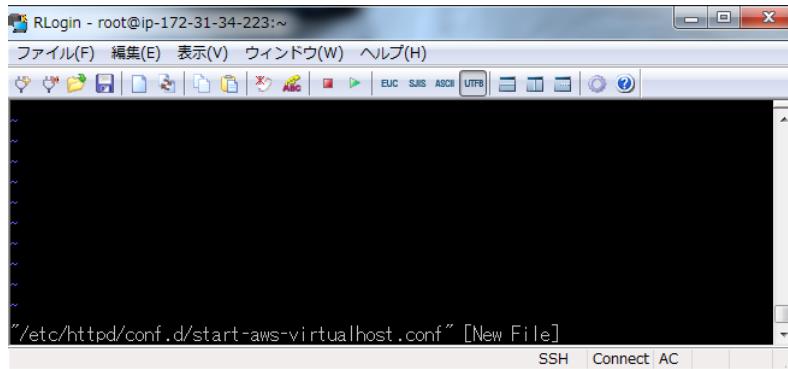
^{*5} Include ディレクティブや IncludeOptional ディレクティブについては <https://httpd.apache.org/docs/2.4/ja/mod/core.html#include> を参照。

ServerRoot^{*6}ではベースとなるディレクトリを指定しています。これで「conf.d/*.conf」は実際は「/etc/httpd/conf.d/*.conf」であることが分かりました。

では自分のドメインのサイト用にバーチャルホストを「/etc/httpd/conf.d」の下で作成してみましょう。vi コマンドで新しい設定ファイルを作ります。

```
# vi /etc/httpd/conf.d/start-aws-virtualhost.conf
```

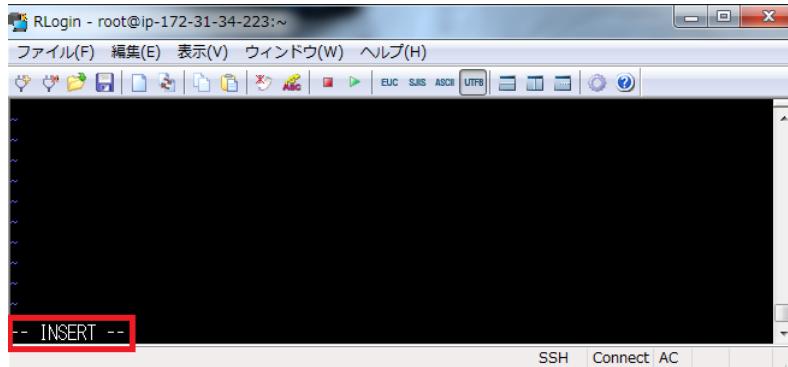
こんな画面（図4.8）で「"/etc/httpd/conf.d/start-aws-virtualhost.conf" [New File]」と表示されましたか？



▲図4.8 viでバーチャルホストの設定ファイルを作ろう

この状態で i (アイ) を押すと「編集モード」に変わります。（図4.9）左下に「-- INSERT --」と表示されていたら「編集モード」です。

^{*6} ServerRoot ディレクトリについては <https://httpd.apache.org/docs/2.4/ja/mod/core.html#serverroot> を参照。



▲図 4.9 i (アイ) を押すと「-- INSERT --」と表示される「編集モード」になった

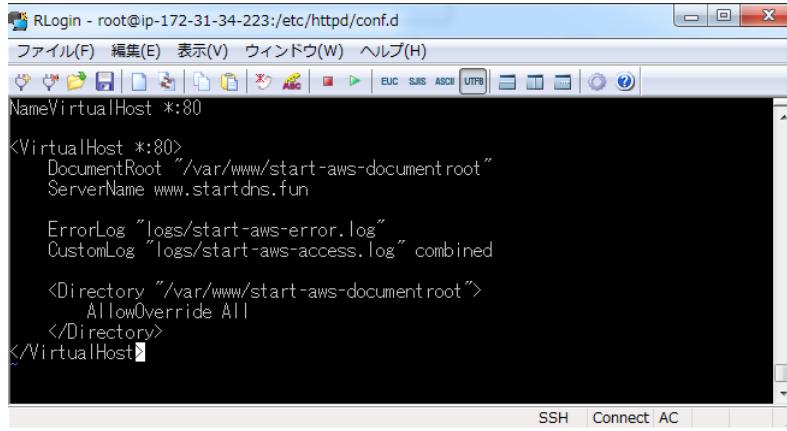
「編集モード」になったら次のようにバーチャルホストの設定を書いていってください。
「自分のドメイン」のところはそのまま日本語で書かず、自分のドメイン名に置き換えて
ください。

```
<VirtualHost *:80>
    DocumentRoot "/var/www/start-aws-documentroot"
    ServerName www. 自分のドメイン

    ErrorLog "logs/start-aws-error.log"
    CustomLog "logs/start-aws-access.log" combined

    <Directory "/var/www/start-aws-documentroot">
        AllowOverride All
    </Directory>
</VirtualHost>
```

