

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、そこに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 やサーバやインフラは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？ という人

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？

をコラムやドリルで解説しています。

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
- クラウドがなんなのか？ や、その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、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？ | 13 |
| 1.1 AWSを理解するには先ずインフラを知ろう | 14 |
| 1.2 インフラとは？ | 14 |
| 1.3 サーバとは？ | 15 |
| 1.3.1 サーバの姿を見てみよう | 19 |
| 1.3.2 サーバはデータセンターにいる | 21 |
| 1.3.3 物理サーバと仮想サーバ | 25 |
| 1.4 オンプレミスとクラウド | 27 |
| 1.4.1 オンプレミスとは？ | 27 |
| 1.4.2 クラウドとは？ | 28 |
| 1.4.3 クラウドのメリットとデメリット | 28 |
| 1.4.4 AWSはAmazonがやっているクラウド | 30 |
| 1.4.5 パブリッククラウドとプライベートクラウド | 31 |
| 1.4.6 AWS以外のクラウド | 32 |
| 第2章 AWSを使い始めたら最初にやること | 33 |
| 2.1 AWS無料利用枠を使おう | 34 |
| 2.2 AWSのアカウント作成 | 35 |
| 【コラム】「DNSをはじめよう」はどこで買える？ | 36 |
| 2.3 マネジメントコンソールにサインイン | 36 |
| 2.3.1 【ドリル】AWSの管理画面はなんて名前？ | 38 |

| | | |
|------------|---|-----------|
| 2.4 | CloudWatch で請求アラートを設定しよう | 39 |
| 2.5 | IAM でユーザの権限管理 | 45 |
| 2.5.1 | ルートユーザーの普段使いはやめよう | 45 |
| 2.5.2 | IAM ユーザーを作ろう | 46 |
| 2.5.3 | MFA (多要素認証) で不正利用から IAM ユーザーを守る | 59 |
| 2.5.4 | ルートユーザーも MFA を有効にする | 72 |
| 2.6 | リージョンの変更 | 74 |
| 2.7 | CroudTrail でいつ誰が何をしたのか記録 | 76 |
| 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 | 85 |
| 3.2.1 | ステップ 1: Amazon マシンイメージ (AMI) | 86 |
| 3.2.2 | ステップ 2: インスタンスタイプの選択 | 87 |
| | 【コラム】T2 系バーストモードの落とし穴 | 89 |
| 3.2.3 | ステップ 3: インスタンスの詳細の設定 | 90 |
| 3.2.4 | ステップ 4: ストレージの追加 | 90 |
| 3.2.5 | ステップ 5: タグの追加 | 91 |
| 3.2.6 | ステップ 6: セキュリティグループの設定 | 92 |
| 3.2.7 | ステップ 7: インスタンス作成の確認 | 93 |
| 3.2.8 | キーペアのダウンロード | 94 |
| 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? | 99 |
| | 【コラム】サーバは「立てる」もの? 「建てる」もの? | 100 |
| 3.3 | サーバに「入る」とは? | 100 |
| 3.3.1 | SSH とは? | 102 |
| 3.3.2 | パスワード認証と鍵認証 | 103 |
| 3.3.3 | 接続先となるサーバの IP アドレス | 103 |
| 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 | 114 |
| 3.4.4 | 再起動しても変わらない ElasticIP をつけよう | 118 |

| | | |
|--------------|---|------------|
| 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 | 137 |
| 3.5.4 | ターミナルはなんのためにある？ | 145 |
| 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 | 152 |
| 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 | 158 |
| 4.2.4 | ドキュメントルートを作成 | 158 |
| 4.2.5 | index.html を置いてみよう | 159 |
| 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 | 161 |
| 4.3.1 | 自分のサイトのドメイン名を作ろう | 161 |
| 4.3.2 | 【ドリル】 / (スラッシュ) で終わる URL を開いたときに index.html 以外を返したい | 164 |
| 4.3.3 | ブラウザでページを見てみよう | 165 |
| 4.3.4 | アクセスログとエラーログの大切さ | 166 |
| 第 5 章 | データベースサーバを立てよう | 169 |
| 5.1 | WordPress にはデータベースが必要 | 170 |
| 5.1.1 | CMS とは? | 170 |
| | 【コラム】MySQL と MariaDB | 171 |
| 5.1.2 | EC2 にインストールするか? RDS を使うか? | 172 |
| 5.2 | インスタンスを立てる事前準備 | 172 |
| 5.2.1 | パラメータグループを作成 | 173 |
| 5.2.2 | パラメータの設定 | 176 |
| 5.2.3 | オプショングループを作成 | 181 |
| 5.3 | RDS でインスタンスを立てよう | 183 |
| 5.3.1 | ステップ 1:エンジンの選択 | 183 |
| 5.3.2 | ステップ 2:DB 詳細の指定 | 184 |

| | | |
|--------------|---|------------|
| 5.3.3 | ステップ 3:[詳細設定] の設定 | 185 |
| 5.3.4 | セキュリティグループで 3306 番ポートの穴あけをしよう | 187 |
| 5.3.5 | エンドポイントのドメイン名をメモしておこう | 191 |
| 5.3.6 | ウェブサーバから接続確認してみよう | 193 |
| 第 6 章 | WordPress でサイトを作ろう | 197 |
| 6.1 | WordPress のインストール | 198 |
| 6.1.1 | WordPress をダウンロード | 198 |
| 6.1.2 | 展開してドキュメントルートに設置 | 198 |
| 6.1.3 | サイトにアクセスしてインストール実行 | 200 |
| 6.2 | 管理画面にログイン | 204 |
| 6.2.1 | 【ドリル】WordPress からのメールが迷惑メール扱いされてしまう | 207 |
| 6.2.2 | 管理画面にダイジェスト認証をかけよう | 209 |
| 6.3 | 画像を S3 に保存する | 213 |
| 6.3.1 | Amazon S3 とは? | 214 |
| 6.3.2 | S3 アップ用の IAM ユーザーを作ろう | 214 |
| 6.3.3 | WordPress に S3 を使うためのプラグインを入れよう | 215 |
| 6.3.4 | 投稿を試してみよう | 222 |
| 第 7 章 | サーバのバックアップを取っておこう | 227 |
| 7.1 | EBS スナップショットと AMI | 228 |
| 7.2 | インスタンスから AMI を作ろう | 230 |
| 第 8 章 | ELB と Auto Scaling で複数台のサーバを運用しよう | 235 |
| 8.1 | ELB はなんのためにある? | 236 |
| 8.2 | ロードバランサーを作ろう | 236 |
| 8.2.1 | ロードバランサーの種類の選択 | 238 |
| 8.2.2 | ステップ 1: ロードバランサーの設定 | 239 |
| 8.2.3 | ステップ 2: セキュリティ設定の構成 | 240 |
| 8.2.4 | ステップ 3: セキュリティグループの設定 | 240 |
| 8.2.5 | ステップ 4: ルーティングの設定 | 241 |
| 8.2.6 | ステップ 5: ターゲットの登録 | 242 |
| 8.2.7 | ステップ 6: 確認 | 243 |
| 8.2.8 | ヘルスチェックの確認をしよう | 244 |
| 8.3 | WordPress のサイトを表示する経路を変更しよう | 245 |
| 8.3.1 | 「www. 自分のドメイン」と紐づく IP アドレスを変更しよう | 245 |

| | | |
|-----------------|--|------------|
| 8.3.2 | Route53 で A レコードを Alias に変更しよう | 247 |
| 8.3.3 | HTTP を通すのは ELB 経由のアクセスだけにしよう | 252 |
| 8.4 | Auto Scaling | 255 |
| 8.4.1 | 起動設定を作成しよう | 256 |
| 8.4.2 | Auto Scaling グループを作成しよう | 261 |
| 8.4.3 | インスタンスを削除して自動復旧を試してみよう | 266 |
| 第 9 章 | もっと AWS について勉強したい！ | 271 |
| 9.1 | 公式のオンラインセミナーや資料集 | 272 |
| 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 | 272 |
| 9.3 | Linux やコマンドも学びたいなら | 273 |
| 第 10 章 | AWS をやめたくなったらすること | 275 |
| 10.1 | 無料の 1 年が終わる前にすべきこと | 276 |
| 10.1.1 | AWS アカウントを停止する | 276 |
| 付録 A | 本当の Git | 279 |
| A.1 | Git - ぎゅっと言えないトワインクル | 280 |
| あとがき | | 281 |
| Special Thanks: | | 281 |
| レビュアー | | 281 |
| 参考ウェブサイト | | 281 |
| 著者紹介 | | 283 |

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 Amazon がやっているクラウドです。

「AWS は Amazon 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 Amazon 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？

初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？

IT系、その中でも特に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。また日常生活でも「テレビで紹介されたらアクセスが殺到してサーバが落ちたんだって」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」というように普通に使われます。

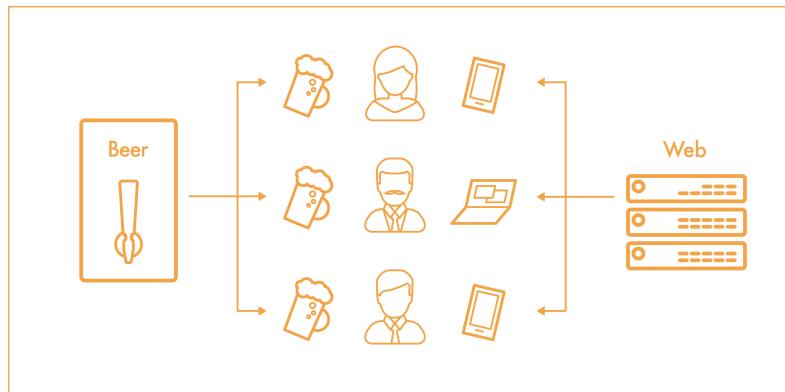
でももし後輩から「サーバってなんですか？」という直球の質問を投げつけられたら、しっかりとホームランで打ち返せますか？

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の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」、こちらも具体的になつて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。そして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。

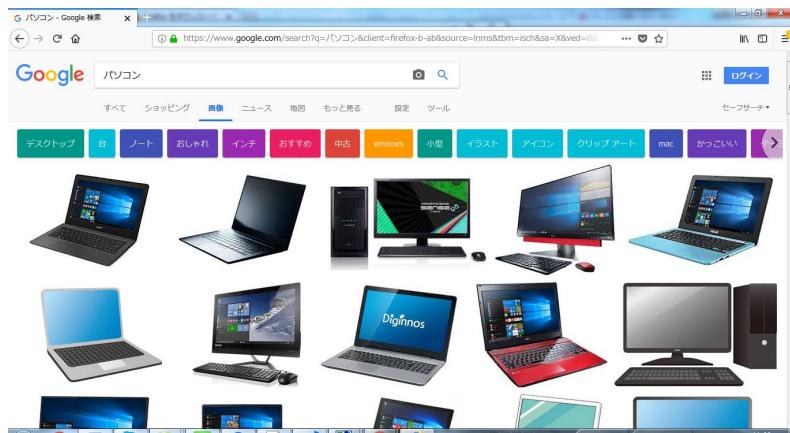
ところでちょっと頭の中でサーバの姿を思い浮かべてみてください。できればイラストじゃなくて実写でお願いします。思い描いたサーバの姿を「？」が浮かんでいるもやもやの中に描いてみましょう。



▲図 1.2 サーバの姿を実写で思い浮かべて描いてみよう

サーバの姿を思い浮かべて絵は描けましたか？ あんまりくっきりとは浮かばないですよね。サーバの姿なんて明確に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、パソコンのときとは打って変わってなかなか思い浮かびません。

では皆さんを悩ませる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) が出している 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} という専用の棚に収めことが多いです。

^{*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 Ownd 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

サーバラックや、その中に詰まったラックマウント型サーバを実際に見たことはありますか？

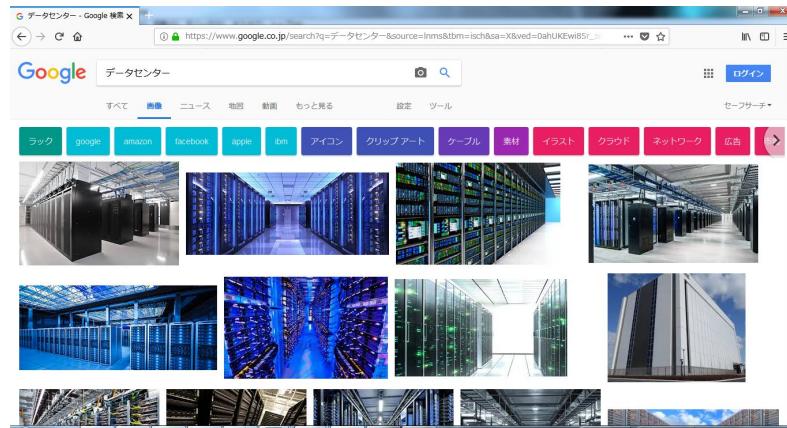
どんなウェブサイトも、世界中のどこかにあるサーバの中で稼動しているはずなのですが、インフラエンジニアでなければサーバを見る機会はなかなかないかも知れません。

^{*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に必要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 データセンターにはサーバのための設備が整っている

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

〈サーバに適した設備〉1. 防犯設備

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

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

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



▲図 1.8 攻撃や窃盗はインターネット越しでも直接でもできる

データセンターは後者の「物理的な攻撃や侵入」からサーバを守るための設備を整えています。堅牢さはデータセンターによって異なりますが、たとえば次のような防犯対策が取られています。

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

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

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

〈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台があれば大丈夫！」という冗長構成についていても、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。電気は使っ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。^{*12}

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。

*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」というやり取りをした後で、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！」という問い合わせが来た、という話も聞きます。サーバがなければサイトは見られない、というのは決して万人にとって当たり前のことではないのです・・・・ないのです・・・・。

*¹² 2018年9月に北海道の胆振地方で震度7の地震が発生して道内のほぼ全域が停電してしまったとき、さくらインターネットの石狩データセンターはUPSと自家発電で稼動を続けました。筆者が使っている「さくらのVPS」というサーバも石狩DCにありましたが止まることなく動き続けていて、普段から「こういうとき」のために備えていたんだなと涙が出そうな気持ちになりました。一部電源切替がうまくいかなかったUPSが原因で数時間停止したサービスもあったそうですが、それでも事業と地震の規模を考えると本当に頭が下がります。ありがとうございます、さくらインターネット！

〈サーバに適した設備〉 3. 空調設備

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

暑い部屋ではサーバが故障したり落ちてしまったりする^{*13}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、サーバはいったいどこにいるの？ というお話をでした。

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は「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よね。データセンターを借りるときもまったく同じで「ラックの 1/2 なら 12 万円、1 ラックまるごとなら 20 万円」のように、ラックサイズによって月額の費用が変わってきます。

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

2000 年代前半、インターネットが盛り上がってきてサーバが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？ と各社が試行錯誤した結果、前述の省スペースなブレードサーバや、奥行きが 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 にサーバを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。^{*14} データセンターによっては、電源容量を増やせるオプションを提供しているところもありますが、それはそれで「10A 増やしたら 3 万円」のように月額費用に跳ね返ってきます。

^{*13} 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なことでの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。

^{*14} 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、を延々と繰り返していた・・・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。

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。



▲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^{*15}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^{*16}。

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サーバであり、ゲスト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

^{*15} 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。

^{*16} ホストOS、ゲストOSという呼び方をすることもあります。

理サーバはスペックが高いのでさらに高額・・・となると中小企業やスタートアップ企業が自社で物理サーバや仮想サーバを所有・管理するのはなかなか大変なことです。

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

長々と物理サーバと仮想サーバについて説明してきましたが、なんとなくゴールがお分かりでしょうか？ 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。しかしサーバを買うと言っても、お店に行ってぱっと買って持ち帰れるという訳ではありません。

「どのメーカーのサーバにしよう？ HPE かな？ それとも IBM かな？ DELL がいいかな？」と各社の見積もりを販社経由で取り、値引き交渉をして、それでも数十万から数百万するので社内の裏議を通してやっと購入。購入してもすぐ届くわけではなく、数週間待ってやっと届きます。そして届いたらサーバを段ボールから出して、データセンターもしくは自社のサーバルームにあるサーバラックのところまで持つて行って、がっちゃんこと設置。^{*17}

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、何をかも自分で用意しないといけないため、

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

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

^{*17} 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。

1.4.2 クラウドとは？

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

クラウドなら「自社のウェブサイト作りたいなあ・・・だからサーバが必要だ！」と思ったら、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もう少し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！」^{*18}と思いついたとします。

ピザ屋をオープンすべく、土地を買ってそこに店舗となる建物を建て、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をはじめてみよう」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つてしまつ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 なら使い始めるとき

^{*18} そうしたら「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。

の初期費用もなく 1 秒単位の従量課金なので「ピザ屋もうやめたいけど、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。^{*19}

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

さらに、もしピザ屋が大繁盛したら、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、すぐにはじ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れる、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^{*20}と比べると高額です。

*19 とはいえる現実のフードコートで出店する場合は契約期間の縛りがあるでしょうし、止めようと思っても即日でやめられるものではないですよね。実は EC2 の料金体系も従量課金の「オンデマンド」の他に「リザーブドインスタンス」といって長期利用が確約できれば料金が最大 75% も安くなるプランがあります。つまり「僕と契約して低コストになってよ！」ということなので途中解約ができないリスクがあることは認識した上で利用しましょう。

*20 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。

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しまっても、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。

この可用性を自力で確保しようとしたら大変です。ピザ屋を常に営業し続けておくために、いつ来るか分からぬ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かつたら相当なコストがかかります。これはオンプレミスに当てはめると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^{*21}です。構成次第でこれが簡単に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きは、問答無用でピザ屋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^{*22}しました。こうした障害の際にもAWSが発表してくれる内容がすべてですので、原因が分かるまで自分で徹底的に調べる、あるいは自力で何とかする、ということはできません。

またフードコートの通路やトイレ、駐車場といった共有スペースは他店舗（ドーナツ屋やラーメン屋など）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^{*23}を使っていても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

*21 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。

*22 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>

*23 他店舗というのは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のことです。

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 の略で、欲しいものをばっつとな！ すると翌日には届くあのAmazon がやっているクラウドなのです。

AWS がなんなのか、お分かりいただけましたでしょうか？

1.4.5 パブリッククラウドとプライベートクラウド

ところでパブリッククラウドやプライベートクラウド、という言葉は聞いたことがありますか？ 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があります。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、みんなでサーバを共有するのってちょっと抵抗あるな・・・」と思った人たちを安心させるため、「プライベートクラウド」という言葉が生まれました。このプライベートクラウドとはいったい何なのでしょう？

たとえばオンプレミスの環境で高スペックな物理サーバを買って、ホストサーバにしてその上でゲストサーバ（仮想サーバ）を立てられるようになったとします。これを見て「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リソース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、話をよくよく聞いてみないと分からない、ということです。^{*24}

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^{*25}、Microsoft の Azure (アジュール)^{*26}、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^{*27}、お名前.com と同じ GMO グループの GMO クラウド^{*28}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^{*29}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く、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^{*30}、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 の良いところを見つかりたいですね。

^{*24} 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。

^{*25} <https://cloud.google.com/>

^{*26} <https://azure.microsoft.com/ja-jp/>

^{*27} <https://cloud.sakura.ad.jp/>

^{*28} <https://www.gmocloud.com/>

^{*29} 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

^{*30} 画面や機能もどんどん変わっていくので、この後出てくる設定画面も、皆さんのが手を動かしてやってみると頃には本著のキャプチャとは違うものになっているかも知れません。AWS のいいところでもあり、本やマニュアルを作つて説明する側にとってはつらいところでもあります。

第 2 章

AWS を使い始めたら最初にやること

この章では AWS の管理画面にサインインして、AWS を使い始めたら最初にやるべき設定をします。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つ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ので、そういうならないう後ほど「利用金額が 2 ドルを超えたらメールで知らせる」という請求アラートの設定をしつ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。

2.2 AWS のアカウント作成

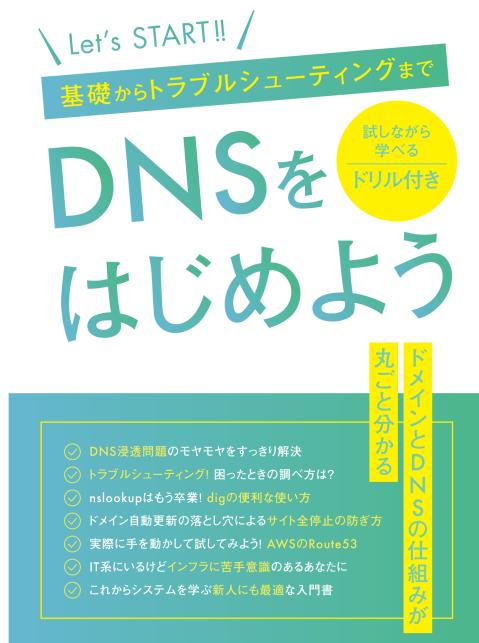
AWS を使うには、先ず AWS アカウントを作成する必要があります。

AWS アカウントの作成は「DNS をはじめよう」(図 2.2)*2 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

*2 <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 営業日以内に本が BOOTH 倉庫からネコポスで送られてきます。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^d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/>

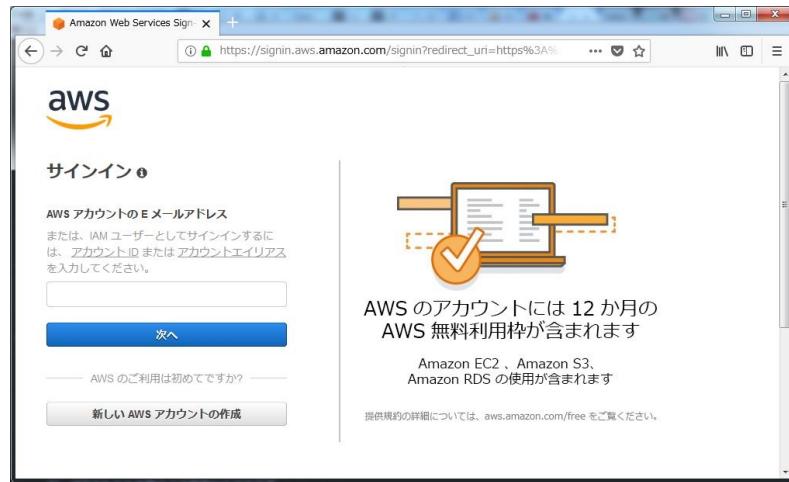
^b イラストを投稿できる SNS、pixiv でお馴染み。<https://www.pixiv.net/>

^c <https://booth.pm/ja/search/技術書典>

^d この顛末は 2018 年 10 月 8 日の技術書典 5 で頒布される「技術季報 vol.4」に掲載予定ですの
で、よかつたらそちらもお読みになってください。

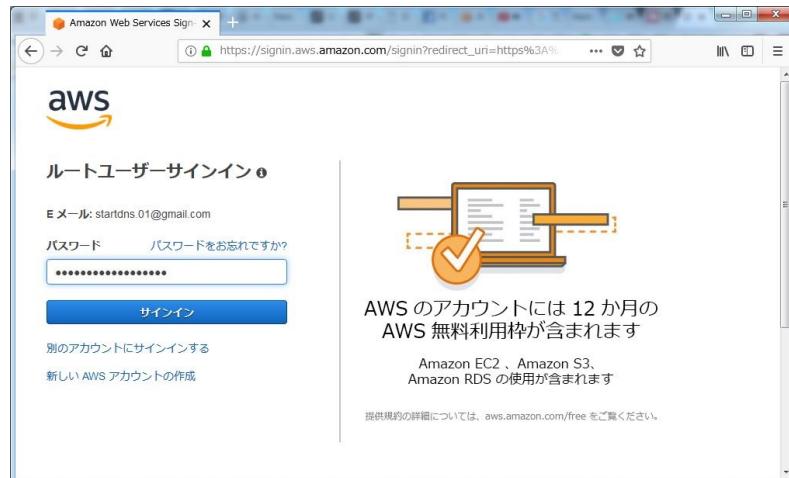
2.3 マネジメントコンソールにサインイン

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。



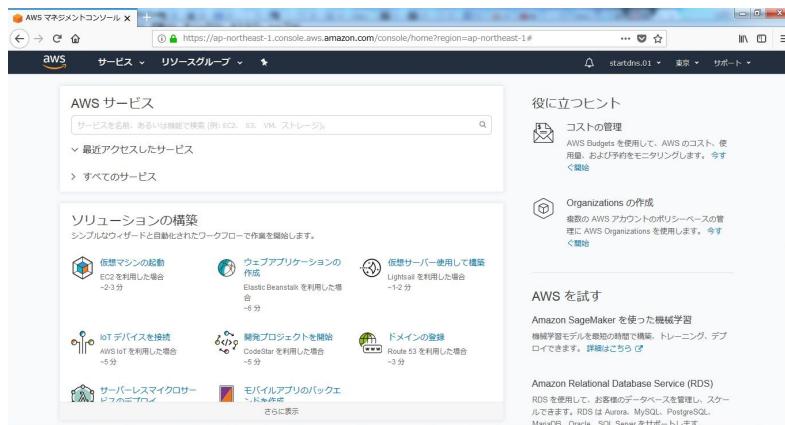
▲図 2.3 マネジメントコンソールのサインイン画面

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」をクリックします。



▲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。これからウェブサーバを立てたりデータベース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 _____

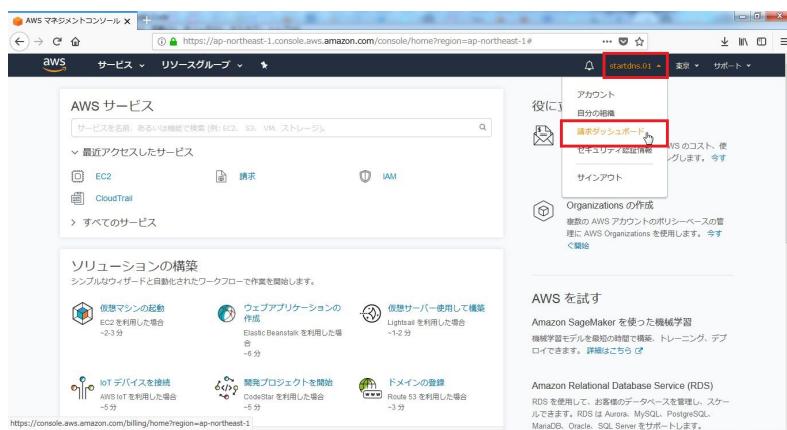
解答

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 は基本的に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えた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。



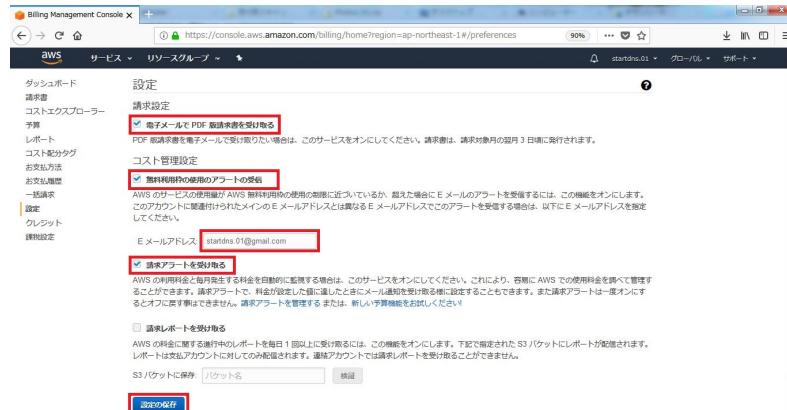
▲ 図 2.6 ルートユーザー名>請求ダッシュボード

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

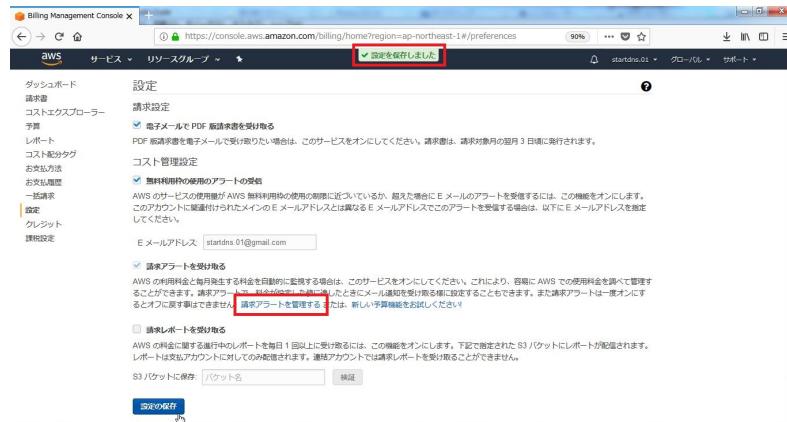
先ず一番上の「電子メールで PDF 版請求書を受け取る」にチェック（図 2.8）を入れます。続いて「無料利用枠の使用のアラートの受信」には元々チェックが入っていると思いますので、その下の「E メールアドレス:」に自分のメールアドレスを記入してください。その下にある「請求アラートを受け取る」にもチェックを入れたら「設定の保存」をクリックします。



▲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」の欄に 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は最大でも月 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

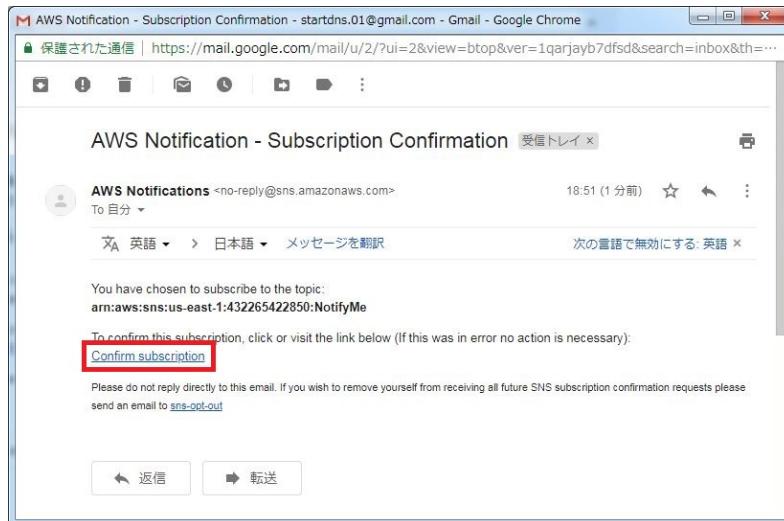
「超過」の金額と「通知の送信先」のメールアドレスを記入したら「アラームの作成」をクリックしてください。「新しいメールアドレスの確認」(図 2.12) と表示されます。



▲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

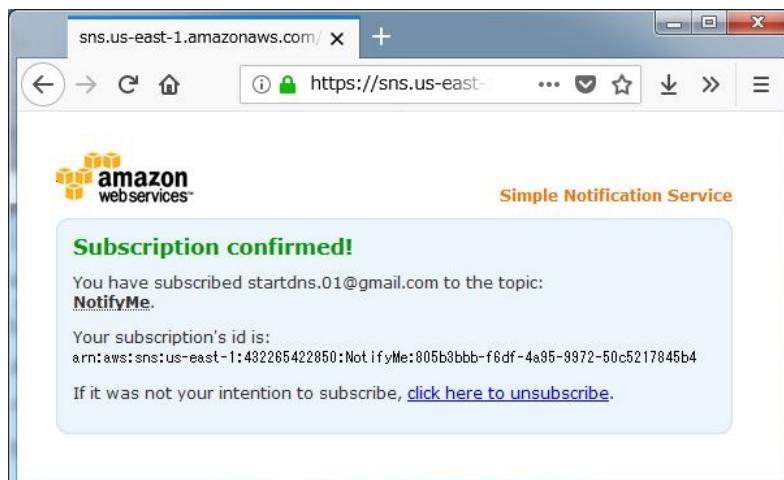
通知の送信先として新しいメールアドレスを登録した場合、「AWS Notification - Subscription Confirmation」という件名でメールアドレスの確認メール(図 2.13)が届きます。メール本文にある「Confirm subscription」というリンクを踏んでください。

2.4 CloudWatch で請求アラートを設定しよう



▲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

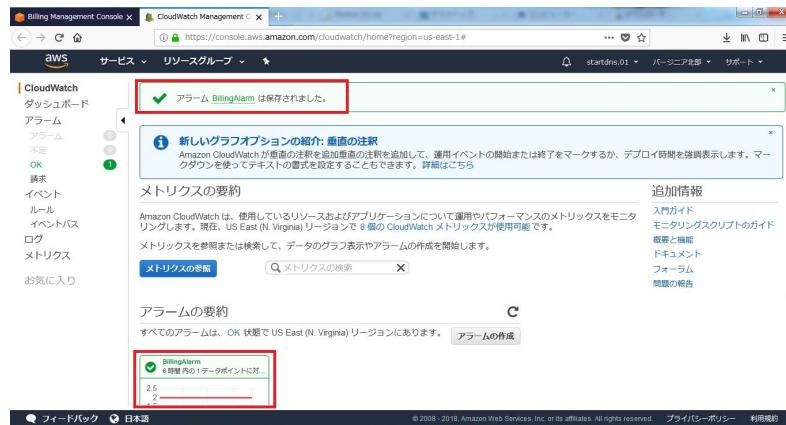
メールアドレスの確認が終わると元の画面でメールアドレスの横に緑のチェックマーク

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。



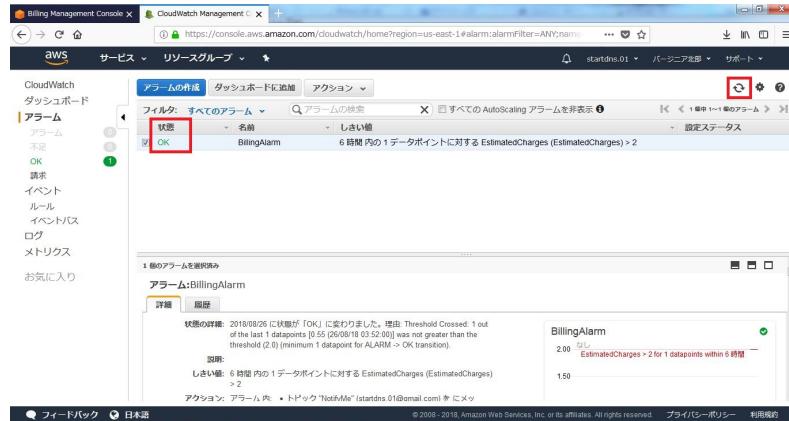
▲図2.15 緑のチェックマークがついていたら「アラームの表示」をクリック

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。もし状態に「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 ドルを超えたたら「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。

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

AWS のユーザにはクレジットカードを紐付けているのですから、お財布もルートユーザーも等しく扱いには気をつけなければいけません。たとえば悪い人があなたのルートユーザーの E メールアドレスとパスワードを盗んで、こっそりマネジメントコンソールにサインインしたとします。ルートユーザーならなんでもできるので、景気良いくらい高

いサーバ^{*3}を100台立てた^{*4}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^{*5}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の体験記がたくさん

このようにルートユーザーだと本当になんでもできてしまうため、権限が必要ないときに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無邪気に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 ユーザーを作ろう

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

^{*3} EC2 の d2.8xlarge というサーバは、東京リージョンだと1時間あたり 6.752 ドル。日本円にすると1日でおおよそ 18,000 円です。もちろん大量に立てられないように台数制限はありますが、ルートユーザーならその制限を緩和するリクエストを出すことだって可能です。

^{*4} 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。

^{*5} 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 の利用者を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 カードを持っている社内の人しかオフィスフロアへ入れないようになっていたり、さらに特定のプロジェクトチームにはそこのプロジェクトメンバーしか入れないようになっていたりします。

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

IAM ユーザーは 1 人に 1 つ

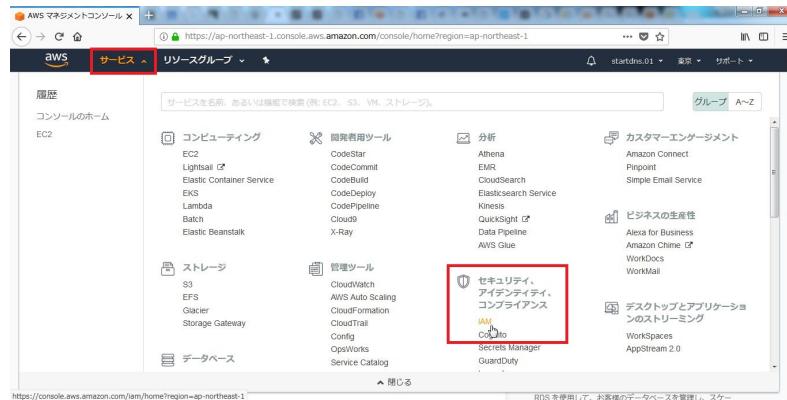
本著では 1 人だけで作業をする想定なので、IAM ユーザーも 1 人しか作りません。ですがもし業務などで複数名でマネジメントコンソールにサインインする場合は、**IAM ユーザーは必ず 1 人につき 1 つずつ作成してください**。まったく同じ作業をするから開発チーム内の A さんと B さんは同じ IAM ユーザーを共有すればいいのでは？ と思われる場合でも、必ず A さん B さんそれぞれに別々の IAM ユーザーを用意することをお勧めします。

なぜならば AWS では「いつ・どの IAM ユーザーが・なにをしたのか」をすべて記録していて、後から調べることができるのですが、仮に A さんと B さんが 1 つの IAM ユーザーを共用していた場合、何か重大なトラブルが起きたとき^{*6}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

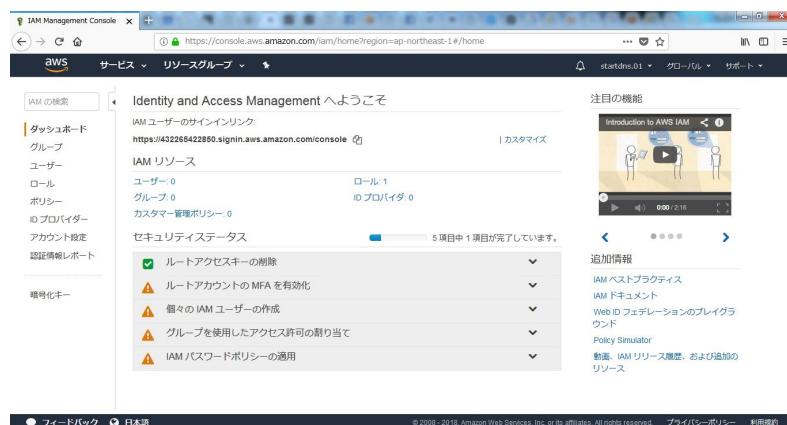
それではマネジメントコンソールの左上にある「サービス」から、「セキュリティ、アイデンティティ、コンプライアンス」の下にある「IAM」（図 2.19）をクリックしてください。

^{*6} 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、その IAM ユーザーを使っていた人全員に疑いがかかってしまいます。



▲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

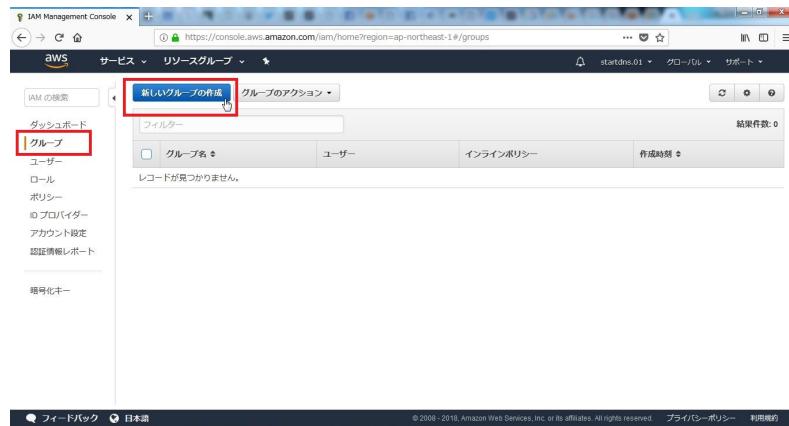
IAM のグループを作成

では先ず IAM ダッシュボードの左メニューから「グループ」をクリックしてください。

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、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 ユーザを 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。



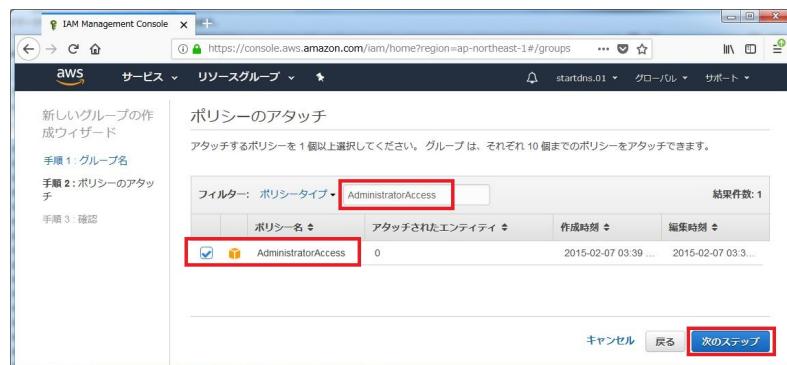
▲図 2.22 グループ名に「start-aws-group」と入力して「次のステップ」をクリック

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、この中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。前述のルートユーザーほどではありません^{*7}が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。



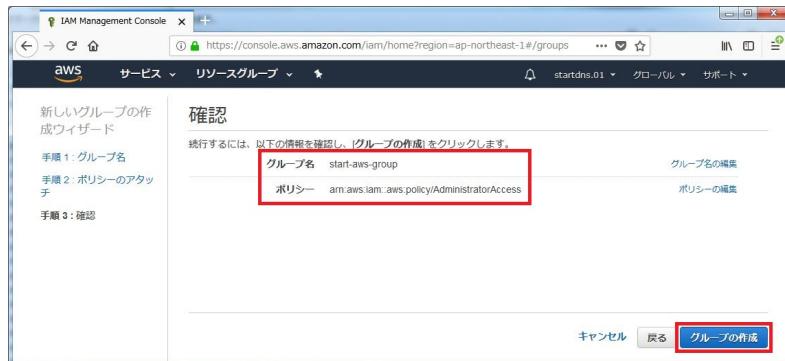
▲図2.23 「AdministratorAccess」にチェックを入れて「次のステップ」をクリック

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

^{*7} もっとも権限の強い「AdministratorAccess」であってもルートユーザーのように全権を持っているわけではありません。たとえば「AdministratorAccess」はデフォルト設定では請求情報にはアクセスできません。またAWSアカウントの解約などもルートユーザーでなければ行えません。

^{*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」をクリックすると、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、忘れないようしっかりと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

次にステップ 2 (図 2.28) です。先に作っておいた「start-aws-group」というグループに、ユーザーを追加します。「start-aws-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

