

DNS をはじめよう

基礎からトラブルシューティングまで 改訂第 2 版

mochikoAsTech 著

2019-09-22 版 mochikoAsTech 発行

はじめに

2019 年 9 月 mochikoAsTech

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

本著は 2018 年 4 月の技術書典 4 で初版を出して以来、2019 年 8 月までの約 1 年半で 3,600 冊ほどが読者の皆さんとのところへ旅立っていました。さらに続編となる「AWS をはじめよう」も合わせると、お陰さまでシリーズの累計は 6,400 冊以上となりました。

「100 冊完売できたらすごい！」と言われる同人誌の世界で、本著がこんなに手に取ってもらえた理由はいくつか考えられますが、その中で最も大きいものとして、実は多くの人が心ひそかに「DNS とか AWS とかよく分からぬ…インフラ怖い…でも分かりたい！」と感じていた、という背景があると筆者は思っています。

技術を学ぶ過程は単純な右肩上がりの一直線ではなく、「分かった！」「やっぱり分かってなかった…」という山と谷の繰り返しです。筆者も本著を書いたことで何度も「DNS 何も分からぬ…」という谷間を迎えました。谷は暗く、とても恐ろしく感じられます。みんなが当たり前に知っていることを、自分ただひとりだけが知らないように感じられ、今更学んでもとても追いつけないと焦ったり諦めたり…何より情けなくて無知な自分を周囲に知られたくない、という気持ちになります。

ですが安心してください。技術書典を訪れたたくさんのエンジニアの中で、DNS が分かっていなかったのは筆者だけでもあなただけでもありません。少なくとも 3,600 人が「DNS 何も分からぬ…」と初心者向けの本著を手に取ったのです。

自分以外にも分かっていない人はいっぱいいたんだ、と思うと、ちょっとほっとしませんか？ 不安な気持ちで心に蓋をしていると、新しい知識はあなたの中になかなか入ってきません。楽しい話をしましょう。

本著を手に取ったあなた、DNS は好きですか？ 筆者は DNS が大好きです！ なぜか分からぬけど、DNS 絡みの障害が起きたニュースを見てはわくわくして勝手に調査してしまうし、DNS のことで困っている人がいると解説したくてうずうずします。

普段、システムの開発や運用をしている Web アプリケーションエンジニアを料理人に

例えると、インフラエンジニアは台所そのものを作るのが仕事です。料理人は料理を作るのは得意でいつも台所にいますが、台所そのものの作り方まで詳しいか？ というと、必ずしもそうとは言えません。（もちろん家の建て方から、料理から、テーブルコーディネイトから、後片付けまで全部出来るフルスタックなエンジニアもいますが、みんながみんなそうではなく分担しあって頑張っていますよね）

台所の造りに詳しくなくても料理は作れます。ですが台所の造りに詳しいと、困ったときに「自分でなんとかできる範囲」が広がります。たとえばシンクの蛇口から水漏れして台所が水浸しになってしまっても、水道管や蛇口のことをある程度知っているれば、まずは元栓を閉める応急処置をしたり、あるいはパッキンを自分で交換して直すこともできます。修理を頼んだ場合でも、修理業者の説明が理解できるので金額の妥当性もちゃんと判断できます。

DNS は水道管のように地味なインフラです。蛇口を捻れば水が出るのは当たり前なのと同じように、ドメイン名を聞けば IP アドレスが返ってくるのは当たり前で意識されることすらありません。ですがひとたび DNS で問題があれば、サイトは見られなくなり、メールは送れなくなり、水道が止まったのと同じくらい影響範囲の大きい障害となります。

筆者はインフラエンジニアを経験した後、Web アプリケーションエンジニアへジョブチェンジしたので、インフラの知識が土台としてあったことでその上にアプリケーションの知識が載せやすく「この順番で学んでおいてよかったな」と思う場面が多々ありました。本著「DNS をはじめよう」はそんな筆者が、普段開発や運用をしているエンジニアに「DNS のことを知っておくと自分でなんとか出来る範囲が広がるよ」と伝えたくて書いた一冊です。

「自分は DNS を知らない。よく分かっていない」と素直に認めることは、理解に向かう長い道のりのはじめの一歩です。一緒に「分かった！」の山と「やっぱり分かってなかつた…」の谷を越えて、ゆっくり DNS を好きになっていきましょう。

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。

- 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ること」にコンプレックスのある人
- ドメイン名を買ったりするけど DNS はあまり分かっていない人
- ブログやポートフォリオサイトを独自ドメインで作ってみたい人

-
- AWS や Route53 という単語に興味のある人

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。

- BIND をインストールしてネームサーバやフルリゾルバの運用をはじめたい人
- named.conf の推奨設定を知りたい人
- フルリゾルバを BIND から Unbound に移行したい人
- ISP（インターネットサービスプロバイダ）やクラウド事業者の中で DNS のお守りをしている人
- ネームサーバやフルリゾルバの運用トラブル事例を知りたい人

本著の特徴

本著では実際にドメイン名を1つ購入します。買ったドメイン名を使って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。

- ドメイン名を買うときは何に注意してどこで買ったらしいか分かっている
- Whois 情報に何を登録すべきか分かっている
- 障害が起きたときに黒い画面（ターミナル）で dig コマンドや whois コマンドを駆使して原因を調査できる

-
- サイト移管時に「DNS 浸透待ちで 8~24 時間くらいは切り替わりません」みたいなことを言わない
 - 読む前より DNS が好きになっている

免責事項

本著に記載されている内容は筆者の所属する組織の公式見解ではありません。

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

不正確あるいは誤認と思われる箇所がありましたら、必要に応じて適宜改訂を行いますので GitHub の 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/startDNS>

目次

はじめに	3
想定する読者層	4
マッチしない読者層	5
本著の特徴	5
本著のゴール	5
免責事項	6
第1章 ドメイン名と Whois	11
1.1 ドメイン名のお店選び	12
1.1.1 お名前.com	13
1.1.2 名づけてねっと	13
1.1.3 Yahoo! ドメイン	14
1.1.4 ムームードメイン	15
1.1.5 VALUE DOMAIN	16
1.1.6 ゴンベエドメイン	17
1.2 値段やお店によってドメイン名の品質は違うのか	17
1.3 ドメイン名を売るお店はレジストラとリセラの2種類	18
1.3.1 お店選びのポイント	19
【コラム】ドメイン界の巨人 GMO インターネット	21
1.4 ドメイン名が生まれてから手元に届くまで	21
1.4.1 ドメイン名の卸元はレジストリ	21
1.4.2 1つのTLDには1つのレジストリ	22
1.4.3 〈トラブル〉 Mackerel のアラート誤報は io ドメイン名が原因	24
1.4.4 【ドリル】ネットショップのドメイン名は○○.xxx? ○○.com?	25
【コラム】ブランド TLD と GMO ドメインレジストリ	26
1.5 なぜレジストリがそのTLDを独占するのか	26

【コラム】他のドメイン名と比べて jp ドメイン名が高い理由	27
1.6 ICANN がレジストリを決める	28
1.7 取り扱う TLD はお店が自由に決められる	28
1.8 「ドメイン」なのか「ドメイン名」なのか?	29
1.8.1 サブドメインとは?	31
1.9 実際にお名前.com でドメイン名を買ってみよう	32
【コラム】類似のドメイン名を確保しておくべきか?	40
1.9.1 自動更新はオフにしておこう	42
1.9.2 〈トラブル〉ドメイン自動更新の落とし穴	48
1.9.3 【ドリル】連絡先メールアドレスは自分だけでいい?	49
【コラム】ドメイン名は「買う」ものか?	50
1.10 Whois とは	51
1.10.1 JPRS WHOIS でドメイン名の所有者情報を見てみよう	52
【コラム】ドメイン名は大文字小文字の区別をしない	55
1.10.2 Whois の項目はレジストリごとに微妙に違う	56
1.10.3 Whois を正確に登録しなければいけない理由	58
1.10.4 〈トラブル〉ドメイン情報認証メールを無視して全サイトが停止	59
1.10.5 【ドリル】Whois に登録すべきなのはクライアントの情報?	61
1.10.6 プライバシーを守るための Whois 情報公開代行	62
1.10.7 〈トラブル〉レジストラが倒産したらドメイン名はどうなる?	63
1.11 co.jp は国内企業しか買えないドメイン名	64
1.11.1 【ドリル】co.jp ドメイン名は 2 つ買える?	65
1.12 ドメイン名の有効期限が切れるとどうなるのか?	66
1.12.1 〈トラブル〉ドロップキャッチされたイオンシネマ	67
1.12.2 【ドリル】ドメイン名を手放してよい条件は?	68
1.13 ドメイン名を買ったたらサイトが見られるか?	69
第 2 章 DNS の仕組み	71
2.1 DNS とは	72
2.1.1 ネームサーバ	72
2.1.2 フルリゾルバ	72
2.2 お名前.com のページが表示されるまで	74
2.3 ゾーンと委任	75
2.4 リソースレコード	77
2.5 hosts ファイル	78

2.5.1	Windows で hosts を使ってみよう	78
2.5.2	Mac や Linux で hosts を使ってみよう	80
第3章	AWS のネームサーバ (Route53) を使ってみよう	81
3.1	AWS とは	82
3.2	AWS アカウント作成	82
3.2.1	AWS のマネジメントコンソールにログイン	90
3.3	ドメイン名のネームサーバを Route53 に変更	92
3.3.1	Route53 でホストゾーンを作成	92
3.3.2	自分のドメイン名のネームサーバが何か確認	95
3.3.3	ネームサーバをお名前.com から Route53 に変更	99
3.3.4	TTL が過ぎるまではネームサーバが切り替わらない	105
【コラム】	名前解決はプッシュ型じゃなくてプル型	107
3.3.5	使われるのは親と子どちらの NS レコードの TTL?	107
3.3.6	【ドリル】ネームサーバを変えること≠レジストラを変えること	109
【コラム】	DNS の次は AWS をはじめたい?	110
第4章	dig と whois を叩いて学ぶ DNS	113
4.1	dig と whois のインストール	114
4.1.1	Mac も Linux のサーバ環境もない場合	114
4.2	nslookup はもう卒業! dig コマンドの便利な使い方	115
4.2.1	host や nslookup じゃダメですか?	116
4.3	Whois を叩いてドメイン名や IP の持ち主を調べよう	117
4.3.1	【ドリル】ドメイン名の有効期限が分からぬ	118
4.4	dig を叩いてリソースレコードを確認してみよう	120
4.4.1	A レコード	120
4.4.2	【ドリル】ウェブサーバはどこにある?	120
【コラム】	同じドメイン名で A レコードを複数作るとどうなる?	122
4.4.3	MX レコード	123
4.4.4	〈トラブル〉 test@test.co.jp を使って情報漏洩	125
【コラム】	dig のオプションは略せる	128
4.4.5	NS レコード	129
4.4.6	【ドリル】他社へのサイト移管時にネームサーバの所在が不明	129
4.4.7	SPF レコード (TXT レコード)	131
4.4.8	【ドリル】どうしてメールが迷惑メール扱いされるの?	132

4.4.9 〈トラブル〉 SPF を設定したのに時々迷惑メールになる	133
4.4.10 PTR レコード	134
4.4.11 CNAME レコード	135
4.4.12 【ドリル】 CNAME の調べ方と使いどころ	137
4.4.13 CNAME と他のリソースレコードは共存できない	138
4.4.14 グルーレコード	140
【コラム】 この IP アドレスはグローバル IP? プライベート IP?	141
第 5 章 トラブルシューティング	143
5.1 〈トラブル〉 URL は www ありなしどっち?	144
5.2 〈トラブル〉 .dev で終わるテストサイトが見られなくなった	145
5.3 サイト移管の AtoZ	146
5.3.1 【ドリル】 リニューアル後のサイトが人によって表示されない .	147
5.4 〈トラブル〉 サブドメインを追加したのにサイトが見られない	149
5.5 〈トラブル〉 CAA レコードが原因で SSL 証明書が発行できなかった .	156
5.6 〈トラブル〉 AWS で突然ドメイン名が引けなくなった	157
5.6.1 レートリミットを超えると NXDOMAIN を返す	157
5.6.2 勝手に TTL を短くする	157
5.7 〈トラブル〉 ラブライブ! のオフィシャルサイトが乗っ取られた	158
5.7.1 移管できないようロックしておけば安心?	161
付録 A 本当の AWS	163
A.1 AWS - 愛はワガママサンシャイン	164
あとがき	165
PDF 版のダウンロード	165
Special Thanks:	166
レビュー	166
参考書籍	166
著者紹介	167

第 1 章

ドメイン名と Whois

自分のドメイン名を買ったことはありますか？ まだ買ったことがないなら、ドメイン名を買うのはあなたが想像しているよりずっと簡単です。なんといっても DNS の D はドメインの D ですし、DNS を理解するにはドメイン名を買って手を動かしてあれこれ試してみるのがいちばんです。という訳でまずは自分のドメイン名を買ってみましょう。

1.1 ドメイン名のお店選び

それではまずどこでドメイン名を買う^{*1}のか、お店を決めましょう。

お手元のパソコンでブラウザを開いて「ドメイン」で検索（図1.1）してみてください。検索結果にはお名前.com、名づけてねっと、Yahoo!ドメイン、ムームードメイン、VALUE DOMAIN、ゴンベエドメインなど、たくさんのお店が出てきます。



▲図1.1 ドメインで検索

どこも「ドメイン取得なら〇〇」「ドメイン取るなら△△」のように書いてありますが、一体どのお店で買うのがいいのでしょうか？ 今回は勉強用ですし出来れば安く買いたいですが、そもそもお店によって値段にどれくらい違いがあるのでしょう？ 取りあえず1軒ずつ見ていくってみましょう。

ここでは仮に「筆者が株式会社イグザンブルという会社の広報担当になったのでexample.co.jpというドメイン名を買おうとしている」という設定であちこちのお店を見てみます。

^{*1} ところでドメイン名って「買う」ものなの？ という疑問については後述します。取りあえず先へ進みましょう

1.1.1 お名前.com

お名前.com^{*2}で example.co.jp を調べてみると、1年で 3,660 円^{*3}でした。(図 1.2) しかも残念ながら売り切れなのか×がついて「登録済みドメインのため、お申込みいただけません。」となっています。^{*4}



▲図 1.2 お名前.com では 3,660 円

1.1.2 名づけてねっと

続いて NTTPC コミュニケーションズの名づけてねっと^{*5}で、example.co.jp の値段を見てみましょう。example.co.jp は属性 JP1 年プランというプラン名で 7,200 円でした。(図 1.3)

^{*2} <https://www.onamae.com/>

^{*3} ドメイン名の値段はすべて税別（ゴンベエドメインのみ税込）表記で、2019 年 8 月に著者が検索した時点の金額です

^{*4} 正確には売り切れというよりも、example.co.jp は例示用のドメイン名なので購入ができません。詳細は第 4 章「dig と whois を叩いて学ぶ DNS」で後述します

^{*5} <https://www.nadukete.net/>



▲図 1.3 名づけてねっとでは 7,200 円

1.1.3 Yahoo! ドメイン

Yahoo! ドメイン^{*6}でも example.co.jp の値段を調べてみましょう。おや？ どうしたのでしょう？ プルダウンに co.jp がありません。（図 1.4）



▲図 1.4 プルダウンに co.jp がない

「取得できるドメインの種類」（図 1.5）を確認したところ、Yahoo! ドメインではどう

^{*6} <https://domains.yahoo.co.jp/>

やら〇〇.co.jp というドメイン名は取り扱いがないため example.co.jp は買えないようです。

取得できるドメインの種類

Yahoo!ドメインでは、以下のトップレベルドメインで独自ドメインが取得できます。

分類	種類	登録資格※1	使える文字と文字数
gTLD	.com	特になし	【文字】 半角英数字 (a～z, 0～9)、 半角ハイフン(-)。ただし、 文字列の最初と最後のハイフンは使用不可。
	.net		
	.org		
	.biz	商用利用のみ ※2	【文字数】 3～63文字
	.info	特になし	
汎用JP	.jp	日本に在住している法人 または個人 ※3	

※1 すべてにおいて、Yahoo!ドメインご契約の条件に同意していただく必要があります。

※2 .bizドメインは、.bizドメインの登録に関する特則に同意していただく必要があります。

※3 .jpドメインは、汎用JPドメイン名の登録に関する追加規約に同意していただく必要があります。

△ 注意

- Yahoo!ドメインでは、「co.jp」や「ne.jp」などの属性JPドメイン、「日本語.jp」の日本語ドメインは取り扱っていません。

▲図 1.5 Yahoo!ドメインでは co.jp の取り扱いなし

1.1.4 ムームードメイン

ムームードメイン^{*7}はお名前.com より少し高い 3,980 円でした。(図 1.6)

^{*7} <https://muumuu-domain.com/>

第1章 ドメイン名とWhois



▲図 1.6 ムームードメインでは 3,980 円

1.1.5 VALUE DOMAIN

VALUE DOMAIN^{*8}は 3,780 円でした。(図 1.7)

The screenshot shows the Value Domain website interface. At the top, there is a navigation bar with links for 'ドメイン登録', '更新・移管', '料金表', 'サーバー', 'セキュリティ', 'サポート', 'アフィリエイトプログラム開始', 'ライブチャット', 'ログイン', and 'ユーザー登録'. Below the navigation bar is a search bar with dropdown options for 'gTLD', 'JP', 'ccTLD', 'New gTLD', '全種類', '一般価格', and 'パレク割引価格'. There is also a 'ドメインの種類を入力' (Enter domain type) input field and a '検索' (Search) button. On the left side, there is a sidebar titled 'お知らせ' (Announcements) with several news items. The main content area is titled 'JPドメイン' (JP Domain) and contains a table of domain prices. The table has columns for 'ドメイン' (Domain), '意味' (Meaning), '1年 (単位: 円)' (1 year (unit: yen)), and 'WHOIS代行' (WHOIS proxy). The table shows the following data:

ドメイン	意味	1年 (単位: 円)		WHOIS代行
		価格	税込み	
.jp (ローマ字)	日本 (汎用JP)	2,840	3,067	○
.jp (日本語)	日本 (汎用JP)	1,190	1,286	○
.jp (日本語) *1	日本 (汎用JP)	629	679	○
.co.jp	日本の会社・法人	新規 3,780	新規 4,082	×
		更新 3,780	更新 4,082	

▲図 1.7 VALUE DOMAIN では 3,780 円

*8 <https://www.value-domain.com/>

1.1.6 ゴンベエドメイン

品揃え日本一！と謳っているゴンベエドメイン^{*9}はどうでしょう？こちらは少し高めの5,616円でした。（図1.8）

ドメイン名：example				
*表示価格は税込です。				
jp ¥3,067 ×	co.jp ¥5,616 アクセス集中のため検索停止中	ne.jp ¥6,480 アクセス集中のため検索停止中	ac.jp ¥6,480 アクセス集中のため検索停止中	ed.jp ¥6,480 アクセス集中のため検索停止中

▲図1.8 ゴンベエドメインでは5,616円

6社の価格を確認しましたが、値段だけで比べるとお名前.comで買うのがよさそうです。（表1.1）

▼表1.1 example.co.jp のドメイン名代

お店	値段
名づけてねっと	7,200円
ゴンベエドメイン	5,616円
ムームードメイン	3,980円
VALUE DOMAIN	3,780円
お名前.com	3,660円
Yahoo! ドメイン	取り扱いなし

しかしこうして金額を並べてみると色々な疑問が湧いてきませんか？一番高い名づけてねっとの7,200円は、一番安いお名前.comの3,660円と比較すると倍近い金額です。なぜこんなに金額に差があるのでしょう？それになぜYahoo!ドメインだけexample.co.jpを取り扱っていなかったのでしょうか？

1.2 値段やお店によってドメイン名の品質は違うのか

あまりに金額に差があるので、もしかして同じ「ドメイン名」いう名前がついていてどのお店で買ったのかや、その金額によって何か違いがあるのだろうか？と株式会社イ

*9 <https://www.gonbei.jp/>

グザンブルに勤める筆者はちょっと不安になってきました。あなたは A と B のどちらだと思いますか？ 答えを書きこんでみましょう。

- A. 同じドメイン名でも、買うお店や金額によって何かしら品質に違いがある
- B. 品質に違いはなく、どこで買ってもドメイン名はドメイン名である

答え _____

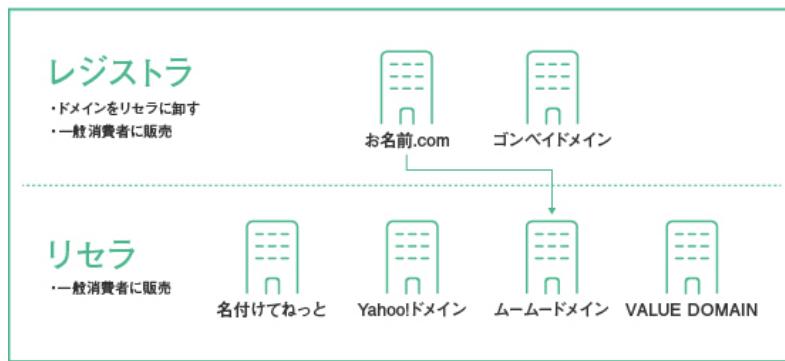
正解は B です。ドメイン名はどこのお店で買っても品質に差はありません。プロ向けに業務用のドメイン名屋さんがあるわけではないので、Web 制作会社でもよくお名前.com やムームードメインでドメイン名を購入しています。ロボット掃除機のルンバはどこで買っても同じルンバですが、ヤマダ電機とヨドバシカメラで値段は異なりますよね。ルンバと同じようにドメイン名もどこで買っても同じドメイン名ですがお店によって値段が違うのです。

「なるほど！ どこで買ってもドメイン名はドメイン名。なら深く考えずに安いお店で買おう！」と思われた方、ちょっと待ってください。ドメイン名の品質に差はなくとも、どこで買うかはよく考えた方がいいんです。なぜならドメイン名を売るお店には大きく分けて 2 つの種類があるからです。

1.3 ドメイン名を売るお店はレジストラとリセラの 2 種類

ここまで「ドメイン名のお店」とひとくくりに呼んでいましたが、実はお名前.com やゴンベエドメインは「レジストラ（登録事業者）」です。そして名づけてねっとや Yahoo! ドメイン、ムームードメイン^{*10}、VALUE DOMAIN はレジストラの下にいる「リセラ（再販事業者）」です。（図 1.9）

^{*10} 正確にはムームードメインは jp ドメイン名においてのみレジストラです。それ以外の TLD はレジストラ（eNom とお名前.com）から仕入れてリセラとして販売しています



▲図 1.9 レジストラとリセラ

レジストラ？リセラ？いきなり知らない単語が出てきましたね。一体何なのでしょう？「リセラ（再販事業者）はレジストラ（登録事業者）からドメイン名を仕入れて、それを再販している」という表現だと分かりやすいでしょうか。つまりムームードメインは、お名前.comなどのレジストラからドメイン名を仕入れてそれを再販している再販事業者なのです。

ドメイン名を買うときに、そのお店がレジストラなのかリセラなのか意識して買う人はほとんどいないと思います。お店にレジストラとリセラという種類があることなんて知らなかった、という人の方が多いのではないでしょうか。

レジストラとリセラのどちらから買ったとしても、ルンバと同じでドメイン名としての品質には変わりありません。しかし中間業者が増えるほど、そのお店が倒産してしまった絡が取れなくなるといったリスクも増えるので、どちらかといえばレジストラから買った方がそのリスクは軽減されます。またお店を選ぶときに見るべきポイントは、レジストラカリセラかだけでなく、他にもいくつかあります。

1.3.1 お店選びのポイント

お店によっては「初年度は99円だけど2年目からは7,980円」のように、最初のキャンペーン価格が安いだけで2年目以降の標準価格が高い場合があります。長く使うと思われるドメイン名の場合は2年目以降の更新にかかる金額も確認しましょう。

それから、ドメイン名を買うとそのドメイン名に関する設定変更は無料で、管理画面上でボタンをぽちっと押せばすぐに反映される、というお店もあれば、何か変更するたびにメールでの依頼が必要で、都度変更手数料がかかって設定完了までは最低3営業日を要する、というお店もあります。ドメイン名の金額が同じなら前者の方がいいですよね。

さらにお店によってはドメイン名を買うとサーバが数か月分無料でついてくる、といった無料オプションがあったりします。

前述のようにドメイン名の品質はどこのお店で買っても同じですので、2年目以降の金額や管理画面の使いやすさ、無料オプションの充実度などを見てどこで買うかを考えるのがいいでしょう。

迷ったら取りあえずお名前.com を使っておけば間違いはありません。何しろ取り扱うドメイン名は550種類以上で、累積登録実績2,000万件^{*11}という国内最大のレジストラです。国内だけでなく世界中のレジストラを対象にした新gTLD^{*12}の保有数ランキング(図1.10)^{*13}でもトップ5に入るほどの規模なのです。

Top 30 Registrars			
Registrar	Domains	±/-	% Share
1. Alibaba Cloud Computing Ltd. (HiChina / www.net.cn, Alibaba Group Holding Ltd.)	3,640,610	-63,173 ↓	13.85%
2. NameCheap, Inc.	3,024,969	-15,084 ↓	11.50%
3. GoDaddy.com, LLC (GoDaddy Group)	2,672,544	539 ↑	10.16%
4. GMO Internet, Inc. d/b/a Onamae.com	2,119,214	14,562 ↑	8.06%
5. Chengdu West Dimension Digital Technology Co., Ltd. (www.west.cn)	1,787,836	15,593 ↑	6.80%
6. Key-Systems, LLC (CentralNic Group PLC)	869,148	-951 ↓	3.31%
7. Eranet International Limited	750,255	27,453 ↑	2.85%
8. PDR Ltd. d/b/a PublicDomainRegistry.com	722,752	-2,287 ↓	2.75%
9. NameSilo, LLC	673,172	4,522 ↑	2.56%
10. Web Commerce Communications Limited dba WebNic.cc	666,523	13,895 ↑	2.53%

▲図1.10 新gTLDの保有数が多いレジストラランキング

*¹¹ 2019年3月15日に累積ドメイン登録実績が2,000万件を突破し、国内のドメイン登録シェアは87%以上だそうです。<https://www.onamae.com/campaign/20MM/>

*¹² generic Top Level Domainの略。gTLDの一覧は<https://www.nic.ad.jp/ja/dom/types.html>を参照

*¹³ <https://ntldstats.com>

【コラム】ドメイン界の巨人 GMO インターネット

余談ですがお名前.com もムームードメインも VALUE DOMAIN もすべて GMO インターネットグループのサービスですので、この 3箇所であればどこで買っても同じといえば同じです。（金額はお名前.com が若干安かったですね）

渋谷でセルリアンタワー^{*14}に描かれた GMO のロゴを見るたび、ここに何かあったら日本のインターネットの何割が止まるんだろう、と思います。

1.4 ドメイン名が生まれてから手元に届くまで

さて折角、レジストラとリセラの話が出てきたので、ドメイン名が生まれてから私たちの手元にやってくるまでの流れも知っておきましょう。

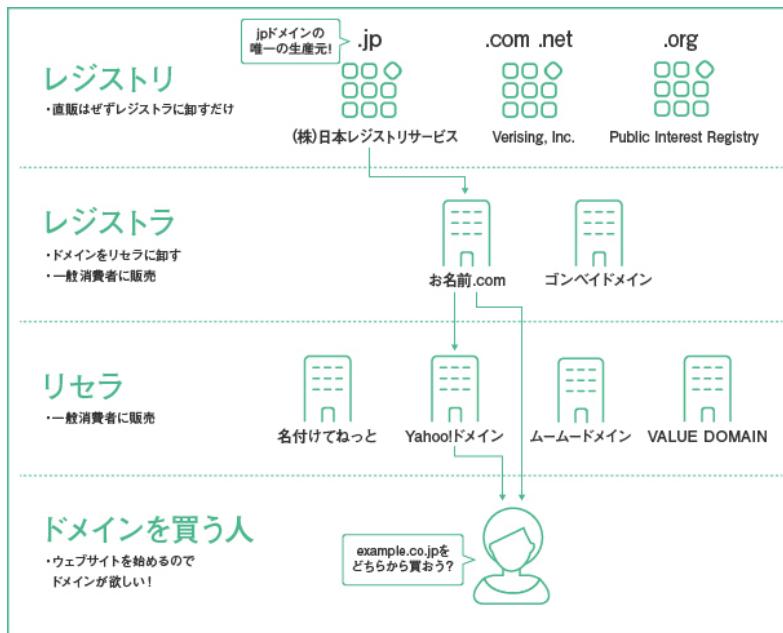
先ほどリセラ（再販事業者）はレジストラ（登録事業者）からドメイン名を仕入れて売っている、という話をしました。ではレジストラはいったいどこからドメイン名を仕入れているのでしょうか？ どこかにドメイン名が湧く泉があるのか、それとも種を植えると畑からドメイン名が収穫できるのでしょうか？

1.4.1 ドメイン名の卸元はレジストリ

残念ながらドメイン名は畑からは採れません。レジストラはドメイン名をレジストリ（登録管理組織）から仕入れています。

レジストラ？ レジストリ？ ごちゃごちゃしてきたのでちょっと図で整理してみましょう。ドメイン名は一番上のレジストリからレジストラに卸され、そこからさらにリセラに卸されます。私たちはドメイン名が買いたかったら、前述のとおりレジストラから買うこともリセラから買うこともできます。（図 1.11）