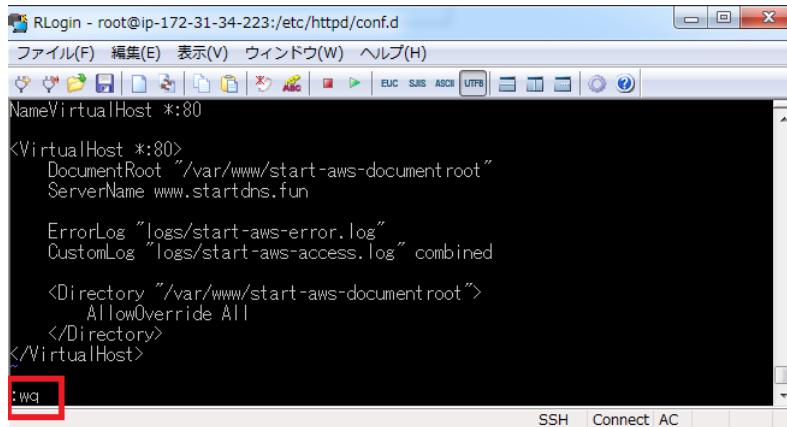
「編集モード」のままだと保存ができないので書き終わったら ESC キーを押します。
(図 4.10) すると左下の「-- INSERT --」が消えて再び「閲覧モード」になります。



```
RLogin - root@ip-172-31-34-223:/etc/httpd/conf.d
ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)
名前 VirtualHost *:80
<VirtualHost *:80>
    DocumentRoot "/var/www/start-aws-documentroot"
    ServerName www.startdns.fun
    ErrorLog "logs/start-aws-error.log"
    CustomLog "logs/start-aws-access.log" combined
    <Directory "/var/www/start-aws-documentroot">
        AllowOverride All
    </Directory>
</VirtualHost>
```

▲図 4.10 ESC を押すと左下の「-- INSERT --」が消えて再び「閲覧モード」になる

「閲覧モード」に戻ったら「:wq」と入力して Enter キーを押せば保存されます。(図 4.11)



```
RLogin - root@ip-172-31-34-223:/etc/httpd/conf.d
ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)
名前 VirtualHost *:80
<VirtualHost *:80>
    DocumentRoot "/var/www/start-aws-documentroot"
    ServerName www.startdns.fun
    ErrorLog "logs/start-aws-error.log"
    CustomLog "logs/start-aws-access.log" combined
    <Directory "/var/www/start-aws-documentroot">
        AllowOverride All
    </Directory>
</VirtualHost>
:wq
```

▲図 4.11 ESC を押すと左下の「-- INSERT --」が消えて再び「閲覧モード」になる

では確認のため、次のコマンドを叩いてみてください。

```
# cat /etc/httpd/conf.d/start-aws-virtualhost.conf
<VirtualHost *:80>
    DocumentRoot "/var/www/start-aws-documentroot"
    ServerName www.startdns.fun
```

```

ErrorLog "logs/start-aws-error.log"
CustomLog "logs/start-aws-access.log" combined

<Directory "/var/www/start-aws-documentroot">
    AllowOverride All
</Directory>
</VirtualHost>

```

「ServerName」のところが日本語の「www.自分のドメイン」ではなく、ちゃんと自分のドメインに置き換わっているか確認してください。たとえば筆者なら「startdns.fun」というドメインなので「ServerName www.startdns.fun」になっています。もし cat コマンドを叩いたときに「そのようなファイルやディレクトリはありません」と表示されたら設定ファイルが作成できていません。作り直しましょう。

4.2.3 設定ファイルの構文チェック

バーチャルホストの設定ファイルが書けたので apachectl で構文チェックをしてみましょう。apachectl は Apache を操作するためのコントローラのようなもので、引数に configtest とつけてやると設定ファイルの構文チェックができます。

```
# apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/start-aws-documentroot] does not exist
Syntax OK
```

早速警告のメッセージが表示されています。落ち着いて読んでみましょう。「DocumentRoot [/var/www/start-aws-documentroot] does not exist」と書いてあります。これは「ドキュメントルートに [/var/www/start-aws-documentroot] というディレクトリを指定しているけどそんなディレクトリ存在しないよ」と言っているのです。

4.2.4 ドキュメントルートを作成

ドキュメントルート^{*7}とは「サイトにアクセスしたらここに置いたファイルが表示されるよ」というディレクトリのことです。

```
DocumentRoot "/var/www/start-aws-documentroot"
ServerName www.startdns.fun
と書いてあったら、
```

^{*7} <https://httpd.apache.org/docs/2.4/ja/mod/core.html#documentroot>

```
/var/www/start-aws-documentroot/ に置いた「index.html」が  
http://www.startdns.fun/index.html で見られる
```

先ほどバーチャルホストの設定で次のように書いたのですが、このディレクトリがまだ存在していないため警告が出てしまっているのです。ですのでこのディレクトリを作成しましょう。

```
DocumentRoot "/var/www/start-aws-documentroot"
```

ディレクトリを作るには `mkdir`*8 というコマンドを使います。`mkdir` コマンドでディレクトリを作ったら、ちゃんと作成できたか `ls` コマンドで確認してみましょう。

```
# mkdir /var/www/start-aws-documentroot  
  
# ls -l /var/www/  
合計 24  
drwxr-xr-x 2 root root 4096 8月 18 07:22 cgi-bin  
drwxr-xr-x 3 root root 4096 9月 1 17:43 error  
drwxr-xr-x 2 root root 4096 8月 18 07:22 html  
drwxr-xr-x 3 root root 4096 9月 1 17:43 icons  
drwxr-xr-x 2 root root 4096 9月 1 17:43 noindex  
drwxr-xr-x 2 root root 4096 9月 4 12:53 start-aws-documentroot
```

これでドキュメントルートができました。再び `apachectl` で構文チェックをしてみましょう。今度は「Syntax OK」とだけ表示されるはずです。

```
# apachectl configtest  
Syntax OK
```

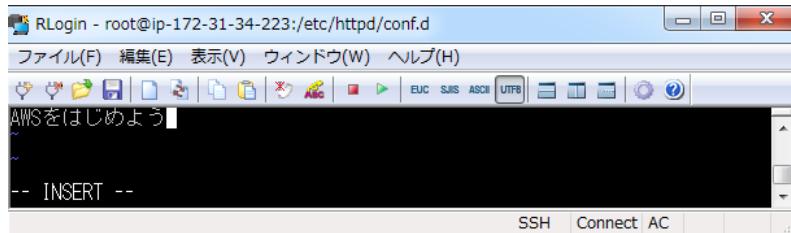
4.2.5 index.html を置いてみよう

ドキュメントルートを用意したのですが、何かしらコンテンツを置いておかないとアクセスしたときに「404 Not Found」になってしまうので、`vi` コマンドでドキュメントルートの下に `index.html` のファイルを用意しておきましょう。