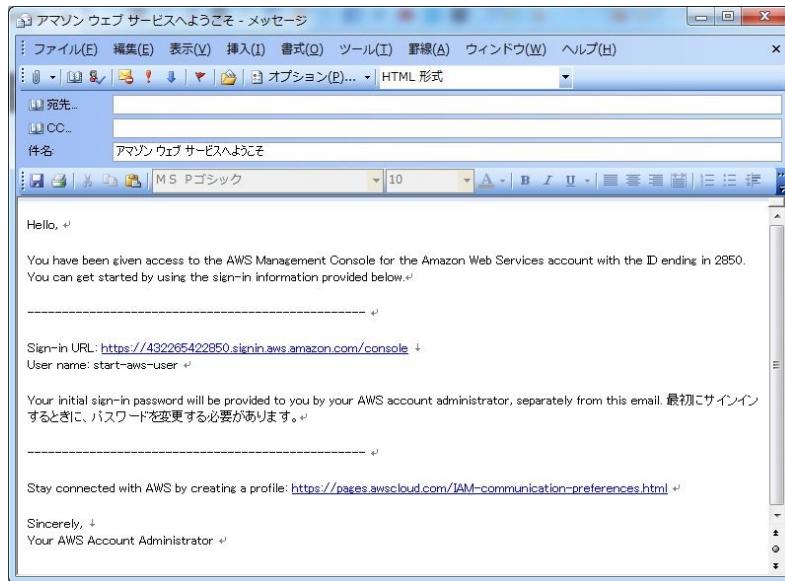


▲図 2.32 パスワードの「表示」をクリックして、パスワードもメモしておこう

▼表 2.1 IAM ユーザの情報

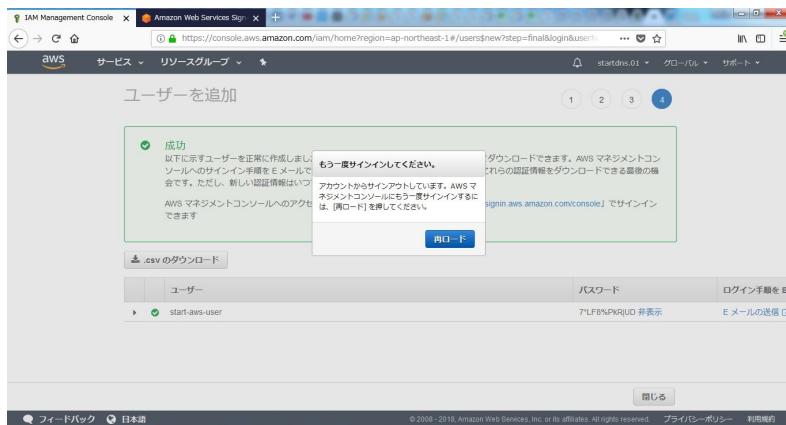
| IAM ユーザー情報 | 例 | 自分の IAM ユーザー情報 |
|---------------------|----------------|----------------|
| AWS アカウント (URL の数字) | 432265422850 | |
| ユーザー名 | start-aws-user | |
| パスワード | 7*LF8%Pkr UD | |

メモできたら、続けてパスワードの右隣にある「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。



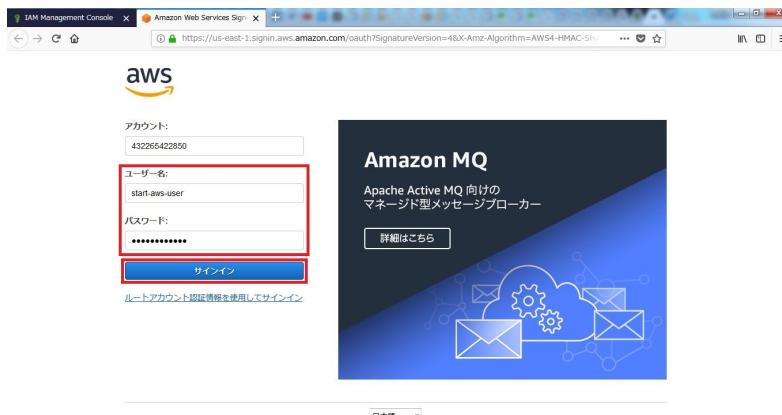
▲図 2.33 サインイン URL や IAM ユーザー名をメールで自分自身宛てに送っておこう

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://432265422850.signin.aws.amazon.com/console>」（数字の部分は人によって異なります）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。



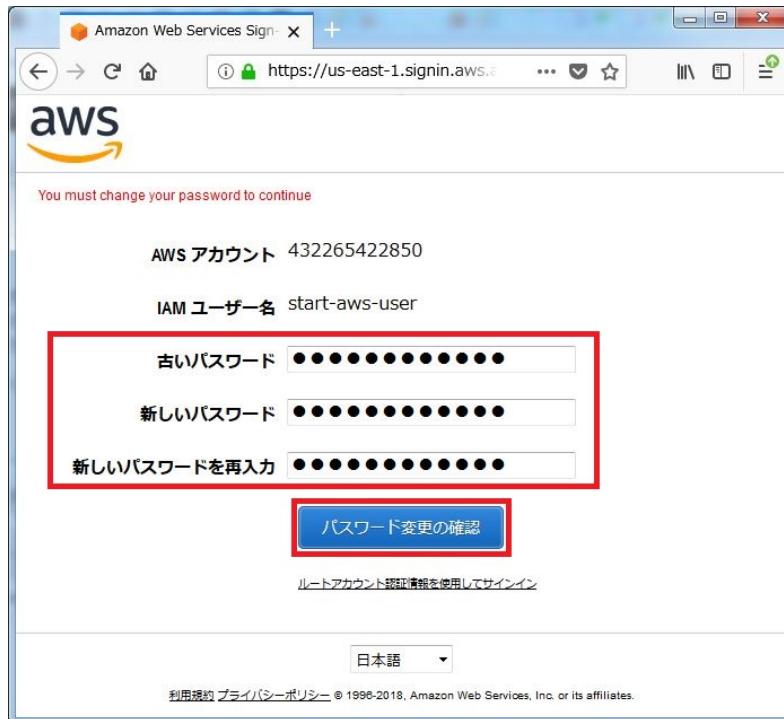
▲図 2.34 ルートユーザーでサインインしていた画面は自動的にサインアウトされる

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。



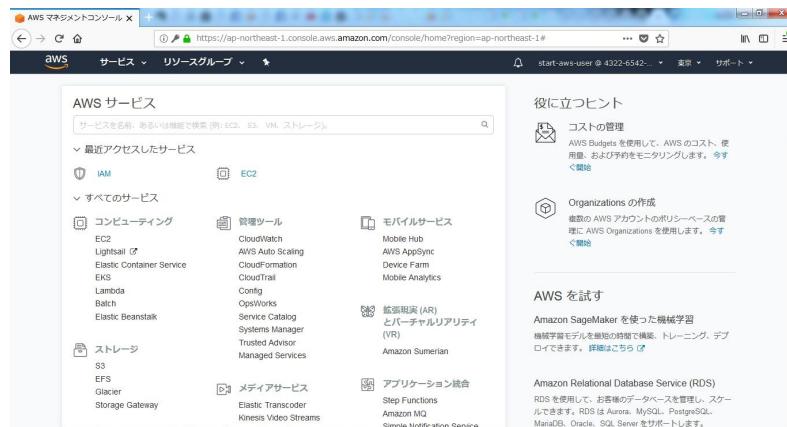
▲図 2.36 新しいパスワードを入力して「パスワード変更の確認」をクリック

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 | 432265422850 | |
| IAM 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。ここを見ると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。

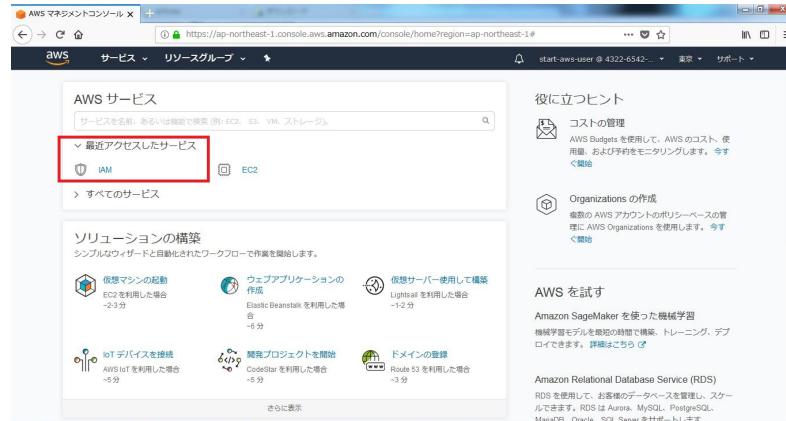


▲図 2.38 IAM ユーザーでサインインしていることを確認

これで 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。

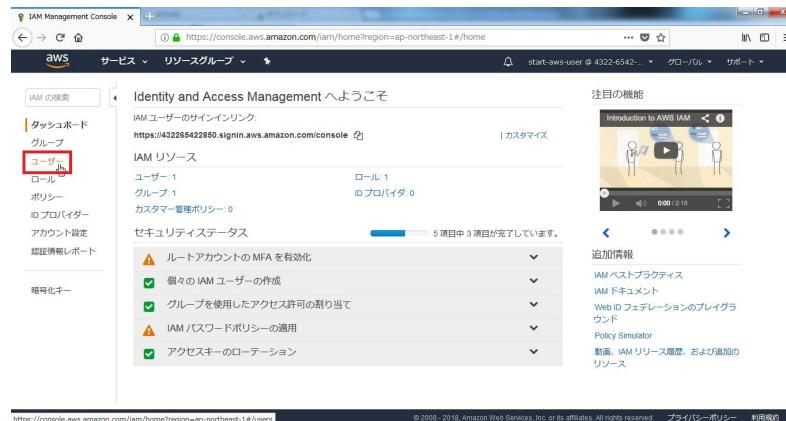
2.5.3 MFA (多要素認証) で不正利用から IAM ユーザーを守る

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（図 2.39）しても構いません。



▲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。

| ユーザー名 | グループ | アクセスキーの有さ | パスワードの有さ | 最後のアクティビティ | MFA |
|-----------------|-----------------|-----------|----------|------------|-------|
| start-aws-user* | start-aws-group | なし | 今日 | 今日 | 有効でない |

▲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」の横にあるエンピ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

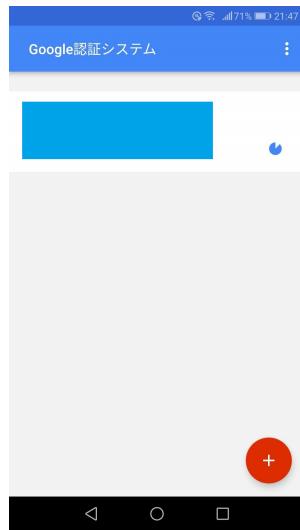
お手元のスマホに「Google 認証システム（Google Authenticator）」をインストール^{*11}したら「次のステップ」をクリック（図 2.45）します。



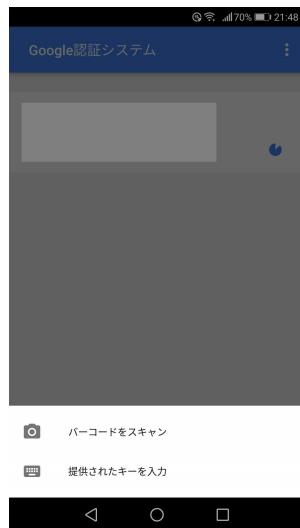
▲図 2.45 スマホに認証アプリをインストールしたら「次のステップ」をクリック

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。



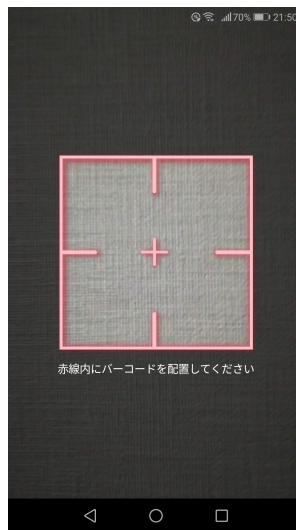
▲図 2.46 「Google 認証システム」のアプリを起動したら右下の赤い+ボタンをタッチ



▲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。

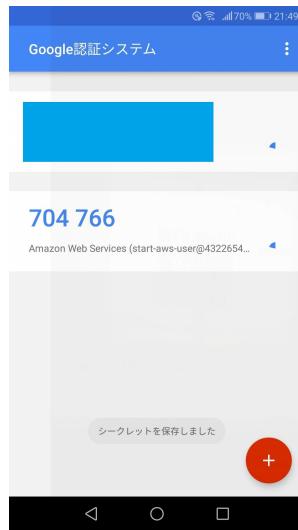


▲図 2.48 「赤線内にバーコードを配置してください」と表示されたら



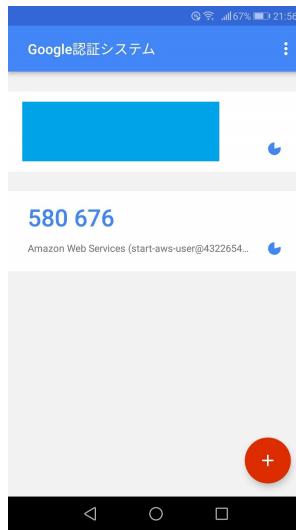
▲図 2.49 マネジメントコンソールに表示されたバーコードをスマホで撮る

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。

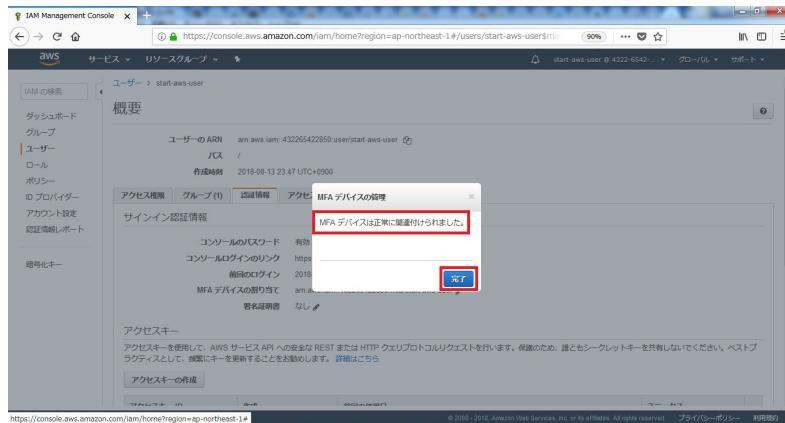


▲図 2.51 認証コードは 30 秒ごとに次々と切り替わる



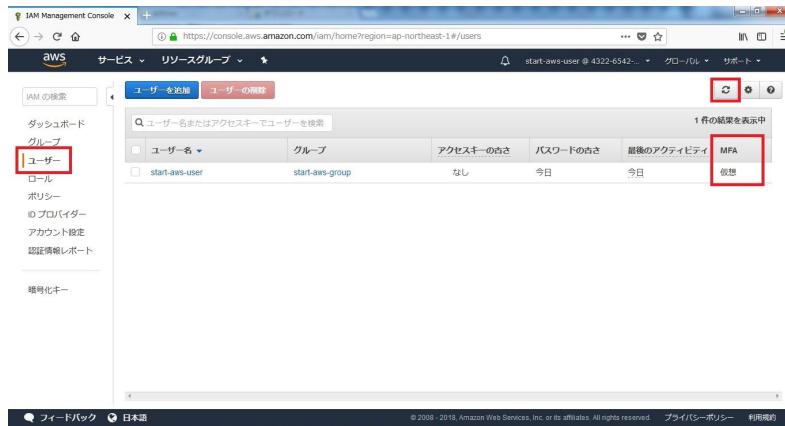
▲図 2.52 「認証コード 1」と「認証コード 2」を入力して「仮想 MFA の有効化」をクリック

「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。



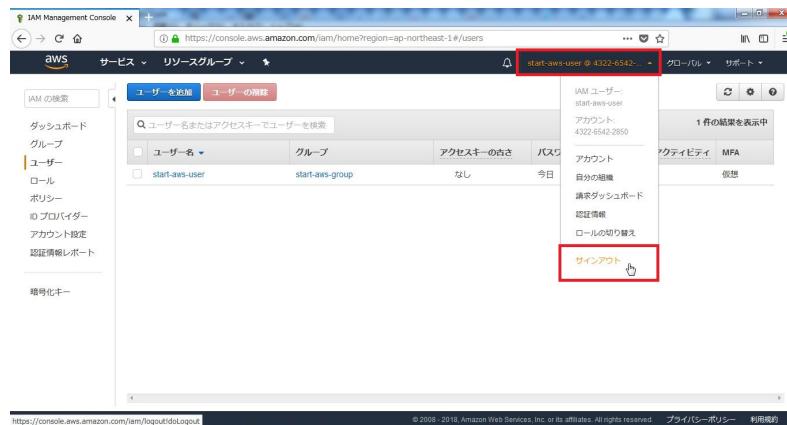
▲図 2.53 「MFA デバイスは正常に関連付けられました。」と表示されたら「完了」をクリック

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

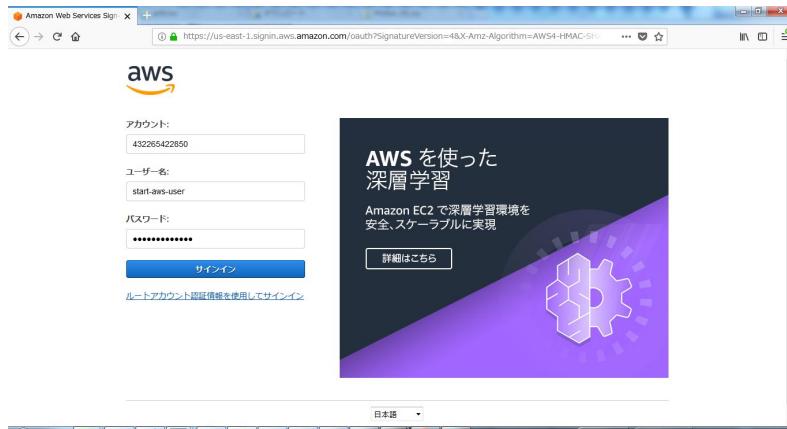
右上の「コンソールへログイン」をクリック（図 2.56）します。



▲図 2.56 「コンソールへログイン」をクリック

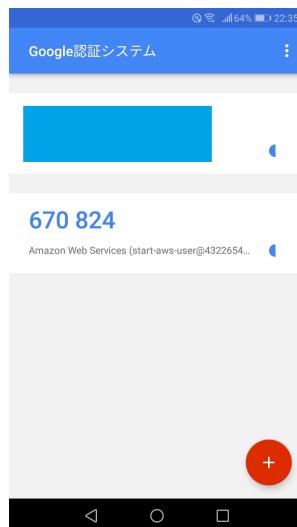
1. アカウント（12桁の数字）
2. ユーザー名
3. パスワード

の3つを入力したら「サインイン」をクリックします。

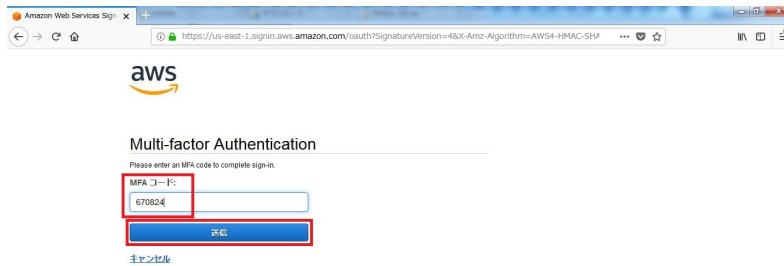


▲図 2.57 「サインイン」をクリック

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。



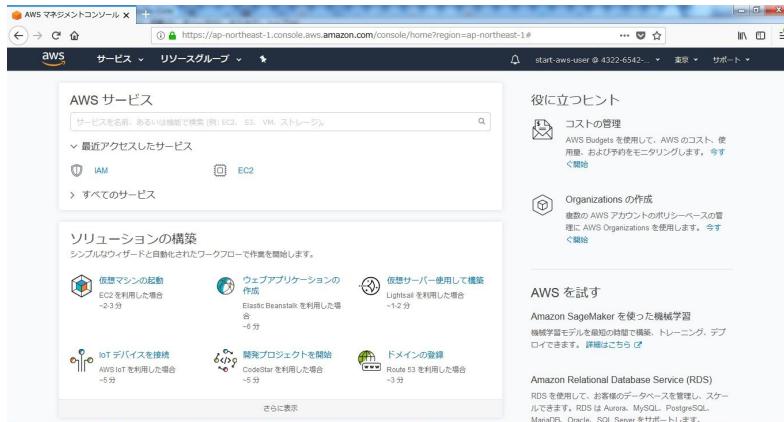
▲図 2.58 スマホで「Google 認証システム」を起動



▲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。

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。

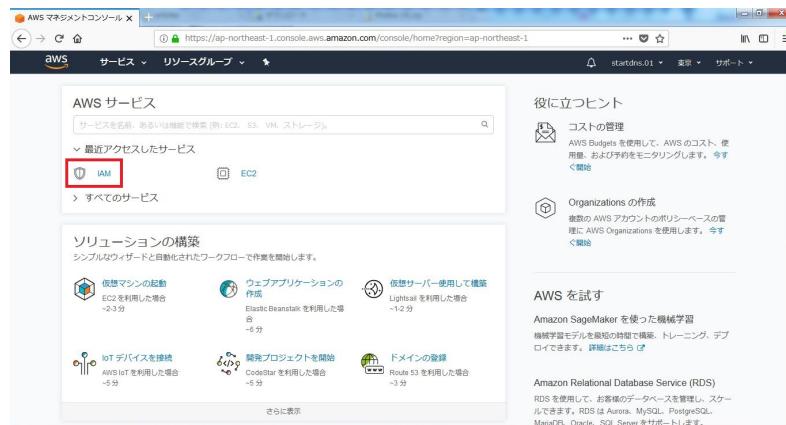


▲図 2.60 MFA を用いたサインインができるようになった！

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。

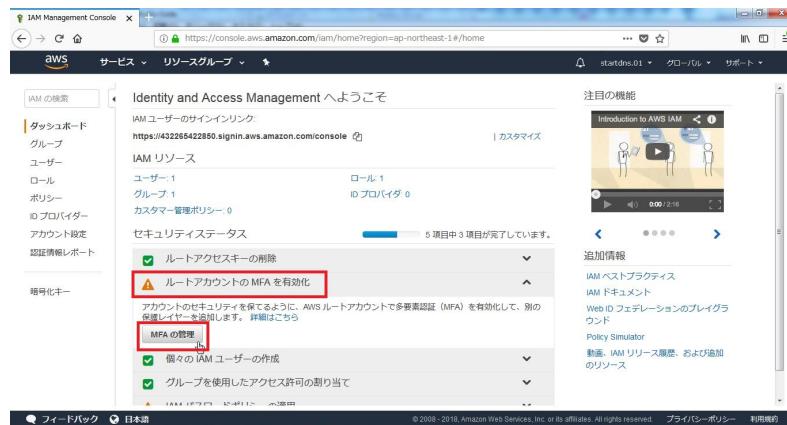
いったん「start-aws-user」でサインアウトしたら、今度はルートユーザーでサインインしなおしてください。そしてマネジメントコンソールの「最近アクセスしたサービス」から「IAM」をクリック（図 2.61）して IAM のダッシュボードを開きます。



▲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

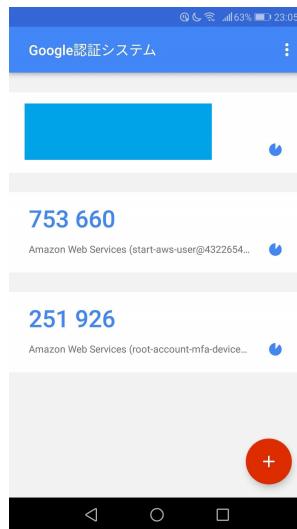
IAM のダッシュボードで「ルートアカウントの MFA を有効化」（図 2.62）^{*12}を開いて「MFA の管理」をクリックします。

^{*12} 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という言葉が混在していたり、日本語のマネジメントコンソールではこうした表記揺れはよくあることです。日本語の翻訳がいまいちちなせいで分かりにくくなっているところもあるので、マネジメントコンソールの言語を英語にした方がいい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 2.62 「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。

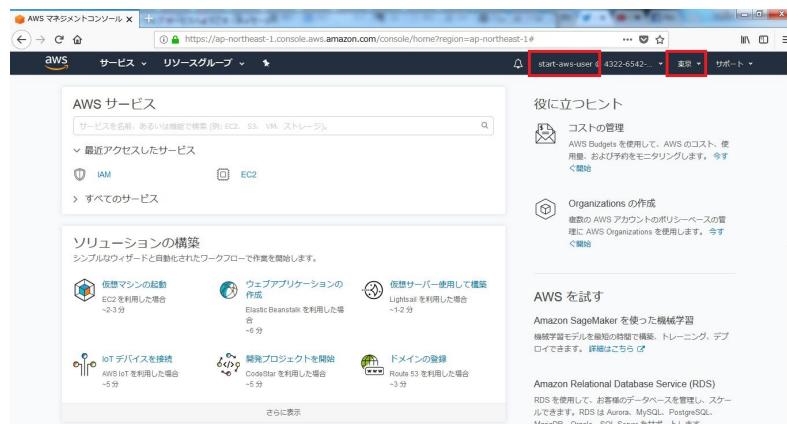


▲図 2.63 IAM ユーザーだけでなくルートアカウント用の認証コード（6 行の数字）も表示されている

ルートアカウントでも MFA が有効になつたら、再び「start-aws-user」でサインインしなおして先へ進みましょう。もしどこの URL からサインインするのか分からなくなつてしまつたら、先ほど送つておいたメールに「Sign-in URL: <https://432265422850.signin.aws.amazon.com/console>」(数字の部分は人によって異なります) 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 衔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つてこられましたか？



▲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」以外になつていたら、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」に変わつ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ため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は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。^{*13}

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^{*14}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」

^{*13} 「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この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ずつ異なります。

^{*14} アベイラビリティゾーンはよく AZ と略されます。

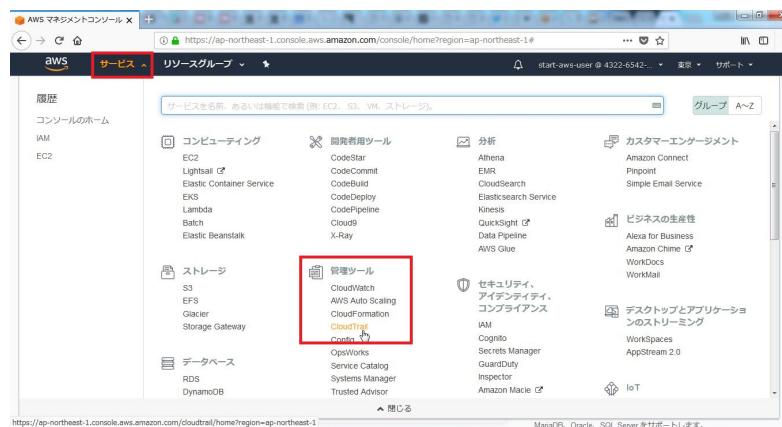
がある、というようなイメージです。

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）の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 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。

2.7 CloudTrail でいつ誰が何をしたのか記録

The screenshot shows the AWS CloudTrail Management Console dashboard. On the left, there's a navigation menu with 'CloudTrail' selected, followed by 'ダッシュボード', 'イベント履歴' (which is highlighted with a red box), and '証跡情報'. The main area is titled 'CloudTrail へようこそ' and contains a section titled '最近のイベント' (Recent Events). It lists five events from August 19, 2018, at 12:03 AM, all related to the user 'start-aws-user'. The events are: 'DescribeConfigurationRecorder...', 'DescribeConfigurationRecorders', 'LookupEvents', 'DescribeConfigurationRecorder...', and 'DescribeConfigurationRecorders'. Below this table is a link 'すべてのイベントを表示' (View all events). To the right, there's a sidebar with '最新情報' (Latest Information) and a '詳細はごちら' (More details) button. At the bottom, there are links for 'フィードバック', '日本語', and copyright information.

▲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。^{*15}

This screenshot shows the same AWS CloudTrail Management Console interface as the previous one, but with a specific filter applied. The 'イベント履歴' section has a dropdown menu where 'ユーザー名' is selected and 'start-aws-user' is entered. The list of events now only shows entries for this user. There are four events listed: 'DescribeTrails', 'ConsoleLogin', 'CheckMFa', and 'ConsoleLogin'. The third event, 'CheckMFa', is highlighted with a red box. The rest of the interface remains the same, including the sidebar and footer.

▲図 2.68 start-aws-user がいつ何をしたのかすべて表示される

*15 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。

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^{*16}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！

^{*16} 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！」と思って証跡を作成しても見られません。

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。
インフラエンジニアのお仕事体験みたいできっと楽しいですよ！

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は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}

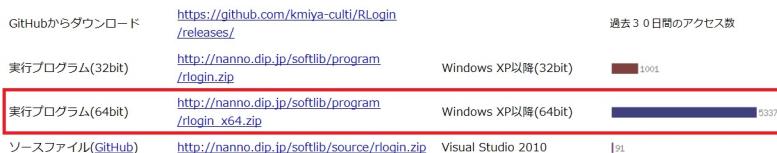


▲図 3.1 RLogin

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。

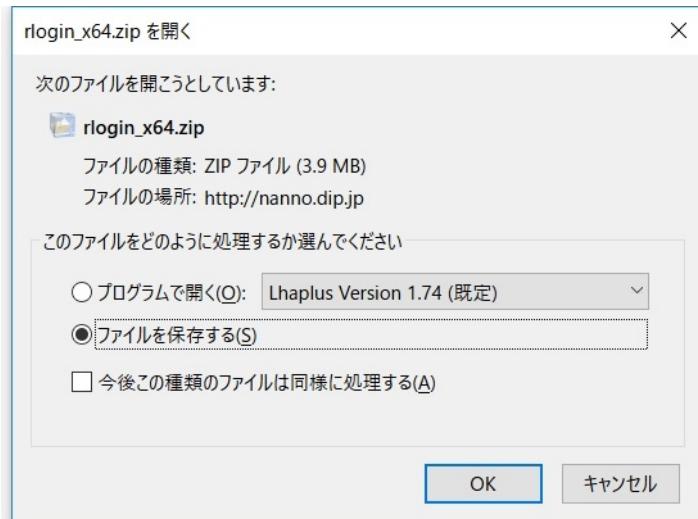
^{*1} PuTTYjp を使う場合、.pem の秘密鍵を PuTTYgen で.ppk に変換する必要が出てくるため、他のターミナルソフトに比べると一手間余計にかかります。

^{*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。



▲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

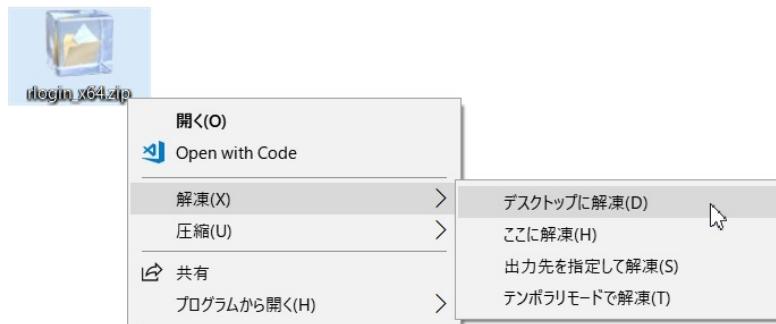
ダウンロードした ZIP ファイルを保存（図 3.3）します。保存場所はどこでも構いませんが、後でどこに置いたか分からなくなりそうな人はデスクトップに保存しておきましょう。



▲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} ZIP ファイルを右クリックしても「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/>



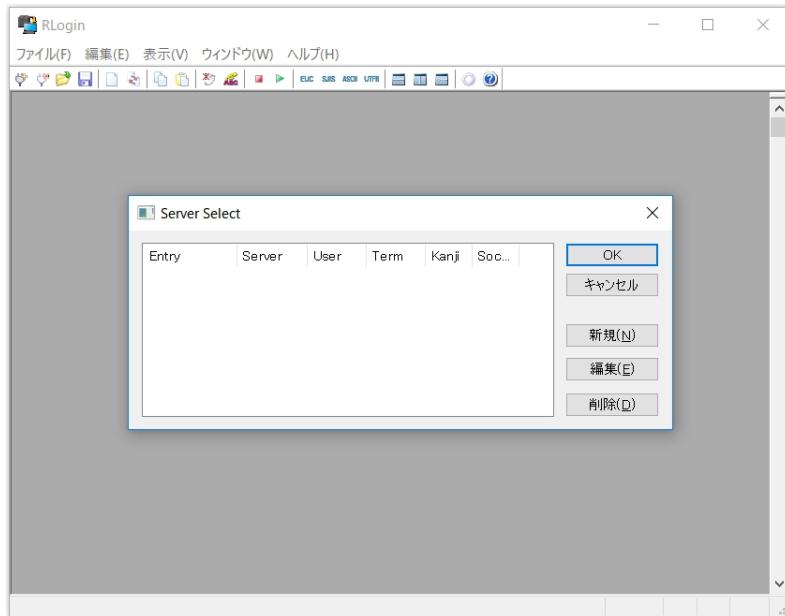
▲図3.4 ZIPファイルを右クリックして解凍>デスクトップに解凍

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何度かでできますので、表示されていない人は「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。

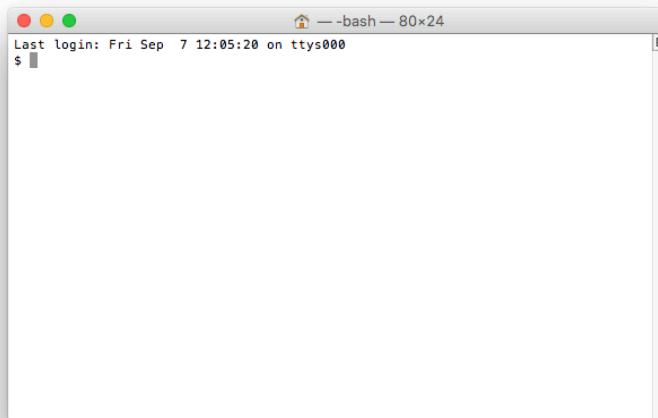


▲図 3.6 RLogin が起動した

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「キャンセル」をクリックして閉じてしまって構いません。また後で使うので、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」(図 3.7) 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。



▲図 3.7 最初からインストールされている「ターミナル」を使おう

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（図 3.8）すれば起動できます。



▲図 3.8 どこにあるのか分からなかつたら Spotlight で「ターミナル」と検索

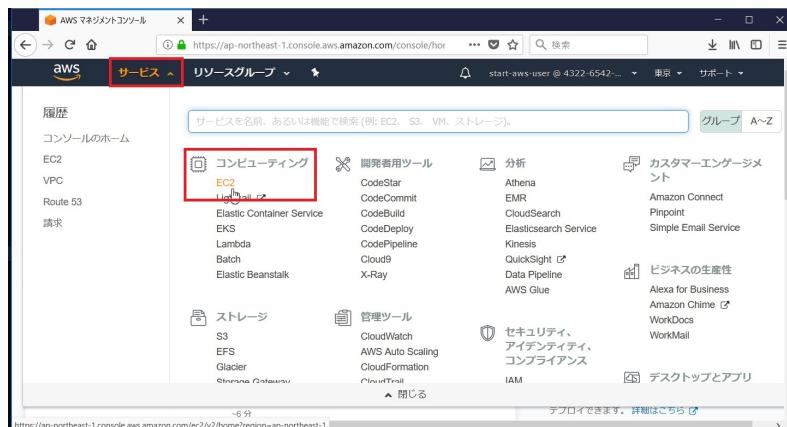
以上で事前準備は完了です。お待たせしました。いよいよサーバを立てましょう。

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。

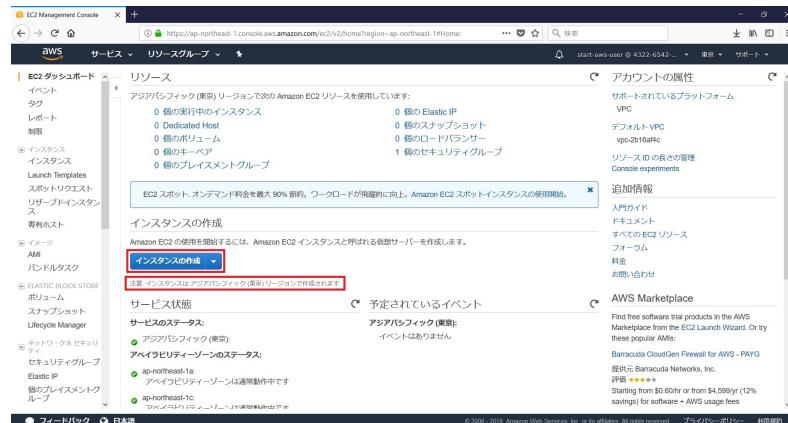
ちなみに AWS ではサーバのことを、**インスタンス**と呼びます。ここから先でインスタンスと書いてあつたらサーバのことだと思ってください。

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 3.9) をクリックしてください。



▲図 3.9 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 3.10）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。



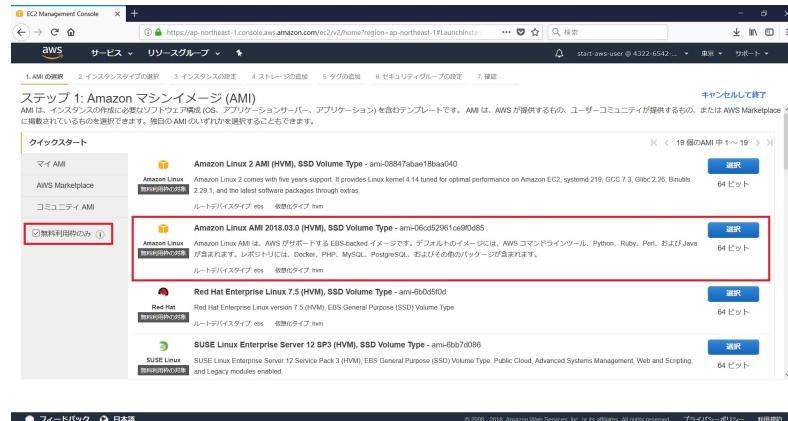
▲図 3.10 EC2 ダッシュボードで「インスタンスの作成」をクリック

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 3.11）と、無料利用枠以外の AMI は選択できなくなります。うっかり Windows Server のような有料 AMI を選択しないようにチェックを入れておきましょう。



▲図 3.11 「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ます。Amazon Linux は Red Hat 系のディストリビューションですので、RHEL 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 月時点です

- Amazon Linux
- Amazon Linux 2

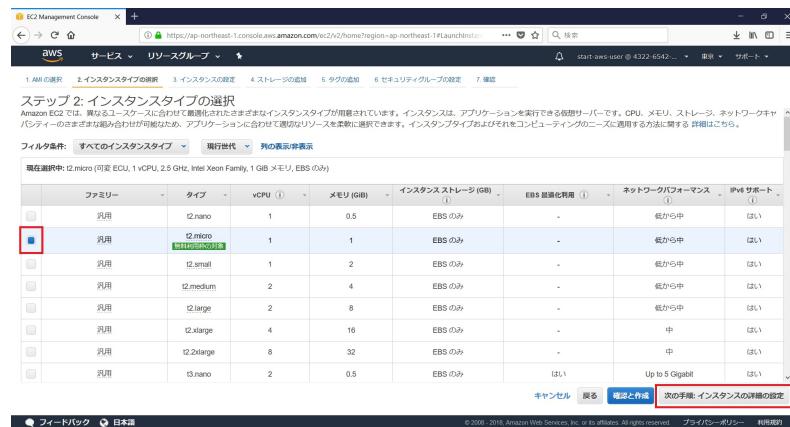
の 2 種類があります。Amazon Linux は RHEL6 系なので CentOS 6、Amazon Linux 2 は RHEL7 系なので CentOS 7 と使い勝手はほぼ同じです。本著では Amazon Linux を使用します。^{*6}

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 3.12）します。小～中規模のウェブサービスなら 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。

^{*5} Amazon による OS のサポートというのは「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。サポートの期限は Amazon Linux が 2020 年 6 月 30 日まで、Amazon Linux 2 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 だとデフォルトが 2.2 系、Amazon Linux 2 だと 2.4 系になります。



▲図 3.12 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「M5」はインスタンスマリと呼ばれるグループを表しており、Tが汎用、Mはもう少し性能の高い汎用、CはCPU重視、Rはメモリ重視、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なら汎用向けのグループで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。

インスタンスマリの後ろにある 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のサイトを作つてみよう！」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} small、medium、largeと段々中身が増えていくなんてマクドナルドのボテトと同じですね。

【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は t2.micro だとなんとたったの 10% なので、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ます、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3.13）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3.13 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3.14）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 3.14 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3.15)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3.15 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3.16)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」のことです「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。



▲図3.16 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

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。(表 3.1)

▼表 3.1 セキュリティグループの設定

| | |
|-------------|---------------------|
| セキュリティグループ名 | ec2-security-group |
| 説明 | Allow from anywhere |

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 3.17）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。

^{*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。

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

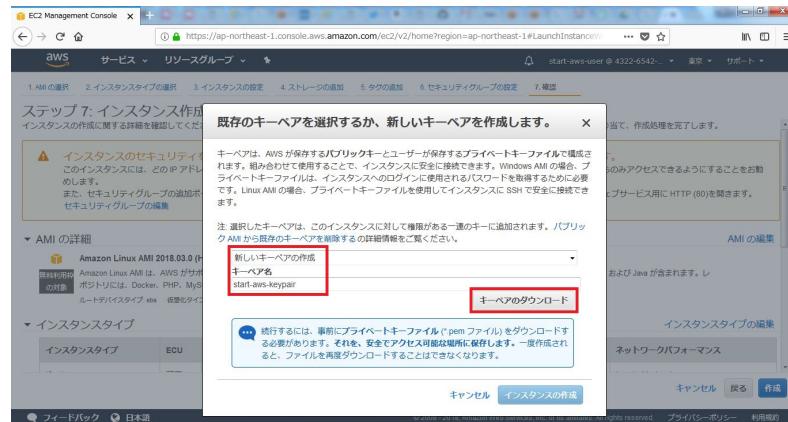
^{*12} サーバにどこからでも接続できてしまうのはよくないので、できれば SSH でのアクセス元も制限したいのですが、ここで制限をかけるとたとえば「出先でモバイル Wi-Fi に繋いだとき」などに接続元の IP アドレスが変わってファイアウォールで阻止され、あなた自身もサーバに接続できなくなってしまいます。今自宅において今後も自宅からしかサーバに SSH で接続しない！ 自宅の IP アドレスは固定だから絶対に変わらない！ という人だけ、ルールのソースを「マイ IP」にして説明に「IP address of my house」と書いておいてください。こうするとあなたの自宅の IP アドレス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