^{*14} <https://www.gmo.jp/company-profile/outline/>



▲図 1.11 レジストラとレジストリ

今回のように example.co.jp というドメイン名が欲しい場合は、リセラの Yahoo! ドメインからでもレジストラのお名前.com からでも購入てきて、実はどちらから買っても卸元は同じということです。

ドメイン名の卸元はレジストリだった、ということを知っていましたか？ お名前.com は知っていてもレジストリのことまで知っている人は少ないのではないでしょうか。このように一般にはほぼ認知されていないレジストリですが、実際ドメイン名を買うときは、ドメイン名を買う人=つまり私たちはレジストリと直接「ドメイン名登録契約」を結んでいます。間に居るレジストラやリセラはその登録契約の仲介をしているにすぎません。

では卸元であるレジストリとはいいったいどんな存在なのでしょうか？

1.4.2 1つのTLDには1つのレジストリ

そもそもドメイン名は「example.co.jp」や「yahoo.com」のように、(ドット)で区切られています。このとき、いちばん右側の jp や com を TLD (トップレベルドメイン) と呼

びます。^{*15}そしてこの TLD は、1 つにつき必ず 1 つのレジストリ（登録管理組織）^{*16}によって管理されています。

たとえば example.co.jp の TLD は jp ですが、この jp という TLD は、日本の「株式会社日本レジストリサービス」（通称 JPRS）がレジストリです。JPRS という社名を見聞きしたことはありますか？一時期は車内広告を出したりしていたのもしかしたら見たことがあるかもしれません。

それから、yahoo.com などで使われている com という TLD のレジストリは「VeriSign Global Registry Services」です。VeriSign Global Registry Services は com だけでなく net や name など複数の TLD を任されています。

2020 年の東京オリンピックに向けて販売に力を入れている.tokyo という TLD（図 1.12）は、お名前.com と同じ GMO グループに属している GMO ドメインレジストリ株式会社がレジストリです。



▲図 1.12 .tokyo ドメイン

また 2014 年ごろに Google が「.みんな」のレジストリになって、「どんな〇〇.みんなが欲しいですか？」というキャンペーン（図 1.13）をしていました。このように TLD は英語だけでなく日本語やさまざまな言語で存在しており、それぞれの TLD に必ず 1 つのレジストリが存在しています。

^{*15} TDL だと東京ディズニーランドになってしまいます。TLD です

^{*16} 「レジストリ」という言葉は、ドメイン名の割当や登録といった管理業務を行う「レジストリオペレーター」を指す場合と、そのレジストリオペレーターが登録情報を蓄積している「レジストリデータベース」を指す場合があります。本著では前者を指してレジストリと呼びます



▲図 1.13 Google のはじめよう. みんな

1.4.3 <トラブル> Mackerel のアラート誤報は io ドメイン名が原因

2017年9月、株式会社はてなが提供しているサーバ監視サービスの Mackerel で、io ドメイン名の不調によってアラート誤報を出してしまった、というトラブル（図1.14）がありました。これは io ドメイン名のネームサーバが、Mackerel の mackerel.io というドメイン名について名前解決のリクエストを受けてもきちんと応答できなかつたことで、監視対象のサーバが Mackerel システムに通信できなくなり、Mackerel システムが「監視対象のサーバから連絡が来ない！死んだのか！」と判断して誤発報してしまった、という障害でした。



▲図 1.14 Mackerel の死活監視アラートの誤報を報告するブログ

このトラブルは信用できるレジストリの TLD を選ぶ以外に回避策がありません。絶対に落とせないサイトやメールを消失させたくないメールアドレスなどで、.io のようにトラブルを起こしたことのある TLD を採用するのはあまりお勧めしません。ドメイン名を買うときは、この TLD はどこのレジストリが管理しているのだろう？過去に障害は発生していないか？などを事前に確認しておきましょう。たとえば jp ドメイン名の JPRS や tokyo ドメイン名の GMO インターネットなら、レジストリとしての実績があるため安心です。

1.4.4 【ドリル】ネットショップのドメイン名は〇〇.xxx？〇〇.com？

問題

あなたはとあるアパレルショップのウェブマーケティング担当です。このたび、担当するブランドでネットショップを作ることになりました。当初はネットショップのドメイン名を〇〇.com にする予定でしたが、ブランド名に「KISS」を含むため「〇〇.xxx」にしたい、という提案がデザインチーム内から上がりました。なお国内外で大規模な宣伝を行うため、上司からは「ネットショップがダウンすることや特定の地域や環境からサイトが

閲覧できない状況は何より避けたい」と言われています。ドメイン名はどちらにすべきでしょうか？

- A. 当初の予定通りドメイン名は○○.com にする
- B. 提案を受け入れてドメイン名は○○.xxx にする

答え _____

解答

正解は A です。.xxx はアダルトコンテンツ専用に用意された TLD で、国によっては TLD ごと閲覧をブロックされています。ブランドイメージを損なうとともに、今回のケースではネットショップが閲覧できない状況を避けるのが何より優先ですので○○.com にすべきです。

【コラム】ブランド TLD と GMO ドメインレジストリ

最近は.canon や.toshiba のように企業がブランド TLD をもつことも増えてきました。ブランド TLD は日本ですと GMO ドメインレジストリ株式会社^{*17}に、TLD の取得やレジストリとしての運用を支援してもらうケースが多いようです。

1.5 なぜレジストリがその TLD を独占するのか

このように TLD にはそれぞれ 1 社ずつレジストリがいて、その TLD の唯一の卸元となっています。

たとえば example.jp のように jp で終わるドメイン名なら、先ほどの JPRS が唯一の卸元です。example.co.jp が欲しいときも、example.ne.jp が欲しいときも、example.jp が欲しいときもすべて JPRS がレジストラに卸して（あるいはレジストラからさらにリセラに卸して）そこで私たちが買うのです。

^{*17} <https://www.gmoregistry.com/>

ちなみに御元のレジストリから、一般消費者がドメイン名を直接買うことはできません。必ずレジストラもしくはリセラを通して購入することになります。

それにしてもお名前.comで買おうがムームードメインで買おうが、jpで終わるドメイン名が売れたら必ず JPRS にお金が入るなんて！ 競争相手もいないし、御元の JPRS は独占で大儲け！ と羨ましく思いますが、実際はレジストリが丸儲けしたいがためにこんな構造になっているわけではありません。

AさんとBさんが同時に example.co.jp を買ってしまい、二人とも使おうとしてどちらのサイトを表示したらいいのか大混乱！ のようなことが起こらないようにするために、つまりドメイン名の一意性を保つためにレジストリがその TLD を一元管理しています。

前述の io ドメイン名のように、レジストリがその TLD をきちんと管理してくれないと大変なことになるので、誰でも彼でもレジストリになれる訳ではなく、レジストリになるにはものすごい大金と政府の推薦などが必要なのだそうです。（ブランド TLD はそうした条件が緩和されているため取りやすいようです）

【コラム】他のドメイン名と比べて jp ドメイン名が高い理由

レジストリはその TLD の唯一の御元ですので、もちろん御値もレジストリが決められます。新興の.work や.xyz などは御値が安いでお名前.comでも 1 円～30 円という破格ですが、jp ドメイン名は元々の御値が高いので 2,340 円^{*18}と少し高めの価格で売られています。

一度、GMO インターネットの熊谷社長が「.jp は他国の ccTLD^{*19}に比べて高すぎる。独占企業だからと高い値段で売るのではなく国際的な水準まで値下げすべきだ」と JPRS に公開質問^{*20}を投げかけたこともあるほどです。

しかし値段の分だけあって、JPRS は管理体制もしっかりしているので前述の io ドメイン名のような障害が起きるリスクは低いといえます。ひとたび障害が起きれば対応する人件費もかかりますし、jp のドメイン名が少々高くても安心代と思えば安いのかも知れません。

高いといえば 1 年で 250,000 円かかる.rich というドメイン名もあります。もしあなたがリッチなことをアピールしたければ、「自分の名前.rich」のようなドメイン名を買ってみてはいかがですか？ もちろんお名前.com で購入できますよ！

1.6 ICANN がレジストリを決める

TLD が生まれてくる源泉となるレジストリは神のごとき存在で責任重大です。そんな重要なレジストリをどの会社に任せなのか、一体誰が決めるのでしょうか？

「君が.jp のレジストリね！」と JPRS を選んだのは誰かというと、ICANN（アイキャン）という組織です。ICANN とはインターネットの IP アドレスやドメイン名などの資源を全世界的に調整・管理する非営利法人です。ここがルートネームサーバを管理しているので、全ドメイン名の本当の生産元と言ってもいいかもしれません。^{*21}

.tokyo や.shop のような新しい TLD を「作ろう！」と決める決定権があるのも ICANN ですし、TLD ごとのレジストリを決めるのも ICANN です。余談ですが HTTP が 80 で HTTPS は 443、SSH は 22 のようなプロトコルごとのポート番号を決めているのも ICANN です。

前述の.canon や.toshiba のようなブランド TLD も、ICANN に対して各社が「このブランド TLD を作りたいんです！ うちがレジストリになります！」と申請をだして、ICANN が承認することでできあがります。

1.7 取り扱う TLD はお店が自由に決められる

ところで.cat という TLD もあります。スペインのカタルーニャという場所のドメイン名で筆者のような猫好きにはたまらないのですが、残念ながら.cat はお名前.com では買なうことができません。なぜでしょう？

なぜかというとそれぞれの卸元（レジストリ）からどの TLD を入荷するか？ は、それぞれのお店（レジストラやリセラ）が自由に決められるからです。

お店の方針次第で「うちにすれば.com も.net も.jp も.info も.mobi もなんでも買えますよ」というお店にもなれるし、「うちは.jp しか扱わないよ」というお店にもなれるので、ゴンベエドメインのようにありとあらゆる TLD を入荷して品揃えを売りにしているレジストラもあれば、品ぞろえを絞っている Yahoo! ドメインのようなリセラもあるのです。

最初に「ドメイン」で検索したとき、example.co.jp は Yahoo! ドメインでは「取扱いなし」で買えなかっただですよね。これは Yahoo! ドメインが「うちでは.co.jp を扱わない」と

*18 お名前.com における 2019 年 9 月時点の値段

*19 Country Code Top Level Domain の頭文字を取って ccTLD。国ごとに日本は.jp、フランスは.fr、アメリカは.us のように割り当てられています

*20 <https://www.kumagai.com/?eid=718>, <https://www.kumagai.com/?eid=732>

*21 ルートネームサーバについて第 2 章「DNS の仕組み」で後述します

いう判断をしたからです。

以上がレジストリ、レジストラ、リセラの関係でした。

1.8 「ドメイン」なのか「ドメイン名」なのか？

ところで、ちょっとここで言葉の整理をしましょう。先ほど、購入のために各社の値段を比較した「example.co.jp」は「ドメイン」でしょうか？それとも「ドメイン名」なのでしょうか？²²結論から言うと、「example.co.jp」はドメイン名です。

たとえば

- startdns.fun
- example.co.jp
- example.jp
- example.com

というドメイン名があったとき、次の図（図 1.15）のように example（3箇所にある）や co や jp や com といった個々の名前空間²³の領域を「ドメイン」と呼びます。²⁴それぞれのドメインを「.(ドット)」でつないで一意に識別する「example.co.jp」や「example.com」といった名前がドメイン名²⁵です。²⁶

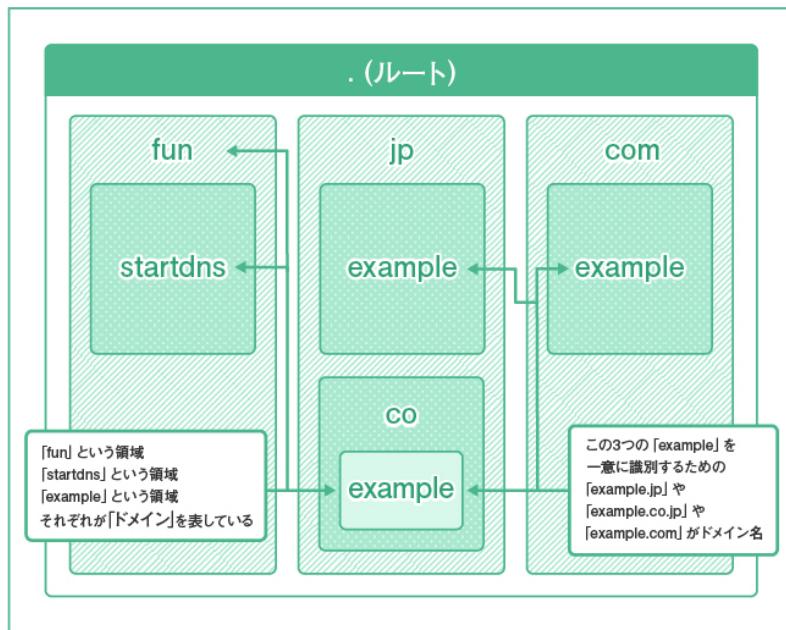
*²² 初版を書いたときは 2 つの表記の違いが分かつておらず、ドメインもドメイン名もごっちゃに書いていました。書くことによって「分かったつもりになっていた自分」が次々見つかるので、これからもまだまだ「DNS なにも分からぬ」の谷が来るんだろうなと思います

*²³ 名前空間とは、その中のものが一意な名前を持つ空間のことです。たとえばパソコンのデスクトップ上にある「会議資料」というフォルダ（名前空間）の直下には、「議事録.txt」という名前のファイルは 1 つだけで、同名のファイルをもう 1 つ作ることはできません。ですが、「会議資料」の中にさらに「全社総会」というフォルダを作ると、そこは別の名前空間なので「議事録.txt」というファイルを作ることはできます。PHP などのプログラミング言語においても、名前空間は自分が作った関数と PHP の組み込みの関数の名前が衝突してしまうときの解決手段として存在しています

*²⁴ 英語のドメイン（domain）を直訳すると「範囲」とか「領域」のことなので、「この領域をドメインと呼びます！」と言われても「この猫をキャットと呼びます！」という感じですごくアホっぽいのですが、そうとしか説明できなくて広い気持ちで許されたい

*²⁵ より正確に言うと example.co.jp は FQDN（Fully Qualified Domain Name、日本語だと完全修飾ドメイン名）で、個々の example や co や jp は相対ドメイン名です。「どこ行きたいの？」と問われたときに「大阪府大阪市中央区！」と言うのが FQDN で、大阪府大阪市に居る状態で「中央区！」と言うのが相対ドメイン名というイメージですね

*²⁶ 本当はドメイン名の末尾には「ルート」というドメインを表す。（ドット）があるので、省略されることが殆どです

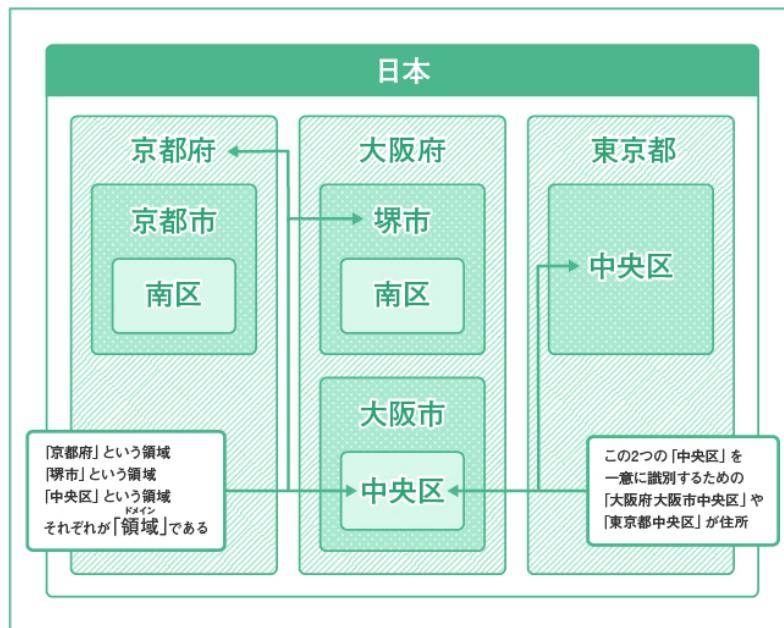


▲図 1.15 個々の名前空間がドメインで、識別するための一意な名前がドメイン名

これは

- 東京都中央区
- 大阪府大阪市中央区
- 大阪府堺市南区
- 京都府京都市南区

のような住所と大変よく似ています。次の図（図 1.16）のように、先ほどの example や startdns と同様、中央区や南区と呼ばれるエリアは国内の複数箇所に存在していますが、都道府県からの住所を言えば一意に識別できます。



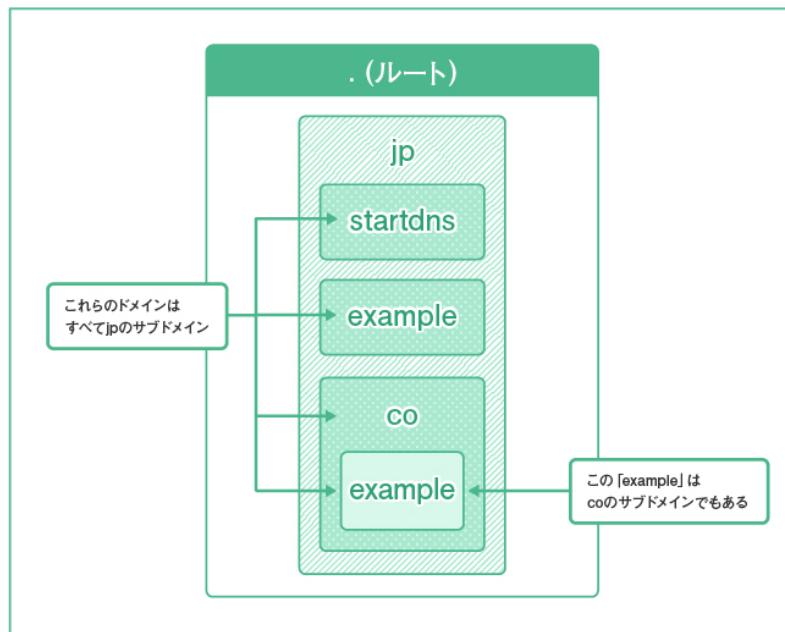
▲図 1.16 example 同じように中央区は複数箇所にあり、一意に識別するために住所がある

1.8.1 サブドメインとは？

このとき、jp というドメイン（領域）に内包されているドメインは、すべて「jp のサブドメイン」と呼ばれます。次の図（図 1.17）のとおり

- startdns
- example
- co
- (co の中の) example

といったドメインはすべて jp のサブドメインです。さらに co の中の example は co というドメインにも含まれているので、「co のサブドメイン」でもあります。



▲図 1.17 example と同じように中央区は複数箇所にあり、一意に識別するために住所がある

それぞれのドメインを一意な名前、つまりドメイン名で呼んで説明すると次のようにになります。

- startdns.jp は jp のサブドメインである
- example.jp は jp のサブドメインである
- co.jp は jp のサブドメインである
- example.co.jp は jp のサブドメインである
- example.co.jp は co.jp のサブドメインである

ちなみにドメイン名を買うと、サブドメインは作りたい放題です！ サブドメインを作るたびに、ドメイン名を買ったときのような支払いは発生しません。

1.9 実際にお名前.com でドメイン名を買ってみよう

では実際に、お名前.com でドメイン名を買ってみましょう。「お名前.com」で検索してトップページ（図 1.18）を開きます。

1.9 実際にお名前.comでドメイン名を買ってみよう



▲図 1.18 お名前.com のトップページ

トップページにある「全ドメイン検索」をクリックして、ドメイン検索の画面（図 1.19）を開きます。



▲図 1.19 ドメイン検索

好きな文字²⁷を入力して、検索ボタンをクリックします。（図 1.20）筆者は「startdns.○○」というドメイン名が欲しいので、startdnsと入力しました。

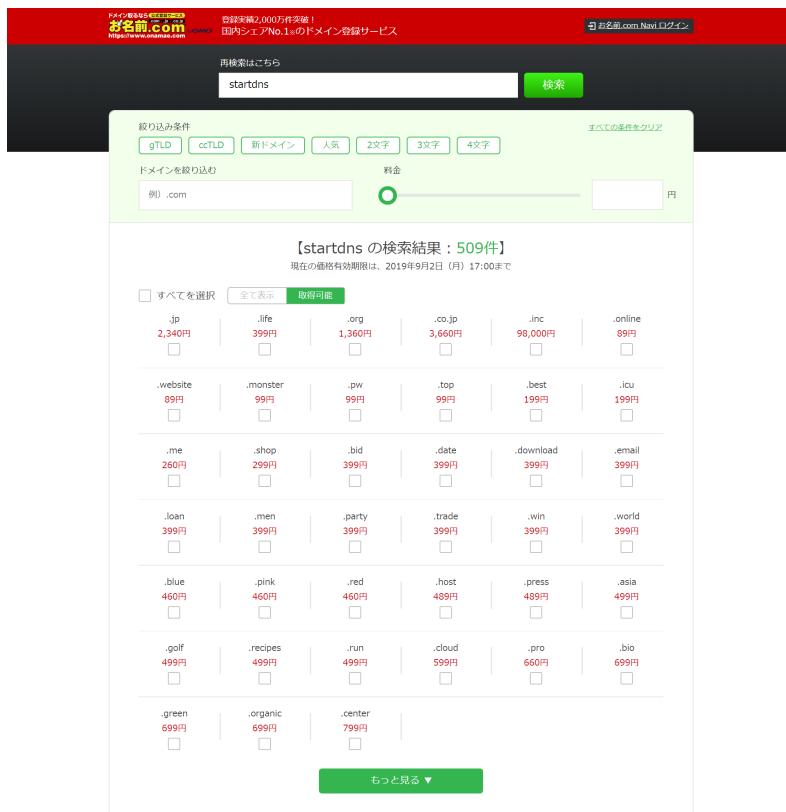
*27 ドメイン名には_（アンダーバー）は使えませんのでご注意ください。英数字や-（ハイフン）は使えます



▲図 1.20 好きな文字を入力して検索

すると次の画面で「これは買えるよ、これは売り切れたよ」と教えてくれます。「全て表示」から「取得可能」に切り替えると、現時点で購入可能なドメイン名だけに絞り込まれます。気に入るものが見つかるまで「もっと見る▼」を押してどんどん表示してみましょう。

1.9 実際にお名前.comでドメイン名を買ってみよう



▲図 1.21 取得可能なドメイン名だけを表示して選ぼう

今回はちゃんとしたサービスで使う訳ではなく、ドメイン名や DNS の仕組みを学ぶことが目的なので 1 円～100 円程度の安いドメイン名で構いません。それから詳細は後述しますが .jp は持ち主の名前が Whois で出てしまうため、本名をインターネットで公表したくない人にはお勧めしません。

また記載されているのは有効期間 1 年のドメイン名の料金ですので、いま買うと 1 年後に更新タイミングがやってきます。前述のとおり 2 年目以降はぐっと値段が上がったりしますので、TLD を決めたら更新料金^{*28}も確認しておきましょう。購入するドメイン名が決まつたらチェックを入れてください。

選択肢がいくつかある中で筆者は startdns.fun を買うことに決めて、.fun の下にあるチェックボックスにチェックを入れました。(図 1.22) 「1 ドメイン選択中」と表示され

*28 <https://www.onamae.com/service/d-price/>

ていることを確認したら、「お申し込みへ進む」をクリックします。



▲図 1.22 欲しいドメイン名にチェックを入れて「お申し込みへ進む」

次に表示されるお申込内容の確認画面（図 1.23）でとても大切な作業があります。ここでドメイン名の値段を再度確認するとともに、「登録年数/プラン/オプションなど」の欄に「Whois 情報公開代行」と表示されていることを必ず確認してください。Whois 情報公開代行オプションはドメイン名を買うタイミングで申し込みれば無料ですが、後から頼もうとすると 980 円もかかってしまいます。Whois 情報公開代行がどういうものなのかは後述します。

1.9 実際にお名前.comでドメイン名を買ってみよう



▲図 1.23 「Whois 情報公開代行」と表示されていることを確認

なお画面下の方で、まるでセットのポテトのように「一緒にサーバはいかがですか?」と聞かれますが、今回はサーバの契約は不要ですのでデフォルトの「利用しない」のままで構いません。右側の「初めてご利用の方」に自分のメールアドレスとパスワードを入れたら「次へ」を押してください。ここで入力したパスワードはこの後すぐに使用しますのでしっかりメモしておいてください。(表 1.2)

お名前 ID が発行されたら会員情報の登録画面に進みます。画面上部に表示されているお名前 ID もメモしておいてください。(図 1.24) 今回は個人として購入しますので種別は「個人」を選択して名前や住所、電話番号、メールアドレスなどを入力します。ドメイン名を買った直後、ここで入力したメールアドレスに対して重要なメールが届きますので必ずメール受信が可能なアドレスを入力してください。必須になっている箇所をすべて入力したら右側にある「次へ進む」を押します。

▼表 1.2 お名前.com に登録した情報

項目	例	あなたが登録した情報
メールアドレス	startdns.01@gmail.com	
パスワード	自作のpassword	
ドメイン名	startdns.fun	
お名前 ID	8397576	

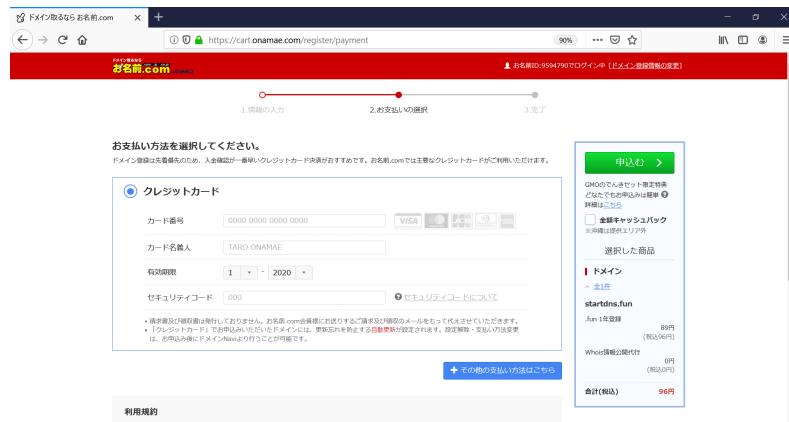
The screenshot shows a web browser window for 'お名前.com' (Onamae.com) with the URL https://cart.onamae.com/register/input_account. The page title is 'お名前ID登録' (Registration of Oname ID). A success message at the top states: '続いて必要事項をご入力ください。' (Please enter the following required items.) and 'お名前ID:8397576 が発行されました。' (The Oname ID: 8397576 has been issued.). Below this, there is a form titled '会員情報の入力' (Member Information Input) with various fields for personal information. The 'お名前' (Name) field contains 'お名前' (Onamae), '姓' (Last name) '太郎' (Taro), and '名' (First name) 'Onamae'. Other fields include '国' (Country) 'Japan', '郵便番号' (Postal code) '123-4567', '都道府県' (Prefecture) '北海道', '市区' (City/Zone) '渋谷区' (Shibuya-ku), '町村' (Town/Village) '桜ヶ丘町' (Sakuragaokacho), '番地' (Address number) '26-1', '建物名' (Building name) 'セルリアンタワー 11F' (Cerulean Tower, 11F), '電話番号' (Phone number) '090-XXXX-XXXX', and 'メールアドレス' (Email address) 'example@gmo.jp'. A green button on the right says '次へ進む >' (Next Step).