*8 `mkdir` は MaKe DIRectory の略。

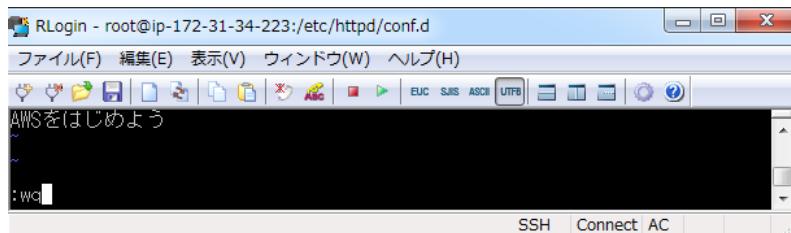
```
# vi /var/www/start-aws-documentroot/index.html
```

i (アイ) を押して「編集モード」に変わったら「AWS をはじめよう」と書いてみましょう。(図 4.12)



▲図 4.12 i (アイ) を押して「-- INSERT --」と表示されたら「AWS をはじめよう」と書いてみよう

書き終わったら ESC キーを押して「閲覧モード」に戻り、「:wq」と入力して Enter キーを押して保存しましょう。(図 4.13)



▲図 4.13 ESC を押すと「閲覧モード」に戻ったら「:wq」で保存

それではバーチャルホストの設定を有効にするため、apachectl で Apache を再起動しましょう。

```
# apachectl restart
```

何のメッセージも表示されなければ問題なく Apache が再起動できています。

4.2.6 curl でページを確認しよう

これでバーチャルホストの設定は完了です。バーチャルホストで作った自分のウェブサイトに「ページ見せて」と頼んだら、ちゃんとページを返してくれるのか確認してみましょう。次のようなコマンドを叩いてみてください。ちなみに「www. 自分のドメイン」の部分は自分のドメインに置き換えます。たとえば筆者なら「startdns.fun」というドメインなので「www.startdns.fun」にします。

```
# curl -H "Host:www. 自分のドメイン" http://localhost/
AWS をはじめよう
```

これは curl (カール) という「ターミナルにおけるブラウザ」のようなコマンドを使って、localhost (自分自身) の「www. 自分のドメイン名」というホストに対して「ページ見せて」というリクエストを投げています。ちゃんと自分のバーチャルホストが応答して、ドキュメントルートにある index.html の「AWS をはじめよう」が表示されましたね。

4.3 Route53 で自分のサイトのドメインを設定しよう

続いてブラウザでも「http://www. 自分のドメイン」を開いてサイトを確認してみましょう。(図 4.14) なんと「アクセスしようとしているサイトを見つけられません。」と表示されてしまいました。

4.3 Route53 で自分のサイトのドメインを設定しよう

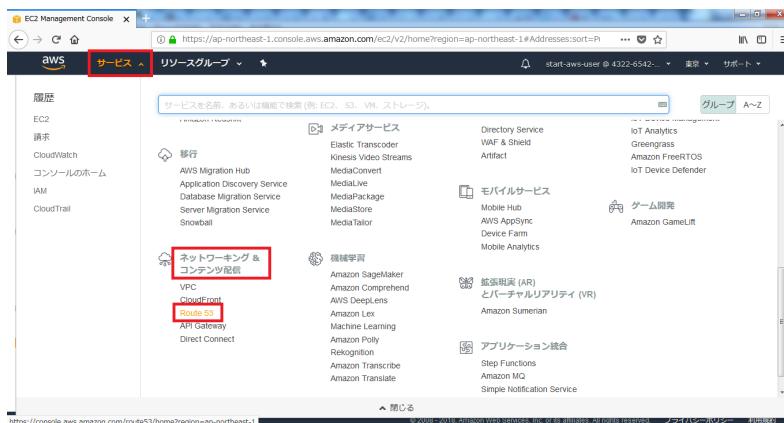


▲図 4.14 ブラウザで「www. 自分のドメイン」を開いたらエラーになってしまった

これはまだ「www. 自分のドメイン」というドメイン名とサーバの IP アドレスをつなぐ A レコードが存在していないからです。Route53 で A レコードを作りましょう。

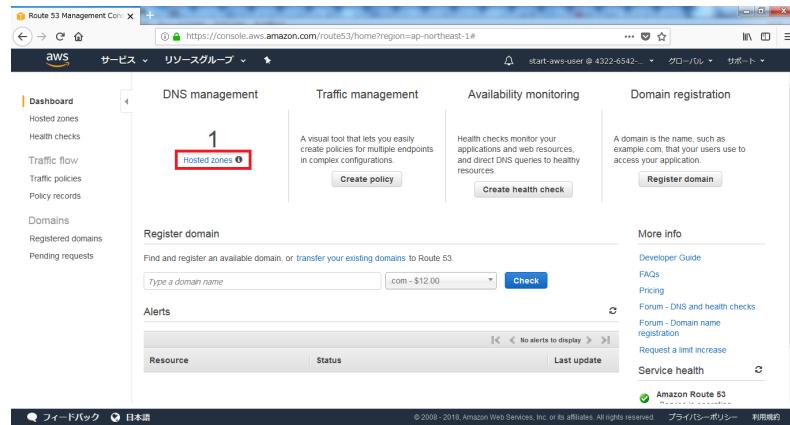
4.3.1 www のサブドメインを作ろう

マネジメントコンソールの左上にある「サービス」から、「ネットワーキング & コンテンツ配信」の下にある「Route53」(図 4.15) をクリックしてください。



