

SSL をはじめよう

証明書の発行からトラブルシューティングまで

mochikoAsTech 著

2020-03-01 版 mochikoAsTech 発行

はじめに

2020 年 2 月 mochikoAsTech

この本を手に取ってくださったあなた、はじめてまして。「SSL をはじめよう」の筆者、mochikoAsTech 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。

- よく分からぬいけど言われるがままに SSL の設定をしている人
- SSL と TLS の関係性がよく分かっていない人
- SSL 証明書がいったい何を証明しているのか知らない人
- SSL は聞いたことがあるけど TLS は知らないという人
- 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
- 「インフラがよく分からぬこと」にコンプレックスのある人
- 証明書の購入や設置はしたことがあるけど SSL はあまり分かっていない人
- サイトを HTTPS 化しなきゃ！ と思っている人

マッチしない読者層

本著は、こんな人が読むと恐らく「not for me だった…（私向けじゃなかった）」となります。

- SSL/TLS の通信を C 言語で実装したい人
- 「プロフェッショナル SSL/TLS」を読んで完全に理解できた人

本著の特徴

本著では実際にサーバを立てて SSL 証明書の設置を行い、HTTPS のサイトを作つてみます。手を動かして試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、

- こんな障害が起きたら原因はどう調べたらいいのか？
- 問題をどう解決したらいいのか？
- どうしたら事前に避けられるのか？

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

本著のゴール

本著を読み終わると、あなたはこのような状態になっています。

- SSL 証明書がどんな役割を果たしているのか説明できる
- 証明書を買うときは何に注意してどんな手順で買ったらいいか分かっている
- 意図せず「保護されていない通信」と表示されてしまったときの対処法が分かる
- 障害が起きたときに原因を調査できる
- 読む前より SSL が好きになっている
- SSL/TLS と併記されている「TLS」の意味が分かっている

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 Issue や Pull request で筆者までお知らせいただけますと幸いです。

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

目次

はじめに	3
想定する読者層	3
マッチしない読者層	3
本著の特徴	4
本著のゴール	4
免責事項	4
第1章 Oracle Cloud のアカウントを作ろう	9
1.1 ウェブサーバを立てよう	10
1.1.1 サイトを作るのにどうしてサーバがいるの？	10
1.1.2 サーバを立てるにはお金が必要？	11
1.1.3 なんで AWS じゃなくて Oracle のクラウドを使うの？	11
1.2 Oracle Cloud でアカウント登録	12
1.2.1 無料でアカウントを作成	12
【コラム】どうしても SMS が届かない！ そんなときは？	18
1.2.2 Oracle Cloud のコンソールにサインイン	23
第2章 Oracle Cloud でサーバを立てよう	27
2.1 事前準備	28
2.1.1 お使いのパソコンが Windows の場合	28
【コラム】パスフレーズは設定すべき？ しなくてもいい？	37
2.1.2 お使いのパソコンが Mac の場合	38
【コラム】ターミナルでコピー＆ペーストするには？	40
2.2 コンピュートでサーバを立てる	41
【コラム】"Out of host capacity."が起きたらどうすればいい？	43
2.2.1 Always Free ではなく無償クレジットの枠でサーバを立てる	44

2.2.2 サーバが起動するまで待とう	46
【コラム】Oracle Cloud のコンピュートの金額計算方法	47
【コラム】Oracle Cloud と AWS はどっちが安い?	48
2.2.3 接続先となるサーバの IP アドレス	49
2.3 SSH でサーバに入ってみよう	49
2.3.1 お使いのパソコンが Windows の場合	49
2.3.2 お使いのパソコンが Mac の場合	59
2.4 ターミナルでサーバを操作・設定してみよう	62
2.4.1 ターミナルの基本操作に慣れよう	62
2.4.2 ミドルウェアをインストール	64
2.4.3 OS の基本設定をしておこう	66
2.4.4 ターミナルはなんのためにある?	75
2.5 ドメイン名の設定	76
2.6 まずは HTTP でサイトを公開	76
2.7 証明書を取得しよう	76
2.7.1 密密鍵を作ろう	76
2.7.2 CSR を作ろう	76
2.7.3 証明書の取得申請	76
2.7.4 取得した証明書をサーバに置こう	76
2.8 HTTPS でサイトを公開	76
第 3 章 基本	77
3.1 SSL ってなに?	78
3.2 TLS ってなに?	78
3.3 SSL と TLS の違いは?	78
3.4 SSL と SSH って似てる? 何が違うの?	78
3.5 HTTPS で始まるページで鍵のマークが壊れて表示された	78
3.6 種類	78
3.6.1 SSL サーバ証明書	78
3.6.2 SSL クライアント証明書	78
3.7 どんなシーンで使われている?	78
3.8 SSL 証明書は全然違う 2 種類の仕事をしている	78
3.8.1 Web サイトで送受信する情報を暗号化すること	78
3.8.2 Web サイト運営者の身元を証明すること	78
3.9 鍵マークが壊れるケース	78

3.9.1	すべて HTTP で通信しているとき	78
3.9.2	HTTPS だけど一部が HTTPS じゃないとき	78
3.10	ウェブページが表示されるまで	78
3.10.1	1 往復で表示されるわけじゃない	78
3.11	SSL 証明書は何を証明してくれるのか?	78
3.11.1	ネットバンクの事例	78
3.12	認証局事業者の身元は誰が証明する?	78
3.12.1	身元保証の連鎖をつなぐ中間 CA 証明書とルート証明書	78
3.13	SSL 証明書はどうしてあんなに値段に差があるの?	78
3.14	同じ「SSL 証明書」という名前でも 3 つの種類がある	78
3.14.1	EV 証明書	78
3.14.2	OV 証明書	78
3.14.3	DV 証明書	78
3.14.4	3 つの違いは何か?	78
3.14.5	ブラウザベンダーによる EV 証明書の扱いの変化	78
3.15	その他の証明書	78
3.15.1	中間証明書	78
3.15.2	クロスルート証明書	78
3.16	どの証明書を買えばいい?	78
3.16.1	ワイルドカード証明書	78
3.16.2	www ありにリダイレクトしたいだけなのに www なしの証明書 もいるの?	78
3.16.3	コモンネームが*.example.com の証明書は example.com で使 える?	78
3.16.4	Let'sEncrypt	78
3.17	CDN と証明書	78
3.17.1	CDN を使ったら古い端末でサイトが見られなくなった	78
3.17.2	同じサーバで複数サイトを HTTPS 化したら古い端末で別サイ トが表示された	78
3.17.3	SNI Server Name Indication	78
あとがき		79
PDF 版のダウンロード		79
Special Thanks:		79
レビュアー		79

目次

参考文献	79
著者紹介	81

第 1 章

Oracle Cloud のアカウントを作 ろう

この章では Oracle Cloud というクラウドでアカウントを作ります。

SSL を理解するには、実際に手を動かしてやってみるのがいちばんです。実際に SSL 証明書を取得して、HTTPS のサイトを作ってみましょう。

HTTPS でサイトを作るのに必要な材料は次の 3 つです。

- ウェブサーバ
- ドメイン名
- SSL 証明書

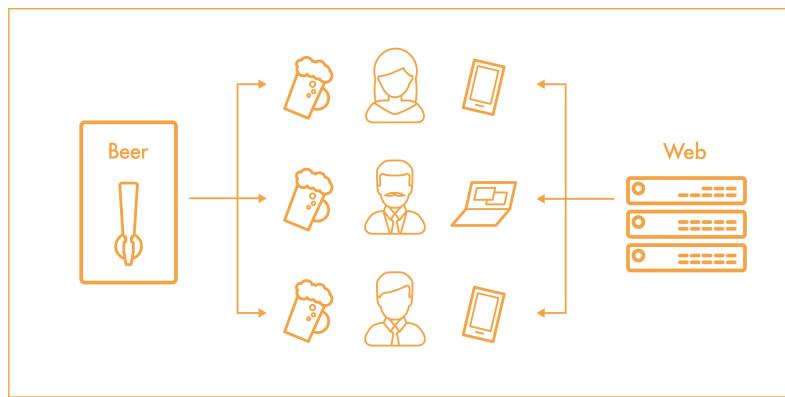
まずは 1 つめのウェブサーバを立てるため、アカウント作成から始めましょう。

1.1 ウェブサーバを立てよう

1.1.1 サイトを作るのにどうしてサーバがいるの？

これからウェブサーバを立てますが…どうしてサイトを作りたいだけなのに、ウェブサーバが必要なのでしょう？

そもそもですが、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です**。居酒屋にあるビアサーバに「ビールをください」というリクエストを投げる…つまりコックを「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。同様にあなたがブラウザでURLを入力したり、リンクをクリックしたりして、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たら、ウェブページというレスポンスが返ってきます。（図1.1）



▲図1.1 ビアサーバもウェブサーバもリクエストしたらサービスが提供される

つまり、せっかくHTMLや画像でサイトのコンテンツを作っても、それを載せておくウェブサーバがなければ、サイトはあなたのパソコンの中でしか見られず、インターネットで公開できないのです。^{*1}

というわけでウェブサイトを提供するために、まずはウェブサーバを立てましょう！

^{*1} サーバについては、はじめようシリーズの2冊目、「AWSをはじめよう」の「CHAPTER1 インフラとサーバってなに？」で、より詳しく解説しています。仮想サーバと物理サーバ、クラウドとオンプレミス、ホストサーバとゲストサーバなどサーバ周りの用語をもう少し理解したい！という方はそちらも併せて読んでみるのがお勧めです

1.1.2 サーバを立てるにはお金が必要？

ウェブサイトを作るにはサーバが必要です。そしてサーバを立てるには、普通はお金がかかります。ですがオラクルがやっている「Oracle Cloud（オラクル クラウド）」というサービスなら、なんと有効期限なしでずっと無料で使える「Always Free」という枠があります。「Always Free」の範囲内であれば、サーバも無料で立てて使えるので今回はそれを使いましょう。

オラクルがやっているクラウド、と言われても、そもそもクラウドがなんだか分からないといまいちピンと来ないかもしれません。あなたが「ウェブサイト作りたいなあ…だからサーバが必要だ！」と思ったとき、**自分でサーバを買って自分で管理しなければいけないのがオンプレミスで、従量課金ですぐに使って性能や台数の増減も簡単にできるのがクラウドです。**

Oracle Cloud とはオラクルがやっているクラウドなので、ブラウザでぽちぽちとスペックを選んでいくだけで、すぐにサーバが使えます。

1.1.3 なんで AWS じゃなくて Oracle のクラウドを使うの？

クラウドは Oracle Cloud だけではありません。かの有名な AWS こと Amazon Web Services や、Google の Google Cloud Platform^{*2}、Microsoft の Azure（アジュール）^{*3}、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^{*4}、お名前.com でお馴染み GMO グループの GMO クラウド^{*5}などたくさんあります。

2019年11月時点、クラウド市場では AWS がシェア約40%でトップを独走中^{*6}です。そのため仕事で AWS を使ったことがある、あるいはこれから使う予定だ、というエンジニアも多いと思います。

しかし最近は、Alibaba Cloud や Tencent Cloud といった中国のクラウド事業者も追い上げを見せています。こうした新興のクラウドは、先に行く AWS を見て学んだ上で生まれてきているだけあって、よりスマートな作りになっているのがいいところです。

たくさんのクラウドがある中でどこを選ぶのか、その理由は、本来であれば使う人やそ

^{*2} <https://cloud.google.com/>

^{*3} <https://azure.microsoft.com/ja-jp/>

^{*4} <https://cloud.sakura.ad.jp/>

^{*5} <https://www.gmocloud.com/>

^{*6} IaaS + PaaS クラウド市場、AWS の首位ゆるがず。AWS、Azure、Google、Alibaba の上位4社で市場の7割超。2019年第3四半期、Synergy Research Group — Publickey https://www.publickey1.jp/blog/19/iaaspasawsazuregooglealibaba4720193synergy_research_group.html

