

SSL をはじめよう

「なんとなく」から「ちゃんとわかる！」へ

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？	19
1.2.2 Oracle Cloud のコンソールにサインイン	24
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】SSH の秘密鍵にパスフレーズは設定すべき？	35
2.1.2 お使いのパソコンが Mac の場合	36
【コラム】ターミナルでコピー＆ペーストするには？	38
2.2 コンピュートでサーバを立てる	39
【コラム】"Out of host capacity."が起きたらどうすればいい？	41
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	43

2.2.2 サーバが起動するまで待とう	45
【コラム】Oracle Cloud のコンピュートの金額計算方法	46
【コラム】Oracle Cloud と AWS はどっちが安い？	47
2.2.3 接続先となるサーバの IP アドレス	48
2.3 SSH でサーバに入ってみよう	48
2.3.1 お使いのパソコンが Windows の場合	48
2.3.2 お使いのパソコンが Mac の場合	58
2.4 ターミナルでサーバを操作・設定してみよう	60
2.4.1 ターミナルの基本操作に慣れよう	61
2.4.2 ミドルウェアをインストール	63
2.4.3 Firewalld で HTTP と HTTPS を許可しよう	65
2.4.4 SELinux を無効にしておこう	66
2.4.5 OS 再起動してみよう	70
2.4.6 ターミナルはなんのためにある？	70
2.4.7 なぜかサイトが見られない	71
2.4.8 サーバの手前にあるファイアウォールにも穴をあけよう	72
2.4.9 HTTP でサイトを見てみよう	76
2.5 ドメイン名の設定	77
2.6 SSL 証明書を取得しよう	80
2.6.1 SSL 証明書にまつわる登場人物	80
【コラム】SSL 証明書の秘密鍵にパスフレーズは設定すべき？	81
2.6.2 【ドリル】CSR で入力すべきなのはクライアントの情報？	84
2.6.3 証明書の取得申請	85
2.6.4 Windows で証明書と中間 CA 証明書をアップしよう	98
2.6.5 Mac で証明書と中間 CA 証明書をアップしよう	100
2.6.6 SSL 証明書と中間 CA 証明書を 1 ファイルにまとめよう	101
2.6.7 NGINX で HTTPS のバーチャルホストを作ろう	102
2.7 HTTPS でサイトを開いてみよう	103
2.7.1 LB のところで SSL ターミネーションする方法もある	104
第 3 章 SSL/TLS について学ぼう	105
3.1 「サイトを HTTPS 化する」とは何か？	106
3.1.1 個人情報をやりとりしないサイトでも HTTPS にしなきゃだめ？	106
3.1.2 HTTP 化しないと起きる「わるいこと」	106
3.1.3 HTTPS 化すると起きる「いいこと」	108

3.2	SSL/TLS とは？	109
3.2.1	SSL と TLS はどういう関係？	109
3.2.2	SSL イコール HTTPS ではない	109
3.3	SSL 証明書とは	110
3.3.1	SSL サーバ証明書と SSL クライアント証明書	110
3.3.2	SSL 証明書はどんな場面で使われている？	111
3.4	SSL 証明書は全然違う 2 種類の仕事をしている	112
3.4.1	Web サイトで送受信する情報を暗号化すること	113
3.4.2	Web サイト運営者の身元を証明すること	113
3.5	鍵マークが壊れるケース	113
3.5.1	すべて HTTP で通信しているとき	113
3.5.2	HTTPS だけど一部が HTTPS じゃないとき	113
3.6	ウェブページが表示されるまで	113
3.6.1	1 往復で表示されるわけじゃない	113
3.7	SSL 証明書は何を証明してくれるのか？	113
3.7.1	ネットバンクの事例	113
3.8	認証局事業者の身元は誰が証明する？	113
3.8.1	身元保証の連鎖をつなぐ中間 CA 証明書とルート証明書	113
3.9	SSL 証明書はどうしてあんなに値段に差があるの？	113
3.10	同じ「SSL 証明書」という名前でも 3 つの種類がある	113
3.10.1	EV 証明書	113
3.10.2	OV 証明書	113
3.10.3	DV 証明書	113
3.10.4	3 つの違いは何か？	113
3.10.5	プラウザベンダーによる EV 証明書の扱いの変化	113
3.11	その他の証明書	113
3.11.1	中間証明書	113
3.11.2	クロスルート証明書	113
3.12	どの証明書を買えばいい？	113
3.12.1	ワイルドカード証明書	113
3.12.2	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？	113
3.12.3	コモンネームが*.example.com の証明書は example.com で使え る？	113
3.12.4	Let'sEncrypt	113

目次

3.13	CDN と証明書	113
3.13.1	CDN を使ったら古い端末でサイトが見られなくなった	113
3.13.2	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	113
3.13.3	SNI Server Name Indication	113
あとがき		115
	PDF 版のダウンロード	115
	Special Thanks:	115
	レビュー	115
	参考文献	115
著者紹介		117

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

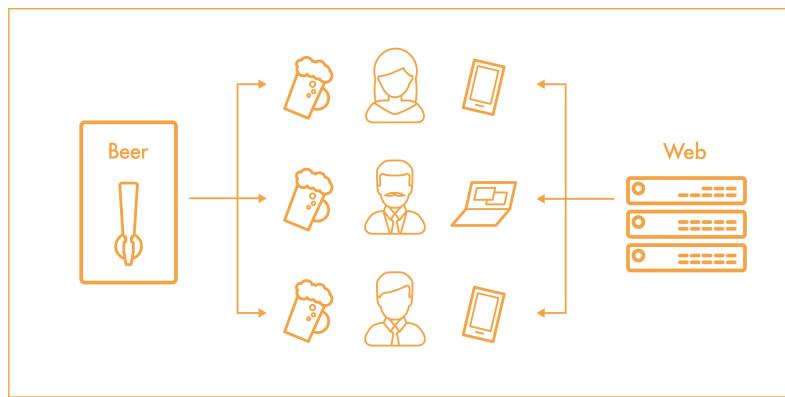
まずは 1 つめのウェブサーバを立てるため、アカウント作成から始めましょう。

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？

本著では次の2つ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

Oracle Cloud Free Tier のページが表示されたら、[今すぐ始める（無償）] をクリックします。（図 1.3）

第1章 Oracle Cloud のアカウントを作ろう



▲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」と表示されたら、[電子メール・アドレス] と [国/地域] 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) 後で分からなくなないように、登録した項目を表 1.1 にメモしておきましょう。

▼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	



▲図 1.4 入力したら [次] をクリック

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って応答時間も遅くなります。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〔日本東部(東京)〕の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)	

The screenshot shows the 'Account details input' screen. It has three main sections: 'Account type' (radio buttons for 'Corporate use' and 'Personal use', with 'Personal use' selected), 'Cloud account name' (text input field containing 'startdns01'), and 'Home region' (dropdown menu showing 'Japan East (Tokyo)'). A note at the bottom right of the region dropdown says: 'Always Freeのサービスが選択したリージョンで使用可能になりました。サービスの可用性については、[リージョン] を参照してください。' (The Always Free service you selected is now available in the chosen region. For information on service availability, see [Region].)

▲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 registration form with several input fields highlighted by red boxes:

- 名 *: 猫村
- 姓 *: もち子
- 部門名 *: 借入
- 役職: (empty)
- 住所 *: 新宿四丁目1番6号
JR新宿ミライナタワー
- 市区町村 *: 新宿区
- 都道府県 *: TOKYO
- 郵便番号 *: 160-0022
- 国/地域: 日本

▲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)

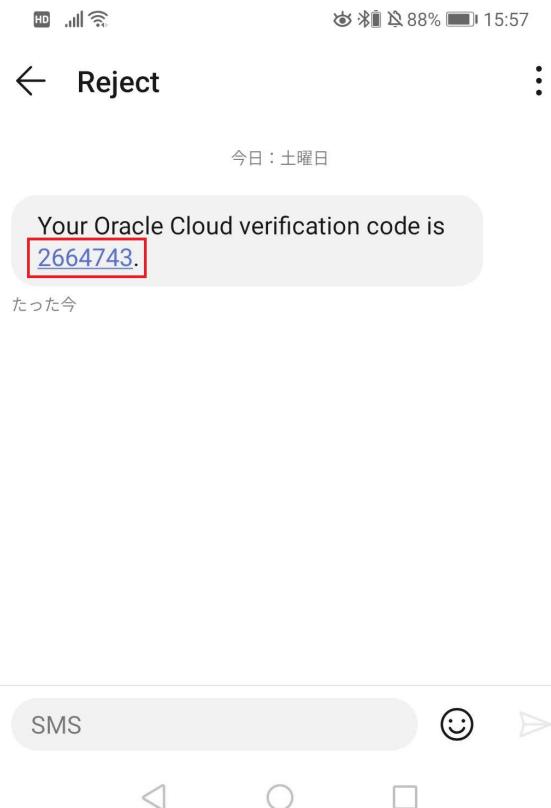
The screenshot shows a mobile number input screen with fields highlighted by red boxes:

- モバイル番号 *: 日本(81)
- 番号: 90 [red box]
- Next button: 次: モバイル番号の確認

上記のモバイル番号および電子メール startdn0@gmail.com に基づく検証コードがすでにある場合、[ここをクリックしてコードを確認してください。](#)

▲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

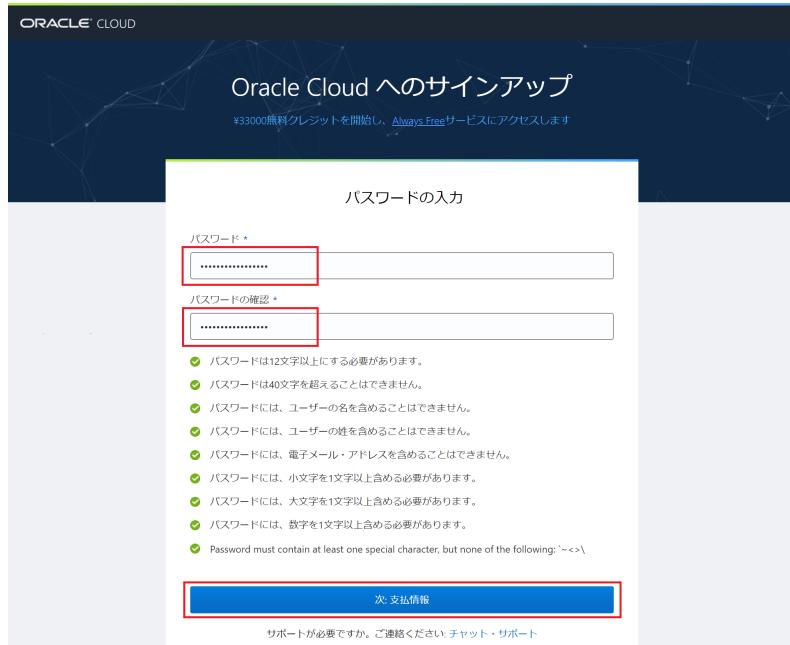
【コラム】どうしても SMS が届かない！ そんなときは？

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、最初は届かず、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、この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] が表示されます。[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ます。本著では基本的にこ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。

なおページに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。

*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



▲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

カード情報 ▲

カードの種類 *

Visa Mastercard
 Amex JCB

カードの番号 *

有効期限 *

CVN *

このコードは、クレジットカードの裏面または表面に印字されている3行または4行の番号です。

Finish

▲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>

1.2 Oracle Cloud でアカウント登録



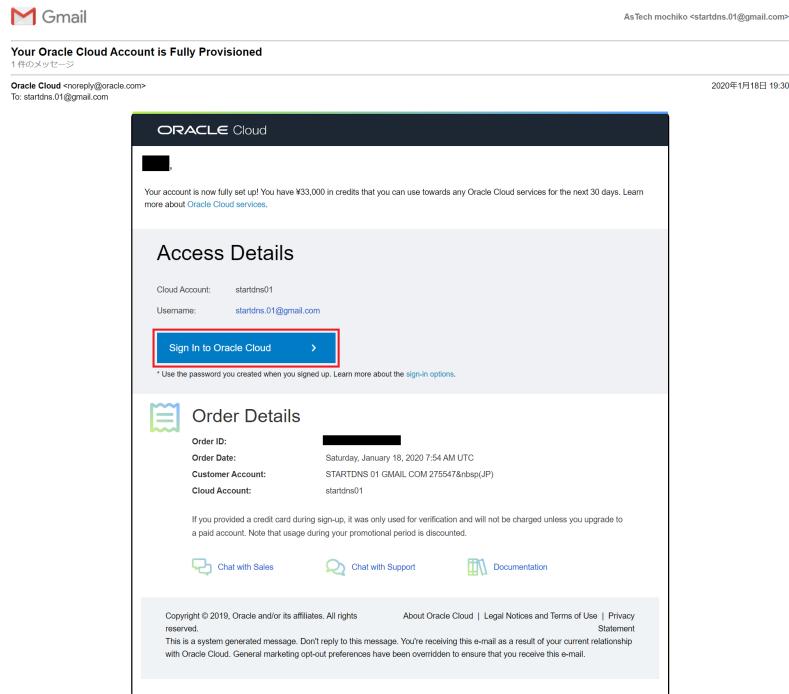
▲図 1.13 チェックを入れて [サインアップの完了] をクリック

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、一旦休憩にしましょう。



▲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 数時間後、準備完了を知らせるメールが届く

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

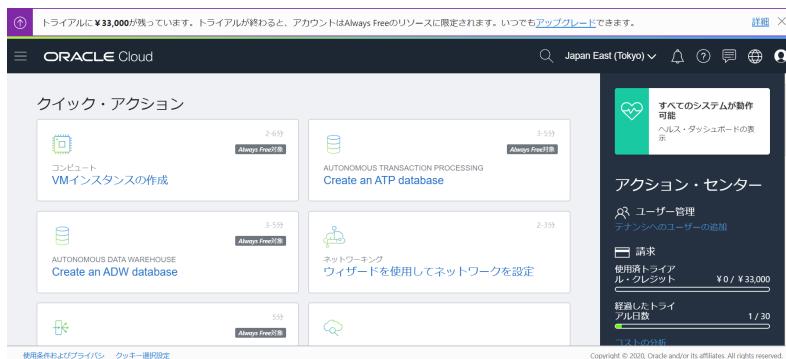
*14 日本語だとログインの方が馴染みがあるかも知れませんが、サインインはログインと同じ意味です

*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



▲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！ 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 Oracle Cloud からの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。

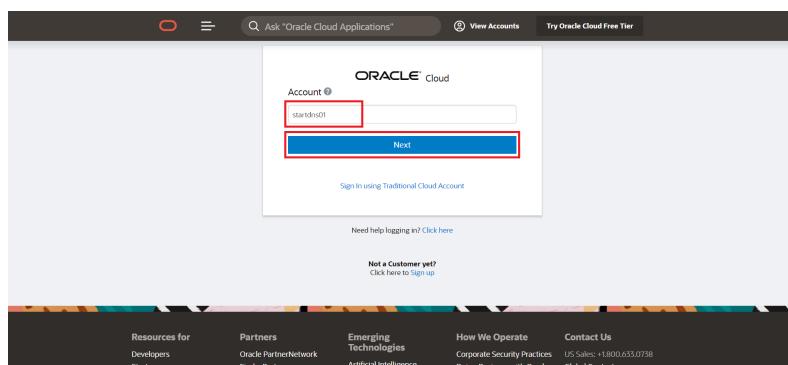
*16 <https://www.oracle.com/jp/>

第1章 Oracle Cloud のアカウントを作ろう



▲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>)



▲図 2.1 RLogin

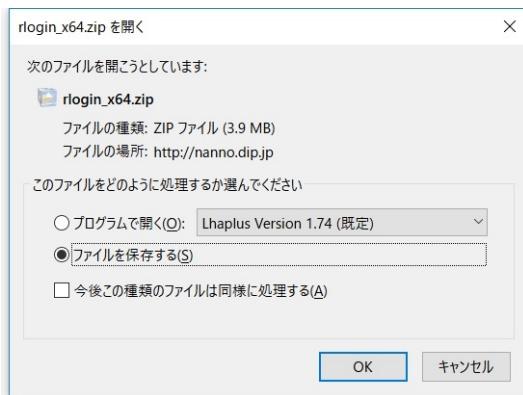
本著ではいちばん上の RLogin (図 2.1) を使って説明していきますので、特にこだわりがない場合は RLogin を使うことをお勧めします。RLogin の「実行プログラム (64bit)^{*1}」(図 2.2) の URL、http://nanno.dip.jp/softlib/program/rlogin_x64.zip をクリックしてください。

^{*1} 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

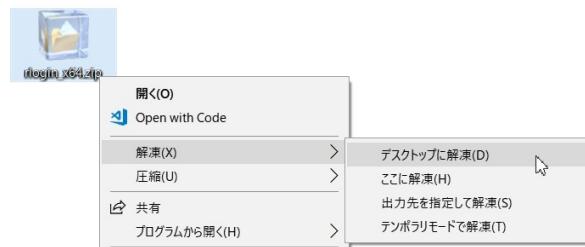
ダウンロードした ZIP ファイルを保存（図 2.3）します。保存場所はどこでも構いませんが、後でどこに置いたか分からなくなりそうな人はデスクトップに保存しておきましょう。