▲図 3.17 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

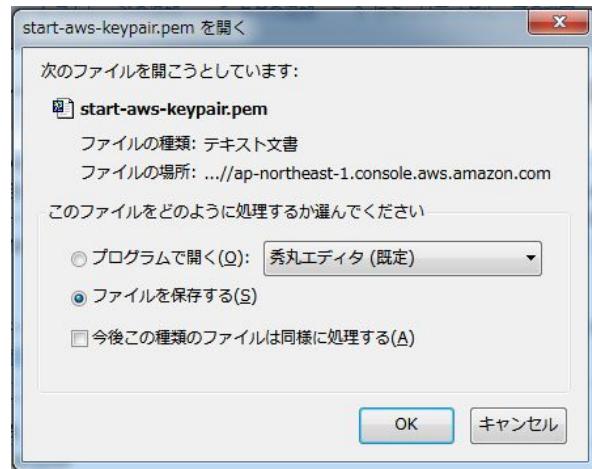
「作成」をクリックすると「既存のキーペアを選択するか、新しいキーペアを作成します。」と表示（図 3.18）されます。「新しいキーペアの作成」を選んでキーペア名を「start-aws-keypair」にしてください。



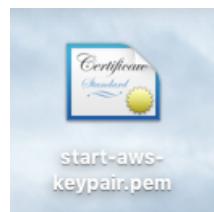
▲図 3.18 キーペア名を「start-aws-keypair」にして「キーペアのダウンロード」をクリック

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 3.19）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 3.20、図 3.21）しておいてください。



▲図 3.19 ダウンロードしたキーペア（start-aws-keypair.pem）を保存



▲図 3.20 Mac の人は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



▲図 3.21 Windows の人も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) はこれ以降二度とダウンロードできません。家の鍵と同じで、キーべ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。

キーべ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 3.22）してください。



▲図 3.22 キーべ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 3.23) と表示されます。

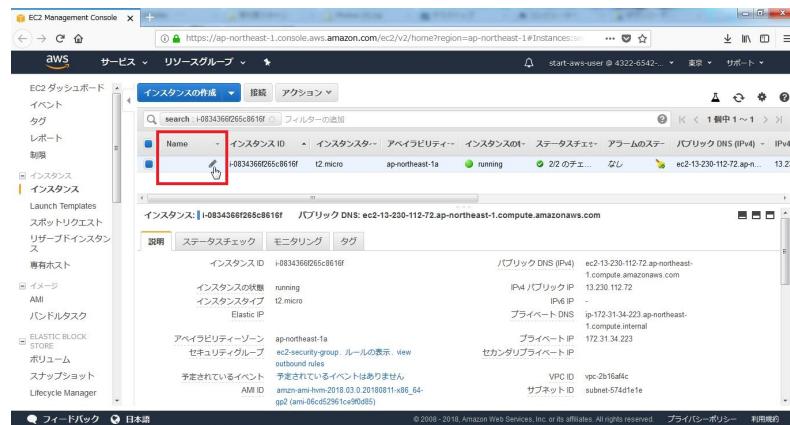


▲図 3.23 「インスタンスは現在作成中です」と表示されたらインスタンス ID をクリック

「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。

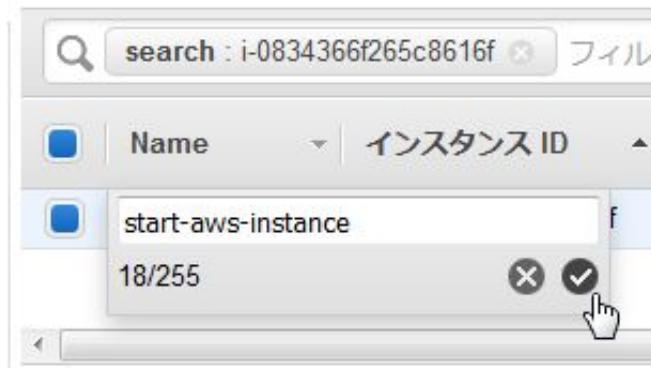
3.2.9 作成したインスタンスに名前をつける

作成したインスタンスが表示 (図 3.24) 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3.24 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3.25）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。



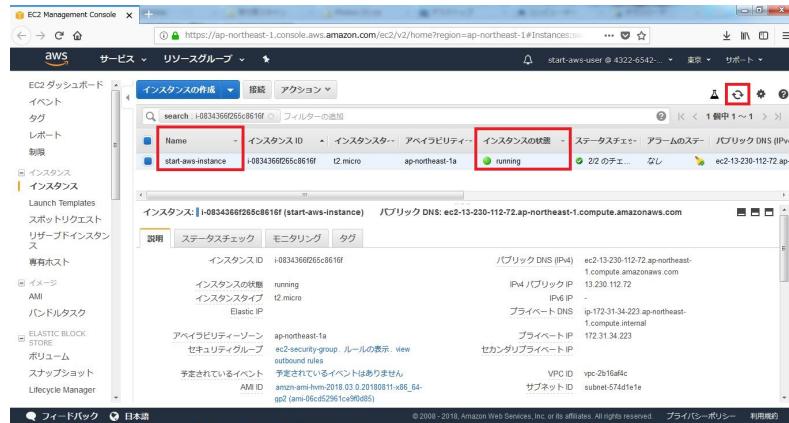
▲図3.25 Nameに「start-aws-instance」と書いたらチェックボタンを押す

- 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3.26）したらインスタンスの作成は完了です。^{*13}

^{*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。

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 3.26 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PE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PE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PEアを再発行すればよい
- C. キーPEアをなくしたらそのサーバには二度とログインできない

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「立ち上がる」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ので筆者は「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。

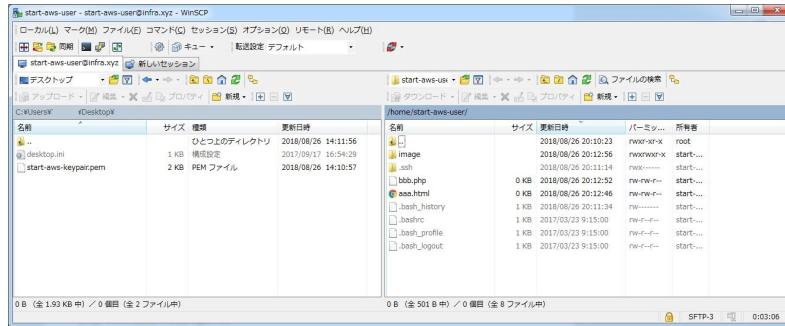
3.3 サーバに「入る」とは？

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。

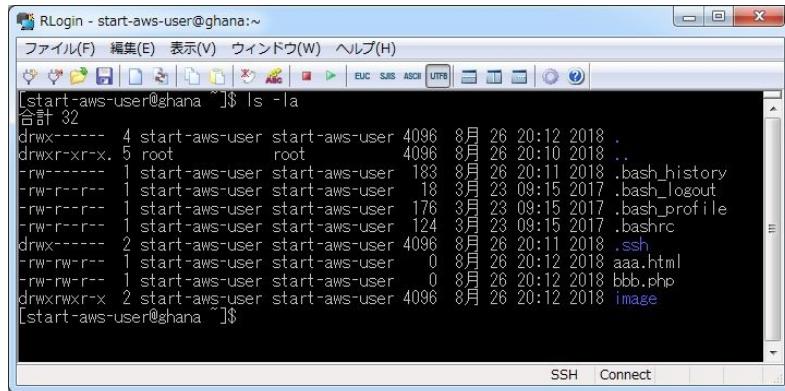
「今までサーバになんか入ったことない！」という方でも、こんな画面（図 3.27）を見たことはありませんか？



▲図 3.27 WinSCP でサーバに「入る」

これは「SCP 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 3.28）も見たことはありませんか？



▲図 3.28 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。エンジニア以外の方には、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 SCP クライアントの右側とターミナルを見比べてみてください。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て、少しほんの「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、ステップ6で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^{*14}とは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と似ている仕組みに「FTP」^{*15}や「Telnet」^{*16}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ふ、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。通信データはきちんと全身タイツを着て正体が見えない状態で流れしていくので、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「SFTP」^{*17}や「SCP」^{*18}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。

*¹⁴ Secure Shell の略。

*¹⁵ File Transfer Protocol の略。

*¹⁶ Telecommunication network の略。

*¹⁷ SSH File Transfer Protocol の略。

*¹⁸ Secure Copy の略。

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^{*19}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 SSH の鍵認証で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タブ（図 3.29）にある「IPv4 パブリック IP」をメモ（表 3.2）してください。



▲図 3.29 「説明」タブの「IPv4 パブリック IP」をメモしておこう

「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」

*19 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。

▼表 3.2 インスタンスの IPv4 パブリック IP

| 例 | IPv4 パブリック IP |
|---------------|---------------|
| 13.230.112.72 | |

と表示（図 3.30）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 3.30 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。

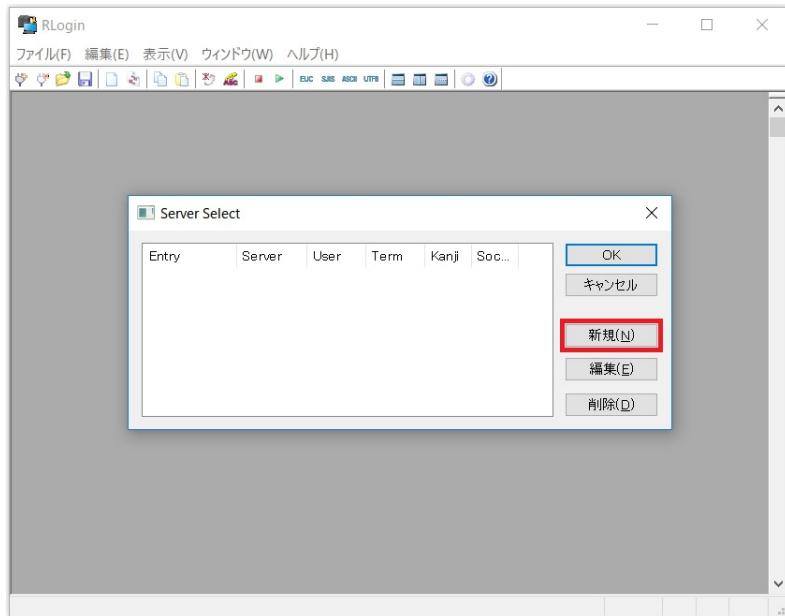
3.4 SSH でサーバに入ってみよう

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 3.31）をダブルクリックして RLogin を起動（図 3.32）してください。起動したら「新規」をクリックします。

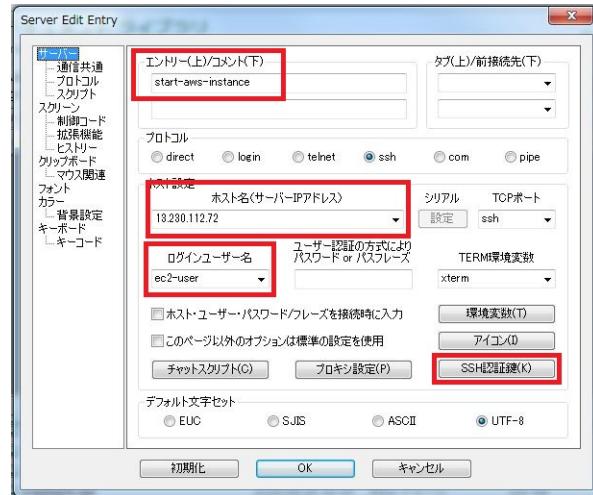


▲図 3.31 RLogin.exe をダブルクリック



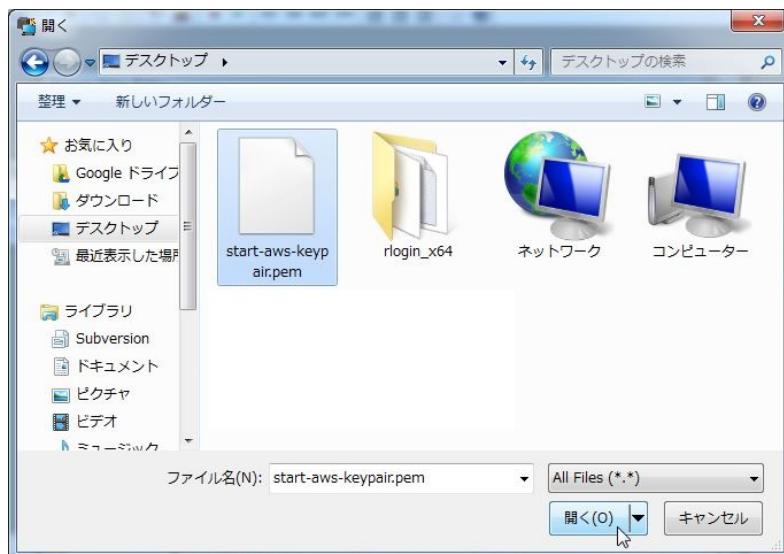
▲図 3.32 RLogin が起動したら「新規」をクリック

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 3.33）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 Amazon Linux AMI を使って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。



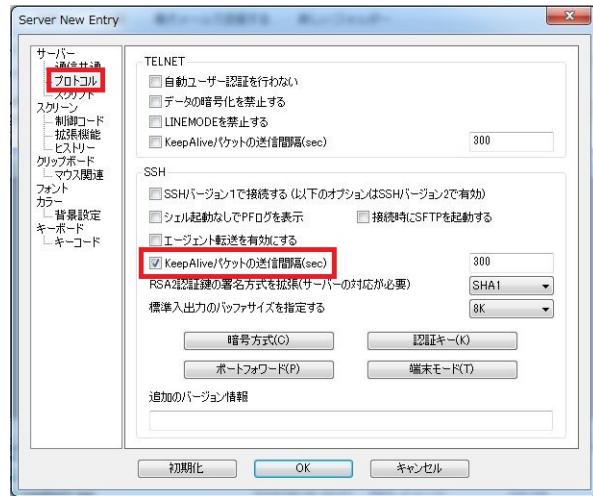
▲図 3.33 「ホスト名（サーバー IP アドレス）」と「ログインユーザー名」を入力

続いて「SSH 認証鍵」をクリック（図 3.34）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。



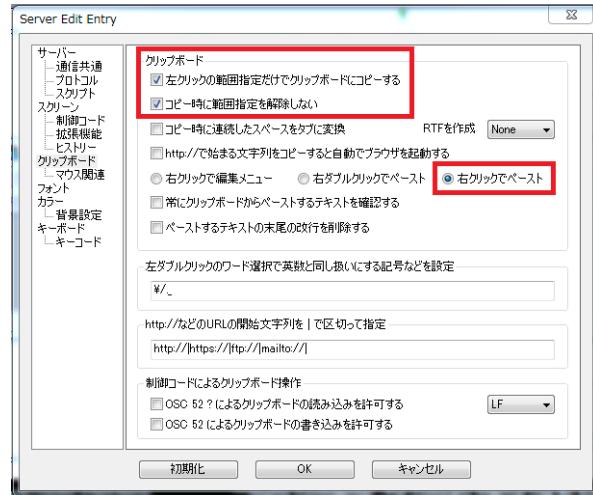
▲図 3.34 「SSH 認証鍵」をクリックして「start-aws-keypair.pem」を選択

次に左メニューで「プロトコル」を選択（図 3.35）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。



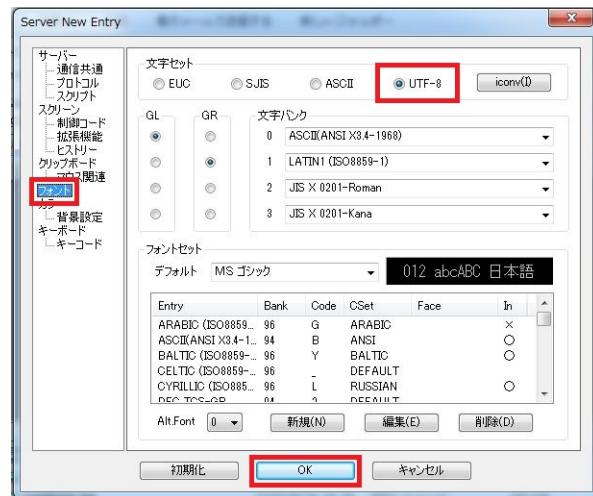
▲図 3.35 「KeepAlive パケットの送信間隔(sec)」にチェックを入れる

続いて左メニューで「クリップボード」を選択（図 3.36）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3.36 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3.37）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。



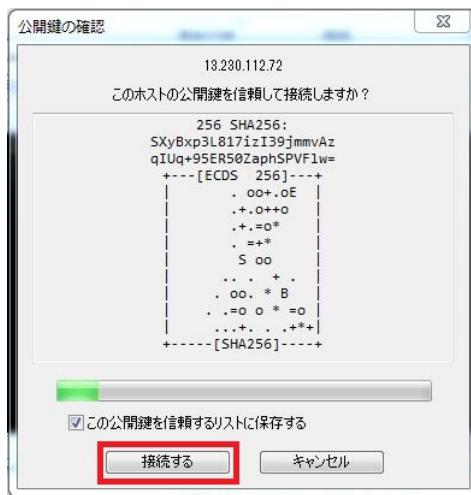
▲図3.37 文字セットを「UTF-8」に変更

設定が保存できたら「OK」をクリック（図3.38）してください。



▲図 3.38 設定が保存できたら「OK」をクリック

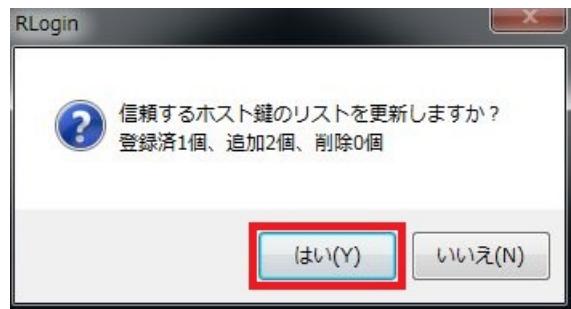
すると初回のみ、この「公開鍵の確認」が表示（図 3.39）されます。これは「初めて入るサーバだけど信頼していいですか？本当に接続しますか？」と聞かれているので、「接続する」をクリックしてください。サーバにはそれぞれフィンガープリントという固有の指紋があるため、下部の「この公開鍵を信頼するリストに保存する」にチェックが入っていれば RLogin が覚えていてくれて、次回以降は「これは前に信頼していいって言われたサーバだ！」と判断してそのまま接続させてくれます。



▲図 3.39 「公開鍵の確認」が表示されたら「接続する」をクリック

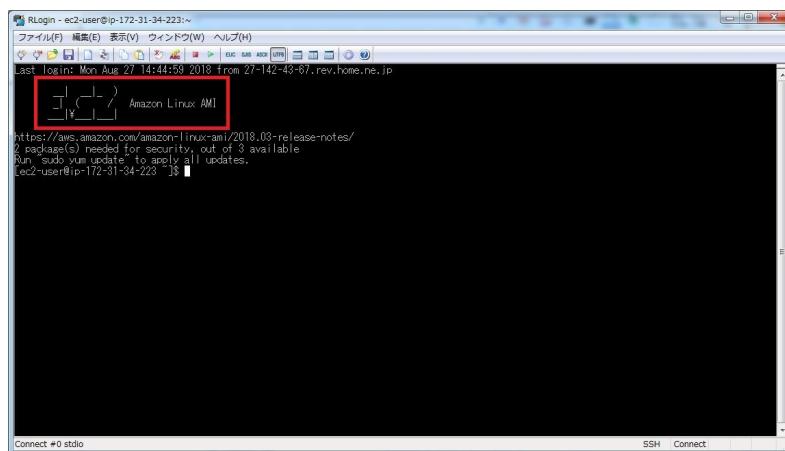
続いて「信頼するホスト鍵のリストを更新しますか？」と聞かれたら「はい」をクリッ

ク（図3.40）してください。



▲図3.40 「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

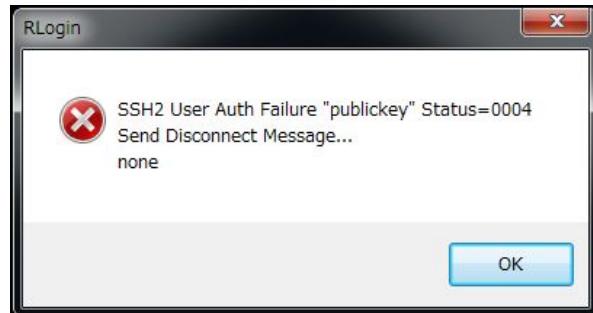
「Amazon Linux AMI」と表示（図3.41）されたら無事サーバに入っています。SSHでのログイン成功、おめでとうございます！



▲図3.41 「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3.42）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 3.42 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

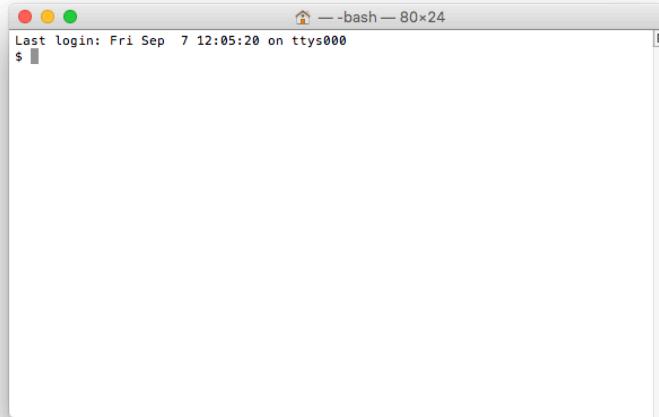
「接続済みの呼び出し先が一定の時間を過ぎても正しく応答しなかったため、接続できませんでした。」というエラーメッセージが表示（図 3.43）されてしまった場合は、「ホスト名（サーバー IP アドレス）」に書いた「IPv4 パブリック IP」が間違っているものと思われます。「ホスト名（サーバー IP アドレス）」の IP アドレスを確認してみてください。



▲図 3.43 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 お使いのパソコンが Mac の場合

Mac を使っている方は、ターミナル（図 3.44）を起動してください。



▲図 3.44 最初からインストールされている「ターミナル」を使おう

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（図 3.45）すれば起動できます。



▲図 3.45 どこにあるのか分からなかったら Spotlight で「ターミナル」と検索

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。もし「start-aws-keypair.pem」を保存した場所がデスクトップ^{*20}以外の場合は適宜書き換えてください。

```
chmod 600 ~/Desktop/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、「start-aws-keypair.pem」を保存した場所がデスクトップ以外だった場合はこちらも適宜書き換えてください。

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

^{*20} 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。

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)?
```

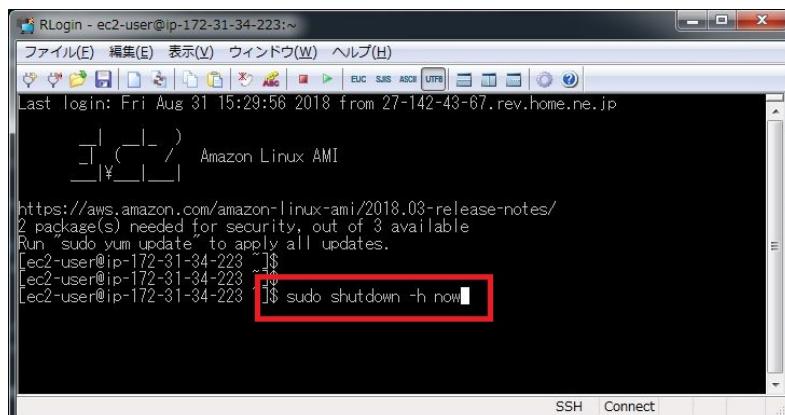
「Amazon Linux AMI」と表示されたら無事サーバに入っています。おめでとうございます！

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
```

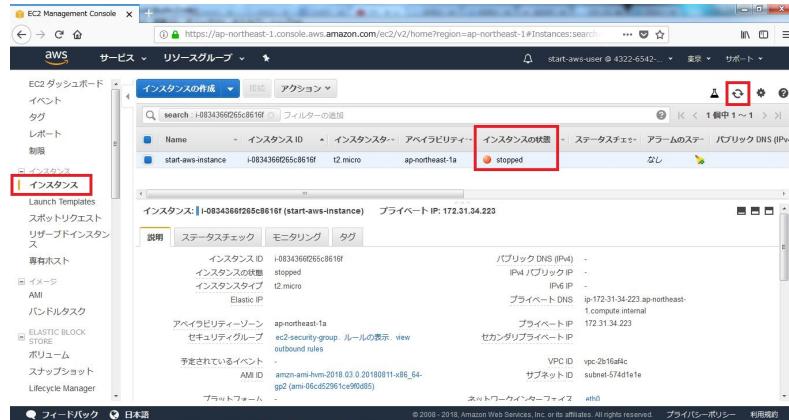
と入力したらEnterキー（もしくはReturnキー）を押してください。（図3.46）これはサーバをシャットダウンするコマンドです。シャットダウンが気軽に試せるのは勉強用のインスタンスならではですね。



▲図3.46 サーバをシャットダウンしてみよう

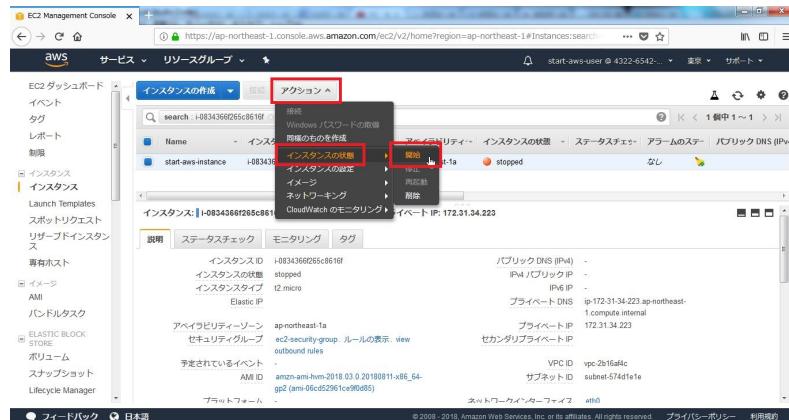
3.4 SSH でサーバに入ってみよう

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 3.47）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 3.47 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 3.48）してください。



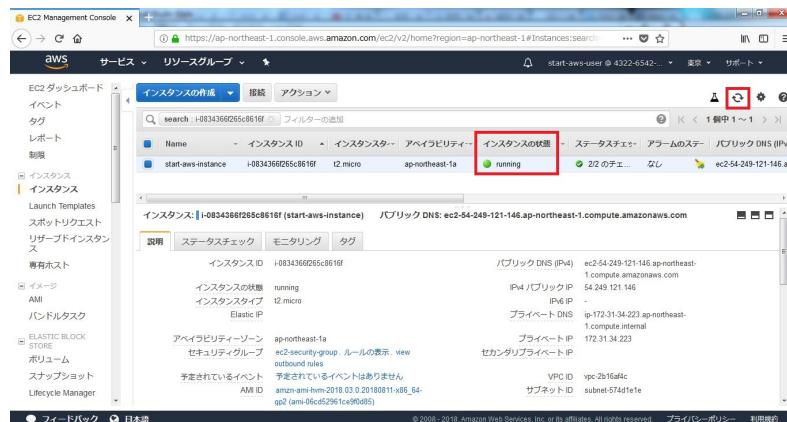
▲図 3.48 アクション>インスタンスの状態>開始をクリック

「これらのインスタンスを開始してよろしいですか?」(図 3.49)と確認が出るので「開始する」をクリックしてください。



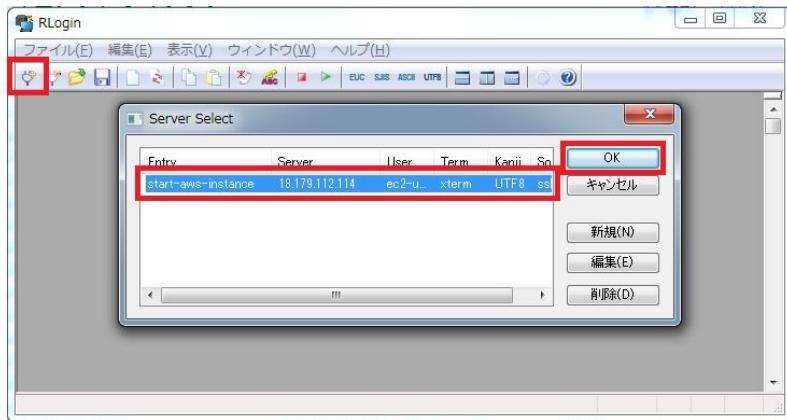
▲図 3.49 「開始する」をクリック

インスタンスの状態が「pending」になったら「running」になるまで、何度か右上の更新マークをクリック(図 3.50)してみてください。



▲図 3.50 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 3.51)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 3.51 「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 3.52）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。



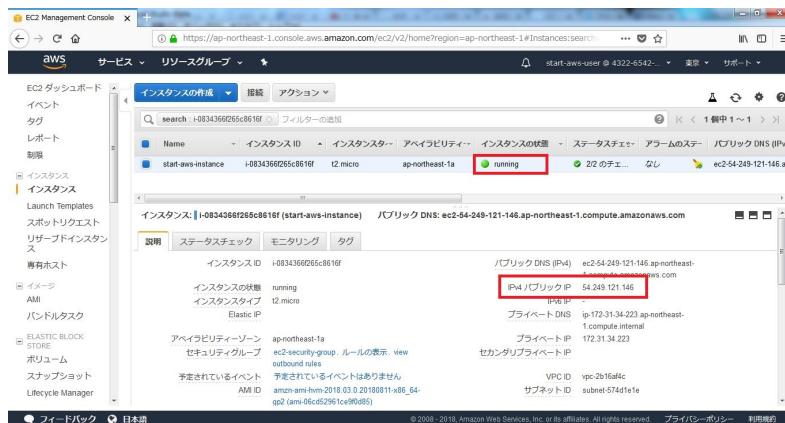
▲図3.52 シャットダウンして起動したらサーバに繋がらなくなった！

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 3.53）Rlogin のエラーメッセージには「応答しなかった」とありましたが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3.53 「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 3.3）してみてください。

▼表 3.3 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 |

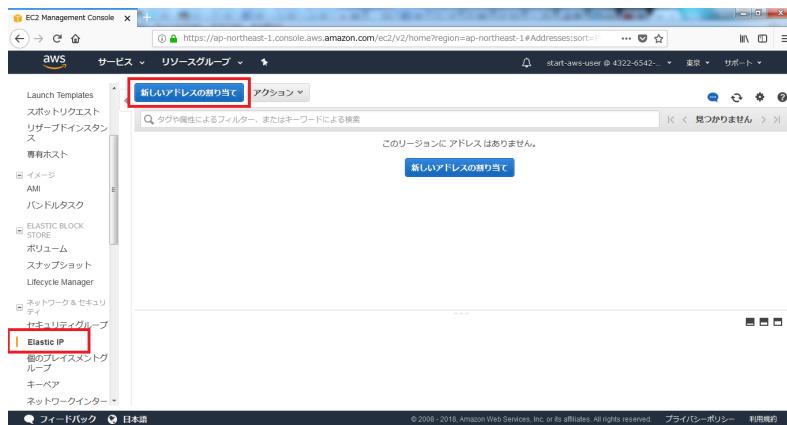
そうなんです。シャットダウンする前と、起動しなおした今の「IPv4 パブリック IP」を比較すると、まったく別の IP アドレスになっているのです。

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^{*21}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ブ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ね・・・。

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「IPv4 パブリック IP」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。

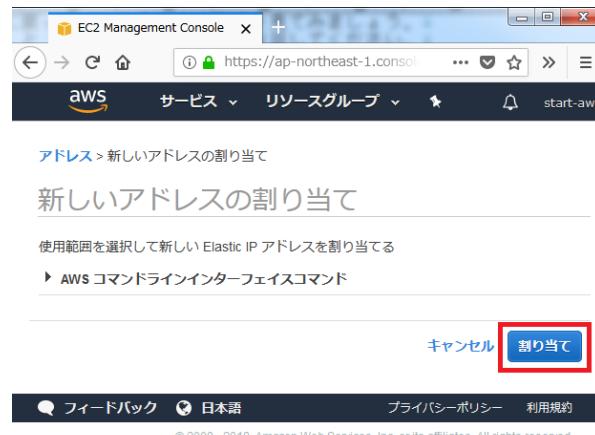
EC2 ダッシュボードの左メニューで「ネットワーク & セキュリティ」の中にある「Elastic IP」をクリック（図 3.54）してください。まだ Elastic IP がひとつもないため、「このリージョンに アドレス はありません。」と表示されます。「新しいアドレスの割り当て」をクリックします。

*21 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。

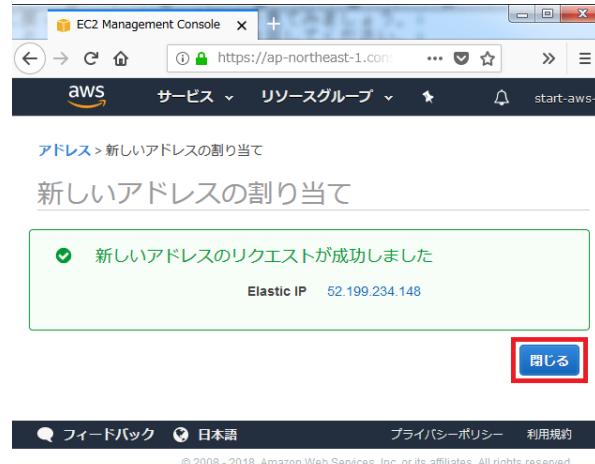


▲図3.54 「Elastic IP」で「新しいアドレスの割り当て」をクリック

「割り当て」をクリック（図3.55）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3.56）されたら「閉じる」をクリックします。

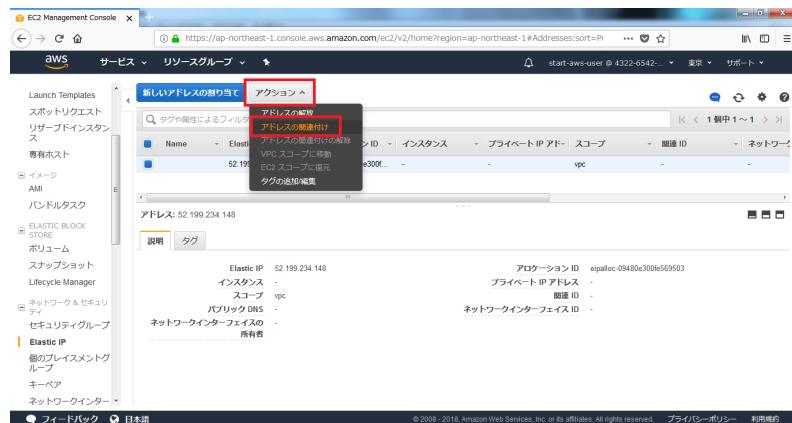


▲図3.55 「割り当てる」をクリック



▲図 3.56 「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 3.57）してください。



▲図 3.57 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 3.58）します。



▲図 3.58 「start-aws-instance」をクリック

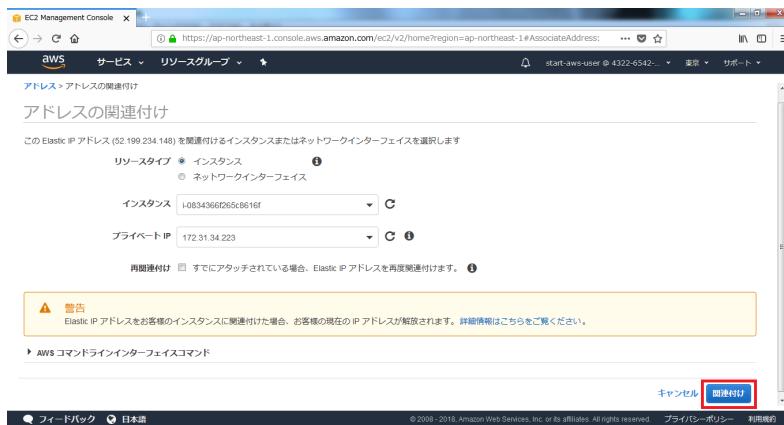
プライベート IP は「プライベート IP の選択」と書かれた箇所をクリックすると、「start-aws-instance」の IP アドレスが表示されるのでクリック（図 3.59）します。



▲図 3.59 「start-aws-instance」の IP アドレスをクリック

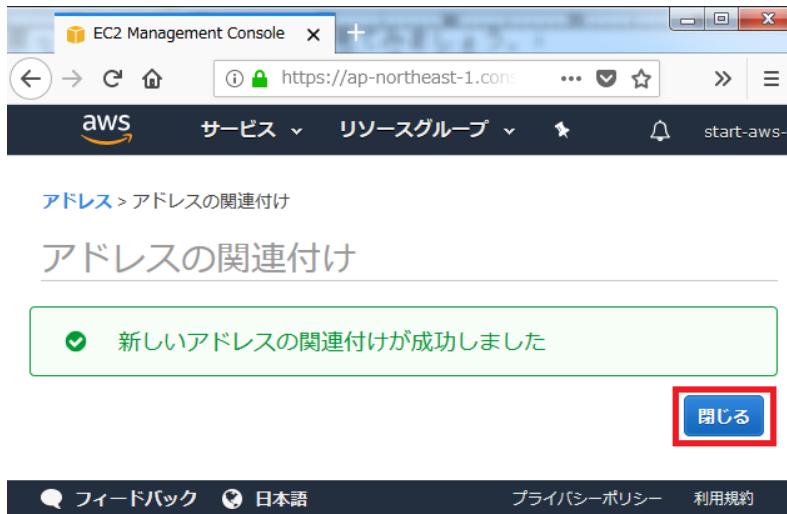
再関連付けにチェックは入れません。この状態（図 3.60）で「関連付け」をクリックしてください。

3.4 SSH でサーバに入ってみよう



▲図 3.60 「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 3.61）されたら「閉じる」をクリックします。



▲図 3.61 「閉じる」をクリック

Elastic IP をしっかりメモ（表 3.4）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 3.62）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 3.62 「閉じる」をクリック

▼表 3.4 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^{*22}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、そのドメイン名を使ってサーバに入れるようになります。

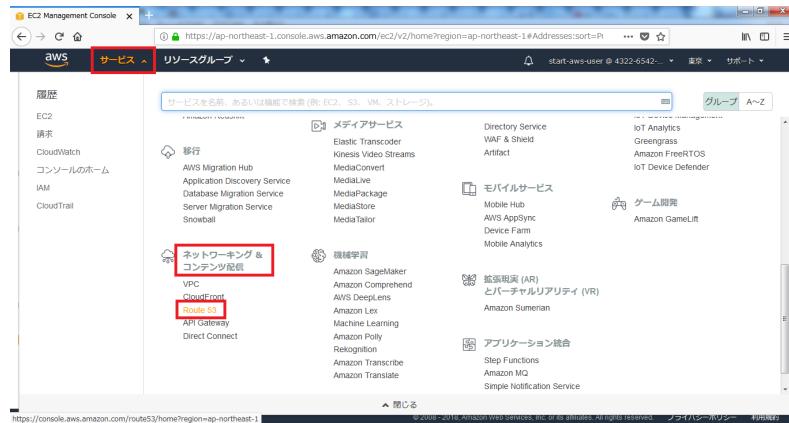
皆さんは「DNS をはじめよう」で買った自分のドメインを持っているので、早速ドメイン名と IP アドレスの組み合わせをネームサーバに登録してみましょう。ドメイン名の登録は Route53 で行います。

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 3.63）をクリックしてください。

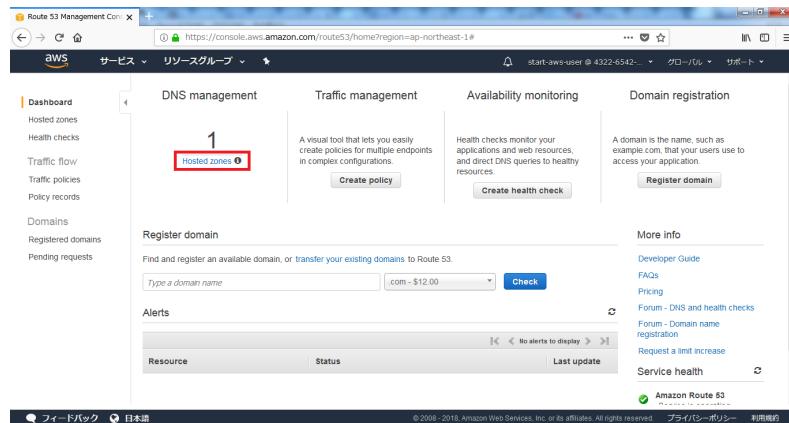
^{*22} 今回はサーバが 1 台だけなので直接ログインしていますが、通常は踏み台サーバ（英語だと Bastion host と呼ばれることが多い）を用意して個々のウェブサーバやデータベースサーバには踏み台を経由しないと入れない、という構成にすることが多いです。サーバは大切な箱入り娘なので、怖いお父さん（踏み台サーバ）を介さないと話しかけられないようにすることで安全性を高めているのです。

3.4 SSH でサーバに入ってみよう



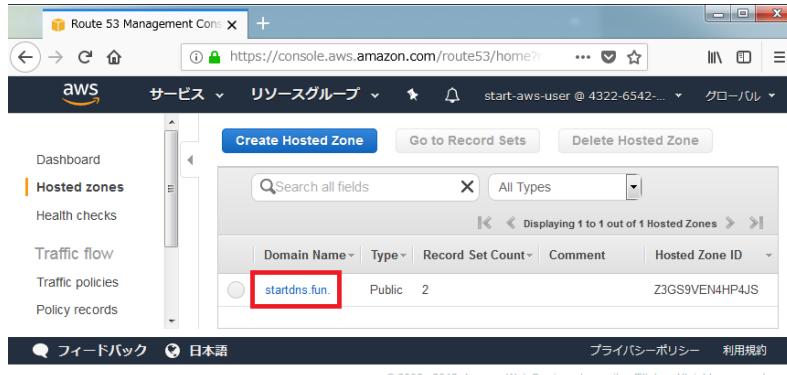
▲図 3.63 サービス>ネットワーキング & コンテンツ配信>Route53

Route53 ダッシュボードを開いたら DNS management の「Hosted zones」をクリック（図 3.64）します。



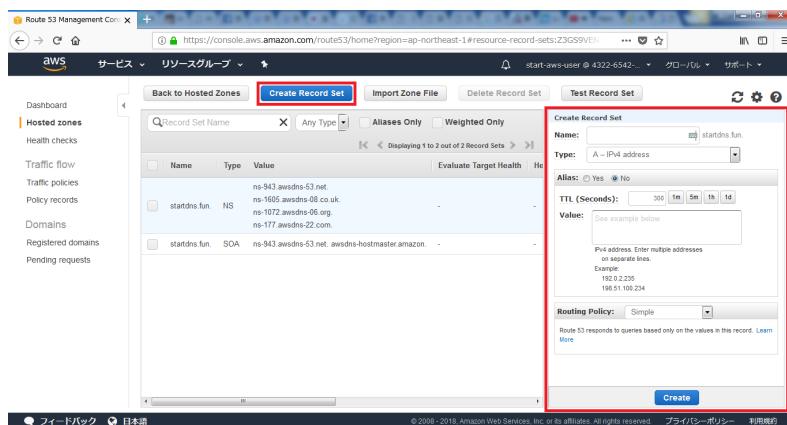
▲図 3.64 「Hosted zones」をクリック

Domain Name の自分のドメイン名（筆者の場合は startdns.fun）をクリック（図 3.65）します。



▲図 3.65 自分のドメイン名をクリック

新しくリソースレコード^{*23}を作りたいので「Create Record Set」をクリック（図 3.66）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 3.66 「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 3.67）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。

^{*23} 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

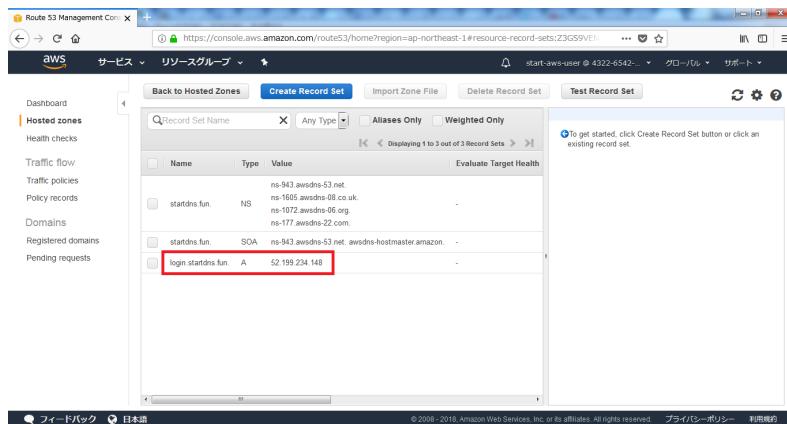
Routing Policy: Simple

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

Create

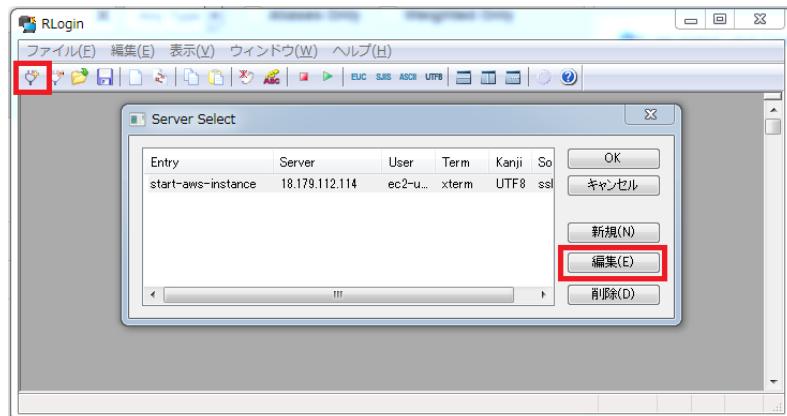
▲図 3.67 Name には「login」、Value には先ほどメモした Elastic IP を入力

これで「login. 自分のドメイン名」(図 3.68) という A レコードが作成できました。

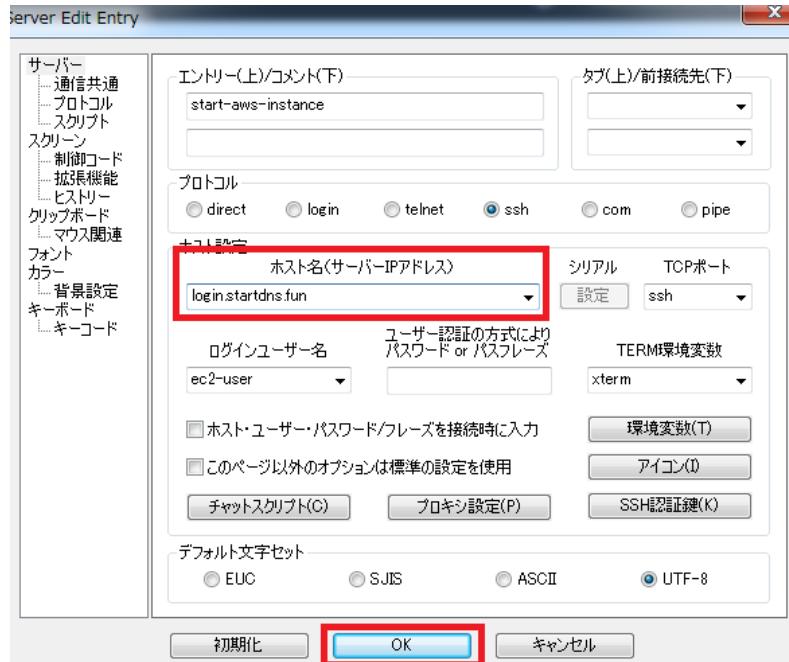


▲図3.68 「login. 自分のドメイン名」というAレコードができた

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3.69）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3.70）したらOKをクリックします。

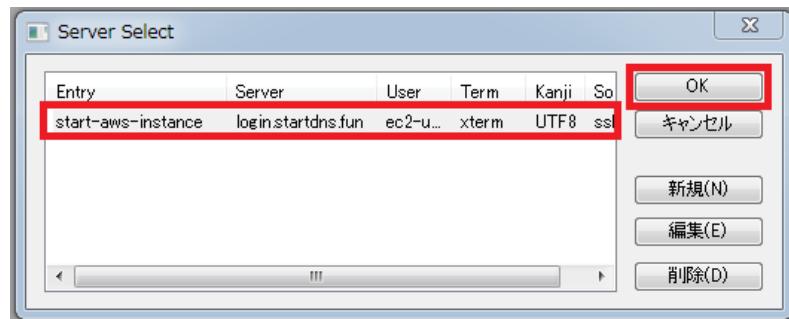


▲図3.69 「編集」をクリック



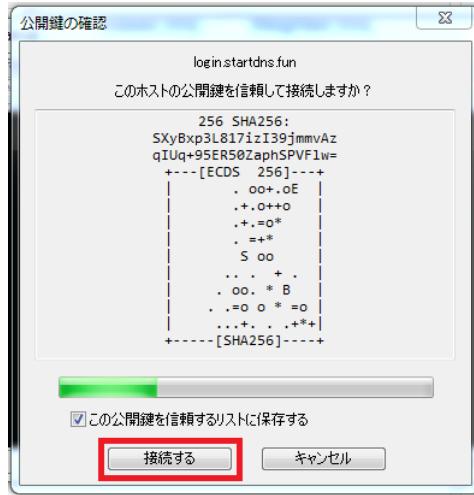
▲図 3.70 「ホスト名（サーバー IP アドレス）」を「login. 自分のドメイン名」にして OK をクリック

「start-aws-instance」を選択（図 3.71）したら「OK」をクリックしてください。



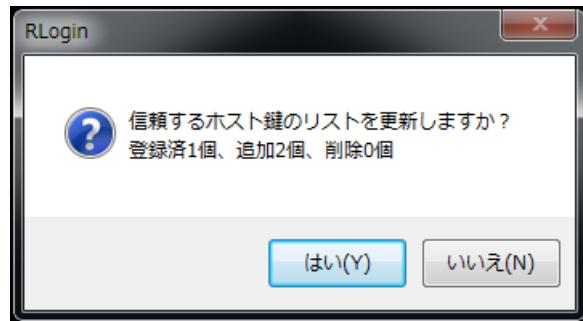
▲図 3.71 「start-aws-instance」を選択したら「OK」をクリック

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 3.72）されますが「接続する」をクリックしてください。



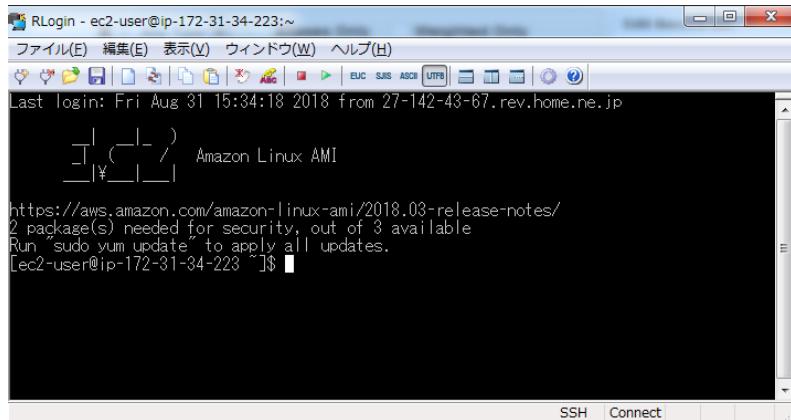
▲図3.72 「公開鍵の確認」が表示されたら「接続する」をクリック

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3.73）してください。



▲図3.73 「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3.74）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 3.74 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。

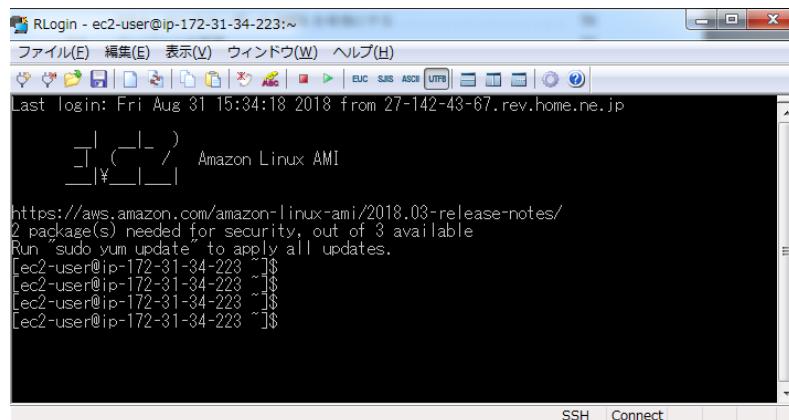
3.5 ターミナルでサーバを操作・設定してみよう

ようやくサーバに安定して入れるようになったので、ここからはターミナルの基本的な操作を試してみましょう。

3.5.1 ターミナルの基本操作に慣れよう

プロンプトとは？

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 3.75）普通に改行されますよね。



▲図 3.75 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

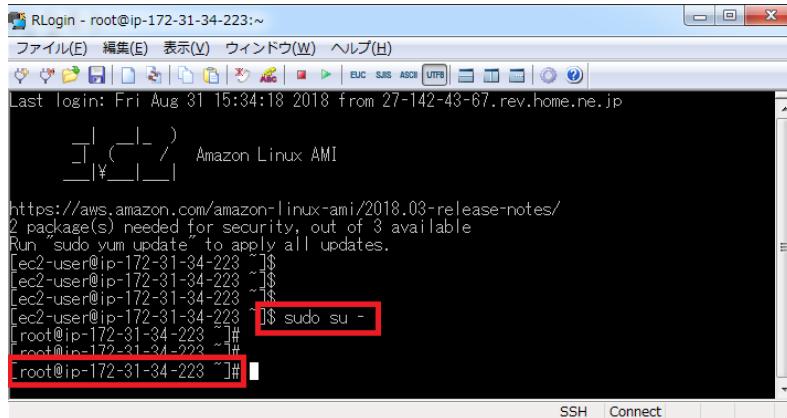
このとき左側にずっと出ている次のような表示は「プロンプト」といって、ログインしているユーザ名やサーバの名前などが表示されています。

[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す。