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は、いったいどのクラウドが適しているのでしょうか？

本著では以下を目的としていますので、それに適した Oracle Cloud で学びを進めていきたいと思います。

- SSL 証明書を自分で取得して設置する一通りの流れを試したい
- お金をかけずに無料で試したい

1.2 Oracle Cloud でアカウント登録

先ずは Oracle Cloud のアカウントを作りますので次の 2 つを用意してください。

- クレジットカード
- SMS 受信が可能な携帯電話（電話番号認証で使用するため）^{*7}

なお Oracle Cloud を利用する際は、前述のとおり Always Free という無料枠^{*8}があります。

1.2.1 無料でアカウントを作成

「Oracle Cloud 無料」で検索（図 1.2）したら、いちばん上の [Oracle Cloud Free Tier | Oracle 日本]^{*9}をクリックします。

^{*7} ショートメッセージサービスの略。宛先に電話番号を指定してメッセージを送れるサービス

^{*8} 期限なしでずっと無料ですが、無料で利用できる範囲は決まっていて、何をどれだけ使っても無料という訳ではありませんので注意してください。Always Free の他に、30 日間だけ有効な 300 ドル分の無償クレジットも付いてきますので、Always Free の範囲外のサービスはそちらで試せます。詳細は <https://www.oracle.com/jp/cloud/free/> を確認してください

^{*9} <https://www.oracle.com/jp/cloud/free/>

1.2 Oracle Cloud でアカウント登録



▲図 1.2 「Oracle Cloud 無料」で検索

[今すぐ始める（無償）] をクリックします。（図 1.3）



▲図 1.3 [今すぐ始める（無償）] をクリック

「Oracle Cloud へのサインアップ」と表示されました。それでは次の情報を入力して、使用条件を確認した上で【次】をクリックしましょう。(図 1.4) 後で分からなくならないように、登録した項目をメモしておきましょう。(表 1.1)



▲図 1.4 入力したら【次】をクリック

▼表 1.1 Oracle Cloud に登録した情報

項目	例	あなたが登録した情報
電子メール・アドレス	startdns.01@gmail.com	
国/地域	日本	

次は「アカウント詳細の入力」です。(表 1.2) 今回は仕事ではなく個人での利用ですので【アカウント・タイプ】は【個人使用】を選択してください。【クラウド・アカウント名】には任意のアカウント名を入力します。【クラウド・アカウント名】には英字小文字と数字のみ使えます。記号や英字大文字は使えないで注意してください。筆者は startdns01 にしました。この【クラウド・アカウント名】は、後で管理画面にサインインするときのアカウント URL になります。(図 1.5)

【ホーム・リージョン】は【日本東部(東京)】を選択してください。Oracle Cloud は世界の各地域にデータセンターを所有しており、サーバはそのデータセンターの中で元気に動いています。この【ホーム・リージョン】とは、**各地域の中でどこを使うか？を指定するものです。** ウェブサイトにアクセスするとき、パソコンのある場所からサーバまで物理

的に距離が遠いと、それだけ通信にも時間がかかるて応答時間も遅くなりますので、日本国内向けにウェブサイトを開設する場合は基本的にこの「東京リージョン」を選びましょう。ただし Oracle Cloud 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(アッシュバーン)」を選択してください。

▼表 1.2 Oracle Cloud に登録した情報

項目	例	あなたが登録した情報
アカウント・タイプ	個人使用	-
クラウド・アカウント名	startdns01	
ホーム・リージョン	日本東部(東京)	-

▲図 1.5 [クラウド・アカウント名] には好きな名前を入力

続いて名前や住所を入力していきます。入力内容は日本語表記で構いません。個人利用なのですが「部門名」が必須であるため、ここでは「個人」と入力しておきましょう。[名]・[姓]・[部門名]・[住所]・[市区町村]・[都道府県]・[郵便番号]をすべて入力できましたか？（図 1.6）

The screenshot shows a form for creating an Oracle Cloud account. The address input section is highlighted with red boxes around the following fields: Name (名), Surname (姓), Department (部門名), Position (役職), Address (住所), City/Town/Village (市区町村), Prefecture/Province (都道府県), and Zip Code (郵便番号). The 'Address' field contains '新宿四丁目1番6号' and 'JR新宿ミライナタワー'. The 'City/Town/Village' field contains '新宿区' and the 'Prefecture/Province' field contains 'TOKYO'.

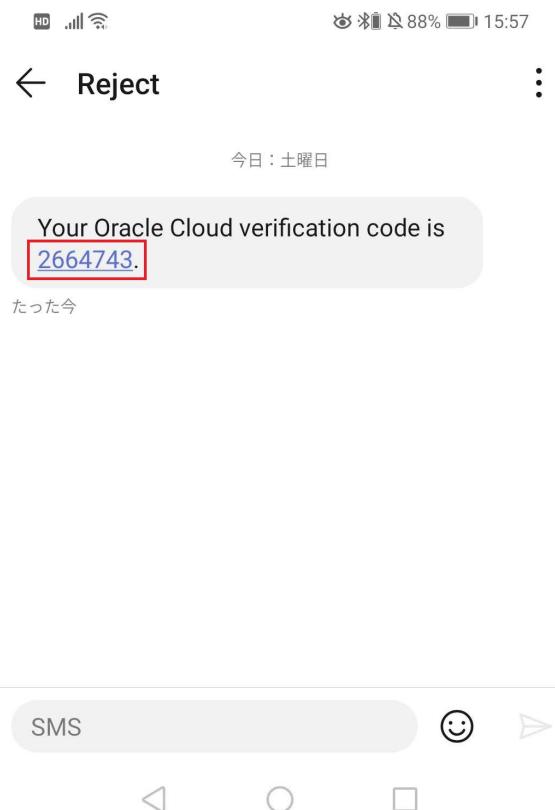
▲図 1.6 名前や住所を入力

では最後に「モバイル番号」です。国番号は「日本(81)」を選択して、自分の携帯電話番号を入力します。このとき電話番号の先頭の 0 は不要です。例えば「090-〇〇〇〇-〇〇〇〇」という携帯電話番号であれば「90-〇〇〇〇-〇〇〇〇」と入力してください。携帯電話番号を入力したら【次: モバイル番号の確認】をクリックしてください。(図 1.7)

The screenshot shows the 'Mobile Phone Number Verification' step. It displays the selected country code '日本 (81)' and the entered mobile number '90 [REDACTED]'. A note below explains that the first digit '0' is not required for Japanese numbers. A green checkmark icon is visible next to the mobile number input field. Below the input fields, there is a button labeled '次: モバイル番号の確認' (Next: Mobile phone number confirmation).

▲図 1.7 携帯電話の番号を入力

数分以内に [Your Oracle Cloud verification code is ○○○○○○○○.] と書かれた SMS が届きます。(図 1.8)



▲図 1.8 コードの書かれた SMS が届いた

SMS で届いた「〇〇〇〇〇〇〇」の数字を [コード] に入力して、[コードの確認] をクリックします。(図 1.9)



▲図 1.9 SMS で届いた数字を [コード] に入力して [コードの確認] をクリック

【コラム】どうしても SMS が届かない！ そんなときは？

電話番号を入力したのに SMS が届かないときは、まず自分が契約している携帯キャリアの迷惑メール設定で、SMS をスパムとしてはじく設定をしていないか確認してみましょう。たとえば海外の事業者から送信された SMS を拒否する設定になっていたり、海外からの着信を拒否する設定になっていると、SMS が届かないことがあるようです。^{*10}

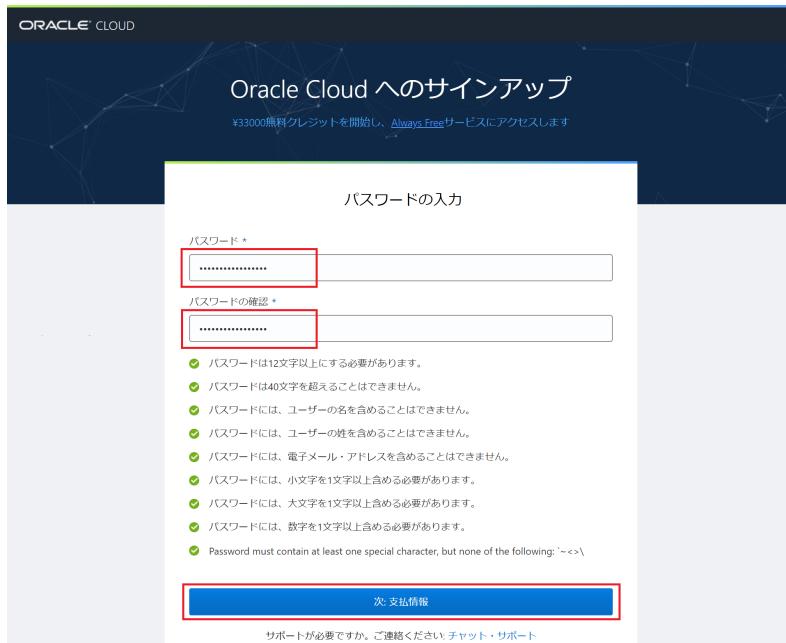
ちなみに筆者の場合は、特に設定変更をせず同じ番号で 2 回試してみたのですが、1 回目は届かず、もう 1 回試してようやく届きました。

迷惑メールの設定を確認して何回か試して、それでも SMS が届かなかったら、ページ下部の「[サポートが必要ですか。ご連絡ください: チャット・サポート]」からサポートにチャットで問い合わせてみましょう。

残念ながら英語でしか対応してもらえませんが、「I am trying to register on Oracle Cloud. But I can't receive SMS. What should I do?」（アカウント登録

しようとしてるけど SMS が届かないの、どうしたらいい?) という感じで聞いてみると、「じゃあ登録情報をこのチャットで教えて。そうしたらこちらでコードを発行して、チャットで伝えてあげる」(意訳) という感じでサポートしてもらえます。

正しいコードが入力できたら、[パスワードの入力] と表示されます。[パスワード] と [パスワードの確認] を入力して、[次: 支払情報] をクリックします。(図 1.10)



▲図 1.10 パスワードを入力して [次: 支払情報] をクリック

パスワードを入力すると、今度は [支払情報] のページが表示されます。(図 1.11) 繰り返しあ伝えしているとおり、Oracle Cloud には Always Free という無料枠があり、本書ではその無料枠の範囲内で Oracle Cloud を使っていくつもりですが、それでもクレジットカードは登録しておく必要があります。記載されているとおり、この後、管理画面で [アカウントのアップグレード] という作業をしない限り、請求は発生しませんので安

*10 「Oracle Cloud の SMS は海外の事業者から届く」という確証がある訳ではないです。あくまで SMS が届かないときによくある話と思ってください

心してカード情報を登録してください。[クレジット・カード詳細の追加] をクリックします。



▲図 1.11 [クレジット・カード詳細の追加] をクリック

[ご注文者様情報] はそのまま変更不要です。[カード情報] の [カードの種類] を選択し、[カードの番号]・[有効期限]・[CVN] を入力したら [Finish] をクリックします。
(図 1.12)*¹¹

*¹¹ Oracle Cloud では、クレジットカード登録時に「1 ドル認証」と呼ばれる認証方法で、そのクレジットカードが決済可能かをチェックしています。クレジットカードによってはこの 1 ドル認証を不審な決済と判断して通さないため、それによってエラーが発生することがあります。その場合は別のクレジットカードで試すか、Oracle Cloud のチャット・サポートで問い合わせてみてください

The screenshot shows a 'Card Information' form. At the top, there's a heading 'カード情報' with a small orange info icon. Below it, a section for 'Card Type' has four options: Visa, Mastercard, Amex, and JCB. The Visa and Mastercard options are highlighted with a red box. The 'Card Number' field and the 'Expiration Date' dropdown are also highlighted with red boxes. The 'CVN' field is highlighted with a red box. A note below the CVN field explains that it is the 3 or 4-digit number printed on the back of the card. A green 'Finish' button at the bottom right is also highlighted with a red box.