▲図 1.24 会員情報の登録画面でお名前 ID を確認

次は支払方法の選択画面（図 1.25）が表示されます。右側の「お申込み内容」に、選択したドメイン名の値段および Whois 情報公開代行（0円）が表示されていることを確認してください。「GMO のでんきセット限定特典」は今回は不要ですので、「全額キャッシュバック」のチェックは外しておきましょう。お申込み内容に誤りがなければクレジットカード情報を記入^{*29}して右側にある緑色の「申込む」を押します。

^{*29} クレジットカードの他にコンビニ支払いや銀行振込、請求書払いが可能ですが説明は省略します

1.9 実際にお名前.comでドメイン名を買ってみよう



▲図 1.25 支払方法の選択画面でお申込内容を確認

「ねえ、本当にサーバー要らないの？」というアラート画面（図 1.26）が出てきますが、本当に不要ですので「申込まない」を押します。



▲図 1.26 サーバは不要なので「申込まない」を選択

おめでとうございます！ 無事に自分のドメイン名が買えたら購入完了画面（図 1.27）が表示されます。先ほど登録したメールアドレス宛てに「[お名前.com] ドメイン登録 料

金ご請求／領収明細」というメールも届いていると思いますので確認してください。^{*30}



▲図 1.27 ドメイン名の購入完了画面が表示された

さてめでたくドメイン名が買えたのですが、ここで必ず設定しておかないといけないことがあります。

【コラム】類似のドメイン名を確保しておくべきか？

ヤマト運輸や佐川急便といった宅配業者の不在通知などを装って、SMS^{*31}に書かれたURLから偽サイトに誘導し、個人情報を盗み取る「スミッシング^{*32}」というサイバー攻撃^{*33}があります。たとえば佐川急便では sagawa-exp.co.jp というドメイン名を使用していますが、偽サイトは sagawa-umo.com^{*34}のようなドメイン名です。(図 1.28)

*30 お名前.comでドメイン名を買うと毎日何通も「.co.jpのクーポンを発行したよ！」「今ならこんないいドメイン名が空いてるよ！」と広告メールが届くので、購入完了画面の「メールマガジンの確認」の欄にある「お受け取りを希望されないお客様はこちらをご参照ください。」というリンクから「お名前.comドメインニュースの配信停止」を行っておくことをお勧めします。https://help.onamae.com/app/answers/detail/a_id/8126/



▲図 1.28 佐川急便の偽サイトを開こうとするとブラウザが警告してくれる

こうしたフィッシング詐欺や、あたかも公式かのように誤認させるサイトによる被害を防ぐ方法の1つに、類似のドメイン名を予め確保しておく、というものがあります。たとえば任天堂株式会社の国内向けオフィシャルサイトで使用しているドメイン名は `nintendo.co.jp` ですが、任天堂では `nintendo.jp` というドメイン名も登録しています。

このように「株式会社イグザンブル」が `example.co.jp` というドメイン名を登録するとき、その時点で使用する予定がなくても `example.jp` と `example.com` を一緒に確保しておく、というやり方はよく見かけます。

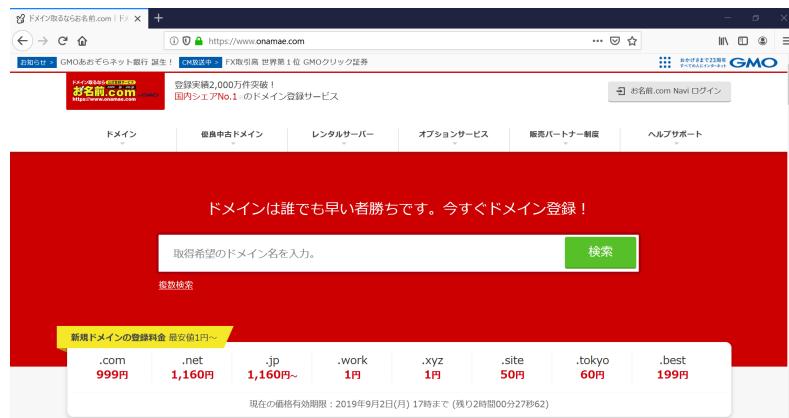
ただ TLD はたくさんの種類があるため、`example.jp` と `example.com` は確保したけど `example.inc` は他人に取られてもいいの？ `example.site` や `example.net` は放っておいていいの？ というように、いったいどこまで確保すべきなのか、あるいは `example.jp` と `example.com` すらも本当に確保しておく意味があるのか、は迷うところかもしれません。ドメイン名は登録した分だけお金がかかります。類似のドメイン名すべてを確保しておくのは現実的ではないので、事業の規模や、フィッシング詐欺の対象となるようなサービスなのか否かを考えて決めるのがよいと思います。

フィッシング詐欺に限らず、著名な社名やサービス名をドメイン名に含めることで誤認によってサイトに集客する、あるいは今後使われそうなドメイン名を先に登録して高値での転売を持ちかける、といった不正行為に対しては、ドメイン紛争処理方針（DRP）^{*35}という仕組みでも対応が可能です。正当な権利者が申し立てを行い、DRP の紛争処理機関に認められれば、そのドメイン名の登録を取

り消したり権利者に移転したりできます。

1.9.1 自動更新はオフにしておこう

ドメイン名の購入が完了したら、お名前.com のトップページに戻って右上の「ドメイン Navi ログイン」をクリックしてください。(図 1.29)「ドメイン Navi」とはお名前.com の管理画面のことです。ログイン画面(図 1.30)が表示されたら、先ほど発行されたばかりのお名前 ID^{*36}とパスワードを入力して「ログイン」を押してドメイン Navi にログインします。^{*37}



▲図 1.29 右上の「ドメイン Navi ログイン」をクリック

*³¹ 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

*³² SMS を用いたフィッシング詐欺のこと。SMS とフィッシング(phishing)を組み合わせてスミッシング(Smishing)と呼ばれている

*³³ 偽 SMS で個人情報狙う「スミッシング」が激化 携帯事業者装い新手口も <https://www.sankei.com/affairs/news/190415/afr1904150018-n1.html>

*³⁴ 言わずもがなですが、偽サイトにはアクセスしないようご注意ください

*³⁵ Domain name dispute Resolution Policy の頭文字を取って DRP と呼ばれる。ドメイン名の不正な登録や使用によって生じた紛争を解決するための仕組み

*³⁶ もしお名前 ID をメモし忘れてしまったらドメイン登録したときに届くメールにも記載されています

*³⁷ 「ログイン」を押すと、操作しているのが本当に人間なのかを確認するために「信号機のタイルをすべて選択してください」のような reCAPTCHA の画面が表示されることがあります。筆者はよく失敗します。頑張って信号機のタイルだけを選んでください

1.9 実際にお名前.com でドメイン名を買ってみよう



▲図 1.30 お名前 ID とパスワードを入れてドメイン Navi にログイン

ログインするとドメイン Navi のトップページではなく「ドメイン契約更新」の画面(図 1.31)が表示されます。^{*38}



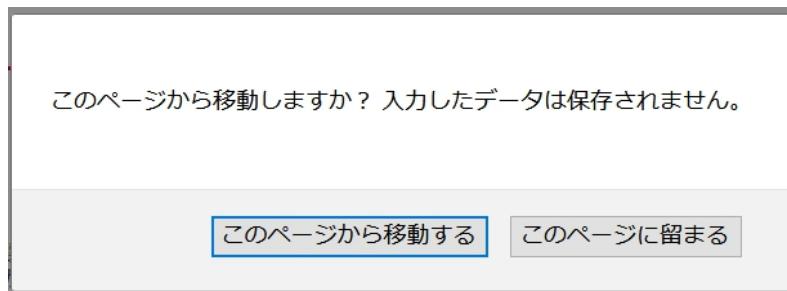
▲図 1.31 ドメイン Navi ドメイン契約更新

「ドメイン契約更新」の画面が表示されるのは、ドメイン Navi がログインするたびに「来年の更新をしませんか？」と聞いてくる仕様だからです。ついさっきドメイン名を買ったばかりで有効期限は 1 年先ですので無視して先へ進みましょう。先ほど買ったドメ

^{*38} お名前.com では定期的に AB テストを行っているのか、ログインするお名前 ID によってドメイン Navi の見た目が大きく異なることがあります。人によって本著とお手元で見ている画面の見た目が一致しないかも知れません。ですが見た目にかかわらず、やりたいことは「今買ったドメイン名で自動更新をオフにしたい」だけですので、画面が違っていても負けずに先へ進みましょう

イン名の設定変更がしたいので、上部のメニューで「ドメイン一覧」をクリックしてください。

他のページへ移動しようとすると「このサイトを離れますか？ 行った変更が保存されない可能性があります。」と警告が出る（図 1.32）のでちょっとドキッとしますが、ドメイン名の有効期限は1年も先ですので今更新しなくともまったく問題ありません。強い気持ちで「このページから移動する」を押してください。



▲図 1.32 ドメイン Navi 更新アラート

ドメイン一覧（図 1.33）を開くと、今買ったばかりのドメイン名（筆者なら startdns.fun）が表示されます。自分が買ったドメイン名をクリックしてください。

▲図 1.33 ドメイン Navi ドメイン一覧

ドメイン詳細（図 1.34）が表示されたら「自動更新」という項目を確認してください。きっと「設定済み」になっていると思います。

1.9 実際にお名前.com でドメイン名を買ってみよう

The screenshot shows the 'お名前.com Navi' website interface. At the top, there are tabs for 'お名前.com 買い方' (How to Buy), 'お名前.com Navi' (selected), 'お名前.com レンタルサーバー' (oName.com Rental Server), and 'お名前.com ドメイン登録' (Domain Registration). The top right includes user information ('お名前ID: [8397576] ログアウト'), a 'FAQ' link, and a 'レンタルサーバー管理' (Server Management) link.

The main navigation bar has tabs for 'ドメイン一覧' (Domain List), 'ドメイン設定' (Domain Settings), 'お名前.com会員情報' (oName.com Member Information), and 'オプション設定' (Option Settings). A blue circle highlights the 'ドメイン一覧' tab.

The 'Domain Details' section is highlighted with a red border. It displays the domain name 'startdns.fun'. Below it, a message says '対象ドメインの詳細情報を表示します。' (Displaying detailed information for the target domain).

Two sections are listed: '対象ドメイン' (Target Domain) containing 'startdns.fun' and 'ドメイン詳細' (Domain Details) which is currently selected. A blue button labeled '変更履歴を見る' (View Change History) is visible next to it.

The 'Domain Details' table contains the following information:

ドメイン名	startdns.fun
登録日	2018/03/01
更新期限日	2020/03/01
登録期限日	2020/03/01
自動更新	設定済み
ドメイン状態	移動中
Whois情報公開代行	サービス利用中
Whois代行メール転送オプション	未設定
Whois代行メール転送アドレス	未設定
ドメインプロテクション	未設定
AuthCode	表示する

▲図 1.34 ドメイン Navi ドメイン詳細

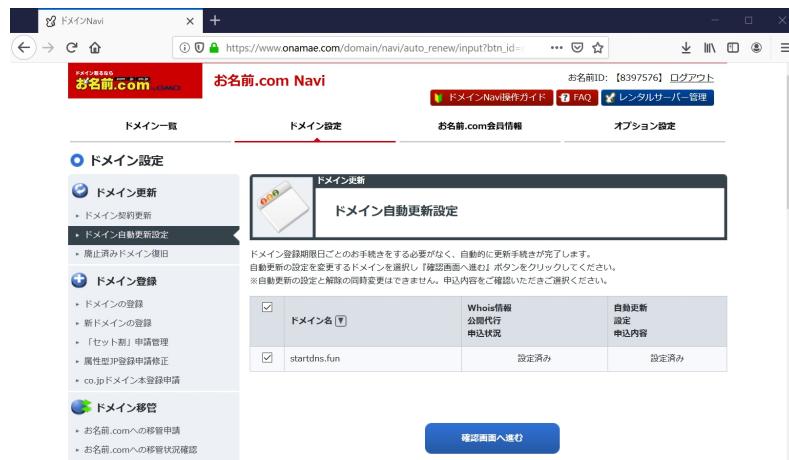
この設定を変更したいので、上部の「ドメイン設定」から「ドメイン自動更新設定」を押します。(図 1.35)

第1章 ドメイン名とWhois



▲図 1.35 「ドメイン自動更新設定」をクリック

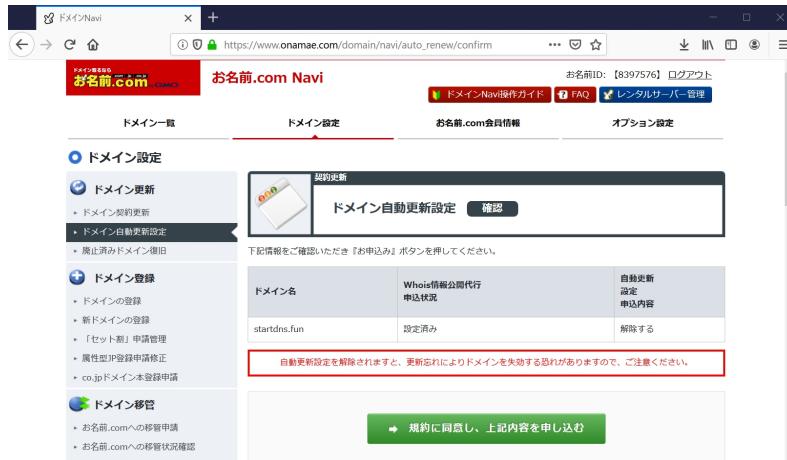
ドメイン自動更新設定（図 1.36）の画面で、自動更新をオフにしたいドメイン名にチェックを入れたら「確認画面へ進む」をクリックして自動更新を解除します。



▲図 1.36 ドメイン Navi ドメイン自動更新設定

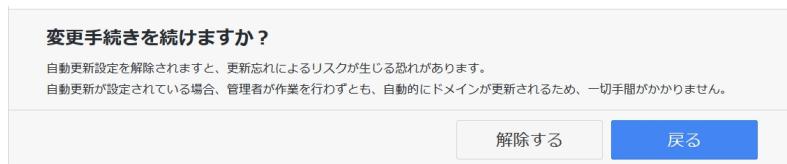
「自動更新設定を解除されますと、更新忘れによりドメインを失効する恐れがありますので、ご注意ください。」と表示されます（図 1.37）が構いません。「規約に同意し、上記内容を申し込む」を押してください。

1.9 実際にお名前.comでドメイン名を買ってみよう



▲図 1.37 ドメイン自動更新解除の確認

再度「変更手続きを続けますか？」という確認（図 1.38）が表示されますので、「解除する」を押してください。



▲図 1.38 ドメイン自動更新解除の確認

図 1.39 のように「ドメイン自動更新解除」が「完了」になったら設定変更は完了です。



▲図 1.39 「ドメイン自動更新解除」が完了になっていたら OK

いまドメイン Navi で確認したとおり、購入時のデフォルト設定ではドメイン名は「自動更新がオン」になっています。購入時の設定のまま放っておくと来年も再来年もドメイン名は更新されてクレジットカードからドメイン名代が引き落とされていきます。自動更新は最初にオフにしておきましょう。

「でも大事なウェブサイトならドメイン名の更新忘れを防ぐため、むしろ自動更新にしておくべきじゃないの？」と思われるかも知れません。しかし実は自動更新には恐ろしい落とし穴があるのです。

1.9.2 <トラブル> ドメイン自動更新の落とし穴

前述のとおりドメイン名は1年ごとに更新されます。対してクレジットカードは基本的に有効期限が5年です。

たとえば、とある広告代理店 A 社でクライアント B 社の新製品サイト用にドメイン名を買ったとします。本来であればきちんと経理を通して請求書払いでの購入すべきですが、サービスインまで日もなかったためディレクターの C さんが「後で経費精算すればいいや」と考えて自分の個人クレジットカードで買い、Web 制作会社 D 社とウェブサーバを運用している E 社へ「ドメイン名は買ついたからこれを使って」と伝えました。^{*39}

ディレクター C さんがドメイン名を買った2年後に転職して会社を去り、そこからさらに3年後に C さんのクレジットカードの有効期限が来ました。お名前.com からは再三に渡って「与信に失敗して決済できなかったよ！ クレジットカード情報を新しくして！」

*39 重ねて書いておきますがこのトラブルはフィクションです。登場する A 社や C さんなどの人物・団体・名称等はすべて架空であり、実在のものとは関係ありません

更新しないと期限切れちゃうよ！」というメール^{*40}が届きますが、宛先は辞めてしまったCさんのメールアドレスなのでA社の人は誰も気づきません。

そしてある日ドメイン名の期限が切れて、突然B社の新製品サイトが見られなくなり、お問い合わせメールも届かなくなって一体何が起きた？！と大騒ぎになります。

「サーバが落ちたのか？」「いいや落ちていない！」「調べたらドメイン名の期限切れらしいぞ、早く更新しなければ！」「誰だ？誰がドメイン名を更新してたんだ！」「クライアントのB社か？委託先のD社か？」「いいや、ドメイン名だしサーバを面倒みてるE社じゃないのか？」「それともうちか？」とA社の中はてんやわんやです。

後任ディレクターのFさんが過去のメールをひっくり返して調べた結果、ようやく「ドメイン名は3年前に辞めたCさんが買っていたらしい」と分かりましたが、Fさんは直接Cさんに面識がありません。まして退職直後ならまだしもCさんがA社を辞めてから既に3年が経過していたため連絡を取るまでがまた大変です。Cさんの連絡先を知っている人を探し回ってFさんはなんとかCさんを捕まえましたが、Cさんも既にお名前.comのお名前ID（会員ID）やパスワードを忘れており…失効直後だったらまだ買い戻せたドメイン名は、そうこうしているうちに日にちが経過してまた市場へ売りに出されてしまい、まったく関係のない業者に買われてしまいました。後日、大枚をはたいてなんとかドメイン名を譲ってもらったものの、このトラブルを通してクライアントB社から広告代理店A社への心象はすっかり悪くなってしまいました…。

いかがでしたか？想像しただけで怖いですよね。こんなトラブルを起こさないためにも、ドメイン名の自動更新をオンのままにしておくときは慎重に！そして更新のお知らせが届く連絡先のアドレスは個人のメールアドレスではなく、チームのメールアドレスなどにしておきましょう。

1.9.3 【ドリル】連絡先メールアドレスは自分だけでいい？

問題

あなたはとあるスイムウェアブランドのマーケティング担当者です。今年の夏向けに水着キャンペーンサイトを作るので新しくドメイン名を取ることにしました。お名前.comでドメイン名を買うとき、連絡先メールアドレスには何を入れるべきですか？

- A. 担当者である自分のメールアドレス
- B. マーケティングチームのメーリングリスト
- C. 個人情報流出が怖いので「test@example.com」のような適当なメールアドレス

^{*40} ドメイン名の期限が切れる15日前に自動更新をしようと試み、与信に失敗して「[お名前.com] ドメイン自動更新 与信失敗」という件名のメールが届きます

答え _____

解答

正解はBです。ただしIT系の会社では部署編成などで部署そのものがなくなり、部署に紐づいたメーリングリストもなくなってしまうことがあります。できるだけ長持ちしそうな社内メーリングリストを選びましょう。本著で買ったドメイン名のように個人利用のためのドメイン名であれば、連絡先メールアドレスも個人のもので構いません。

【コラム】ドメイン名は「買う」ものか？

さて、ここまでドメイン名を「買う」「購入する」のように表現してきましたが、ドメイン名は本当に「買う」ものなのでしょうか？^{*41}ちょっと立ち止まって考えてみましょう。

たとえば花屋さんで一輪のガーベラを買ったら、そのガーベラはあなたの物です。あなたの部屋に飾ってもいいし、他の人にあげてもいいし、瓶詰めのハーバリウムにしてメルカリで売っても構いません。ですが先ほどお名前.comで「買った」ドメイン名は、花と同じようにあなたのものなのでしょうか？これからずっといつまでも使えるのでしょうか？

実はドメイン名は「買う」ものではなく、登録申請を出して「登録する」ものです。お名前.comのサイトでも「国内シェアNo.1のドメイン登録サービス」「今すぐドメイン登録！」のように、しっかり「登録」と書かれています。

たとえば「startdns.fun」というドメイン名を使いたい！」と思ったら、そのドメイン名を「買う」ではありません。レジストラ（お名前.com）を通して、.funというTLDを管理しているレジストリ（DotSpace, Inc.^{*42}）に対してドメイン名の登録を申請するのです。登録申請を受け付けたレジストリが、内容を確認した上でレジストリデータベースに情報を登録することでドメイン名の登録は完了します。

ドメイン名を登録することで、私たちはそのドメイン名をレジストリが定めたルールの範囲内で一定期間自由に使う権利を得ます。^{*43}登録したドメイン名には有効期限がありますが、期限が切れる前に更新の手続きをすれば引き続き使用できます。有効期間は基本的に1年単位です。細かな規定はレジストラやTLDに

もよりますが、複数年の登録もできますので、長く使うことが先に分かっているドメイン名であれば、最初に登録するタイミングで「10年の登録」をしておくことも可能^{*44}です。

有効期限が切れるごとにそのドメイン名は使えなくなり、さらに一定期間が経過すると再び誰でも登録できる状態になります。携帯電話の電話番号と同じようなイメージですね。

ですので、ドメイン名は正確には「買う」ものではなく、花と違って自分の所有物にもなりません。それを踏まえた上で、本著では感覚的な分かりやすさを優先して「ドメイン名を買う」「購入したドメイン名」「ドメイン名の持ち主」といった表現を用いることとしています。

1.10 Whois とは

ところで先ほどドメイン名を購入するとき「Whois 情報公開代行」と表示されていることを確認しました。もし「Whois 情報公開代行」のオプションを付けずにドメイン名を購入した場合、「Whois 情報を登録してください」という住所や氏名を入力するページが出てきます。

この Whois 情報というのは一体何なのでしょう？ 自宅の住所とか氏名とか、絶対登録しなきゃいけないのでしょうか？

- A. 絶対登録しなきゃいけない！
- B. 任意だから登録しなくてもいい

答え _____

正解は A です。Whois 情報は絶対に登録しなければいけません。ではその絶対に登録しなければならない Whois とは何なのでしょう？

*⁴¹ ドメイン名の登録 - JPNIC <https://www.nic.ad.jp/ja/dom/registration.html>

*⁴² <https://radix.website/>

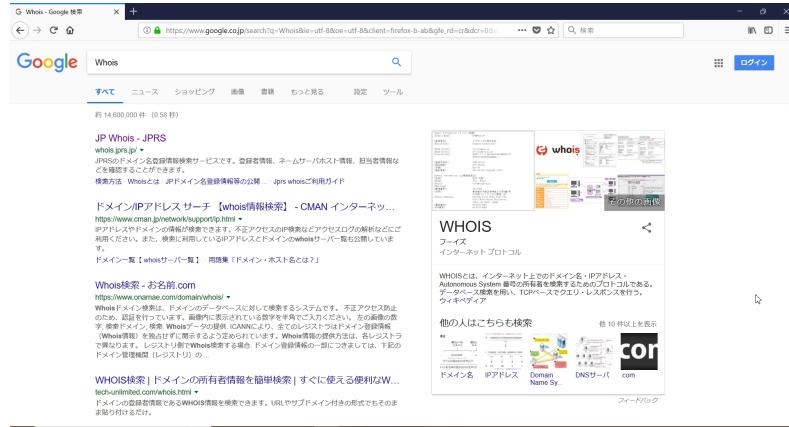
*⁴³ より正確に言うと、「委任」という形でそのドメイン名の管理権限を任せられます。委任については第 2 章「DNS の仕組み」で詳しく説明します

*⁴⁴ 複数年の登録をしていたのにレジストラが倒産してしまった、という場合のリスクについては後述します

第1章 ドメイン名と Whois

Whois とは、そのドメイン名を所有している組織や担当者の氏名、連絡先（住所・電話番号・メールアドレス）、ドメイン名の有効期限などがインターネットで誰でも見られるサービスのことです。本当にインターネットで誰でも見られます。

Whois で検索（図 1.40）すると Whois 情報を見るためのサイトがたくさん出てきます。でもドメイン名の所有者情報が見られるなんて一体誰がそんなサービスを提供しているのでしょうか？



▲図 1.40 Whois で検索するとたくさんのサイトが出てくる

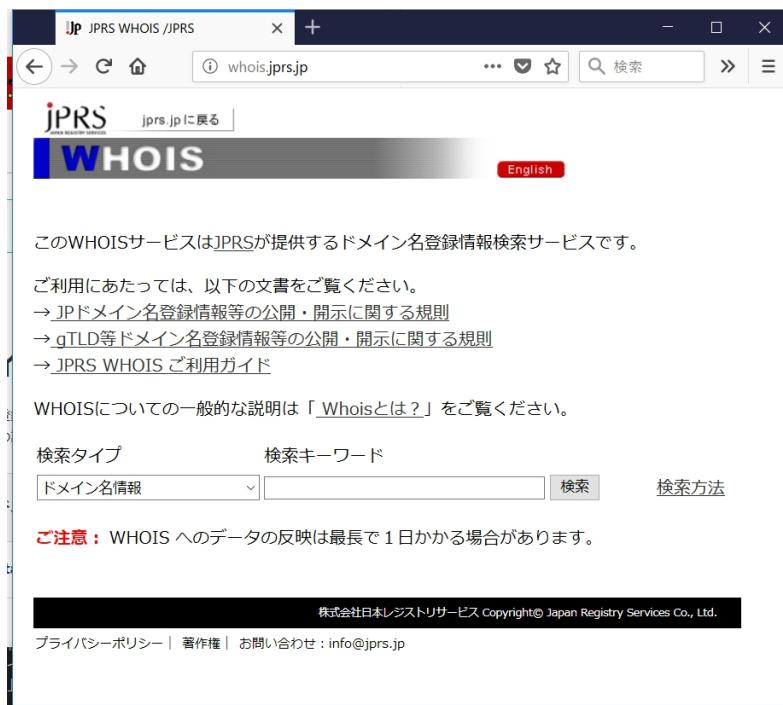
Whois 検索はレジストリが提供しているサービスです。レジストリ、覚えていませんか？ついさっき出てきましたね。TLD1 つにつき必ず 1 つ存在している、唯一の御元であるレジストリです。思い出せましたか？

レジストリがその TLD の Whois 情報を管理・公開しているのです。

1.10.1 JPRS WHOIS でドメイン名の所有者情報を見てみよう

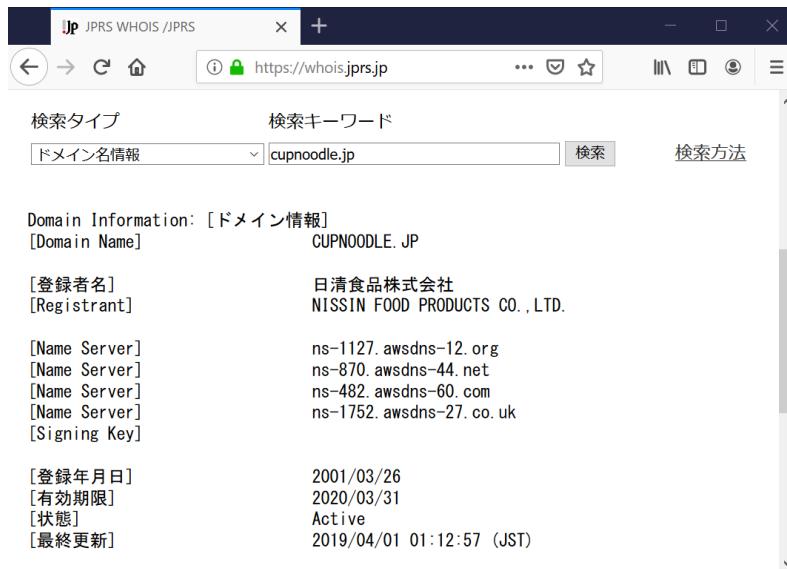
たとえば.jp で終わるドメイン名を管理している JPRS が提供する「Whois 情報確認サイト」はこちら（図 1.41）*45です。

*45 <http://whois.jprs.jp/>



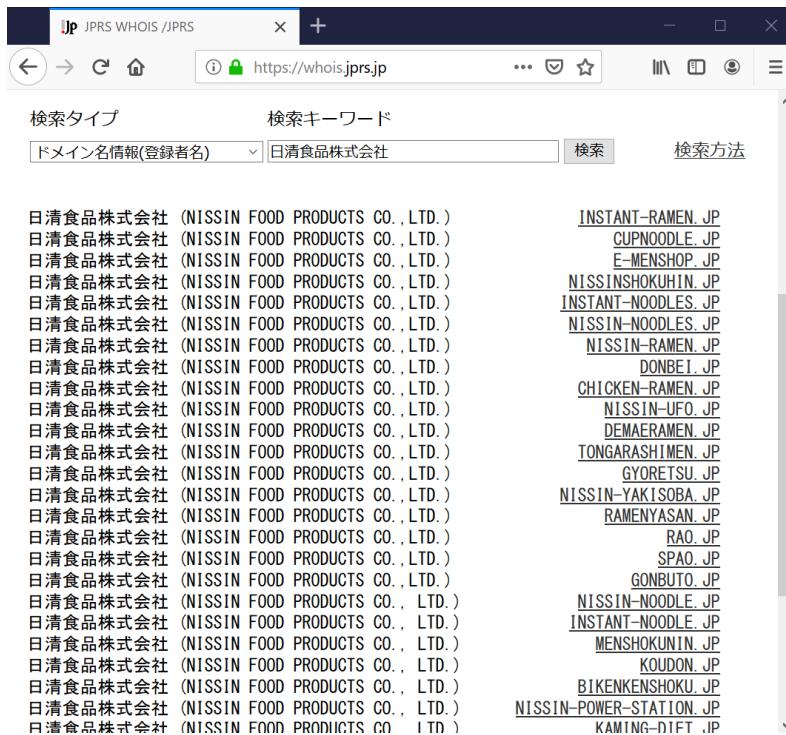
▲図 1.41 JPRS WHOIS

例として検索キーワードに日清カップヌードルの「cupnoodle.jp」を入れて検索(図 1.42)してみると、cupnoodle.jp の Whois 情報が出てきます。登録者名を見れば「cupnoodle.jp」というドメイン名の持ち主は日清食品株式会社なんだな」と分かりますし、登録年月日や有効期限を見れば「2001/03/26 から使い始めて 2020/03/31 が次の更新タイミングなんだな」と分かります。



▲図 1.42 cupnoodle.jp の Whois 情報

このように Whois を使えばドメイン名から所有者を調べることができます。それだけではなく逆に所有者名から所有しているドメイン名の一覧を調べることも出来ます。今度は検索タイプのプルダウンを「ドメイン名情報（登録者名）」にして、検索キーワードに「日清食品株式会社」と入れて検索（図 1.43）してみましょう。DONBEI.JP や CHIKINRAMEN.JP など日清食品が所有しているドメイン名の一覧が出てきました。