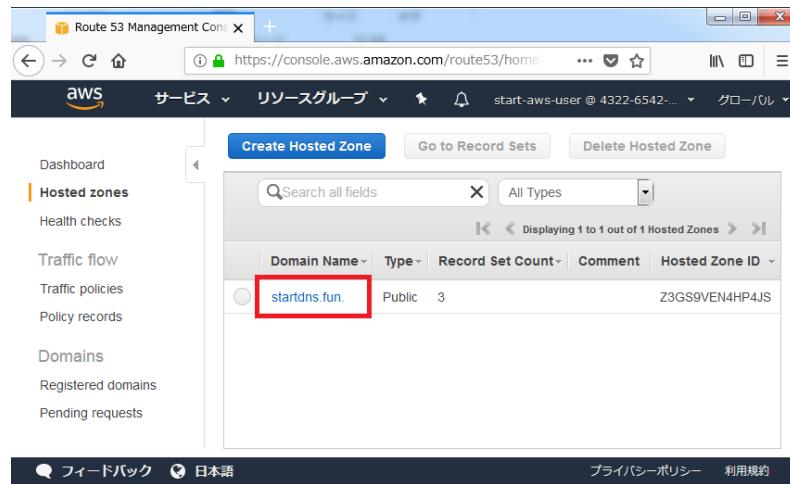
▲図 4.15 サービス>ネットワーキング & コンテンツ配信>Route53

Route53 ダッシュボードを開いたら DNS management の「Hosted zones」をクリック（図 4.16）します。



▲図 4.16 「Hosted zones」をクリック

Domain Name の自分のドメイン名（私の場合は startdns.fun）をクリック（図 4.17）します。

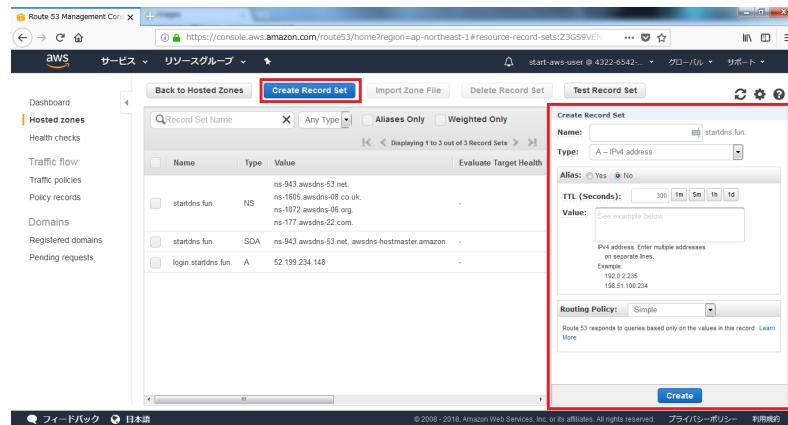


▲図 4.17 自分のドメイン名をクリック

「Create Record Set」をクリック（図 4.18）してください。すると右側にリソースレ

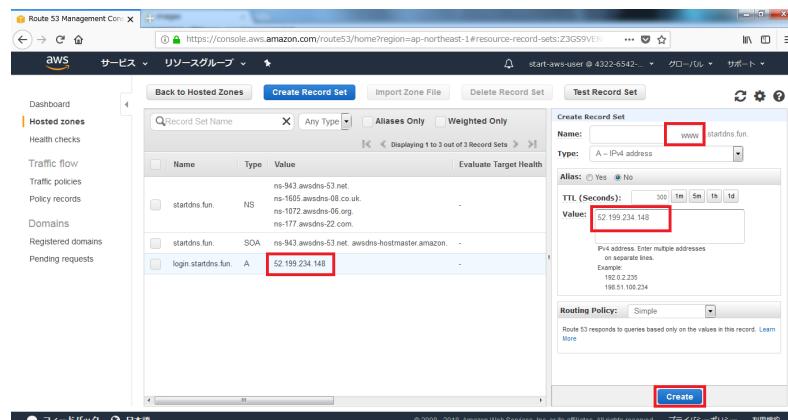
4.3 Route53 で自分のサイトのドメインを設定しよう

コードの情報を入力するフォームが出てきます。



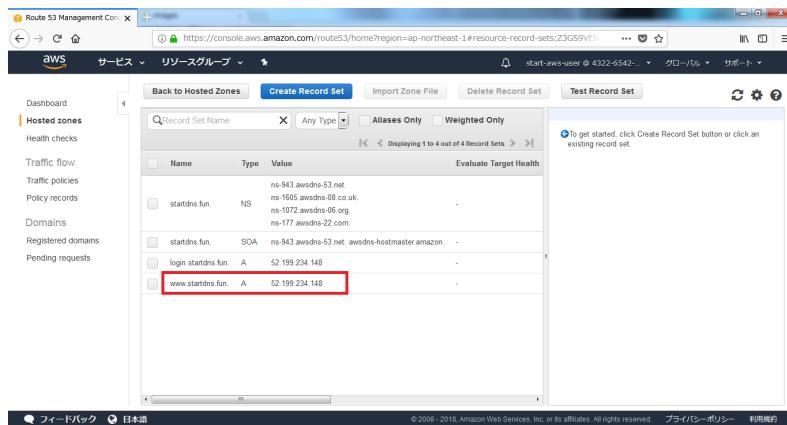
▲図 4.18 「Create Record Set」をクリック

Name には「www」、Value には Elastic IP を入力（図 4.19）します。Elastic IP は左側の「login. 自分のドメイン名」のところにも書いてあるので、それをコピーしてきても構いません。入力できたら「Create」をクリックします。