▲図 1.12 カード情報を入力して [Finish]

[クレジット・カード詳細をご提供いただきありがとうございます。] と表示（図 1.13）されたら、支払い情報の登録は完了です。Oracle Cloud の Service Agreement^{*12}を確認した上で、チェックボックスにチェックを入れて、[サインアップの完了] をクリックします。

*12 <https://www.oracle.com/goto/oraclecsa-jp-en>

第1章 Oracle Cloud のアカウントを作ろう



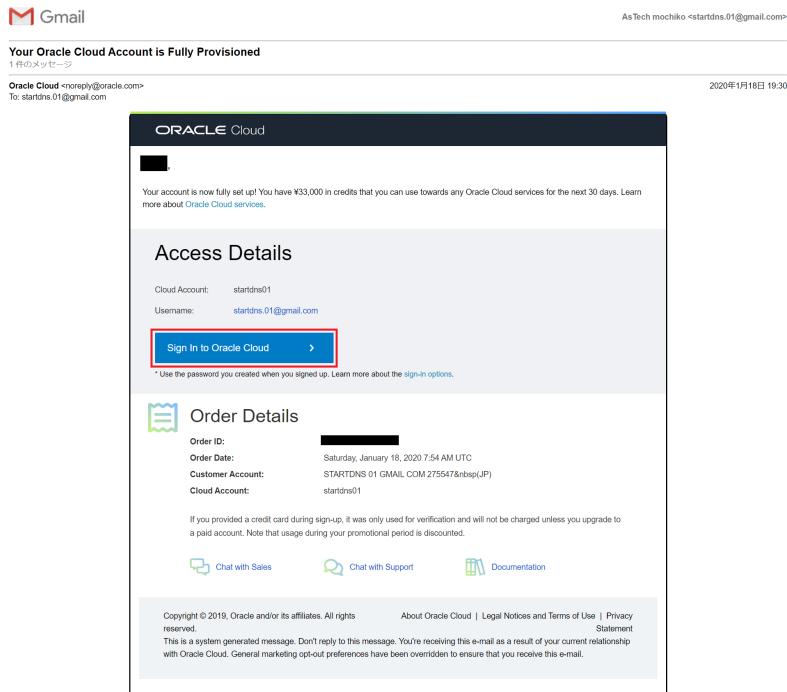
▲図 1.13 チェックを入れて [サインアップの完了] をクリック

これでアカウント登録の手続きはおしまいです。[アカウントの設定が完了するまでお待ちください。] と表示（図 1.14）されます。準備が整うとサインイン画面にリダイレクトされますが、この [アカウントの設定が完了するまでお待ちください。] の画面でかなり時間がかかるので一度ブラウザを閉じてしまって構いません。頑張った自分を褒めて一旦休みましょう。



▲図 1.14 アカウント登録の手続きはおしまい

数時間後^{*13}、[Your Oracle Cloud Account is Fully Provisioned] という件名で、準備完了を知らせるメールが届きます。メールの [Sign In to Oracle Cloud] をクリックしましょう。(図 1.15)



▲図 1.15 準備完了を知らせるメールが届いた

1.2.2 Oracle Cloud のコンソールにサインイン

メールの [Sign In to Oracle Cloud] をクリックすると、コンソールへのサインイン^{*14}画面が表示されます。(図 1.16) [ユーザー名] には先ほど登録したメールアドレスを入力します。^{*15} [パスワード] を入力して、[サイン・イン] をクリックしてください。

^{*13} 筆者の場合は、メールが届くまで 2 時間半かかりました

^{*14} 日本語だとログインの方が馴染みがあるかも知れませんが、サインインはログインと同じ意味です。

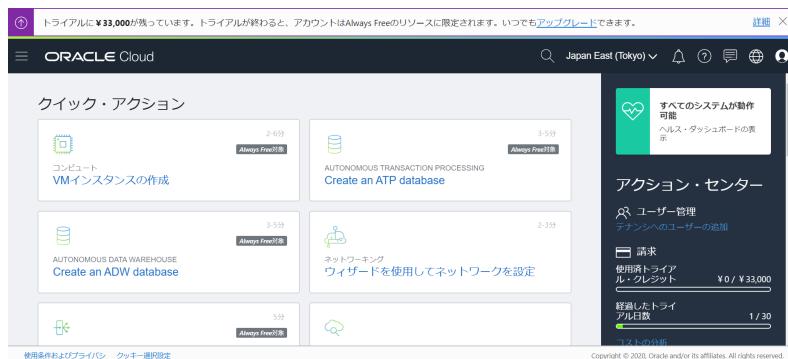
^{*15} メールにも書いてありますが、ここでの [ユーザー名] とは [クラウド・アカウント名] (筆者の場合は startdns01) ではなく、[メールアドレス] のことです。紛らわしいのでご注意ください

第1章 Oracle Cloud のアカウントを作ろう



▲図 1.16 [ユーザー名] と [パスワード] を入力して [サイン・イン]

おめでとうございます！ これでコンソールにサインインできました。



▲図 1.17 コンソールにサインインできた！

なお今後、コンソールにサインインしたくなったら、いちいちメールを探してリンクを踏む必要はありません。まずは Oracle のトップページ^{*16}を開いて、右上の人物マークから [クラウドにサインイン] をクリックしましょう。

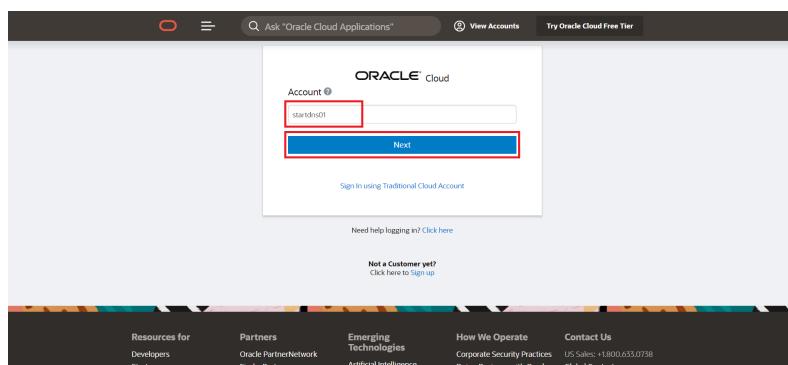
*16 <https://www.oracle.com/jp/>

1.2 Oracle Cloud でアカウント登録



▲図 1.18 右上の人マークから [クラウドにサインイン] をクリック

サインインのページ^{*17}で [Account] の欄にクラウド・アカウント名^{*18}を入力して [Next] をクリックすれば、メールのリンクを踏んだときと同じ [サイン・イン] のページにたどり着けます。あとは同じように [ユーザー名] にはメールアドレスを、[パスワード] にはパスワードを入力して、[サイン・イン] をクリックするだけです。



▲図 1.19 [Account] の欄にクラウド・アカウント名を入力して [Next] をクリック

*17 <https://www.oracle.com/cloud/sign-in.html>

*18 筆者の場合は startdns01 です。アカウント登録時に、あなたの [クラウド・アカウント名] をメモしているはずですでの、数ページ戻って確認してみましょう

第2章

Oracle Cloud でサーバを立てよう

この章では実際に Oracle Cloud でサーバを立てます。
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！

2.1 事前準備

2.1.1 お使いのパソコンが Windows の場合

RLogin のインストール

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 2.1 RLogin

本著ではいちばん上の RLogin (図 2.1) 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 2.2) の URL、http://nanno.dip.jp/softlib/program/rlogin_x64.zip をクリックしてください。

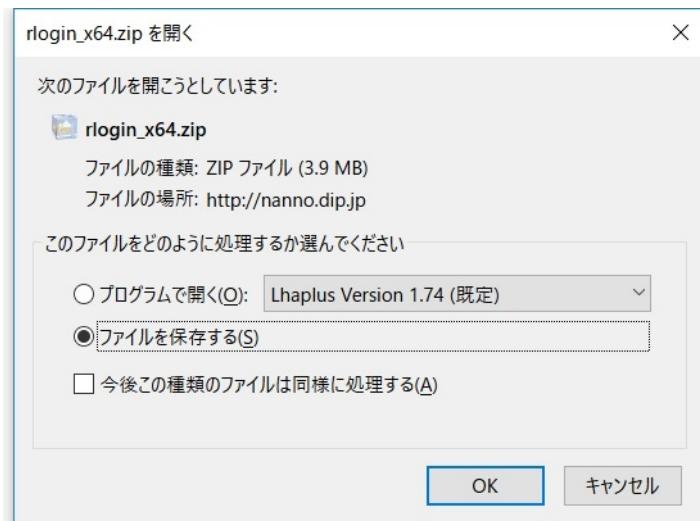
^{*1} PuTTYjp を使う場合、.pem の秘密鍵を PuTTYgen で.ppk に変換する必要が出てくるため、他のターミナルソフトに比べると一手間余計にかかります。

^{*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 2.2 「実行プログラム(64bit)」の URL をクリックしてダウンロード

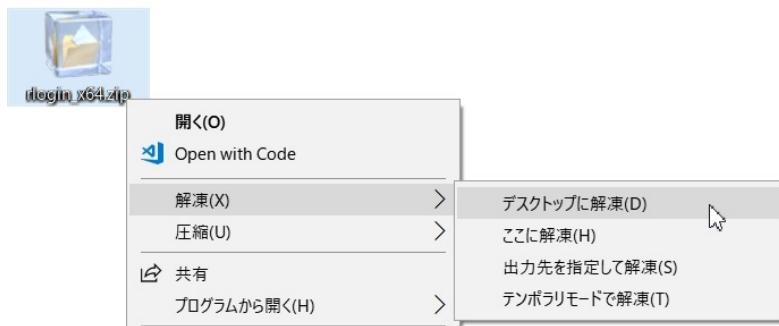
ダウンロードした ZIP ファイルを保存（図 2.3）します。保存場所はどこでも構いませんが、後でどこに置いたか分からなくなりそうな人はデスクトップに保存しておきましょう。



▲図 2.3 「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 2.4）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/>



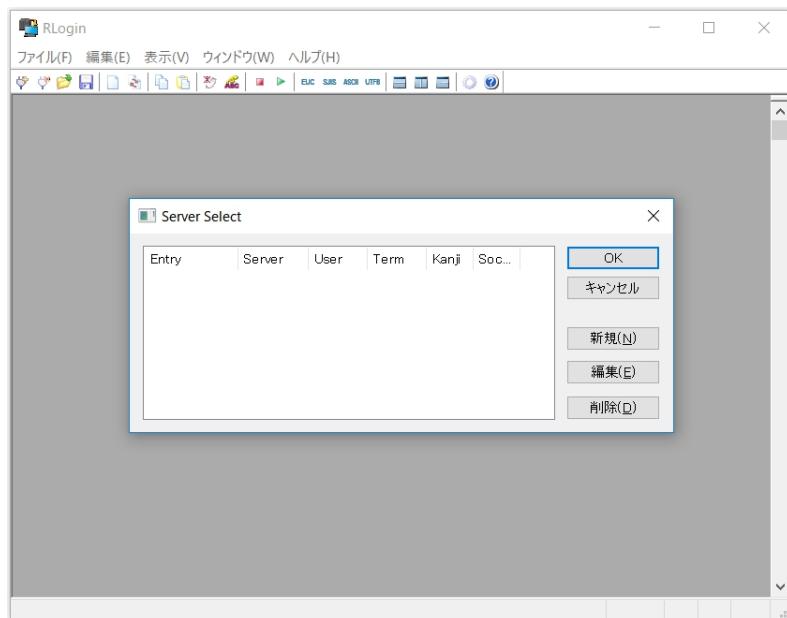
▲図 2.4 ZIP ファイルを右クリックして解凍>デスクトップに解凍

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 2.5）をダブルクリックすれば RLogin が起動（図 2.6）します。