▲図 1.43 登録者名が日清食品株式会社のドメイン名

【コラム】ドメイン名は大文字小文字の区別をしない

ところで JPRS WHOIS のサイトで「cupnoodle.jp」を入れて検索（図 1.42）してみると、一番上のドメイン名が CUPNOODLE.JP のように大文字になっています。CUPNOODLE.JP ではなく、cupnoodle.jp を検索したのになぜでしょうか？

実はドメイン名は大文字小文字の区別をしません。cupnoodle.jp も CUPNOODLE.JP も cUpNoOdLe.Jp もすべて同一のドメイン名です。

ためしに大文字小文字混ぜこぜにした <https://www.cUpNoOdLe.Jp/> をブラウザで開いてみると、ちゃんとカップヌードルのサイトが見られます。特に最近はブラウザが勝手に大文字を小文字に置換してくれるので、たとえば商品パッケージや紙媒体で目立たせるために URL のドメイン名の部分を全て大文字にしても

サイトには問題なくアクセスできます。

1.10.2 Whois の項目はレジストリごとに微妙に違う

ではプルダウンを「ドメイン名情報」に戻して今度は検索キーワードに「yahoo.com」を入れ、再びドメイン名から所有者名を調べてみましょう。(図 1.44)



▲図 1.44 yahoo.com を検索すると「該当するデータがありません。」と出る

yahoo.com の所有者を検索すると、なぜか「該当するデータがありません。」と出てきました。yahoo.com はちゃんと実在するドメイン名なのになぜでしょう？

なぜならば、この JPRS WHOIS というサイトは JPRS が管理・提供している「.jp で終わるドメイン名の Whois 情報が検索できるサイト」なので、com や net といった jp 以外の TLD は対象範囲外だからです。

前述のとおり Whois はそれぞれのレジストリが管理・提供しているサービスです。.jp のレジストリは JPRS、.com のレジストリは VeriSign Global Registry Services、.shop のレジストリは GMO ドメインレジストリ、というように TLD ごとにレジストリ = Whois

の管理・提供者が異なるため、Whois 情報検索サイトも TLD ごと^{*46}に別々（表 1.3）なのです。

▼表 1.3 TLD ごとの Whois 情報検索サイト

TLD	レジストリ	Whois 情報検索サイト
jp	JPRS（日本レジストリサービス）	http://whois.jprs.jp/
com	VeriSign Global Registry Services	https://www.verisign.com/ja_JP/domain-names/whois/
net	VeriSign Global Registry Services	https://www.verisign.com/ja_JP/domain-names/whois/
shop	GMO ドメインレジストリ	http://whois.nic.shop/

Whois を調べたいとき TLD ごとにサイトがばらばらなのは面倒だな、と思った方は、TLD に関わらずどのドメイン名でも調べられる aguse.jp（図 1.45）^{*47}というサイトがおすすめです。ですが aguse.jp では各レジストリの Whois サイトほど詳しい情報は出てこないですし、更新された Whois 情報もすぐには反映されません。最新の Whois 情報を全部正確に知りたい！ というときは、やはり先ほどのようなレジストリのサイトか、この後第 4 章「dig と whois を叩いて学ぶ DNS」で説明する Whois コマンドを叩いて確認しましょう。



▲図 1.45 どのドメイン名でも Whois が調べられる aguse.jp

^{*46} 正確には TLD ごとではなくレジストリごとに別々です。たとえば VeriSign Global Registry Services は com と net のレジストリなので、VeriSign Global Registry Services のサイトではこの 2 つの TLD の Whois を確認できます

^{*47} アグスジェーピー <https://www.aguse.jp/>

管理がレジストリごとに分かれているため、Whois はフォーマットが統一されておらず、登録しなければいけない氏名や住所といった項目もレジストリによって微妙に異なります。また Whois に登録した情報は誰でも見られるため「あなたが持っているドメイン名はもうすぐ有効期限が切れるからここに\$30 振り込んで！」のような英語の詐欺メールが届くこともあります。

このように Whois には「フォーマットが統一されていない」「詐欺業者に悪用される」などの問題があるため、最近は次世代 WHOIS と呼ばれる RDAP^{*48}への移行が提唱されています。

1.10.3 Whois を正確に登録しなければいけない理由

まとめると Whois とは、そのドメイン名を所有している組織や担当者の氏名、連絡先(住所・電話番号・メールアドレス)、ドメイン名の有効期限などがインターネットで誰でも見られるサービスのことでした。そして前述のとおり Whois 情報は基本的に「正確に登録しなければいけない」もの^{*49}です。

でも個人情報は保護！ 住所や氏名が漏洩したらすぐにお詫びのクオカード！ というこのご時世に、なぜドメイン名の持ち主の情報をインターネットで全公開させているのでしょうか？

それはトラブルが発生したときにインターネットの利用者同士が連絡しあって、自律的にトラブルを解決できるようにするためです。トラブルというのは、たとえば「あなたが持っているドメイン名は我が社が商標登録しているブランドの名前です。30 日以内に譲渡しなければ不正競争防止法に基づいてドメイン名の使用停止を求める裁判を起こします」というようなものです。

Whois 情報がちゃんと登録されていなかったり、あるいは更新されずに古いまになっていたりするとドメイン名の持ち主に連絡ができませんよね？ みんなが「このドメイン名の持ち主に連絡取りたいんだけど、持ち主だれ？」といちいちレジストリに問い合わせをしていたらレジストリはてんてこまいです。なので ICANN は各レジストラに対して「最低年 1 回は登録者に Whois 情報を最新化してもらうように！」という確認を義務付けています。

Whois に情報を登録しなかったり嘘の情報を登録したりすると、場合によってはドメイン名の登録を抹消されてしまいます。ドメイン名を買ったら Whois 情報はきちんと登録してください。

*48 Registration Data Access Protocol の略。詳しくは <https://blog.nic.ad.jp/blog/rdap-intro/> を参照

*49 厳密には TLD によって方針が少しずつ異なります

1.10.4 <トラブル> ドメイン情報認証メールを無視して全サイトが停止

各レジストラは Whois 情報をちゃんと登録してもらうために様々な取り組みを行っています。

たとえばお名前.com の場合、ドメイン名を新たに買ったり Whois 情報を変更したりした場合、Whois に登録されたメールアドレスが正しいものか確認するため、登録したメールアドレス宛てに「ドメイン情報認証」という URL 付きメール（図 1.46）が飛んできます。そして 2 週間以内にメール本文中の認証 URL を踏まなかった場合、ドメイン名が利用停止になってしまいます。

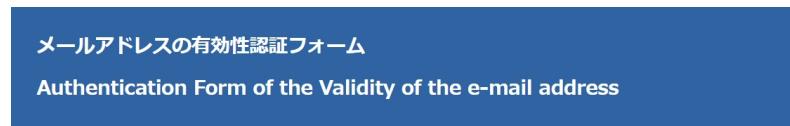


▲図 1.46 正しいメールアドレスか確認するためのドメイン情報認証メール

先ほどお名前.com でドメイン名を買ったあなたのところにも「【重要】[お名前.com] ド

メイン 情報認証のお願い」という件名のメールが飛んできている^{*50}と思います。

本文中の「ドメイン登録者情報のメールアドレスとして情報が正しい場合は、期日までに以下 URL ヘアクセスしてください。」の下に書かれた URL を踏むと「メールアドレスの有効性認証フォーム」という画面（図 1.47）が表示されます。URL を踏んでこの画面が表示されればあとは特に何もしなくて大丈夫です。



メールアドレスの有効性を確認させていただきました。

お申込み時のご登録情報にお間違いはありませんか?
ご登録情報に不備がございましたと、以下のようなケースが懸念されます。今一度ご登録情報をご確認ください。

※各種情報の確認・修正等はご利用のドメイン管理会社へご相談くださいますようお願いいたします。

- 各種ご案内ができず、結果として大切なドメインの失効を招く可能性があります。
- ドメイン紛争などに発展した際に、所有権の正当な主張ができない場合があります。
- ICANN(※)のポリシーのよりドメインのご利用に期限が生じる場合があります。

※ICANN:インターネット上で使用されるドメイン名やIPアドレスといったアドレス資源の割当管理を行う
米国の非常利団体で、ドメイン登録業務を行うレジストラ（登録業者）を公認する権限を持っています。

▲図 1.47 メールアドレスの有効性認証フォーム

繰り返しになりますが、この URL を踏まないまま 2 週間が経つとドメイン名は利用停止になります。しかも今回買った（あるいは Whois 情報を更新した）ドメイン名だけでなく、Whois に同じメールアドレスを登録している全てのドメイン名が同時に利用停止となりますので注意が必要です。お名前.com でドメイン名を買ったらこの 2 つを忘れずに行いましょう。

- Whois 情報をきちんと登録する
- ドメイン情報認証のメールで URL を踏んで正確性確認を行う

^{*50} ドメイン情報認証はメールアドレス 1 つにつき 1 回しか行われません。1 つのお名前アカウントで複数回ドメイン名を購入して、いずれも Whois には example@example.co.jp を登録した場合、正確性確認のメールが送られてくるのは最初の 1 回のみです。またまったく別のお名前アカウントでドメイン名を購入した場合も、Whois に登録したメールアドレスが既に認証を済ませていれば正確性確認のメールは送られません

1.10.5 【ドリル】Whois に登録すべきなのはクライアントの情報？

問題

とある化粧品メーカー A 社のウェブサイトで使うドメイン名を、広告代理店の B 社が代わりに買って、さらにサイトの制作や運用は A 社から Web 制作会社の C 社に委託した場合、Whois 情報には A 社・B 社・C 社のどれを登録すべきでしょうか？

- A. A 社のウェブサイトなんだから A 社を登録すべき
- B. A 社から任されてドメイン名を買ったのは B 社だから B 社を登録すべき
- C. 実際にサイトの管理を任せているのは C 社だから C 社を登録すべき

答え _____

解答

正解は A です。A 社のウェブサイトですので基本的には A 社（クライアント）の情報を記載すべきです。Whois 情報は誰でも見られるため、広告代理店の B 社や Web 制作会社の C 社が請け負っていることを公にしてはいけないような守秘義務のあるケースでうっかり Whois が B 社や C 社になっていると、見る人が見れば「A 社のサイトは B 社や C 社が関わっているんだ」と分かってしまいます。^{*51}

一方で Whois 情報は前述のとおりトラブルがあったときの連絡先として公開されているものなので、実際何かトラブルがあればそこに連絡が来ます。先ほどの 2 週間無視したらドメイン利用停止になる「ドメイン情報認証」のメールも、Whois のメールアドレス宛に送られてきます。また SSL 証明書を購入するときにも Whois の連絡先に対して「このドメイン名の SSL 証明書を発行していいですか？」という確認メールが届きます。あるいはドメイン名の管理を B 社から別の広告代理店 D 社へ移管するようなときにも「D 社に移管していいですか？」という確認メールが届きます。

クライアントである A 社にそうした技術的な問い合わせや連絡が来ても困ってしまう…という場合は社名や担当者名などは A 社にしておいて、メールアドレスはメーリングリストを A 社に作ってもらって B 社や C 社をメンバーに入れるのが得策でしょう。

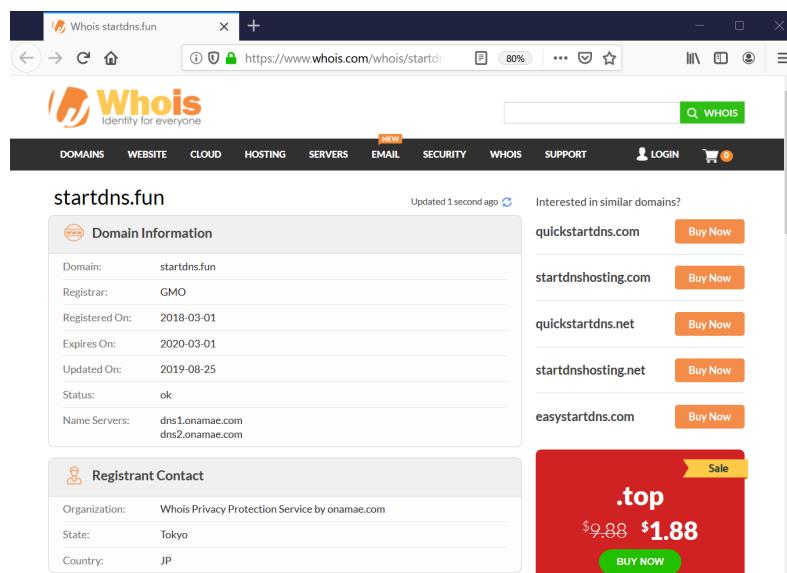
^{*51} 2014 年に NPO 法人が小学 4 年生になりました作った「#どうして解散するんですか？」というサイトも、Whois から NPO 法人が関わっていることが明らかになって炎上していました。http://nlab.itmedia.co.jp/nl/articles/1411/24/news015.html

1.10.6 プライバシーを守るためにWhois情報公開代行

仕事の場合は前述のようにWhoisに正しい情報を登録すべきですが、個人でドメイン名を買ったとき自宅の住所や名前をWhoisに載せるのは抵抗があります。そこで「プライバシーを守るために個人情報をWhoisに載せたくない」という人のためにあるのが、先ほど出てきたお名前.comの「Whois情報公開代行」というオプションサービスです。

これはWhois上で表示される組織名や連絡先を代理でレジストラ（お名前.com）の情報にしてくれるサービスで、一般的にはプロキシサービスやプライバシーサービスと呼ばれています。Whois情報公開代行を使えば、Whoisの所有者の欄には自分の名前の代わりに「Whois Privacy Protection Service by onamae.com」と出るので個人情報を晒さなくて済みます。

実際に筆者のstartdns.funのWhois情報を検索（図1.48）してみると、持ち主の情報が「Whois Privacy Protection Service by onamae.com」になっていることが確認できました。あなたもレジストラのWhois情報検索サイトで、自分が買ったドメイン名のWhoisを確認してみてください。



▲図1.48 startdns.funのWhois情報を見るとお名前.comになっている

ただしjpドメイン名はレジストリであるJPRSの方針として、レジストラによる

Whois 情報公開代行を「登録者名」を除いて許可しています。そのためお名前.com で.jp のドメイン名を買って Whois 情報公開代行を頼んだ場合、登録者名だけは Whois 情報公開代行が適用されずに自分の氏名が表示されてしまいます。

その代わりドメイン名を買った人がレジストラを通して JPRS に「登録者名を非表示にして欲しい」という申請^{*52}を出せば、JPRS 側で Whois の登録者名を「登録者からの申請により非表示」のようになります。しかし残念ながらお名前.com ではドメイン名を買った人から JPRS に対する「登録者名非表示の申請代行」を受け付けていません。そのため結論としてはお名前.com で.jp のドメイン名を買った場合、Whois の登録者名に自分の氏名が出てしまうのは回避できない、という状況になっています。

1.10.7 <トラブル> レジストラが倒産したらドメイン名はどうなる？

レジストラが倒産すると、そこが取り次いでいたドメイン名の管理は他のレジストラへ移転されます。たとえば 2006 年に有限会社愉快堂出版というレジストラが倒産した際は、業務が完全に停止する前に JPRS によって jp ドメイン名の管理が別のレジストラ（株式会社ヒューメイアレジストリ）へと移されました。^{*53*54}

愉快堂出版でドメイン名を登録していた人たちには、移管先のレジストラであるヒューメイアレジストリから「ドメイン名の登録維持を希望されるお客様につきましては、当社での更新手続きが必要となります」という連絡が行われたのですが、このとき貧乏くじを引いた状態になってしまったのは「愉快堂出版でドメイン名を複数年契約していた人たち」でした。

前述のとおり、ドメイン名は複数年の登録・更新が可能です。ですが JPRS のようにレジストリが最長 1 年の登録・更新しか受け付けていない場合、ドメイン名を使う人とレジストラの間で「10 年」の契約を結んで 10 年分の登録料を支払っていても、実際はその裏側でレジストラが毎年 1 年ずつ更新の手続きを行っていた、ということがあります。この場合、もし 2 年目の途中でレジストラが倒産してしまったら、ドメイン名を買った人は「10 年分の登録料を払ったので、この先まだ 8 年は放っておいても大丈夫！」と思っていても、管理を引き継いだ新しいレジストラは残り 8 年分の料金を受け取っておらず、もちろんレジストリも「1 年目、2 年目の登録料しかもらっていない」という状態になります。

*52 <https://jprs.jp/about/dom-rule/whois-concealment/> にある通り、登録者名非表示の申請は必ずレジストラを通して行う必要があります、ドメイン名を買った人から直に JPRS へ申請することはできません

*53 有限会社愉快堂出版を JP ドメイン名管理指定事業者としていた皆様へ 管理指定事業者変更に関するお知らせ <https://jprs.jp/termination/20060814.html>

*54 有限会社愉快堂出版をご利用の JP ドメイン名登録者の皆様へ 管理指定事業者からのご案内について <https://jprs.jp/termination/20061228.html>

愉快堂出版のケースでもこの事象は発生しました。このときヒューメイアレジストリが出したお知らせ^{*55}の抜粋は次のとおりです。

すでに有限会社愉快堂出版にて複数年の契約を行っている場合でも、当社ではその状況を確認できず、有限会社愉快堂出版から当社への更新手続きも行われていないため、当社にてお客様と有限会社愉快堂出版との契約期間を保障することはできません。当社にて保障いたします期間は、先述の Whois にてご確認いただけます期間に加え、今回更新費用をいただきて更新する 1 年分のみとなります。ご了承ください。

10 年分の登録料を先に支払った人たちにしてみれば「そんな！ 一度払ったのにまた払うの？！」と思われるかもしれません、レジストリに払われるべき費用を預かっていた愉快堂出版は倒産してしまい、また移管先のヒューメイアレジストリに責がある訳でもないでの仕方がないことです。

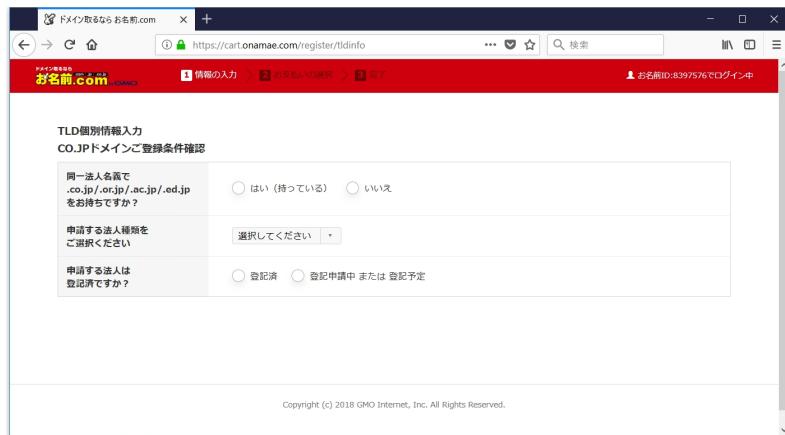
これは国内のレジストラ及びレジストリでの話だったので、まだ日本語で情報が得られただけマシだったかもしれません。海外のレジストラやリセラでドメイン名を購入して、そこが倒産してしまったら、移管先のレジストラとは外国語でやりとりすることになります。さらに Whois にプライバシーサービスを使っていたような場合は、倒産したレジストラから移管先のレジストラに対してドメイン名の契約者情報がきちんと渡されるのか？ も心配です。絶対倒産しないレジストラの見分け方は分かりませんが、ある程度信頼のできるレジストラを選びたいものです。

1.11 co.jp は国内企業しか買えないドメイン名

ところでお名前.com で○○.co.jp を購入しようとすると「CO.JP ドメインご登録条件確認」という確認画面（図 1.49）でドメイン名を購入するための条件が表示されます。

^{*55} <https://web.archive.org/web/20070102174302/http://www.reg.humeia.co.jp/topics/yukaido20061228.html>

1.11 co.jp は国内企業しか買えないドメイン名



▲図 1.49 CO.JP ドメインご登録条件確認

画面で確認されているとおり、次の 2 つの条件を満たさなければ○○.co.jp のドメイン名を購入することはできません。

- ・ 日本国内で法人登記をしている会社、もしくは登記予定・登記申請中の会社であること
- ・ 同一法人組織で既に属性型 JP ドメイン名 (co.jp、or.jp、ac.jp、go.jp) を取得していないこと

さらに後日、申請書と印鑑証明書を郵送しなければなりません。こうした条件があるのは○○.co.jpだけではありません。たとえば○○.ac.jpは日本国内の学校法人しか取得できませんし、○○.go.jpは日本の政府機関しか取得できません。そもそも.jpで終わるドメイン名は、日本国内に住所を持つ組織・個人・団体しか取得できないのです。

すべてのドメイン名でこうした条件があるわけではありません。取得条件が一切ないドメイン名や、あるいは条件はあるが購入時に特に確認されないため実態としては誰でも購入できるドメイン名などもあります。よく見る一般的な.com や.net や.biz などの gTLD は、どこの国の誰でも無条件で購入できます。

1.11.1 【ドリル】co.jp ドメイン名は 2 つ買える？

問題

あなたはとあるソーシャルゲームの開発会社の情シス担当です。コーポレートサイトや会社のメールアドレスでは○○.co.jp というドメイン名を使用しています。この度、社

運を賭けた「△△」というブラウザゲームの開発が決まり、社長から「このゲームでは△△.co.jp というドメイン名を使いたい」と言われました。○○.co.jp を保持したまま、新たに△△.co.jp ドメイン名を買うことは可能でしょうか？

- A. ○○.co.jp を保持したままでは△△.co.jp を買えない
- B. ○○.co.jp を保持したまま△△.co.jp を買える

答え _____

解答

正解は A です。企業向けの.co.jp や大学向けの.ac.jp など、組織の種類ごとに用意されている属性型 JP ドメイン名は基本的に 1 つの組織につき 1 つしか登録できません。ただしこの「1 組織 1 ドメイン名のみ」という原則は以前は絶対でしたが現在は緩和されています。2014 年に「属性型（組織種別型）・地域型 JP ドメイン名登録等に関する規則」が改訂され、企業の合併・組織名変更・事業譲渡などに限って、JPRS に「1 組織 1 ドメイン名制限緩和申請」という申請^{*56}を出せば 1 つの組織で複数の co.jp ドメイン名を持てるようになったのです。^{*57}ですが残念ながら今回のケースは単なる社長の思いつきなのでこの制限緩和には当てはまりません。

1.12 ドメイン名の有効期限が切れるとどうなるのか？

お名前.com でドメイン名を購入した場合、ドメイン名の有効期限が切れたその日からは、本来のサイトの代わりに一時的にレジストラ（お名前.com）の広告ページが表示されることがあります。

またドメイン名の持ち主が更新をせずドメイン名の期限が切れて一定期間が経過すると、そのドメイン名は再び市場で売りに出されて誰でも買える状態になります。そのためサイトをクローズした後、期限切れになったドメイン名を第三者が再登録して、アダルト向けのサイトや詐欺サイト、あるいはウイルスをダウンロードさせるようなサイトを開設

^{*56} <https://jpdirect.jp/organizational/domain-restriction-relaxation.html>

^{*57} この「1 組織 1 ドメイン名制限緩和申請」はレジストラを通して JPRS に申請を行うのですが、レジストラによって申請代行をサービスとして行っているところと、行っていないところがあります。たとえばお名前.com では「1 組織 1 ドメイン名制限緩和申請」は受け付けていないため、どうしても緩和申請がしたい場合は、別のレジストラへ移管した上で申請することになります

することがあります。このように誰かが落とした価値のあるドメイン名を、さっと取得して悪用する行為をドロップキャッチと言います。使用実績のあるドメイン名は一度も使われていないまっさらなドメイン名よりも集客力があるため、その価値を狙ってこうしたドロップキャッチが行われることが多いのです。

世の中には再度売りに出されたドメイン名を確実に買えるよう、常に入荷待ちをしているドロップキャッチ専門のレジストラ^{*58}すら存在しています。

1.12.1 〈トラブル〉ドロップキャッチされたイオンシネマ

2015年6月、イオンシネマ^{*59}のトップページに「【重要なお知らせ（ご注意）】当社HPへアクセスされる場合、旧ドメインはご利用にならないようご注意ください。」というお知らせが出ていました。

どうやら2013年7月にワーナーマイカルシネマズがイオンシネマに統合された後、旧ドメイン名 warnermycal.com を手放したら、まんまと業者にドロップキャッチされてしまったようです。ドメイン名を手に入れた業者が「あなたのコンピュータでウイルスが検出されました」系の詐欺サイトを開設し、これはあくまで推測ですが「ワーナーマイカルシネマズのサイトを見ようとしたエンドユーザ」が詐欺サイトへアクセスして何かしらの被害が出てしまったので、注意喚起としてこのお知らせを出したようです。

Internet Archive Wayback Machine^{*60}で見る限り、2013年7月にサイトをクローズした後も、1年以上はイオンシネマのサイトへリダイレクトをかけていたようです。しかしサイトクローズから1年半以上経った2015年3月頃に旧ドメイン名を手放したところ、直後の4月にドロップキャッチされたものと思われます。

サイトクローズの直後に手放した訳ではないので、一応「ドメイン名をすぐには手放さず新サイトへリダイレクトさせよう」という配慮はしていたようです。ですがサイト統合から1年半以上が経って「もうそろそろいいだろ？」と手放した結果、このトラブルを招いてしまいました。もう存在しないサイトのドメイン名に毎年更新料を払うのは馬鹿らしい気もしますが、ドメイン名を手放した結果、エンドユーザから「サイトにアクセスしたら変なページが表示された！」とクレームが来て、状況を調査しトップページにお知らせを掲示してお詫びをして…といった対応をする人件費を考えたら、あと何年かはドメイン名を持ったままの方が安かったのではないか、と思います。

*58 <http://blogs.itmedia.co.jp/mohno/2014/12/re.html>

*59 <http://www.aeoncinema.com/>

60 [https://web.archive.org/web//http://warnermycal.com](https://web.archive.org/web/*/http://warnermycal.com)

1.12.2 【ドリル】ドメイン名を手放してよい条件は？

問題

あなたはとあるスイムウェアブランドのマーケティング担当者です。昨年夏に作った水着キャンペーンサイトで使っていたドメイン名の有効期限が、1年弱経ってもうすぐ切れてしまいます。キャンペーンはとっくに終わっているので、いつサイトが見えなくなってしまふかも構いません。また今年の夏向けのキャンペーンサイトは別のドメイン名で作りました。ドメイン名は更新しなくていいでしょうか？

- A. 大丈夫！ 更新しません
- B. 更新しないとダメかも知れない…サイトの状況を確認しよう

答え _____

解答

正解は B です。状況を確認しないことには、手放しても大丈夫なのか更新しなければならないのか判断ができません。

前述のとおり、ドメイン名を手放すとドロップキャッチされて詐欺サイトなどを開設されることがあります。このとき自社のコーポレートサイトから、クローズしたキャンペーンサイトへのリンクがうっかり残っていると、「おたくのサイトで『キャンペーンサイトはこちら』というリンクを踏んだらアダルトサイトにつながって架空請求されたんだけど！ どういうこと？！」とエンドユーザからクレームが来ることも考えられます。

キャンペーンが終わった後も、検索結果やニュースサイトの古い記事から流入してきたり、ブックマークで直にサイトを開いたり、というアクセスは少なくありません。そもそもキャンペーンが終わった後は、アクセスしてきた人を自社のコーポレートサイトや翌年のキャンペーンサイトなどにきちんとリダイレクトさせて誘導してあげましょう。

このとき、HTTPステータスコードは「302 Moved Temporarily（一時的な移動）」ではなく「301 Moved Permanently（恒久的に移動）」にしておくことが大切です。ステータスコードを 301 にしていればブラウザ側がリダイレクトをキャッシュしますので、2回目以降は旧サイトにアクセスしようとした時点で新サイトへ即リダイレクトされるようになります。

その状態でドメイン名の有効期限が近付いてきたら、アクセスログなどで「クローズ後

のサイトへアクセスしてきている人がもうほぼいないこと」を確認した上で、さらにコーポレートサイトといった自社の他サイトから、クローズしたサイトへのリンクが残っていないかも確認します。アクセスもほぼないしリンクも残っていなければ、最終的に「ドメイン名がドロップキャッチされても構わないか」を社内で確認して、ドメイン名を手放しても本当に問題ない、と判断できたら手放すようにしましょう。

ドメイン名の更新代はおおよそ数百円から高くても数万円程度です。確認をせずうかつに手放すと、被害をこうむったユーザからの問い合わせで「ドメイン名更新代<人件費」になることも十分あります。どうしても判断がつかなければ更新してしまった方が安全と言えるかもしれません。

1.13 ドメイン名を買ったらサイトが見られるか？

ドメイン名も買ったし Whois 情報も登録しました。でもまだ DNS の設定は何もしていません。この状態だと何が表示されるのか、ブラウザで自分のドメイン名のサイトを見てみましょう。（図 1.50）



▲図 1.50 http://自分のドメイン名を開くとお名前.com の広告が出る

なぜこのページが表示されるのか？ は、第 2 章「DNS の仕組み」で紐解いていきましょう。

第 2 章

DNS の仕組み

ドメイン名を買ってウェブサーバにコンテンツを載せたらそれだけでサイトが見られるでしょうか？

いいえ、ドメイン名とウェブサーバがそれぞれ存在しているだけではサイトは見られません。ドメイン名とウェブサーバを紐づける必要があります。でもサーバはともかくとしてドメイン名は手で触れるようなものではないので、この 2 つをリアルに紐で結ぶことはできません。ドメイン名とサーバを紐づけるには一体どうしたらいいのでしょうか？