```
$ sudo su -
```

と書いて Enter キーを押すと root になれます。(図 3.76) 「\$」はプロンプトを表していますので入力しないでください。root になれたらまた何回か Enter キーを押して改行してみましょう。



▲図 3.76 sudo su -を書いて Enter キーを押すと root になれる

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？

```
[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「\$」で全権を持っている root だと「#」という表示になります。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、例として書いてある部分のプロンプトが「\$」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。例に「\$」や「#」が書いてあってもターミナルで「\$」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ので失敗した

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と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よね。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 3.77）が表示されました。

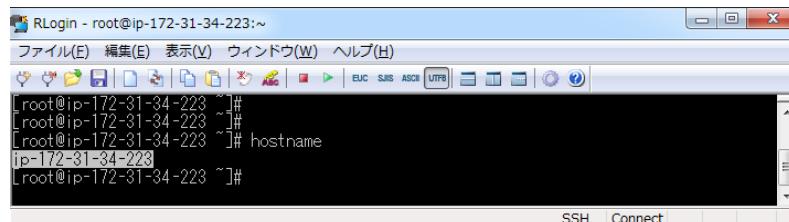


▲図 3.77 「sudooo: command not found」というエラーが表示された

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 3.78）選択してから Ctrl+C を押す必要はありません。



▲図 3.78 マウスで選択するだけでコピーできる

逆にコピーした内容をターミナルへペーストしたいときはターミナル上で右クリックするだけです。Ctrl+P 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 -
```

まずは 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、データベース接続時に必要な 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。Amazon Linux の yum では「PHP は Apache のモジュール（部品）」という扱いなので、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^{*24}も入れてくれます。

```
# 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。

^{*24} 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。

3.5.3 OS の基本設定をしておこう

タイムゾーンの設定

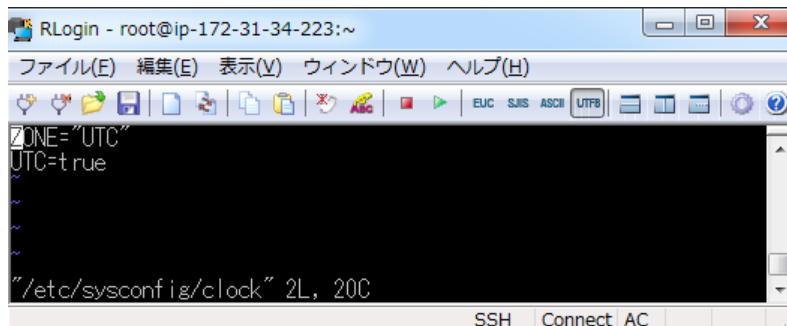
`date` (デート) コマンドでサーバの時間を確認してみましょう。日本はいま 18:14 なのですが、サーバの時間は 9:14 でずれてしまっています。

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

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
```

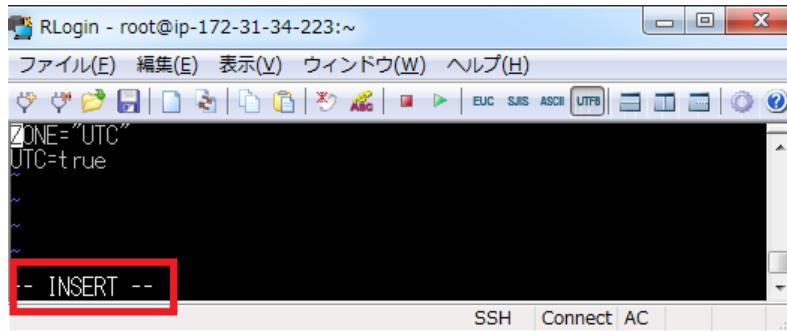
`vi` (ブイアイ) はテキストファイルを編集するためのコマンドです。`vi` コマンドでファイルを開くと、最初は次のような「閲覧モード」の画面（図 3.79）が表示されます。閲覧モードは「見るだけ」なので編集ができません。



▲図 3.79 `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」^{*25}に変わります。（図 3.80）左下に「-- INSERT --」と表示されていたら「編集モード」です。

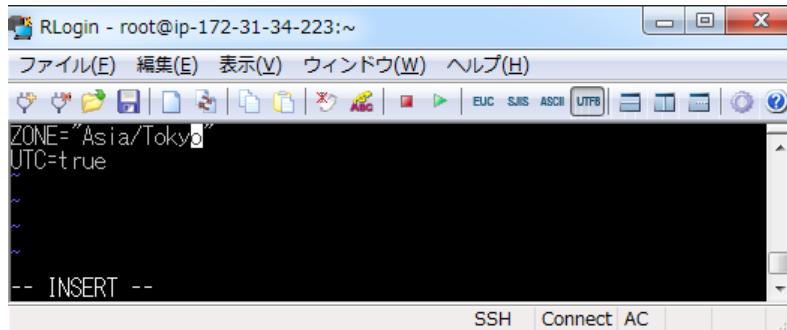
^{*25} ここでは初心者の方でも直感的に分かるよう「閲覧モード」「編集モード」と呼んでいますが、正しくは「ノーマルモード」「インサートモード」です。



A screenshot of the RLogin terminal window titled "RLogin - root@ip-172-31-34-223:~". The terminal shows the command "ZONE='UTC'" and "UTC=true" on the screen. At the bottom of the terminal window, the status bar displays "SSH Connect AC". A red box highlights the text "-- INSERT --" which is displayed at the bottom of the terminal window.

▲図 3.80 i(アイ)を押すと「-- INSERT --」と表示される「編集モード」になった

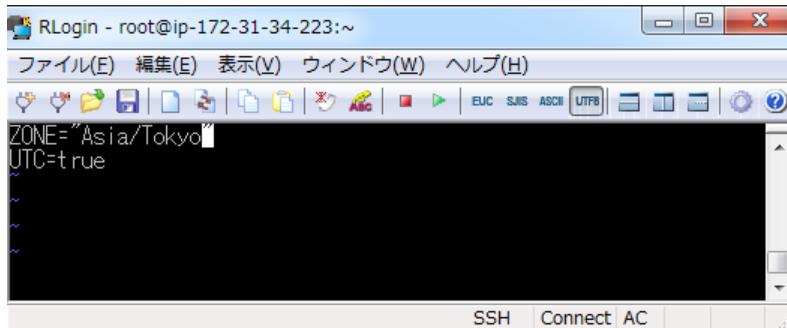
「編集モード」になるとファイルが編集できるようになります。それでは「ZONE="UTC"」を「ZONE="Asia/Tokyo"」(図 3.81)に書き換えてください。



A screenshot of the RLogin terminal window titled "RLogin - root@ip-172-31-34-223:~". The terminal shows the command "ZONE='Asia/Tokyo'" and "UTC=true" on the screen. At the bottom of the terminal window, the status bar displays "SSH Connect AC". A red box highlights the text "-- INSERT --" which is displayed at the bottom of the terminal window.

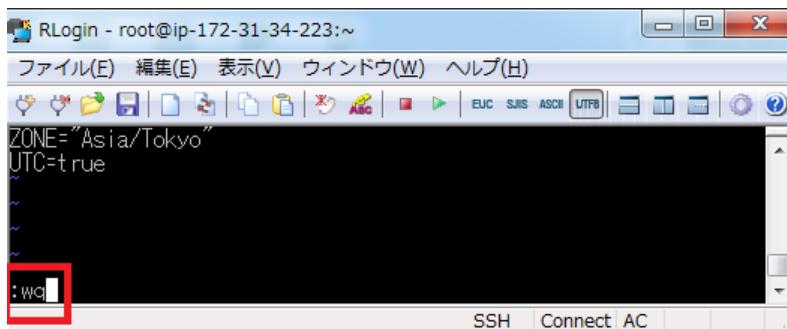
▲図 3.81 「ZONE="UTC"」を「ZONE="Asia/Tokyo"」に書き換える

「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 3.82)



▲図 3.82 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」*26と入力して Enter キーを押せば変更が保存されます。(図 3.83)



▲図 3.83 「: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訳が分からなくなってしまって「今の全部なかつたことにしたい！ 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!」*27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*28でファイルの中身を確認してみましょう。

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

*27 保存せずに強制終了 (quit!) という命令なので q! です。

*28 cat は猫ではなく「conCATenate files and print on the standard output」の略だそうです。筆者はいつも「猫がファイルの中身を全部出して見せてくるんだ」と考えることでちょっと幸運になっています。

```
# 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。

```
# 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

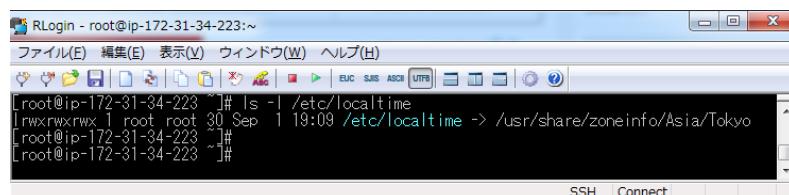
```
# ln -sf /usr/sh
```

まで打ってからTabキーを押すと次のように自動補完されます

```
# ln -sf /usr/share/
```

シンボリックリンクが生成されたかlsコマンド^{*29}で確認してみましょう。(図3.84)

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



▲図3.84 シンボリックリンクが生成された

*29 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。

locale（言語）の設定

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）コマンドでファイルの中身を確認してみましょう。

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

通常の Linux サーバであればこの設定だけでいいのですが、Amazon Linux の場合、AMI からインスタンスを復元^{*30}すると今修正した「LANG=ja_JP.UTF-8」がそっと元の「LANG=en_US.UTF-8」に戻ってしまいます。

元に戻らないよう次のファイルも編集しておきましょう。

```
# vi /etc/cloud/cloud.cfg
```

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」にして一番下に次の 1 行を書き足してください。

^{*30} 「AMI からインスタンスを復元」については第 7 章「サーバのバックアップを取っておこう」と第 8 章「ELB と Auto Scaling で複数台のサーバを運用しよう」で解説します。

```
locale: 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 コマンドでファイルの中身を確認してみましょう。

```
# cat /etc/cloud/cloud.cfg
# WARNING: Modifications to this file may be overridden by files in
# /etc/cloud/cloud.cfg.d

# If this is set, 'root' will not be able to ssh in and they
# will get a message to login instead as the default user (ec2-user)
disable_root: true

# This will cause the set+update hostname module to not operate (if true)
preserve_hostname: true

datasource_list: [ Ec2, None ]

repo_upgrade: security
repo_upgrade_exclude:
- kernel
- nvidia*
- cudatoolkit

mounts:
- [ ephemeral0, /media/ephemeral0 ]
- [ swap, none, swap, sw, "0", "0" ]
# vim:syntax=yaml

locale: ja_JP.UTF-8
```

history の設定

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。

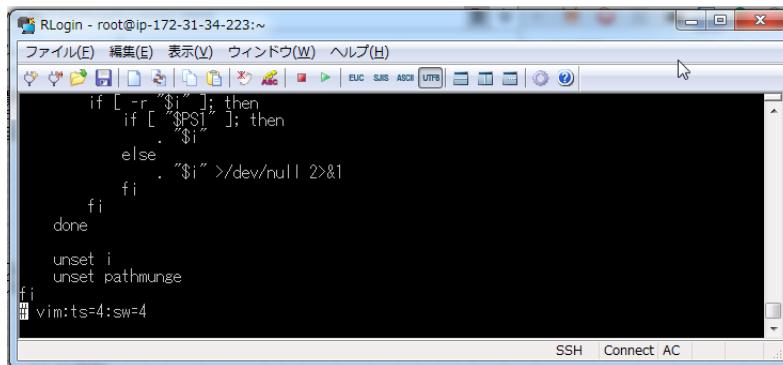
```
# history
1 sudo su -
2 yum install -y jwhois mysql
3 yum install -y php72 php72-mbstring php72-mysqlnd
4 chkconfig --add httpd
5 chkconfig httpd on
6 chkconfig --list httpd
7 date
8 vi /etc/sysconfig/clock
9 cat /etc/sysconfig/clock
10 ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
11 ls -l /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 3.85）してください。



▲図 3.85 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。編集できたら tail (テイル) コマンド^{*31}でファイルの中身を確認してみましょう。

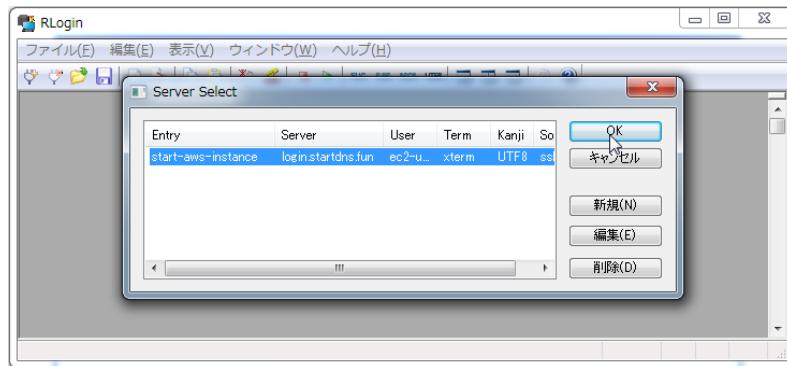
^{*31} tail コマンドは名前のとおりファイルの尻尾、つまり末尾を表示するコマンド。引数で「-2」と指定すれば末尾から 2 行、「-10」と指定すれば末尾から 10 行が表示されます。

```
# 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3.86)してみてください。今度はさっきと同じ設定でそのまま接続できるはずです。



▲図3.86 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。^{*32}

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。

皆さんの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するとと思います。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」という画面を開いて操作します。ディレクトリ^{*33}を開いて移動するときはダブルクリックの代わりに cd^{*34}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もダブルクリックでフォルダを

^{*32} ちなみに history はターミナルを「exit」して閉じ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×を押してウィンドウを閉じたり、ネットワークがぶつつと切れてしまったりするとその分は記録されずに消えてしまいます。

^{*33} Linux ではフォルダのことをディレクトリと呼びます。

^{*34} change directory の略。

開く代わりに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う。^{*35}

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。

^{*35} 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の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。



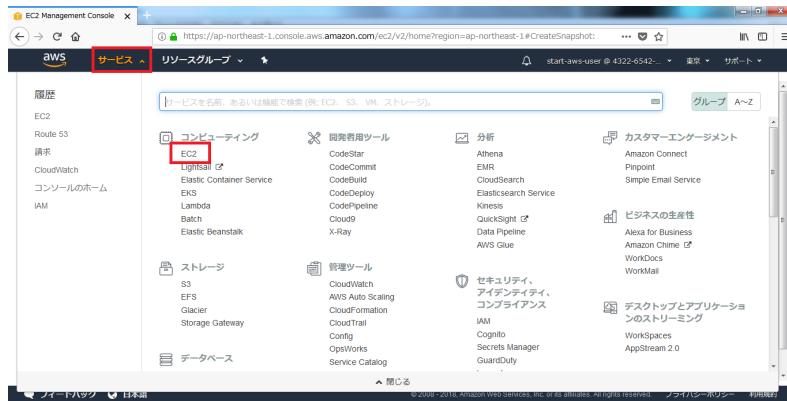
▲図4.1 <http://login.自分のドメイン/> が見られなかった

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）のリクエストをすべて遮断していることが原因です。ポート番号とはサーバという家や、その手前にあるセキュリティグループという壁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、HTTPは80番のドアを、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EC2の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。

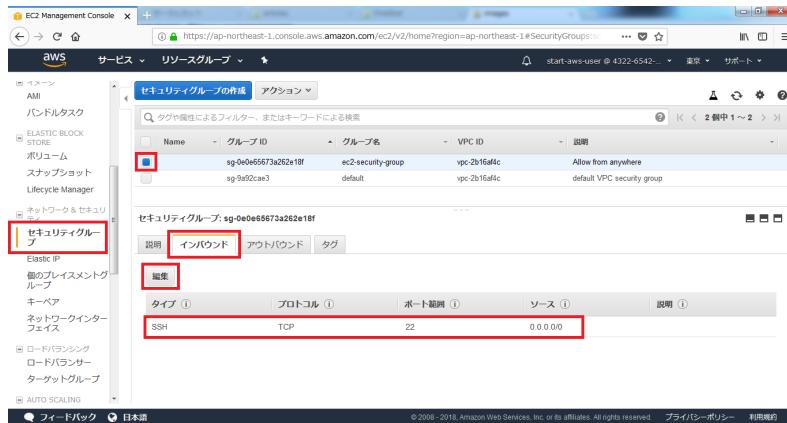
4.1.1 セキュリティグループで 80 番ポートの穴あけをしよう

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。



▲図4.3 現状はSSH（ポート番号22番）しか通れないので「インバウンド」タブの「編集」をクリック

「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。

「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」と自動入力されるので、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

4.1 ウェルカムページを見てみよう

「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。

The screenshot shows the AWS EC2 Management Console with the 'Security Groups' section open. A new rule is being created under the 'Inbound' tab. The rule details are as follows:

| Type | Protocol | Port Range | Source | Description |
|------|----------|------------|-----------|-------------|
| HTTP | TCP | 80 | 0.0.0.0/0 | |
| SSH | TCP | 22 | 0.0.0.0/0 | |

▲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。

The screenshot shows the 'Amazon Linux AMI Test Page'. The page content includes:

- A message for general public: "This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured."
- A note for website administrators: "If you are the website administrator:
The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.
If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name 'webmaster' and directed to the website's domain should reach the appropriate person.
For example, if you experienced problems while visiting www.example.com, you should send e-mail to 'webmaster@example.com'."
- Information about the Amazon Linux AMI: "The Amazon Linux AMI is a supported and maintained Linux image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2). It is designed to provide a stable, secure, and high performance execution environment for applications running on Amazon EC2. It also includes packages from the Amazon Linux repository, which contains many popular open source projects and many popular AWS libraries and tools. Amazon Web Services provides ongoing security and maintenance updates to all instances running the Amazon Linux AMI. The Amazon Linux AMI is provided at no additional charge to Amazon EC2 users."
- A note about welcome.conf: "You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf."
- A note about image usage: "You are free to use the images below on Apache and Amazon Linux AMI powered HTTP servers. Thanks for using Apache and the Amazon Linux AMI!"
- A footer with the Apache logo and version 2.4.

▲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？

バーチャルホストというのは1台のサーバ上で2つ以上のウェブサイトを扱う運用方法のことです。

たとえば筆者の趣味が「スイーツブッフェ」と「読書」だったとして、1台のウェブサーバで

- sweets.startdns.fun
- book.startdns.fun

という2つのバーチャルホストを作つてやれば、同じサーバ上で別々のサイトを運用できます。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（ウェブサイト）があって、その部屋を訪れるお客さんたち（サイトを見る人）がいる、と例えると分かりやすいでしょうか。

これからApacheで自分のサイト用にこのバーチャルホストを作つてみます。

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つてみましょう。^{*2}

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。

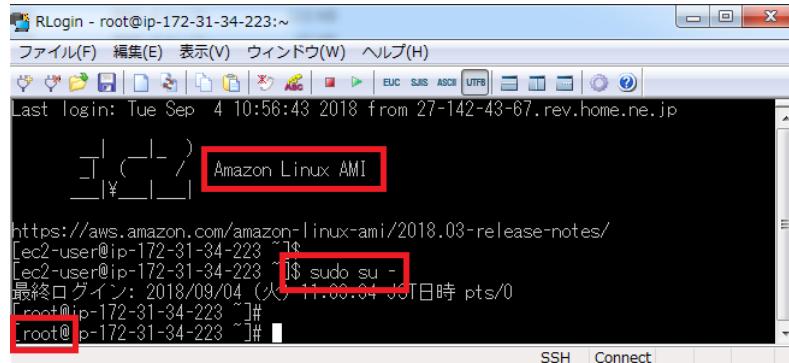
```
ssh ec2-user@login. 自分のドメイン名 -i ~/Desktop/start-aws-keypair.pem
```

「Amazon Linux AMI」と表示されたら「sudo su -」^{*3}でrootになります。（図4.7）

^{*2} 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のですが、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つてみよう」と思ったときに役立つはずです。

^{*3} 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

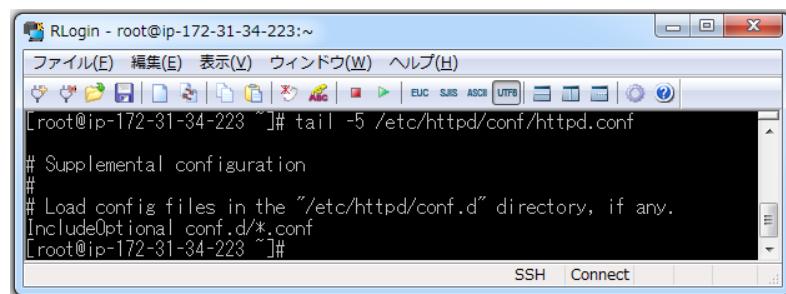
```
$ sudo su -
```



▲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
```



▲図 4.8 tail コマンドで httpd.conf の最後の 5 行を見てみよう

すると最後の行に

コマンドです。「他のユーザー = root」の場合はユーザー名を書かなくと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。

```
IncludeOptional conf.d/*.conf
```

とあるため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内の「*.conf」というファイルをインクルード（取り込み）^{*4}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"
```

ServerRoot^{*5}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」が実際は「/etc/httpd/conf.d/*.conf」であることが分かりました。

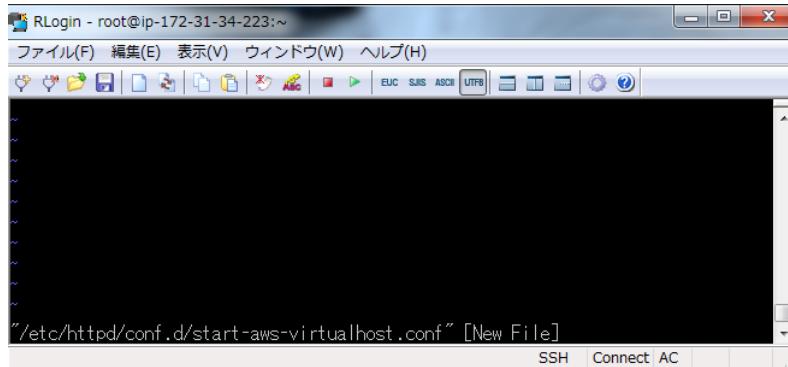
では自分のドメインのサイト用にバーチャルホストを「/etc/httpd/conf.d」の下で作成してみましょう。viコマンドで新しい設定ファイルを作ります。

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

こんな画面（図4.9）で「"/etc/httpd/conf.d/start-aws-virtualhost.conf" [New File]」と表示されましたか？

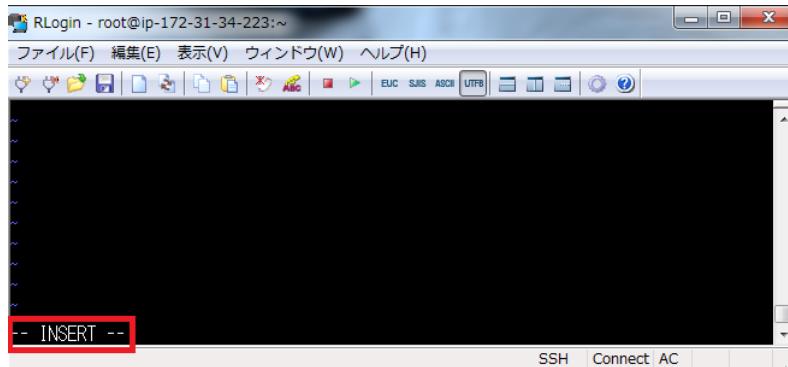
^{*4} IncludeディレクティブやIncludeOptionalディレクティブについては<https://httpd.apache.org/docs/2.4/ja/mod/core.html#include>を参照。

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



▲図 4.9 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 4.10) 左下に「-- INSERT --」と表示されていたら「編集モード」です。



▲図 4.10 i (アイ) を押すと「-- INSERT --」と表示される「編集モード」になった

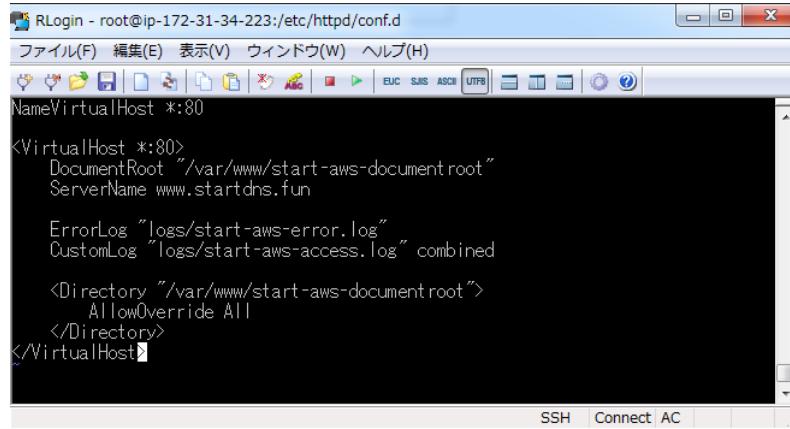
「編集モード」になったら次のようにバーチャルホストの設定を書いていってください。
「自分のドメイン」のところはそのまま日本語で書かず、自分のドメイン名に置き換えて
ください。

```
<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 4.11) すると左下の「-- INSERT --」が消えて再び「閲覧モード」になります。



▲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

「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 4.12)

```
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)
NameVirtualHost *: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 4.12 「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次のようになっています。

ServerName www.startdns.fun

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^{*6}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。つまりバーチャルホストの設定で

```
DocumentRoot "/var/www/start-aws-documentroot"
ServerName www.startdns.fun
```

と書いてあったら、

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

に置いた「index.html」が <http://www.startdns.fun/index.html> で見られるということです。

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ようです。ですのでこのディレクトリを作成しましょう。

^{*6} <https://httpd.apache.org/docs/2.4/ja/mod/core.html#documentroot>

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

ディレクトリを作るには `mkdir7` 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
```

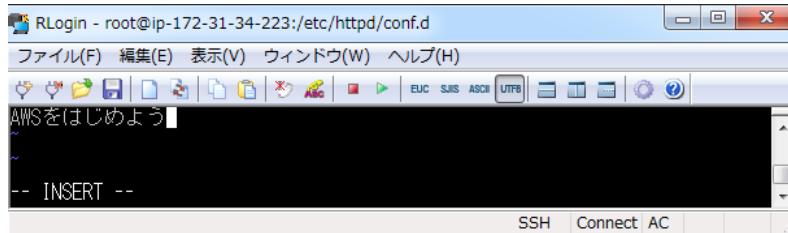
4.2.5 index.html を置いてみよう

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」になってしまふので、`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。

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

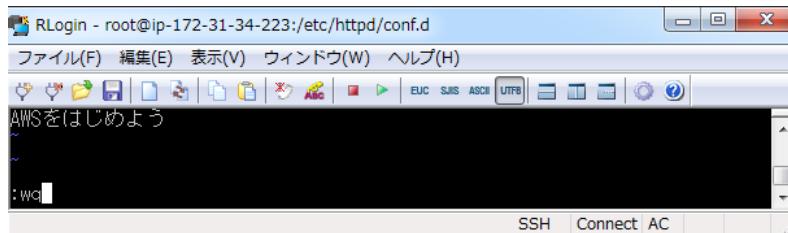
`i` (アイ) を押して「編集モード」に変わったら「AWS をはじめよう」と書いてみましょう。(図 4.13)

⁷ `mkdir` は MaKe DIRectory の略。



▲図 4.13 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 4.14)



▲図 4.14 ESC を押して「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。ウェブサーバに「自分のウェブサイトのページ見せて」と頼んだら、ちゃんとウェブ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。「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/index.html
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 4.15) するとなんと「アクセスしようとしているサイトを見つけられません。」と表示されました。

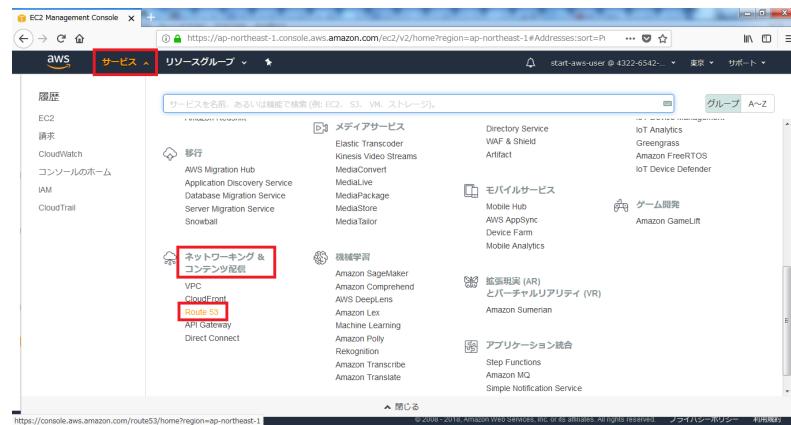


▲図 4.15 ブラウザで「www. 自分のドメイン」を開いたらエラーになってしまった

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。

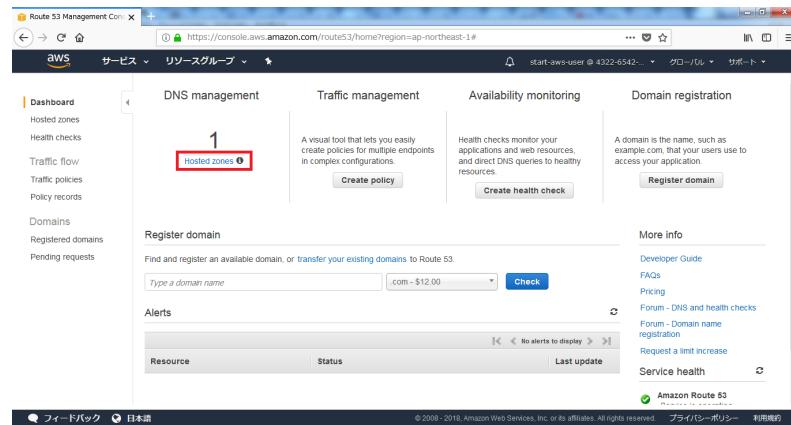
4.3.1 自分のサイトのドメイン名を作ろう

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 4.16) をクリックしてください。



▲図 4.16 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 4.17）します。



▲図 4.17 「Hosted zones」をクリック

Domain Name の自分のドメイン名（筆者の場合は startdns.fun）をクリック（図 4.18）します。

4.3 Route53 で自分のサイトのドメインを設定しよう

The screenshot shows the AWS Route 53 Management Console. On the left, there's a sidebar with options like Dashboard, Hosted zones (which is selected), Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, and Pending requests. The main area has a search bar and a table with columns: Domain Name, Type, Record Set Count, Comment, and Hosted Zone ID. One row in the table is highlighted with a red box, showing 'startdns.fun.' in the Domain Name column. At the bottom of the page, there are links for Feedback, Japanese, Privacy Policy, and Terms of Service.

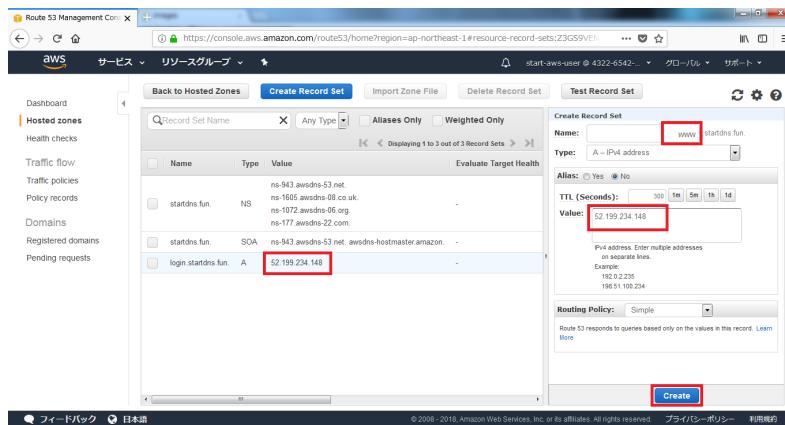
▲図 4.18 自分のドメイン名をクリック

「Create Record Set」をクリック（図 4.19）してください。すると右側にリソースコードの情報を入力するフォームが出てきます。

This screenshot shows the 'Create Record Set' dialog box overlaid on the Route 53 console. The dialog has fields for Name (set to 'startdns.fun.'), Type (set to 'A - IPv4 address'), Value (containing '192.0.2.235\n192.0.1.234'), TTL (Seconds) (set to 300), and Routing Policy (set to 'Simple'). There are also checkboxes for Alias and Weighted. The 'Create' button at the bottom is highlighted with a red box.

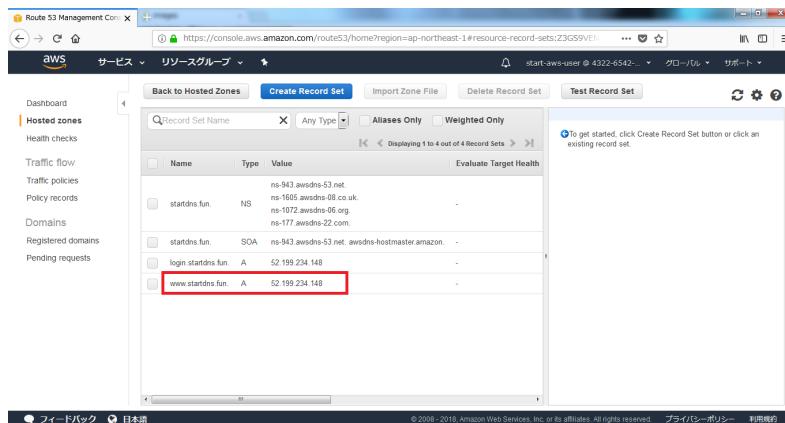
▲図 4.19 「Create Record Set」をクリック

Name には「www」、Value には Elastic IP を入力（図 4.20）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 4.20 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 4.21) という A レコードが作成できました。



▲図 4.21 「www. 自分のドメイン名」という A レコードができた

4.3.2 【ドリル】/(スラッシュ)で終わる URL を開いたときに index.html 以外を返したい

問題

<http://www.startdns.fun/>を開くとファイル名は指定していないのに index.html が表示されます。このようにファイル名を省略して「/ (スラッシュ)」で終わる URL を

開いたとき、index.htmlではなくneko.htmlを表示させたかったらApacheの設定ファイル(httpd.conf)でどの設定を書き換えればよいでしょうか？

- A. DirectorySlash ディレクティブ
- B. DirectoryIndex ディレクティブ
- C. Directory ディレクティブ

答え _____

解答

正解はBです。grepコマンドを使ってApacheの設定ファイル(httpd.conf)で「index.html」を検索してみると、次のような設定が見つかります。

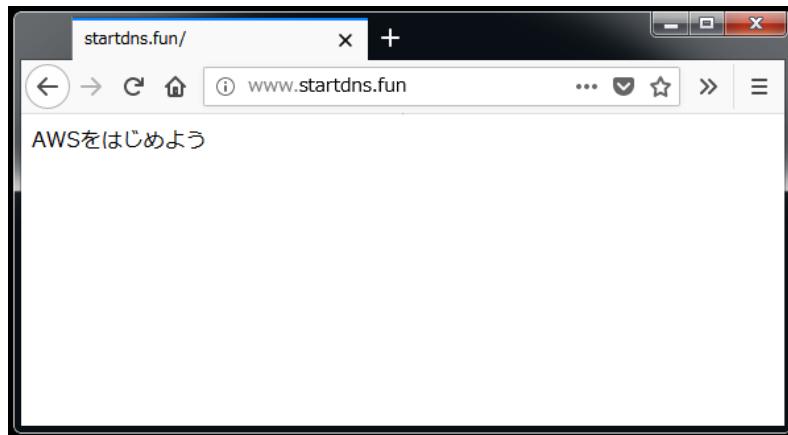
```
# grep index.html /etc/httpd/conf/httpd.conf
DirectoryIndex index.html
```

DirectoryIndexディレクティブではファイル名を省略して「/（スラッシュ）」で終わるURLを開いたときに返すファイルの名前を指定できます。次のように複数指定した場合は「index.html」「index.php」「index.txt」の順に探して、最初に見つかったものを返します。