▲図 2.5 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。

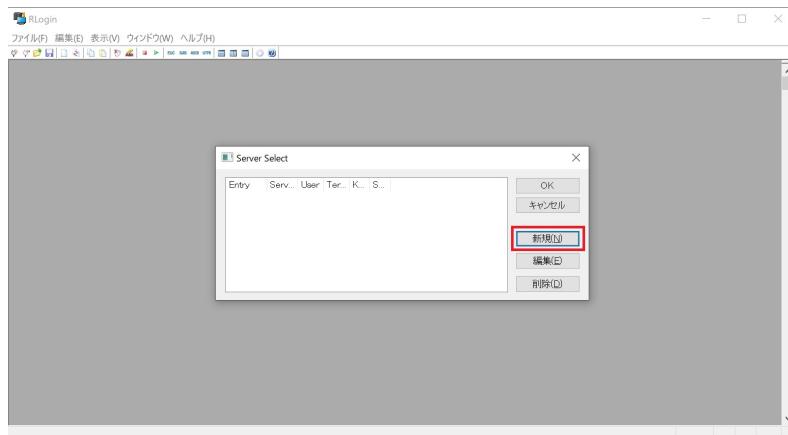


▲図 2.6 RLogin が起動した

これで RLogin のインストールは完了です。

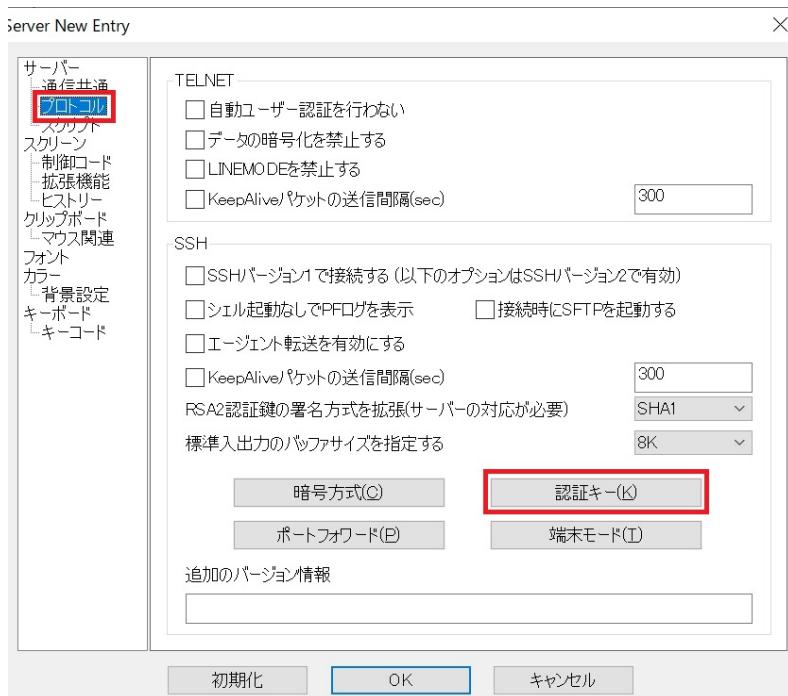
Windows で SSH のキーペア（秘密鍵・公開鍵）を作成する

Windows の方は、起動した RLogin で [新規 (N)] をクリックします。



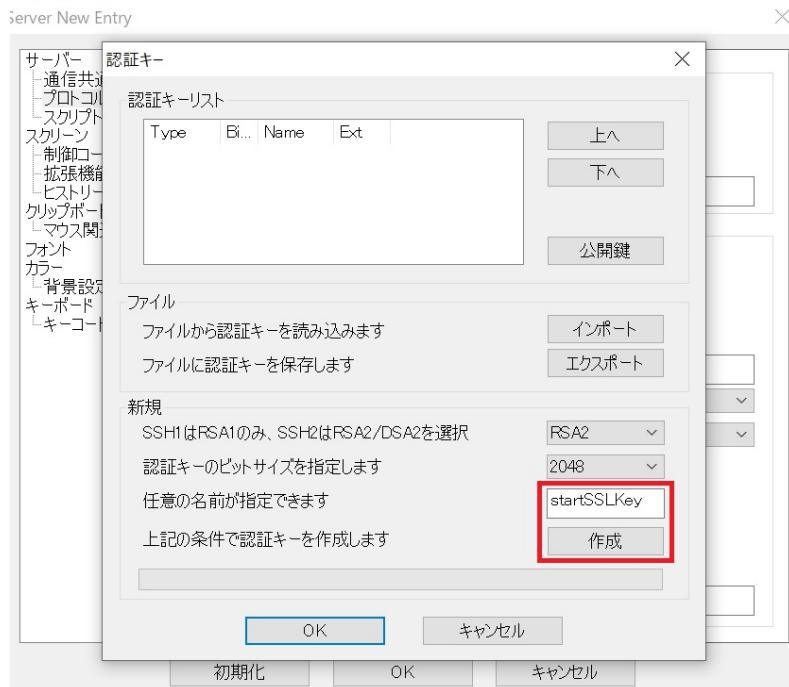
▲図 2.7 [新規 (N)] をクリック

左メニューの [プロトコル] を選択して、[認証キー (K)] をクリックします。



▲図 2.8 [プロトコル] を選択して [認証キー (K)] をクリック

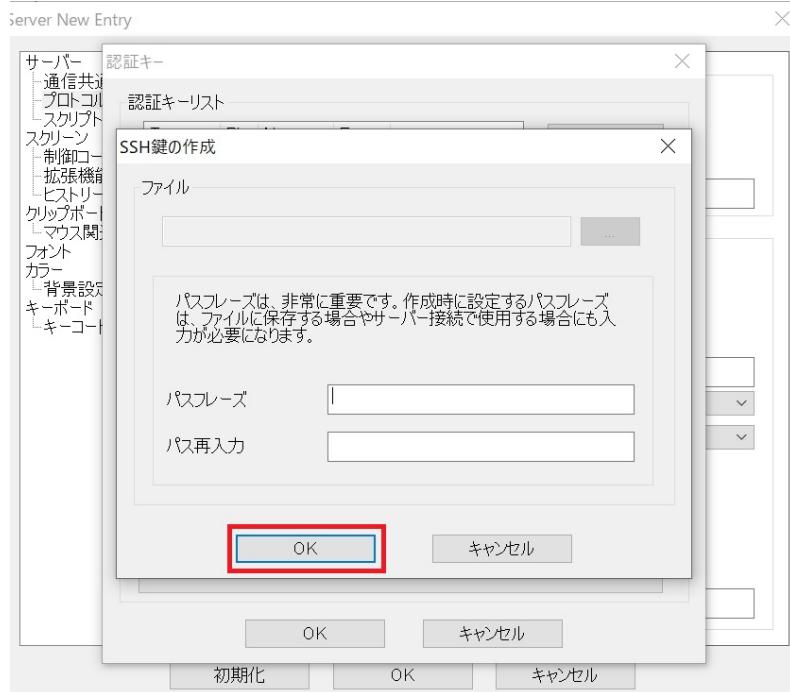
[任意の名前が指定できます] に [startSSLKey] を入力して、[作成] をクリックします。



▲図 2.9 [startSSLKey] を入力して [作成] をクリック

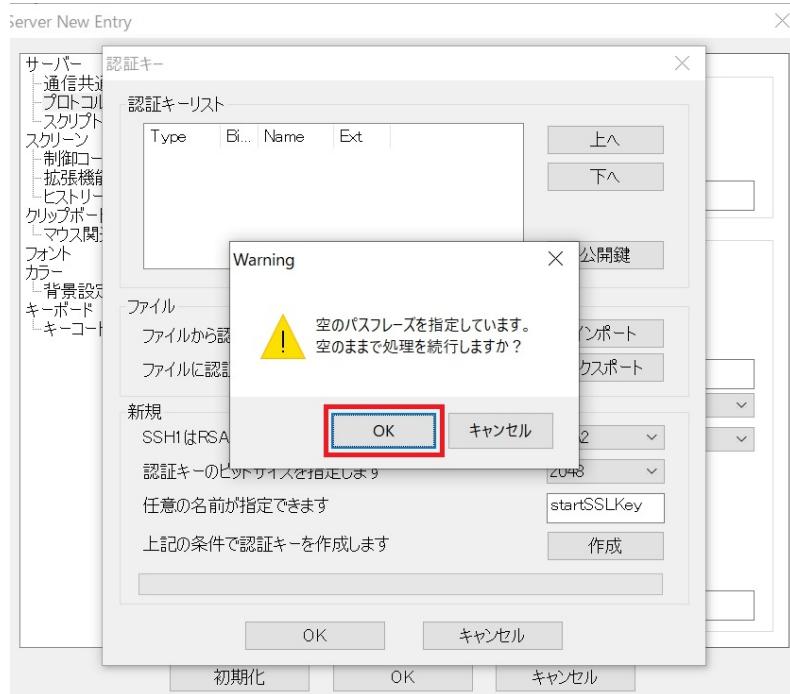
[パスフレーズ] と [パス再入力] には何も入力せず、[OK] をクリックします。^{*5}

^{*5} 「p@\$sw0rd」や「@dm1ni\$strat0r」のように、ひとつの単語でできているのがパスワードです。それに対して「This 1s P@ss\$ Phrase.」のように空白を挟んだ文章（フレーズ）で構成されているのものをパスフレーズと呼びます



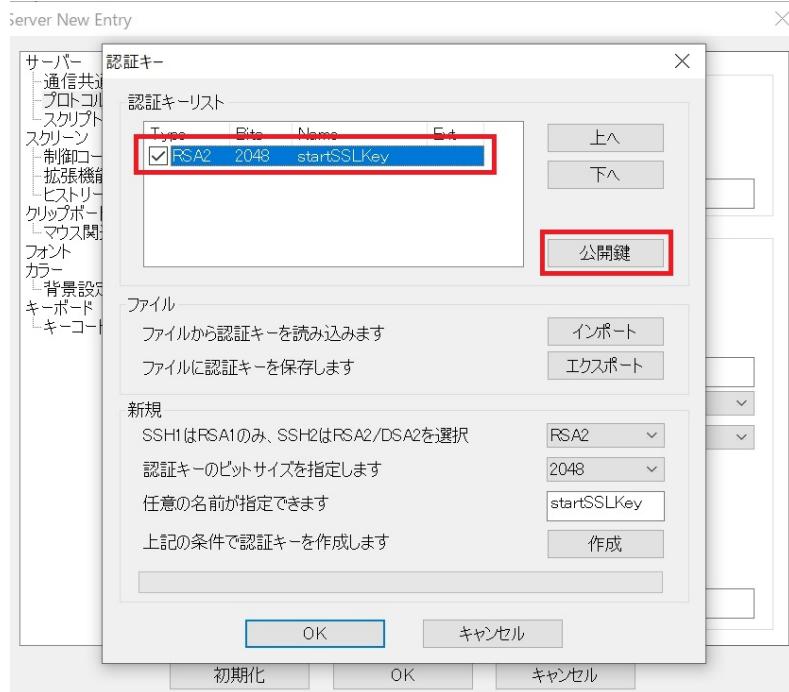
▲図 2.10 何も入力せず [OK] をクリック

[空のパスフレーズを指定しています。空のままで処理を続行しますか？] と表示されますが、そのまま [OK] をクリックします。



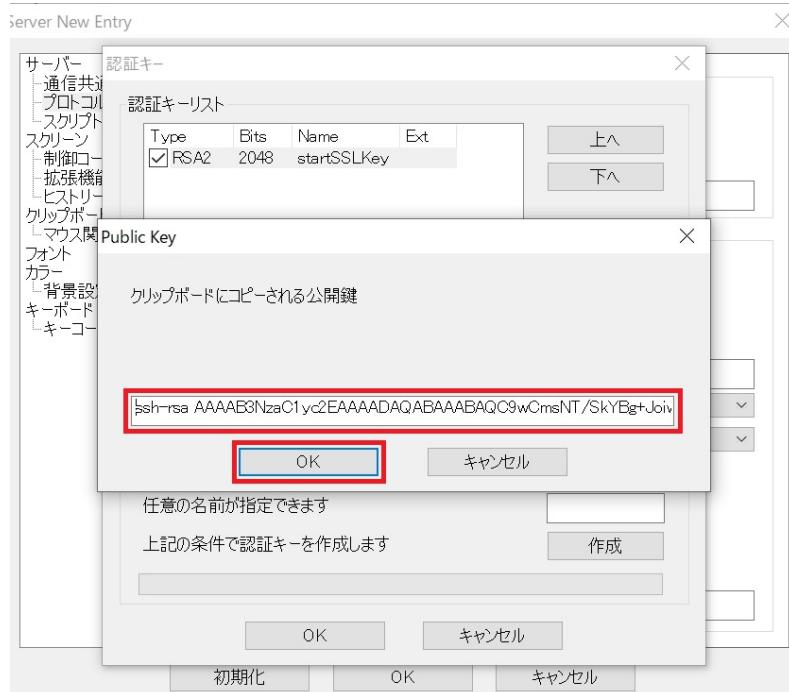
▲図 2.11 [OK] をクリック

[認証キーリスト] に、今作った [startSSLKey] が表示されたら、キーペア（秘密鍵・公開鍵）が無事できています。[公開鍵] をクリックしてください。（図 2.5）