▲図 4.19 Name には「www」、Value には login と同じ Elastic IP を入力

これで「www. 自分のドメイン名」（図 4.20）という A レコードが作成できました。



▲図 4.20 「www. 自分のドメイン名」という A レコードができた

4.3.2 ブラウザでページを見てみよう

A レコードの追加が終わったら再びブラウザで「http://www. 自分のドメイン」を開いてみましょう。（図 4.21）今度こそ index.html の「AWS をはじめよう」が表示されました。おめでとうございます！



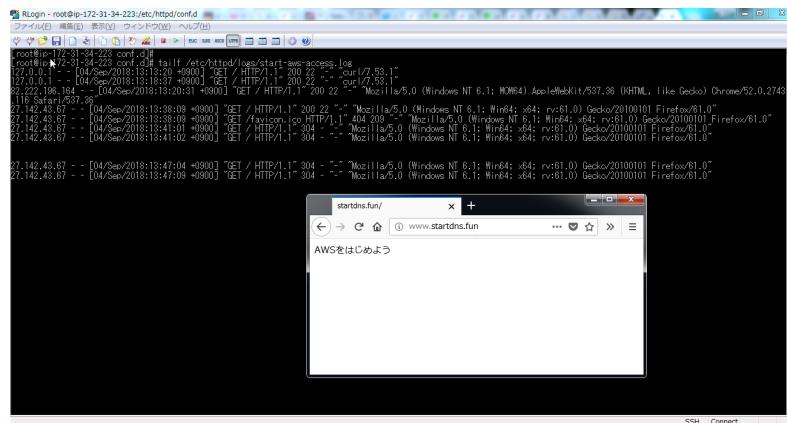
▲図 4.21 index.html の「AWS をはじめよう」が表示された

4.3.3 アクセスログとエラーログの大切さ

画面上は表示されていますが、サーバ側でもアクセスログを確認してみましょう。

```
# tailf /etc/httpd/logs/start-aws-access.log
```

tailf コマンド^{*9}はファイルの追記を監視できるコマンドなので、今来ているアクセスのログをタイムリーに確認できます。何回か Enter キーを叩いて改行したら、この状態でブラウザの更新ボタンをクリックしてみましょう。自分がいまブラウザでアクセスしたログが次々と表示されると思います。（図 4.22）



▲図 4.22 tailf でログを確認しながらブラウザでサイトの表示を更新してみよう

念のためエラーログも確認してみましょう。アクセスログ（1.6K）に対してエラーログはファイルサイズが 0 です。エラーログは 1 行も出ていないようですので問題ありません。

```
# ls -lh /etc/httpd/logs/start-aws-*
-rw-r--r-- 1 root root 1.6K 9月 4 13:51 /etc/httpd/logs/start-aws-access.log
-rw-r--r-- 1 root root 0 9月 4 13:07 /etc/httpd/logs/start-aws-error.log
```

^{*9} tail コマンドに-f オプションをつけてもまったく同じ動作をします。

たとえばサイトが上手く表示されないとき、アクセスログやエラーログを確認すれば「サーバまでたどり着いていない」のか、「サーバにはたどり着いているけれど、ドキュメントルートに置いた PHP ファイルのエラーでちゃんとページが出ない」のかといった問題の切り分けができます。上手くいかないときはログを見るようにしましょう。

第 5 章

DB サーバを立てよう

この章では WordPress で使用する MySQL の設定をします。

5.1 RDS で WordPress 用のインスタンスを立てよう

5.1.1 パラメータグループを作成

5.1.2 RDS インスタンスの作成

5.1.3 ウェブサーバから接続確認してみよう

第 6 章

WordPress でサイトを作ろう

この章では WordPress をインストールしてウェブサイトを作ります。いよいよおしゃれなサイトが見られます。

6.1 WordPressのインストール

6.2 管理画面にログイン

6.2.1 管理画面にダイジェスト認証をかけよう

第7章

サーバのバックアップを取っておこう

この章では AMI と EBS スナップショットを使ってサーバのバックアップをします。サーバが壊れたときに助けてくれる「バックアップ」の存在。今はあまりピンとこないかもしれません、大切なものがなくなったときにありがたみを知るはずです。

7.1 EBS スナップショットとAMI

第3章「AWSでサーバを立てよう」ではEC2で立てたインスタンスにログインして色々な設定をしました。インスタンスのOSやインストールしたApache、修正したさまざまなファイルはEBSボリューム^{*1}に保存されています。ある日うっかり「rm -rf /etc/*^{*2}を叩いてしまって「折角設定したファイルが全部消えちゃった・・・バックアップもない・・・」と泣くことのないよう、EBSボリューム（つまりハードディスクに保存されたデータ）のバックアップを取っておきましょう。

バックアップにはAmazon EBSのスナップショットというサービスを使用します。スナップショットは「ある時点」のEBSボリュームを丸ごと保存しておいてくれるので、うっかりデータを全削除してしまってもスナップショットを取っていた時点まで戻すことができます。

ただしスナップショットを取るときはデータの整合性を保つためにハードディスクへの読み書きを停止する必要があります。インスタンスを起動して使用しているままでもスナップショットを取ることはできますが、たとえば「MySQL（データベース）がトランザクション処理中でゴリゴリ書き込みをしている最中に取るとデータの整合性が取れない」といった可能性があり、ファイルシステムの完全性が保証できないので推奨されていません。ハードディスクへの読み書きがされない状態、つまりインスタンスを先にシャットダウンしておくか、もしくは対象となるEBSボリュームをアンマウント^{*3}してからスナップショットを取るようにしましょう。