```
DirectoryIndex index.html index.php index.txt
```

4.3.3 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4.22）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 4.22 index.html の「AWS をはじめよう」が表示された

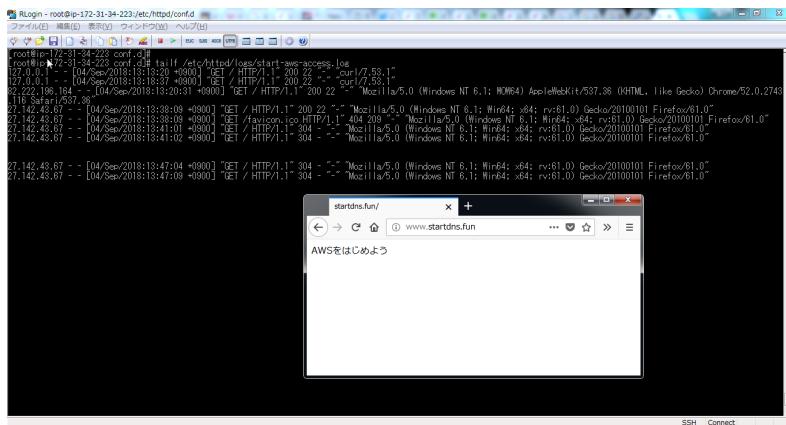
4.3.4 アクセスログとエラーログの大切さ

ブラウザ上ではサイトが表示されましたが、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^{*8}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 4.23) tailf コマンドは Ctrl+C で抜けるまでずっとログの追記が表示され続けます。

^{*8} 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。



▲図 4.23 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
```

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

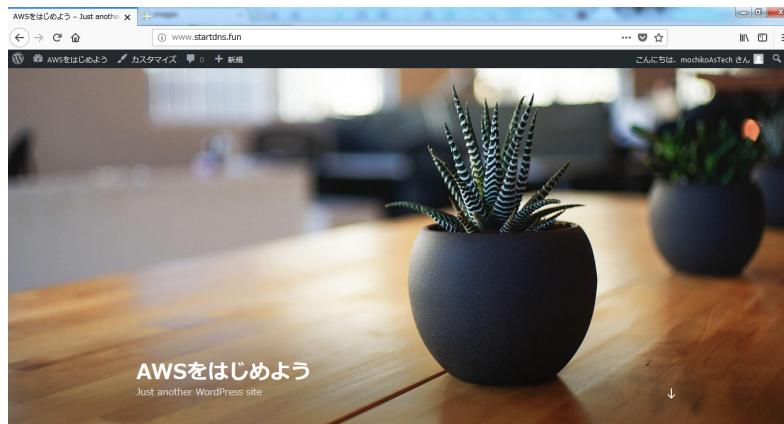
データベースサーバを立てよう

WordPress を使うには MySQL というデータベースが必要です。この章では RDS でデータベースサーバを立てましょう。

5.1 WordPressにはデータベースが必要

5.1.1 CMSとは？

こちらをご覧ください！このおしゃれなブログはWordPressというCMSで動いています。（図5.1）



▲図5.1 WordPressで作ったおしゃれなブログ

CMSとはContents Management Systemの略で、ウェブサイトのコンテンツを管理したり、ウェブページを更新したりするためのソフトのことです。CMSがあればHTMLやCSSなどを自分で直接編集しなくても、Wordのように文章を書いてレイアウトを整え「更新」をクリックするだけでサイトが更新できます。

CMSにはWordPress（ワードプレス）とMovable Type（ムーバブル・タイプ）というツートップがあるのですが、どちらも個人利用であれば無料で使うことができます。今回はCMS界でトップシェアを誇るWordPressを使えるように環境を構築していきましょう。

WordPressの管理画面でブログ記事を書いて保存すると、記事のデータはデータベースに保存されます。そしてページを表示するときはデータベースからデータを取ってきて画面が構成されます。そのためWordPressを使うにはMySQLというデータベースが必須となっています。

【コラム】MySQL と MariaDB

データベースは Oracle Database や PostgreSQL など MySQL 以外にも色々な種類があるのですが、その中で MySQL はもっとも普及しているデータベースでシェア全体の 8 割以上を占めています。

MySQL は以前は Sun Microsystems という会社が所有してオープンソースで公開していましたが、2009 年に Sun が Oracle に買収されたため現在は Oracle のものになっています。Oracle に買収された後もありがたいことに MySQL は引き続きオープンソースで公開されており現在も無償で使えますが、Oracle はもともと Oracle Database という有償のデータベースも販売しているため MySQL はその競合にあたり、今は平気でもいつかはサポートが終了したりオープンソースで公開されなくなるのでは？ という懸念もあります。

また MySQL を最初に作った開発者はすでに Oracle を離れており、MariaDB という MySQL から派生した別のデータベースを開発しています。このことから Linux のディストリビューションである CentOS 6 や Amazon Linux では MySQL が採用されていましたが、CentOS 7 及び Amazon Linux 2 からは MariaDB に切り替わっています。しかも yum で MySQL をインストールしようとすると、何も言わずにそっと MariaDB がインストールされる落とし穴仕様です。サーバの仕様を「OS は CentOS 7 で DB は MySQL です」と決めていたのに、環境構築後によくよく確認したら入っていたのは MariaDB だった、这样一个にならないようご注意ください。

上記のような背景から引き続き MySQL を使うべきか、今後は MariaDB に切り替えるべきかという懸念はここ数年ずっとある気がしますが、個人的には Oracle が「MySQL の開発とサポートやめた！」と言い出すまでは積極的に MariaDB に切り替える理由もないかな、という気持ちです。

ちなみに WordPress は MySQL でも MariaDB でも動きますので、今回は深く考えずに MySQL を使いましょう。

5.1.2 EC2 にインストールするか？ RDS を使うか？

AWS で MySQL を使いたいとき、EC2 でインスタンスを作成して yum で MySQL をインストールする^{*1}方法と、Amazon RDS というサービスを使う方法があります。

Amazon RDS は Amazon Relational Database Service の略で、ざっくり言うとデータベースサーバです。データベースに接続することはできますが、MySQL が動いているサーバに SSH でログインすることはできません。RDS ではデータベースエンジンとして MySQL だけでなく Amazon Aurora^{*2}、Microsoft SQL Server、Oracle、PostgreSQL なども選択可能です。

自力でインストールするのではなく RDS を使うことで、データのバックアップや脆弱性パッチの適用といった色々な面倒ごとを RDS にお任せできます。一方で RDS というサービス上で用意されていない機能は、たとえ MySQL で実装されていても使えないため自由度は若干下がります。

旅行代理店のパック旅行を申し込みば、飛行機のチケットを取ってホテルの予約してレンタカーを借りて・・・といったこまごましたことを自分でしなくて済むから楽だけれど、代わりにできることや行けるところは限られていて自由度が下がる、というのと同じですね。

今回は WordPress と繋いで使うだけで自由度の高さは求めていないため運用コストの低い RDS を選択します。

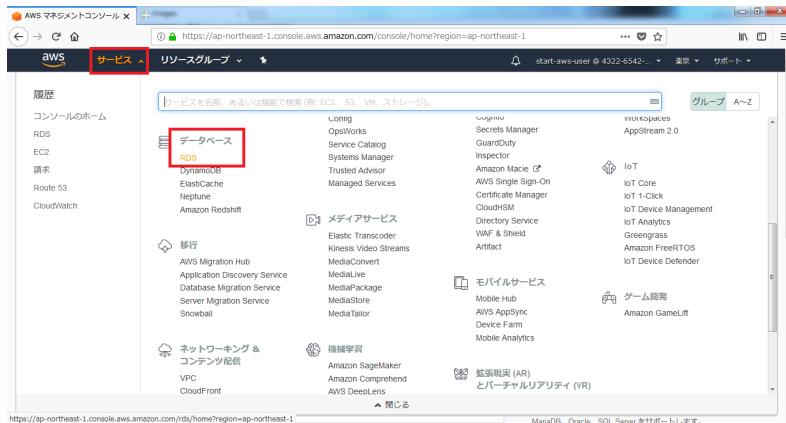
5.2 インスタンスを立てる事前準備

マネジメントコンソールの左上にある「サービス」から、「データベース」の下にある「RDS」（図 5.2）をクリックしてください。

^{*1} サーバに Apache をインストールしてウェブサーバを作ったのと同じように、MySQL をインストールしてデータベースサーバを作る、ということです。既に MySQL がインストールされた AMI からインスタンスを作ることもできます。

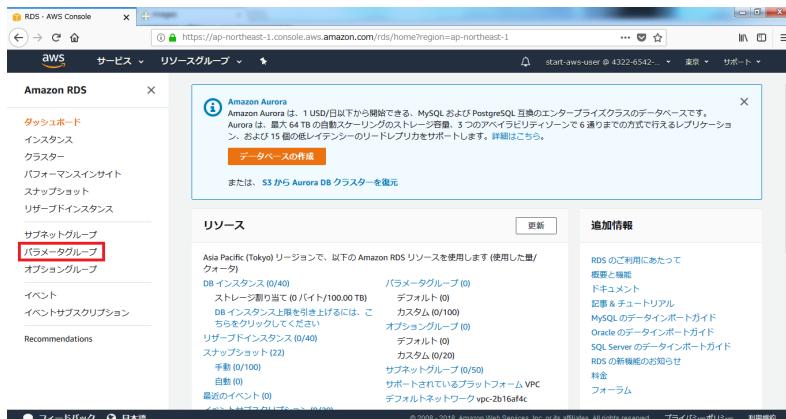
^{*2} 本当は「MySQL の最大 5 倍のパフォーマンス」といわれる MySQL 互換のデータベース、Amazon Aurora を使ったかったのですが AWS の無料利用枠に含まれないため本著では断念しました。WordPress は Aurora でも動くようです。

5.2 インスタンスを立てる事前準備



▲図 5.2 サービス>データベース>RDS

RDS ダッシュボードを開いたら左メニューの「パラメータグループ」をクリック（図 5.3）します。



▲図 5.3 左メニューの「パラメータグループ」をクリック

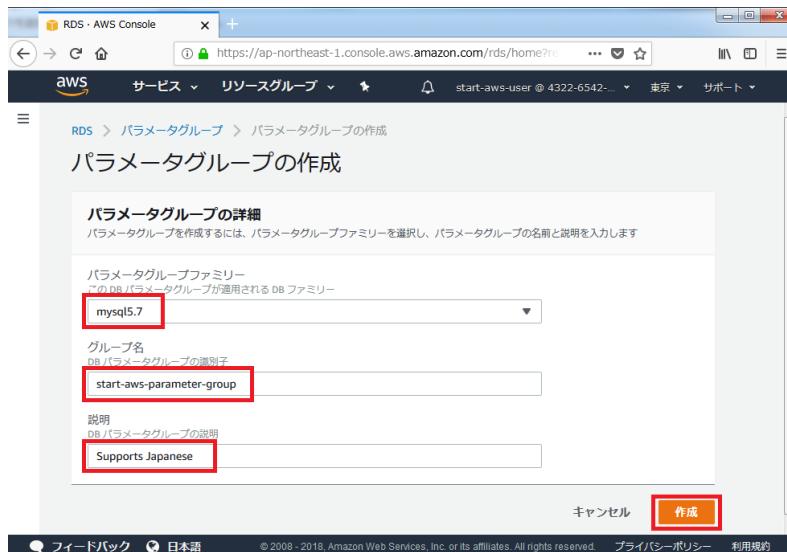
5.2.1 パラメータグループを作成

RDS でのインスタンス作成に先駆けて、パラメータグループを作成します。パラメータグループが何なのかについては追って説明しますので、とりあえず右上にある「パラメータグループの作成」をクリック（図 5.4）してください。



▲図 5.4 「パラメータグループの作成」をクリック

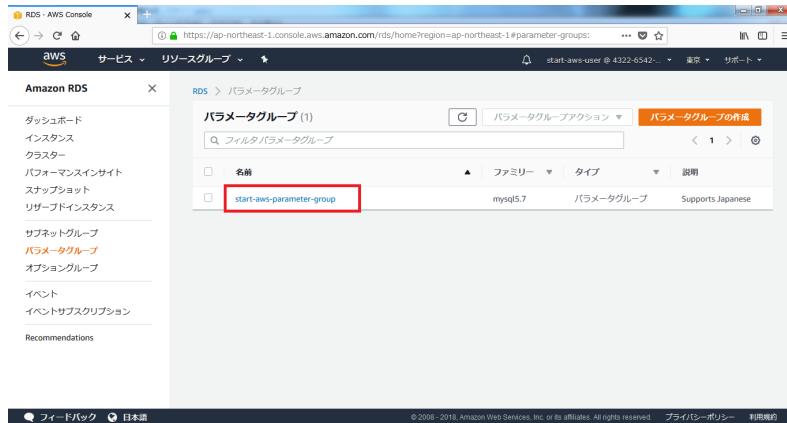
パラメータグループファミリーのプルダウンから「mysql5.7」を選択して、グループ名に「start-aws-parameter-group」、説明に「Supports Japanese」と入力したら「作成」をクリック（図 5.5）します。



▲図 5.5 「mysql5.7」を選択してグループ名と説明を入力したら「作成」をクリック

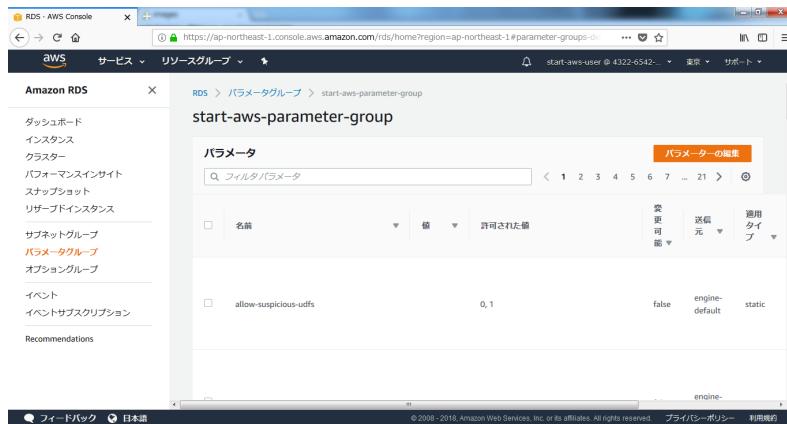
パラメータグループの一覧に、今作成した「start-aws-parameter-group」が表示され

ました。「start-aws-parameter-group」をクリック（図 5.6）してください。



▲図 5.6 「start-aws-parameter-group」をクリック

すると「パラメータ」がたくさん表示（図 5.7）されました。パラメータというのは MySQL におけるさまざまな設定値のことです。Apache に httpd.conf という設定ファイルがあったように、MySQL には my.cnf という設定ファイルがあるのですが、RDS では my.cnf を直接書き換える代わりにこのパラメータを編集します。



▲図 5.7 「パラメータ」がたくさん表示された

先ほど作ったパラメータグループというのはこのパラメータをまとめたグループのことなのですが、この後 RDS でインスタンスを作るときに「どのパラメータグループを使用

するか？」という選択が出てくるため、インスタンスを作る前にパラメータグループを作っておく必要があったのです。

5.2.2 パラメータの設定

ここからは全部で8個のパラメータの値を設定していきます。似たような名前がたくさん出てきますが間違えないようにしっかりついてきてください。

まずは「フィルタ パラメータ」と表示されているところに「character_set」と入力して検索（図5.8）します。検索結果が表示されたら右上にある「パラメータの編集」ボタンをクリックしてください。

| 名前 | 許可された値 | 変更履歴 | 送信元 | 高得点タイプ | データ型 | 説明 |
|--------------------------|--|------|----------------|---------|--------|--|
| character_set_client | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The character set for statements that arrive from the client. |
| character_set_connection | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The character set used for literals that do not have a character set introducer and for number-to-string conversion. |
| character_set_database | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The character set used by the default database. |
| character_set_filesystem | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The file system character set. |
| character_set_results | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The character set used for returning query results to the client. |
| character_set_server | big5, dbcs, cp949, hptk, ksc1, latin1, latin2, swed, asci, ucs, sjis, hebrew, iso8859_1, eucj, koi8r, gbk, gbkcs, gbk2312, gbk32, gbkcn, armcsl, utf8, cp936, keycode2, macos, macosx, cp932, latin7, utf8mb4, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, cp1259, cp932, eucjms | true | engine-default | dynamic | string | The server's default character set. |

▲図5.8 「character_set」でパラメータを検索

そして次の5つのパラメータで値を「utf8mb4」に変更（図5.9）します。

1. character_set_client
2. character_set_connection
3. character_set_database
4. character_set_results
5. character_set_server

画面上、上から4つめの「character_set_filesystem」は変更しませんので注意してください。

ださい。画面には 6 つのパラメータが表示されていますが、utf8mb4 に変更するのはそのうちの 5 つだけです。

The screenshot shows the AWS RDS Parameter Group editor for the 'start-aws-parameter-group' group. It lists six parameters under the 'character-set' category, all currently set to 'utf8mb4'. The parameters are: character_set_client, character_set_connection, character_set_database, character_set_results, and character_set_server. Each parameter has a dropdown menu next to it, and the 'utf8mb4' option is highlighted with a red box. The 'Change' button at the bottom right of the editor is also highlighted with a red box.

| 名前 | 値 | 説明された値 | 変更可能 | 送信元 | 選択タイプ | データ型 | 説明 |
|--------------------------|----------------|---|------|----------------|---------|--------|--|
| character_set_client | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, macos, utf8, cp1251, gbk, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, cp1256, cp1257, binary, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The character set for statements that arrive from the client. |
| character_set_connection | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, swed, hebrew, cp1251, greek, cp1250, gbk, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, utf8, cp1256, cp1257, utf8, cp1251, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The character set used for literals that do not have a character set introducer and for number-to-string conversion. |
| character_set_database | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, cp1256, cp1257, utf8, cp1251, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The character set used by the default database. |
| character_set_filesystem | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, utf8, cp1256, cp1257, utf8, cp1251, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The file system character set. |
| character_set_results | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, cp1256, cp1257, utf8, cp1251, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The character set used for returning query results to the client. |
| character_set_server | utf8mb4 | bigr, detch, cp1250, hdbl; hebrew, latin1, swed, asci, qps, qps, hebrew, iso88591, latin1, armSCII, utf8, cp1254, keybcs2, utf8mb4, cp1251, utf8, cp1256, cp1257, utf8, cp1251, gossi8, cp1251, euqgb1; | true | engine-default | dynamic | string | The server's default character set. |

▲図 5.9 5 つのパラメータで値を「utf8mb4」にして「変更の保存」をクリック

今変更した 5 つのパラメータはそれぞれ次のような設定項目です。

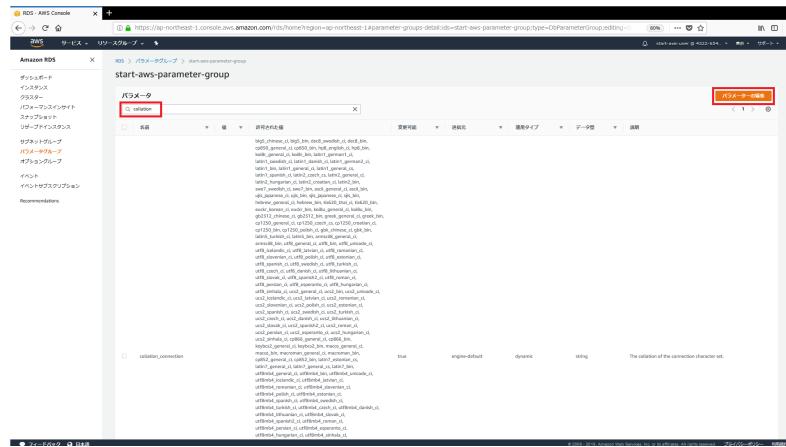
- character_set_client : クライアントから送られてくる SQL ステートメントの文字コード
- character_set_connection : クエリ実行時にリテラル及び数字から文字列への変換に使用される文字コード
- character_set_database : デフォルトデータベースで使用される文字コード
- character_set_results : クライアントへ返す実行結果やエラーメッセージで使用される文字コード
- character_set_server : サーバのデフォルト文字コード

難しそうなことが書いてありますが、要はこれらのパラメータを「utf8mb4」に設定しておくと日本語や絵文字が文字化けしなくなるという効能だけ覚えておいてください。逆に言えばこれらを設定しないと WordPress の記事で日本語や絵文字を使ったときに文字化けしてしまいます。それでは右上にある「変更の保存」をクリックしてください。

ちなみに次の 2 つは似たような名前なのですがこちらは何も設定しなくて大丈夫です。

- character_set_filesystem : ファイルシステムの文字コード。デフォルト設定のままでよいため設定不要。
- character_set_system : サーバで識別子を格納するために使用される文字コードで値は常に utf8。そもそもパラメータが存在しないため設定不要。

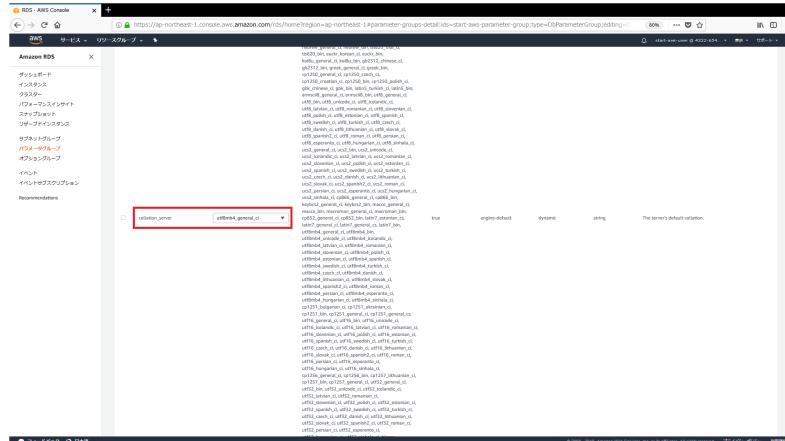
1~5つめの設定が済んだら今度は「collation」で検索（図 5.10）します。検索結果が表示されたら右上にある「パラメータの編集」ボタンをクリックしてください。



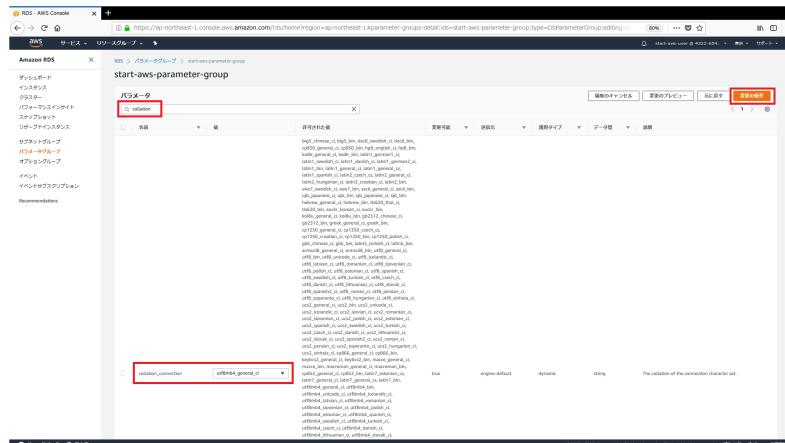
▲図 5.10 「collation」でパラメータを検索

そして次の 2 つのパラメータで値を「utf8mb4_general_ci」に変更して、右上にある「変更の保存」をクリックしてください。

1. collation_server (図 5.11)
2. collation_connection (図 5.12)



▲図 5.11 「collation_server」の値を「utf8mb4_general_ci」にする



▲図 5.12 「collation_connection」の値も「utf8mb4_general_ci」にして「変更の保存」をクリック

今変更した 2 つのパラメータはそれぞれ次のような設定項目です。

- collation_server : character_set_server に対する照合順序
- collation_connection : character_set_connection に対する照合順序

7 つめまでの設定が済んだら最後に「init_」で検索（図 5.13）します。検索結果が表

示されたら右上にある「パラメータの編集」ボタンをクリックしてください。

| 名前 | 値 | 許可された値 | 変更可能 | 送信元 | 適用タイプ | データ型 | 説明 | |
|--------------|---|--------|--------------------------|------|----------------|---------|--------|--|
| init_connect | | | <input type="checkbox"/> | true | engine-default | dynamic | string | String to be executed by the server for each client that connects. |

▲図 5.13 「init_」でパラメータを検索

そして次のパラメータで値に「SET NAMES utf8mb4;」と記入（図 5.14）したら、右上にある「変更の保存」をクリックしてください。

1. init_connect

| 名前 | 値 | 許可された値 | 変更可能 | 送信元 | 適用タイプ | データ型 | 説明 | |
|--------------|--------------------|--------|--------------------------|------|----------------|---------|--------|--|
| init_connect | SET NAMES utf8mb4; | | <input type="checkbox"/> | true | engine-default | dynamic | string | String to be executed by the server for each client that connects. |

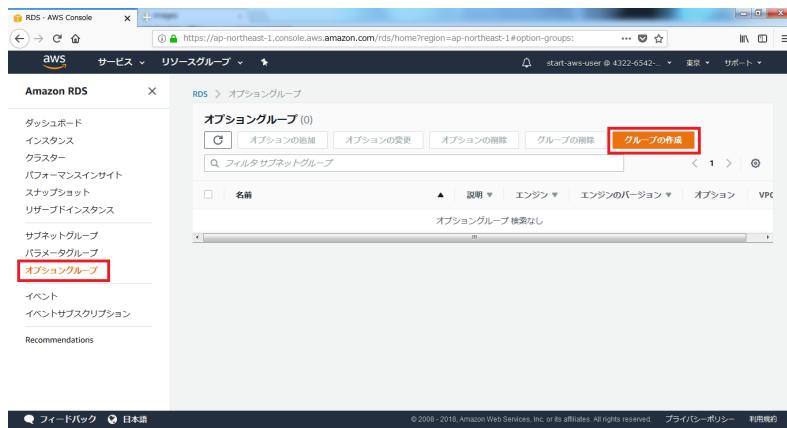
▲図 5.14 値に「SET NAMES utf8mb4;」と記入して「変更の保存」をクリック

今変更した「init_connect」という項目は「サーバから接続元の各クライアントに対して実行される文字列」です。

8個のパラメータの値を設定したのでこれでパラメータグループの作成は完了です。続いてオプショングループも作成します。

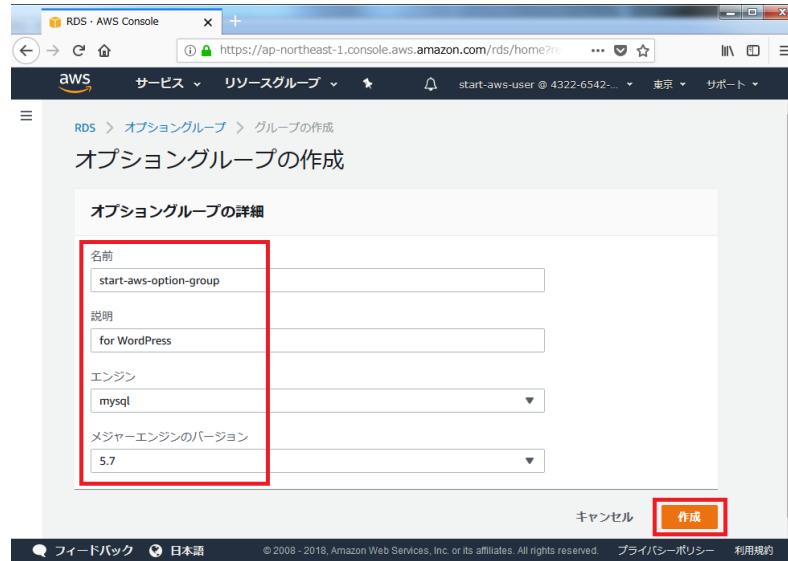
5.2.3 オプショングループを作成

左メニューの「オプショングループ」をクリック（図 5.15）してから、右上にある「グループの作成」をクリックします。



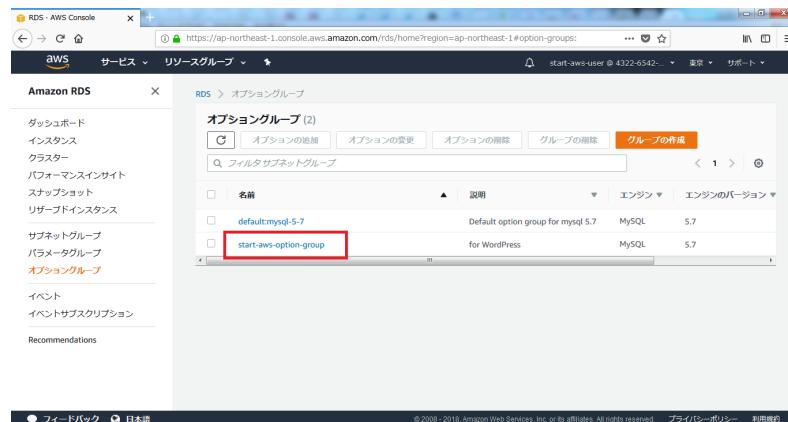
▲図 5.15 「オプショングループ」で「グループの作成」をクリック

名前に「start-aws-option-group」、説明に「for WordPress」と記入します。エンジンは「mysql」、メジャーエンジンのバージョンは「5.7」を選択したら「作成」をクリック（図 5.16）します。



▲図 5.16 オプショングループの詳細を設定したら「作成」をクリック

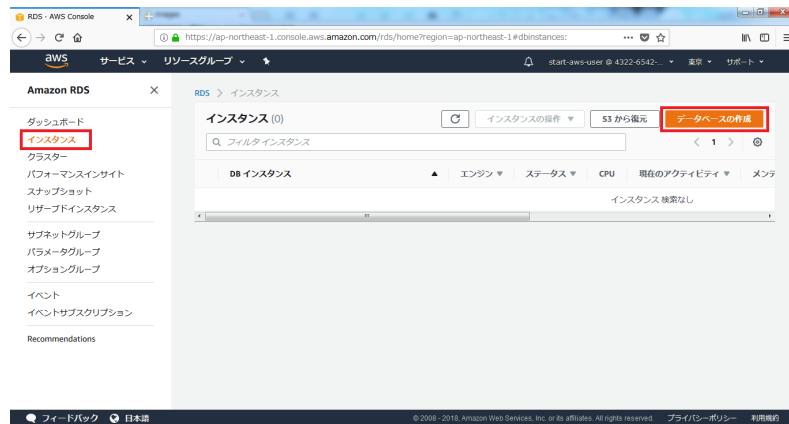
オプショングループの一覧に「start-aws-option-group」が表示（図 5.17）されたらオプショングループの作成は完了です。いよいよ RDS でインスタンスを作りましょう。



▲図 5.17 オプショングループの一覧に「start-aws-option-group」が表示された

5.3 RDS でインスタンスを立てよう

左メニューの「インスタンス」をクリック（図 5.18）してから、右上にある「データベースの作成」をクリックします。

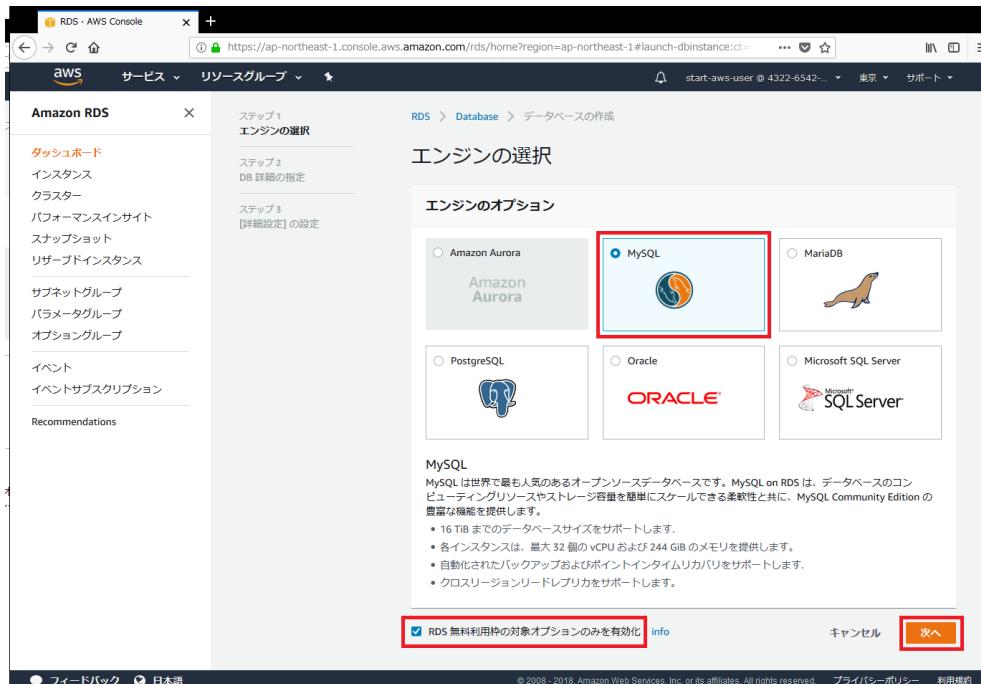


▲図 5.18 「インスタンス」で「データベースの作成」をクリック

ここから 3 つのステップで RDS のインスタンスを作成していきます。

5.3.1 ステップ 1: エンジンの選択

最下部にある「RDS 無料利用枠の対象オプションのみを有効化」にチェック（図 5.19）を入れて、MySQL を選択したら「次へ」をクリックします。



▲図 5.19 MySQL を選択したら「次へ」をクリック

5.3.2 ステップ 2:DB 詳細の指定

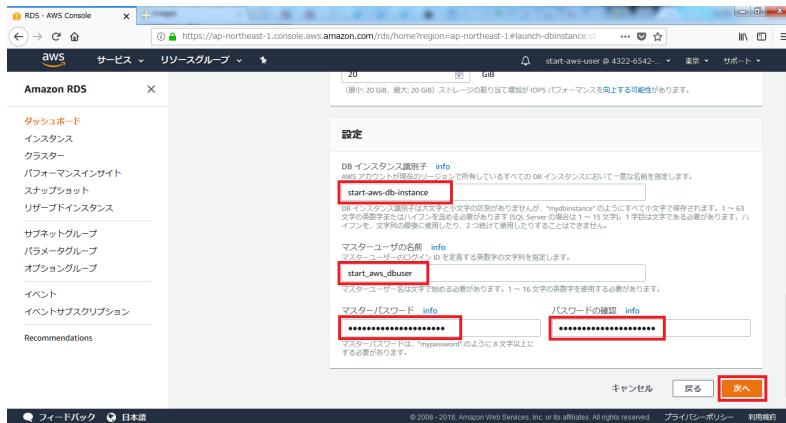
「インスタンスの仕様」は何も変更せずそのまま構いません。

下の方にスクロールしたら設定（図 5.20）を次のようにします。（表 5.1）

▼表 5.1 設定

| | |
|--------------|-----------------------|
| DB インスタンス識別子 | start-aws-db-instance |
| マスターユーザの名前 | start_aws_dbuser |
| マスターpassword | start_aws_db_password |
| passwordの確認 | マスターpasswordと同じ |

ここで設定した「マスターユーザの名前」と「マスターpassword」は、この後で WordPress がデータベースに接続するときに必要となります。入力したら「次へ」をクリックします。



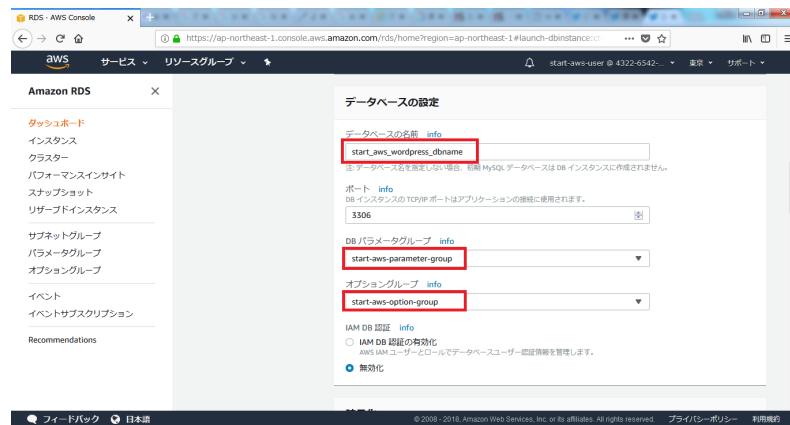
▲図 5.20 「設定」を入力したら「次へ」をクリック

5.3.3 ステップ 3:[詳細設定] の設定

一番上の「ネットワーク&セキュリティ」は何も変更せずそのまま構いません。その下の「データベースの設定」(図 5.21) は次のようにします。(表 5.2)

▼表 5.2 データベースの設定

| | |
|--------------|----------------------------|
| データベースの名前 | start_aws_wordpress_dbname |
| DB パラメータグループ | start-aws-parameter-group |
| オプショングループ | start-aws-option-group |

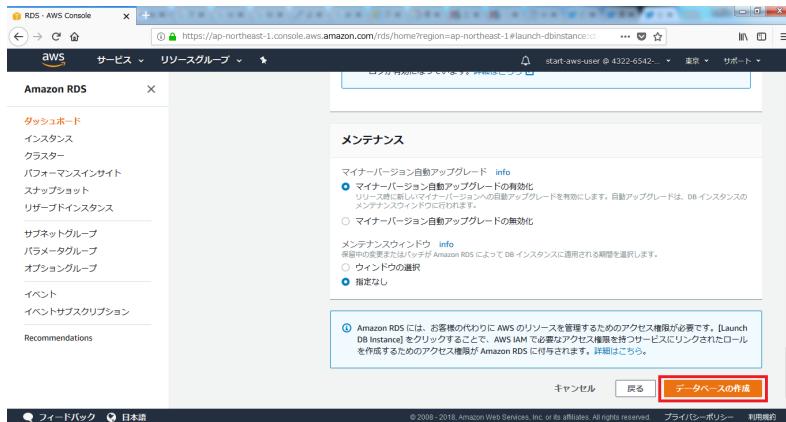


▲図 5.21 「データベースの設定」を入力

ここで設定した「データベースの名前」も、この後で WordPress がデータベースに接続するときに必要となります。「データベースの設定」のそれ以外の項目や、さらに下にある「暗号化」「バックアップ」「モニタリング」「ログのエクスポート」「メンテナンス」*3はすべてデフォルトのままで構いません。