▲図 2.12 [startSSLKey] が表示されたら [公開鍵] をクリック

この後すぐに使いますので、表示された公開鍵（ssh-rsa から始まる文字列）をまるごとコピーして、メモ帳などにペーストしておきましょう。公開鍵をメモしたら [OK] をクリックして閉じます。



▲図 2.13 表示された公開鍵（文字列）はまるごとコピーしてメモ帳にペーストしておこう

あとは [キャンセル] を繰り返しクリックして、起動中の RLogin はいったん閉じてしまって構いません。RLogin はまた後で使いますので、デスクトップの「rlogin_x64」フォルダとその中にある「RLogin.exe」をごみ箱へ捨てないように注意してください。メモした公開鍵も無くさないようご注意ください。

【コラム】パスフレーズは設定すべき？ しなくてもいい？

秘密鍵に「パスフレーズ」を設定しておくと、鍵を使ってサーバに入ろうとしたとき、「鍵を発動するにはパスフレーズを叫べ…！」という感じでパスフレーズを聞かれます。

つまり、もしあなたの秘密鍵が盗まれて勝手に使われそうになんて、パスフレーズを設定していれば鍵の悪用が防げます。スマホが盗まれてしまっても、パスワードが分からなければロック画面が解除できず、勝手に使えないのと同じです。

ただ「パスワード認証じゃなくて鍵認証なのに、パスフレーズも要るの…？」という点で、初心者を混乱に陥れやすいので、本著では秘密鍵をパスフレーズなしで作って使います。

パスフレーズを設定していれば絶対に安心！ というものではありませんが、上記の理由から、本来であれば設定した方がいいものです。

2.1.2 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 2.14）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 2.14 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 2.15）すれば起動できます。



▲図 2.15 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

Mac で SSH のキーペア（秘密鍵・公開鍵）を作成する

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。^{*6}

```
ssh-keygen -f ~/Desktop/startSSLKey
```

すると次のように、パスフレーズの入力待ち状態になります。何も入力せずに、2回 Enter を押してください。

```
$ ssh-keygen -f ~/Desktop/startSSLKey
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): ←何も入力せずに Enter
Enter same passphrase again: ←何も入力せずに Enter
```

次のように表示されたらキーペア（秘密鍵・公開鍵）の作成は完了です。

^{*6} `ssh-keygen` コマンドは名前のとおり、SSH の鍵（key）を生成（generate）するコマンドです。`-f` オプションでは、生成する鍵のファイル名を指定しています。`~`（チルダ）はホームディレクトリを表しますので、`-f ~/Desktop/startSSLKey` は「`/Users/<ユーザ名>/Desktop`」のフォルダの中に「`startSSLKey`」という名前の鍵を作つて、という意味です

```
$ ssh-keygen -f ~/Desktop/startSSLKey
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mochikoAsTech/Desktop/startSSLKey.
Your public key has been saved in /home/mochikoAsTech/Desktop/startSSLKey.pub.
The key fingerprint is:
a2:52:43:dd:70:5d:a8:4f:77:47:ca:f9:69:79:14:48 mochikoAsTech@ghana
The key's randomart image is:
+---[ RSA 2048]----+
|       . . ooE. |
|       . + o . . |
|       . . . . +. |
|       . . . = o |
|       o . So . . +o |
|       . o . . +o |
|       . . . . . |
|       .           |
|               |
+-----+
```

ホームディレクトリに秘密鍵（startSSLKey）と、公開鍵（startSSLKey.pub）ができるあがっているはずです。cat（キャット）コマンド^{*7}で公開鍵を表示してみましょう。

```
$ cat ~/startSSLKey.pub
ssh-rsa AAAAB3NzaC1yc2E=Unidb+6FjiLw== mochikoAsTech@mochikoMacBook-Air.local
```

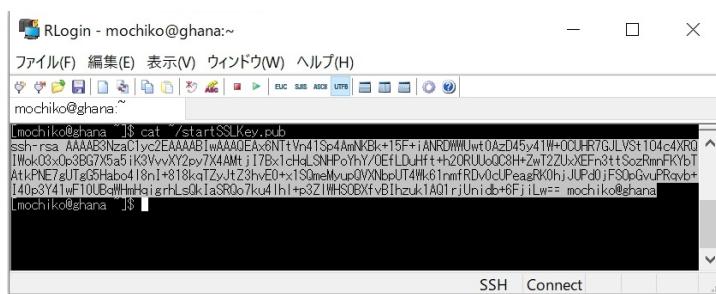
この後すぐに使いますので、表示された公開鍵（ssh-rsa から始まる文字列）をまるごとコピーして、メモ帳などにペーストしておきましょう。

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。

【コラム】ターミナルでコピー＆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 2.16）選択してから Ctrl+c を押す必要はありません。

^{*7} cat は猫ではなく「conCATenate files and print on the standard output」の略です

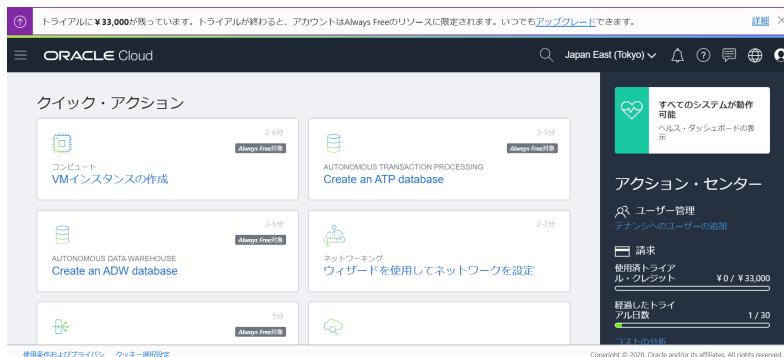


▲図 2.16 マウスで選択するだけでコピーできる

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。

2.2 コンピュートでサーバを立てる

コンソールにサインインしたら、さっそくサーバを立てましょう。[VM インスタンスの作成] をクリックします。(図 2.17)



▲図 2.17 [VM インスタンスの作成] をクリック

[インスタンスの命名] に [startSSLInstance] と入力します。(図 2.18) その下の [オペレーティング・システムまたはイメージ・ソースを選択します] は、何も変更せずそのまま構いません。



▲図 2.18 [インスタンスの命名] に [startSSLInstance] と入力

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、OS はデフォルトの [Oracle Linux 7.7] を使います。Oracle Linux なら Oracle Cloud のツールがあらかじめ入っていますので、**Oracle Linuxでサーバを立てるときは OSはOracle Linuxにすることをお勧めします**。Oracle 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。

Oracle Linux には 2020 年 1 月時点で

- Oracle Linux 6.10
- Oracle Linux 7.7

の 2 種類があります。名前のとおり、Oracle Linux 6.10 は CentOS 6 と同じ RHEL6 系、Oracle Linux 7.7 は CentOS 7 と同じ RHEL7 系なので、使い勝手はほぼ同じです。本著では Oracle Linux 7.7 を使用します。

さらに下に進んで [SSH キーの追加] は、[SSH キーの貼付け] を選択して、そこに先ほどメモしておいた公開鍵をペーストします。公開鍵は改行を含まず、先頭の「ssh-rsa」から末尾の「<ユーザ名>@<ホスト名>」のようなコメントまでで、まるごと 1 行です。(図 2.19)

2.2 コンピュートでサーバを立てる



▲図 2.19 [SSH キーの貼付け] を選択してメモしておいた公開鍵をペースト

公開鍵をペーストしたら [作成] をクリックします。

【コラム】"Out of host capacity."が起きたらどうすればいい？

元気よく [作成] をクリックしたのに、真っ赤な [Out of host capacity.] が表示されてしまった…という方がいらっしゃると思います。大丈夫、あなたは悪くありません。いま理由を説明するので落ち着いてください。「そんなの表示されなかつたよ？」という方は、このコラムは読み飛ばして [サーバが起動するまで待とう] にジャンプしてください。



▲図 2.20 "Out of host capacity."と表示されて何も起きない！

"Out of host capacity."は、直訳すると「ホスト容量が不足しています」という

意味ですが、ホストってなんでしょう？

あなたがいま Oracle Cloud で立てようとしたサーバは、家でいうと「一軒家」ではなくマンションの 101 号室や 403 号室のような「各部屋」にあたります。このときマンションの建物をホストサーバ、各部屋をゲストサーバと呼びます。

「ホストの容量が不足している」ということは…つまり、あなたが Oracle Cloud の無料マンションに入居しようとしたら、「ごめんね、無料マンションは大人気でいま空き部屋がないの」と断られてしまった、という状況なのです。

Oracle Cloud の Always Free は有効期限なしでずっと無料で使える、とても魅力的なサービスなので、順次マンションを建てているものの定期的にリソース不足に陥って、こういう状況になるようです。

この"Out of host capacity."が発生してしまった場合、次の 2 つが起きてホストの容量不足が解消しない限り、Always Free の枠でサーバは立てられません。

- 自分以外のユーザーがサーバーを解約してリソースを開放する
- Oracle Cloud がリソースを増やす

ですが、Always Free とは別に、我々には 30 日間だけ有効な\$300 の無償クレジットがあります！

無料マンションが満室でも、有料マンションなら空きがあります。30 日経ったら消えてしまう\$300 のお小遣いを握りしめて、いざ有料マンションのお部屋を借りにいきましょう！

2.2.1 Always Free ではなく無償クレジットの枠でサーバを立てる

もともと選択していたのは[Always Free 対象]のマークが付いた[VM.Standard.E2.1.Micro(仮想マシン)] という種類のサーバでした。"Out of host capacity."を回避するため、少し上に戻って [シェイプ、ネットワークおよびストレージ・オプションの表示] をクリックしましょう。(図 2.21)

2.2 コンピュートでサーバを立てる



▲図 2.21 「シェイプ、ネットワークおよびストレージ・オプションの表示】をクリック

Oracle Cloud では、サーバスペックごとに「シェイプ」という区分があります。^{*8} インスタンスのシェイプを、いま選択されている「VM.Standard.E2.1.Micro」から変更したいので「[シェイプの変更]」をクリックしてください。



▲図 2.22 「[シェイプの変更]」をクリック

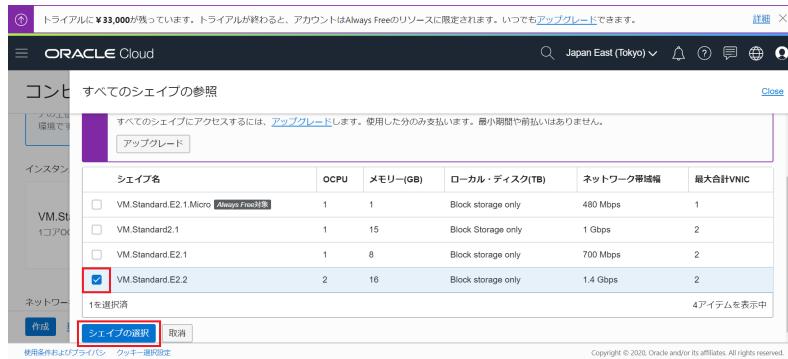
OCPU^{*9}が 2、メモリが 16GB の「VM.Standard.E2.2」^{*10}にチェックを入れて、「[シェ

^{*8} シェイプとはサーバスペックごとの区分のことです。AWS のインスタンスタイプと同じものだと思ってください

^{*9} OCPU は Oracle Compute Units の略で、ごく簡単に言うと物理 CPU です。OCPU (物理 CPU) 1 つは、vCPU (仮想 CPU) 2 つに相当しますので、もし「AWS の EC2 で vCPU が 4 のサーバを使っている。同等スペックのサーバを用意してほしい」と頼まれたら、Oracle Cloud では OCPU が 2 のシェイプを選べば大丈夫です。単純に数字だけで比較して、OCPU が 4 のシェイプを選ぶと CPU のスペックが今までの倍になってしまいしますので注意してください