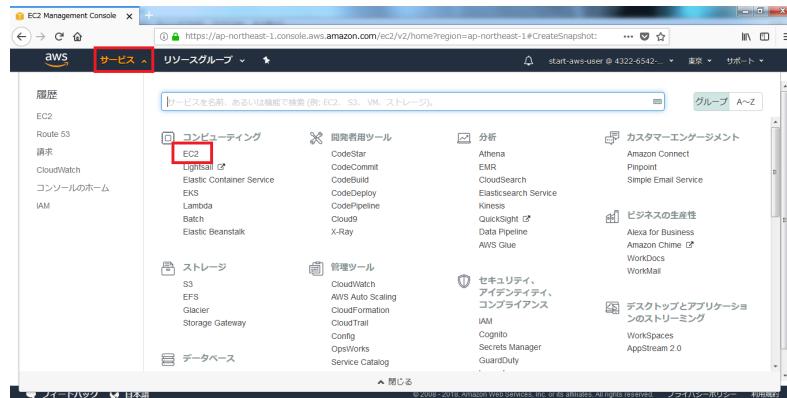
マネジメントコンソールの左上にある「サービス」から、「コンピューティング」の下にある「EC2」（図7.1）をクリックしてください。

^{*1} EBSボリュームってなんだっけ？ という方は、インスタンスを作るとき「ステップ4」でEBSボリュームの種類やサイズを選んだことを思い出してください。ちなみにOSがEBSボリュームではなく「インスタンスストア」というところにインストールされているインスタンスもあるのですがここではその説明は割愛します。

^{*2} ファイルを削除するrmコマンドにrecursive（再帰的）の-rオプションとforce（強制的）の-fオプションをつけて、対象ディレクトリは設定ファイルがたくさん詰まった「/etc/」を指定しているので「/etc/以下にあるすべてのファイルを強制的に削除しろ」という意味のコマンドです。

^{*3} ハードディスクをサーバから取り外すこと。

7.1 EBS スナップショットと AMI



▲図 7.1 サービス>コンピューティング> EC2

「EC2」をクリックすると、EC2 ダッシュボード（図 7.2）が表示されます。左メニューの「インスタンス」をクリックしてください。



▲図 7.2 EC2 ダッシュボードで「インスタンス」をクリック

第3章「AWS でサーバを立てよう」でインスタンスを作るとき、ステップ1で Amazon マシンイメージ、略して AMI を選択したのを覚えていますか？先ほどは AMI というテンプレートのようなものを元にインスタンスを作りましたが、その逆でインスタンスから自分専用のカスタム AMI を作ることもできます。

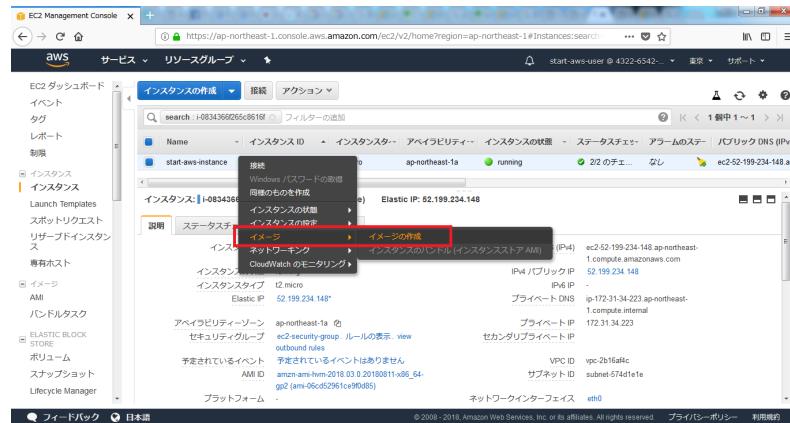
インスタンスを AMI で丸ごとテンプレート化しておけば、その AMI からインスタンスをいくつも複製できます。あるいは間違ってインスタンスを削除してしまったときに、