「データベースの設定」を入力し終わったら「データベースの作成」をクリック（図 5.22）します。

*3 ちなみに「メンテナンス」で「マイナーバージョン自動アップグレード」を選択していると、新しいバージョンが出たときに MySQL を自動でアップグレードして自動で再起動してしまいます。今回は構いませんが、重要なウェブサイトやサービスで何の予告もなくデータベースに繋がらなくなったら大変ですのでもちらは基本的に無効化しておきます。

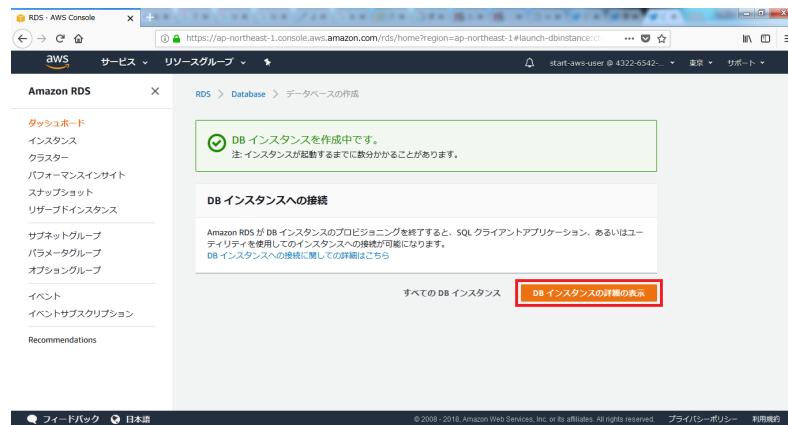


▲図 5.22 「データベースの作成」をクリック

5.3.4 セキュリティグループで 3306 番ポートの穴あけをしよう

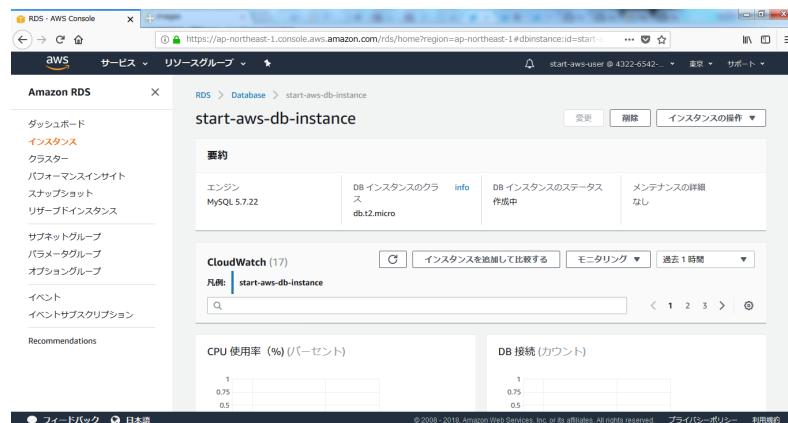
「DB インスタンスを作成中です。」と表示（図 5.23）されたら「DB インスタンスの詳細の表示」をクリックします。RDS のインスタンスができあがるまで少し時間がかかりますので、その間に「このデータベースにはどこからの接続を許可するのか？」というセキュリティグループ^{*4}の設定をしておきましょう。

^{*4} 何度か出てきていますが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 EC2 の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。



▲図 5.23 「DB インスタンスの詳細の表示」をクリック

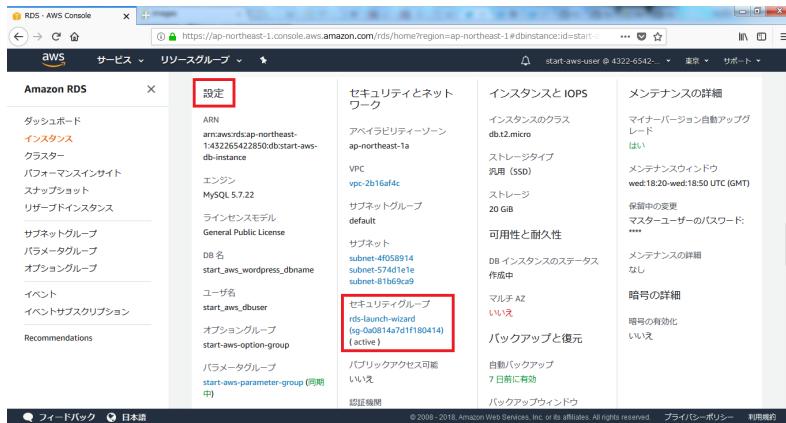
今作った「start-aws-db-instance」という RDS インスタンスの詳細が表示（図 5.24）されます。



▲図 5.24 「start-aws-db-instance」という RDS インスタンスの詳細

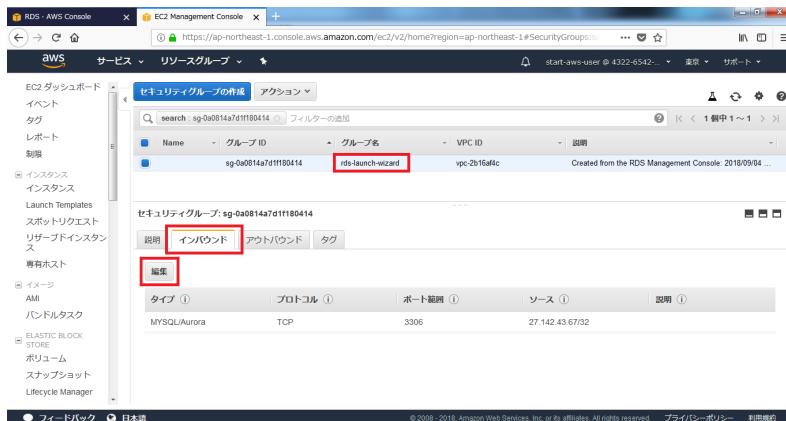
下の方にスクロールして「設定」（図 5.25）を見てください。RDS インスタンスの手前に立ちはだかっているのは「rds-launch-wizard」という名前のセキュリティグループです。「rds-launch-wizard」をクリックしてください。

5.3 RDS でインスタンスを立てよう



▲図 5.25 セキュリティグループの「rds-launch-wizard」をクリック

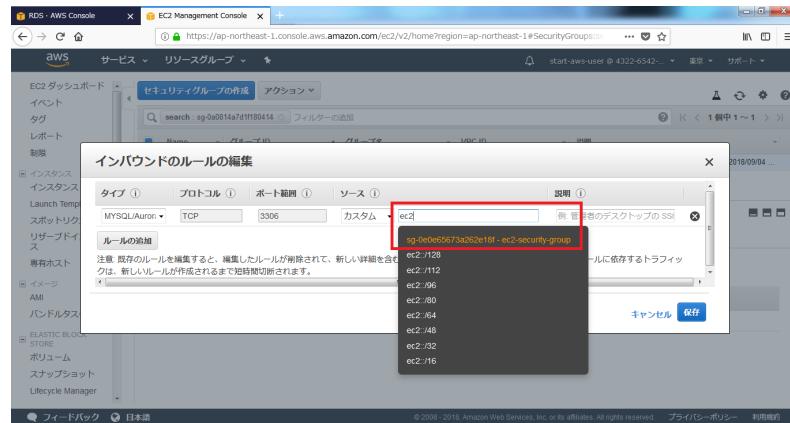
「rds-launch-wizard」の「インバウンド」タブで「編集」をクリック（図 5.26）します。



▲図 5.26 「インバウンド」タブで「編集」をクリック

データベースに記事データを保存したり、データベースへ記事データを取りに行ったりするのは EC2 のインスタンス上で動いている WordPress ですので、このセキュリティグループでは EC2 のインスタンスから RDS の「MySQL (ポート番号 3306 番)」へ通信できるように許可する設定をしてやらなければいけません。

ソースに書かれた IP アドレスを消して「ec2」と入力（図 5.27）すると下に「ec2-security-group」が表示されますのでクリックしてください。



▲図 5.27 「ec2」と入力して「ec2-security-group」が表示されたらクリック

ソースに「sg-0e0e65673a262e18f」のような EC2 のセキュリティグループ ID が表示（図 5.28）されたら「保存」をクリックします。

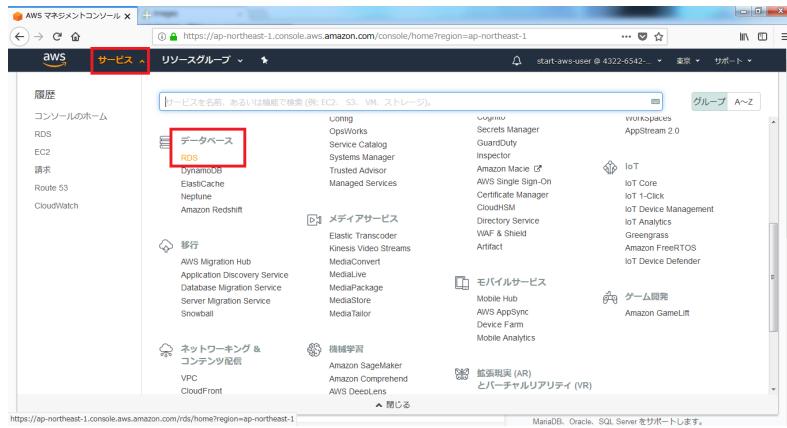


▲図 5.28 「保存」をクリック

これで EC2 のインスタンスから RDS のインスタンスの「MySQL（ポート番号 3306 番）」へ接続できるようになりました。

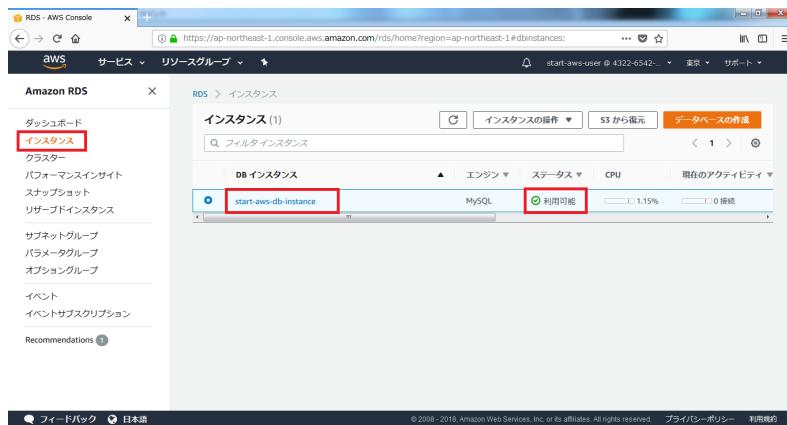
5.3.5 エンドポイントのドメイン名をメモしておこう

そろそろ RDS のインスタンスができあがっている頃です。マネジメントコンソールの左上にある「サービス」から、「データベース」の下にある「RDS」（図 5.29）をクリックしてください。



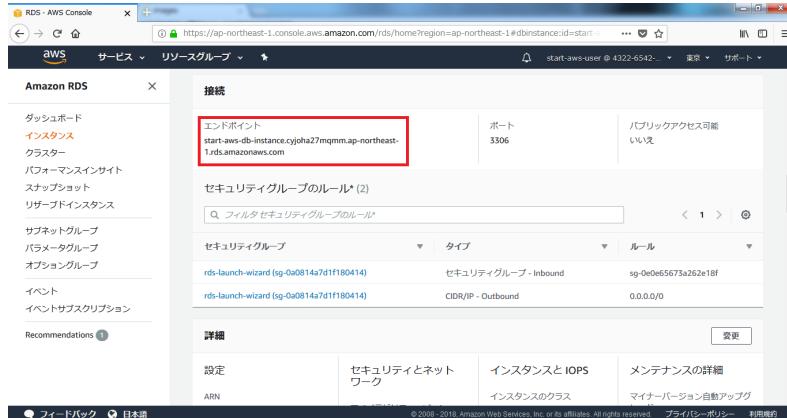
▲図 5.29 サービス>データベース>RDS

RDS ダッシュボードを開いたら左メニューの「インスタンス」をクリック（図 5.30）します。「start-aws-db-instance」というインスタンスのステータスが「利用可能」になっていたら「start-aws-db-instance」をクリックしてください。



▲図 5.30 左メニューの「インスタンス」>「start-aws-db-instance」をクリック

下の方にスクロールすると「接続」のところに「エンドポイント」(図 5.31) があります。この「start-aws-db-instance.cyjoha27mqmm.ap-northeast-1.rds.amazonaws.com」という「エンドポイント」は、後で WordPress からデータベースに接続するときに必要となります。長いのでパソコンのメモ帳にしっかりコピーペーストしておいてください。



▲図 5.31 「エンドポイント」をしっかりメモしておこう

5.3.6 ウェブサーバから接続確認してみよう

それでは EC2 のインスタンスから RDS の MySQL に接続できるか、mysql コマンドを使って試してみましょう。いわゆる「ウェブサーバからデータベースへの疎通確認」ですね。MySQL に接続するときは先ほどメモしておいた

- エンドポイント
- マスターユーザの名前
- マスターpassword

の 3 つを使用します。

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」と表示されたら次のコマンドを叩きます。

```
$ mysql -h エンドポイント -u マスターユーザの名前 -p
```

-h オプションは「接続先ホスト」を指定しています。-u オプションは「このユーザで」、-p オプションは「password認証で」という意味です。筆者だったらこんなコマンドになります。

```
$ mysql -h start-aws-db-instance.cyjoha27mqmm.ap-northeast-1.rds.amazonaws.com  
-u start_aws_dbuser -p  
※実際は改行せずに 1 行で実行
```

コマンドを叩くと次のように表示されます。

```
Enter password:
```

passwordを求められているので「マスターpassword」の「start_aws_db_password」を入力して Enter キーを押します。なおpasswordは入力しても画面上何の変化もありません。カーソルの位置も変わらないし「***」のような表示も一切されません。入力で

きている手ごたえがまったくありませんがちゃんと入力できていますし、間違えたときは BackSpace キーを押せば消せていますので大丈夫です。パスワードを入力したら Enter キーを押してください。

パスワードを入力後、次のように一番下に「mysql>」と表示されたら MySQL へのログインに成功しています。

```
$ mysql -h エンドポイント -u マスターユーザの名前 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 5.7.22-log Source distribution

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

試しに「show databases;」と入力して Enter キーを押すとデータベースの一覧が表示されます。ちゃんと「start_aws_wordpress_dbname」という名前のデータベースも作成されていることが確認できました。

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| mysql               |
| performance_schema |
| start_aws_wordpress_dbname |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

それでは MySQL を抜けて EC2 のインスタンスに戻りたいので「exit」と入力して Enter キーを押してください。

```
mysql> exit
Bye
```

Bye と表示されたら MySQL との接続を切って EC2 のインスタンスへ戻ってこられます。

これでデータベースサーバは用意できました。いよいよ WordPress のインストールを行いましょう。

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。大変な道のりでしたががいよいよおしゃれなサイトがお目見えです！

6.1 WordPressのインストール

これからEC2のインスタンスでWordPressのインストールを行います。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」と表示されたらログイン完了です。ここからの作業はrootにならずにec2-userのままで行ってください。

6.1.1 WordPressをダウンロード

まずはcurlコマンドを使って最新のWordPress(<https://wordpress.org/latest.tar.gz>)をダウンロードしましょう。curlコマンドは何もオプションをつければ結果をそのまま画面に出力しますが、-Oオプションを付けるとファイルとして保存してくれます。

```
$ curl -O https://wordpress.org/latest.tar.gz
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total   Spent    Left Speed
100 8538k  100 8538k    0     0  3342k      0  0:00:02  0:00:02  --:--:-- 3342k
```

ダウンロードしたファイルをlsコマンドで確認してみましょう。

```
$ ls -lh /home/ec2-user/
合計 8.4M
-rw-rw-r-- 1 ec2-user ec2-user 8.4M 9月  5 21:08 latest.tar.gz
```

圧縮された最新版のWordPress(latest.tar.gz)をダウンロードできました。

6.1.2 展開してドキュメントルートに設置

実はWordPressのインストールと言っても、今ダウンロードした圧縮ファイルを展開してドキュメントルートに置き、後はブラウザでぼちぼちと進めていくだけなので簡単です。

それでは圧縮・解凍ソフトのような tar コマンドを使って WordPress の圧縮ファイル (latest.tar.gz) を展開します。ls コマンドで確認すると、展開によって wordpress というディレクトリができていることがわかります。

```
$ tar -xzf /home/ec2-user/latest.tar.gz  
$ ls -lh /home/ec2-user/  
合計 8.4M  
-rw-rw-r-- 1 ec2-user ec2-user 8.4M 9月 5 21:08 latest.tar.gz  
drwxr-xr-x 5 ec2-user ec2-user 4.0K 8月 3 05:39 wordpress
```

展開できたら mv コマンド^{*1}を使って wordpress ディレクトリの中身をすべてドキュメントルートに移動させましょう。特にいちばん後ろのスラッシュは書き忘れないように注意してください。

```
$ sudo mv /home/ec2-user/wordpress/* /var/www/start-aws-documentroot/
```

今後、WordPress の管理画面から画像ファイルをアップしたりプラグイン^{*2}をインストールしたりするためには、ドキュメントルート以下のファイルやディレクトリに対して apache ユーザが権限を持っていないといけません。chown コマンド^{*3}でドキュメントルート以下のオーナー（持ち主）を apache ユーザに変更しておきましょう。

```
$ sudo chown apache:apache -R /var/www/start-aws-documentroot  
$ ls -ld /var/www/start-aws-documentroot  
drwxr-xr-x 6 apache apache 4096 9月 5 22:04 /var/www/start-aws-documentroot
```

それから第4章「ウェブサーバの設定をしよう」で動作確認用に作った「AWS をはじめよう」と書いてある index.html ファイルは削除しておきましょう。

```
$ sudo rm -f /var/www/start-aws-documentroot/index.html
```

これでドキュメントルート以下にあるのは展開した WordPress のファイルだけになりました。

^{*1} mv は move の略。ファイルを移動させたりファイル名を変更したりできるコマンドです。

^{*2} WordPress の拡張機能のこと。

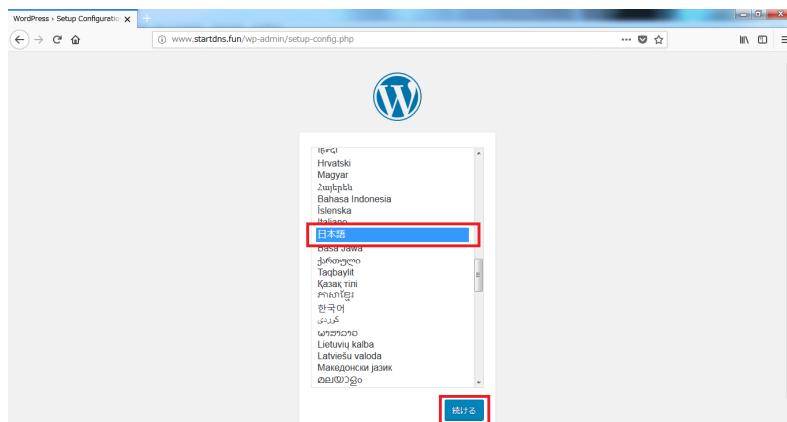
^{*3} chown は change ownership の略。

```
$ ls -lh /var/www/start-aws-documentroot/
合計 196K
-rw-r--r-- 1 apache apache 418 9月 25 2013 index.php
-rw-r--r-- 1 apache apache 20K 1月 7 2018 license.txt
-rw-r--r-- 1 apache apache 7.3K 3月 19 01:13 readme.html
drwxr-xr-x 5 apache apache 4.0K 8月 3 05:39 wordpress
-rw-r--r-- 1 apache apache 5.4K 5月 2 07:10 wp-activate.php
drwxr-xr-x 9 apache apache 4.0K 8月 3 05:39 wp-admin
-rw-r--r-- 1 apache apache 364 12月 19 2015 wp-blog-header.php
-rw-r--r-- 1 apache apache 1.9K 5月 3 07:11 wp-comments-post.php
-rw-r--r-- 1 apache apache 2.8K 12月 16 2015 wp-config-sample.php
drwxr-xr-x 4 apache apache 4.0K 8月 3 05:39 wp-content
-rw-r--r-- 1 apache apache 3.6K 8月 20 2017 wp-cron.php
drwxr-xr-x 18 apache apache 12K 8月 3 05:39 wp-includes
-rw-r--r-- 1 apache apache 2.4K 11月 21 2016 wp-links-opml.php
-rw-r--r-- 1 apache apache 3.3K 8月 22 2017 wp-load.php
-rw-r--r-- 1 apache apache 37K 7月 16 23:14 wp-login.php
-rw-r--r-- 1 apache apache 7.9K 1月 11 2017 wp-mail.php
-rw-r--r-- 1 apache apache 16K 10月 4 2017 wp-settings.php
-rw-r--r-- 1 apache apache 30K 4月 30 08:10 wp-signup.php
-rw-r--r-- 1 apache apache 4.6K 10月 24 2017 wp-trackback.php
-rw-r--r-- 1 apache apache 3.0K 9月 1 2016 xmlrpc.php
```

6.1.3 サイトにアクセスしてインストール実行

それではブラウザで `http://www.自分のドメイン/` を開いてください。筆者なら自分のドメインは「startdns.fun」なので `http://www.startdns.fun/` を開きます。

すると WordPress の言語選択画面（図 6.1）が表示されるので「日本語」を選択して「続ける」をクリックします。



▲図 6.1 「日本語」を選択して「続ける」をクリック

続いて「WordPress へようこそ。」という画面（図 6.2）が表示されたら「さあ、始めましょう！」をクリックします。



▲図 6.2 「さあ、始めましょう！」をクリック

データベースの接続情報を入力する画面（図 6.3）が表示されたら、第 5 章「データベースサーバを立てよう」で設定した内容を思い出しながら次のように入力します。（表 6.1）テーブル接頭辞は変更せずそのままで構いません。

▼表 6.1 データベースの接続情報

| | |
|-------------|--|
| データベース名 | start_aws_wordpress_dbname (データベースの名前) |
| ユーザー名 | start_aws_dbuser (マスターユーザの名前) |
| パスワード | start_aws_db_password (マスターpassword) |
| データベースのホスト名 | エンドポイント |

エンドポイントは「start-aws-db-instance.cyjoha27mqmm.ap-northeast-1.rds.amazonaws.com」のような長いドメイン名です。パソコンのメモ帳にメモしてあると思いますので、そこからコピペーストしましょう。すべて入力したら「送信」をクリックします。



▲図 6.3 データベースの接続情報を入力して「送信」をクリック

データベースの接続情報に誤りがなく、WordPress がデータベースと繋がったら「この部分のインストールは無事完了しました。WordPress は現在データベースと通信できる状態にあります。」と表示（図 6.4）されます。「インストール実行」をクリックしましょう。



▲図 6.4 「インストール実行」をクリック

「WordPress の有名な 5 分間インストールプロセスへようこそ！」と表示（図 6.5）されたら、サイトのタイトルなどを入力します。サイトのタイトルは好きなものにしてください。ユーザー名やパスワードは WordPress の管理画面にログインするときに使用しますのであなた自身で決めてメモ（表 6.2）しておいてください。「検索エンジンでの表示」は

どちらでも構いませんが、作ったサイトを Google などの検索結果に表示したくなければチェックを入れておきましょう。もしパスワードを忘れてしまった場合、ここに書いたメールアドレス宛てにパスワードリセットのメールが届きますのでメールアドレスは正確に入力してください。すべて入力したら「WordPress をインストール」をクリックします。

▼表 6.2 WordPress 管理画面に入るための情報をメモ

| | |
|---------|--|
| ユーザー名 | |
| パスワード | |
| メールアドレス | |



▲図 6.5 「WordPress をインストール」をクリック

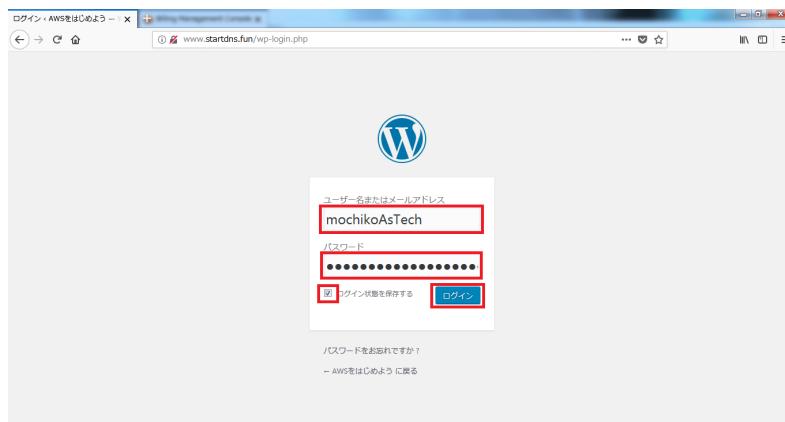
「成功しました!」と表示（図 6.6）されたら WordPress のインストールは完了です。「ログイン」をクリックして WordPress の管理画面に入ってみましょう。



▲図 6.6 「ログイン」をクリックして WordPress の管理画面に入ってみよう

6.2 管理画面にログイン

WordPress の管理画面に入るため、先ほど設定したユーザー名とパスワードを入力（図 6.7）したら「ログイン状態を保存する」にチェックを入れて「ログイン」をクリックします。



▲図 6.7 ユーザー名とパスワードを入力して「ログイン」をクリック

もし左メニューで「更新」に赤いマーク（図 6.8）がついていたら、クリックして WordPress 本体やプラグインの更新をしておきましょう。

6.2 管理画面にログイン



▲図 6.8 「更新」に赤いマークがついていたら「更新」をクリック

「WordPress の新しいバージョンがあります。」と表示されていますので「今すぐ更新」をクリック（図 6.9）します。^{*4}

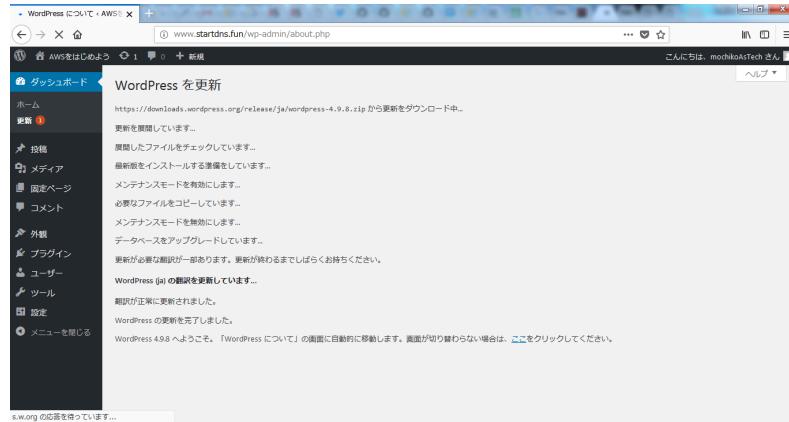


▲図 6.9 「今すぐ更新」をクリック

「今すぐ更新」をクリックすると更新の進捗が表示（図 6.10）され、更新完了すると新

*4 WordPress では定期的に脆弱性が発見されて対策済みのバージョンが公開されます。脆弱性（ぜいじやくせい）というのは悪用が可能なバグや設定不備のことです。もし WordPress に脆弱性が見つかって対策済みの新しいバージョンの WordPress が公開されたとしても、あなたが更新をしないとウェブサイトは脆弱性がある危険な状態のままで放置されていることになります。古いバージョンの WordPress を使っているとサイトを改ざんしてウイルスをばらまかれたりデータベースの個人情報を盗まれたりする可能性もありますので、新しいバージョンが出たらきちんと更新しましょう。

しいバージョンではどんな修正がされたのか？が表示されます。それではできあがったおしゃれなサイトを見てみましょう。左上の家のマークをクリック（図6.11）してください。



▲図6.10 更新の進捗が表示される



▲図6.11 更新完了したら家のマークをクリック

そしてこちらが完成したWordPressのおしゃれなサイトです！（図6.12）ここからは見た目をカスタマイズするもよし、技術的なことを調べたブログ記事を日々綴っていくもよし、自分のサイトを自由に楽しんでください。



▲図 6.12 できあがった「自分のサイト」をここからは自由にカスタマイズしよう

6.2.1 【ドリル】WordPress からのメールが迷惑メール扱いされてしまう

問題

「AWS をはじめよう」を読みながら構築した環境で、WordPress の管理画面に入るパスワードを忘れてしまったためパスワードのリセットを行いました。ところがいくら待ってもメールが届きません。確認したところパスワードリセットのメール（図 6.13）は迷惑メールのトレイに入っていました。



▲図 6.13 パスワードリセットのメールが迷惑メール扱いされていた

原因は「SPF が設定されていないこと」のようです。送信元のメールアドレスは

「WordPress <wordpress@startdns.fun>」でした。このとき次のどれを設定すれば迷惑メール扱いされなくなるでしょうか？

- A. startdns.fun の TXT レコードに 「v=spf1 ip4:xxx.xxx.xxx.xxx ~all」 のように Elastic IP のアドレスを設定する
- B. www.startdns.fun の TXT レコードに 「v=spf1 ip4:xxx.xxx.xxx.xxx ~all」 のように Elastic IP のアドレスを設定する
- C. startdns.fun の TXT レコードに 「v=spf1 a:login.startdns.fun ~all」 のように SSH ログインで使っているドメイン名を設定する

答え _____

解答

正解は A または C です。SPF レコードを設定すべきなのは送信元のメールアドレスで使われているドメイン名です。EC2 のインスタンスからメールが送信されるとき送信元の IP アドレスは Elastic IP となります。A のように直接 Elastic IP の IP アドレスを指定してもいいですし、C のように Elastic IP と紐づいているドメイン名を指定しても構いません。

C のようにしておくと Elastic IP が変更になったときに「login.startdns.fun」の A レコードだけを変更すればいいいため変更箇所が少なくて済みます。

The screenshot shows the AWS Route 53 Management Console interface. The left sidebar is collapsed. The main area displays a table of record sets for the domain 'startdns.fun'. One of the records is highlighted with a red box:

| Name | Type | Value |
|---------------------|------------|--|
| startdns.fun | NS | ns-943.awsdns-53.net ns-1605.awsdns-08.co.uk ns-1072.awsdns-08.org ns-177.awsdns-22.com |
| startdns.fun | SOA | ns-943.awsdns-53.net awrdns-hostmaster.amazon. |
| startdns.fun | TXT | "v=spf1 a:login.startdns.fun ~all" |
| login.startdns.fun | A | 52.199.234.148 |
| www.startdns.fun | A | Alias: dualstack.start.aws-ebs-1653955052.ap-north- |

The 'Test Record Set' button is visible above the table. A tooltip message is displayed: 'To get started, click Create Record Set button or click on existing record set.'

▲図 6.14 筆者は C の方法で SPF レコードを設定しました

6.2.2 管理画面にダイジェスト認証をかけよう

ところで WordPress は利用者が多いため、その管理画面を乗っ取ろうと狙ってくる攻撃も多いです。「こんな作ったばかりの小さなブログに攻撃なんか来ないので？」と思われるかもしれません、攻撃者は IP アドレスを端から順番に試していくだけなので、サイトの開設時期や規模にかかわらずどんなサーバでも攻撃はされると思って間違いありません。

管理画面にログインするにはユーザー名とパスワードの認証が必要ですが、安全のためその手前にもうひとつ「ダイジェスト認証」という認証をかけておきましょう。

EC2 のインスタンスでコマンドを叩きますので RLogin やターミナルに戻って root になってください。

```
$ sudo su -
```

root になったら先ずは htdigest コマンドを使ってダイジェスト認証のパスワードファイルを作成します。

```
# htdigest -c /etc/httpd/conf/htdigest wp-admin-area start-aws-digest-user
```

「New password:」と表示されたら「start-aws-digest-password」と入力して Enter キーを押してください。再度「Re-type new password:」とパスワードを求められますので、もう一度「start-aws-digest-password」と入力して Enter キーを押します。

```
New password:  
Re-type new password:
```

これでダイジェスト認証のパスワードファイルが生成できました。cat コマンドでパスワードファイルを見てみましょう。

```
# cat /etc/httpd/conf/htdigest  
start-aws-digest-user:wp-admin-area:10b74c85e2a0273d00eee83755e329b8
```

続いて vi コマンドでバーチャルホストにダイジェスト認証の設定を書き足します。

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

i (アイ) を押して「編集モード」になつたら、次のようにダイジェスト認証の設定をVirtualHost ディレクティブの中に追記します。

```
<Directory "/var/www/start-aws-documentroot/wp-admin">
    AuthType Digest
    AuthName wp-admin-area
    AuthUserFile /etc/httpd/conf/htdigest
    Require valid-user
</Directory>
```

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 キーを押せば保存されます。保存できたら確認のため cat コマンドを叩いてダイジェスト認証の設定が追加されているか確認してください。

```
# 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>

    <Directory "/var/www/start-aws-documentroot/wp-admin">
        AuthType Digest
        AuthName wp-admin-area
        AuthUserFile /etc/httpd/conf/htdigest
        Require valid-user
    </Directory>
</VirtualHost>
```

apachectl で構文チェックをしてみましょう。「Syntax OK」とだけ表示されれば問題ありません。

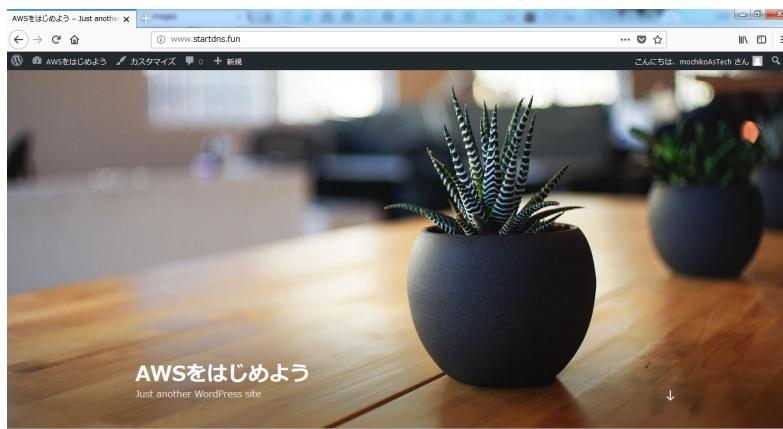
```
# apachectl configtest
Syntax OK
```

それではダイジェスト認証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
```

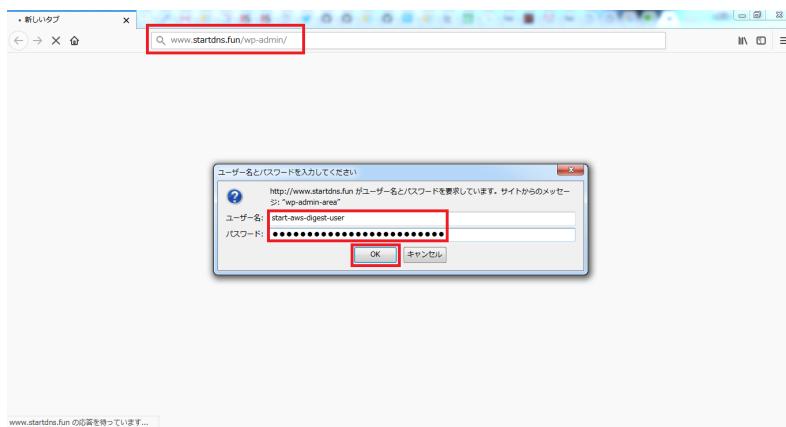
何のメッセージも表示されなければ問題なく Apache が再起動できています。それではブラウザで確認してみましょう。

先ずは管理画面ではなくブログの方を表示してみましょう。ブラウザで `http://www.自分のドメイン/` を開いてください。こちらは先ほどまでと代わらず普通に表示（図 6.15）されるはずです。



▲図 6.15 ブログは認証なしで普通に表示される

続いてブラウザで WordPress の管理画面 (`http://www.自分のドメイン/wp-admin/`) を開いてみましょう。すると「ユーザー名とパスワードを入力してください」というダイジェスト認証のポップアップが表示（図 6.16）されますので、先ほど設定したダイジェスト認証のユーザー名とパスワードを入力して「OK」をクリックしてください。（表 6.3）

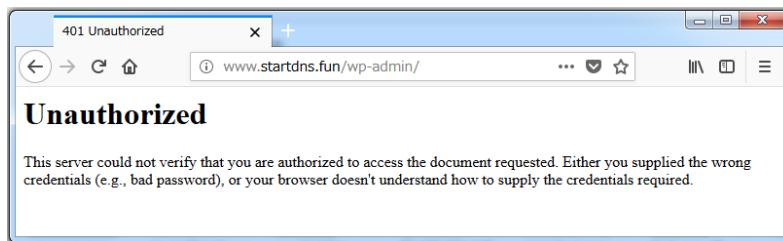


▲図 6.16 管理画面では「ユーザー名とパスワードを入力してください」というダイジェスト認証のポップアップが表示された

▼表 6.3 ダイジェスト認証の情報

| | |
|-------|---------------------------|
| ユーザー名 | start-aws-digest-user |
| パスワード | start-aws-digest-password |

正しいユーザー名とパスワードを入れれば管理画面に入るためのログイン画面（もしくは管理画面）が表示（図 6.18）されますが、間違ったものを入力すると「Unauthorized」と表示（図 6.17）されて管理画面には入れません。これで管理画面は「ダイジェスト認証」と「WordPress の認証」の2段階で守られるようになりました。



▲図 6.17 正しいユーザー名とパスワードを入れないと「Unauthorized」と表示されて管理画面に入れない



▲図 6.18 管理画面が「ダイジェスト認証」と「WordPress の認証」の 2 段階で守られるようになった

6.3 画像を S3 に保存する

ところで WordPress で画像をアップすると、画像ファイルはドキュメントルート以下にある「wp-content/uploads」というディレクトリに保存されます。

```
/var/www/start-aws-documentroot/wp-content/uploads/
```

ですが画像を EC2 インスタンス内に直接保存することによって、次のようなデメリットが発生します。

1. アクセス数が増えてきて負荷分散のためウェブサーバの台数を増やしたときに画像が片方のサーバにしか保存されず、もう 1 台のサーバでは記事内の画像が表示されなくなってしまう
2. ウェブサーバが壊れて AMI^{*5}から復元したとき、画像のフォルダが AMI を作った時点まで先祖返りしてしまい、記事内の画像が表示されなくなってしまう

そこで台数を増やしたりサーバを復元したりしたときでも記事内の画像が問題なく表示されるよう、WordPress でアップした画像はインスタンス内に保存するのではなく Amazon S3 に保存するようにしておきましょう。

^{*5} AMI については第 7 章「サーバのバックアップを取りっておこう」で解説します。

6.3.1 Amazon S3 とは？

Amazon S3 というのは「Amazon Simple Storage Service」の略で、頭文字の S が 3 つなので S3 です。シンプルストレージという名前とおり、S3 は Dropbox や Google クラウドのようにファイルを保存しておけるサービスです。S3 は 99.999999999%^{*6} の耐久性をもち、格納できるデータの容量も無制限ですので「ハードディスクが壊れて保存してた画像が全部消えちゃった・・・」「容量がいっぱいになっちゃってこれ以上保存できない！」といったトラブルとも無縁です。

S3 に保存した画像ファイルにはすべて URL が付与され、HTTPS でどこからでもアクセスができます。

6.3.2 S3 アップ用の IAM ユーザーを作ろう

皆さんがあなたのマネジメントコンソールにログインするときに使っている「start-aws-user」という IAM ユーザーは「AdministratorAccess」という一番強い権限を持っていました。この IAM ユーザーをそのまま WordPress に使わせるのは危険ですので「IAM ユーザーには必要最小限の権限だけを付与すべき」という原則にしたがって、S3 にファイルを保存するためだけの IAM ユーザーを作成しましょう。

WordPress でアップしたファイルを S3 に保存するため、まずは IAM のアクセスキーを発行します。細かな手順は省略^{*7}しますが、マネジメントコンソールで IAM ダッシュボードを開いて次のような流れで S3 アップ専用の IAM ユーザーを作成してください。

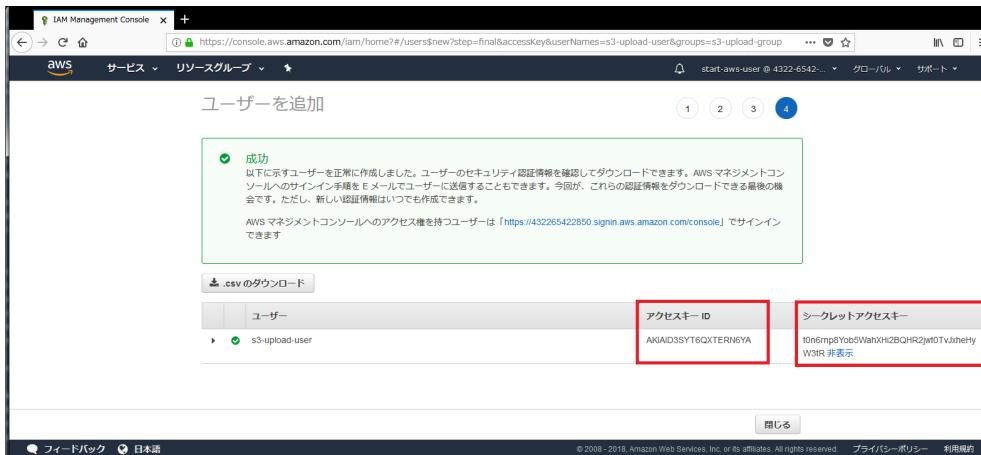
1. s3-upload-group という IAM グループを作る
2. s3-upload-group に AmazonS3FullAccess というポリシーをアタッチする
3. s3-upload-user という IAM ユーザーを作り、「アクセスの種類」を「プログラムによるアクセス」にする
4. s3-upload-user を s3-upload-group に追加する

S3 アップ専用の IAM ユーザーが作成（図 6.19）できたら、アクセスキー ID とシークレットアクセスキーをパソコンのメモ帳にメモ^{*8}しておいてください。