^{*10} シェイプの名前は、まず接頭辞が「VM」なら仮想サーバ (Virtual Machine)、「BM」なら物理サーバ (Bare Metal) を表しています。その後ろの単語は「Standard」(汎用) や「DenseIO」(高密度 IO) と

イプの選択] をクリックしましょう。



▲図 2.23 「VM.Standard.E2.2」に変更して [シェイプの選択] をクリック

それ以外は何も変更せずに、いちばん下の [作成] をクリックします。

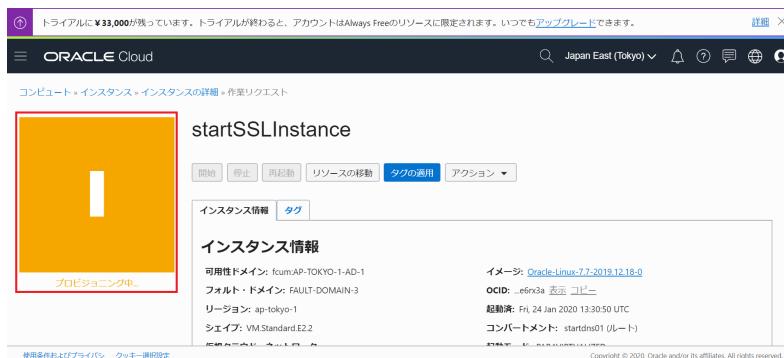


▲図 2.24 [作成] をクリック

2.2.2 サーバが起動するまで待とう

オレンジ色で「プロビジョニング中...」と表示されたら、サーバが用意されるまでそのまま数分待ちましょう。

いった特徴、3番目の「E2」や「3」はシェイプの世代、最後の「2」や「8」はOCPUsの数を表しています



▲図 2.25 「プロビジョニング中...」と表示されたら数分待つ

サーバができあがると、表示が緑色の「実行中」に変わります。おめでとうございます！ これでサーバが立てられました！



▲図 2.26 「プロビジョニング中...」と表示されたら数分待つ

【コラム】Oracle Cloud のコンピュートの金額計算方法

ところで、いま立てた「VM.Standard.E2.2」を1ヶ月使ったら、いったいいいくら分になるのでしょうか？ うっかり\$300を超てしまわないか、ちょっと心配なので計算してみましょう。

コンピュートの価格表^{*11}を見てみると、[VM.Standard.E2.2] は [\$.03]^{*12} と

書いてあります。これは [Pay as You Go (OCPU Per Hour)] と書いてあるとおり、1OCPU につき 1 時間あたりかかる金額です。^{*13}

「VM.Standard.E2.2」は OCPU が 2 なので、\$0.03*2 で 1 時間あたり \$0.06 かかることが分かります。1 ヶ月を 744 時間 (24 時間*31 日) として、\$0.06*744 時間で \$44.64 です。

「VM.Standard.E2.2」を 1 台立てたくらいでは、\$300 の無償クレジットを使い切ることはないので安心しましょう。ちなみに Oracle Cloud では \$1 は 120 円換算^{*14}なので、日本円だと 5356.8 円ですね。

【コラム】Oracle Cloud と AWS はどっちが安い？

Oracle Cloud は他のクラウドに比べて価格が安いのが特徴のひとつです。どれくらい安いのか、同じスペックのサーバで AWS と比較してみましょう。

例えば同スペックの VM.Standard.E2.1 (Oracle Cloud) と m5.large (AWS) を比較すると、Oracle Cloud の価格は AWS の 4 分の 1 以下です。(表 2.1)

▼表 2.1 Oracle Cloud と AWS の価格比較

	Oracle Cloud	AWS
インスタンスの種類	VM.Standard.E2.1	m5.large
CPU	OCPU:1 (vCPU:2相当)	vCPU:2
メモリ	8GB	8GB
1 時間あたり	\$0.03	\$0.124
月額	2678.4 円	11070.72 円

シェアトップを独走する AWS に対して、後発は勝つためにコスト面や性能面でそれぞれ大きなメリットを打ち出してきています。AWS が最適なのであれば AWS を選択すべきですが、「みんなが使っているから」というだけ理由で、あまり深く考えずに AWS を使っているのであれば、他のクラウドにも目を向けてみ

*¹³ <https://www.oracle.com/jp/cloud/compute/pricing.html>

*¹⁴ 2020 年 1 月時点の金額

*¹⁵ ちなみに AWS は、同スペックのサーバでもリージョンごとに価格が異なりますが、Oracle Cloud はどのリージョンでも同一の価格です

*¹⁶ \$1 を 120 円で換算すると \$44.64*120 円で 5356.8 円です

ることを筆者はお勧めします。

2.2.3 接続先となるサーバの IP アドレス

無事にサーバが「実行中」にならたら、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 [startSSLInstance] の、[プライマリ VNIC 情報]（図 2.27）にある [パブリック IP アドレス] をメモ（表 2.2）してください。



▲図 2.27 [プライマリ VNIC 情報] の [パブリック IP アドレス] をメモしておこう

▼表 2.2 インスタンスの [パブリック IP アドレス]

例	パブリック IP アドレス
	140.238.33.51

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。

2.3 SSH でサーバに入ってみよう

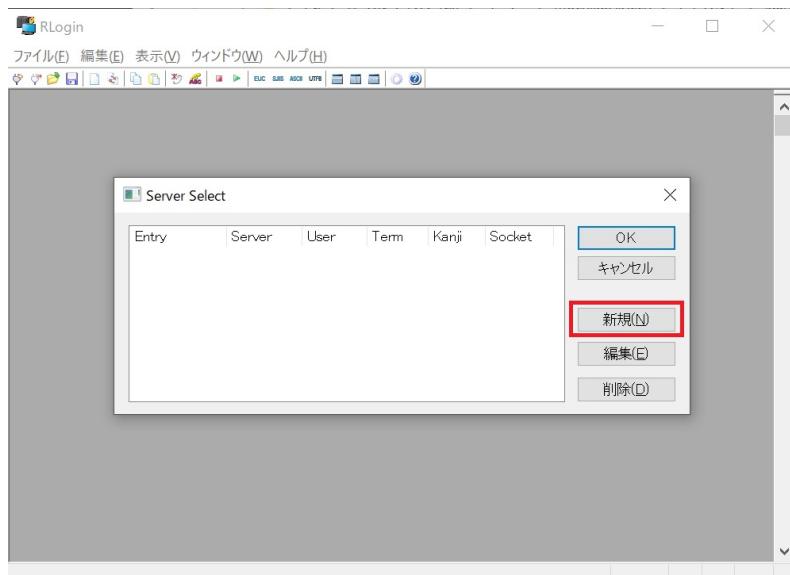
2.3.1 お使いのパソコンが Windows の場合

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 2.28）をダブルクリックして RLogin を起動（図 2.29）

してください。起動したら [新規] をクリックします。



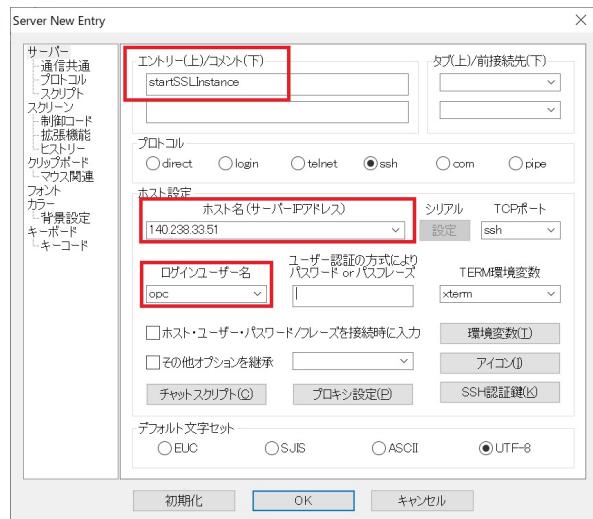
▲図 2.28 RLogin.exe をダブルクリック



▲図 2.29 RLogin が起動したら [新規] をクリック

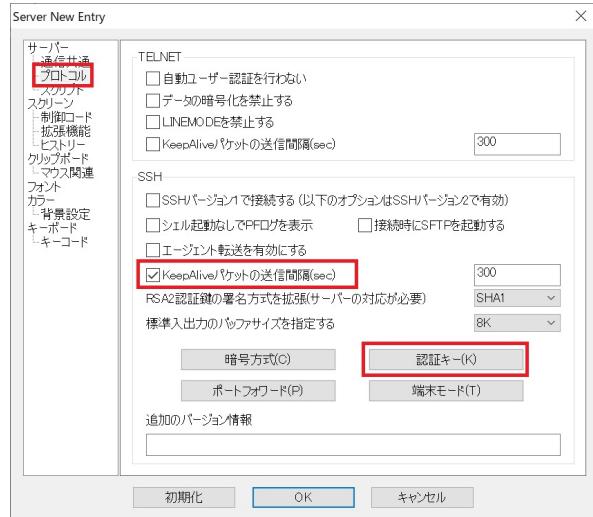
初めに [エントリー (上) /コメント (下)] の上に [startSSLInstance] と入力します。続いて [ホスト名 (サーバー IP アドレス)] に先ほどメモした [パブリック IP アドレス] を入力（図 2.30）します。[ログインユーザー名] には [opc] と入力してください。opc というのは Oracle Linux のインスタンスを作成すると、最初から存在しているデフォルトユーザーです。

2.3 SSH でサーバに入ってみよう



▲図 2.30 [ホスト名 (サーバー IP アドレス)] と [ログインユーザー名] を入力

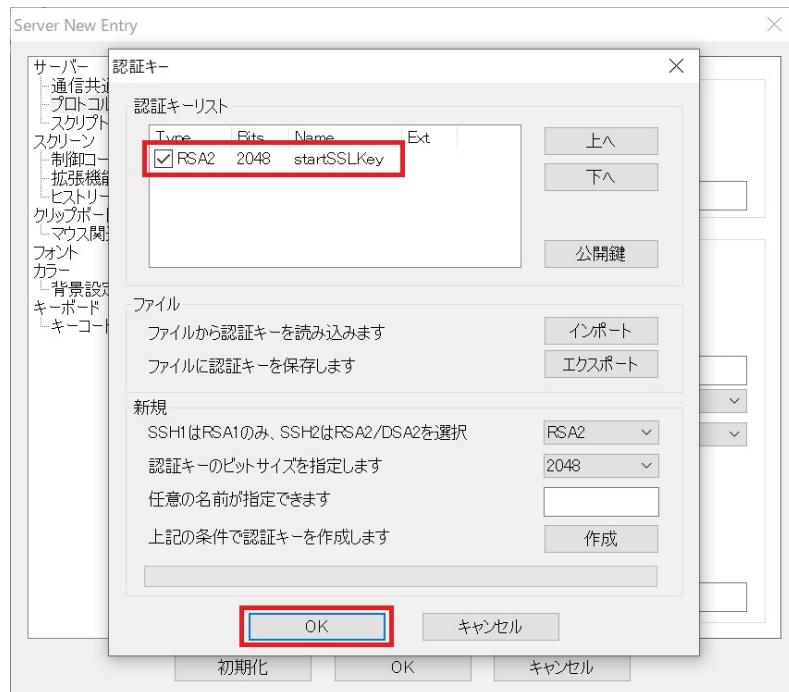
次に左メニューで [プロトコル] を選択 (図 2.31) したら、[KeepAlive パケットの送信間隔 (sec)] にチェックを入れておきます。これを設定しておくとターミナルをしばらく放っておいても接続が勝手に切れません。続いて [認証キー] をクリックします。