AMI からインスタンスを作り直すことができます。

EBS スナップショットと AMI の関係がちょっと分かりにくいかも知れませんので、実際にやりながら理解していきましょう。

7.2 インスタンスから AMI を作ろう

「start-aws-instance」というインスタンスを右クリック（図 7.3）して、「イメージ」の「イメージの作成」をクリックします。



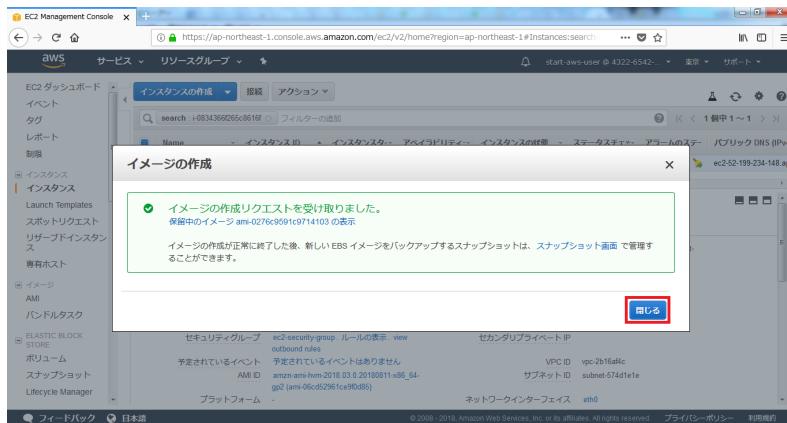
▲図 7.3 インスタンスを右クリックして「イメージの作成」をクリック

イメージ名に「start-aws-ami」と入力したら「イメージの作成」をクリック（図 7.4）してください。



▲図 7.4 イメージ名に「start-aws-ami」と入力したら「イメージの作成」をクリック

「イメージの作成リクエストを受け取りました。」と表示（図 7.5）されたら「閉じる」をクリックしてください。



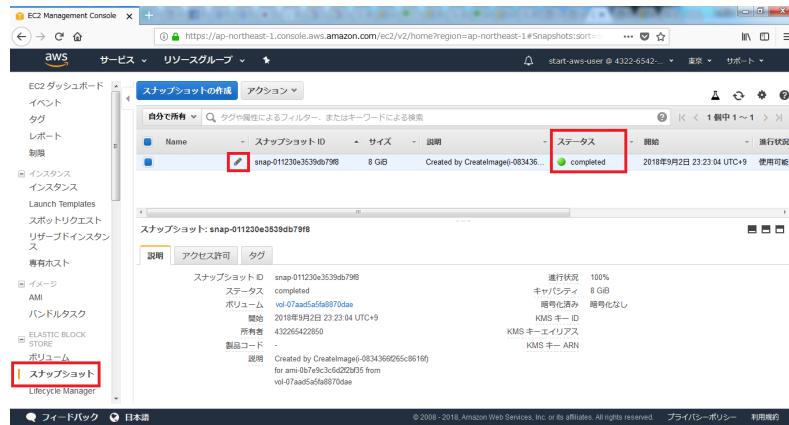
▲図 7.5 「閉じる」をクリック

この「インスタンスを元に AMI を作成する」という作業をすると、実際は裏側で

1. インスタンスを停止する
2. インスタンスに紐づいている EBS ボリュームの EBS スナップショットを取る
3. その EBS スナップショットを元に AMI を作る
4. インスタンスを起動する

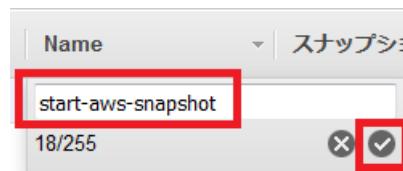
という複数の処理が実行^{*4}されています。

左メニューで「スナップショット」をクリック（図7.6）してEBSスナップショットが生成されたか確認してみましょう。ステータスが「completed」になっていればEBSスナップショットの取得は完了しています。このEBSスナップショットに名前を付けておきましょう。Nameのところにカーソルを持っていくと鉛筆のマークが表示されますのでクリックしてください。



▲図7.6 左メニューで「スナップショット」をクリックしてEBSスナップショットを確認

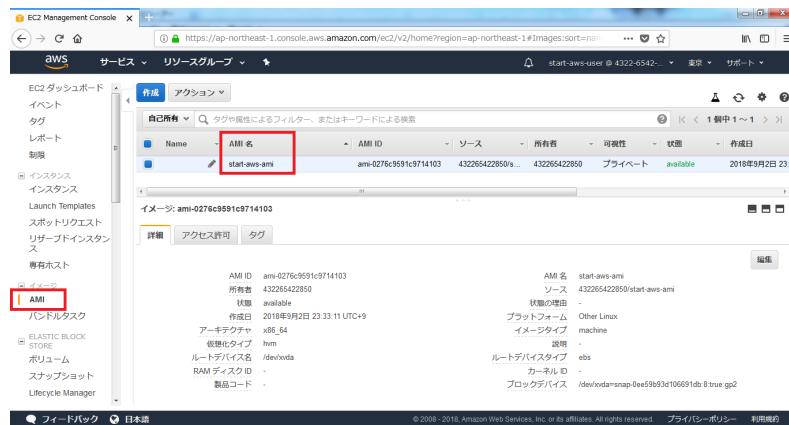
Nameに「start-aws-snapshot」と書いたらチェックボタンを押して（図7.7）ください。スナップショット名には日本語を使わないことをお勧めします。



▲図7.7 作成したインスタンスのNameのところにある鉛筆マークをクリック

続いて左メニューで「AMI」をクリック（図7.8）してAMIが作成されたか確認してみましょう。「start-aws-ami」というAMIが作成されていますね。

^{*4} 当たり前ですがインスタンスを停止している間、その上で動いていたウェブサイトは見られなくなります。AMIを作成する間（1～4の間）はサイトが落ちてしまう、ということです。



▲図 7.8 左メニューで「AMI」をクリックして AMI を確認

今後もしインスタンスを壊したりなくしたりして復元したくなったら、AMI を右クリックして「作成」をクリック（図 7.9）すればこの AMI からインスタンスを作成できます。



▲図 7.9 インスタンスを復元したくなったら AMI を右クリックして「作成」をクリック

第 8 章

ELB でバランスингやサーバの台数管理をしよう

この章ではロードバランサーの設定や、サーバが停止してしまったときに自動復旧するオートスケールの設定をします。

8.1 ELB

8.2 Auto Scaling

8.2.1 スケーリングに使える

8.2.2 サーバが1台死んでも自動で1台立ち上がる

8.3 Route53 で A レコードを変更して Alias にしよう

8.3.1 ページを確認してみよう

8.4 セキュリティグループの変更

8.4.1 ELB 経由の 80 番ポートを許可しよう

左上のサービス>コンピューティング>EC2を開いて、左メニューの「セキュリティグループ」を開きます。

EC2SecurityGroupを選択して、「インバウンド」タブの「編集」を押します。

「ルールの追加」を押して、タイプで「HTTP」を選択、ソースを「カスタム」にしてELBと入力すると、下に「ELBSecurityGroup」が表示されるので、それを選択してください。

SSHとHTTP、2つルールがあることを確認したら「保存」ボタンを押して、ルールが追加されたことを確認します。これによってELBを通ってきたHTTPも通すようになりました。

再度、ブラウザで自分のドメイン名でサイトを見てみましょう。<http://自分のドメイン名/>

私だったら<http://nekonekoekoneko.xyz/>にアクセスすると、「nekonekonekoneko.xyz index page」というindex.htmlの内容が表示されます。

ちなみにヘルスチェックは、30秒おきに「生きてる?」というチェックをして、5回連続で成功すれば合格ですので、HTTP/HTTPSを許可してから2分半待たないと見られないと思います。