^{*6} イレブンナインと読みます。格好いい・・・！

^{*7} 第2章「AWSを使い始めたら最初にやること」で IAM グループと IAM ユーザーを作ってポリシーをアタッチしたときとほぼ同じ手順ですのでそちらを参照してください。大きく異なるのは IAM ユーザーの「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」ではなく「プログラムによるアクセス」になっている点です。マネジメントコンソールにログインして操作するのではなく、WordPress のプログラムから S3 を利用したいので「プログラムによるアクセス」を選択しています。

^{*8} シークレットアクセスキーが表示されるのはこの1回きりで以降は絶対に表示されません。もしシーク



▲図 6.19 「アクセスキー ID」と「シークレットアクセスキー」をメモしておこう

この 2 つはとても大切なものです。他の人に知られると S3 へ無断でファイルをアップされたり、S3 にアップしたファイルを消されたりしてしまいますので、アクセスキー ID とシークレットアクセスキーは外部へ絶対に漏らさない^{*9}ようにしてください。ソースコードに直接書いてそれを GitHub の公開リポジトリへプッシュしてしまう、というのがありがちな失敗です。

S3 アップ専用の IAM ユーザが作成できたら、WordPress プラグインのインストールに進みましょう。

6.3.3 WordPress に S3 を使うためのプラグインを入れよう

WordPress のプラグイン（拡張機能）を使って、記事の画像を S3 に保存する設定を行います。使用するプラグインは次の 2 つです。

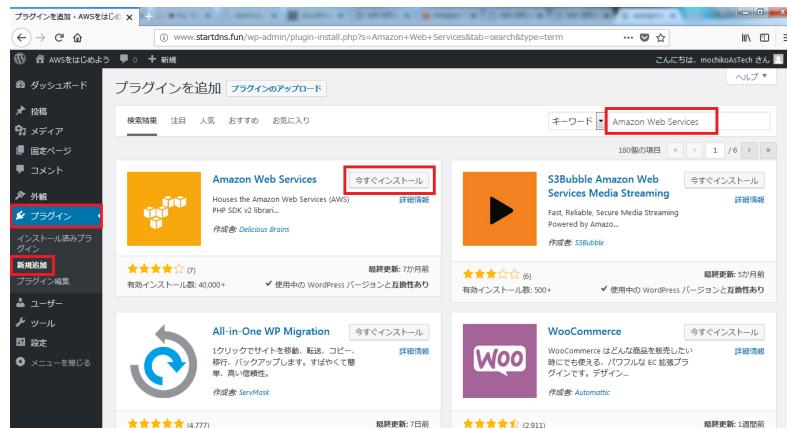
1. Amazon Web Services プラグイン
2. WP Offload S3 Lite プラグイン

レットアクセスキーをメモし忘れた場合は、いったんアクセスキーを削除して作成しなおしてください。IAM ダッシュボードから左メニューの「ユーザー」を開いて、「s3-upload-user」をクリックすると「認証情報」タブの中でアクセスキーの削除や作成ができます。

^{*9} 筆者は堂々と本著に載せていますが、勿論このアクセスキーはすでに無効にしてありますのでご安心ください。

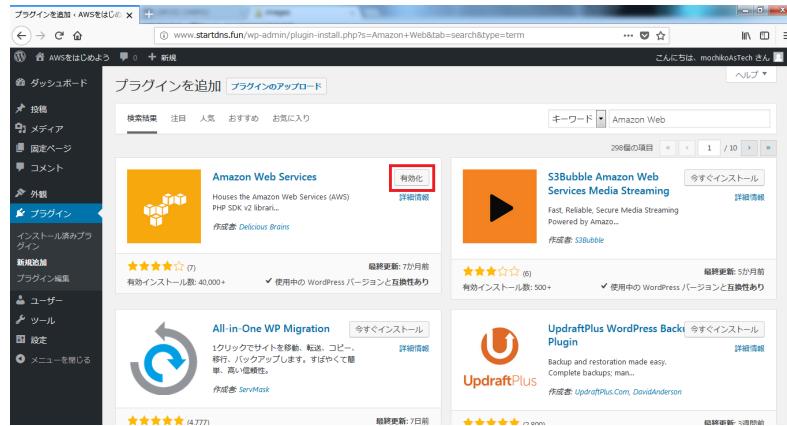
Amazon Web Services プラグイン

「プラグイン」の「新規追加」をクリック（図6.20）して、「プラグインの検索...」と書かれたところに「Amazon Web Services」と入力します。「Amazon Web Services」が表示されたら「今すぐインストール」をクリックします。



▲図6.20 「Amazon Web Services」を検索して「今すぐインストール」をクリック

ボタンの表示が「インストール中」の後に「有効化」になったらクリック（図6.21）します。



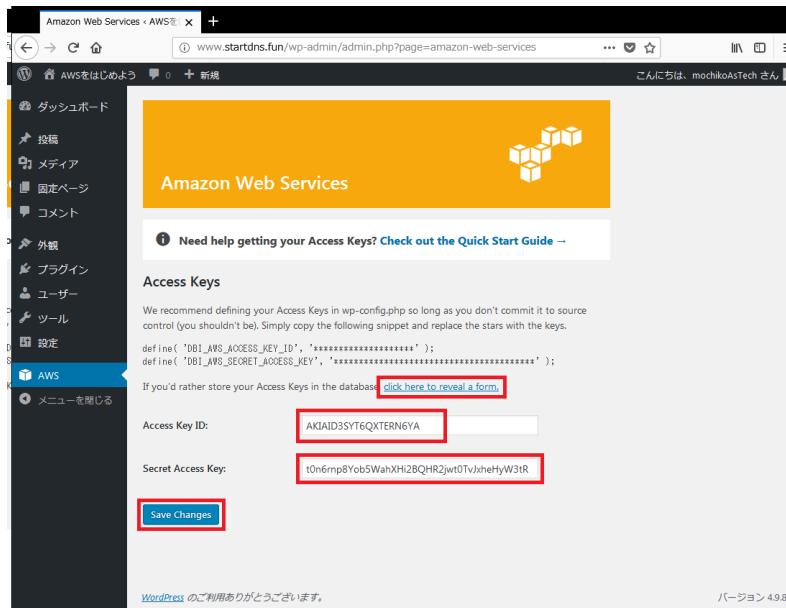
▲図6.21 「有効化」になったらクリック

Amazon Web Services プラグインがインストールできたので「Access Keys」をクリック（図 6.22）します。



▲図 6.22 Amazon Web Services プラグインの「Access Keys」をクリック

「click here to reveal a form.」をクリックして、先ほど作った S3 アップ専用の IAM ユーザである「s3-upload-user」のアクセスキー ID とシークレットアクセスキーを入力します。どちらも入力できたら「Save Changes」をクリックします。



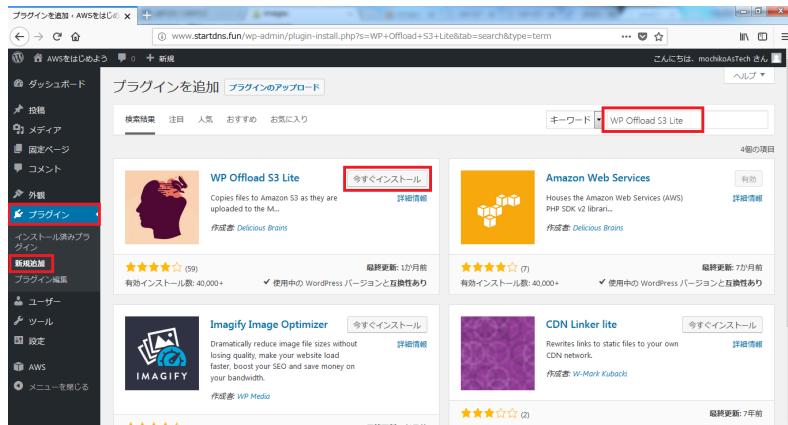
▲図 6.23 アクセスキー ID とシークレットアクセスキーを入力したら「Save Changes」をクリック

「Settings saved.」と表示されたら Amazon Web Services プラグインの設定は完了です。

WP Offload S3 Lite プラグイン

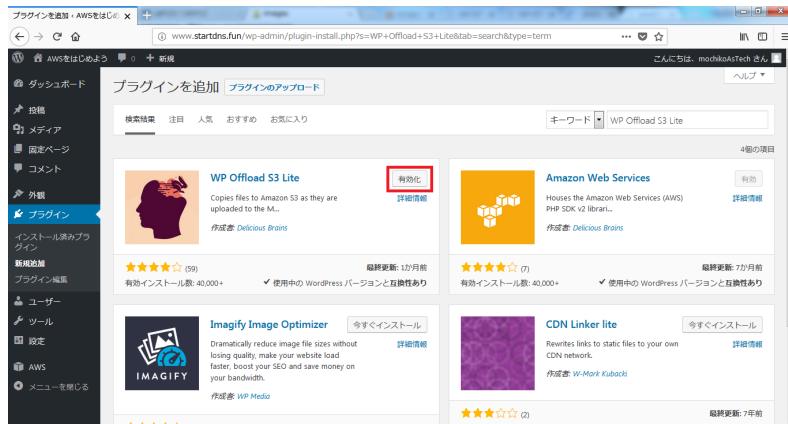
「プラグイン」の「新規追加」をクリック（図 6.24）して、「プラグインの検索...」と書かれたところに「WP Offload S3 Lite」と入力します。「WP Offload S3 Lite」が表示されたら「今すぐインストール」をクリックします。

6.3 画像を S3 に保存する



▲図 6.24 「WP Offload S3 Lite」を検索して「今すぐインストール」をクリック

ボタンの表示が「インストール中」の後に「有効化」になったらクリック（図 6.25）します。



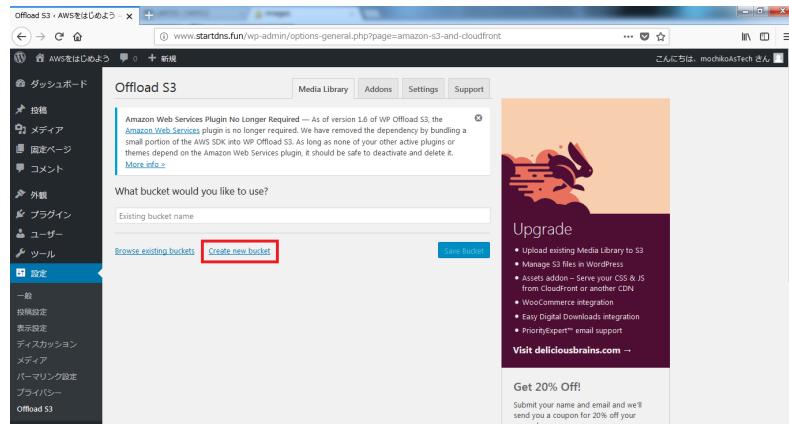
▲図 6.25 「有効化」になったらクリック

WP Offload S3 Lite プラグインがインストールできたので「Settings」をクリック（図 6.26）します。



▲図 6.26 WP Offload S3 Lite プラグインの「Settings」をクリック

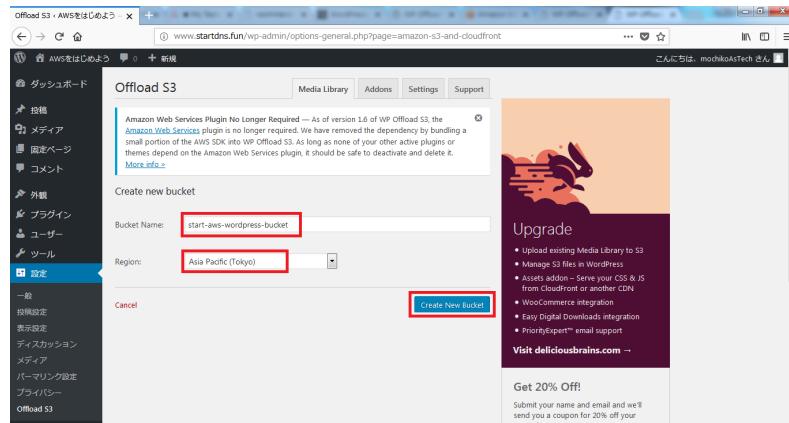
WordPressでアップした画像を保存しておくための「バケット」と呼ばれる入れ物がS3にまだないので「Create new bucket」をクリック（図6.27）します。



▲図 6.27 バケットがまだないので「Create new bucket」をクリック

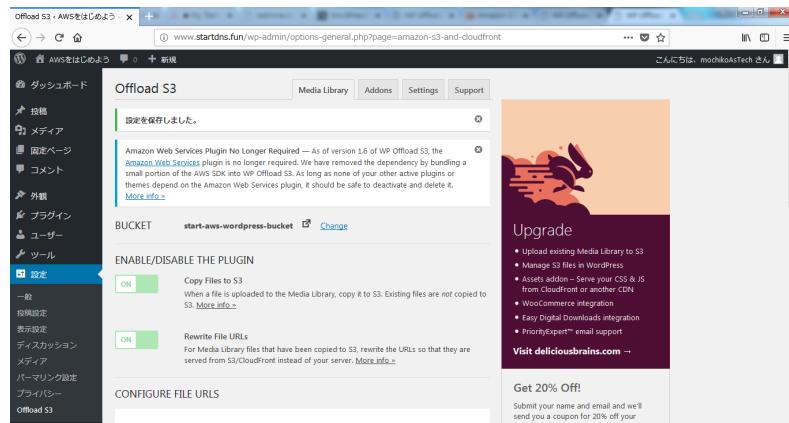
「Bucket Name」に「start-aws-wordpress-bucket」と入力（図6.28）し、「Region」は「Asia Pacific (Tokyo)」を選択したら「Create new bucket」をクリックします。

6.3 画像を S3 に保存する



▲図 6.28 バケットの名前とリージョンを入れて「Create new bucket」をクリック

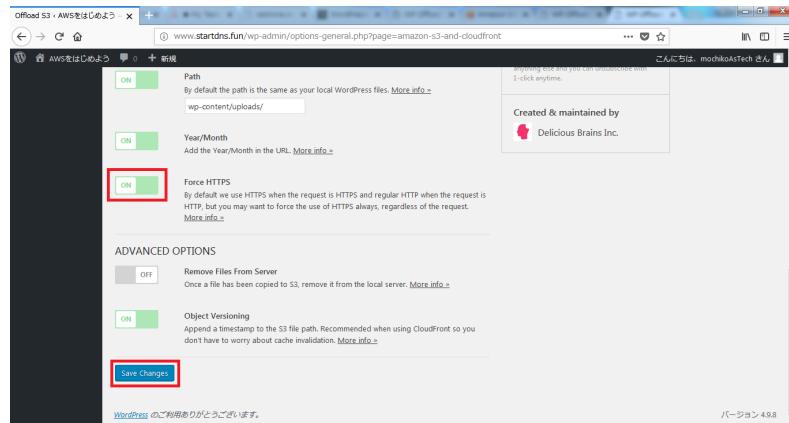
「設定を保存しました。」と表示（図 6.29）されました。これでバケットの作成が完了して画像が S3 へ保存されるようになりました。



▲図 6.29 バケットの作成が完了して画像が S3 へ保存されるようになった

下にスクロールして「Force HTTPS」をオン（図 6.30）にしたら「Save Changes」をクリックします。^{*10}

*¹⁰ デフォルトだと「ページを HTTP で開いたら画像も HTTP で表示、ページを HTTPS で開いたら画像も HTTP で表示」という設定なのですが、「Force HTTPS」をオンにしておくと「画像は常に HTTPS で表示」になります。今後サイトを HTTPS 化したときに画像の URL を HTTP から HTTPS へ書き換えなくて済むようオンにしておくことをお勧めします。



▲図 6.30 画像は常に HTTPS で表示されるよう「Force HTTPS」をオンにして「Save Changes」をクリック

「設定を保存しました。」と表示されたら WP Offload S3 Lite プラグインの設定は完了^{*11}です。

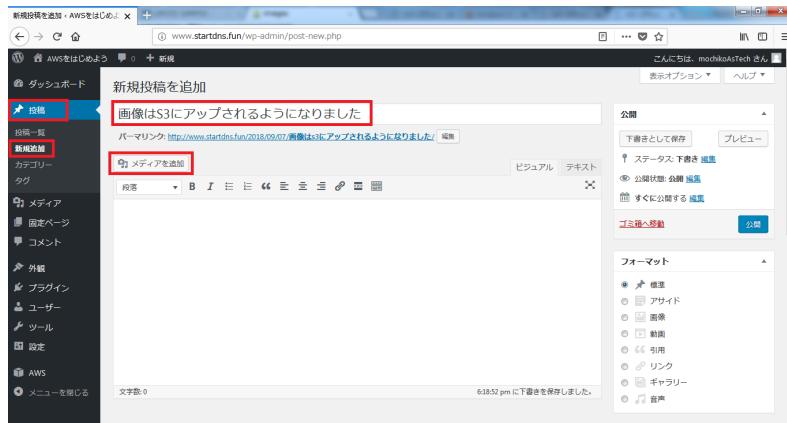
6.3.4 投稿を試してみよう

画像を S3 にアップする設定がされたか、試しに記事を投稿してみましょう。

左メニューの「投稿」から「新規追加」をクリック（図 6.31）したら、タイトルに「画像は S3 にアップされるようになりました」と入力します。タイトルが入力できたら「メディアを追加」をクリックしてください。

^{*11} ちなみに WP Offload S3 Lite プラグインでは S3 ではなく CloudFront（クラウドフロント）を使う設定にもできるようです。本著では扱いませんが CloudFront は CDN（Content Delivery Network）のサービスです。CloudFront を使うと画像や動画などのコンテンツを世界中の CDN サーバを使って配信できるため、ウェブサイトが高速で表示されるようになります。

6.3 画像を S3 に保存する



▲図 6.31 タイトルを入力したら「メディアを追加」をクリック

「ファイルを選択」をクリック（図 6.32）して適当な画像を選択したら「投稿に挿入」をクリックします。



▲図 6.32 適当な画像を選択したら「投稿に挿入」をクリック

「公開」をクリック（図 6.33）します。



▲図 6.33 「公開」をクリック

投稿が公開されたら「投稿を表示」をクリック（図 6.34）してください。



▲図 6.34 「投稿を表示」をクリック

記事を確認したら画像を右クリックして「画像だけを表示」をクリック（図 6.35）します。



▲図 6.35 画像を右クリックして「画像だけを表示」をクリック

すると画像の URL が「https://s3-ap-northeast-1.amazonaws.com/start-aws-wordpress-bucket/wp-content/uploads/2018/09/07182149/circle_cut_color.png」のようになっている（図 6.36）ので、WordPress でアップした画像が S3 に保存されて記事に挿入されていることが分かります。



▲図 6.36 画像の URL が S3 になっている

以上で WordPress のインストールと設定は完了です。お疲れさまでした！

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。第4章「ウェブサーバの設定をしよう」ではバーチャルホストを作りました。第6章「WordPressでサイトを作ろう」ではWordPressのインストールをしました。頑張っていっぱい設定しましたね。

ところでもし設定した内容が全部消えてしまって「1からやり直し」と言わされたらどうなるでしょう？きっと心が折れてしまいますが・・・。

viで頑張って修正した設定ファイルやドキュメントルートに設置したWordPressのファイルなどはすべてEBSボリューム^{*1}に保存されています。ある日うっかり次のようなrmコマンド^{*2}を叩いてしまって「何もかも全部消えちゃった・・・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。

```
# rm -rf /
```

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。

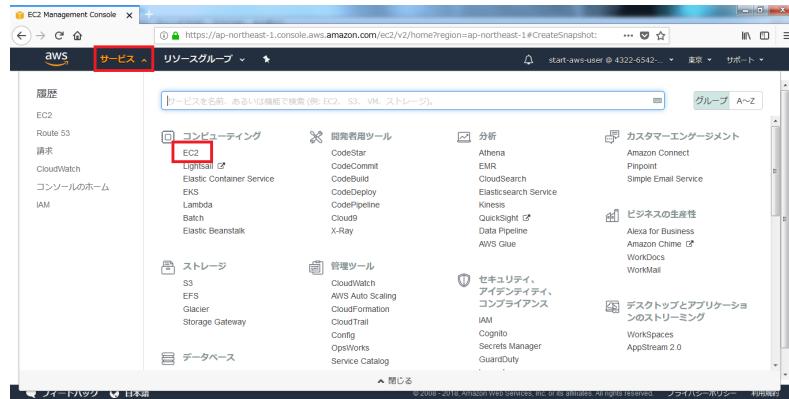
ただしスナップショットを取るときはデータの整合性を保つためにハードディスクへの読み書きを停止する必要があります。インスタンスを起動して使用しているまでもスナップショットを取ることはできますが、たとえば「MySQL（データベース）がトランザクション処理中でゴリゴリ書き込みをしている最中にスナップショットを取るとデータの整合性が取れない」といった可能性があり、ファイルシステムの完全性が保証できないので推奨されていません。ハードディスクへの読み書きがされない状態、つまりインスタンスを先にシャットダウンしておくか、もしくは対象となるEBSボリュームをアンマウ

^{*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、削除対象に「/」（一番上のルートディレクトリ）を指定しているので「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。ただ実際はこのコマンドを叩いても「rm: it is dangerous to operate recursively on '/'」となしなめられるだけで、ファイルは1つも消えませんので安心してください。

ント^{*3}してからスナップショットを取るようになります。

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。



▲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

^{*3} ハードディスクをサーバから取り外すこと。

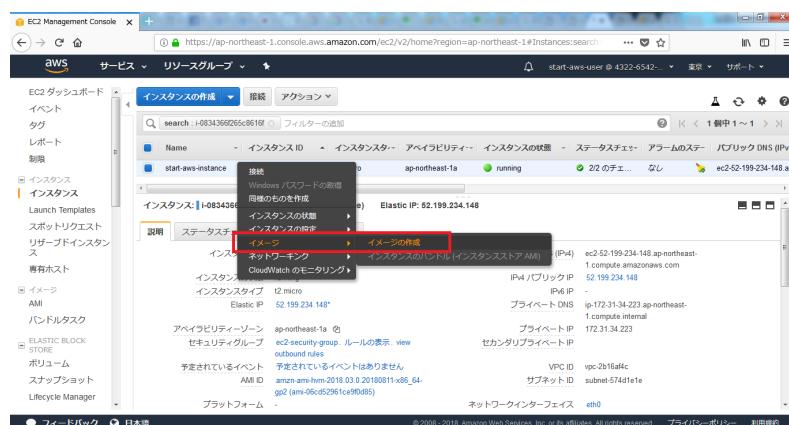
マシンイメージ、略して AMI を選択したのを覚えていますか？ そのときは AMI というテンプレートのようなものを元にインスタンスを作りましたが、その逆でインスタンスから自分専用のカスタム AMI を作ることもできます。

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。



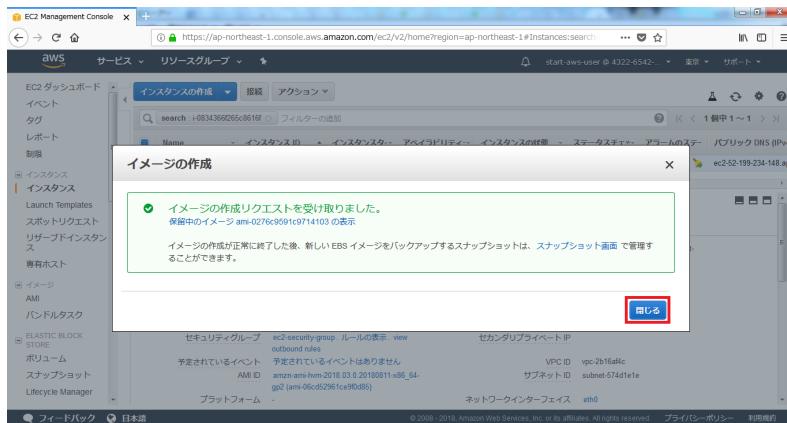
▲図 7.3 インスタンスを右クリックして「イメージの作成」をクリック

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」をクリックします。



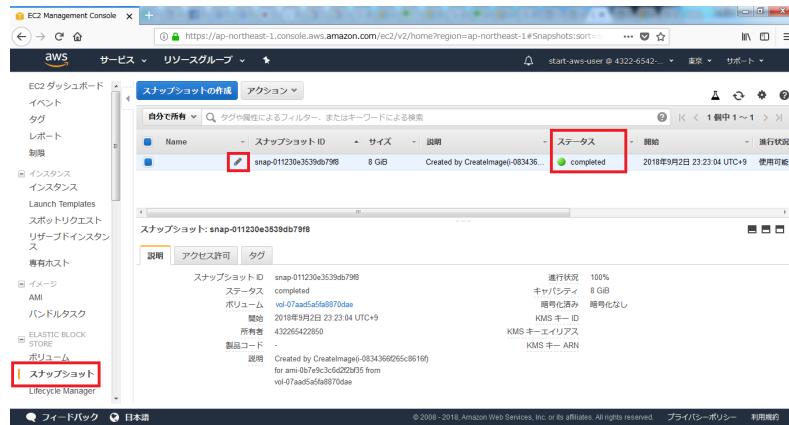
▲図 7.5 「閉じる」をクリック

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

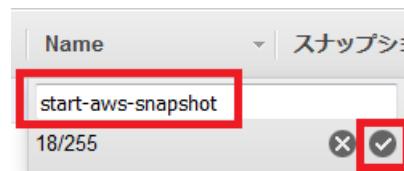
という複数の処理が実行^{*4}されています。

少し待ってから左メニューで「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」になつていれば 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

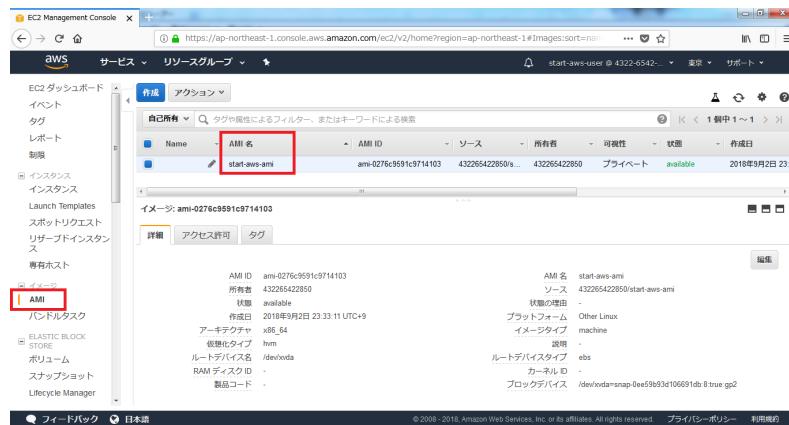
Name に「start-aws-snapshot」と書いたらチェックボタンを押して（図 7.7）ください。スナップショット名には日本語を使わないことをお勧めします。



▲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復元できます。



▲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 と Auto Scaling で複数台の サーバを運用しよう

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 Auto Scaling の設定をします。

8.1 ELB はなんのためにある？

ウェブサーバやデータベースサーバなどのインフラ環境を構築するときには「ここが壊れたり止まつたりしたらサービス全体が停止する」という SPOF^{*1}をつぶしておくことが大切です。

皆さんのが本著で構築した環境は「EC2 のウェブサーバ 1 台 + RDS のデータベースサーバ 1 台」という低コストのお手軽構成ですので、ウェブサーバもデータベースサーバもどちらも SPOF になっています。たとえば EC2 のウェブサーバが壊れたらその瞬間に WordPress のサイトは見られなくなります。^{*2}

この SPOF をなくすための手段の 1 つとして ELB^{*3}、いわゆる負荷分散のためのロードバランサーのサービスがあります。もし EC2 のウェブサーバが 2 台あって、その手前のロードバランサーがアクセスを 2 台に振り分けていたら、1 台が停止している間ももう 1 台が応答できるのでサイトは停止しません。

「予算がないのでウェブサーバを 2 台用意するのは無理です」とか「小規模なサイトなのでそこまでの冗長性は求めていません」という場合でも ELB は役に立ちます。ELB を使って 1 台のウェブサーバにアクセスを流していた場合、Auto Scaling（オートスケーリング）というサービスを使うことで、その 1 台が死んでしまったときに自動でインスタンスを立ててサイトを自動復旧させてくれます。

他にも ELB を使うと無料で SSL 証明書を取得してサイトを楽に HTTPS 化できるなど色々なメリットがあります。ELB は無料利用枠にも含まれていますので、ここでは ELB から EC2 のインスタンスへアクセスを流すロードバランシングの設定をしてみましょう。

8.2 ロードバランサーを作ろう

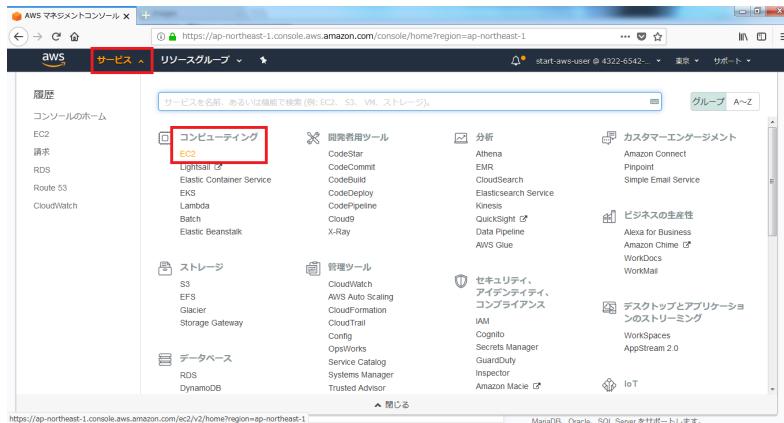
それではマネジメントコンソールの左上にある「サービス」から、「コンピューティング」の下にある「EC2」（図 8.1）をクリックしてください。

^{*1} Single Point of Failure の略。「ここが壊れたらシステム全体が停止する」という単一障害点のこと。

^{*2} サーバが壊れると言わてもピンと来ないかも知れませんが、ホストサーバのハードウェアが壊れてしまえば、当然その上で動いていたゲストサーバ（EC2 のインスタンス）も影響を受けて「突然うんともすんとも言わなくなった！」状態になることがあります。ただその場合も OS やデータが入った EBS ボリュームは生きているので、EC2 ダッシュボードを開いて手動でインスタンスを起動させれば別のホストサーバ上でまた元気に働きだすはずです。

^{*3} Elastic Load Balancing の略。

8.2 ロードバランサーを作ろう



▲図 8.1 サービス>コンピューティング> EC2

「EC2」をクリックすると、EC2 のダッシュボード（図 8.2）が表示されます。左メニューの「ロードバランサー」をクリックしてください。



▲図 8.2 左メニューの「ロードバランサー」をクリック

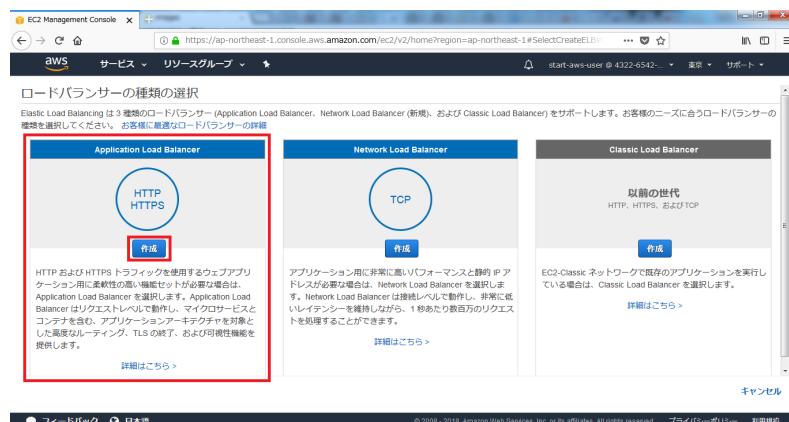
「ロードバランサーの作成」をクリック（図 8.3）します。



▲図 8.3 「ロードバランサーの作成」をクリック

8.2.1 ロードバランサーの種類の選択

ロードバランサーにもいくつか種類があるのですが、今回はいちばん左の「Application Load Balancer」を使います。「作成」をクリック（図 8.4）してください。



▲図 8.4 「Application Load Balancer」の「作成」をクリック

ここからは 6 つのステップでロードバランサーを作成していきます。

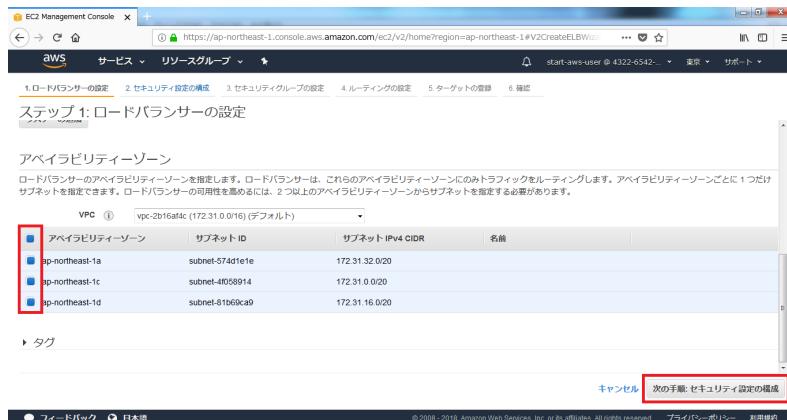
8.2.2 ステップ 1: ロードバランサーの設定

名前に「start-aws-elb」と入力（図 8.5）したらそのまま下の方へスクロールします。



▲図 8.5 名前に「start-aws-elb」と入力

アベイラビリティーゾーンはすべてにチェック（図 8.6）を入れて、「次の手順: セキュリティ設定の構成」をクリックします。



▲図 8.6 AZ はすべてにチェックを入れて「次の手順: セキュリティ設定の構成」をクリック

8.2.3 ステップ2: セキュリティ設定の構成

いきなり黄色で「ロードバランサーのセキュリティを向上させましょう。ロードバランサーは、いずれのセキュアリスナーも使用していません。」と表示（図8.7）されました。が、これは「HTTPじゃなくともっとセキュアなHTTPSにした方がいいよ」とアドバイスをしてくれているだけですので、今はそのままスルーで構いません。「次の設定: セキュリティグループの設定」をクリックします。



▲図8.7 「次の設定: セキュリティグループの設定」をクリック

8.2.4 ステップ3: セキュリティグループの設定

「セキュリティグループの割り当て」で「新しいセキュリティグループを作成する」を選択（図8.8）します。「セキュリティグループ名」と「説明」は次のように入力してください。（表8.1）

▼表8.1 セキュリティグループの設定

| | |
|-------------|--------------------------|
| セキュリティグループ名 | elb-security-group |
| 説明 | HTTP Allow from anywhere |

続いて「ここからのアクセスのみを通す」というルールを設定します。ルールはタイプが「HTTP」で、ソースを「カスタム」の「0.0.0.0/0」にします。



▲図 8.8 セキュリティグループを設定したら「次の手順: ルーティングの設定」をクリック

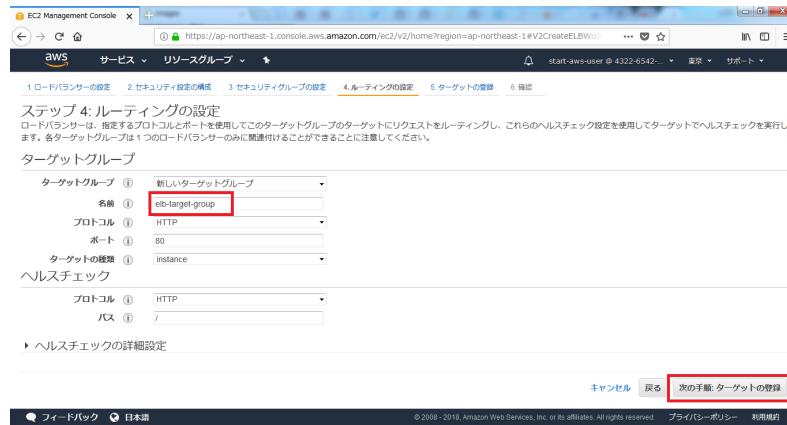
これはロードバランサーの手前にあるセキュリティグループ（ファイアウォール）を「HTTP（ポート番号 80 番）ならどこからのリクエストでも通す」という設定にしています。ルールを設定したら「次の手順: ルーティングの設定」をクリックします。

8.2.5 ステップ 4: ルーティングの設定

ターゲットグループというのはロードバランサーの分散先となるサーバのグループのことです。つまり「このグループに入っているインスタンスにアクセスを割り振ります」ということです。

名前に「elb-target-group」と入力（図 8.9）したら、それ以外の設定^{*4}はすべてのそのまま構いません。「次の手順: ターゲットの登録」をクリックします。

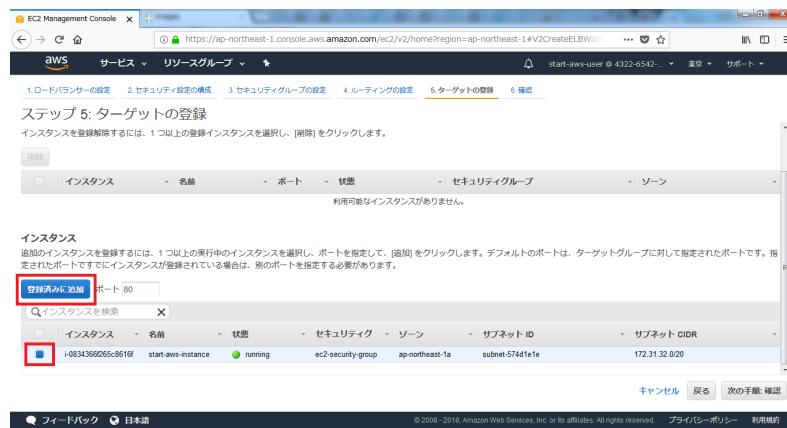
^{*4} ちなみにこのページで設定している「ヘルスチェック」とは、名前のとおりロードバランサーがターゲットグループのインスタンスに対して行う「ねえウェブサーバ生きてる？ ねえねえ生きてる？」という死活チェックのことです。このチェックに対してウェブサーバが応答をしなくなると「あ、死んでるからこのウェブサーバにアクセス流すのやめよう」となって負荷分散対象から除外されるのです。



▲図 8.9 名前に「elb-target-group」と入力したら「次の手順: ターゲットの登録」をクリック

8.2.6 ステップ 5: ターゲットの登録

下部の「インスタンス」にある「start-aws-instance」にチェック（図 8.10）を入れたら「登録済みに追加」をクリックします。



▲図 8.10 「start-aws-instance」にチェックを入れたら「登録済みに追加」をクリック

上部の「登録済みターゲット」に「start-aws-instance」が表示（図 8.11）されたら「次の手順: 確認」をクリックします。

8.2 ロードバランサーを作ろう



▲図 8.11 「登録済みターゲット」に「start-aws-instance」が表示されたら「次の手順: 確認」をクリック

8.2.7 ステップ 6: 確認

設定内容を確認したら「作成」をクリック（図 8.12）します。



▲図 8.12 設定内容を確認したら「作成」をクリック

「ロードバランサーを正常に作成しました」と表示（図 8.13）されたら「閉じる」をクリックします。



▲図 8.13 設定内容を確認したら「作成」をクリック

これでロードバランサーの作成は完了です。(図 8.14)

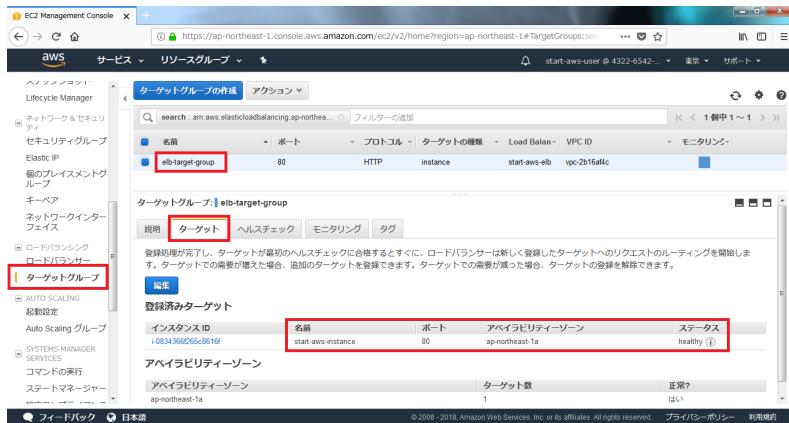


▲図 8.14 ロードバランサーの作成は完了

8.2.8 ヘルスチェックの確認をしよう

ロードバランサーの作成が完了したら、EC2 ダッシュボードの左メニューで「ターゲットグループ」をクリック(図 8.15)して「elb-target-group」の「ターゲット」タブを確認してみましょう。

8.3 WordPress のサイトを表示する経路を変更しよう



▲図 8.15 「start-aws-instance」のステータスを確認してみよう

ターゲットである「start-aws-instance」のステータスが healthy になっていれば、ロードバランサーからの「ねえ生きてる？」に対して EC2 で作ったウェブサーバが「生きてる！」と元気に応答しているということです。

もしステータスが「unhealthy」になっていたら「ねえ生きてる？」にちゃんと応答できませんので、次の確認をしましょう。

- EC2 のインスタンス (start-aws-instance) が起動しているか？
- EC2 のインスタンスで Apache が起動しているか？
- EC2 の手前にいるセキュリティグループ (ec2-security-group) で HTTP (ポート番号 80 番) のアクセスを許可しているか？

8.3 WordPress のサイトを表示する経路を変更しよう

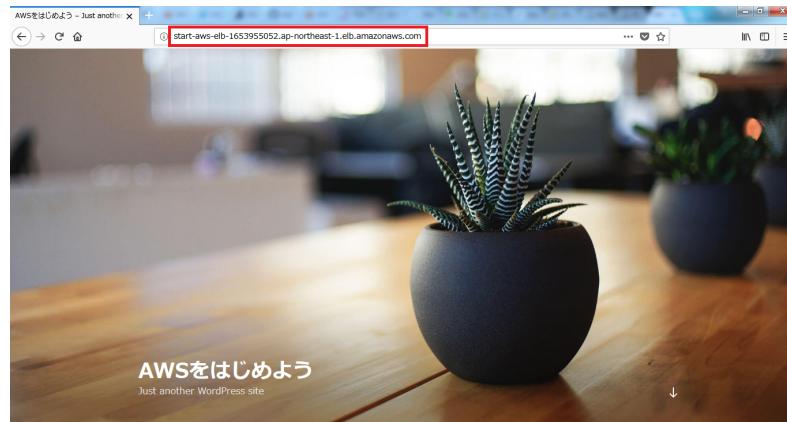
8.3.1 「www. 自分のドメイン」と紐づく IP アドレスを変更しよう

EC2 ダッシュボードの左メニューで「ロードバランサー」をクリック（図 8.16）したら「start-aws-elb」の「説明」タブにある「DNS 名」をコピーしてください。