▲図 2.31 [KeepAlive パケットの送信間隔 (sec)] にチェックを入れて [認証キー] をクリック

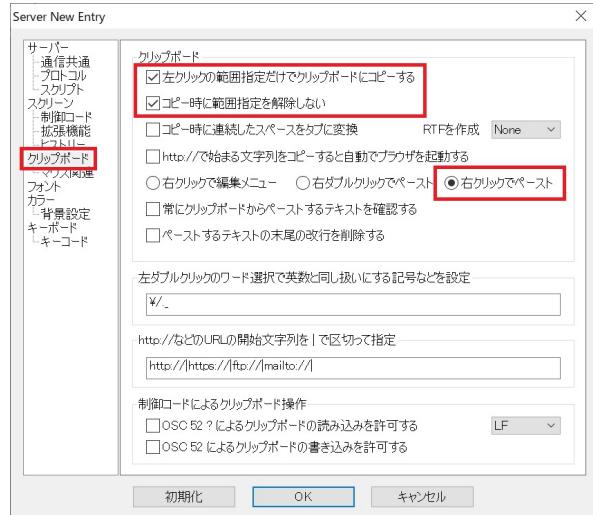
[認証キー] リストで [startSSLKey] にチェックが入っていることを確認（図 2.32）します。これは「ログインするときにこの鍵を使います」というリストです。チェックが入っていたら [OK] をクリックして閉じて構いません。

2.3 SSH でサーバに入ってみよう



▲図 2.32 [startSSLKey] にチェックが入っていることを確認

続いて左メニューで「クリップボード」を選択（図 2.33）したら、「左クリックの範囲指定だけでクリップボードにコピーする」と「コピー時に範囲指定を解除しない」にチェックを入れて「右クリックでペースト」を選択します。



▲図 2.33 右クリックや左クリックの設定

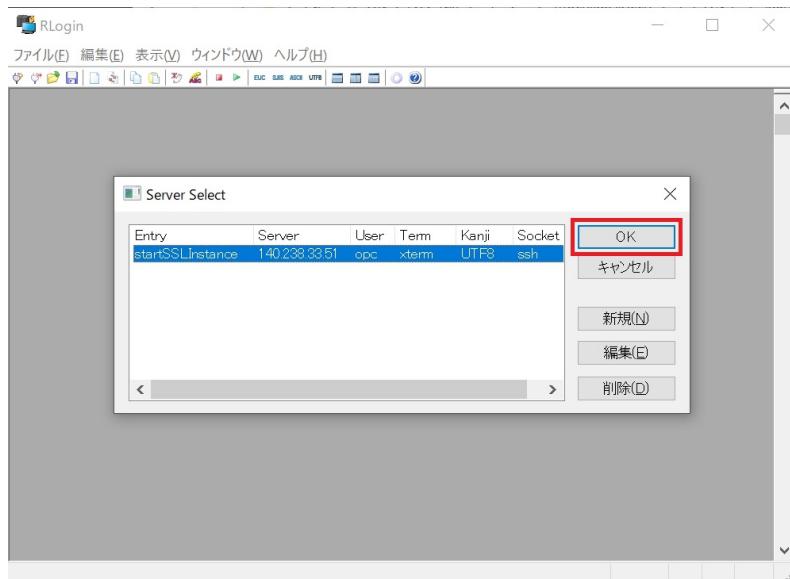
次に左メニューで「フォント」を選択（図 2.34）したら、文字セットを「UTF-8」に変更します。すべて設定できたら「OK」をクリックしてください。



▲図 2.34 文字セットを「UTF-8」に変更

設定が保存できたら「OK」をクリック（図 2.35）してください。

2.3 SSH でサーバに入ってみよう



▲図 2.35 設定が保存できたら「OK」をクリック

すると初回のみ、この「公開鍵の確認」が表示（図 2.36）されます。これは「初めて入るサーバだけど信頼していいですか？本当に接続しますか？」と聞かれているので、「接続する」をクリックしてください。サーバにはそれぞれフィンガープリントという固有の指紋があるため、下部の「この公開鍵を信頼するリストに保存する」にチェックが入っていれば RLogin が覚えていてくれて、次回以降は「これは前に信頼していいって言われたサーバだ！」と判断してそのまま接続させてくれます。



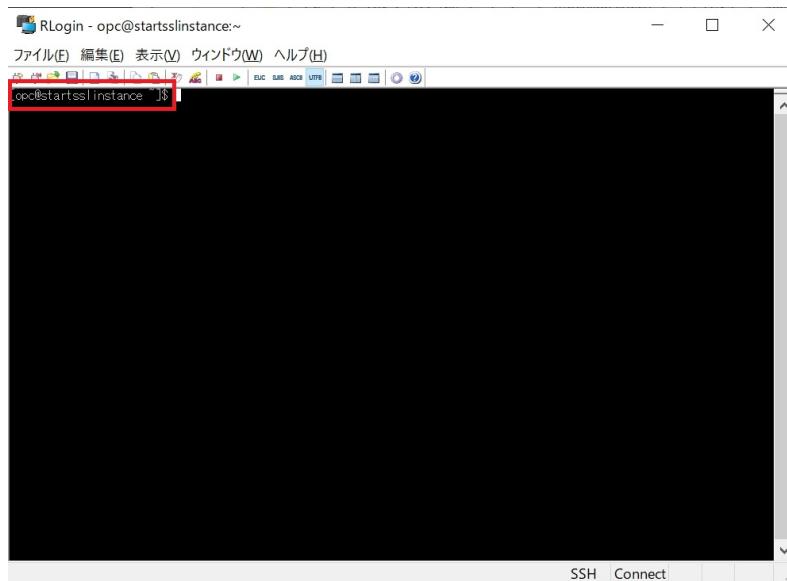
▲図 2.36 「公開鍵の確認」が表示されたら「接続する」をクリック

続いて「信頼するホスト鍵のリストを更新しますか？」と聞かれたら「はい」をクリック（図 2.37）してください。



▲図 2.37 「信頼するホスト鍵のリストを更新しますか？」と表示されたら「はい」をクリック

「opc@startsslinstance」と表示（図 2.38）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！



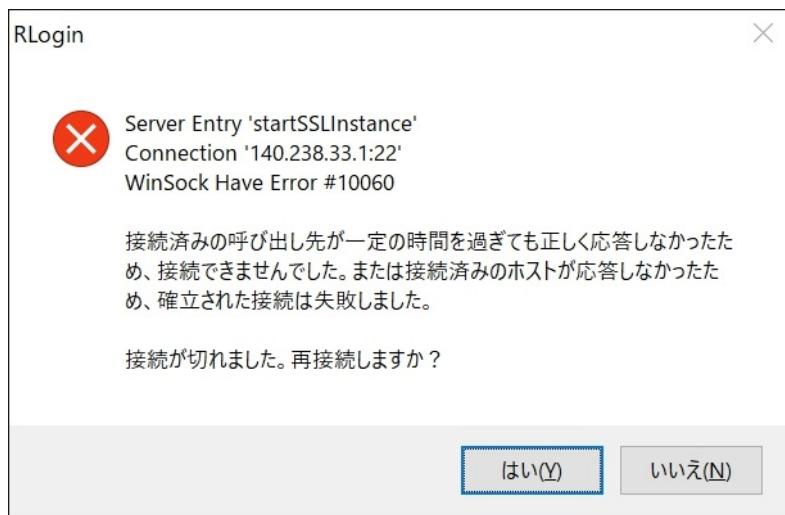
▲図 2.38 「opc@startsslinstance」と表示されたら成功！

もし「opc@startsslinstance」と表示されず、代わりに「SSH2 User Auth Failure "publickey,ssapi-keyex,ssapi-with-mic" Status=1004 Send Disconnect Message... ssapi-with-mic」というようなエラーメッセージが表示（図2.39）されてしまったら、これは「鍵がない人は入れないよ！」とお断りされている状態です。【認証キー】リストで【startSSLKey】にチェックが入っていないものと思われますので【認証キー】の設定を確認してみてください。



▲図2.39 このエラーが表示されたら【認証キー】を確認しよう

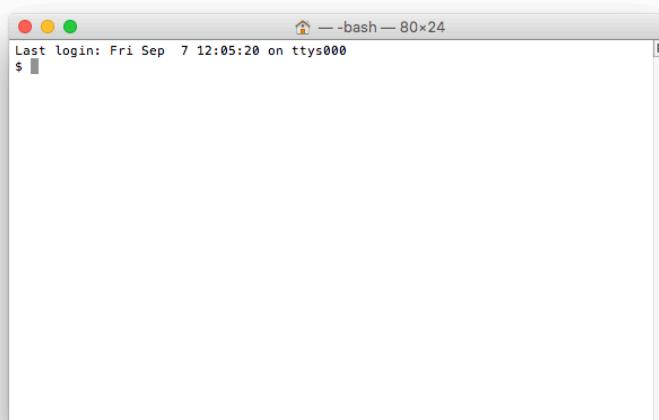
「接続済みの呼び出し先が一定の時間を過ぎても正しく応答しなかったため、接続できませんでした。」というエラーメッセージが表示（図2.40）されてしまった場合は、「ホスト名（サーバーIPアドレス）」に書いた「パブリックIPアドレス」が間違っているものと思われます。「ホスト名（サーバーIPアドレス）」のIPアドレスを確認してみてください。



▲図 2.40 このエラーが表示されたら「ホスト名（サーバー IP アドレス）」の IP アドレスを確認しよう

2.3.2 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 2.41）を起動してください。



▲図 2.41 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 2.42）すれば起動できます。



▲図 2.42 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

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以外の場合は適宜書き換えてください。

```
chmod 600 ~/Desktop/startSSLKey
```

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。「パブリック IP アドレス」の部分は先ほどメモした「パブリック IP アドレス」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、「startSSLKey」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 opc@パブリック IP アドレス -i ~/Desktop/startSSLKey
```

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)?
```

「opc@startsslinstance」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！

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。

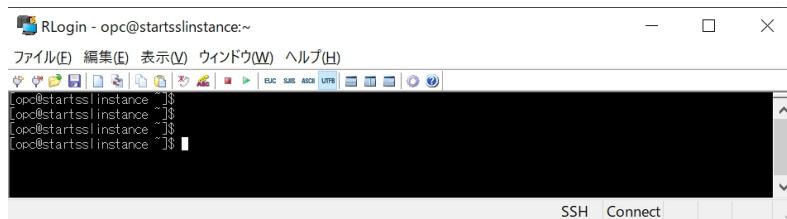
2.4 ターミナルでサーバを操作・設定してみよう

ようやくサーバに入れたので、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。

2.4.1 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 2.43）普通に改行されますよね。



▲図 2.43 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。

```
[opc@startsslinstance ~]$
```

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

```
$ sudo su -
```

と書いて Enter キーを押すと root になります。（図 2.44）「\$」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 2.44 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@startsslinstance ~]#
```

ユーザ名が「opc」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 opc 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

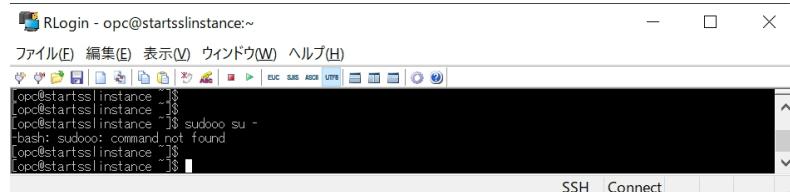
前述の「sudo su -」のようなものを「コマンド」と呼びます。コマンドとはサーバに対して「あれをして」「これをして」と頼む命令のようなものです。

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で失敗したときだけエラーを吐きます**。ですのでコマンドを打った時に何も表示されなくても不安にならなくて大丈夫です。

先ほどの「私を root にして！」という命令である「sudo su -」も、上手くいってちゃんと root になれたのでメッセージは一切出ていないですよね。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 2.45）が表示されました。



▲図 2.45 「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。

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 opc に戻れます。そして opc 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張くような乱暴な切り方なのでお勧めしません。

2.4.2 ミドルウェアをインストール

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

```
$ sudo su -
```

先ずは yum で色々アップデートしておきましょう。Windows アップデートみたいなものです。画面にたくさん文字が流れ、少し時間がかかりますが、最後に [Complete!] と表示されたら問題なく完了しています。(図 2.46) 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
```