▲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]*2をクリックします。

*2 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/>



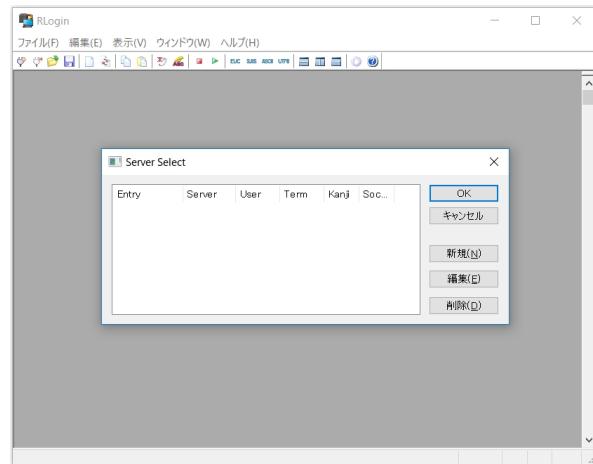
▲図 2.4 ZIP ファイルを右クリックして解凍>デスクトップに解凍

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」*3（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

*3 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。

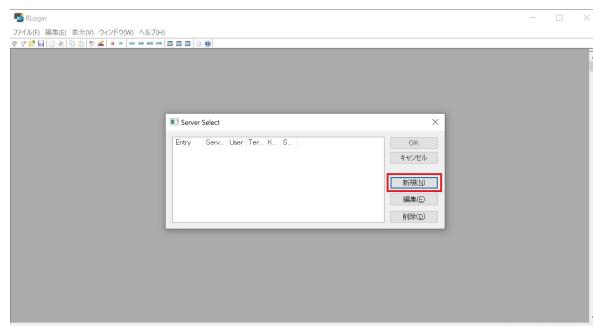


▲図 2.6 RLogin が起動した

これで RLogin のインストールは完了です。

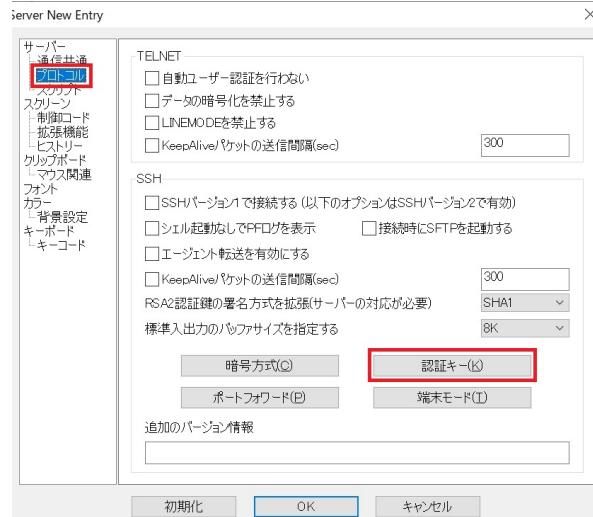
Windows で SSH のキーペア（秘密鍵・公開鍵）を作成する

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



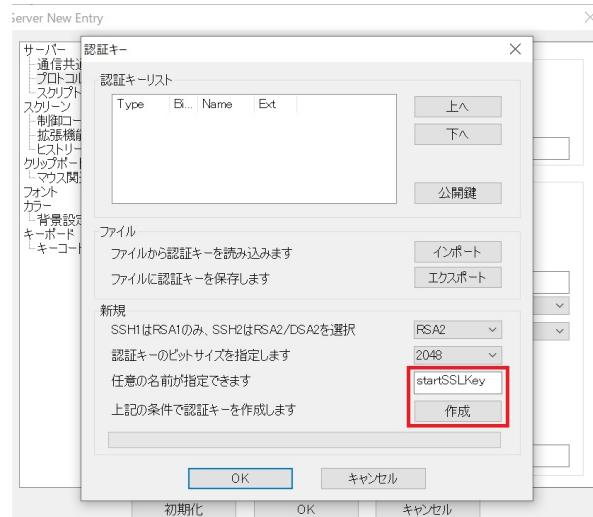
▲図 2.7 [新規 (N)] をクリック

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。



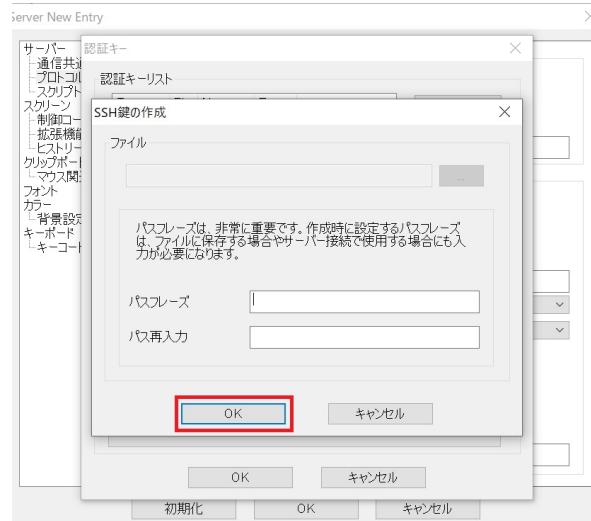
▲図 2.8 [プロトコル] を選択して [認証キー (K)] をクリック

[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

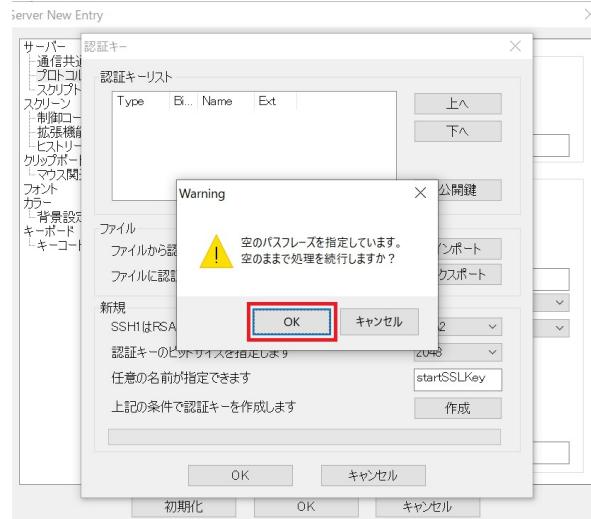
[パスフレーズ] と [パス再入力] には何も入力せず、[OK] をクリックします。^{*4}



▲図 2.10 何も入力せず [OK] をクリック

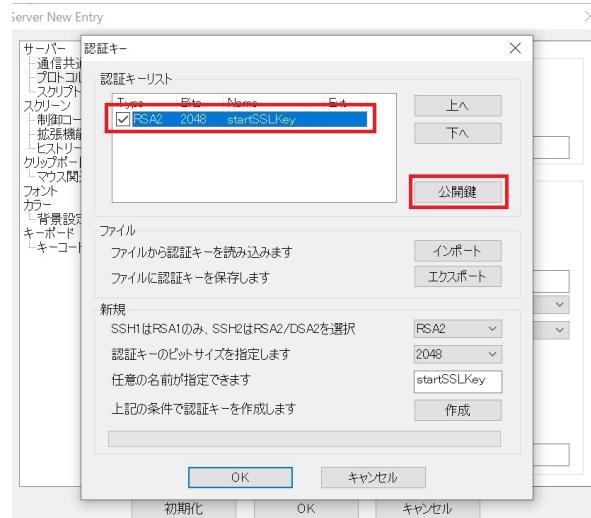
[空のパスフレーズを指定しています。空のままで処理を続行しますか？] と表示されますが、そのまま [OK] をクリックします。

^{*4} 「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がパスフレーズです



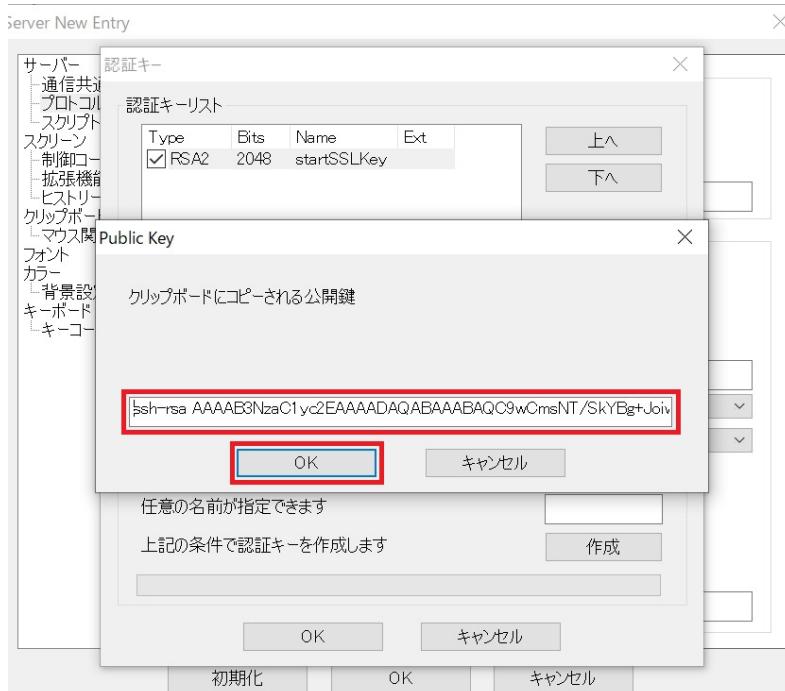
▲図 2.11 [OK] をクリック

【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、[クリップボードにコピーされる公開鍵] 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、後でサーバへ入るときに使いますので、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】SSH の秘密鍵にパスフレーズは設定すべき？

秘密鍵に [パスフレーズ] を設定しておくと、鍵を使って SSH で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。

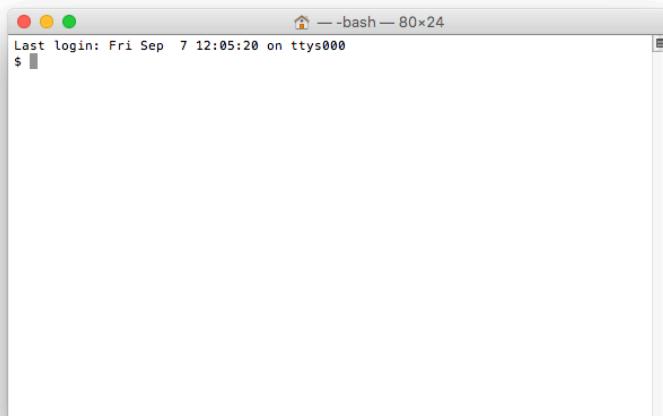
つまり、もしあなたの秘密鍵が盗まれて誰かに勝手に使われそうになっても、パスフレーズを設定していれば鍵の悪用が防げます。スマホ本体が盗まれてしまっても、パスワードが分からなければロック画面が解除できず、勝手に使えないのと同じです。

ただ「パスワード認証じゃなくて鍵認証なのに、やっぱりパスフレーズが要るの…？」という具合に、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。

パスフレーズは「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。後で「やっぱり設定しておこう」と思ったら、一度作成した秘密鍵に後からパスフレーズを設定することも可能です。

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。^{*5}

```
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.
```

^{*5} `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

```
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。

```
$ 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）ができるがっているはずです。cat（キャット）コマンド^{*6}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。

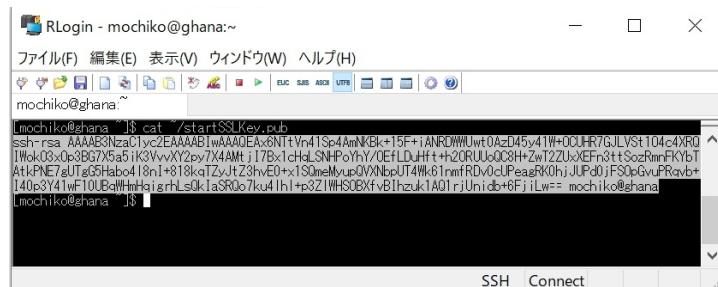
以上で事前準備は完了です。お待たせしました。いよいよサーバを立てましょう。

【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。

^{*6} cat は猫ではなく「conCATenate files and print on the standard output」の略です

せん。



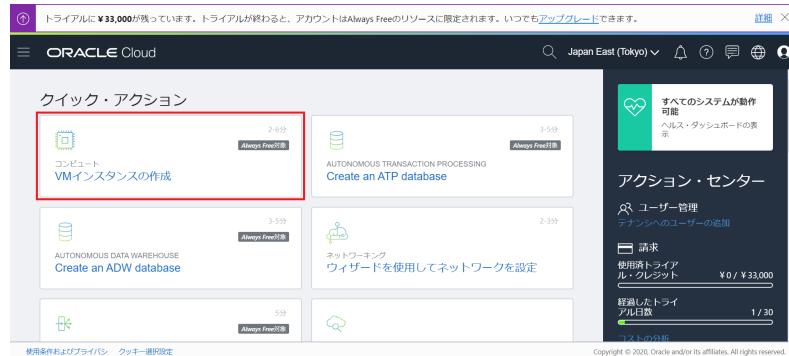
▲図 2.16 マウスで選択するだけでコピーできる

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。ターミナルのソフトにもよりますが、基本的に Ctrl+p は使えないでの注意してください。

2.2 コンピュートでサーバを立てる

それでは下準備ができたので、Oracle Cloud のコンソールに戻ってサーバを立てましょう。コンソールにサインインしたら、[VM インスタンスの作成] をクリック（図 2.17）します。ちなみに Oracle Cloud にはデータベース管理やストレージなど、さまざまなサービスがありますが、クラウドサーバや物理サーバなどのサーバが立てられるサービスは「コンピュート」と呼ばれています。そして Oracle Cloud 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「サーバのことだな」とだと思ってください。

第2章 Oracle Cloud でサーバを立てよう



▲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、何も変更せずに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。

本著では、OSはデフォルトの【Oracle Linux 7.7】を使用します。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。

2020年1月時点でのOracle Linuxには次の2種類があります。

- Oracle Linux 6.10
- Oracle Linux 7.7

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。

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、まるごと1行です。(図2.19)



▲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？

さて、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ったよ?」という方は、このコラムは読み飛ばして、この先の「サーバが起動するまで待とう」まで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。



▲図 2.21 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ぶ

「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」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です。そのため Oracle Cloud 側も定期的に新築マンションを追加しているものの、定期的にリソース不足に陥っては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、次のどちらか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与えられています。たとえ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たら、次の方法で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

次の手順は、"Out of host capacity."が表示された人だけ実施してください。

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、少し上にスクロールして【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 2.22)



▲図 2.22 【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。⁷ 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 2.23 [シェイプの変更] をクリック

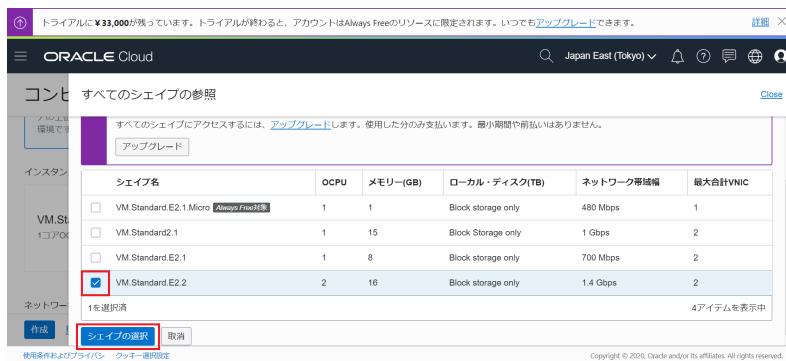
OCPU⁸が 2、メモリが 16GB の [VM.Standard.E2.2]⁹にチェックを入れて、[シェイプの選択] をクリックしましょう。

⁷ 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

⁸ 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

⁹ 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 OCPU の数を表しています

2.2 コンピュートでサーバを立てる



▲図 2.24 「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 2.25 [作成] をクリック

2.2.2 サーバが起動するまで待とう

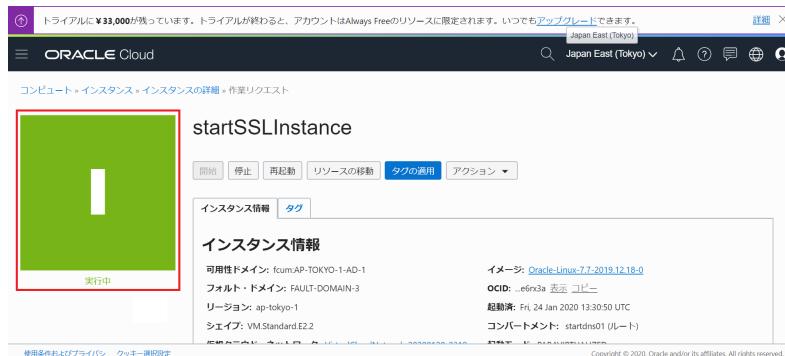
オレンジ色で「プロビジョニング中...」と表示されたら、サーバが用意されるまでそのまま数分待ちましょう。