▲図 8.16 「start-aws-elb」の「説明」タブにある「DNS名」をコピー

この DNS 名をそのままブラウザで開く（図 8.17）と WordPress のサイトが表示されます。



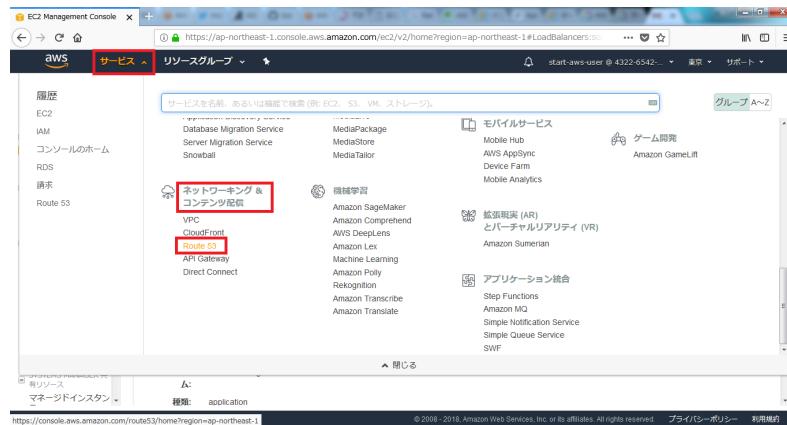
▲図 8.17 DNS名をそのままブラウザで開くと WordPress のサイトが表示される

これでロードバランサーを経由して WordPress のサイトが見られることが確認できたので、Route53 で DNS の変更を行いましょう。

8.3 WordPress のサイトを表示する経路を変更しよう

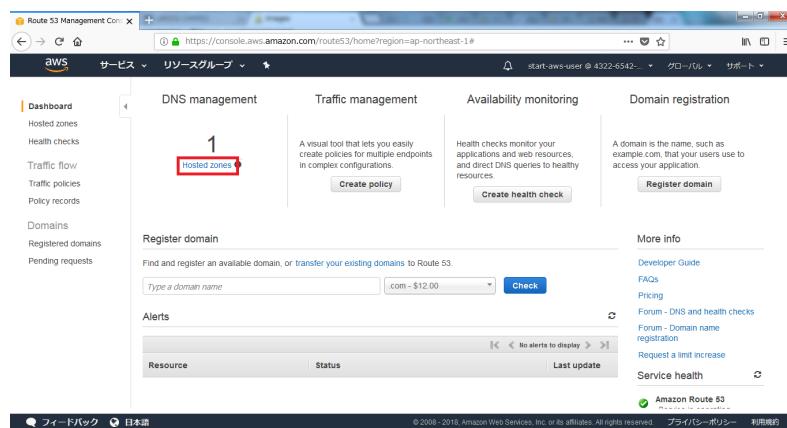
8.3.2 Route53 で A レコードを Alias に変更しよう

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 8.18）をクリックしてください。



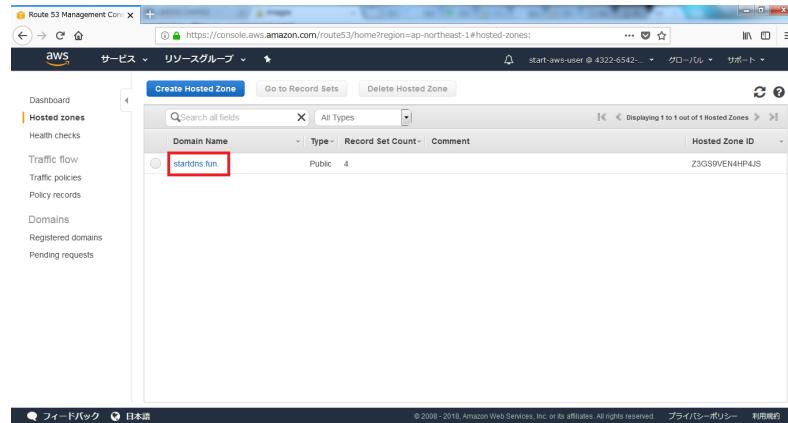
▲図 8.18 サービス>ネットワーキング&コンテンツ配信>Route53

Route53 ダッシュボードを開いたら DNS management の「Hosted zones」をクリック（図 8.19）します。



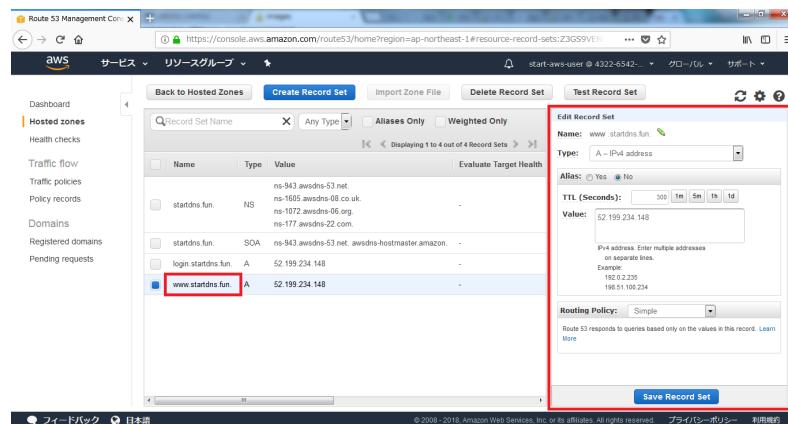
▲図 8.19 「Hosted zones」をクリック

Domain Name の自分のドメイン名（筆者の場合は startdns.fun）をクリック（図 8.20）します。



▲図 8.20 自分のドメイン名をクリック

「www. 自分のドメイン名」（筆者の場合は「www.startdns.fun」）をクリック（図 8.21）すると右側に既存の A レコードが表示されます。

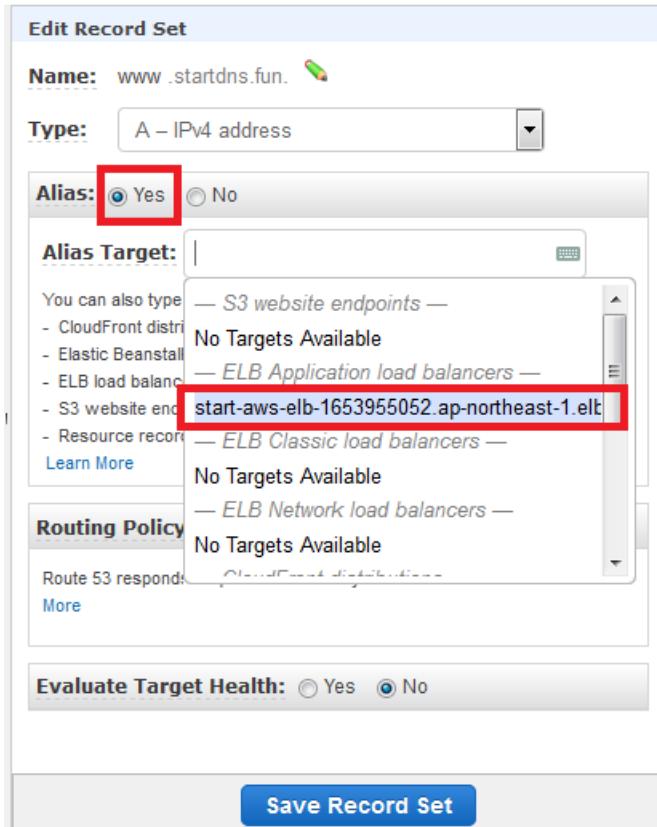


▲図 8.21 「www. 自分のドメイン名」をクリックすると右側に既存の A レコードが表示される

現状は value に EC2 インスタンスの Elastic IP が設定されています。つまりこのままだと「<http://www.自分のドメイン名/>」を開いたときに、ロードバランサー経由ではな

く直接 EC2 インスタンスへ「ウェブページを見せて！」とリクエストが行ってしまい、ロードバランサーを作成した意味がありません。「www. 自分のドメイン名」と紐づく IP アドレスを Elastic IP から ELB に変更しましょう。

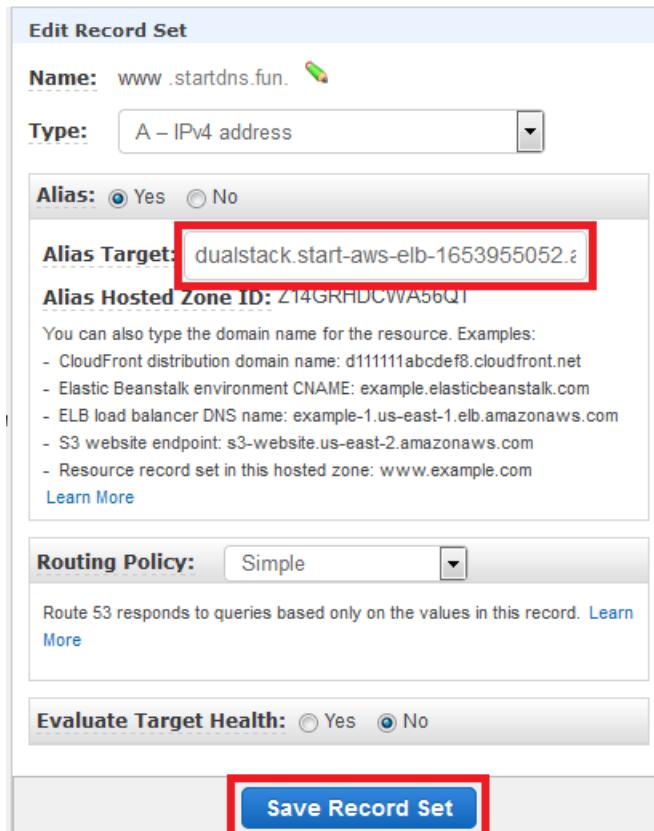
Alias を「No」から「Yes」に変更して「Alias Target」の「Enter target name」と書かれたところをクリック（図 8.22）すると、プルダウンで「start-aws-elb」の DNS 名^{*5}が表示されるのでクリックしてください。



▲図 8.22 「www. 自分のドメイン名」をクリックすると右側に既存の A レコードが表示される

「Alias Target」が「start-aws-elb」になつたら「Save Record Set」をクリック（図 8.23）します。

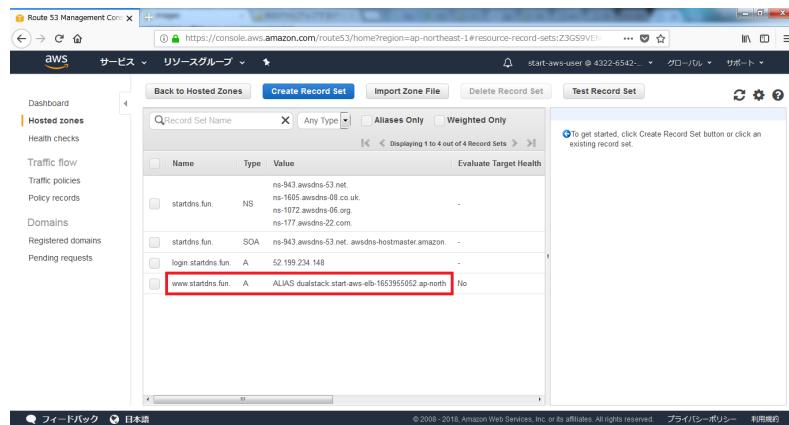
^{*5} さつきブラウザで開いで WordPress のサイトが見られた DNS 名です。



▲図 8.23 「Alias Target」が「start-aws-elb」になったら「Save Record Set」をクリック

これで「www. 自分のドメイン名」と紐づく IP アドレスが Elastic IP から ELB に変わりました。

8.3 WordPress のサイトを表示する経路を変更しよう



▲図 8.24 「www. 自分のドメイン名」と紐づく IP アドレスが Elastic IP から ELB に変わった

それでは EC2 のインスタンスで dig コマンド^{*6}を叩いて A レコードの設定が変更されたか確認してみましょう。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」と表示されたらログイン完了です。次のコマンドを叩いてみてください。

```
$ dig www. 自分のドメイン名 +short
```

筆者ならドメイン名が「startdns.fun.」なので次のようになります。

```
$ dig www.startdns.fun +short  
54.178.188.244  
54.64.131.11
```

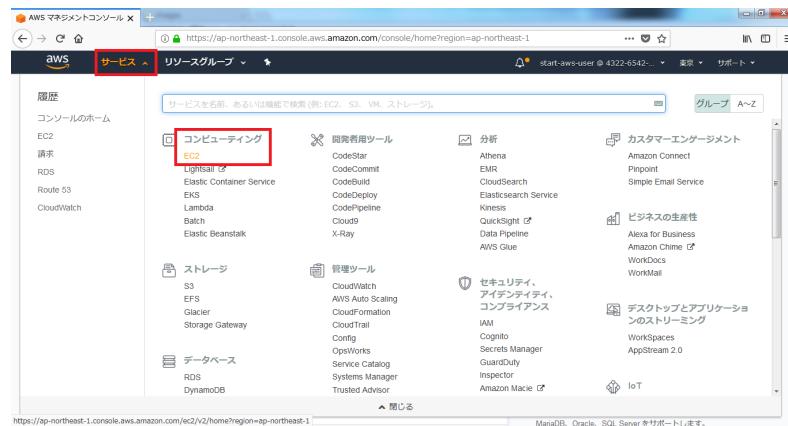
^{*6} dig コマンドについては「DNS をはじめよう」の第 4 章「dig と whois を叩いて学ぶ DNS」を参照してください。

このようにIPアドレスが2つ返ってきたらAレコードの設定はちゃんと変更されています。これで「<http://www.自分のドメイン名/>」を開いたときに、ロードバランサーを経由してWordPressのサイトが表示されるようになりました。

8.3.3 HTTPを通すのはELB経由のアクセスだけにしよう

Aレコードを変更したことで、今後は「サイトを見たい人→ロードバランサー→EC2のインスタンス」という経路だけを使うことになるので「サイトを見たい人→EC2のインスタンス(Elastic IP)」という不要な経路はふさいでおきましょう。

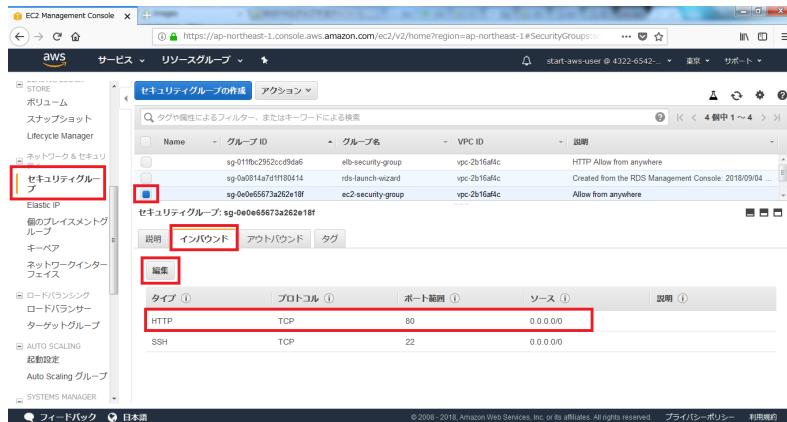
それではマネジメントコンソールの左上にある「サービス」から、「コンピューティング」の下にある「EC2」(図8.25)をクリックしてください。



▲図8.25 サービス>コンピューティング>EC2

EC2のダッシュボードが表示(図8.26)されたら左メニューの「セキュリティグループ」をクリックします。「ec2-security-group」のインバウンドタブをクリックすると、現状は「HTTP(ポート番号80番)はどこからのリクエストでも通す」という設定になっています。ここを「HTTP(ポート番号80番)はELB経由のリクエストのみ通す」という設定にしたいので「編集」をクリックしてください。

8.3 WordPress のサイトを表示する経路を変更しよう



▲図 8.26 「ec2-security-group」のインバウンドタブをクリック

HTTP の「ソース」に書いてある「0.0.0.0/0」を消して、代わりに「elb」と入力（図 8.27）するとプルダウンで「elb-security-group」が表示されますのでクリックしてください。



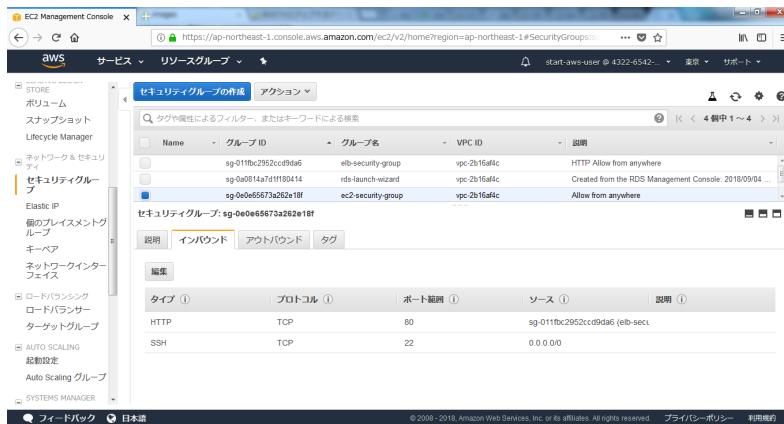
▲図 8.27 「elb」と入力すると「elb-security-group」が表示されるのでクリック

ソースが「elb-security-group」の（図 8.28）グループ ID になったら「保存」をクリックします。



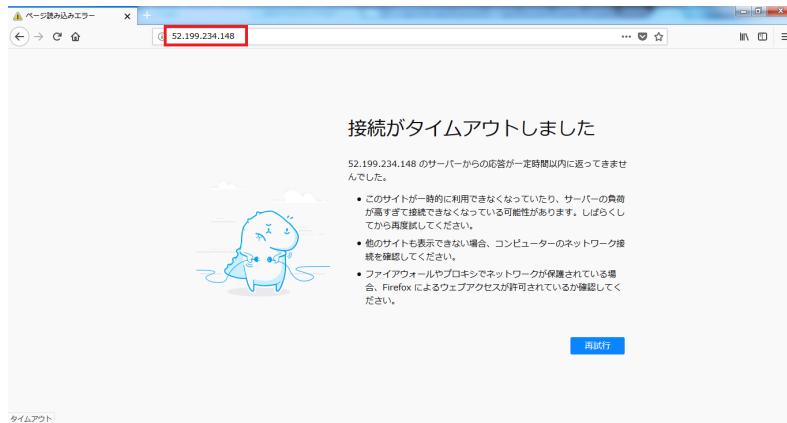
▲図 8.28 ソースが「elb-security-group」のグループ ID になったら「保存」をクリック

これで EC2 インスタンスの手前にいるセキュリティグループが「HTTP（ポート番号 80 番）は ELB 経由のリクエストのみ通す」という状態（図 8.29）になりました。



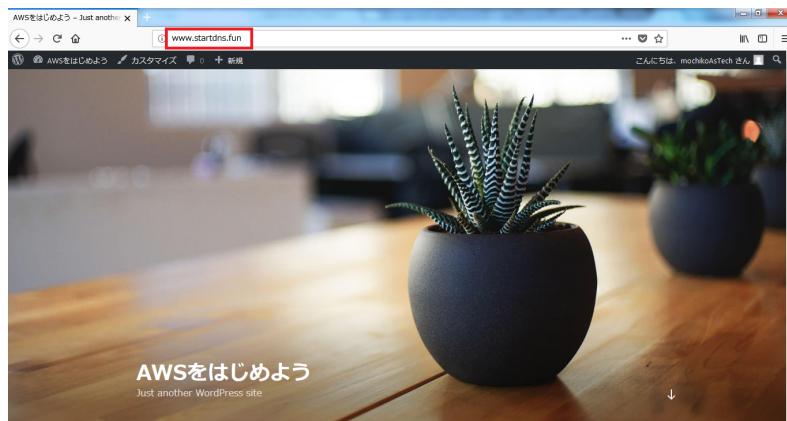
▲図 8.29 セキュリティグループが「HTTP（ポート番号 80 番）は ELB 経由のリクエストのみ通す」という状態になった

ブラウザで EC2 インスタンスに紐づいている Elastic IP を直接叩く（図 8.29）とセキュリティグループに阻まれて WordPress のサイトは表示されません。



▲図 8.30 Elastic IP を直接叩くとセキュリティグループに阻まれてサイトが表示されない

でも ELB を経由する「<http://www.自分のドメイン名/>」を開いたときはちゃんと表示（図 8.31）されます。



▲図 8.31 ELB を経由する「<http://www.自分のドメイン名/>」を開いたときはちゃんと表示される

8.4 Auto Scaling

AWS Auto Scaling (オートスケーリング) はサーバの自動拡張・縮小をしてくれるサービスです。アクセスが増えてきてウェブサーバ 1 台ではさばききれなくなったら Auto

Scalingが自動的に追加のサーバを立ててくれますし、アクセス数が落ち着いてきて1台で十分な状態になったら自動的に不要なサーバを削除してくれます。

しかし本著ではAuto Scalingを拡張や縮小ではなく「インスタンス数の維持」のために利用します。何か問題が起きてEC2のインスタンスが停止してしまっても、Auto Scalingによって新たにインスタンスが1台立てられてウェブサイトが自動復旧する状態を目指します。

8.4.1 起動設定を作成しよう

EC2ダッシュボードの左メニューで「AUTO SCALING」の下にある「起動設定」をクリック（図8.32）してください。この「起動設定」ではAuto Scalingが自動で立てるEC2インスタンスのAMIやインスタンスタイプを指定します。続いて「起動設定の作成」をクリックしてください。



▲図8.32 「起動設定」で「起動設定の作成」をクリック

ここからは6つのステップで起動設定を作成していきます。

1. AMIの選択

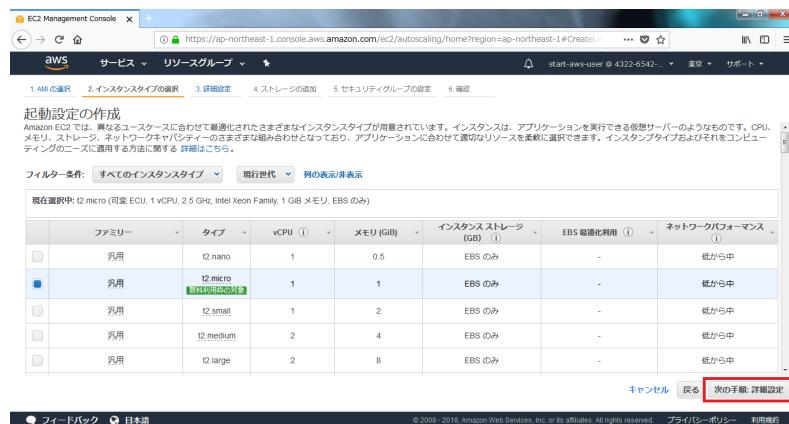
「マイAMI」をクリックしたら第7章「サーバのバックアップを取っておこう」で作成した「start-aws-ami」というAMIを選択（図8.33）します。



▲図 8.33 「マイ AMI」で「start-aws-ami」を選択

2. インスタンスタイプの選択

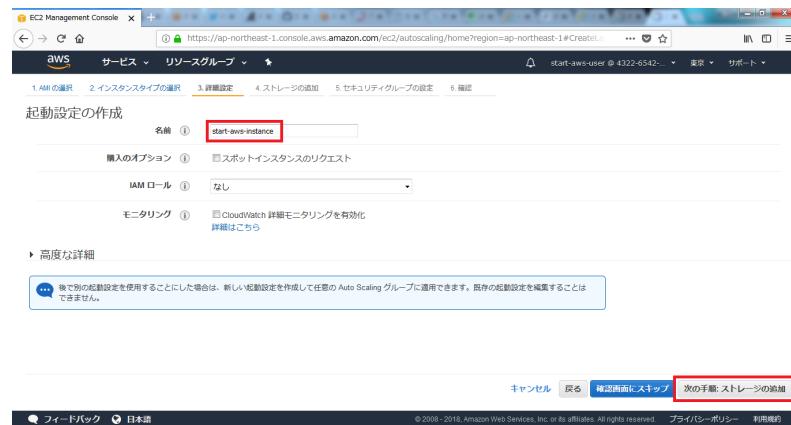
インスタンスタイプは現在のインスタンスと同じ「t2.micro」が選択されているので、そのまま「次の手順: 詳細設定」をクリック（図 8.34）します。



▲図 8.34 「t2.micro」が選択されているのでそのまま「次の手順: 詳細設定」をクリック

3. 詳細設定

「名前」には現在のインスタンスと同じ「start-aws-instance」を入力して「次の手順: ストレージの追加」をクリック（図 8.35）します。



▲図 8.35 「名前」に「start-aws-instance」を入力して「次の手順: ストレージの追加」をクリック

4. ストレージの追加

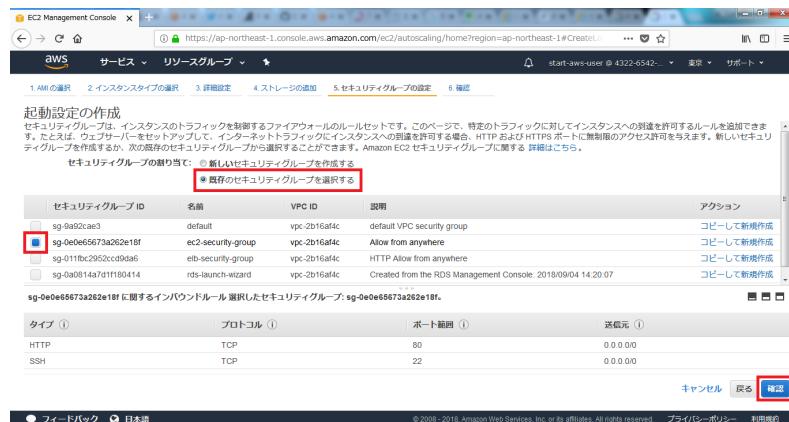
現在のインスタンスと同じ 8GB のままでいいので、そのまま「次の手順: セキュリティグループの設定」をクリック（図 8.36）してください。



▲図 8.36 そのまま「次の手順: セキュリティグループの設定」をクリック

5. セキュリティグループの設定

「既存のセキュリティグループを選択する」をクリック（図 8.37）して、現在のインスタンスと同じ「ec2-security-group」にチェックを入れたら「確認」をクリックします。



▲図 8.37 「ec2-security-group」にチェックを入れたら「確認」をクリック

6. 確認

「起動設定」の内容を確認したら「起動設定の作成」をクリック（図 8.38）してください。



▲図 8.38 「起動設定」の内容を確認したら「起動設定の作成」をクリック

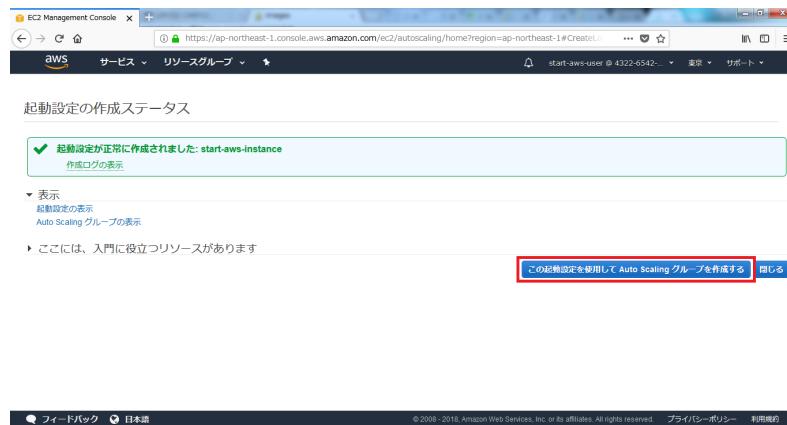
既存のキーペアを選択

「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8.39）されました。これはAuto Scalingが自動で立てるEC2インスタンスに、既存のキーペアの鍵穴を設置しますか？それとも新しくキーペアを作り直してその鍵穴を設置しますか？と聞かれています。Auto Scalingで自動起動したインスタンスにも同じ鍵でSSHログインしたいので「既存のキーペアを選択」で「start-aws-keypair」になっていることを確認したら、チェックボックスにチェックを入れて「起動設定の作成」をクリックしてください。



▲図8.39 チェックボックスにチェックを入れて「起動設定の作成」をクリック

「起動設定が正常に作成されました: start-aws-instance」と表示されたら「この起動設定を使用してAuto Scalingグループを作成する」をクリック（図8.40）します。



▲図 8.40 起動設定が作成できたら「この起動設定を使用して Auto Scaling グループを作成する」をクリック

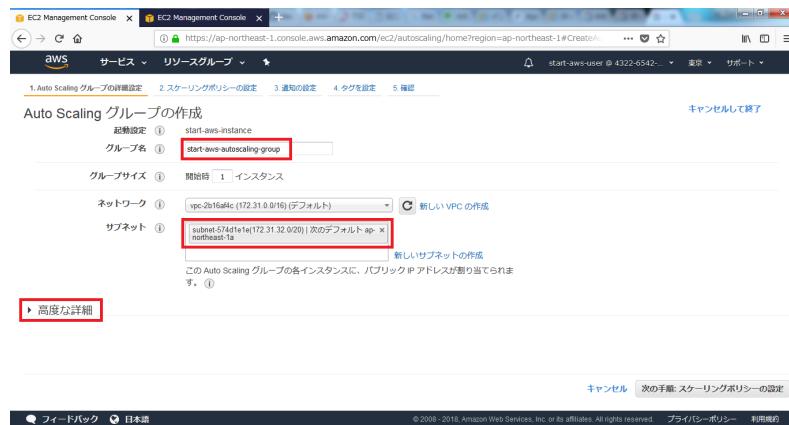
8.4.2 Auto Scaling グループを作成しよう

「起動設定」ができたら続いて「Auto Scaling グループ」を作成します。Auto Scaling グループは EC2 インスタンスのグループのことです、このグループで常に維持したいインスタンスの数などを設定します。インスタンスの数はここで設定した最小数と最大数の間で増減します。

ここからは 5 つのステップで Auto Scaling グループを作成していきます。

1. Auto Scaling グループの詳細設定

グループ名に「start-aws-autoscaling-group」と入力して、サブネットは既存の EC2 インスタンスと同じ「ap-northeast-1a」を選択したら「高度な詳細」をクリック（図 8.41）します。



▲図 8.41 グループ名に「start-aws-autoscaling-group」と入力したら「高度な詳細」をクリック

「高度な詳細」では「ロードバランシング」にチェック（図 8.42）を入れます。「ターゲットグループ」で「elb-target-group」を選択して、「ヘルスチェックのタイプ」は「ELB」を選択します。これで ELB からのヘルスチェックに対してインスタンスが応答しなくなったら、Auto Scaling によって自動的にインスタンスが立てられることになります。すべて設定したら「次の手順: スケーリングポリシーの設定」をクリックします。



▲図 8.42 設定したら「次の手順: スケーリングポリシーの設定」をクリック

2. スケーリングポリシーの設定

本著では Auto Scaling を拡張や縮小ではなく「インスタンス数の維持」のために利用したいので、「このグループを初期のサイズに維持する」のままで「次の手順: 通知の設定」をクリック（図 8.43）します。



▲図 8.43 何も変更せず「次の手順: 通知の設定」をクリック

3. 通知の設定

「通知の追加」をクリックします。（図 8.44）



▲図 8.44 「通知の追加」をクリック

「通知の送信先」に自分の名前、「受信者」に自分のメールアドレス^{*7}を記入したら「次の手順: タグを設定」をクリック（図 8.45）します。



▲図 8.45 自分の名前とメールアドレスを記入したら「次の手順: タグを設定」をクリック

4. タグを設定

何も変更せず「確認」をクリック（図 8.46）します。



▲図 8.46 何も変更せず「確認」をクリック

^{*7} メールアドレスの確認のため「AWS Notification - Subscription Confirmation」というメールが届きます。メール本文中にある「Confirm subscription」というリンクを踏んでおいてください。

5. 確認

内容を確認したら「Auto Scaling グループの作成」をクリック（図 8.47）してください。



▲図 8.47 内容を確認したら「Auto Scaling グループの作成」をクリック

「Auto Scaling グループが正常に作成されました」と表示されたら「閉じる」をクリック（図 8.48）します。

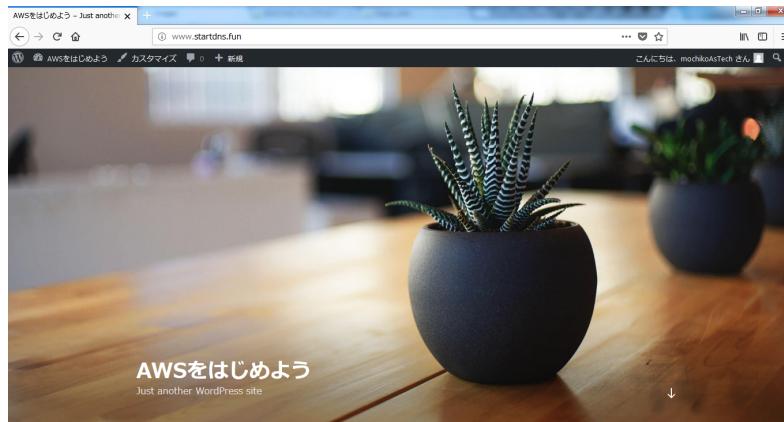


▲図 8.48 「閉じる」をクリック

これで「start-aws-autoscaling-group」という Auto Scaling グループが作成できました。早速インスタンスを削除して自動復旧するかテストしてみましょう。

8.4.3 インスタンスを削除して自動復旧を試してみよう

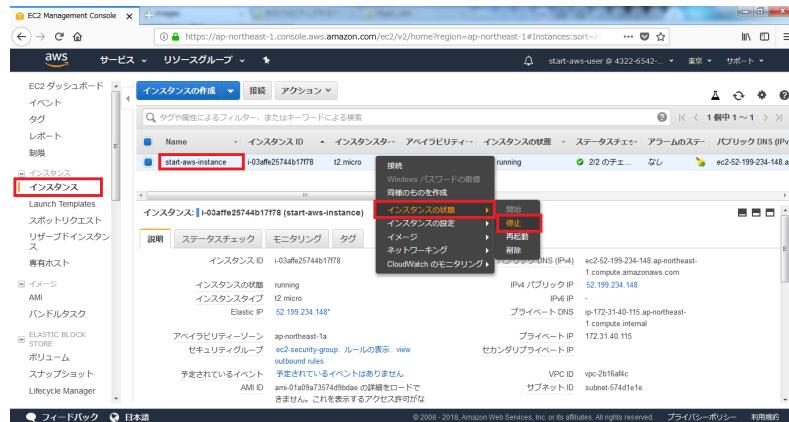
それではインスタンスを停止してもサイトが自動復旧するのか試してみましょう。まずブラウザで「<http://www.自分のドメイン名/>」を開いてサイトが表示（図 8.49）されていることを確認します。



▲図 8.49 <http://www.自分のドメイン名/>を開くとサイトが表示される

続いて EC2 ダッシュボードの左メニューで「インスタンス」をクリックしたら「start-aws-instance」を右クリックして、「インスタンスの状態」から「削除」（図 8.50）をクリックしてみましょう。^{*8}

^{*8} もし Auto Scaling の設定に失敗していて自動復旧しなくても、手動で AMI からインスタンスを作り直せば復旧できるので大丈夫です。



▲図 8.50 「インスタンスの状態」から「削除」をクリック

恐ろしい警告が表示（図 8.51）されますが「はい、削除する」をクリックします。



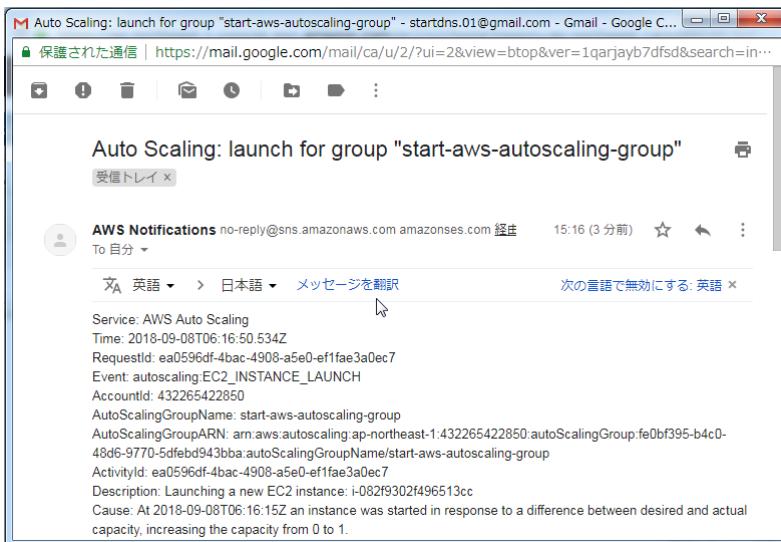
▲図 8.51 「はい、削除する」をクリック

インスタンスを削除すると状態がまず「shutting-down」に変わります。再びブラウザで「<http://www.自分のドメイン名/>」を見てみましょう。サイトがウェブサーバごといなくなってしまったので「503 Service Temporarily Unavailable」と表示（図 8.52）されています。



▲図 8.52 再び `http://www.自分のドメイン名/` を開くと「503 Service Temporarily Unavailable」と表示される

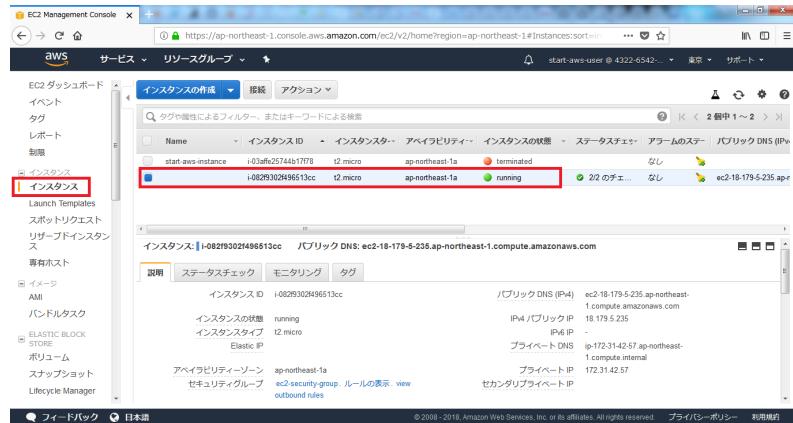
停止から1分後、「Auto Scaling: launch for group "start-aws-autoscaling-group"」という件名のメール（図8.53）が届きました。どうやらAuto ScalingによってAMIからインスタンスが生成されたようです。



▲図 8.53 Auto Scalingによるインスタンスの追加を知らせるメールが届いた

左メニューで「インスタンス」をクリックすると、先ほど削除したインスタンスの状態

は「terminated」*9になり、その下に新たなインスタンスが表示（図 8.54）されています。



▲図 8.54 先ほど削除したインスタンスの下に新たなインスタンスが表示されている

停止から 3 分後、もう一度ブラウザで「<http://www.自分のドメイン名/>」を開いてみると、見事にサイトは復旧していました。



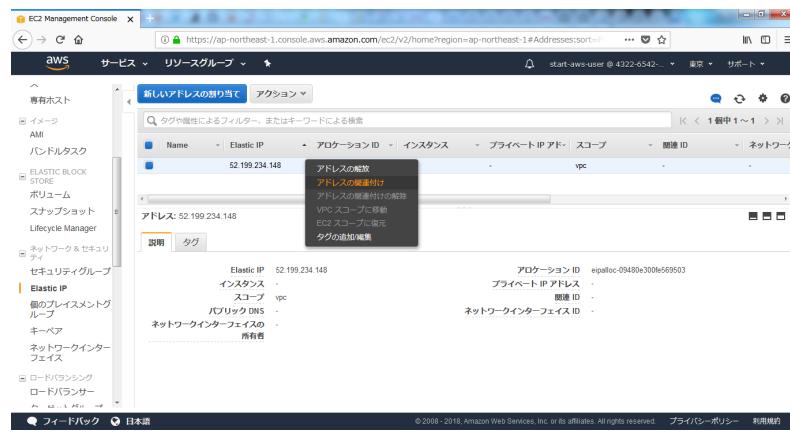
▲図 8.55 もう一度 <http://www.自分のドメイン名/> を開くとサイトが復旧していた！

このようにインスタンスを削除したことで一時的に WordPress のサイトが見られなくなりましたが、すぐに Auto Scaling によってインスタンスが生成されサイトも自動復旧

*9 インスタンスの状態が「terminated」になってからしばらくすると一覧に表示されなくなります。

しました。このとき記事データはRDS(データベースサーバ)に、画像はS3にあるので何もかも元通りに表示されます。

但しAuto Scalingでインスタンスが自動生成された場合、SSHログイン時に使っていたElastic IPだけは新しいインスタンスに自動で紐づきません。Elastic IPは元々紐づいていたインスタンスを失って宙ぶらりんな状態になっています。左メニューの「Elastic IP」をクリックして、新しいEC2インスタンスに「アドレスの関連付け」をしてやれば再びSSHログインできるようになります。



▲図8.56 新しいEC2インスタンスに宙ぶらりんなElastic IPの「アドレスの関連付け」をしてあげよう

ウェブサーバを削除しても数分で何事もなかったかのように自動で直るなんてすごいですか？これでAuto ScalingでEC2インスタンスを自動復旧させる設定はおしまいです。

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 1 年間の無料利用期間が終わりに近づいて、AWS の利用をやめたくなったらしておきべき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

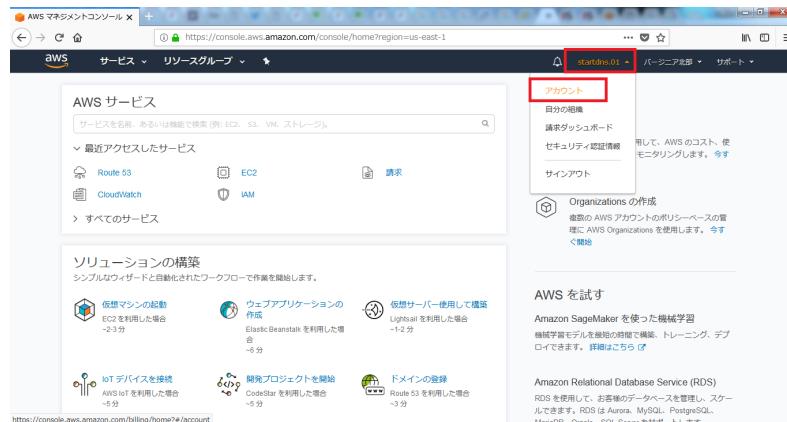
第 2 章「AWS を使い始めたら最初にやること」で書いたとおり、AWS アカウントを作成してから 1 年間は無料利用枠の範囲内であれば利用料が無料となります。

たとえ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。〆切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:

- 茹でたプロッコリーが好きな茶色い折れ耳の猫に捧ぐ

レビュアー

- Takeshi Matsuba
- 深澤俊

参考ウェブサイト

- Developers.IO <https://dev.classmethod.jp/>

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