そこで出てくるのが DNS です。この章では DNS の仕組みをしっかり学んでいきましょう。

2.1 DNS とは

DNS サーバの DNS は Domain Name System の略で、日本語に直訳すると「ドメイン名の管理システム」といったところです。一口に DNS サーバと言っても、その実態は「ネームサーバ」と「フルリゾルバ」の 2 つに分かれています。異なる働きをする 2 つのサーバがどちらも「DNS サーバ」と呼ばれていることが、DNS を分かりにくくしている一因だと筆者は思います。DNS の仕組みの解説に入る前にきちんとこの 2 つを整理しておきましょう。

2.1.1 ネームサーバ

ネームサーバは「電話帳」のような役割を果たします。

電話帳には名前とそれに紐づく電話番号が書いてありますが、ネームサーバにはドメイン名とそれに紐づく IP アドレスが登録されています。

ネームサーバは「DNS コンテンツサーバ」「権威 DNS サーバ」^{*1}と呼ばれることもありますが、本著では統一してネームサーバと呼びます。

2.1.2 フルリゾルバ

フルリゾルバは「秘書」のような役割を果たします。

フルリゾルバに「このドメイン名に紐づく IP アドレスが知りたいの。調べてきて」と言うと、あちこちのネームサーバに聞きまわって IP アドレスを調べてきて教えてくれます。しかも一度調べると一定期間はそのドメイン名と IP アドレスの紐づけを記憶（キャッシュ）するため、もう 1 回同じことを聞くと今度はすぐに教えてくれます。

フルリゾルバは「DNS キャッシュサーバ」「フルサービスリゾルバ」と呼ばれることがあります、本著では統一してフルリゾルバと呼びます。

あなたがブラウザでウェブサイトを見るとときは必ずこのフルリゾルバにドメイン名の名前解決を頼んでいます。「でもそんなもの使う設定をした記憶ないけど…」と思われるかも知れませんが、会社のオフィスなら情シスが、家庭なら契約している ISP^{*2}がフルリゾルバを用意していて、自動で「このフルリゾルバを使ってね」と割り当てられているので意識していないだけです。

^{*1} 権威 DNS サーバという日本語は「Authoritative Server」（権威のあるサーバ）から来ているようです。

DNS で用いられる用語の定義は RFC7719 を参照。<https://tools.ietf.org/html/rfc7719>

^{*2} インターネットサービスプロバイダの略。インターネットへの接続サービスを提供する電気通信事業者のこと

情シスや ISP がそれぞれのネットワーク内で提供しているフルリゾルバの他に、Google Public DNS^{*3} の 8.8.8.8 や Cloudflare の 1.1.1.1^{*4} のようにだれでも無料で使えるオープンリゾルバというものがあります。2013 年 8 月に OCN^{*5} のフルリゾルバが死んで OCN ユーザがインターネットに接続できなくなる^{*6}障害が発生しました。そのとき「DNS サーバの設定を 8.8.8.8 にすれば直るよ」という情報が Twitter で出回りました（図 2.1）。

▲図 2.1 DNS サーバを 8.8.8.8 にするとつながるというツイート

^{*3} <https://developers.google.com/speed/public-dns/>

^{*4} <https://1.1.1.1/ja-jp/>

^{*5} NTT コミュニケーションズが運営する日本最大級のインターネットサービスプロバイダ

^{*6} ドメイン名から IP アドレスを引く名前解決ができないことでウェブサーバの IP が分からず、サイトが見られなくなったりゲームやアプリなどのサービスが使えなくなったりした、ということです

これは使用するフルリゾルバを故障中のOCNのものからGoogle Public DNSに変更することで名前解決が出来るようになってサイトも見られるようになった、ということです。

フルリゾルバはもちろんサーバでも使っています。Linuxサーバなら/etc/resolv.confというファイルを見てみましょう。resolv.confのnameserverという項目で指定されているのがそのサーバのフルリゾルバです。

```
$ cat /etc/resolv.conf
options timeout:2 attempts:5
nameserver 172.31.0.2
```

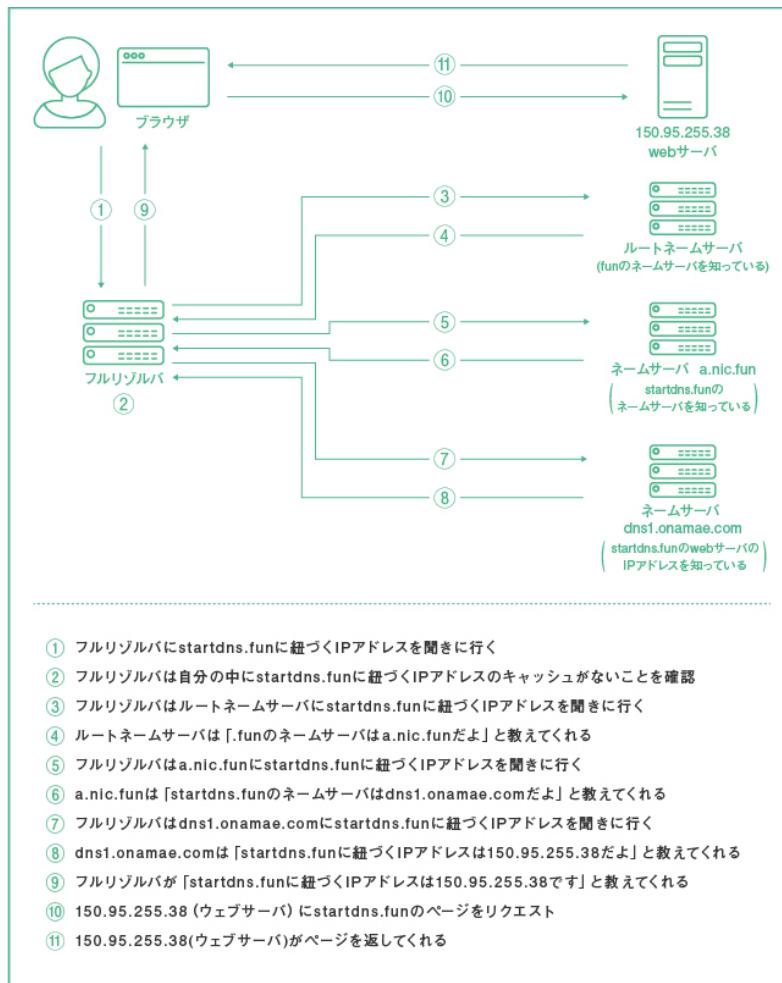
2.2 お名前.comのページが表示されるまで

第1章「ドメイン名とWhois」の最後で、自分が買ったドメイン名をブラウザで開いてみたところ、お名前.comの「このドメインは、お名前.comで取得されています。」(図2.2)というページが表示されました。



▲図2.2 http://自分のドメイン名を開くとお名前.comの広告が出る

ブラウザでhttp://自分のドメイン名を開いたときに、このページが表示されるまでの流れは次(図2.3)のようになっていました。

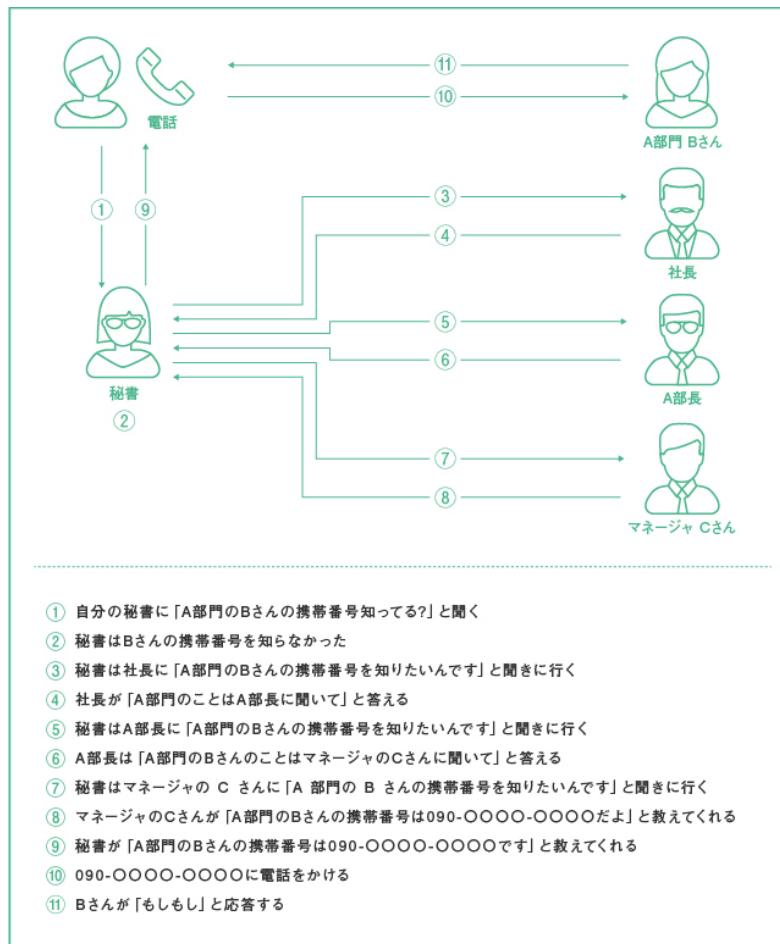


▲図 2.3 ページが表示されるまでのフルリゾルバとネームサーバの仕組み

ブラウザのアドレスバーに自分のドメイン名を入れて Enter を押してからページが表示されるまでの一瞬の間に、なんとこれだけの名前解決が行われていたのです。

2.3 ゾーンと委任

このように DNS は「1台が何もかも知っている」という集中管理ではなく、「いくつかのネームサーバに聞けば答えにたどり着く」という管理権限の分散された仕組みになっています。先ほどの名前解決を会社に例えると次のようになります。



▲図 2.4 Bさんに電話をかけるまでの流れ

秘書がフルリゾルバで、社長や A 部長やマネージャの C さんがネームサーバ、B さんがウェブサーバに当てはまります。

社長が A 部門のことは A 部長に任せていたように、ルートネームサーバは fun のことは a.nic.fun に任せていきました。このとき「A 部門」や「fun」、「B さん」や「startdns.fun」のような範囲をゾーンと呼びます。

A 部長は自分が任されている A 部門の中で、B さんについては C マネージャに管理を任せていきました。同様に a.nic.fun というネームサーバは、自分が任されている fun というゾーンの中で、fun のサブドメインにあたる startdns.fun というゾーンについては dns1.onamae.com に任せていきました。

このように自身が任せられているゾーンを分割して、その一部のゾーンを他のネームサーバに任せることを委任と呼びます。

つまり社長が A 部門というゾーンを A 部長に委任していたように、ルートネームサーバは fun というゾーンを a.nic.fun に委任していた、ということです。ゾーンを委任されているネームサーバは、そのドメインについて権威を持つので、サブドメインを作ったり、任されたゾーンをさらに分割して他のネームサーバに委任したりできます。

2.4 リソースレコード

ネームサーバのお腹の中にある電話帳は管理しやすいように「startdns.fun の電話帳」「example.com の電話帳」のようにドメインごとに分かれています。この一冊一冊の電話帳が管理している範囲を前述のとおりゾーンと呼びます。

そしてこのゾーンの中にある「ドメイン名と IP アドレスの紐づけ」ひとつひとつのことをリソースレコードと呼びます。たとえば startdns.fun のゾーンの中には「startdns.fun とそれに紐づく IP アドレス」や「www.startdns.fun とそれに紐づく IP アドレス」、「staging.startdns.fun とそれに紐づく IP アドレス」のようにたくさんのリソースレコードを書くことができます。

先ほどの会社の例で言うと、A 部長は A 部門というゾーンを管理していて、マネージャの C さんは B さんというゾーンを管理しています。そして C さんが管理する B さんというゾーンの中には「B さんの携帯番号は 090-〇〇〇〇-〇〇〇〇」や「B さんの自宅番号は 03-〇〇〇〇-〇〇〇〇」といったリソースレコードがあるのです。

リソースレコードには次（表 2.1）のように A レコードや MX レコードといった種類があり、それぞれ書き方も決められています。

▼表 2.1 リソースレコードの種類

リソースレコードのタイプ	値の意味
A レコード	ドメイン名に紐づく IP アドレス（例：ウェブサーバ）
NS レコード	ドメイン名のゾーンを管理するネームサーバ
MX レコード	ドメイン名に紐づくメール受信サーバ
TXT (SPF)	このドメイン名のメール送信元サーバ
SOA	ドメイン名のゾーンの管理情報
CNAME	このドメイン名の別名でリソースレコードの参照先

それぞれのリソースレコードをどういうときに使うのか？ については第 3 章「AWS のネームサーバ（Route53）を使ってみよう」や第 4 章「dig と whois を叩いて学ぶ DNS」で具体例を見て、手を動かしながら確認ていきましょう。

2.5 hosts ファイル

ところで DNS という管理権限が分散された仕組みが生まれる以前は、hosts ファイル^{*7}と呼ばれるテキストファイルに「名前と IP アドレスの対応」がすべて詰め込まれて一元管理されていました。hosts ファイルは現在も使われており、実は名前解決は「先ずは hosts ファイルで探す→ hosts ファイルになければフルリゾルバに問い合わせ」という順番で行われることが多い^{*8}です。

それでは実際に hosts ファイルの利用を試してみましょう。

2.5.1 Windows で hosts を使ってみよう

「メモ帳」で検索したらアイコンを右クリックして「管理者として実行」をクリック（図 2.5）します。



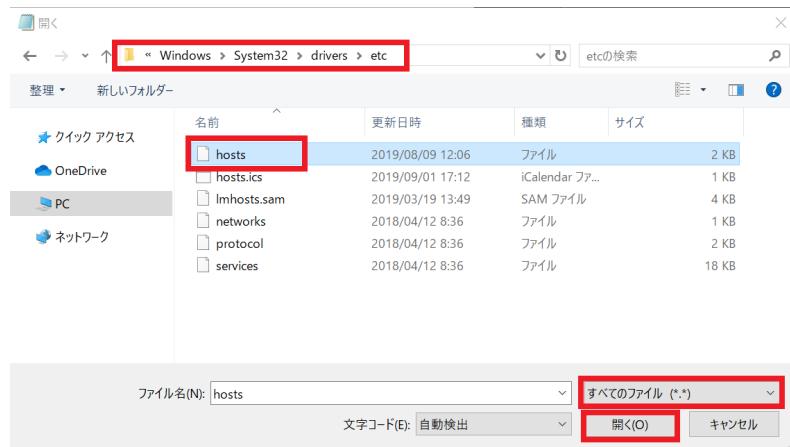
▲図 2.5 メモ帳を管理者として実行

メモ帳が起動したら「ファイル>開く」で C:\Windows\System32\drivers\etc というフォルダの中にある hosts というファイルを開いてください。（図 2.6）フォルダ内にファイルが見当たらない場合は、右下の「テキスト文書 (*.txt)」を「すべてのファイル (*.*)」にします。

^{*7} ホストファイルと読みます。ホスト名（サーバの名前）と IP アドレスの対応がたくさん記述されているので host の複数形で hosts です

^{*8} Linux サーバの場合、名前解決の順番は/etc/nsswitch.conf の host 行で hosts: files dns のように指定されています。files は hosts ファイルのこと、dns は DNS に問い合わせることを表しています

2.5 hosts ファイル

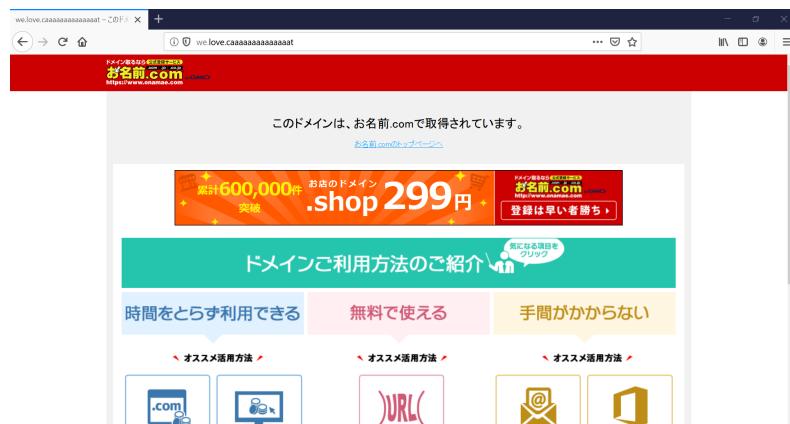


▲図 2.6 メモ帳で hosts ファイルを開く

メモ帳で hosts ファイルを開いたら、いちばん下に次の 1 行を書き足してみましょう。

150.95.255.38 we.love.caaaaaaaaaaaaat

猫への愛が爆発していますが、もちろんこんなドメイン名は実在しません。ctrl+s で hosts ファイルの変更を保存したら、ブラウザで <http://we.love.caaaaaaaaaaaaat/> を開いてみましょう。(図 2.7) 「このドメインは、お名前.com で取得されています。」というページが表示されました。



▲図 2.7 存在しないドメイン名でページが表示された

これは `we.love.caaaaaaaaaaaaat` という架空のドメイン名と、お名前.com のウェブサーバ（150.95.255.38）の IP アドレスを hosts ファイルで紐付けたことで、ページが表示されるようになったという仕組みです。^{*9}もちろんこの URL を叩いても、hosts ファイルに 150.95.255.38 `we.love.caaaaaaaaaaaaat` を書き足したパソコンでしかページは表示されません。

このように hosts ファイルに書き込むことで、その環境でだけドメイン名に紐付く IP アドレスを変えられるので、サイトリニューアルでウェブサーバを引っ越すような場合に「hosts ファイルにこれを書くとブラウザで新しいサイトが確認できる」のような使い方ができます。

2.5.2 Mac や Linux で hosts を使ってみよう

Mac や Linux で同じことを試したかったら、`/etc/hosts` というファイルのいちばん下に次の 1 行を書き足してみましょう。

```
150.95.255.38 we.love.caaaaaaaaaaaaat
```

その上で、ブラウザで `http://we.love.caaaaaaaaaaaaat/` を開くか、ターミナルで次の curl コマンドを叩くと動作確認ができます。^{*10}

```
$ curl http://we.love.caaaaaaaaaaaaat/
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Style-Type" content="text/css" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<title>we.love.caaaaaaaaaaaaat - このドメインはお名前 .com で取得されています。
</title>
<link href="http://error.gmo.jp/contents/setstyle.css"
      rel="stylesheet" type="text/css" />
</head>
(後略)
```

^{*9} \$ dig we.love.caaaaaaaaaaaaat @dns1.onamae.com +short +nored で 150.95.255.38 を返してくれるネームサーバすごいなという気持ちです。お名前.com さん、勝手に実験台にしてすみません

^{*10} これはお名前.com のウェブサーバ（150.95.255.38）がどんなホスト名でリクエストが来てもレスポンスを返すようになっていたため、勝手な名前でリクエストしてもページが表示されています。hosts ファイルを設定しただけでは、ウェブサーバ側の設定次第でページが返ってこない可能性も十分あります

第3章

AWS のネームサーバ (Route53) を使ってみよう

第1章「ドメイン名とWhois」で自分のドメイン名を買って、第2章「DNSの仕組み」ではDNSの仕組みを学びました。この章ではAWSのRoute53（ルートファイフティースリー）というDNSサービスで自分のドメイン名のゾーンやリソースレコードを作ってみましょう。

3.1 AWS とは

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

アマゾンがやっているクラウドと言われても、そもそもクラウドってなんだろう？ という方もいますよね。クラウドとは「ブラウザ上でスペックを選んでぽちっとするだけで誰でもすぐにウェブサーバや DB サーバを立てたりネームサーバを使ったりできるサービス」^{*1}のことです。AWS では基本的に 1 秒単位の従量課金なので、たとえばサーバを立てて 3 分使ったら 3 分ぶんのお金しかかりません。

今回は AWS の中でも Route53（ルートファイフティースリー）という DNS のサービスを使用します。クラウドといえば Google の Google Cloud Platform や Microsoft の Azure (アジュール)、最近シェアを伸ばしてきた Alibaba Cloud、さくらインターネットのクラウドや GMO クラウドなど AWS 以外にもたくさんあります。ですが現状のシェアトップはダントツで AWS^{*2}です。AWS なら何か困って検索したときに出でてくる情報も多いので、今回は AWS の Route53 で自分のドメイン名のゾーンを作成します。

3.2 AWS アカウント作成

先ずは AWS 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
- 通話可能な携帯電話（電話番号認証で使用するため）

なお AWS を初めて使用する場合、利用料が 1 年分無料^{*3}となります。

「AWS 無料」(図 3.1)で検索して上から 2 つめの「AWS クラウド無料利用枠 | AWS」^{*4}をクリックします。

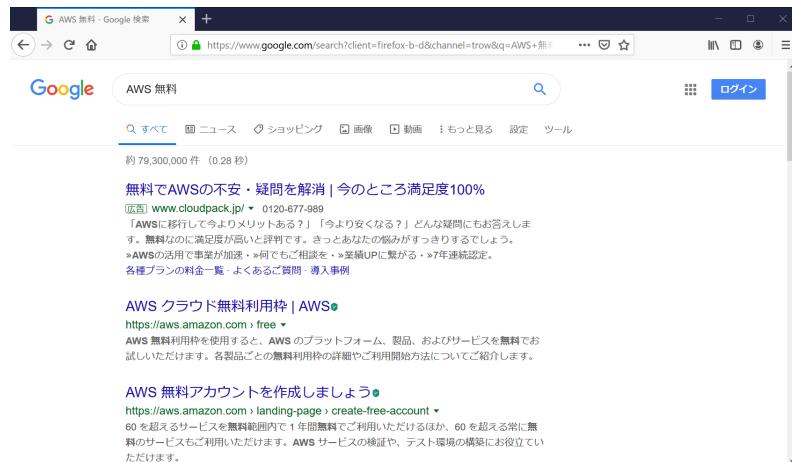
^{*1} AWS やクラウド、サーバについては本著の続編「AWS をはじめよう」で詳しく解説していますので、ここでは正確さよりも分かりやすさを優先したゆるい説明にしています

^{*2} 2018 年時点、AWS はシェア全体の 3 割以上を占めクラウド界のトップを独走中です

^{*3} 無料利用枠の範囲が決まっており、何をどれだけ使っても無料という訳ではありませんので注意してください。たとえばこの後利用する Route53 という DNS のサービスは、1 つのドメイン名につき毎月 50 セントかかります。詳細は <https://aws.amazon.com/jp/free/> を確認してください

^{*4} <https://aws.amazon.com/jp/free/>

3.2 AWS アカウント作成



▲図 3.1 「AWS 無料」で検索

「AWS クラウド無料利用枠」(図 3.2) のページを開いたら、「まずは無料で始める」をクリックします。



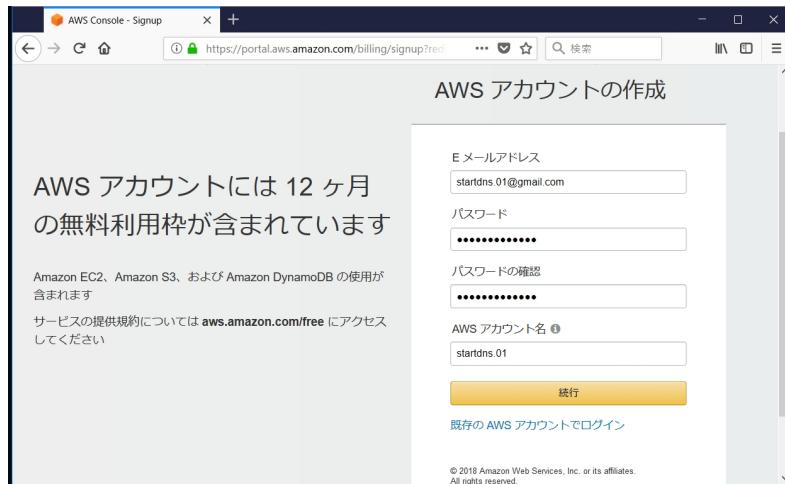
▲図 3.2 「まずは無料で始める」をクリック

AWS アカウントの作成画面(図 3.3)で次の 4つを入力したら、「続行」をクリックします。後で分からなくならないように、何を登録したのかはメモしておいてください。

- E メールアドレス
- パスワード

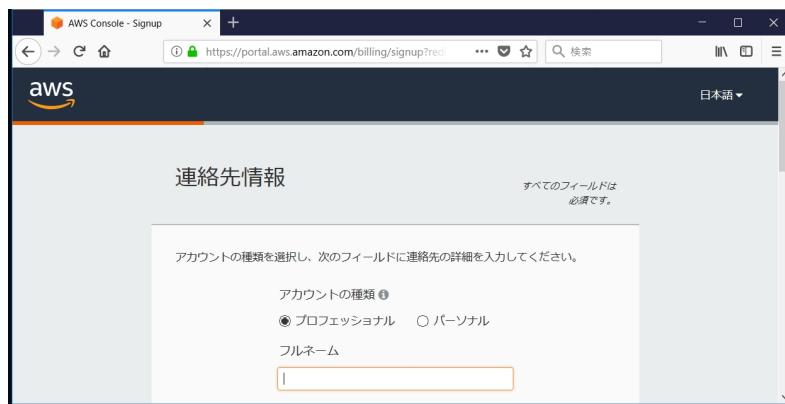
第3章 AWS のネームサーバ (Route53) を使ってみよう

- パスワードの確認
- AWS アカウント名



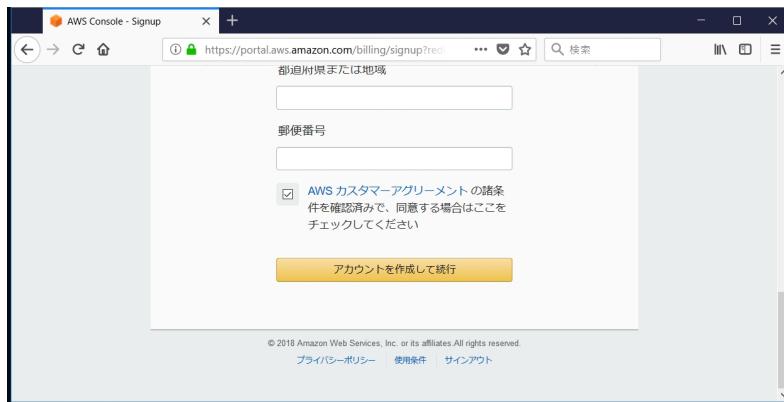
▲図 3.3 メールアドレスとパスワードと AWS アカウント名を記入して「続行」を押す

連絡先情報（図 3.4・図 3.5）はすべて英語表記で登録します。今回は仕事ではなく個人での利用ですのでアカウントの種類は「パーソナル」を選択してください。住所と電話番号を記入したら「AWS カスタマーアグリーメントの同意」にチェックを入れて「アカウントを作成して続行」を押します。



▲図 3.4 連絡先情報は英語で登録

3.2 AWS アカウント作成

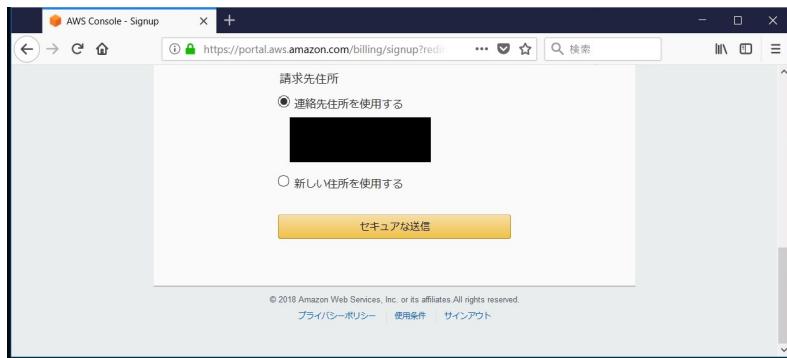


▲図 3.5 連絡先を入力したら「アカウントを作成して続行」を押す

前述のとおり、AWS を初めて使用する場合は利用料が 1 年分無料なのですが、クレジットカードは登録しておく必要があります。支払情報（図 3.6・図 3.7）にクレジットカード情報を記入して「連絡先住所を使用する」を選択したら「セキュアな送信」を押します。