第2章 Oracle Cloud でサーバを立てよう



▲図 2.26 「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 2.27 「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^{*10}を見てみると、[VM.Standard.E2.2] は [\$.03]^{*11} 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。^{*12}

「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^{*13}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

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

*13 2020 年 1 月時点の金額

*14 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

*15 \$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 2.28）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 2.28 [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

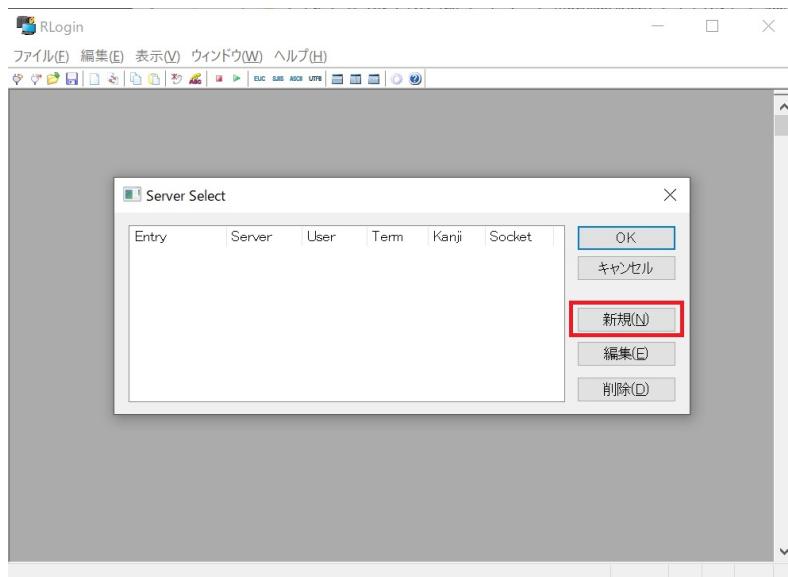
Windows のパソコンを使っている方は、デスクトップの [rlogin_x64] というフォルダの中にある [RLogin.exe]（図 2.29）をダブルクリックして RLogin を起動（図 2.30）

2.3 SSH でサーバに入ってみよう

してください。起動したら [新規] をクリックします。

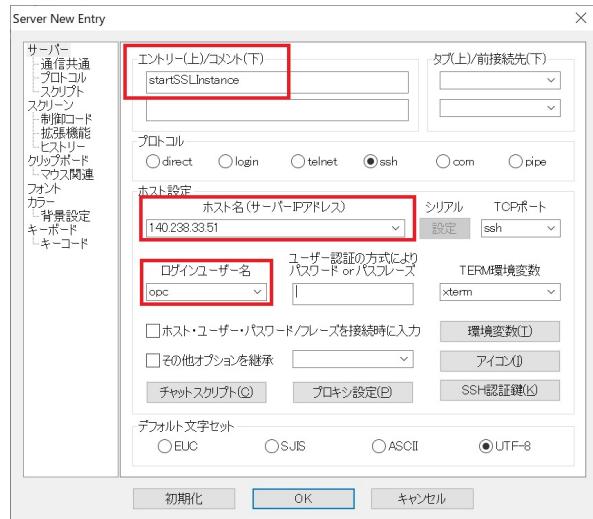


▲図 2.29 RLogin.exe をダブルクリック



▲図 2.30 RLogin が起動したら [新規] をクリック

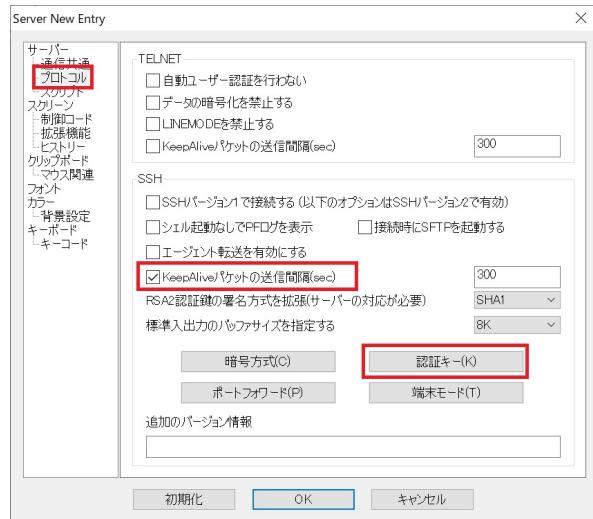
初めに [エントリー (上) /コメント (下)] の上に [startSSLInstance] と入力します。続いて [ホスト名 (サーバー IP アドレス)] に先ほどメモした [パブリック IP アドレス] を入力（図 2.31）します。[ログインユーザー名] には [opc] と入力してください。opc というのは Oracle Linux のインスタンスを作成すると、最初から存在しているデフォルトユーザーです。



▲図 2.31 [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

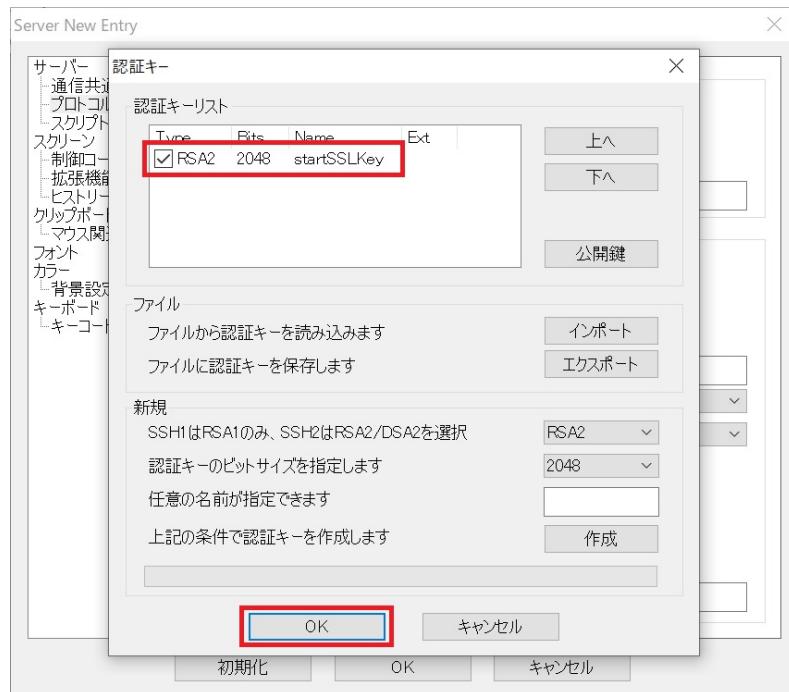
次に左メニューで [プロトコル] を選択 (図 2.32) したら、[KeepAlive パケットの送信間隔 (sec)] にチェックを入れておきます。これを設定しておくとターミナルをしばらく放っておいても接続が勝手に切れません。続いて [認証キー] をクリックします。

2.3 SSH でサーバに入ってみよう



▲図 2.32 [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

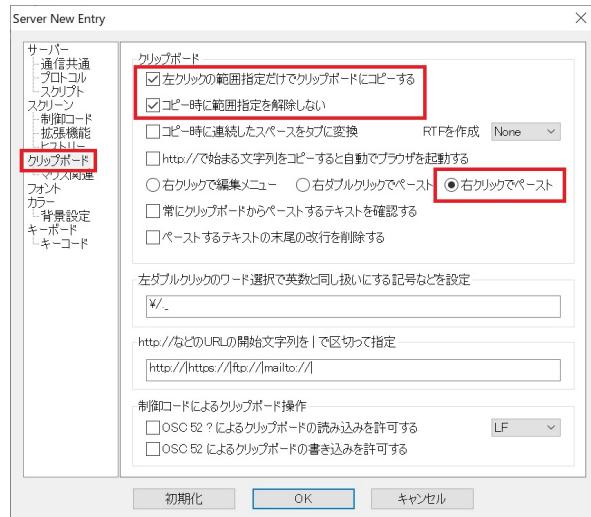
[認証キー] リストで [startSSLKey] にチェックが入っていることを確認（図 2.33）します。これは「ログインするときにこの鍵を使います」というリストです。チェックが入っていたら [OK] をクリックして閉じて構いません。



▲図 2.33 [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 2.34）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。

2.3 SSH でサーバに入ってみよう



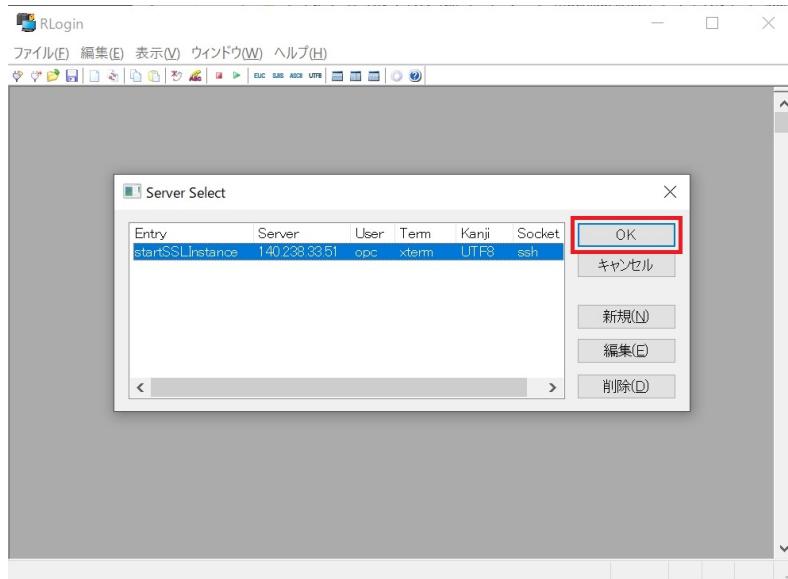
▲図 2.34 右クリックや左クリックの設定

次に左メニューで「フォント」を選択（図 2.35）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 2.35 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 2.36）してください。



▲図 2.36 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 2.37）されます。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。



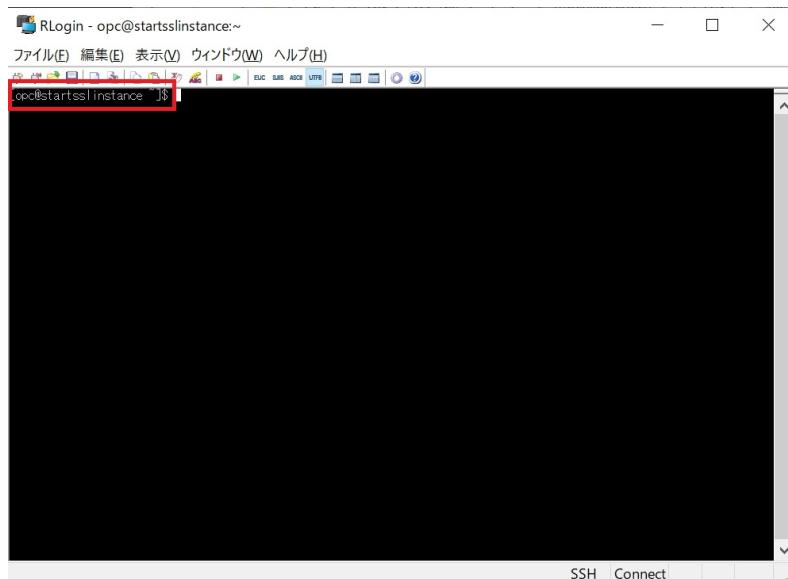
▲図 2.37 「公開鍵の確認」が表示されたら「接続する」をクリック

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 2.38）してください。



▲図 2.38 「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 2.39）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！



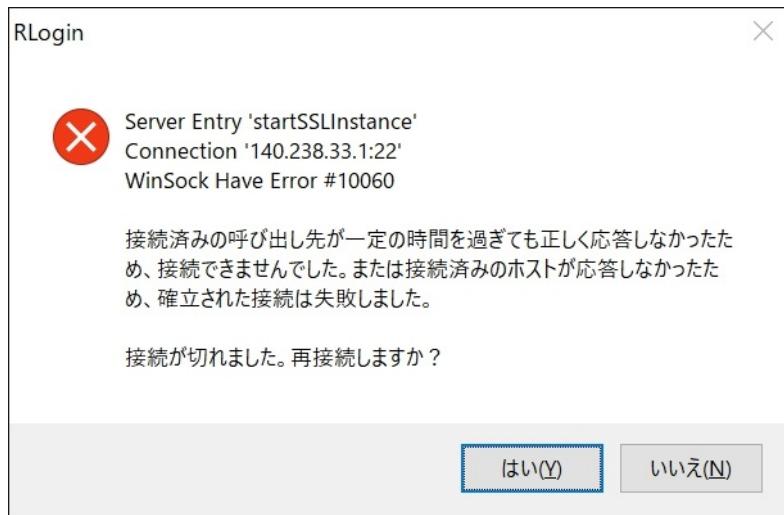
▲図 2.39 「opc@startsslinstance」と表示されたら成功！

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... gssapi-with-mic」というようなエラーメッセージが表示（図 2.40）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 2.40 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 2.41）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。

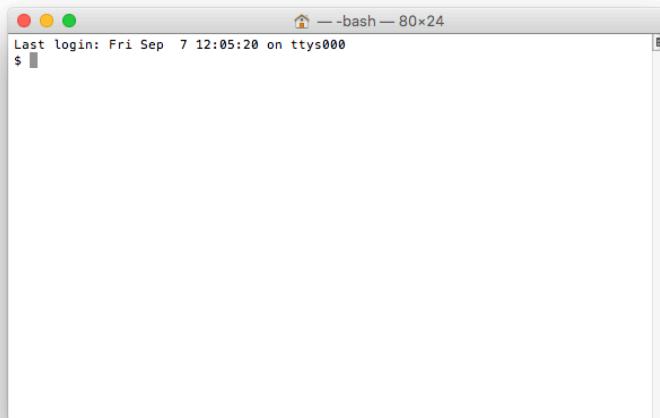


▲図 2.41 このエラーが表示されたら「ホスト名（サーバー IP アドレス）」の IP アドレスを確認しよう

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 2.42）を起動してください。

2.3 SSH でサーバに入ってみよう



▲図 2.42 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 2.43）すれば起動できます。



▲図 2.43 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 回だけ構いません。もし「startSSLKey」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

ようやくサーバを入れたので、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 2.44）普通に改行されますよね。



▲図 2.44 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す。

```
$ sudo su -
```

と書いて Enter キーを押すと root になります。（図 2.45）「\$」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 2.45 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

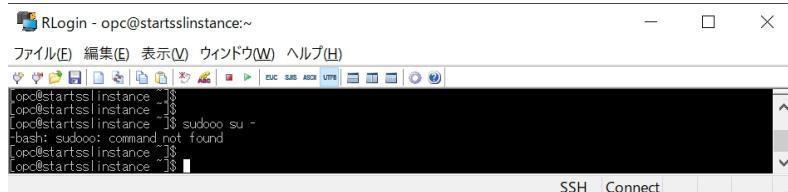
前述の「sudo su -」のようなものを「コマンド」と呼びます。コマンドとはサーバに対して「あれをして」「これをして」と頼む命令のようなものです。

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 2.46）が表示されました。



▲図 2.46 「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。

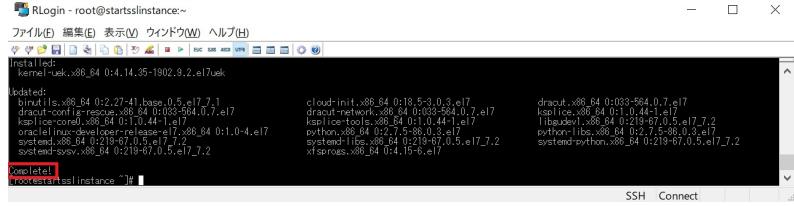
2.4.2 ミドルウェアをインストール

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 2.47) 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 2.47 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^{*14}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-1.16.2-1.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。

^{*14} 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しれませんが、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、サーバを再起動した場合も NGINX が自動で立ち上がりてくるよう、自動起動の設定もオンにしておきましょう。systemctl コマンドで、NGINX の自動起動を有効 (enable) します。

```
# systemctl enable nginx  
# systemctl is-enabled nginx  
enabled ←有効になったことを確認
```

2.4.3 Firewalld で HTTP と HTTPS を許可しよう

続いてサーバの中で動いているファイアウォールの設定を変更します。まずは現状、何がファイアウォールを通れるようになっているのか確認してみましょう。

```
# firewall-cmd --list-services  
dhcpcv6-client ssh
```

dhcpcv6-client と ssh は通つていいけれど、それ以外は誰であろうと通さないぞ！ という設定になっています。このままではブラウザでサイトを見ようとしても、ウェブページを返してくれるはずの NGINX までリクエストが届きません。そこで次のように、許可対象に http と https を追加して、変更が反映されるよう再読み込みします。それぞれ [success] と表示されたら成功しています。

```
# firewall-cmd --add-service=http --permanent  
success  
  
# firewall-cmd --add-service=https --permanent  
success  
  
# firewall-cmd --reload  
success
```