では無事サイトも見られたので、今日の作業のおさらいをします。

8.4.2 EIP の 80 番ポートは閉じておこう

第 9 章

もっと AWS について勉強したい！

本著を読んでもっと AWS や Linux について勉強したくなったあなたにお勧めの資料を紹介します。

9.1 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/>

9.2 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 9.1）^{*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。



▲図 9.1 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？

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

9.3 Linux やコマンドも学びたいなら

AWS に限らず Linux やコマンドについてもっと学びたいときは「Linux 教科書 LPIC レベル 1」^{*2}という本がお勧めです。こちらは LPIC という資格試験の教科書なのですが、Linux の基礎的な内容を網羅しているので私も何かあるとこの本をすぐに開いて確認しています。

難しい内容だと挫折してしまいそう・・・という人には「まんがでわかる Linux システム女子」^{*3}がお勧めです。Linux なんてまったく分からぬ素人なのにシステム部に配属されてしまった「みんとちゃん」が、素朴な疑問をどんどん先輩にぶつけてくれるので、初心者の「それが知りたかった！」という内容が詰まっています。

旬な内容なら雑誌の Software Design^{*4}や WEB+DB PRESS^{*5}がお勧めです。特に 4 月ごろは「新人のための Linux 入門」や「Web 開発 基礎の基礎」など、新人向けのとつきやすい記事が多いのでバックナンバーを探してみてください。

^{*2} <https://www.amazon.co.jp/dp/4798141917>

^{*3} <https://system-admin-girl.com/>

^{*4} <https://gihyo.jp/magazine/SD>

^{*5} <https://gihyo.jp/magazine/SD>

第 10 章

AWS をやめたくなったらすること

この章では AWS を使うのをやめたくなったらしておくべき最後の手続きを紹介します。

10.1 無料の 1 年が終わる前にすべきこと

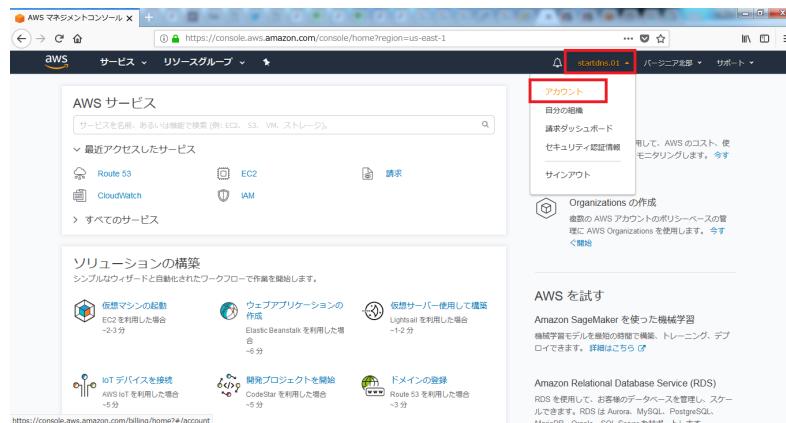
第 2 章「AWS を使い始めたら最初にやること」で書いたとおり、AWS アカウントを作成してから 1 年間は無料利用枠の範囲内であれば利用料が無料となります。

全然使わない月があったとしてもその分の無料利用枠が翌月以降に繰り越されることはありません。1 年経過後、無料利用枠の有効期限が切れるとき以降は通常の従量課金できっちり請求が来ますので、不要になったインスタンスやサービスは忘れずに削除してください。

何を使ったか忘れてしまって全部消せるか心配・・・という場合は、AWS のアカウントごと停止するという方法もあります。

10.1.1 AWS アカウントを停止する

ルートユーザーでサインインしたら、マネジメントコンソールの右上にあるルートユーザー名（図 10.1）から「アカウント」をクリックしてください。



▲図 10.1 ルートユーザ名>アカウント

アカウントのページを一番下までスクロールして「アカウントの解約」（図 10.2）の記載内容を確認してください。

いくつか注意点があります。たとえば EC2 や RDS のインスタンスを削除せずにアカウントを解約した場合、そのインスタンスは解約後すぐに消えるわけではありません。イ

ンスタンスが消えるタイミング^{*1}は AWS にゆだねられます。サイト自体をすぐにクローズしたいのであれば先にインスタンスを削除してからアカウントを解約するようにしましょう。

また月の途中で解約した場合、その日までの利用料は請求されます。たとえば 2018 年 9 月 4 日^{*2}にアカウントを解約した場合、9 月 1 日～4 日までの利用料は 10 月の初めに請求されます。解約後、90 日間はアカウントを再開（再有効化）できるようですが、その期間を過ぎると再開はできず、同じメールアドレスで新しいアカウントを作ることもできなくなります。

内容を確認し、同意して解約する場合はチェックボックスにチェックを入れて「アカウントの解約」をクリックします。



▲図 10.2 チェックボックスにチェックを入れて「アカウントの解約」をクリック

「本当にアカウントを解約してもよろしいですか？」と表示されるので、解約してよければ「アカウントの解約」をクリックします。（図 10.3）

^{*1} 検証していないので推測ですが「AWS アカウントに残されたコンテンツは、閉鎖後期間が過ぎると削除されます。」という記載がありますので、アカウント解約後も 90 日間はそのままなのかも知れません。

^{*2} 冒険とイマジネーションの海よ、17 周年おめでとう！



▲図 10.3 解約してよければ「アカウントの解約」をクリック

アカウント解約を知らせるメールが届き、以降はルートユーザーでも IAM ユーザーでもマネジメントコンソールにはサインインできなくなります。

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:

- プロシコリーが好きな茶色い猫に捧ぐ

レビュアー

- Takeshi Matsuba
- 深澤俊

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