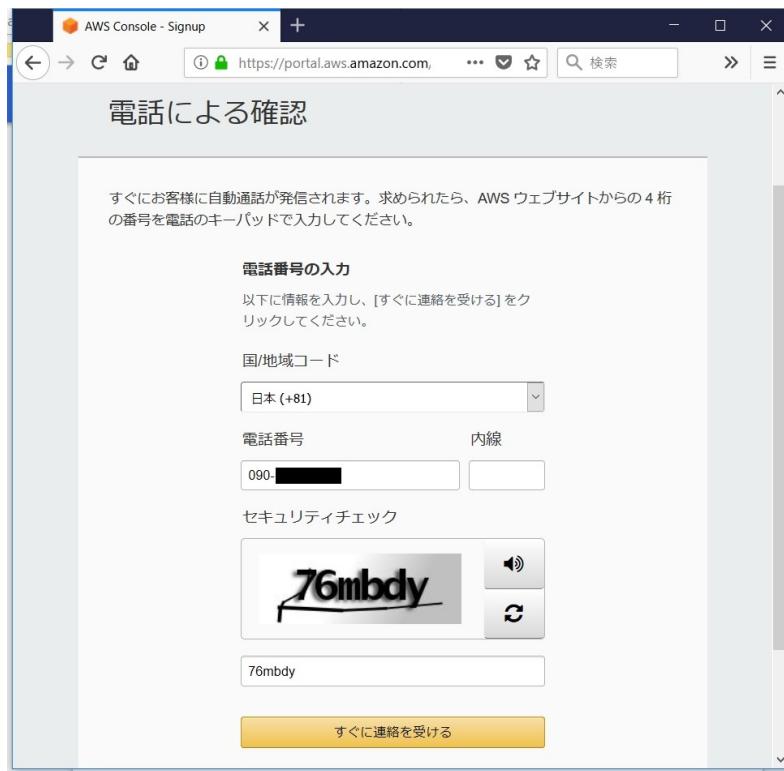


▲図 3.6 クレジットカード情報を記入



▲図 3.7 請求先住所を選択して「セキュアな送信」を押す

電話による確認（図 3.8）で電話番号（例：090-1234-5678）とセキュリティチェックを入力したら、携帯電話を手元に用意した状態で「すぐに連絡を受ける」を押します。



▲図 3.8 電話番号とセキュリティチェックを入力したら「すぐに連絡を受ける」を押す

このとき「エラー：お支払情報に問題があります」（図 3.9）と表示されて、何度試しても電話確認に進めない場合があります。AWS では初回登録時に「1 ドル認証」と呼ばれる認証方法でクレジットカードが決済可能かをチェックしているのですが、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、AWS のチャットサポートで問い合わせてみてください。

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.9 クレジットカードに起因するエラー

「すぐに連絡を受ける」を押すとすぐに通知不可能で電話（日本語の自動音声）がかかってくるので、パソコン側で表示されている 4 ケタの番号（図 3.10）を電話でプッシュしてください。



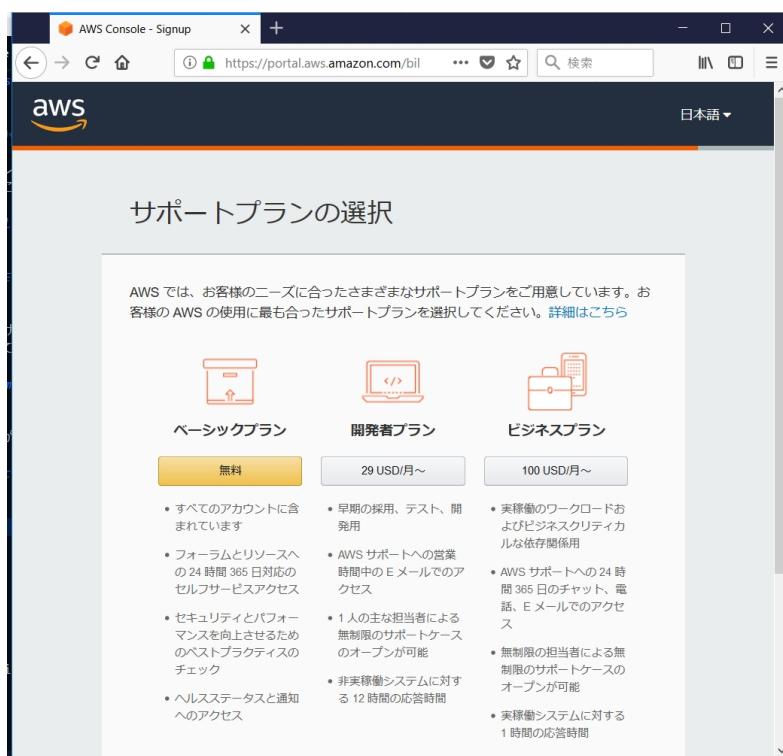
▲図 3.10 パソコンで表示された 4 ケタの番号を電話でプッシュ

確認完了して電話が切れたら、パソコン側で「本人確認が終了しました」（図 3.11）と表示されるので「続行」を押します。



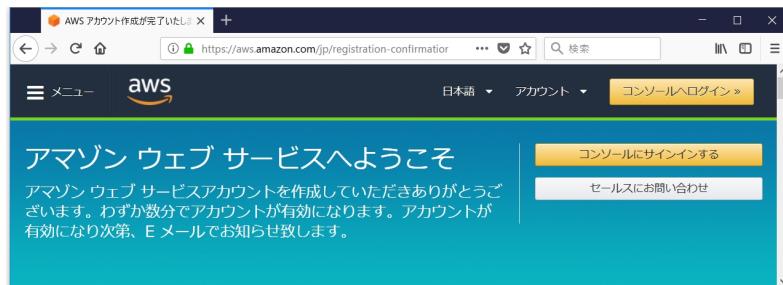
▲図 3.11 「本人確認が終了しました」と表示されたら「続行」を押す

本人確認が完了したらサポートプランの選択に進みます。サポートプランの選択（図 3.12）はベーシックプランがよいので、いちばん左の「無料」を押します。



▲図 3.12 サポートは無料のベーシックプランを選択

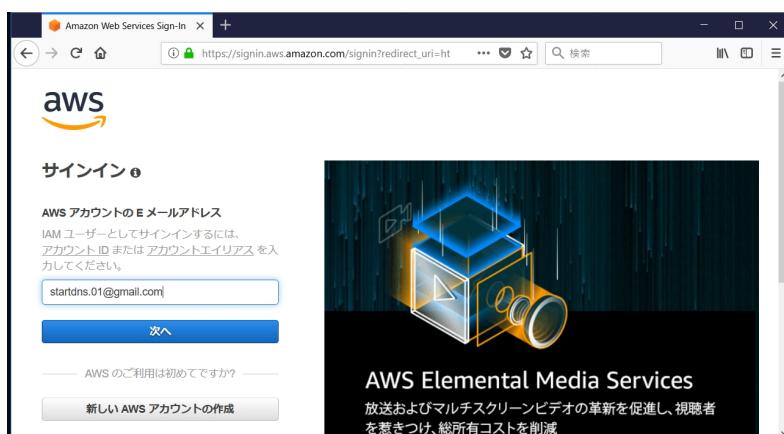
「アマゾン ウェブ サービスへようこそ」(図 3.13) と表示されたら AWS のアカウント作成は完了です。おめでとうございます！「コンソールにサインインする」ボタンを押して次のステップへ進みましょう。



▲図 3.13 AWS のアカウント作成完了

3.2.1 AWS のマネジメントコンソールにログイン

「コンソールにサインインする」を押す^{*5}と、マネジメントコンソールへのログイン画面が表示されます。(図 3.14) 先ほど登録した AWS アカウントの E メールアドレスを入力して「次へ」を押してください。

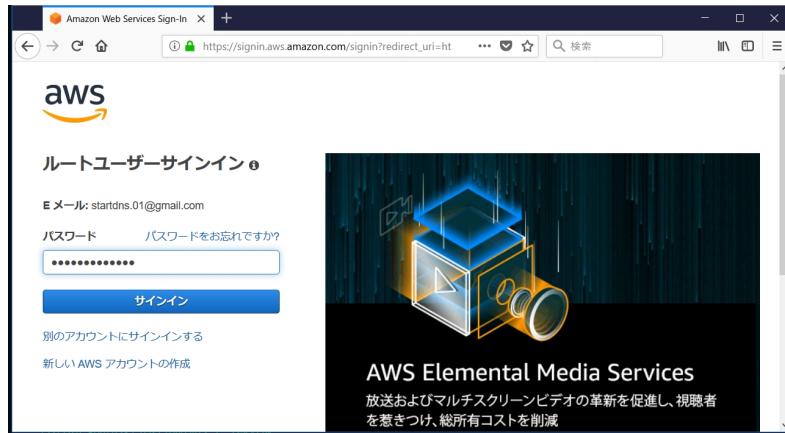


▲図 3.14 AWS のアカウントの E メールアドレスを入力

^{*5} 後でまたマネジメントコンソールにログインしたくなったら <https://aws.amazon.com/> で「コンソールへサインイン」を押せばログイン画面が表示されます

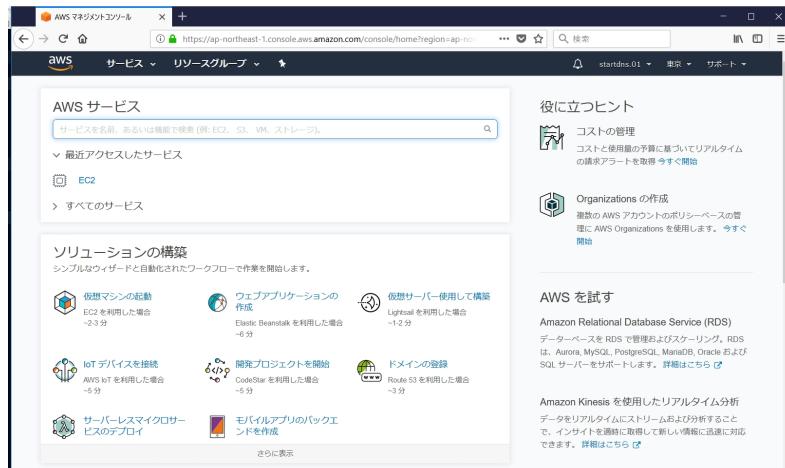
3.2 AWS アカウント作成

続いてルートユーザーサインインの画面（図 3.15）でパスワードを入力したら「サインイン」を押します。



▲図 3.15 パスワードを入力してサインイン

無事にログインできたら、マネジメントコンソールという AWS の管理画面（図 3.16）が表示されます。



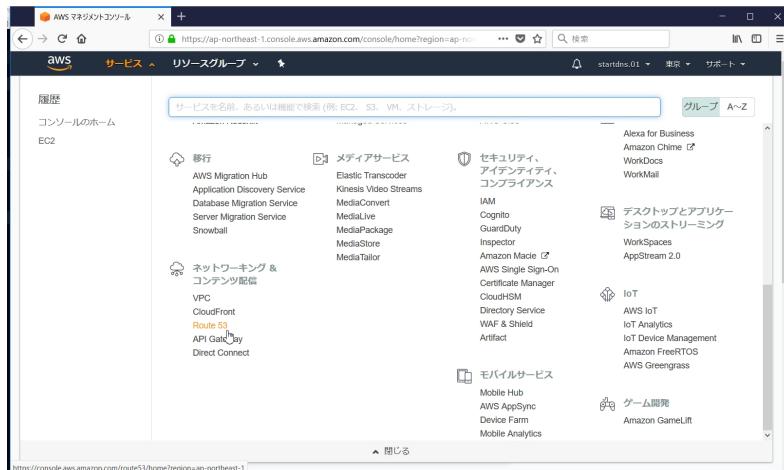
▲図 3.16 マネジメントコンソール（AWS の管理画面）

3.3 ドメイン名のネームサーバを Route53 に変更

それではお名前.com で買ったドメイン名のネームサーバを Route53 に変更する作業を行いましょう。まずは Route53 で自分のドメイン名のゾーンを作成します。筆者は startdns.fun というドメイン名を使いますので、あなたも自分のドメイン名のゾーンを作成してみてください。

3.3.1 Route53 でホストゾーンを作成

AWS のマネジメントコンソールで、左上の「サービス」を押すと AWS のサービス一覧（図 3.17）が表示されます。^{*6} 「ネットワーキング & コンテンツ配信」というグループにある Route53 を選択してください。

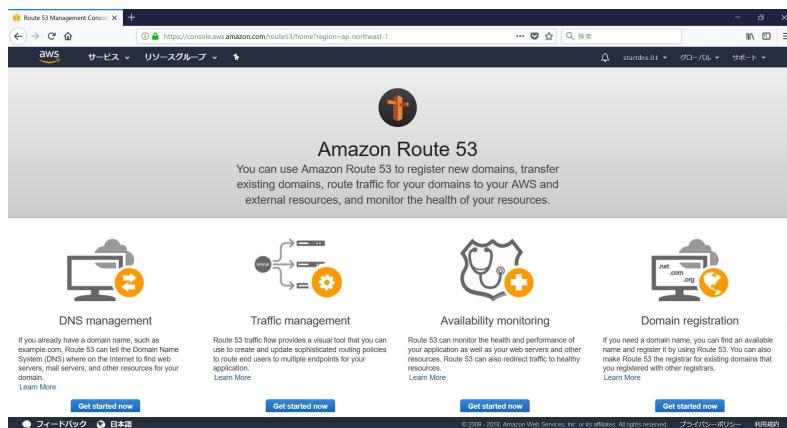


▲図 3.17 サービス一覧から Route53 を選択

Route53 を押すといくつかメニューが表示される（図 3.18）ので、いちばん左側にある DNS management の「Get started now」を押してください。

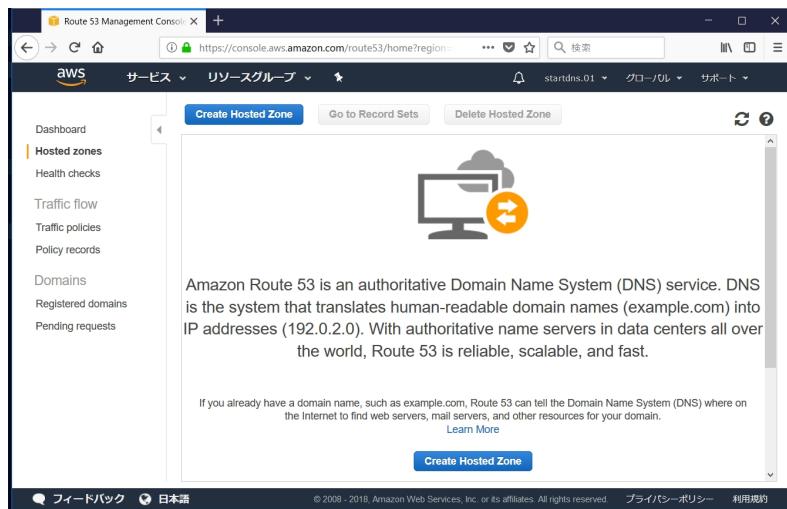
^{*6} クラスマソッドの Developers.IO で 2019 年時点の AWS 全サービスがまとめられているので、AWS 全体をざっと俯瞰したい方はそちらがお勧めです。 <https://dev.classmethod.jp/cloud/aws/aws-summary-2019-1/>

3.3 ドメイン名のネームサーバを Route53 に変更



▲図 3.18 DNS management の「Get started now」を押す

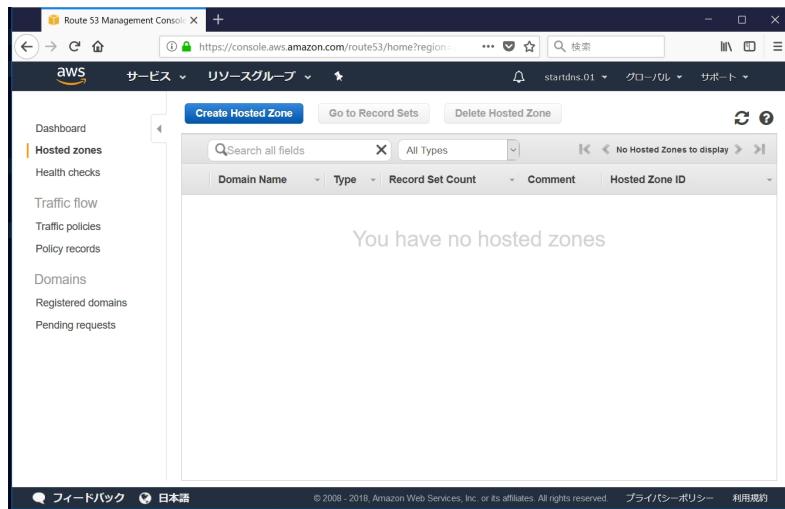
Route53 というネームサーバの中に自分のドメイン名のゾーンを作りたいので、ホストゾーンの画面（図 3.19）が表示されたら「Create Hosted Zone」を押してください。



▲図 3.19 「Create Hosted Zone」を押す

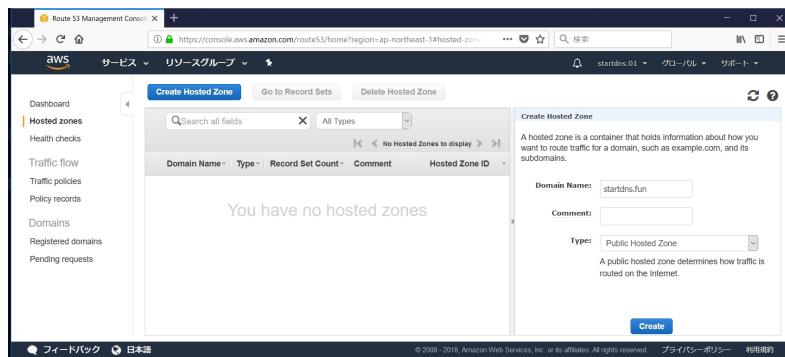
まだゾーンがひとつもないため「You have no hosted zones」と表示（図 3.20）されます。再び「Create Hosted Zone」ボタンを押してください。

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.20 「You have no hosted zones」と表示されたら再び「Create Hosted Zone」を押す

ホストゾーンの作成画面（図 3.21）を開いたら Domain Name のところに先ほどお名前.com で買った自分のドメイン名を書きます。筆者は startdns.fun というドメイン名を買ったので、Domain Name のところに startdns.fun と書きました。



▲図 3.21 お名前.com で買った自分のドメイン名を書く

Type は Public Hosted Zone を選択（表 3.1）します。「Public Hosted Zone」にしておけば外から「このドメイン名に紐づいてる IP アドレスはなに？」と聞かれたときに Route53 のネームサーバが「IP アドレスは○○だよ」と返事をします。ここを「Private

3.3 ドメイン名のネームサーバを Route53 に変更

Hosted Zone for Amazon VPC」にしてしまうと AWS 内での名前解決しか出来なくなり、外から聞かれても何も答えてくれなくなってしまいますので、必ず「Public Hosted Zone」にしておいてください。入力完了したら「Create」を押します。

▼表 3.1 ホストゾーン作成時の設定

項目	入力するもの
Domain Name	お名前.com で買った自分のドメイン名
Comment	何も入力しない
Type	Public Hosted Zone

これで Route53 というネームサーバの中に自分のドメイン名のゾーンが出来て、ゾーンの中にドメイン名のネームサーバを示す NS レコードと管理情報を示す SOA レコードというリソースレコードも出来ました。(図 3.22)

The screenshot shows the AWS Route 53 Management Console interface. On the left, there's a sidebar with options like Dashboard, Hosted zones, Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, and Pending requests. The main area is titled 'Hosted zones' and shows a list of zones. One zone, 'standins.fun', is selected. To the right of the zone list, there's a 'Create Record Set' button and several other buttons like Import Zone File, Delete Record Set, and Test Record Set. The 'Edit Record Set' panel is open for the 'standins.fun' zone. It shows two entries: an SOA record and an NS record. The SOA record is for 'ns-943.awsdns-53.net' with a TTL of 900 seconds. The NS record lists four name servers: 'ns-943.awsdns-06.co.uk', 'ns-1005.awsdns-06.co.uk', 'ns-1072.awsdns-06.org', and 'ns-1177.awsdns-22.com'. The NS record has a TTL of 1728K seconds. The panel also includes fields for 'Aliases' (checkboxes for 'None' or 'Yes'), 'TTL (Seconds)', and 'Values' (a dropdown menu showing the four name servers). A note at the bottom says 'The domain name of a name server must be the first name server in the list. Name servers are separated by commas.' Below the panel are buttons for 'Save Record Set' and 'Cancel'.

▲図 3.22 自分のドメイン名のゾーンとその中のリソースレコードが出来た

3.3.2 自分のドメイン名のネームサーバが何か確認

ではこれで自分のドメイン名のネームサーバが Route53 になったのか、ちょっと確認してみましょう。

ブラウザの別タブで「dig」を検索（図 3.23）して、一番上の「nslookup(dig) テスト【DNS サーバ接続確認】」^{*7}というページを開きます。

*7 <https://www.cman.jp/network/support/nslookup.html>

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.23 「dig」で検索

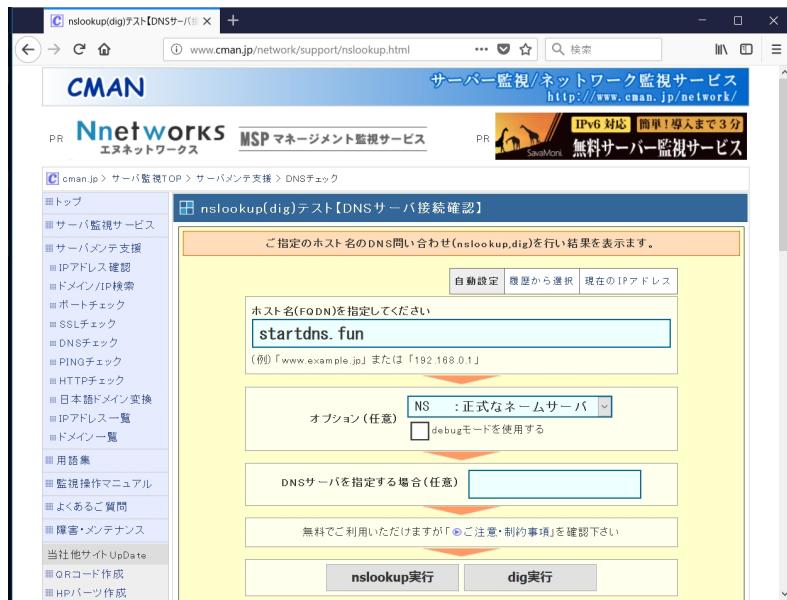
このサイトでは自分のドメイン名のネームサーバが何になっているのか、今の状態を確認できます。では「nslookup(dig) テスト【DNS サーバ接続確認】」のページ（図 3.24）で「ホスト名 (FQDN) を指定してください」という欄に自分のドメイン名を入力（表 3.2）して、オプションで「NS：正式なネームサーバ^{*8}」を選択したら「dig 実行」を押してください。

▼表 3.2 ネームサーバを調べる際の入力項目

項目	入力するもの
ホスト名 (FQDN) を指定してください	お名前.com で買った自分のドメイン名
オプション (任意)	NS：正式なネームサーバ

^{*8} NS レコードはドメイン名のゾーンを管理するネームサーバを示すリソースレコードです

3.3 ドメイン名のネームサーバを Route53 に変更



▲図 3.24 自分のドメイン名を入力してネームサーバを調べる

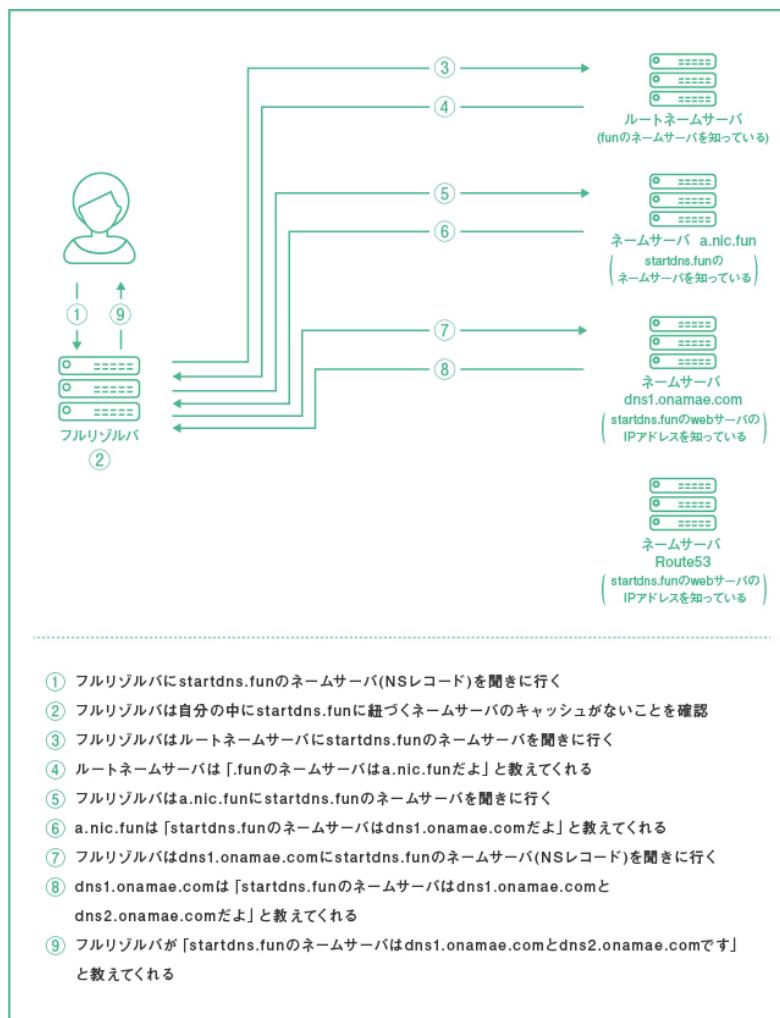
ちょっと下にスクロールして確認結果の ANSWER SECTION というところを見てください。次のように表示されましたか？

```
; ANSWER SECTION:  
startdns.fun.      300 IN NS    dns1.onamae.com.  
startdns.fun.      300 IN NS    dns2.onamae.com.
```

ANSWER SECTION は名前のとおり「startdns.fun の NS レコードは？」という質問に対する答えです。NS レコードはそのドメイン名のゾーンを管理するネームサーバを示すリソースレコードなので「startdns.fun というドメイン名のネームサーバは、お名前.com のネームサーバ (dns1.onamae.com と dns2.onamae.com だよ」という答えが返ってきた、ということです。

先ほど AWS の Route53 というネームサーバの中に自分のドメイン名のゾーンを作ったのに、なぜかネームサーバはまだお名前.com のままになっています。これはどういうことなのでしょう？

これは startdns.fun というゾーンがお名前.com のネームサーバに委任されていることで、次のような流れになっているからです。



▲図 3.25 ゾーンがお名前.com のネームサーバに委任されている

このようにお名前.com のネームサーバと Route53 のネームサーバ、どちらにも startdns.fun のゾーンがあるのですが、上位ネームサーバの a.nic.fun が「startdns.fun については dns1.onamae.com に委任する」という設定なので、お名前.com のネームサーバが startdns.fun の権威を持った状態になっています。そのため startdns.fun の NS レコードを問い合わせたときに誰も Route53 のネームサーバには聞きに来ないです。

そもそもお名前.com でドメイン名を買うと、デフォルトの設定でネームサーバには次の 2 つが設定されています。

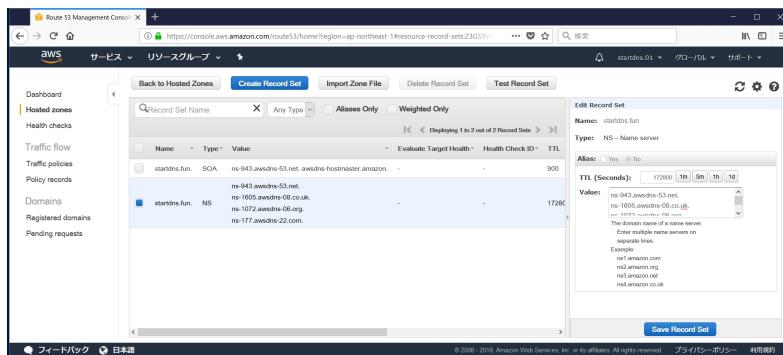
3.3 ドメイン名のネームサーバを Route53 に変更

- dns1.onamae.com.
- dns2.onamae.com.

そのためネームサーバを Route53 に変更したければ、Route53 で自分のドメインのゾーンを用意するだけでなく、さらにお名前.com の管理画面で「このドメイン名はお名前.com のネームサーバを使う」という設定を「このドメイン名は Route53 のネームサーバを使う」という設定に書き換えなければいけなかったのです。

先ほどのマネジメントコンソールに戻って（図 3.26）NS レコードの Value を見てみましょう。次のような 4 つのネームサーバが表示されていると思います。^{*9} この 4 つは後ほど必要になりますのでパソコンのメモ帳に書き留めておいてください。

- ns-943.awsdns-53.net.
- ns-1605.awsdns-08.co.uk.
- ns-1072.awsdns-06.org.
- ns-177.awsdns-22.com.