これでファイアウォールの設定が変更できたので、もう一度、誰がファイアウォールを通してもらえるのか確認してみましょう。http と https が追加されていれば問題ありません。

```
# firewall-cmd --list-services  
dhcpcv6-client http https ssh
```

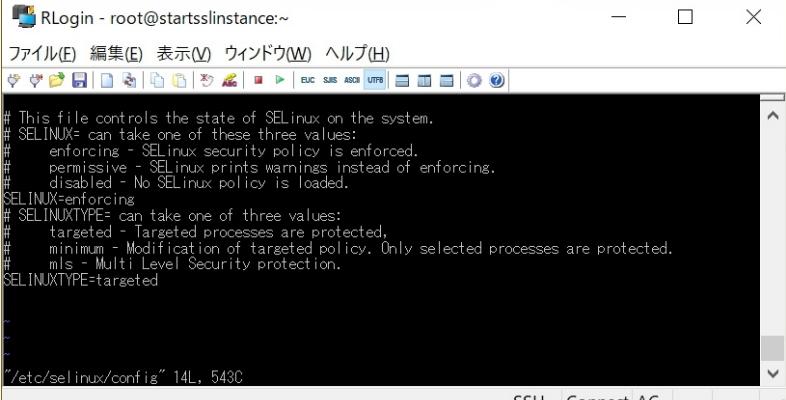
2.4.4 SELinux を無効にしておこう

SELinux を無効化 (disabled) します。

```
# getenforce  
Enforcing ←起動直後は有効になっている  
  
# vi /etc/selinux/config
```

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 2.48）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。

2.4 ターミナルでサーバを操作・設定してみよう



RLogin - root@startsslinstance:~

ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)

This file controls the state of SELinux on the system.
SELINUX= can take one of these three values:
enforcing - SELinux security policy is enforced.
permissive - SELinux prints warnings instead of enforcing.
disabled - No SELinux policy is loaded.
SELINUX=enforcing
SELINUXTYPE= can take one of three values:
targeted - Targeted processes are protected.
minimum - Modification of targeted policy. Only selected processes are protected.
mls - Multi Level Security protection.
SELINUXTYPE=targeted

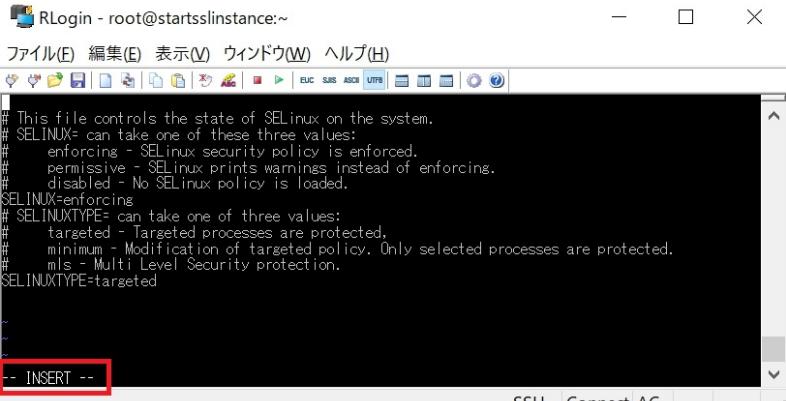
"/etc/selinux/config" 14L, 543C

SSH Connect AC

This screenshot shows a terminal window titled "RLogin - root@startsslinstance:~". The window contains the SELinux configuration file, "/etc/selinux/config". The file's content is displayed in a monospaced font. At the bottom of the terminal window, there are two tabs: "SSH" and "Connect AC".

▲図 2.48 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」*15に変わります。(図 2.49) 左下に「-- INSERT --」と表示されていたら「編集モード」です。



RLogin - root@startsslinstance:~

ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)

This file controls the state of SELinux on the system.
SELINUX= can take one of these three values:
enforcing - SELinux security policy is enforced.
permissive - SELinux prints warnings instead of enforcing.
disabled - No SELinux policy is loaded.
SELINUX=enforcing
SELINUXTYPE= can take one of three values:
targeted - Targeted processes are protected.
minimum - Modification of targeted policy. Only selected processes are protected.
mls - Multi Level Security protection.
SELINUXTYPE=targeted

-- INSERT --

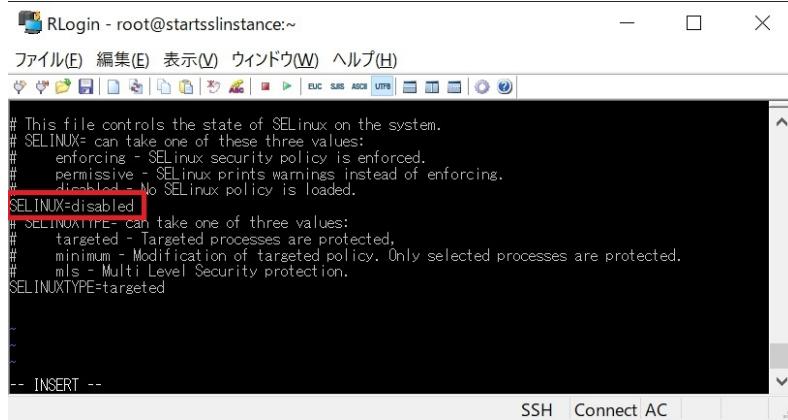
SSH Connect AC

This screenshot shows the same terminal window as in Figure 2.48, but now in edit mode. The status bar at the bottom left displays "-- INSERT --", indicating that the user is in insert mode. The rest of the screen shows the SELinux configuration file content.

▲図 2.49 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「SELINUX=enforcing」を「SELINUX=disabled」(図 2.50) に書き換えてください。

*15 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。



RLogin - root@startsslinstance:~

ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)

This file controls the state of SELinux on the system.
SELINUX= can take one of these three values:
enforcing - SELinux security policy is enforced.
permissive - SELinux prints warnings instead of enforcing.
disabled - No SELinux policy is loaded.
SELINUX=disabled
SELINUXTYPE= can take one of three values:
targeted - Targeted processes are protected,
minimum - Modification of targeted policy. Only selected processes are protected.
mls - Multi Level Security protection.
SELINUXTYPE=targeted

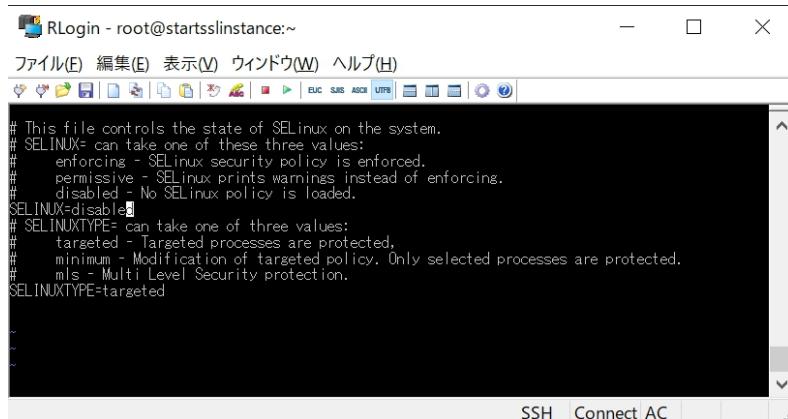
-- INSERT --

SSH Connect AC

This screenshot shows a terminal window titled "RLogin - root@startsslinstance:~". The window contains a text editor displaying the SELinux configuration file. The line "SELINUX=disabled" is highlighted with a red box. Below the text editor, the status bar shows "SSH Connect AC".

▲図 2.50 「SELINUX=enforcing」を「SELINUX=disabled」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 2.51)



RLogin - root@startsslinstance:~

ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)

This file controls the state of SELinux on the system.
SELINUX= can take one of these three values:
enforcing - SELinux security policy is enforced.
permissive - SELinux prints warnings instead of enforcing.
disabled - No SELinux policy is loaded.
SELINUX=disabled
SELINUXTYPE= can take one of three values:
targeted - Targeted processes are protected,
minimum - Modification of targeted policy. Only selected processes are protected.
mls - Multi Level Security protection.
SELINUXTYPE=targeted

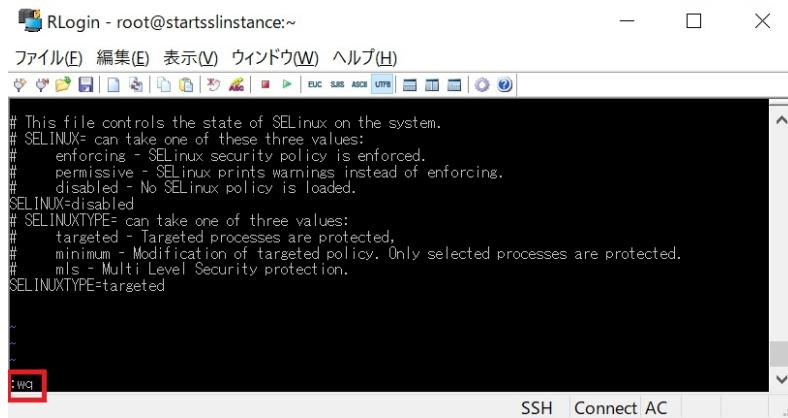
SSH Connect AC

This screenshot shows the same terminal window as in Figure 2.50, but the "INSERT" mode indicator at the bottom left has disappeared, indicating that the changes have been saved and the window is now in "View Mode".

▲図 2.51 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^{*16}」と入力して Enter キーを押せば変更が保存されます。(図 2.52)

^{*16} 書き込んで (write)、抜ける (quit) という命令なので wq です。



▲図 2.52 「: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!」^{*17}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^{*18}でファイルの中身を確認してみましょう。

```
# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

以上で SELinux を無効化する設定は終わりです。

^{*17} 保存せずに強制終了 (quit!) という命令なので q!です。

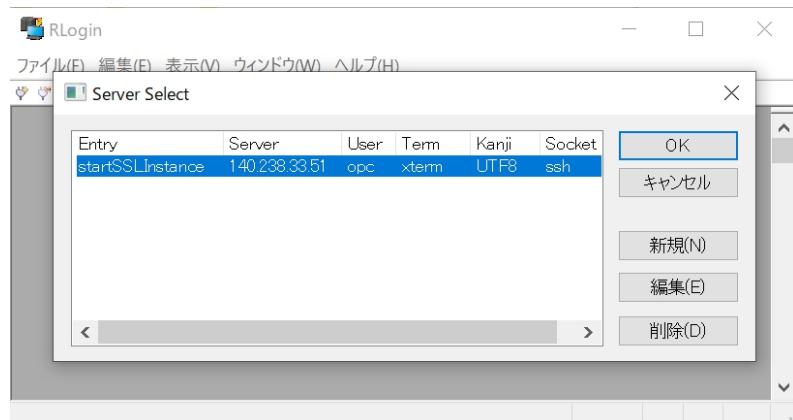
^{*18} cat は猫ではなく「conCATenate files and print on the standard output」の略です

2.4.5 OS 再起動してみよう

変更した設定を反映させるため 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 2.53）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 2.53 さっきと同じ設定で接続してみよう

以上で「サーバを立てる」という作業はおしまいです。

2.4.6 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^{*19}を開いて移動するときはダブルクリックの代わりに cd^{*20}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。

皆さんのがいま使っている 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。^{*21}

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。

2.4.7 なぜかサイトが見られない

ウェブサーバも立てたし、SELinux はとめたし、サーバの中のファイアウォールに穴も空けました。これで準備完了！ サーバを立てたときにメモした [パブリック IP アドレス] を、ブラウザで開いてみました。するとしばらくぐるぐるした後で、[接続がタイムアウトしました] と表示されてしまいました。（図 2.54）

^{*19} Linux ではフォルダのことをディレクトリと呼びます。

^{*20} change directory の略。

^{*21} Mac のターミナルはそもそも黒じゃなくて白ですね。



▲図 2.54 なぜか HTTP でサイトが表示されない…

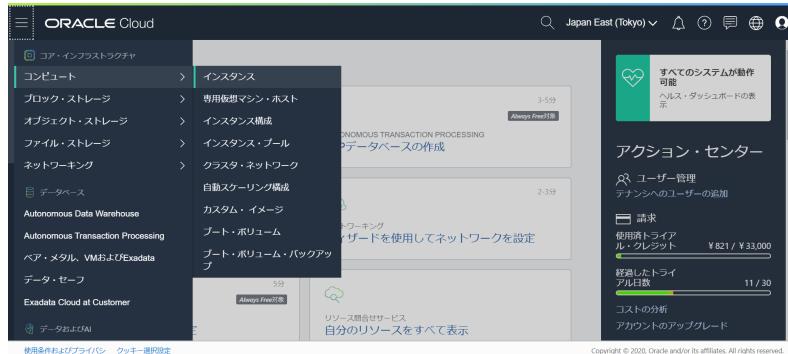
403 や 404 や、あるいは 502 などのステータスコードも返ってきていないので、そもそも NGINX までたどり着けていないようです。

2.4.8 サーバの手前にあるファイアウォールにも穴をあけよう

さきほどサーバの中にあるファイアウォールの設定を変更して、HTTP と HTTPS が通れるようにしましたが、実はサーバの中だけでなく、サーバの外にももう 1 つファイアウォールがいます。サイトが表示されなかったのは、「ウェブページを見せて！」というリクエストが、サーバの手前のファイアウォールで阻まれていたためなのです。サーバの手前にあるファイアウォールにも穴を空けて、HTTP と HTTPS がサーバまでたどり着けるようにしましょう。

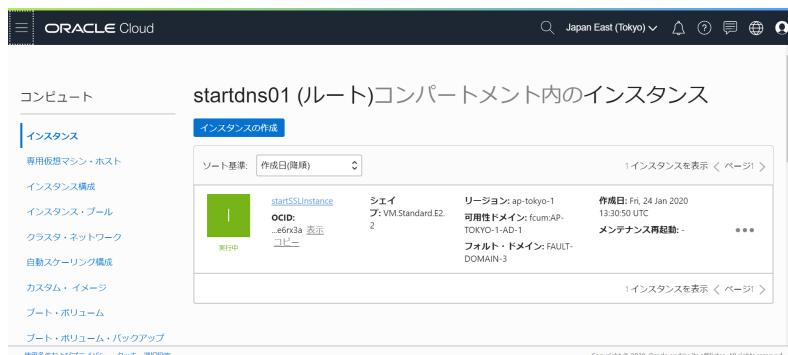
再び Oracle Cloud のコンソールに戻って、左上の [ハンバーガーメニュー] から [コンピュート] の [インスタンス] を開きます。(図 2.55)

2.4 ターミナルでサーバを操作・設定してみよう



▲図 2.55 [ハンバーガーメニュー] から [コンピュート] の [インスタンス] を開く

インスタンスの一覧が表示されるので [startSSLInstance] をクリックします。(図 2.56)



▲図 2.56 [startSSLInstance] をクリック

[パブリック・サブネット] をクリックします。(図 2.57)

第2章 Oracle Cloud でサーバを立てよう

The screenshot shows the Oracle Cloud Instances page. A green server icon is displayed with the status "実行中" (Running). The "Instances Information" tab is selected. The "Instance Information" section contains the following details:

- 可用性ドメイン: (run)AP-TOKYO-1-AD-1
- フォルト・ドメイン: FAULT-DOMAIN-3
- リージョン: ap-tokyo-1
- シティ: VM.Standard.E2.2
- 仮想クラウド・ネットワーク: VirtualCloudNetwork-20200120-2319
- メンテナンス再起動: -

The "Primary VNIC Information" section shows:

- プライベートIPアドレス: 10.0.0.2
- パブリックIPアドレス: 140.238.33.51
- ネットワーク・セキュリティ・グループ: None (無効)

Other visible information includes:

- イメージ: Oracle-Linux-7.7-20191218-0
- OCI ID: ocid1.instance.oc1.ap-tokyo-1.241... (link)
- 起動日: Fri, 24 Jan 2020 13:30:50 UTC
- コンバートメント: startdns01 (ルート)
- 起動モード: PARAVIRTUALIZED

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

▲図 2.57 [パブリック・サブネット] をクリック

もう一度、[パブリック・サブネット] をクリックします。(図 2.58)

The screenshot shows the Oracle Cloud Subnets page. A green server icon is displayed with the status "使用可能" (Available). The "startdns01 (ルート) ルート" section displays:

- CIDRブロック: 10.0.0.0/16
- コンバートメント: startdns01 (ルート)
- 作成日: 2020年1月20日(月) 14:55:54 UTC

The "サブネットの作成" (Create Subnet) section shows:

名前	状態	CIDRブロック	サブネット・アクセス	作成日
[パブリック・サブネット]	● 使用可能	10.0.0.0/24	パブリック (リージョナル)	2020年1月20日(月) 14:55:56 UTC

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

▲図 2.58 もう一度、[パブリック・サブネット] をクリック

[セキュリティ・リスト] の中にある [Default Security List for VirtualCloudNetwork ~] をクリックします。(図 2.59) このセキュリティ・リストが、サーバの手前にいるファイアウォールです。

2.4 ターミナルでサーバを操作・設定してみよう

The screenshot shows the Oracle Cloud interface for managing security lists. The top navigation bar includes the Oracle Cloud logo, search bar, and account information for Japan East (Tokyo). The main content area is titled "セキュリティ・リスト" (Security List) and displays the "Default Security List for VirtualCloudNetwork-20200120-2319". Key details shown include CIDR ブロック: 10.0.0.0/4, 仮想ルーターMACアドレス: 00:00:17:42:14:84, and DNSドメイン名: subnet... 条件 ルート。A table lists the security rules:

名前	状態	コンバートメント	作成日
Default Security List for VirtualCloudNetwork-20200120-2319	● 使用可能	startdns01 (ルート)	2020年1月20日(月) 14:55:54 UTC

Below the table, a note says "1アイテムを表示中 < ページ1 >". The bottom right corner contains the copyright notice: "Copyright © 2020, Oracle and/or its affiliates. All rights reserved."

▲図 2.59 [Default Security List for VirtualCloudNetwork～] をクリック

[イングレス・ルール] で、[イングレス・ルールの追加] をクリックします。(図 2.60)

The screenshot shows the Oracle Cloud interface for creating an ingress rule. The top navigation bar includes the Oracle Cloud logo, search bar, and account information for Japan East (Tokyo). The main content area is titled "イングレス・ルール" (Ingress Rule) and shows the "Default Security List for VirtualCloudNetwork-20200120-2319". A table lists existing rules:

ステータス	ソース	IPプロトコル	ソース・ポート範囲	宛先ポート範囲	タイプとコード	許可	説明
いいえ	0.0.0.0/0	TCP	All	22		●	ポートのTCPトラフィック: 22 SSH Remote Login Protocol
いいえ	0.0.0.0/0	ICMP		3, 4		●	次に対するICMPトラフィック: 3, 4宛先に到達できません: ループアタックが発生する場合が あります。

The bottom right corner contains the copyright notice: "Copyright © 2020, Oracle and/or its affiliates. All rights reserved."

▲図 2.60 [イングレス・ルールの追加] をクリック

[ソース CIDR] に [0.0.0.0/0]^{*22}を入力します。[宛先ポート範囲] には [80,443]^{*23}を入力します。(図 2.61) どちらも入力できたら、[イングレス・ルールの追加] をクリックします。

*²² ソース CIDR は接続元の IP アドレス範囲のこと、0.0.0.0/0 はすべての IP アドレスを指します。つまり接続元がどんな IP アドレスでもファイアウォールを通れます、ということですね

*²³ ポート番号とは、サーバという家や、その手前のファイアウォール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

第2章 Oracle Cloud でサーバを立てよう



▲図 2.61 [ソース CIDR] に [0.0.0.0/0]、[宛先ポート範囲] には [80,443] を入力

[イングレス・ルール] に、HTTP（80 番ポート）と HTTPS（443 番ポート）へのリクエストを通す設定が追加されました。（図 2.62）



▲図 2.62 ルールが追加された！

2.4.9 HTTP でサイトを見てみよう

サーバを立てたときにメモした [パブリック IP アドレス] を、ブラウザで開いてみましょう。（図 2.63）

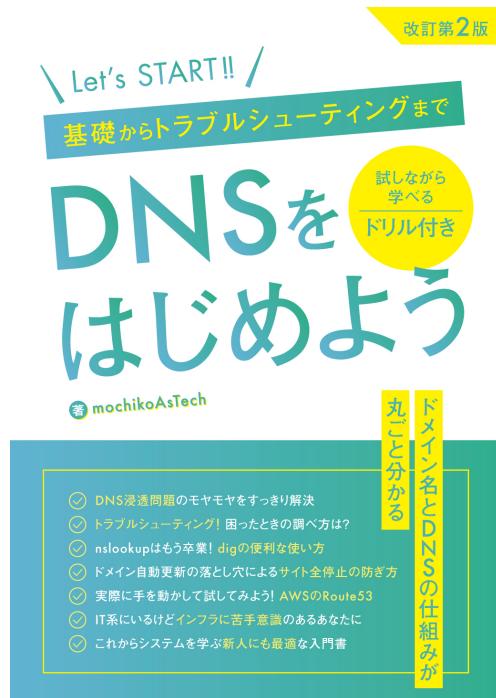


▲図 2.63 HTTP でサイトが見られた！

[Welcome to nginx!] と表示されました！ これでまず、「HTTP でサイトを表示する」はクリアです。

2.5 ドメイン名の設定

サーバの準備ができたので、HTTPS のサイト用にドメイン名を用意します。自分のドメイン名？ そんなの持っていないよ！ という人は、先に「DNS をはじめよう」(図 2.64)で、ドメイン名を買ってからこの先へ進むようにしてください。



▲図 2.64 「DNS をはじめよう」(1,000 円) は BOOTH や Amazon (Kindle) で好評発売中

あなたが買ったドメイン名を使って「ssl. 自分のドメイン名」の A レコードを作成して、サーバの [パブリック IP アドレス] と紐付けてください。ネームサーバはお名前.com を使用してもいいですし、AWS の Route53 で設定しても構いません。

なお筆者が「DNS をはじめよう」で購入したのは `startdns.fun` というドメイン名だったので、`ssl.startdns.fun` という A レコードを作って、さっそく立てたばかりのサーバの [パブリック IP アドレス] と紐付けます。例えばネームサーバが「お名前.com」なら、DNS 設定の画面でこのように A レコードを追加します。(図 2.65)

ホスト名	TYPE	TTL	VALUE	優先	状態	追加
ssl .startdns.fun	A ▾	3600	140 238 33 51		有効 ▾	追加

▲図 2.65 「ssl. 自分のドメイン名」の A レコードを作成する

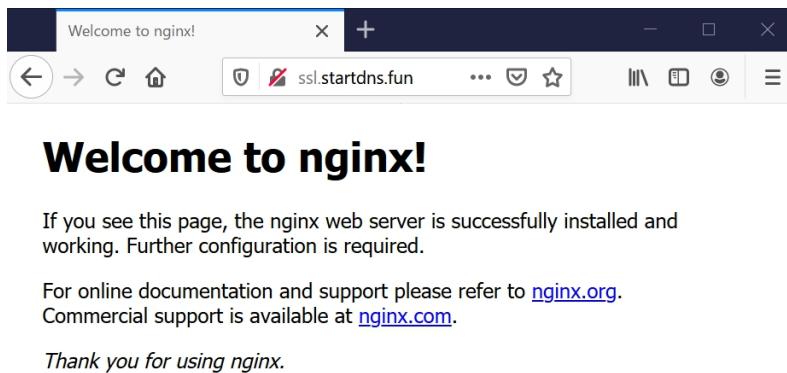
A レコードができたかどうかは、次の dig コマンドで確認できます。dig コマンドをたいたいた結果、サーバの IP アドレスが返ってくれれば A レコードは設定できています。