▲図 2.46 最後に [Complete!] と表示されたらアップデートは完了

続いて NGINX を入れます。2020 年 1 月現在の安定バージョン^{*15}である 1.16 系をインストールしたいので、yum のリポジトリ（どこから NGINX をダウンロードしていくか）に NGINX 公式を追加しましょう。

```
# rpm -ivh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx-1.16.0-1.el7.noarch.rpm
# cat /etc/yum.repos.d/nginx.repo

[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

yum で NGINX をインストールします。

^{*15} サーバに入っている NGINX 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 NGINX の設定ファイルを書こうと思って調べたとき、例えば 1.12 系の設定方法を参考にしてしまうと、1.16 系の NGINX の環境では上手く動かない可能性があります。

```
# yum install -y nginx
```

[Complete!] と表示されたらインストール完了です。バージョン情報を表示することで、ちゃんとインストールされたか確認してみましょう。

```
# nginx -v  
nginx version: nginx/1.16.1 ←バージョン情報が表示されればインストールできている
```

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 NGINX 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 NGINX を入れましたが、ウェブサーバのミドルウェアは他にも Apache をはじめとして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 NGINX 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 (NGINX のこと) を追加した上で、自動起動をオンにします。

↓ ここ sysctl の何かに要変更

```
# 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。

2.4.3 OS の基本設定をしておこう

タイムゾーンの設定

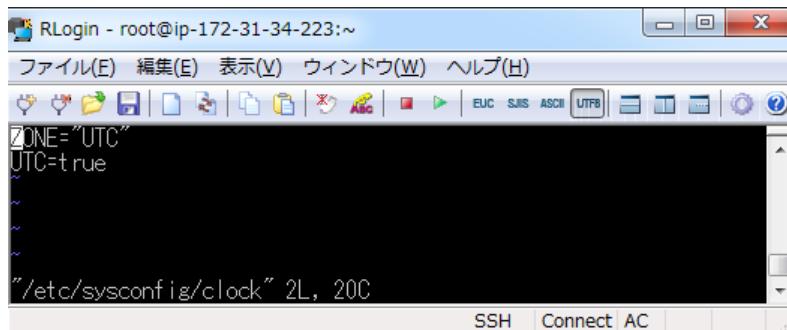
date (データ) コマンドでサーバの時間を確認してみましょう。日本はいま 18:14 なのですが、サーバの時間は 9:14 でずれてしまっています。

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

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
```

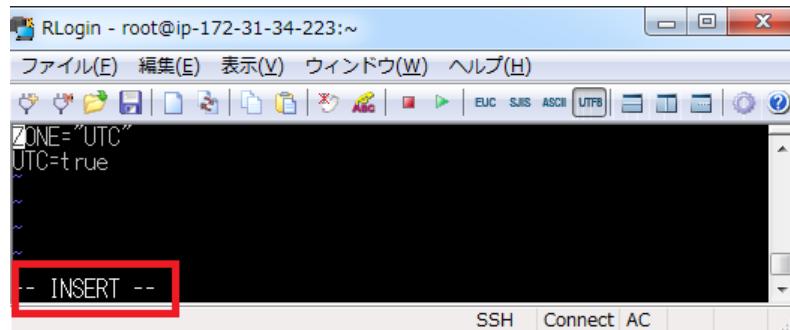
vi（ブイアイ）はテキストファイルを編集するためのコマンドです。vi コマンドでファイルを開くと、最初は次のような「閲覧モード」の画面（図 2.47）が表示されます。閲覧モードは「見るだけ」なので編集ができません。



▲図 2.47 vi コマンドでファイルを開いた

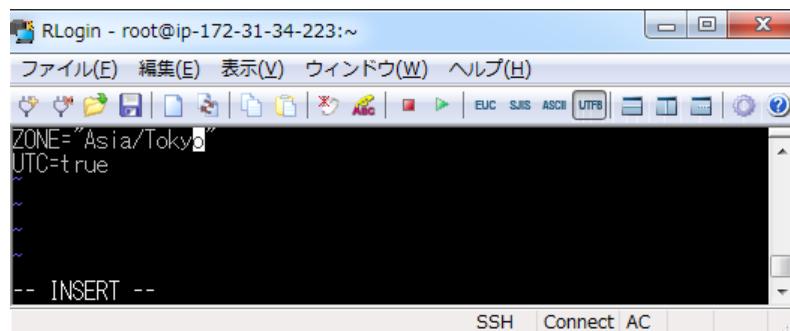
この状態で i（アイ）を押すと「編集モード」*16に変わります。（図 2.48）左下に「-- INSERT --」と表示されていたら「編集モード」です。

*16 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。



▲図 2.48 i (アイ) を押すと「-- INSERT --」と表示される「編集モード」になった

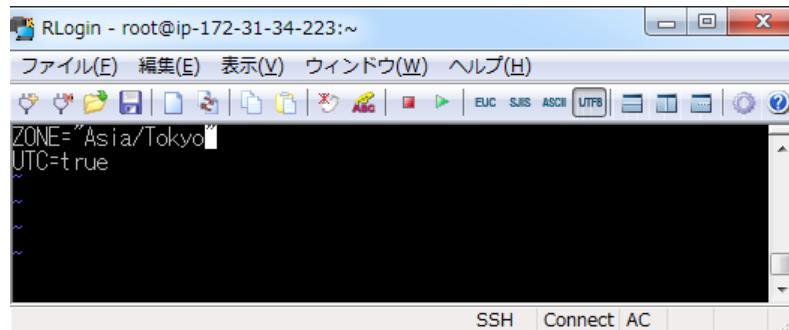
「編集モード」になるとファイルが編集できるようになります。それでは「ZONE="UTC"」を「ZONE="Asia/Tokyo"」(図 2.49) に書き換えてください。



▲図 2.49 「ZONE="UTC"」を「ZONE="Asia/Tokyo"」に書き換える

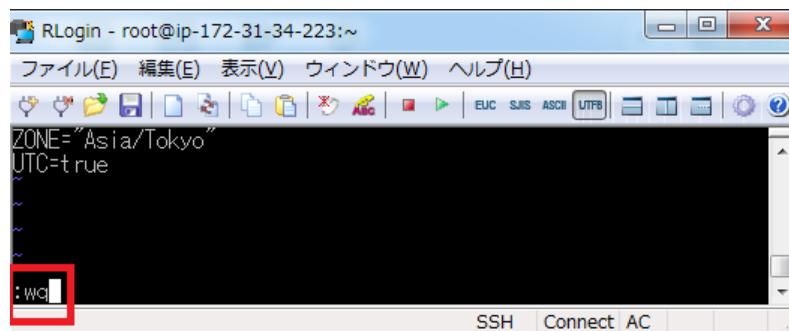
「編集モード」のままだと保存ができないので書き終わったら ESC キーを押します。すると左下の「-- INSERT --」が消えて再び「閲覧モード」になります。(図 2.50)

2.4 ターミナルでサーバを操作・設定してみよう



▲図 2.50 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」*17と入力して Enter キーを押せば変更が保存されます。(図 2.51)



▲図 2.51 「: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「今の全部なかったことにしたい！ 取り合えず vi からいったん抜けたい！」と思ったときは、ESC キーを押して「:q!」*18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*19でファイルの中身を確認してみましょう。

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

*18 保存せずに強制終了 (quit!) という命令なので q! です。

*19 cat は猫ではなく「conCATenate files and print on the standard output」の略です

```
# 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

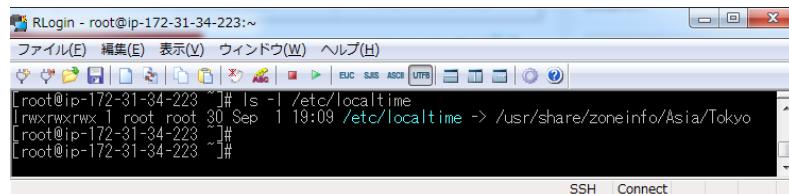
```
# ln -sf /usr/sh
```

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^{*20}で確認してみましょう。（図 2.52）

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



▲図 2.52 シンボリックリンクが生成された

「->」の矢印が実体であるファイルを指しているので、シンボリックリンクができることがあります。

^{*20} 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いたら 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いたら 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。

```
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 (opc)
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^{*21}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
```

^{*21} history の設定は任意ですが、あとで自分の作業を振り返るときに便利なので設定しておくことをお勧めします。

```
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 2.53）してください。

```
--[[path = startaws37 (not exist)]]--
```

▲図 2.53 shift+g で最終行に移動した

最終行に移動したら i (アイ) で「編集モード」にして次の 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^{*22}でファイルの中身を確認してみましょう。

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

^{*22} 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。

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 2.54）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。

```
--[[path = startaws38 (not exist)]]--
```

▲図 2.54 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。^{*23}

以上で「サーバを立てる」という作業はおしまいです。

2.4.4 ターミナルはなんのためにある？

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

ターミナルはサーバを操作するための画面です。

皆さんがパソコン使うときはモニタに表示された画面を見ながらキーボードとマウスを使って「フォルダを開いて先週作った Word ファイルを探す」とか「Word ファイルを開いて今週の報告書を書く」というような操作をするとと思います。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^{*24}を開いて移動するときはダブルクリックの代わりに cd^{*25}というコマンドを叩いて移動しますし、ディレクトリの中を見るときもダブルクリックでフォルダを開く代わりに 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って初めて役に立つ道具なのです。

^{*23} 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。

^{*24} Linux ではフォルダのことをディレクトリと呼びます。

^{*25} change directory の略。

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。^{*26}

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。

2.5 ドメイン名の設定

2.6 まずは HTTP でサイトを公開

2.7 証明書を取得しよう

2.7.1 密密鍵を作ろう

2.7.2 CSR を作ろう

2.7.3 証明書の取得申請

2.7.4 取得した証明書をサーバに置こう

2.8 HTTPS でサイトを公開

^{*26} Mac のターミナルはそもそも黒じゃなくて白ですね。

第3章

基本

3.1 SSL ってなに？

3.2 TLS ってなに？

3.3 SSL と TLS の違いは？

3.4 SSL と SSH って似てる？ 何が違うの？

3.5 HTTPS で始まるページで鍵のマークが壊れて表示された

3.6 種類

3.6.1 SSL サーバ証明書

3.6.2 SSL クライアント証明書

3.7 どんなシーンで使われている？

3.8 SSL 証明書は全然違う 2 種類の仕事をしている

3.8.1 Web サイトで送受信する情報を暗号化すること

3.8.2 Web サイト運営者の身元を証明すること

3.9 鍵マークが壊れるケース

3.9.1 すべて HTTP で通信している⁷⁸

3.9.2 HTTPS だけど一部が HTTPS じゃないとき

画像と CSS の指定が絶対パスだった

3.10 ウェブページが表示されるまで

あとがき

数ある技術書の中から「SSL をはじめよう」を手に取ってくださったあなたに感謝します。

2020年2月
mochikoAsTech

PDF 版のダウンロード

本著（紙の書籍）をお買い上げいただいた方は、下記の URL から PDF 版を無料でダウンロードできます。

- ダウンロード URL : <https://mochikoastech.booth.pm/items/xxxxxx>
- パスワード : **xxxxxx**

Special Thanks:

- ネコちゃん

レビュー

- Takeshi Matsuba

参考文献

- ぶんけん
 - ??

著者紹介

mochiko / @mochikoAsTech

元 Web 制作会社のシステムエンジニア。技術書典で出した本がきっかけで、テクニカルライターの仕事を始めた。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したのち、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。「分からない気持ち」に寄り添える技術者になれるように日々奮闘中。技術書典 4,5,6 で頒布した「DNS をはじめよう」「AWS をはじめよう」「技術をつたえるテクニック」「技術同人誌を書いたあなたへ」は累計で 7,800 冊を突破。

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

Hikaru Wakamatsu

表紙デザインを担当。

Shinya Nagashio

挿絵デザインを担当。

SSLをはじめよう

証明書の発行からトラブルシューティングまで

2020-02-29/2020-03-01 技術書典 8 初版

著 者 mochikoAsTech

デザイン Hikaru Wakamatsu / Shinya Nagashio

発行所 mochikoAsTech

印刷所 日光企画

(C) 2020 mochikoAsTech