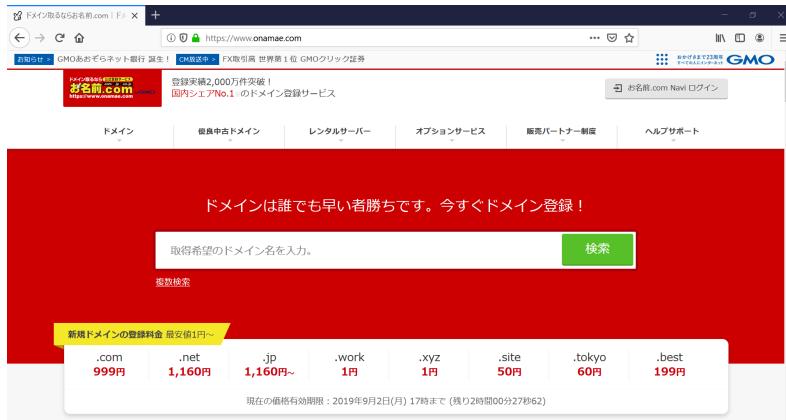
▲図 3.26 NS レコードの Value に書かれたネームサーバをメモしておく

3.3.3 ネームサーバをお名前.com から Route53 に変更

それでは自分のドメイン名のネームサーバを Route53 に変更する設定を行いましょう。ブラウザの別タブでお名前.com のトップページを開いて右上の「ドメイン Navi ログイン」をクリックしてください。（図 3.27）

^{*9} 数字や TLD は人によって異なります。あなたのドメイン名の NS レコードをメモしてください

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.27 右上の「ドメイン Navi ログイン」をクリック

ログイン画面（図 3.28）が表示されたら、ドメイン名を買ったときに発行されたお名前 ID^{*10}とパスワードを入力し、「ログイン」を押して管理画面のドメイン Navi にログインします。



▲図 3.28 お名前 ID とパスワードを入れてドメイン Navi にログイン

ログインするとドメイン Navi のトップページではなく「ドメイン契約更新」の画面（図 3.29）が表示されます。^{*11}

*¹⁰ もしお名前 ID を忘れてしまったら、ドメイン登録したときに届くメールに記載されています

*¹¹ AB テストの影響でログインするお名前 ID によってドメイン Navi の見た目が異なるかもしれません。

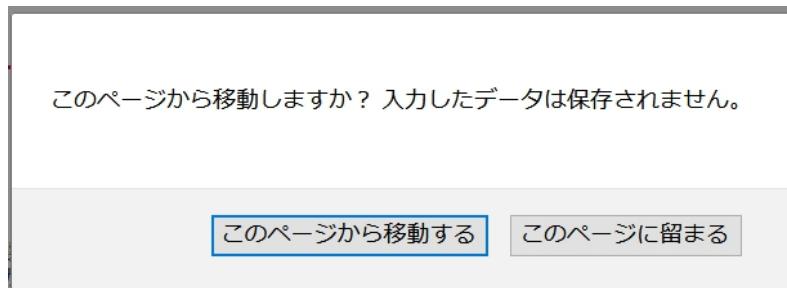
3.3 ドメイン名のネームサーバを Route53 に変更



▲図 3.29 ドメイン Navi ドメイン契約更新

繰り返しになりますが「ドメイン契約更新」の画面が表示されるのは、ドメイン Navi がログインするたびに「来年の更新をしませんか？」と聞いてくる仕様だからです。先ほど買ったドメイン名の設定変更がしたいので、上部のメニューで左から 2 つ目の「ドメイン」をクリックしてください。

他のページへ移動しようとすると「更新手続きをお忘れではございませんか？」（中略）更新が必要なドメインはお早めにお手続きされることをお勧めいたします。」と警告が出ますが（図 3.30）、前述のとおり有効期限は 1 年も先です。問題ありませんので「更新画面から移動する」を押してください。



▲図 3.30 ドメイン Navi 更新アラート

ドメイン一覧（図 3.31）には自分が買ったドメイン名（筆者なら startdns.fun）があります。買ったばかりのドメイン名はネームサーバが「初期設定」になっているので、そこ

見た目に関わらず管理画面でやりたいことは同じで「自分が買ったドメイン名のネームサーバを、デフォルトのお名前.com から Route53 に変更したい」だけです

第3章 AWS のネームサーバ (Route53) を使ってみよう

をクリックして「ネームサーバー設定」の画面に進みます。

ドメイン名	更新期限日	契約更新	Whois情報 公開代行
startdns.fun	2020/03/01 (残 180日)	更新する	

▲図 3.31 ドメイン Navi ドメイン一覧で「初期設定」を押す

ネームサーバー設定の画面（図 3.32）を開いたら、少し下の「2. ネームサーバーの選択」までスクロールしてください。

▲図 3.32 ドメイン Navi ネームサーバー設定

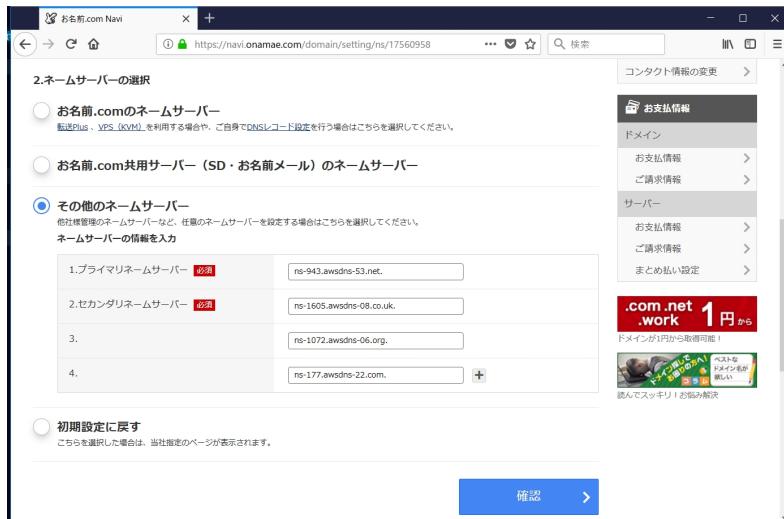
変更前（図 3.33）は「お名前.com のネームサーバー」になっているので「その他のネームサーバー」というラジオボタンを選択（図 3.34）します。「ネームサーバー情報を入力」と書いてあるところに、さきほどメモした Route53 のネームサーバを 1 行ずつ書いてく

3.3 ドメイン名のネームサーバを Route53 に変更

ださい。初めは入力欄が3つしかないですが+を押すと追加できます。4つとも入力出来たら「確認」を押します。



▲図 3.33 ネームサーバーの選択 変更前



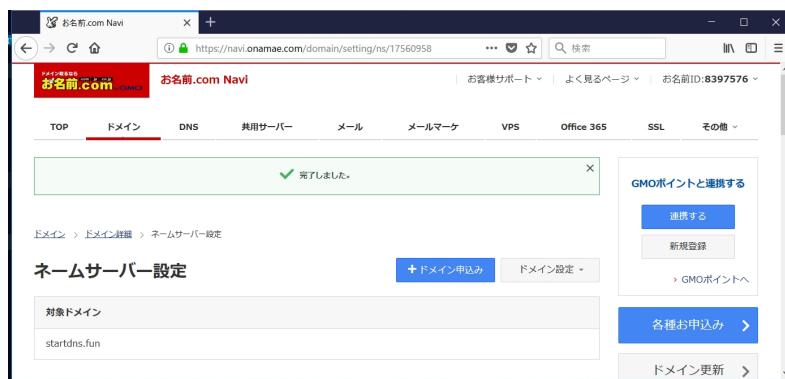
▲図 3.34 さきほどメモした Route53 のネームサーバを 4 つ書く

「確認」を押すと確認画面（図 3.35）が出てくるので、ネームサーバー情報がさきほどメモした Route53 のネームサーバーになっていることを確認して「OK」を押します。



▲図 3.35 ネームサーバの設定確認

「完了しました。」(図 3.36) と表示されたらネームサーバの設定変更は完了です。ネームサーバの設定変更が完了すると、少し経ってから「[お名前.com] ネームサーバー情報変更完了通知」という件名のメールが届きます。



▲図 3.36 ネームサーバの設定変更完了

3.3.4 TTL が過ぎるまではネームサーバが切り替わらない

ところで今、ネームサーバをお名前.com から Route53 に変更しましたが、古い設定の TTL が過ぎるまでフルリゾルバにはキャッシュが残っています。

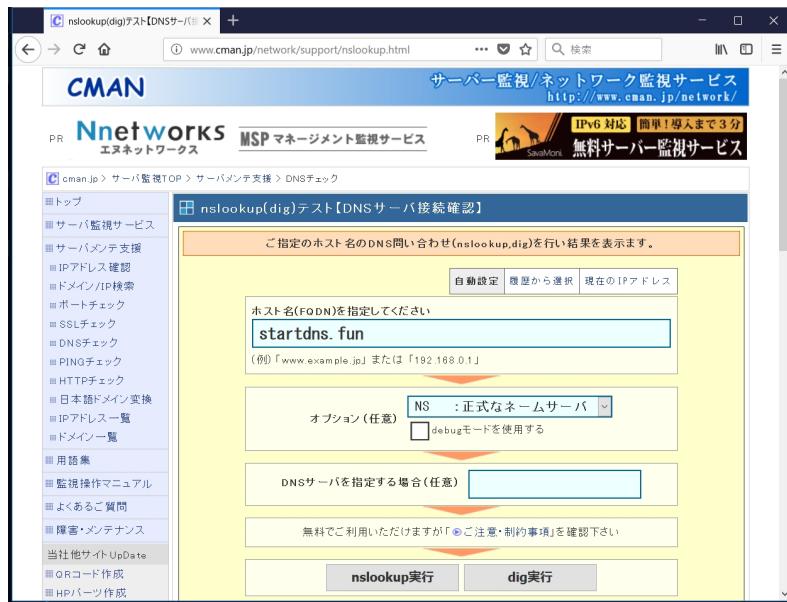
TTL とは Time To Live の略でキャッシュ保持時間のことです。先ほど「nslookup(dig) テスト【DNS サーバ接続確認】^{*12}」で NS レコードを確認したとき、ANSWER SECTION に表示されていた「300」というのが TTL の値です。

```
;; ANSWER SECTION:  
startdns.fun.      300 IN NS    dns1.onamae.com.  
startdns.fun.      300 IN NS    dns2.onamae.com.
```

TTL に 300 と書かれているので、最大で 300 秒 = 5 分待てばキャッシュの期限が切れてネームサーバは Route53 に切り替わるはずです。再び「nslookup(dig) テスト【DNS サーバ接続確認】」で「ホスト名 (FQDN) を指定してください」という欄に自分のドメイン名を入力（図 3.37）して、オプションで「NS：正式なネームサーバ」を選択したら「dig 実行」を押してください。

^{*12} <https://www.cman.jp/network/support/nslookup.html>

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.37 再び自分のドメイン名のネームサーバを調べる

「dig 実行」を押したら、ちょっと下にスクロールして確認結果の ANSWER SECTION というところを見てください。次のように表示されていればネームサーバはちゃんと Route53 へ切り替わっています。数字や TLD は人によって異なるため、先ほどメモしておいた Route53 のネームサーバが表示されていることを確認してください。

```
; ANSWER SECTION:  
startdns.fun. 3600 IN NS ns-1072.awsdns-06.org.  
startdns.fun. 3600 IN NS ns-1605.awsdns-08.co.uk.  
startdns.fun. 3600 IN NS ns-177.awsdns-22.com.  
startdns.fun. 3600 IN NS ns-943.awsdns-53.net.
```

よく「DNS を変更したけど浸透には時間がかかる」のように言われますが、これは正確に言えば「TTL に指定された時間が過ぎるまではフルリゾルバに古いリソースレコードのキャッシュが残っているため、ネームサーバにある新しいリソースレコードの情報を取りに行かず、キャッシュの情報が使われてしまう」ということです。人によって異なるフルリゾルバを使っているため、たとえばサイトリニューアルで A レコードを書き換えたときに、Web 制作会社の A 社は新しいサイトが見られるがクライアントの B 社ではまだ古いサイトが表示される、といった現象が起こります。

理屈さえ分かれば、「浸透」というふんわりした言葉を信じて「いつ切り替わるのか

なー」と気長に待つ必要はありません。変更前の TTL を把握しておけば、最大でどれだけ待てば新しいリソースレコードの情報に切り替わるのかは分かります。事前に TTL を短くしておくことで反映までにかかる時間を短くすることができます。いつまでに切り替わるのか正確に分かっていれば、逆にその時間までに切り替わらなかったときにも「何かがおかしいぞ」ときちんと気付くことができます。

逆に当面は書き換える予定がないリソースレコードについては、TTL を長くしておけばフルリゾルバが都度ネームサーバまで聞きに来なくてよいのでキャッシュヒット率が上がりりますし、万が一ネームサーバが死んでしまったときもしばらくはキャッシュでしのげる所以障害の影響範囲を狭めることができます。

「どうしてもキャッシュを今すぐ消したい！」という場合は、もし情シスが自社のフルリゾルバを管理しているのであれば「フルリゾルバのキャッシュをクリアして！」と頼めばキャッシュがなくなっていて、改めてネームサーバへリソースレコードを問い合わせに行ってくれます。

【コラム】名前解決はプッシュ型じゃなくてプル型

名前解決の仕組みは「エンドユーザからの問い合わせを契機にフルリゾルバがネームサーバへプルしにいく」ものであり、「リソースレコードの追加や変更を契機にネームサーバがフルリゾルバへプッシュしにいく」ものではありません。

MySQL のレプリケーションなどもそうですが、マスターがスレーブにデータをプッシュしにいくのか、スレーブがマスターからデータをプルしてくるのか、つまり「何を契機にどっちが渡しに（もしくは取りに）動くのか？」を理解しておくのは大切なことです。^{*13}

3.3.5 使われるのは親と子どちらの NS レコードの TTL？

ちなみに「ネームサーバがいつ切り替わるのかは NS レコードの TTL を見れば分かる」と前述しましたが、実は NS レコードは 2 箇所にあります。技術書典のウェブサイトで使われている techbookfest.org というドメイン名を例に、あなた自身がフルリゾルバになった気持ちでルートネームサーバから順を追って名前解決の流れをたどっていってみましょう。

^{*13} ちなみに MySQL のレプリケーションは、スレーブがマスターからデータをプルします

先ずはフルリゾルバになりきって、dig コマンドでルートネームサーバに「techbookfest.org の NS レコードを教えて」と話しかけてみましょう。もちろんルートネームサーバ自身は techbookfest.org の NS レコードは知らないので、答え (ANSWER SECTION) は返ってこず、代わりに「org については a0.org.afilias-nst.info (他 5つ) というネームサーバに任せているよ」という委任先の情報 (AUTHORITY SECTION) が返ってきます。^{*14}これが 1 往復目です。

```
# dig +norecurse +noall +author techbookfest.org ns @a.root-servers.net
org.          172800  IN      NS      a0.org.afilias-nst.info.
org.          172800  IN      NS      a2.org.afilias-nst.info.
org.          172800  IN      NS      b0.org.afilias-nst.org.
org.          172800  IN      NS      b2.org.afilias-nst.org.
org.          172800  IN      NS      c0.org.afilias-nst.info.
org.          172800  IN      NS      d0.org.afilias-nst.org.
```

ルートネームサーバから「org については a0.org.afilias-nst.info (他 5つ) に任せている」という情報が得られたので、続けて a0.org.afilias-nst.info に「techbookfest.org の NS レコードを教えて」と話しかけてみましょう。するとやはり答え (ANSWER SECTION) は返ってこず、また代わりに「techbookfest.org については ns-cloud-b1.googledomains.com (他 3つ) というネームサーバに任せているよ」という委任先の情報 (AUTHORITY SECTION) が返ってきました。これが 2 往復目です。

```
$ dig +norecurse +noall +author techbookfest.org ns @a0.org.afilias-nst.info
techbookfest.org.    86400   IN      NS      ns-cloud-b1.googledomains.com.
techbookfest.org.    86400   IN      NS      ns-cloud-b4.googledomains.com.
techbookfest.org.    86400   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.    86400   IN      NS      ns-cloud-b3.googledomains.com.
```

「techbookfest.org については ns-cloud-b1.googledomains.com (他 3つ) に任せている」という情報が得られたので、3 度目の正直で ns-cloud-b1.googledomains.com に「techbookfest.org の NS レコードを教えて」と話しかけてみましょう。するとようやく答え (ANSWER SECTION) が返ってきました。^{*15}これが 3 往復目です。

^{*14} フルリゾルバになりきって名前解決がしたかったので非再帰的に問い合わせる「+norecurse」というオプションを、そして見やすくするため AUTHORITY SECTION だけを表示したかったので「+noall +author」というオプションを付けています。またフルリゾルバではなく、ネームサーバに直接問い合わせたかったので@で対象のネームサーバを指定しています

^{*15} 今度は ANSWER SECTION だけを表示したかったので「+noall +answer」というオプションを付けています

```
$ dig +norecurse +noall +answer techbookfest.org ns @ns-cloud-b1.googledomains.com
techbookfest.org.      21600   IN      NS      ns-cloud-b1.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b3.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b4.googledomains.com.
```

2 往復目と 3 往復目を並べて比較してみましょう。2 往復目の org を任されている a0.org.afilias-nst.info を「親ネームサーバ」、3 往復目の techbookfest.org を任されている ns-cloud-b1.googledomains.com を「子ネームサーバ」とした場合、親にも子にも NS レコードがあることが分かります。

```
$ dig +norecurse +noall +author techbookfest.org ns @a0.org.afilias-nst.info
techbookfest.org.      86400   IN      NS      ns-cloud-b1.googledomains.com.
techbookfest.org.      86400   IN      NS      ns-cloud-b4.googledomains.com.
techbookfest.org.      86400   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.      86400   IN      NS      ns-cloud-b3.googledomains.com.

$ dig +norecurse +noall +answer techbookfest.org ns @ns-cloud-b1.googledomains.com
techbookfest.org.      21600   IN      NS      ns-cloud-b1.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b3.googledomains.com.
techbookfest.org.      21600   IN      NS      ns-cloud-b4.googledomains.com.
```

親と子の NS レコードはどちらも同じ内容ですが、TTL だけが 86400 秒（1 日）と 21600 秒（6 時間）で異なります。このとき、techbookfest.org を任されている、つまり権威（AUTHORITY）を持っているのは子の方なので、フルリゾルバでのキャッシュ保持時間は子の TTL で指定されている 6 時間になる、と筆者は思っていました。

しかし親切な読者の方から教えていただいたのですが、この挙動はフルリゾルバの実装に依存し、子の TTL に従うフルリゾルバの方が多いものの、OCN のように親の TTL に従うフルリゾルバも一部存在するのだそうです。その場合、フルリゾルバでのキャッシュ保持時間は 6 時間ではなく、親の TTL で指定されている 1 日になってしまいます。^{*16}

3.3.6 【ドリル】ネームサーバを変えること ≠ レジストラを変えること

問題

あなたは今、自分が買ったドメイン名のネームサーバをデフォルト設定のお名前.com から、Route53 のネームサーバに変更しました。1 年後に「ドメイン名がもうすぐ有効期

^{*16} ただしこの件については、筆者が契約している ISP は OCN ではなかったため自身で検証ができていません。「親の TTL に従うフルリゾルバも存在するそうだ」という伝聞で申し訳ないのですが、2018 年 5 月に投稿されたこちらの Qiita で紹介されています。技術書典 4 で販売された「DNS をはじめよう」をフォローする感じの記事にしたい - Qiita <https://qiita.com/soushi/items/7432710308a78a684299>

限を迎えるので更新しましょう！」と連絡してくるのはどちらですか？

- A. Route53
- B. お名前.com

答え _____

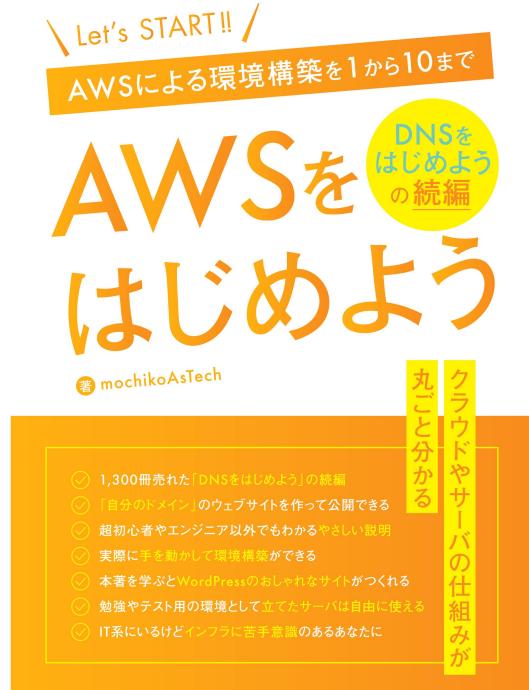
解答

正解は B です。今行った作業はドメイン名のネームサーバを、デフォルト設定のお名前.com から AWS の Route53 というネームサーバに変更しただけです。レジストラ（ドメイン名を購入するお店）をお名前.com から Route53 に変更する「レジストラ移管」を行ったわけではないので、1 年後に更新確認の連絡をしてくるのもお名前.com です。

Route53 の Registered domains というメニューからドメイン名を購入することもできますがお名前.com に比べると値段が高いので、Route53 のネームサーバを使いたいだけであれば、本著のようにドメイン名はお名前.com で買って Route53 のネームサーバを使う方がお勧めです。

【コラム】 DNS の次は AWS をはじめたい？

「DNS をはじめよう」の続編である「AWS をはじめよう」(図 3.38) は、AWS の環境で実際にサーバを立て、自分のドメイン名でウェブサイトを作って公開するところまでがつりやるハンズオン形式の本です。BOOTH^{*17}で購入できますので、本著を読み終わった後に「折角ドメイン名買ったし、サイトを作るところでやってみたい！」と思ったら続けてチャレンジしてみてください。



▲図 3.38 AWS をはじめよう

ちなみに BOOTH はピクシブ株式会社^{*18}が運営している同人誌の通販及びダウンロード販売サイトで、書籍版を購入すると 1~2 営業日以内に本が BOOTH 倉庫からネコポスで送られてきます。PDF 版なら購入後すぐにダウンロードして読むことができます。技術書典で頒布されている同人誌の多くは BOOTH でも購入できますので、気になる方は「技術書典」のタグで検索^{*19}してみてをお勧めします。

「AWS をはじめよう」は Amazon の Kindle でも販売していますが、そちらは PDF を一度画像に変換しており本文の検索ができないため、BOOTH の PDF 版の方がお勧めです。

*¹⁷ <https://mochikoastech.booth.pm/>

*¹⁸ イラストを投稿できるSNS、pixivでお馴染み。<https://www.pixiv.net/>

*¹⁹ <https://booth.pm/ja/search/技術書典>

第4章

dig と whois を叩いて学ぶ DNS

性能向上のための格言に「推測するな計測せよ」というものがありますが、障害発生時にも同じことが言えます。DNS 絡みのトラブルが起きたとき、あるいはサイトが意図した動作をしないとき、調べ方さえ知っていれば原因を調査できますし「恐らく DNS の浸透が原因じゃないかと…48 時間ほど待てば多分…」のようなエンジニアらしからぬ回答をしなくて済みます。

この章では dig コマンドと whois コマンドを実際に叩いて色々なことを調べながら、DNS の仕組みを学んでいきましょう。

4.1 dig と whois のインストール

dig コマンドや whois コマンドを使いたくても、そこにインストールされていなければ使えません。Mac にはどちらのコマンドも元々インストールされているので、あなたが Mac ユーザであればターミナルを起動すればすぐに使うことができます。

CentOS や AmazonLinux などのサーバ環境がある人は、そこで root になって次の yum コマンドを叩けばインストールできます。

```
dig コマンドのインストール  
# yum -y install bind-utils  
  
whois コマンドのインストール  
# yum install -y jwhois
```

4.1.1 Mac も Linux のサーバ環境もない場合

パソコンは Windows だし手近にコマンドを叩けるような環境はない…という方は、Route53 を使うために折角 AWS でアカウントを作ったので、EC2^{*1}でインスタンス^{*2}を立ててみるのがお勧めです。サーバスペックごとにインスタンスタイプという区分があるので、t2.micro というインスタンスタイプならアカウント作成から 1 年間は毎月 750 時間まで無料で使えます。ここでは詳しく説明しませんが^{*3}、AWS のマネジメントコンソールで EC2 ダッシュボードを開いて「インスタンスの作成」を押したら、後は画面の表示に従ってぽちぽち選んでいくだけで 5 分もあればすぐにサーバが作れます。^{*4}

インスタンス立てるのはちょっと無理！ という場合は、第3章「AWS のネームサーバ（Route53）を使ってみよう」で使った「nslookup(dig) テスト【DNS サーバ接続確認】」^{*5}というページや各レジストリの Whois 情報確認サイトで疑似的に dig や whois を叩く形で代用しても構いません。

^{*1} Elastic Compute Cloud の略。AWS にはいろいろなサービスがありますが EC2 はいわゆるサーバのことです

^{*2} AWS ではサーバのことをインスタンスを呼びます

^{*3} 本著の続編である「AWS をはじめよう」には EC2 でインスタンスを立てる手順が詳しく載っています

^{*4} 月に 750 時間まで無料とありますが 24 時間 × 31 日で 744 時間なので要は 1 年間ずっと無料ということです

^{*5} <https://www.cman.jp/network/support/nslookup.html>

4.2 nslookup はもう卒業！ dig コマンドの便利な使い方

ドメイン名に紐づく IP アドレスを調べたいとき、あるいは IP アドレスに紐づくドメイン名を調べたいときは dig コマンド^{*6}を使用します。

たとえば本著（改訂第 2 版）を頒布する技術書典 7 のウェブサイトは <https://techbookfest.org/event/tbf07> という URL です。このサイトがどこのウェブサーバに乗っているのか知りたかったら dig コマンドで引数に「techbookfest.org」というドメイン名を渡してやれば調べられます。早速 dig を叩いてみましょう。

```
$ dig techbookfest.org

; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.68.rc1.el6_10.3 <>> techbookfest.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54916
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 4, ADDITIONAL: 8

;; QUESTION SECTION:
;techbookfest.org.           IN      A

;; ANSWER SECTION:
techbookfest.org.        300     IN      A      216.239.38.21
techbookfest.org.        300     IN      A      216.239.32.21
techbookfest.org.        300     IN      A      216.239.34.21
techbookfest.org.        300     IN      A      216.239.36.21

;; AUTHORITY SECTION:
techbookfest.org.       86400   IN      NS      ns-cloud-b4.googledomains.com.
techbookfest.org.       86400   IN      NS      ns-cloud-b3.googledomains.com.
techbookfest.org.       86400   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.       86400   IN      NS      ns-cloud-b1.googledomains.com.

;; ADDITIONAL SECTION:
ns-cloud-b1.googledomains.com. 345600 IN A      216.239.32.107
ns-cloud-b1.googledomains.com. 345600 IN AAAA    2001:4860:4802:32::6b
ns-cloud-b2.googledomains.com. 345600 IN A      216.239.34.107
ns-cloud-b2.googledomains.com. 345600 IN AAAA    2001:4860:4802:34::6b
ns-cloud-b3.googledomains.com. 345600 IN A      216.239.36.107
ns-cloud-b3.googledomains.com. 345600 IN AAAA    2001:4860:4802:36::6b
ns-cloud-b4.googledomains.com. 345600 IN A      216.239.38.107
ns-cloud-b4.googledomains.com. 345600 IN AAAA    2001:4860:4802:38::6b

;; Query time: 260 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
```

^{*6} dig は domain information groper の略です。ちなみに Google に翻訳してもらったら「ドメイン情報痴漢」でした。gropo は手さぐりするという意味なので、ドメイン情報を手探りして調べててくれるということです。余談ですが ping の g も同じ groper なので dig が痴漢なら ping も痴漢です

```
; ; WHEN: Thu Aug 29 20:22:39 2019
;; MSG SIZE  rcvd: 395
```

ただ techbookfest.org に紐づく IP アドレスが知りたかっただけなのに、ものすごくいっぱい出てきました。

dig はまるでブキチ^{*7}のようなコマンドなのでちょっと聞いただけで「調べてきた結果を教えまし！ まずキミの質問はこれでし！ このドメイン名に紐づく IP アドレスはこれとこれとこれとこれなのでし！ ちなみに IP アドレスを教えてくれたネームサーバの名前はこれでし、ネームサーバの IP アドレスはこっちなのでし！ それからフルリゾルバは 127.0.0.1 で調査には 260msec かかったのでし！」と調査の過程や付加情報まで含めて全部教えてくれます。

有難いのですが情報は多くありすぎても混乱します。「いいから簡潔に techbookfest.org に紐づく IP アドレスだけを教えて！」という場合は +short というオプションを付けましょう。+short さえつければ dig はごく簡潔に答えてくれます。

```
$ dig techbookfest.org +short
216.239.34.21
216.239.36.21
216.239.38.21
216.239.32.21
```

4.2.1 host や nslookup じゃダメですか？

ちなみに host コマンドや nslookup コマンドでも同じように調べることができます。

```
$ host -t a techbookfest.org
techbookfest.org has address 216.239.36.21
techbookfest.org has address 216.239.38.21
techbookfest.org has address 216.239.32.21
techbookfest.org has address 216.239.34.21
```

```
$ nslookup
> set type=a
> techbookfest.org
```

^{*7} スプラトゥーン 2 に出てくる武器屋の店主。ブキの話になると超早口で果てしなく解説するブキマニア。語尾が「でし！」でオタク感があふれていて素晴らしく可愛い

```
Server:          172.31.0.2
Address:         172.31.0.2#53

Non-authoritative answer:
Name:  techbookfest.org
Address: 216.239.36.21
Name:  techbookfest.org
Address: 216.239.38.21
Name:  techbookfest.org
Address: 216.239.32.21
Name:  techbookfest.org
Address: 216.239.34.21
> exit
```

大昔は nslookup を叩くと「nslookup は非推奨だし、将来的には廃止されるから今後は dig や host を使ってね^{*8}」という警告メッセージが都度出ていたので、その頃を知っている人は「nslookup っていざれなくなるんでしょう？」という認識かと思いますが、実際は BIND^{*9}9.9.0a3 が公開されたタイミングで nslookup からこの警告メッセージは消え、リリースノートには「nslookup を非推奨として扱うのはもうやめるね。非推奨の警告も消したよ^{*10}」と書かれていますので、nslookup が非推奨だの廃止だのという話は一旦なくなったようです。

実際、何もトラブルが起きておらず、単純に名前解決した結果を知りたいだけであれば host や nslookup でも事足ります。ですがトラブル発生時の調査手段として host や nslookup を使おうとすると、必要な情報が不足していたり調べてきた結果を下手に加工して出力したりするため、どちらもあまり使いやすいコマンドとは言えません。これを機に今後は dig を使っていきましょう！

4.3 Whois を叩いてドメイン名や IP の持ち主を調べよう

whois コマンドを使うと、第 1 章「ドメイン名と Whois」でお話ししたような Whois 情報を調べることができます。しかも whois コマンドはドメイン名の持ち主だけでなく、IP アドレスの持ち主を調べることもできます。

^{*8} Note: nslookup is deprecated and may be removed from future releases. Consider using the 'dig' or 'host' programs instead. と表示されていました

^{*9} BIND はフルリゾルバとネームサーバ両方の機能を持つ DNS サーバで、ISC (Internet Software Consortium) によって開発が行われています。多くの DNS サーバで BIND が採用されていますが「夏の BIND 脆弱性祭り」などと揶揄されるほど脆弱性の注意喚起とそれに伴うアップデート推奨が多いため、最近は Unbound など他の DNS サーバへの乗り換えもよく聞かれます

^{*10} nslookup is no longer to be treated as deprecated. Remove "deprecated" warning message.