```
$ dig ssl. 自分のドメイン名 a +short
```

筆者の場合は、次のように表示されました。

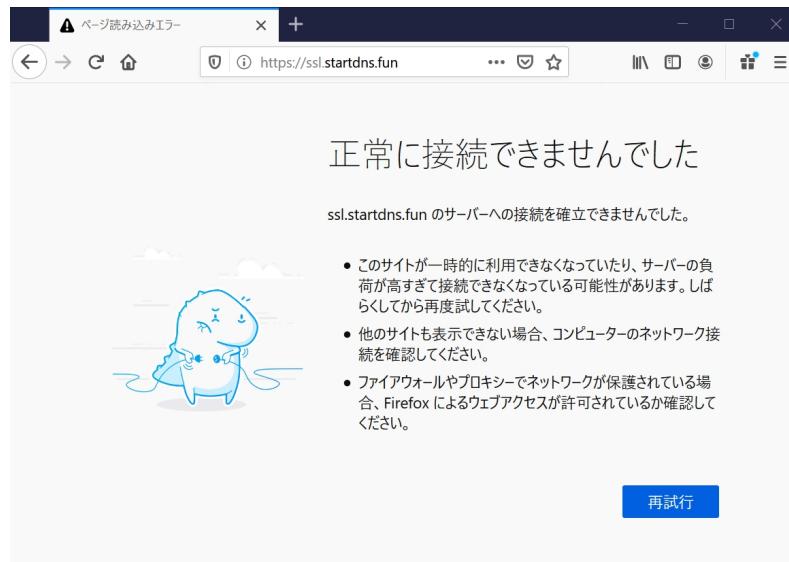
```
$ dig ssl.startdns.fun a +short  
140.238.33.51
```

ドメイン名が設定できたら、ブラウザでも「http://ssl. 自分のドメイン名」を叩いてみましょう。先ほどと同じ NGINX のページが表示されるはずです。（図 2.66）



▲図 2.66 「ssl. 自分のドメイン名」でサイトが表示された！

HTTP でサイトを見ることができましたが、同じドメイン名を HTTPS で開いてみるとどうでしょう？ ブラウザで「https://ssl. 自分のドメイン名」を開いてみると、「正常に接続できませんでした」と表示（図 2.67）されました。



▲図 2.67 HTTPS で開くと [正常に接続できませんでした] と表示された

それでは HTTPS でもサイトが見られるように、SSL 証明書を取得して設定をしていきましょう。

2.6 SSL 証明書を取得しよう

HTTP でサイトが見られたので、今度は HTTPS でも見られるよう、必要な材料を「SSL 証明書」を入手しましょう。

登場人物が多いので、先に概要を紹介しておきます。

2.6.1 SSL 証明書にまつわる登場人物

最初に登場人物全員が集まる場所として、/etc/nginx/の下に ssl というディレクトリを作成しておきます。^{*24}

^{*24} もし `mkdir: cannot create directory '/etc/nginx/ssl': Permission denied` と表示されてしまったら、あなたは今うっかり一般ユーザのまま mkdir コマンドを実行しています。コマンドの例で、左側のプロンプトが「#」のときは、root で実行してください。「sudo su -」と書いて Enter キーを押すと root になります

```
# mkdir /etc/nginx/ssl/
# cd /etc/nginx/ssl/
```

秘密鍵（startssl.key）を作ろう

先ず最初に作成するのが秘密鍵です。秘密鍵は「.key」や「.pem」^{*25}という拡張子で作成されることが多いです。秘密鍵は名前のとおり「秘密」にすべきです。つまり、限られた人だけが触れるように管理すべきです。決してメールに添付して送ったり、サーバ内で誰でも見られる/tmp/以下に置いたりしてはいけません。

次の openssl コマンドを叩いて、秘密鍵を生成しましょう。左から順に「openssl コマンドで、鍵アルゴリズムが RSA で、鍵の長さは 2048 ビットで、/etc/nginx/ssl/以下に startssl.key というファイル名で秘密鍵を作って」という意味です。

```
# openssl genrsa 2048 >/etc/nginx/ssl/startssl.key
Generating RSA private key, 2048 bit long modulus
.....+ ++
e is 65537 (0x10001)
```

できあがった秘密鍵を、cat コマンドで見てみましょう。

```
# cat /etc/nginx/ssl/startssl.key
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAO+s/Gzdb0bzg6QWzCvK5JofMv6izHzlCfMCMhcU7SeBd2tHN
icRA7g5CZq09aaEqv1949cFX5C3bgHx140+epeudrKyUjRwZSpS70mznDBFQByTY
(中略)
InsCw9qu+iZknMKiISw3Krht/898/hq0jqLFJUTbfg9BP8w+JVW4+8hp40Sklymc
NRcvPYUBQy3wK+w527rksodkGZ77c6Q+XxRtH/wpo3H+xwhmJvi+T2o=
-----END RSA PRIVATE KEY-----
```

【コラム】SSL 証明書の秘密鍵にパスフレーズは設定すべき？

openssl コマンドで秘密鍵を作るとき、-aes128 や-aes256 というオプションを付けると、パスフレーズを聞かれて、指定の暗号で暗号化された秘密鍵が output されます。^{*26} [Enter pass phrase:] や [Verifying - Enter pass phrase:] と表示された際に、いくらキーボードを叩いて入力しても、黒い画面上は何も表示されま

^{*25} 秘密鍵が PEM と呼ばれるテキスト形式で生成されることから、pem という拡張子が使われるようです

せん。ですが、ちゃんと文字入力はできているので大丈夫です。入力を間違えたら Backspace キーで消して、書き直すこともできます。

```
# openssl genrsa -aes128 2048 >/etc/nginx/ssl/with-passphrase.key
Generating RSA private key, 2048 bit long modulus
(中略)
Enter pass phrase: ←秘密鍵のパスフレーズとして「startssl」と入力
Verifying - Enter pass phrase: ←もう一度「startssl」と入力
```

秘密鍵がパスフレーズで保護されると、秘密鍵を盗んだ誰かが使おうとしても、パスフレーズが分からなければ使えないで安心です。

しかし秘密鍵にパスフレーズが設定されていると、Apache や Nginx といったウェブサーバを再起動した際にも、必ずパスフレーズを聞かれます。もし何かトラブルがあってサーバが OS ごと再起動してしまった場合、折角 Nginx が自動起動する設定になっていても、パスフレーズを聞くところで止まって起動できず、サイトが自動復旧しない、というトラブルが発生します。

これを回避するには、パスフレーズをファイルに書いておいて、Nginx の自動起動時にそれを読み込むようにする、という方法があります。しかし折角設定したパスフレーズをファイルに書いて同じウェブサーバ内に置いてしまうと、秘密鍵を盗むとき一緒にパスフレーズのファイルも盗まれてしまう可能性が高くなり、もはやパスフレーズを設定した意味がありません。そのため筆者は、ウェブサーバに設置して使う秘密鍵にはパスフレーズは設定しなくてよい、という考えです。

ちなみに、次のように一度パスフレーズありで作った秘密鍵を、パスフレーズなしで複製することも可能です。

```
# cd /etc/nginx/ssl/
# openssl rsa -in with-passphrase.key -out without-passphrase.key
Enter pass phrase for with-passphrase.key: ←秘密鍵のパスフレーズ
「startssl」を入力
writing RSA key
```

利便性を保つつつ安全性も向上させたければ、本番のウェブサーバで使う秘密鍵はパスフレーズなし、バックアップサーバで保管しておく秘密鍵はパスフレーズありにしておく、という方法がいいでしょう。

CSR (startssl.csr) を作ろう

秘密鍵ができたら、続いて CSR (Certificate Signing Request) の作成に進みましょう。CSR は「.csr」という拡張子で作成されることが多いです。CSR は「Certificate Signing Request (証明書署名リクエスト)」という名前のとおり、認証局に対して「こういう内容で SSL 証明書を作って署名して」とリクエストする、申請書のようなものです。

左から順に、「openssl コマンドで、CSR を、新しく、秘密鍵は startssl.key で、startssl.csr というファイル名で作って」という意味です。

```
# cd /etc/nginx/ssl/
# openssl req -new -key startssl.key -out startssl.csr
```

CSR を作成するため、いくつか質問をされます。(表 2.3)*27

▼表 2.3 CSR 作成時に聞かれる質問

必須/任意	質問	説明	入力する内容
必須	Country Name	国名 (C)	JP
必須	State or Province Name	都道府県名 (S/ST)	Tokyo
必須	Locality Name	市町村名 (L)	Shinjuku-ku
必須	Organization Name	組織名 (O)	mochikoAsTech
任意	Organizational Unit Name	組織単位名 (OU)	入力なし
必須	Common Name	コモンネーム (CN)	ssl.startdns.fun

```
Country Name (2 letter code) [XX]:JP
State or Province Name (full name) []:Tokyo
Locality Name (eg, city) [Default City]:Shinjuku-ku
Organization Name (eg, company) [Default Company Ltd]:mochikoAsTech
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:ssl.startdns.fun
```

次の 3 つは入力不要です。

*27 さっきは-aes128 や-aes256 といった「どの暗号で暗号化するか?」のオプションを何も指定しなかったので、パスフレーズは聞かれず、秘密鍵も暗号化されませんでした

*27 本著では DV 証明書を取得するため「組織単位名 (OU)」は任意としていますが、証明書の種類や認証局によっては必須のところもあるようです。取得時に認証局のサイトで確認しましょう。DV 証明書については後述します

```
Email Address []:
A challenge password []:
An optional company name []:
```

質問に全て答えると、CSR ができあがります。「Subject」と書かれた行を見て、生成された CSR の内容が正しいか確認しましょう。

```
# openssl req -text -in /etc/nginx/ssl/startssl.csr -noout | head -4
Certificate Request:
Data:
Version: 0 (0x0)
Subject: C=JP, ST=Tokyo, L=Shinjuku-ku, O=mochikoAsTech, CN=ssl.startdns.fu
```

CSR を cat コマンドで表示して、「-----BEGIN CERTIFICATE REQUEST-----」から「-----END CERTIFICATE REQUEST-----」までをメモしておきましょう。メモした CSR はこの後使用します。

```
# cat /etc/nginx/ssl/startssl.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICqzCCAZMCAQAwZjELMAkGA1UEBhMCS1AxDjAMBgNVBAgMBVRvA3lvMRQwEgYD
VQQHDAtTaGluanVrdS1rdTEWMBQGA1UECgwNbW9jaGlrboFzVGVjaDEZMBcGA1UE
(中略)
jP1RMQS3PuYDE6QIVJ5zbMC+RIydSQ/ODr9VUHWiYqDPjx+BpphYT5AxMwbw9/m5
noKGvJl1Mt7G03Awa/TX
-----END CERTIFICATE REQUEST-----
```

2.6.2 【ドリル】CSR で入力すべきなのはクライアントの情報？

問題

A 銀行のウェブサイトを HTTPS で作ることになりました。SSL 証明書^{*28}の取得は A 銀行の代わりに広告代理店の B 社が行い、さらにサイトの制作や運用は A 銀行から Web 制作会社の C 社に委託する場合、CSR で入力する住所や会社名は A 銀行・B 社・C 社のどれにすべきでしょうか？

- A. A 銀行のウェブサイトなんだから A 銀行を入力すべき
- B. A 銀行から任されて SSL 証明書を買うのは B 社だから B 社を入力すべき
- C. 実際にサイトの管理を任せているのは C 社だから C 社を入力すべき

^{*28} SSL 証明書の種類は「EV 証明書」とします。EV 証明書については後述します

答え _____

解答

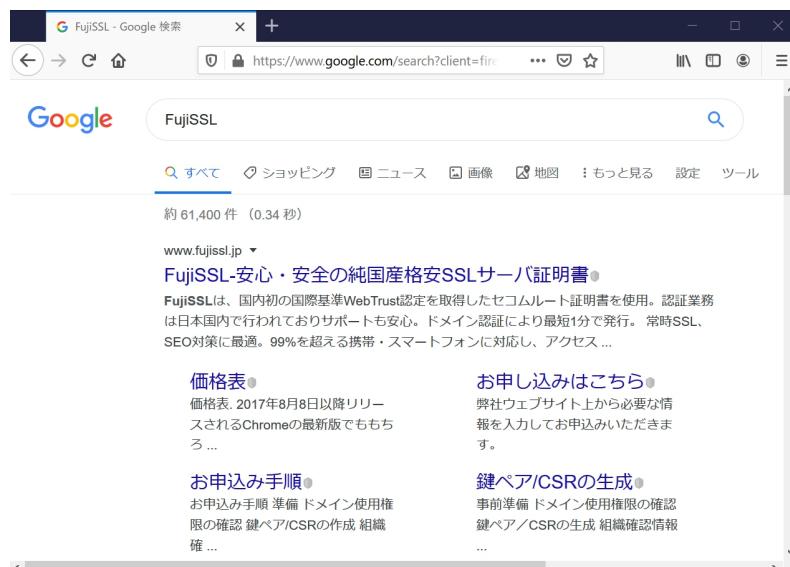
正解は A です。ウェブサイトの運営元が A 銀行であることを証明するための SSL 証明書なので、CSR では A 銀行（クライアント）の情報を記載すべきです。

A 銀行のフィッシングサイトが出てきたときに、エンドユーザが「本物のサイトか確認しよう」と思って証明書の情報を見て、B 社や C 社の情報が表示されたら「A 銀行じゃない！」となってしまいます。

こんなあった。[https://docs.oracle.com/en/operating-systems/oracle-linux/8/nginx-install/index.html#ConfigureFirewallRules\(Optional\)](https://docs.oracle.com/en/operating-systems/oracle-linux/8/nginx-install/index.html#ConfigureFirewallRules(Optional))

2.6.3 証明書の取得申請

[FujiSSL] で検索して、[FujiSSL-安心・安全の純国産格安 SSL サーバ証明書] をクリック（図 2.68）します。



▲図 2.68 [FujiSSL-安心・安全の純国産格安 SSL サーバ証明書] をクリック

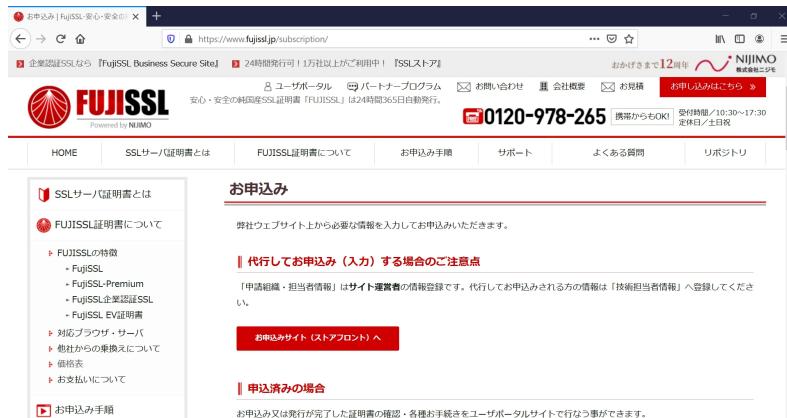
第2章 Oracle Cloud でサーバを立てよう

右上の「お申し込みはこちら」をクリック（図 2.69）します。



▲図 2.69 「お申し込みはこちら」をクリック

続いて「お申し込みサイト（ストアフロント）へ」をクリック（図 2.70）します。



▲図 2.70 「お申し込みサイト（ストアフロント）へ」をクリック

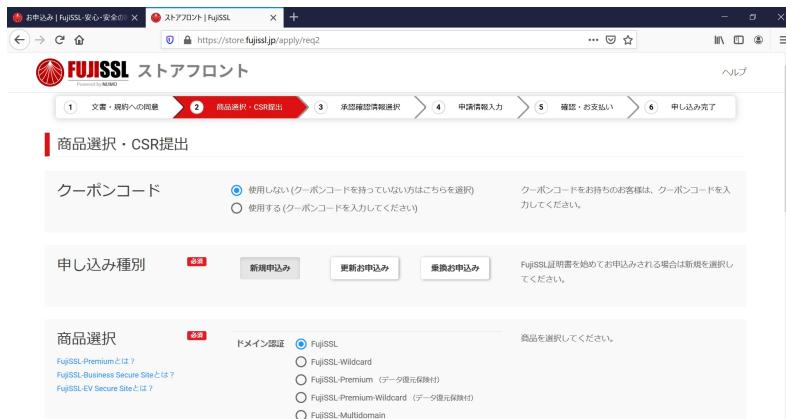
指定された文書と「収集した個人情報の利用目的」を確認した上で、チェックボックスにチェックを入れて「次へ」をクリック（図 2.71）します。

2.6 SSL 証明書を取得しよう



▲図 2.71 チェックを入れて [次へ] をクリック

クーポンコードは [使用しない]、申し込み種別は [新規申し込み]、商品選択は [ドメイン認証] の [FujiSSL] になっていることを確認（図 2.72）します。



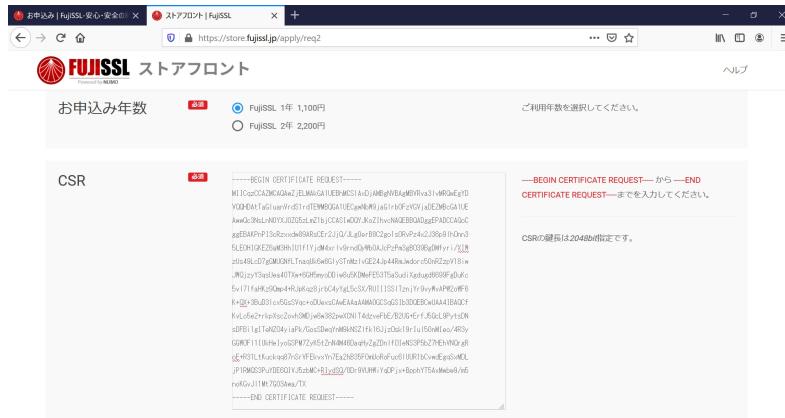
▲図 2.72 クーポンコード、申し込み種別、商品選択を確認

下へスクロールして、お申込み年数が [FujiSSL 1 年 1,100 円]^{*29} になっていることを確認（図 2.73）したら、CSR に、さきほど「-----BEGIN CERTIFICATE REQUEST-----」から「-----END CERTIFICATE REQUEST-----」までをメモしておいた CSR をペース

^{*29} 2019 年 2 月現在、FujiSSL の「ドメイン認証シングルタイプ」という SSL 証明書は、有効期間 1 年で税込 1,100 円です

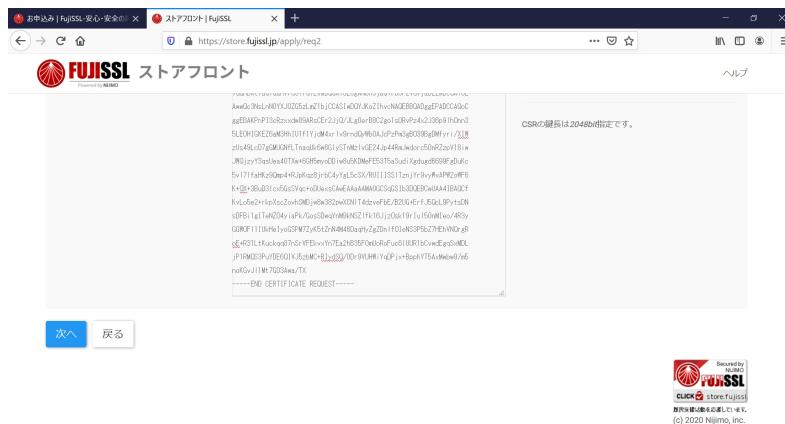
第2章 Oracle Cloud でサーバを立てよう

トします。



▲図 2.73 CSR をペーストする

[次へ] をクリック（図 2.74）してください。



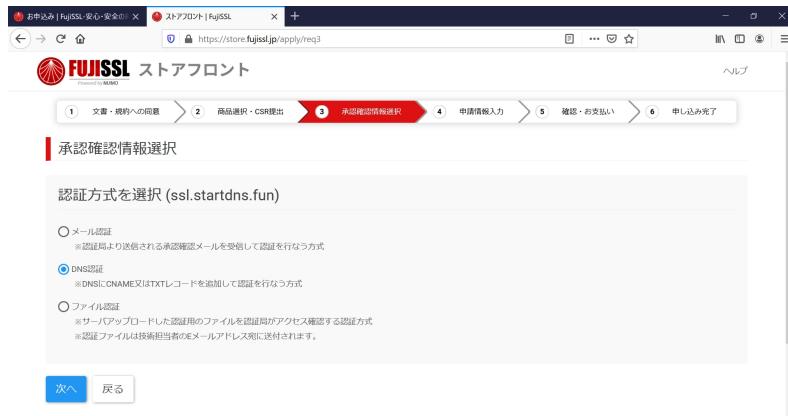
▲図 2.74 CSR をペーストする

認証方式は「DNS 認証」を選択（図 2.75）してください。これは「CSR のコモンネームで指定したドメイン名が、あなたの持ち物であることをどうやって証明しますか？」ということを聞かれています。対象のドメイン名で、メールを受信して URL を踏むか、指定された内容のリソースレコードを DNS で追加するか、サイトに指定された内容のファイルをアップするか、いずれかの方法で「このドメイン名（筆者なら ssl.startdns.fun）は

2.6 SSL 証明書を取得しよう

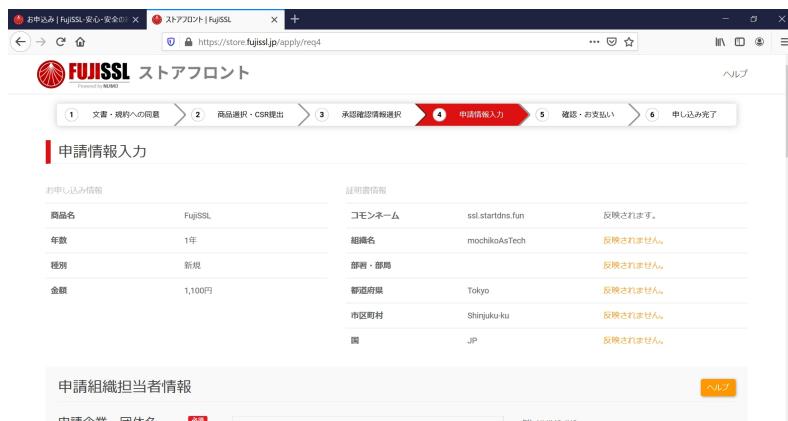
私の持ち物です」ということを証明しなければいけません。

今回は DNS で TXT レコードを追加する、という方法で証明するので、[DNS 認証] を選択して [次へ] をクリックします。



▲図 2.75 [DNS 認証] を選択して [次へ] をクリック

今回購入するのは DV 証明書です。DV 証明書は「そのドメイン名の使用権があること」だけを証明してくれます。サイトの運営者が日本にいることや、東京の新宿区にオフィスがあること、mochikoAsTech という組織であることなどは確認も証明もしないので、実際の証明書にもそれらの情報は反映されません。(図 2.76)



▲図 2.76 [DNS 認証] を選択して [次へ] をクリック

【申請組織担当者情報】と【技術担当者情報】を英語表記で入力（図 2.77）します。ここで入力した住所や電話番号、個人名などは外向けに公開されることはありませんので、安心して入力してください。^{*30}この後、ここで入力した【E メールアドレス】宛てに、SSL 証明書が送られてきます。メールアドレスを間違えないよう注意してください。すべて入力したら、【次へ】をクリック（図 2.78）します。

▲図 2.77 【申請組織担当者情報】と【技術担当者情報】を入力

▲図 2.78 【次へ】をクリック

^{*30} EV 証明書や OV 証明書の場合は、ここで入力した担当者宛てに実在確認の連絡がります。詳細については後述します

2.6 SSL 証明書を取得しよう

[決済情報入力] に、SSL 証明書代を支払うクレジットカード情報を入力（図 2.79）します。

The screenshot shows the Fujissl storefront interface for applying for an SSL certificate. The main title is '決済情報入力' (Payment Information Input). It displays a total amount of '1,100 円' (1,100 yen) with a note '(税込)' (tax included). Below this, there are fields for credit card type selection (JCB, MasterCard, VISA, American Express, Diners Club, etc.), card number, expiration date (month/year), security code, and cardholder name. A note at the bottom right indicates that the card number should be written on three or four lines. The background shows a light gray '書類送付先' (Document Delivery Address) section.

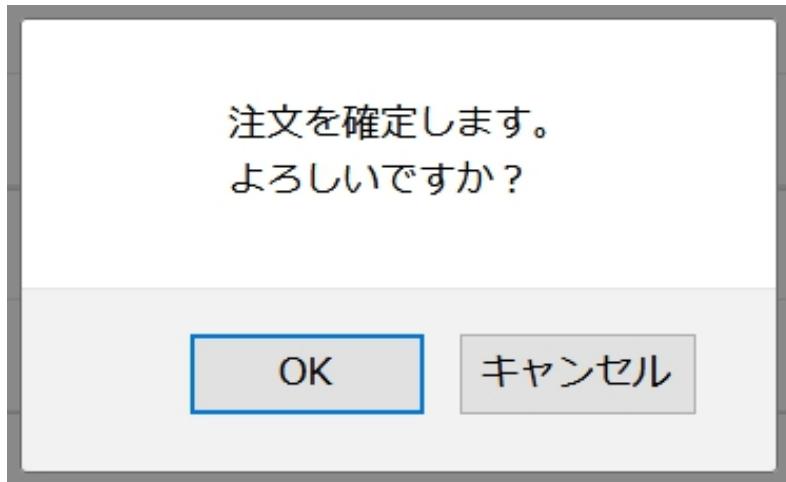
▲図 2.79 クレジットカード情報を入力

[書類送付先] で [請求書宛名]、[納品書宛名]、[領収書宛名] を記入したら、[上記の内容で注文を確定する] をクリック（図 2.80）します。

The screenshot shows the Fujissl storefront interface for entering document delivery addresses. It has three input fields: '請求書宛名' (Billing Statement Recipient) with value 'mochikoAsTech', '納品書宛名' (Delivery Document Recipient) with value 'mochikoAsTech', and '領収書宛名' (Receipt Document Recipient) with value 'mochikoAsTech'. To the right of each field, a note specifies that the document will be sent to the recipient's email address. At the bottom, there are two buttons: '上記の内容で注文を確定する' (Confirm order with above content) and '戻る' (Back).

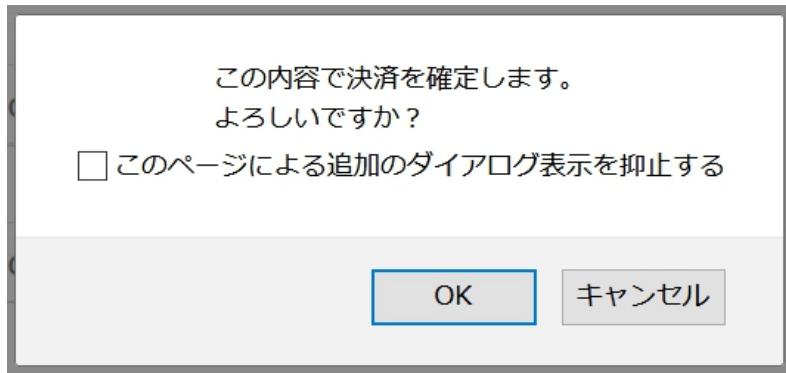
▲図 2.80 [上記の内容で注文を確定する] をクリック

「注文を確定します。よろしいですか？」と表示（図 2.81）されたら、[OK] をクリックします。



▲図 2.81 [OK] をクリック

「この内容で決済を確定します。よろしいですか？」と表示（図 2.82）されます。「1年間有効な SSL 証明書を 1,100 円で買うんだ！」というケツイ^{*31}をしたら、[OK] をクリックします。



▲図 2.82 1,100 円払うケツイをして [OK] をクリック

お申し込み完了のページが表示（図 2.83）されました。先ほど登録したメールアドレス宛に、DNS 設定情報を知らせるメールが届いていますので確認しましょう。

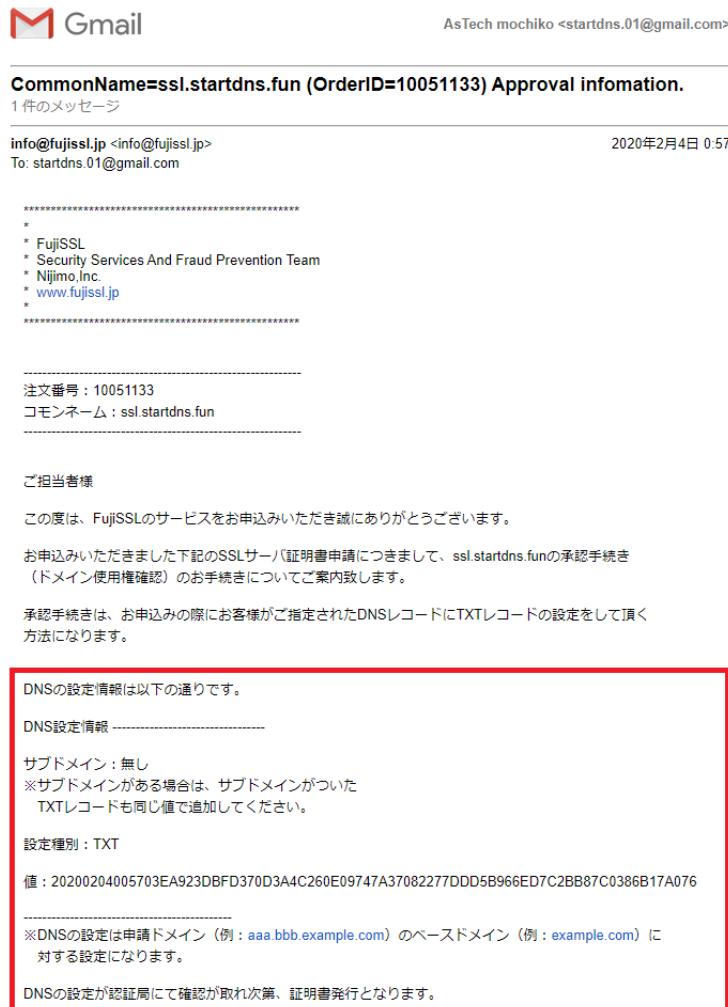
^{*31} UNDERTALE というゲームでは「ケツイを ちからに かえるんだ…！」という台詞が繰り返し出でています

2.6 SSL 証明書を取得しよう



▲図 2.83 1,100 円払うケツイをして [OK] をクリック

[CommonName=ssl. 自分のドメイン名] から始まる件名のメール（図 2.84）がすぐに届きました。「CSR のコモンネームで指定したドメイン名が ssl.startdns.fun の場合、startdns.fun の TXT レコードを追加して、メールに書いてある値を設定するよう書いてあります。



▲図 2.84 追加すべき TXT レコードの値がメールで届いた

例えばネームサーバが「お名前.com」なら、DNS 設定の画面でこのように TXT レコードを追加（図 2.85）します。今回はサブドメインを含まないドメイン名を追加するので、[ホスト名] には何も入力しません。[VALUE] には、メールに書いてあった値をそのままコピペーストします。

2.6 SSL 証明書を取得しよう

A/AAAA/CNAME/MX/NS/TXT/SRV/DS/CAAレコード						
ホスト名	TYPE	TTL	VALUE	優先	状態	追加
.startdns.fun	TXT ▾	3600	20200204005703EA923DBF	有効 ▾	有効	<input type="button" value="追加"/>

▲図 2.85 「自分のドメイン名」の TXT レコードを追加する

TXT レコードができたかどうかは、次の dig コマンドで確認できます。dig コマンドをたたいた結果、サーバの IP アドレスが返ってくれば A レコードは設定できています。

```
$ dig 自分のドメイン名 txt +short
```

筆者の場合は、次のように表示されました。

```
$ dig startdns.fun txt +short  
"20200204005703EA923DBFD370D3A4C260E09747A37082277DDD5B966ED7C2BB87C0386B17A076"
```

TXT レコードを追加してから、おおよそ 30 分後に SSL 証明書がメール（図 2.86）で届きました。

Gmail AsTech mochiko <startdns.01@gmail.com>

(OrderID=10051133) FujiSSL Fulfilment E-mail

2件のメッセージ

info@fujissl.jp <info@fujissl.jp> 2020年2月4日 1:30
To: startdns.01@gmail.com

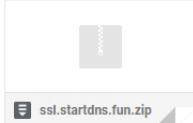
*
* FujiSSL
* Security Services And Fraud Prevention Team
* Nijimo,Inc.
* www.fujissl.jp
*

注文番号 : 10051133
コモンネーム : ssl.startdns.fun

ご担当者様
このたびはFujiSSLのサービスをご利用いただき誠にありがとうございます。
お客様の証明書が発行いたしましたのでご案内いたします。
証明書はこのメールの下部に貼付されております。
当メールの添付ファイルから証明書一式を取得いただくことも可能です。

▲図 2.86 SSL 証明書がメールで届いた

メールに添付されている ZIP ファイル (ssl_自分のドメイン名.zip) に SSL 証明書が入っていますのでダウンロードします。(図 2.87)



▲図 2.87 メールに添付されている ZIP ファイル

Windows の方も Mac の方も、ダウンロードした ZIP ファイルはデスクトップに置いてください。(図 2.88)



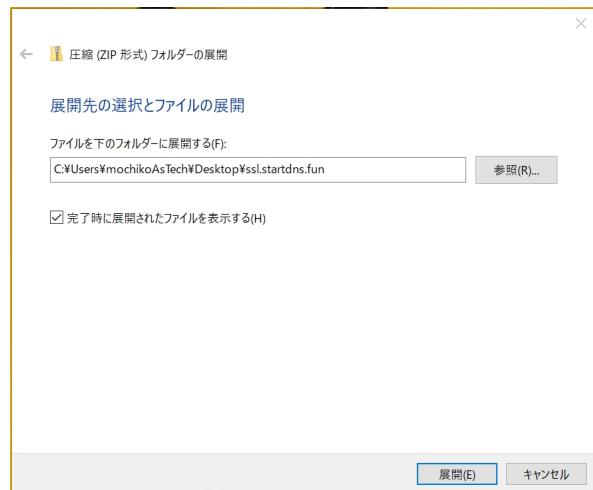
▲図 2.88 ダウンロードした ZIP ファイルはデスクトップに置く

ZIP ファイルを右クリックして、[すべて展開] をクリックします。(図 2.89)



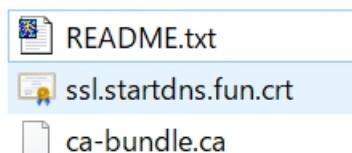
▲図 2.89 右クリックして [すべて展開] をクリック

[展開] をクリックします。(図 2.90)



▲図 2.90 [展開] をクリック

展開したフォルダ（図 2.91）の中の [server.crt] が SSL 証明書で、[ca-bundle.ca] が中間 CA 証明書です。README はファイルの説明書です。



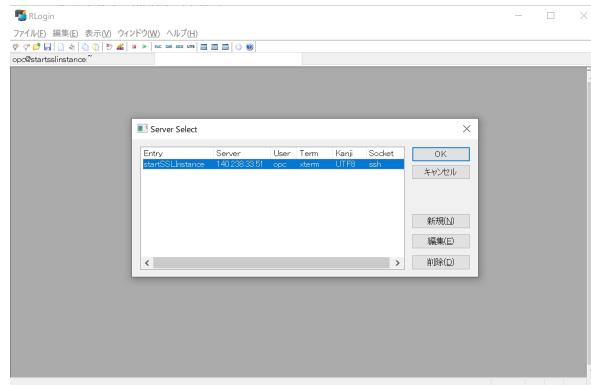
▲図 2.91 server.crt が SSL 証明書で、ca-bundle.ca が中間 CA 証明書

どちらも必要なものなので、この 2 つのファイルをサーバにアップロードしましょう。

2.6.4 Windows で証明書と中間 CA 証明書をアップしよう

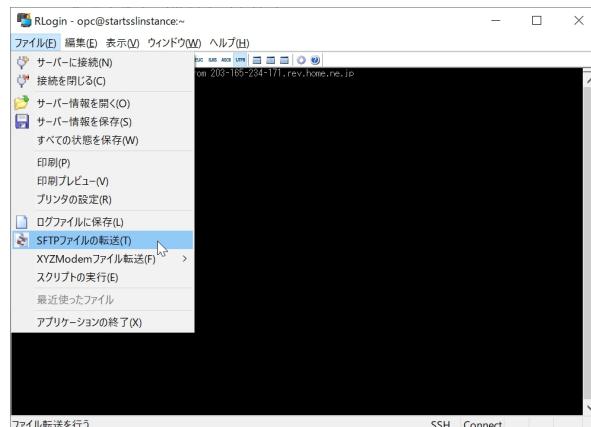
Windows のパソコンを使っている方は、RLogin を起動します。[startSSLInstance] を選択して、[OK] をクリック（図 2.92）します。

2.6 SSL 証明書を取得しよう



▲図 2.92 RLogin を起動して [startSSLInstance] を選択してログイン

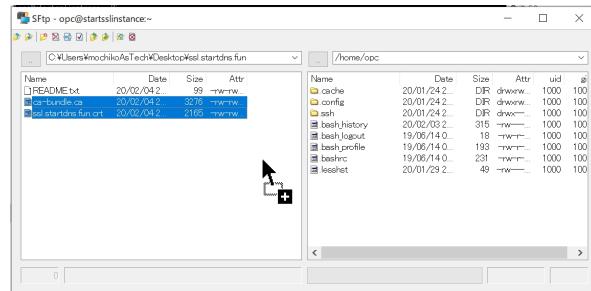
黒い画面が開いたら、[ファイル] から [SFTP ファイルの転送] をクリック（図 2.93）します。



▲図 2.93 [ファイル] から [SFTP ファイルの転送] をクリック

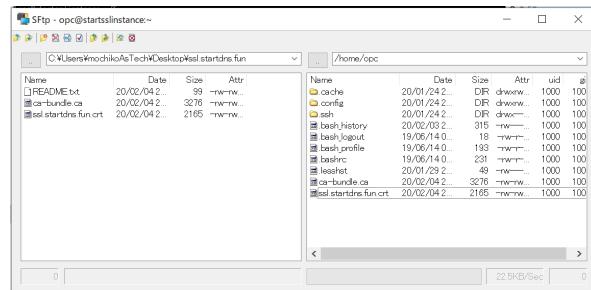
左側があなたのパソコンで、右側がサーバです。左側をデスクトップに展開したフォルダ^{*32}にしたら、[server.crt] と [ca-bundle.ca] を右側にドラッグ&ドロップ（図 2.94）してください。これでサーバの「/home/opc」にファイルがアップロードされます。

*32 筆者の場合は「C:\Users\mochikoAsTech\Desktop\ssl.startdns.fun」でした



▲図 2.94 [ファイル] から [SFTP ファイルの転送] をクリック

右側に [server.crt] と [ca-bundle.ca] がアップされたら、×を押してファイル転送の画面は閉じて構いません。(図 2.95)



▲図 2.95 [ファイル] から [SFTP ファイルの転送] をクリック

2.6.5 Mac で証明書と中間 CA 証明書をアップしよう

Mac を使っている方は、ターミナルを起動してください。scp コマンドを使って、Mac の中にある [server.crt] と [ca-bundle.ca] を、サーバにアップロードします。[展開した フォルダ] と [パブリック IP アドレス] の部分は、ご自身のものに書き換えてください。

```
$ cd ~/Desktop/展開したフォルダ
$ scp -i ~/Desktop/startSSLKey server.crt ca-bundle.ca opc@パブリック IP アドレス:/home/opc/
server.crt    100%   0     0.0KB/s   00:00
ca-bundle.ca  100%   0     0.0KB/s   00:00
```

2.6.6 SSL 証明書と中間 CA 証明書を 1 ファイルにまとめよう

サーバにログインしている状態で、次のコマンドを叩いて、先ほどアップロードした [server.crt] と [ca-bundle.ca] が、ちゃんと「/home/opc/」以下に存在していることを確認します。

```
$ sudo su -
# ls /home/opc/
ca-bundle.ca  ssl.startdns.fun.crt
```

続いて 2 つのファイルから、新たに [startssl.crt] というファイルを作りましょう。^{*33}NGINX では SSL 証明書と中間 CA 証明書という 2 つのファイルを、1 つのファイルにがっちゃんとこ！ とつなげて使うためです。

```
# cd /etc/nginx/ssl/
# awk 1 /home/opc/ssl.startdns.fun.crt /home/opc/ca-bundle.ca > startssl.crt
```

cat コマンドで [startssl.crt] を確認してみましょう。次のように「-----BEGIN CERTIFICATE-----」と「-----END CERTIFICATE-----」が、繰り返し 3 つ表示されれば大丈夫です。

```
# cat /etc/nginx/ssl/startssl.crt
-----BEGIN CERTIFICATE-----
MIIF/DCCB0SgAwIBAgIQaoS/P1b4mCR8mn5/WUrI6zANBgkqhkiG9w0BAQsFADBn
MQswCQYDVQQGEwJKUDE1MCMGA1UEChMcU0VDT0ogVHJ1c3QgU3lzdGVtcyBDTy4s
(中略)
31pTIUPabkFxDPjs1Vw9c7z3Vgk2fnpuwE2lrE+46zrJ3oTRqsABDbYreK1a5vsG
tcnpUlL1hrk/rC3JuI2ttHVaHU+1JCgTSRpvO3a44azDy15T5C97dGxuowPgcaMQ
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEpzCCA4+gAwIBAgIJIrnxVPMSX14AMA0GCSqGSIb3DQEBCwUAMFOxCzAJBgNV
BAYTAkpQMSUwIwYDVQQKExxTRUNPTSBUcnVzdCBTeXN0ZW1zIENPLixMVEQuMScw
(中略)
g3tiJA1FIpfXXD4cArZo6Z1XJ26B4H7vk5GmyR6poDy/CRvC7VIZ3xp6o2348W1j
32S9pEuZhtxtMvPjnsHIWPNdz8pHv21x7bYwDncwN2uk3Qrr1jxTQ9evg==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
```

^{*33} このとき cat コマンドで結合すると、SSL 証明書と中間 CA 証明書の間に改行が入らず、「-----END CERTIFICATE----------BEGIN CERTIFICATE-----」のようになってしまふため、awk コマンドを使っています

```
MIIEcjCCA1qgAwIBAgIJErmw+nLg2EjGMA0GCSqGSIb3DQEBCwUAMFAxCzAJBgNV  
BAYTAkpQMrgwFgYDVQQKEw9TRUNPTSBUnVzdC5uZXQxJzA1BgNVBAstH1N1Y3Vy  
(中略)  
u5ZuCjxerxj3qS1rM46bcEfjopnaD7hnJXSYiL1d0yw5zSW2PEe+LHdoIAb2I6D8  
8UFJH0Cl6sY518jhjk0Os1yeu1C/RcY0+NBNHKZkFEeEb6ez0sg=  
-----END CERTIFICATE-----
```

これでファイルの準備は完了です。

2.6.7 NGINX で HTTPS のバーチャルホストを作ろう

「/etc/nginx/conf.d/」というディレクトリの下で、もともとあった設定ファイルをバックアップしておきます。

```
# cd /etc/nginx/conf.d/  
# mv default.conf default.conf.backup
```

vi コマンドで、同じ場所に新しい設定ファイルを作ります。vi コマンドでファイルを編集するときは、i（アイ）を押してから入力します。書き終わったら ESC キーを押して、「:wq」と入力して Enter キーを押せば変更が保存されます。

```
# vi startssl.conf  
server {  
    listen 80 default_server;  
    return 301 https://$host$request_uri;  
}  
  
server {  
    listen      443 ssl http2;  
    server_name  ssl.startdns.fun;  
  
    # 密密鍵  
    ssl_certificate_key /etc/nginx/ssl/startssl.key;  
    # SSL 証明書  
    ssl_certificate     /etc/nginx/ssl/startssl.crt;  
  
    # 暗号スイート  
    ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AE  
    # プロトコルバージョン  
    ssl_protocols       TLSv1.2;  
    # 暗号スイートの順序はサーバが決める  
    ssl_prefer_server_ciphers   on;  
  
    location / {  
        root   /usr/share/nginx/html;  
        index  index.html index.htm;  
    }
```

```
}
```

設定ファイルが書けたら、構文エラーがないかテストをします。もし書き間違いがあれば、ここでエラーメッセージとして表示されます。

```
# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

[test is successful] と表示されたら、NGINX を再起動して設定を反映しましょう。

```
# systemctl restart nginx.service
```

それからアクセスしたときに表示される、index.html の文字も [Let's start SSL/TLS] に変えておきましょう。

```
# cd /usr/share/nginx/html/
# cp -p index.html index.html.backup
# echo "Let's start SSL/TLS" > index.html
```

2.7 HTTPS でサイトを開いてみよう

証明書を設置して、NGINX の設定ファイルを修正し、NGINX の再起動もしたのでブラウザで「http://ssl.自分のドメイン名」を開いてみましょう。鍵マーク付きで HTTPS のページが表示されるはずです。(図 2.96)



▲図 2.96 HTTPS でサイトが表示された！

2.7.1 LB のところで SSL ターミネーションする方法もある

なお今回はウェブサーバに SSL 証明書を設置しましたが、ウェブサーバの手前にロードバランサーを置いて、そこに SSL 証明書を設置し、SSL ターミネーションを行う方法もあります。その場合、SSL で通信するのはエンドユーザのパソコンからロードバランサーまでで、ロードバランサーからウェブサーバは HTTP で通信するのが一般的です。次のようなメリットがあります。

- 暗号化や復号の処理を行う終端がロードバランサーになるので、ウェブサーバのスペックが低くてもよいパフォーマンスが得られる- 証明書の取得や更新が自動で行われる

第 3 章

SSL/TLS について学ぼう

この章では SSL/TLS について学びます。

3.1 「サイトを HTTPS 化する」とは何か？

そもそもですが「サイトを HTTPS 化する」とは、なんでしょう？

サイトの HTTPS 化とは、サイト全体を「https://」から始まる URL にすることで、「常時 SSL 化」や「常時 SSL/TLS 化」あるいは「AOSSL (Always On SSL の略)」などとも呼ばれています。

今まででは問い合わせページや会員登録など、個人情報を入力したり表示したりする画面のみ HTTPS で暗号化してやり取りしていましたが、これをウェブサイト全体に適用しましょう、というのがサイトの HTTPS 化です。

3.1.1 個人情報をやりとりしないサイトでも HTTPS にしなきゃだめ？

でも個人情報をやり取りするわけでもない、普通のページまで、なぜ HTTPS 化しなければいけないのでしょう？

サイトを HTTPS 化するメリットには、

- 検索順位が上がる- サイトの表示が速くなる- アクセス解析の精度があがる

などがよく挙げられます。確かに上記のようなメリットはありますが、これまでやらないと何かまずいことが起こる、というわけではありませんでした。そのため、今まで先送りにしてきたサイト担当者の方も多いのではないでしょうか？では、この先もサイトを HTTPS 化しなくても大丈夫なのでしょうか？ 答えは NO です！ このまま HTTPS 化しないでいると、どんなデメリットが起きるのか？ を具体的に解説していきましょう。

3.1.2 HTTP 化しないと起きる「わるいこと」

サイトが「安全でない」と表示されてしまう

Google は HTTP から HTTPSへの移行を強く推し進めています。その施策の1つとして、Google が提供するブラウザの「Chrome」では、2018年7月にリリースされた「Chrome バージョン 68」から、HTTPS でないページに対して「保護されていない通信」という表示をするようになりました。

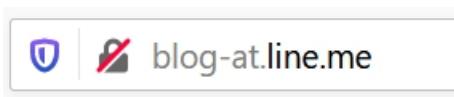
ページを開いた際に、ページの URL が「https://」で始まるものでなければ、次のように URL の左側に「保護されていない通信」という表示（図 3.1）がされます。

① 保護されていない通信 | blog-at.line.me

▲図 3.1 HTTP のサイトを開くと「保護されていない通信」と表示される

エンドユーザがサイトを開いたときに、URL の真横に「保護されていない通信」と表示されたら、さすがに「何か問題があるのかな、見ない方がいいのかも…？」と心配になってしまいますよね。

Chrome だけでなく、同様に Firefox も錠前に赤い斜め線が入ったマークの表示（図 3.2）を行っており、サイトを HTTPS 化しないことで、エンドユーザにサイトが「安全でない」と判断されてしまう状況にあります。



▲図 3.2 HTTP のサイトを開くと「保護されていない通信」と表示される

Wi-Fi スポットでセッションハイジャックされる恐れがある

皆さんが出没する SNS などのサイトを開くと、まだログインしていないのに、ログイン済みのページが表示されることがありますよね？ これは過去にログインした際に、サイトからクッキーで渡された「セッション ID」（一時的な通行証のようなもの）を、次に開いたときにも提示することで、ログイン済みのユーザーとして扱われるからです。

ログインページだけが HTTPS のサイトで、それ以外のページを HTTP で開いたとき、クッキーに Secure 属性が付いていないと、この「セッション ID」は暗号化されない状態で送信されてしまいます。では、誰でも使える Wi-Fi スポットに、スマホやタブレットを繋いだ状態で、HTTP のページを開いて「セッション ID」が送られたらどうなるでしょう？

なんと同じ Wi-Fi につないでいる悪意の第三者によって、暗号化されていない「セッション ID」を盗まれ、なりすましてサイトにログインされる恐れがあるのです。これが「セッションハイジャック」と呼ばれる攻撃です。

こうしたことが発生しないように、サイト全体を HTTPS にして、常にクッキーの「セッション ID」を暗号化しておかないと危ないですよ、ということです。またログインなどが一切ない、完全に静的コンテンツのみのサイトであっても、Wi-Fi スポットで見知らぬ誰かに「あの人はどんなページを見ているのかな？」とデータを窺視されるリスクは

あります。サイトとの通信が HTTPS で暗号化されていれば、情報を盗み見られるエンドユーザにとってのリスクを防ぐことができます。

周りが HTTPSになると、ますますリファラーが取れなくなる

こちらは Google アナリティクスなどを使って、サイト流入元の情報を確認されている方にとって、重要なデメリットです。

自社のサイトが HTTP だと、HTTPS のサイトからリンクを踏んで飛んできた場合に、リファラ（利用者が直前に訪問していたサイトの情報）を取得することができません。実際に Google アナリティクスを開いて、集客の「参照元/メディア」を確認してみてください。アクセス元が「(direct) / (none)」と表示されて、どこから飛んできたのか分からぬるものはありませんか？ その中には、ブラウザのブックマークや、メール内のリンクから飛んできた、本当に「直前に訪問していたサイトがないもの」だけでなく、HTTPS のサイトから飛んできたアクセスも含まれています。

今後、周囲のサイトの HTTPS 化が進んで、自社サイトだけが HTTP で取り残されると、この「リファラーが取得できる割合」はますます下がっていくことになります。ですが、自社のサイトを HTTPS にすれば、HTTP のサイトから飛んできた場合も、HTTPS のサイトから飛んできた場合も、リファラーを取得することができるようになります。

検索順位が下がる

Google は 2014 年の時点で既に、HTTPS に対応しているウェブサイトを検索ランキングで優遇する方針を発表しています。

3.1.3 HTTPS 化すると起きる「いいこと」

表示速度が上がる

Same Site 問題に対応できる

すべてのサイトが暗号化されれば「暗号化されている情報は重要な情報」というアタリが付けられなくなる

流れしていくデータの殆どが平文な中で、たまに暗号化されたデータが流れてくると「暗号化されてるってことは、あれは大事な情報だな！」とアタリを付けることができます。

ですが、すべてのサイトが HTTPS になって、流れてくるデータがすべて暗号化されなければ、どれが重要な重要なデータなのかアタリが付けられなくなります。

大事なものも大事でないものもすべて同じ頑丈なアタッシュケースにしまって運ぶことで、泥棒がどのアタッシュケースを狙えばいいのか分からなくなる。木は森に隠せ、という戦法ですね。

3.2 SSL/TLS とは？

SSL (Secure Socket Layer) /TLS (Transport Layer Security) とは、インターネット上で安全にデータを送受信するための約束事（プロトコル¹）です。

SSL も TLS もあくまでプロトコル、つまり通信するための「約束事」なので、実際に通信するときは、そのプロトコルに従って実装されたソフトウェアを使います。SSL/TLS では、OpenSSL というオープンソースのソフトウェアを使うことが殆ど²です。

3.2.1 SSL と TLS はどういう関係？

SSL と TLS は別々の名前ですが、その役割に大きな違いはありません。「SSL3.0」からバージョンアップする際に、「SSL3.1」ではなく「TLS1.0」という新しい名前が付けられました。つまり「TLS は SSL の後継バージョン」ということです。

ちなみに名前がまだ SSL だったときの最後のバージョン「SSL3.0」は、重大な脆弱性³が見つかり、2015 年の RFC 7568⁴でもう使わないよう「Do Not Use SSL Version 3.0」と示されています。

ですが SSL という言葉の認知度が高く、TLS と呼んでもピンとこない人の方が多いため、現状は分かりやすさと正確さを両立するために SSL/TLS と併記されることが多いです。

本著では SSL/TLS のことを指して、SSL という言葉を使用します。

3.2.2 SSL イコール HTTPS ではない

サイト全体を「https://」から始まる URL にすることを、「常時 SSL 化」のように呼ぶため、皆さんの中には SSL = HTTPS だと思っている人もいるかも知れません。

HTTP と SSL を組み合わせて使うことで、通信を保護するプロトコルが HTTPS です。ですが、SSL は HTTPS においてのみ使われるプロトコルではありません。例えばサーバにファイルをアップするときの FTP と組み合わせて使う FTPS (FTP over SSL)

*¹ 例えば手紙は「メッセージを書いた便せんを封筒に入れて、表に宛先、裏に差出人を書き、重さに応じた切手を貼ってポストに入れる」という取り決めに従えば、きちんと相手に届きます。手紙の例と同じように、インターネットで通信を行う際、どんなデータをどんな方法で送受信するのか、という手順の約束事のことをプロトコルと呼びます。SSL も TLS もプロトコルですし、HTTP や HTTPS、DNS も SMTP もプロトコルです

*² さっき証明書を取得するときに打ったコマンドも、openssl コマンドでしたね

*³ 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

*⁴ Deprecating Secure Sockets Layer Version 3.0 <https://tools.ietf.org/html/rfc7568>

や、メール送信のSMTPと組み合わせて使うSMTPS(SMTP over SSL)など、HTTPS以外にもSSLを使う場面はいろいろあります。

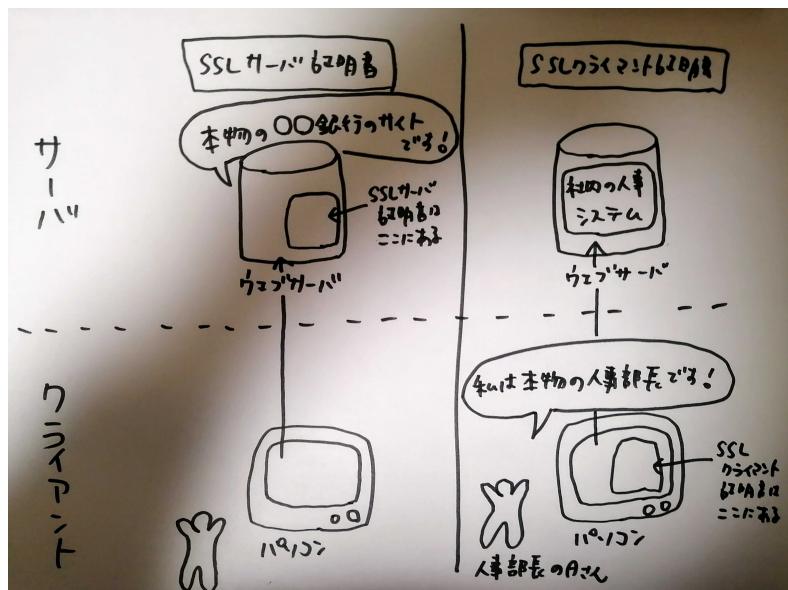
繰り返しになりますが、SSLはインターネット上で安全にデータを送受信するためのプロトコルです。上位のアプリケーション層^{*5}がHTTPであれ、FTPであれ、SSLと組み合わせて使うことでデータを安全に送受信できるようになります。

3.3 SSL証明書とは

3.3.1 SSLサーバ証明書とSSLクライアント証明書

「SSL証明書」という名前を聞いたことはありますか？名前だけは知っているという人も、実際に仕事で使ったことがあるという人もいると思います。

SSL証明書には、SSLサーバ証明書とSSLクライアント証明書の2種類（図3.3）があります。ざっくり言うとサーバの身元を証明するのがSSLサーバ証明書で、クライアントの身元を証明するのがSSLクライアント証明書です。



▲図3.3 SSLサーバ証明書とSSLクライアント証明書

*5 OSI参照モデルのアプリケーション層のこと。大学の授業や新卒研修で、誰しも一度は「アセトネデブ」という語呂合わせを聞いたことがあるのでは…

単に「SSL 証明書」という略称で呼んだときは、「SSL サーバ証明書」のことを指すことが殆どです。本著でも SSL サーバ証明書のことを指して、SSL 証明書という言葉を使用します。

3.3.2 SSL 証明書はどんな場面で使われている？

SSL 証明書はどんな場面で使われているのでしょうか？

SSL 証明書は、あなたがブラウザで「https://」から始まる URL のサイトを開いて、次のようなマーク（図 3.4、図 3.5）が表示されているときに使われています。



▲図 3.4 Chrome で HTTPS のサイトを開いたとき

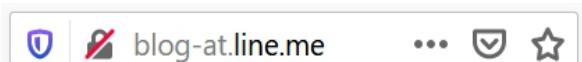


▲図 3.5 Firefox で HTTPS のサイトを開いたとき

逆に「http://」から始まる URL のサイトを開いて、次のようなマーク（図 3.6、図 3.7）が表示されているときは使われていません。



▲図 3.6 Chrome で HTTP のサイトを開いたとき



▲図 3.7 Firefox で HTTP のサイトを開いたとき

3.4 SSL 証明書は全然違う2種類の仕事をしている

「SSL 証明書は https から始まるページで使われている」ということが分かったところで、次に、https から始まるページでは、SSL 証明書は一体何をしているのでしょうか。実は、SSL 証明書は全然異なる次の2つの仕事をしています。この点が SSL 証明書の話の分かりにくさの原因なのです。

- Web サイトで送受信する情報を暗号化すること
- Web サイト運営者の身元を証明すること

それでは1つ目の役割、「Web サイトで送受信する情報を暗号化すること」をから、詳しく説明していきます。

3.4.1 Web サイトで送受信する情報を暗号化すること

3.4.2 Web サイト運営者の身元を証明すること

3.5 鍵マークが壊れるケース

3.5.1 すべて HTTP で通信しているとき

3.5.2 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

3.6 ウェブページが表示されるまで

3.6.1 1 往復で表示されるわけじゃない

3.7 SSL 証明書は何を証明してくれるのか？

3.7.1 ネットバンクの事例

3.8 認証局事業者の身元は誰が証明する？

3.8.1 身元保証の連鎖をつなぐ中間 CA 証明書とルート証明書

3.9 SSL 証明書はどうしてあんなに値段に差があるの？

3.10 同じ「SSL 証明書」という名前でも 3 つの種類がある

3.10.1 EV 証明書

さよならグリーンバー

3.10.2 OV 証明書

3.10.3 DV 証明書

3.10.4 3 つの違いは何か？

3.10.5 ブラウザベンダーによる EV 証明書の扱いの変化

3.11 その他の証明書

3.11.1 中間証明書

113

3.11.2 クロスルート証明書

3.12 どの証明書を買えばいい？

3.12.1 ワイルドカード証明書

ただし SNI は、1つの IP アドレスを複数のバーチャルホストで共用するため、HTTPS で使用した場合、SNI 非対応のクライアントではデフォルトのホストが応答します。2019 年現在、SNI 非対応端末を「対象端末」としているサービスはあまり多くないかもしれません。

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

レビュアー

- Takeshi Matsuba

参考文献

- プロフェッショナル SSL/TLS Ivan Ristić、齋藤孝道（監訳）
– <https://www.lambdanote.com/products/tls>
- プロフェッショナル IPv6 小川晃通

付録 あとがき

- <https://www.lambdanote.com/products/ipv6>
- 食べる！SSL！—HTTPS環境構築から始めるSSL入門 小島拓也、中嶋亜美、吉原恵美、中塚淳
 - <https://www.amazon.co.jp/dp/B00PHC4480>

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

「なんとなく」から「ちゃんとわかる！」へ

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