ドメイン名の持ち主を調べる
\$ whois ドメイン名

IP アドレスの持ち主を調べる
\$ whois IP アドレス

4.3.1 【ドリル】ドメイン名の有効期限が分からぬ

問題

あなたはかき氷を作れるきょろちゃん^{*11}のサイト運用担当者です。(図 4.1)



▲図 4.1 きょろちゃん

このたび運用担当を新人へ引き継ぐことになりました。引継ぎ資料を読んだ新人から「ここに毎年ドメイン名の更新作業が必要って書いてあるんですけど、次の更新っていつごろですか?」と聞かれました。ドメイン名の有効期限を知るにはどのコマンドを叩けばいいですか?

- A. dig tigerkyoro.jp +short
- B. whois tigerkyoro.jp

^{*11} <http://tigerkyoro.jp/>

答え _____

解答

正解は B です。whois コマンドを叩くと「Expires on」や「Registry Expiry Date」といった項目に有効期限が表示されます。今回の tigerkyoro.jp であれば、次のように whois コマンドを叩けば、ドメイン名の有効期限が 2020/01/31 であることが分かります。

```
$ whois tigerkyoro.jp
[Querying whois.jprs.jp]
[whois.jprs.jp]
[ JPRS database provides information on network administration. Its use is      ]
[ restricted to network administration purposes. For further information,      ]
[ use 'whois -h whois.jprs.jp help'. To suppress Japanese output, add '/e'      ]
[ at the end of command, e.g. 'whois -h whois.jprs.jp xxx/e'.                  ]

Domain Information:
[Domain Name]          TIGERKYORO.JP

[Registrant]           Tiger Corporation

[Name Server]          ns.namedserver.net
[Name Server]          ns2.namedserver.net
[Signing Key]

[Created on]            2016/01/18
[Expires on]            2020/01/31
[Status]                Active
[Last Updated]          2019/02/01 01:17:18 (JST)

Contact Information:
[Name]                  Clarivate Analytics (Japan) Co.,Ltd.
[Email]                 domain.jp@markmonitor.com
[Web Page]
[Postal code]           107-6119
[Postal Address]        Akasaka Park Building, 19F,
                        5-2-20, Akasaka,Minato-ku,Tokyo
[Phone]                 03-4589-3900
[Fax]                   03-4589-3240
```

但し jp ドメイン名の場合、たとえば 2019 年末の時点で新人が更新費用を支払って更新の手続きを済ませたとしても、Whois の「Expires on」の表示が変わるのは、いま表示されている有効期限（2020/01/31）の翌月 1 日（2020/02/01）になってからです。^{*12}com

^{*12} JP ドメインの更新費用を支払ったのですが 有効期限が更新されていません。 – さくらのサポート情報 <https://help.sakura.ad.jp/hc/ja/articles/206057382-JP%E3%83%89%E3%83%A1%E3%82%>

や net などのドメイン名であれば、更新手続きを完了すると Whois の有効期限の表示もすぐに更新されます。

4.4 dig を叩いてリソースレコードを確認してみよう

ここからは調べたい内容に応じて、どう dig コマンドを叩けばいいのか？を 1つ1つ確認していきます。dig を叩いてもサーバは壊れないし、地球も爆発しません。軽い気持ちでどんどん叩いて dig に慣れていきましょう。

4.4.1 A レコード

たとえば「www.example.jp は 203.0.113.222」や「mx.example.com は 203.0.113.22」のように、ドメイン名と IP アドレスを紐づけているのが A レコードです。あなたが「このサイトはどこにあるんだろう？」と思ったら、次のように dig コマンドを叩いてみましょう。^{*13}するとそのドメイン名に紐づく IP アドレスが表示されるはずです。

```
$ dig ドメイン名 a +short
```

4.4.2 【ドリル】ウェブサーバはどこにある？

問題

あなたは CIAO ちゅ～るの「ちゅ～るメーカー」というサイト (<https://www.inaba-petfood.co.jp/ciao-chuuuuuuuuuuuru/>) の運用担当になりました。^{*14} (図 4.2)

ところが前任者が急に辞めてしまって引継ぎ資料も見当たらず、このサイトがどこのウェブサーバで動いているのかちっとも分かりません。

A4%E3%83%B3%E3%81%AE%E6%9B%B4%E6%96%B0%E8%B2%BB%E7%94%A8%E3%82%92%E6%94%AF%E6%89%95%E3%81%A3%E3%81%9F%E3%81%AE%E3%81%A7%E3%81%99%E3%81%8C-%E6%9C%89%E5%8A%B9%E6%9C%9F%E9%99%90%E3%81%8C%E6%9B%B4%E6%96%B0%E3%81%95%E3%82%8C%E3%81%A6%E3%81%84%E3%81%BE%E3%81%9B%E3%82%93-

^{*13} 一番左の「\$」はサーバにログインしたときにユーザ名やサーバの名前が表示される「プロンプト」というものを表しています。この「\$」は実際は入力しなくていいので、dig 以降を入力してください

^{*14}もちろん例えです。本著及び筆者は「ちゅ～るメーカー」をはじめ、例題に使用させていただいたサイトと何の関係もありません。「ちゅ～る」と聞くと目の色を変える猫を飼っているだけです

4.4 dig を叩いてリソースレコードを確認してみよう



▲図 4.2 あなたのねこちゃんとちゅ～る CM をつくろう！ | CIAO ちゅ～るメーカー

「ちゅ～るメーカー」のウェブサーバはどこのサービスを使っているのでしょうか？

- A. さくらインターネット
 - B. AWS
 - C. IDCF クラウド

答え _____

解答

正解は A です。

先ず「dig ドメイン名 a +short」でドメイン名から IP を引いてみましょう。サイトの URL が <https://www.inaba-petfood.co.jp/ciao-chuuuuuuuuuru/> ので、確認すべきドメイン名は www.inaba-petfood.co.jp です。

```
$ dig www.inaba-petfood.co.jp +short  
133.242.53.94
```

IP アドレスは分かりましたが、IP アドレスだけではまだこのクラウドサービスなのか分かりません。さらに whois コマンドでこの IP アドレスの持ち主を調べてみましょう。

う。¹⁵

whois 133.242.53.94 を叩いてみると、Organization に SAKURA Internet Inc. とありますので、この IP アドレスはさくらインターネットのものであることが分かります。これで CIAO ちゅ～るのサイトがさくらインターネットのサービス上で動いていることが分かりました。

```
$ whois 133.242.53.94
(中略)
Network Information:
a. [Network Number]           133.242.53.0/24
b. [Network Name]             SAKURA-NET
g. [Organization]              SAKURA Internet Inc.
(後略)
```

【コラム】同じドメイン名で A レコードを複数作るとどうなる？

たとえばウェブサーバが 2 台あった場合、次のようにまったく同じドメイン名で IP アドレスが異なる A レコードを複数作ることができます。

```
www.example.com. 300 IN A 203.0.113.111
www.example.com. 300 IN A 203.0.113.222
```

このような A レコードを設定した状態で、dig コマンドで www.example.com の A レコードを問い合わせると、次のような応答が返ってきます。

```
$ dig www.example.com a +short
203.0.113.111
203.0.113.222

または

$ dig www.example.com a +short
203.0.113.222
203.0.113.111
```

問い合わせるたびに応答される 2 つの IP アドレスの順番が変わって「203.0.113.111」が先になったり「203.0.113.222」が先になったりすることで、

¹⁵ この例では dig コマンドで出てきた IP が 1 つだけでしたが、複数出てきた場合はどちらでも構いません

エンドユーザのアクセスを2つに割り振ることができます。これをDNS ラウンドロビンと言います。ロードバランサーがなくてもこのようにDNS の設定だけで簡易な負荷分散^{*16}ができます。

ちなみに同じドメイン名で複数のレコードを作る場合、次のようにレコードによって異なる TTL を設定することは非推奨となっています。^{*17}

```
www.example.com. 300 IN A 203.0.113.111  
www.example.com. 600 IN A 203.0.113.222
```

お名前.com のネームサーバでこのようなレコードを作ろうとすると「ホスト名が同じ文字列のレコードを複数ご登録頂く場合は、異なる TTL は設定できません」と表示されて、そもそも設定できません。蛇足ですが筆者が個人で運用しているネームサーバ^{*18}で前述のような異なる TTL を設定してみた上でdigで問い合わせてみたところ、次のようにTTLは300に揃えられた状態で応答が返ってきました。

```
www.example.com. 300 IN A 203.0.113.111  
www.example.com. 300 IN A 203.0.113.222
```

4.4.3 MX レコード

「○○@example.co.jp」というメールアドレス宛てにメールを送ったらこのメールサーバで受信します、という設定をしているのがMX レコードです。「このメールアドレスってメールはどこで受信してるんだっけ？」と思ったら、次のようにdigコマンドを叩いてみましょう。

^{*16} 但し応答で返ってきた複数のIP アドレスのうち、どのIP アドレスにアクセスするのかはクライアントの実装に依存します。もし片方のウェブサーバが落ちてしまっても、ロードバランサーのように自動で分散対象から外してくれる訳でもないので、あくまで擬似的な負荷分散と捉えた方がいいでしょう

^{*17} <https://tools.ietf.org/html/rfc2181#section-5.2>

^{*18} BIND 9.8.2

```
$ dig ドメイン名 mx +short
```

たとえば任天堂の問い合わせ窓口である info@nintendo.co.jp にメールを送ったら、どのメールサーバが受信するのか確認してみましょう。

```
$ dig nintendo.co.jp mx +short  
10 nintendo-co-jp.mail.protection.outlook.com.
```

「nintendo-co-jp.mail.protection.outlook.com」というのがメール受信サーバであることが分かりました。先頭の 10 はプリファレンス値といって「メールサーバが複数台ある場合の優先度」を表しています。MX レコードは複数設定できるため、プリファレンス値が 10 のメールサーバを複数台用意して負荷を分散したり、プリファレンス値が 10 のメールサーバが落ちていたら代わりにプリファレンス値が 20 のメールサーバで受信する、というように冗長性を高めたりできます。

MX レコードから察するに任天堂は Office 365^{*19}を使っているようです。さらに nintendo-co-jp.mail.protection.outlook.com の A レコードを引くと、最終的にはメール受信サーバの IP アドレスまでたどり着くことができます。

```
# dig nintendo-co-jp.mail.protection.outlook.com a +short  
104.47.93.36  
104.47.92.36
```

MX レコードがなければ代わりにウェブサーバ宛てにメールが届く

ところで、もし MX レコードが存在しなかったらどうなるのでしょうか？ メールを送ろうとしたとき、そのドメイン名の MX レコードが存在しなかったらメールを送信できない…のではなく、代わりに A レコードで紐づけられている IP アドレスに対してメールを送ろうとします。^{*20}

つまり「今回は <https://example.co.jp/> というサイトを公開したいだけで、メールを送受信する要件はまったくないので example.co.jp の MX レコードは設定しません」と思っていても、example.co.jp の A レコードさえあれば、そこに書かれたウェブサーバに対してメールが飛んできてしまう可能性があるのです。さらに Postfix^{*21}は CentOS

^{*19} Microsoft の法人向けサービス。メールやグループウェア、Microsoft Officeなどを利用できる

^{*20} <https://tools.ietf.org/html/rfc5321#section-5.1>

^{*21} メール転送エージェント。要はメールを受信したり送信したりするためのメールサーバ

を最小限の構成でインストールしたときでも入っているので、全然意図していなかったけれども、どうせウェブサーバで Postfix が動いていて、エンドユーザから送られてきた個人情報満載なメールをひっそり受信していた！ という可能性もあります。

そのドメイン名でメールを受信する予定がないのであれば、次のようにプリファレンス値を「0（ゼロ）」、メールサーバ名を「.（ドット）」にした Null MX^{*22}を設定することで「メールを受信しません」という意図を明示しておく方が、メールを送る側も受信する側にも余計な負担がなくなってよいかも知れません。

```
example.co.jp.      IN  MX      0  .
```

ただし、メールを受信する要件がなかったとしても、メールを送信する要件がある場合はバウンスマール^{*23}受信のために MX レコードを設定しておくべきです。

4.4.4 〈トラブル〉 test@test.co.jp を使って情報漏洩

あなたは広告代理店 B 社のプロデューサーで、A 社の新製品である花粉症用マスクの先行体験キャンペーンを企画・担当しています。

キャンペーンサイトのフォームから新製品の先行体験キャンペーンに申し込むと、お客様には「応募を受け付けました」というメールが届きます。このメールにはお客様がフォームで記入された住所や電話番号などが書いてあります。

この受付完了メールですが、実は送信元メールアドレス (From) を何にするのか、実装が始まった時点では仕様が決まっていませんでした。そのため送信元メールアドレスは、ディレクターが仕様書に取り合えずのつもりで書いておいた info@test.co.jp というアドレスで実装されてしまい、テストでも誰も気づかないまま本番リリースを迎ってしまいました。

キャンペーンが始まり、花粉症の C さんは早速フォームから申し込みました。しかし C さんは最近引っ越ししたばかりだったので、うっかり古い住所をフォームで送ってしまいました。「応募を受け付けました」メールを見て住所が間違っていることに気づいた C さんは、すぐにメールの返信で info@test.co.jp 宛てに「入力した住所が間違っています。正しい住所は○○なので訂正したいです」という問い合わせを送りました。

しかしあいつまで経っても返事が来ません。C さんがキャンペーン事務局にクレームの電話を入れてエンジニアの D さんが調査をした結果、送信元メールアドレスが A 社ではな

*22 <https://tools.ietf.org/html/rfc7505>

*23 宛先のメールアドレスが間違っていた、などの理由でメールが正常に配信できなかった場合に送信元に対して戻ってくるメールのこと。MAILER-DAEMON やリターンメール、エラーメールとも呼ばれる

↙ info@test.co.jp になっていた問題が発覚しました。果たして C さんが info@test.co.jp 宛てに送ったメールはどこへ行ってしまったのでしょうか？

プロデューサーのあなたもエンジニアの D さんと一緒にこの問題を調査してみましょう。メール受信サーバを知りたいときは dig ドメイン名 mx +short です。

```
$ dig test.co.jp mx +short  
10 mail10.heteml.jp.
```

C さんが info@test.co.jp 宛てに送ったメールを受信しているのは mail10.heteml.jp であることが分かりました。mail10.heteml.jp についてウェブで検索してみたところ、どうやら GMO ペパボがやっているヘテムル^{*24}というクラウドサービスのメールサーバのようです。

OO@test.co.jp に送ったメールはヘテムルで受信している、というところまで分かりましたが、そもそも test.co.jp というドメイン名は誰の持ち物なのでしょうか？ 今度は whois でドメイン名の持ち主を調べてみましょう。

```
$ whois test.co.jp  
[Querying whois.jprs.jp]  
[whois.jprs.jp]  
[ JPRS database provides information on network administration. Its use is      ]  
[ restricted to network administration purposes. For further information,      ]  
[ use 'whois -h whois.jprs.jp help'. To suppress Japanese output, add '/e'      ]  
[ at the end of command, e.g. 'whois -h whois.jprs.jp xxx/e'.                  ]  
  
Domain Information:  
a. [Domain Name]           TEST.CO.JP  
g. [Organization]          EDUCATIONAL ASSESSMENT INSTITUTE LIMITED COMPANY  
l. [Organization Type]    Limited Company  
m. [Administrative Contact] TY2813JP  
n. [Technical Contact]   KT071JP  
p. [Name Server]           dns0.heteml.jp  
p. [Name Server]           dns1.heteml.jp  
s. [Signing Key]  
[State]                   Connected (2020/02/29)  
[Registered Date]         2000/02/14  
[Connected Date]          2000/02/16  
[Last Update]             2019/03/01 01:04:30 (JST)
```

Whois で調べてみると、このように test.co.jp というドメイン名は「EDUCATIONAL ASSESSMENT INSTITUTE LIMITED COMPANY」という名前の有限会社の持ち物である、という情報が出てきました。プロデューサーのあなたとエンジニアの

^{*24} <https://heteml.jp/>

4.4 dig を叩いてリソースレコードを確認してみよう

Dさんはなんだか嫌な予感がして、ブラウザで <http://test.co.jp/>（図 4.3）を開いてみました。すると有限会社教育評価研究所という会社のウェブサイトが表示されました。



▲図 4.3 教育評価研究所

なんと花粉症の Cさんが info@test.co.jp宛てに送ったお問い合わせメールは、クライアントの A社とも広告代理店の B社とも何の関係もない有限会社教育評価研究所という会社に飛んでしまったようです。「エンドユーザの個人情報漏洩だ、やってしまった…」とあなたと Dさんは頭を抱えました。

自分のものでないドメイン名を使うべきでない理由

このように「自分の持ち物でないドメイン名」を勝手に使うのはトラブルの元です。

- test.co.jp
- test.com
- aaa.com
- xxx.com

などはいかにもサンプルっぽいので、なんとなく「田中一郎」や「山田太郎」のようなノリで使いたくなってしまいますが、これらのドメイン名にはいずれもちゃんと持ち主^{*25}がいます。

^{*25}たとえば test.com というドメイン名は DOSarrest というサービスをやっている Internet Security LTD, という会社が持ち主ですし、前述のとおり test.co.jp というドメイン名は有限会社教育評価研究所

そのためたとえば宅配ピザのステージングサイトで、仮に `tanaka@test.co.jp` というメールアドレスのテストユーザを作ってテストを行うと、実際に有限会社教育評価研究所という会社に会員登録完了メールや注文完了メールが飛んでいってしまい、相手方にご迷惑であると共に情報漏洩の恐れもあります。なかなかぴんと来ないかも知れませんが、誰かがアンケートに適当な住所を書いたせいで全然関係ない自分の家にダイレクトメールが届いたら迷惑ですよね。

ありがちですがお問い合わせフォームのメールアドレスの例に `sample@test.com` のように書くのもやめましょう。銀行に行って申請用紙のサンプルに自分の住所が部屋番号まではばっちり書かれていたらぎょっとしますよね？ それと同じことです。実在する他人の住所や電話番号を勝手に使わないのと同じように、他人のドメイン名も勝手に使わないようしましょう。

例示やテスト専用のドメイン名を使おう

他人が持っているドメイン名を勝手に使ってはいけない、ということは分かりました。では誰のものでないドメイン名ならいいのでは？ と思われるかも知れませんが、現時点で誰のものでもなくとも明日には誰かがそのドメイン名を登録するかも知れませんのでこれもやはりお勧めしません。

じゃあどうしたらいいのかというと、実はインターネットでは「例示やテストで使っていいドメイン名」というものが定められて^{*26}います。テストユーザのメールアドレスや、フォームで例として書くメールアドレスには次のドメイン名を使いましょう。

- `example.co.jp`
- `example.jp`
- `example.com`
- `example.net`

これらのドメイン名であれば将来的に誰かのものになる可能性もありませんし、予期せぬ第三者へうっかりメールが飛んでしまうことも避けられます。

【コラム】dig のオプションは略せる

という会社が持ち主です

*²⁶ RFC2606 で定められている `example.com` や `example.net`、もしくは JPRS が定めている `example.jp` や `example.co.jp` などを使いましょう。<https://jprs.jp/faq/use/>

dig コマンドには「+short」や「+norecurse」のようなクエリオプションがありますが、それぞれ後ろの何文字かを省略して「+shor」や「+norec」でも実行可能ですね。

このような「コマンドオプションって略せるの？！」といった dig の細かい仕様や仕組みをもっと詳しく知りたい！ という方は、JPRS が公開している「初心者のための DNS 運用入門 - トラブル事例とその解決のポイント -」^{*27}という資料がお勧めです。

4.4.5 NS レコード

ドメイン名のネームサーバを指定しているのが NS レコードです。キャンペーンサイト用にサブドメインを増やしたいけど、サブドメインってどこで作ればいいんだっけ？ というときは次のような dig コマンドを叩いて NS レコードを調べれば分かります。

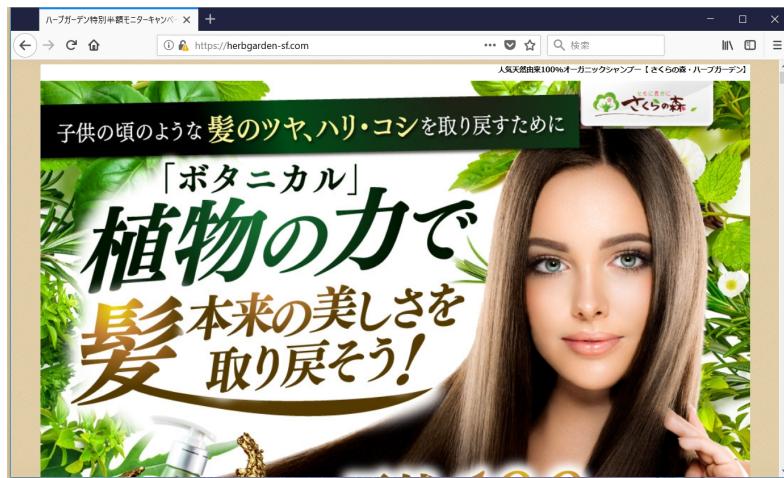
```
$ dig ドメイン名 ns +short
```

4.4.6 【ドリル】他社へのサイト移管時にネームサーバの所在が不明

問題

あなたは Web 制作会社 A 社でボタニカルシャンプーのサイト (<https://herbgarden-sf.com/>) の運用を担当しているエンジニアです。（図 4.4）

^{*27} <https://dnsops.jp/event/20140626/dns-beginners-guide2014-mizuno.pdf>



▲図 4.4 ボタニカルシャンプーのサイト

このたびクライアントの意向でサイトの運用を A 社からライバルの B 社へ移管することになりました。クライアント経由で B 社から「DNS も弊社管理のネームサーバに移管するので現状のゾーン情報をください」と言わされたのですが、インフラに関する資料がなくてネームサーバがどこにあるのか分かりません。このドメイン名のネームサーバはどこにあるのでしょうか？

- A. AWS の Route53
- B. Microsoft の Azure DNS
- C. IDCF クラウド DNS

答え _____

解答

正解は A です。次のように dig コマンドを叩くと「awsdns」という文字を含むネームサーバ名が返ってくるので、ネームサーバが AWS の Route53 であることが分かります。

```
$ dig herbgarden-sf.com ns +short
ns-1030.awsdns-00.org.
ns-910.awsdns-49.net.
ns-1805.awsdns-33.co.uk.
ns-418.awsdns-52.com.
```

ゾーンの中のリソースレコードを確認するには AWS のマネジメントコンソールにログインする必要がありますが、AWS アカウント名やパスワードまではさすがに dig では分からないので、そこはなんとか過去資料から探してください。また Route53 にはホストゾーンのエクスポート機能がないので、リソースレコードを 1つ1つコピーペーストするか CLI^{*28}を駆使して取得する必要があります。

4.4.7 SPF レコード（TXT レコード）

AさんがBさんに手紙を書くとき、封筒の差出人欄に「Cさんより」と書けば簡単に送信元を騙ることができます。メールもそれと同じで、送信元のメールアドレスを詐称して送信することは容易です。迷惑メールの多くはこのように送信元を詐称しているなりすましメールです。

このとき送り主が本物なのか、あるいは自分の持ち物でないドメイン名で送信元を詐称して勝手にメールを送っている「なりすましメール」なのかを確認する手段として、SPF^{*29}レコードというものがあります。SPF レコードは次の dig コマンドで確認できます。

```
$ dig ドメイン名 txt +short
```

SPF レコードなのに txt なの？ と疑問に思われるかも知れません。昔は SPF を設定する方法として SPF レコードと TXT レコードという 2種類のリソースレコードに書くことが推奨されていたのですが、SPF レコードに書く方法は普及せず、最終的に「SPF は TXT レコードで設定すること」となりました。^{*30}

たとえばポケモンだいすきクラブ^{*31}から届く「ポケモンだいすき！ 通信」というメールは、送信元のメールアドレスが noreply@pdc.pokemon.jp です。pdc.pokemon.jp の

^{*28} コマンドラインインターフェイスの略。AWS はブラウザでぼちぼち操作するだけでなく、CLI を用いてコマンドラインからも操作できます

^{*29} Sender Policy Framework の略

^{*30} <https://tools.ietf.org/html/rfc7208#section-3.1> で SPF records MUST be published as a DNS TXT (type 16) Resource Record(RR) [RFC1035] only と書かれています

^{*31} ポケモンの情報サイト。ミミッキュのうたは名曲です。http://www.pokemon.jp/

SPF レコードを確認してみましょう。

```
$ dig pdc.pokemon.jp txt +short  
"v=spf1 include:spf.pdc.pokemon.jp include:spf.pokemon.mailds.jp ~all"  
"google-site-verification=Vg3Ar1_hRlrET3eyTiQS80Ntb_ijVAZh1ME3FaWX-Mw"
```

pdc.pokemon.jp の TXT レコードは複数あるようですが、SPF レコードは 1 つめの v=spf1 で始まる方です。include は引数で渡しているドメイン名の SPF レコードを含むという意味ですので、さらに spf.pdc.pokemon.jp と spf.pokemon.mailds.jp の SPF レコードを引いてみましょう。

```
$ dig spf.pdc.pokemon.jp txt +short  
"v=spf1 +ip4:203.216.217.0/24 +ip4:122.212.36.0/24 +ip4:202.8.80.0/23  
+ip4:202.74.4.160/27 +ip4:59.159.71.0/24 +ip4:220.110.139.188/32  
+ip4:211.120.127.41/32 +ip4:125.29.35.0/26 +ip4:124.211.29.64/26  
+ip4:122.215.202.64/26 +ip4:203.138.159.219 ~all"  
  
$ dig spf.pokemon.mailds.jp txt +short  
"v=spf1 ip4:118.238.144.96/29 ip4:118.238.144.120/30 ~all"
```

最初の +ip4:203.216.217.0/24 は「203.216.217.0～203.216.217.255 の IP アドレスをメールの送信元サーバとして認める」という意味です。ひとつひとつ解説するには量が多いすぎるので省略しますが、要はここに書いてある IP アドレスから届いたメールは本物で、それ以外は恐らく迷惑メールだよ、ということを示しています。実際に届いた「ポケモンだいすき！ 通信」の送信元 IP は「118.238.144.99」でしたので、きちんと SPF レコードの中に含まれており、それによってメールは迷惑メール扱いされることなく受信ボックスに届きました。

4.4.8 【ドリル】どうしてメールが迷惑メール扱いされるの？

問題

あなたは Web 制作会社 A 社のフロントエンドエンジニアです。小規模なクライアント B 社からの依頼でキャンペーンサイト (<https://campaign.example.com/>) を構築したのですが、後から「問い合わせフォームが欲しい」という追加要望が出てきました。

予算もあまりないためフォームは実装せず、問い合わせのページだけは簡単にフォームが作れる外部の ASP サービスを使うことにしました。エンドユーザがフォームから問い合わせを行うと、その ASP サービスが B 社とエンドユーザの両方に「問い合わせを受け付けました」というメールを送ってくれます。このとき送信元のメールアドレスは info@example.com です。

ところが B 社の担当者から「問い合わせ受付メールが迷惑メール扱いされて迷惑メールボックスに入ってしまう。エンドユーザの方でも同じ現象が起きているようだ」というクレームが入りました。原因を調査するにはどの dig コマンドを叩くべきでしょうか？

- A. dig example.com spf +short
- B. dig campaign.example.com txt +short
- C. dig example.com txt +short

答え _____

解答

正解は C です。送ったメールが迷惑メール扱いされてしまうときは dig ドメイン名 txt +short で SPF レコードを確認しましょう。SPF レコードには、そのドメインでのメール送信を許可されているサーバのリストが書かれています。

今回、問い合わせ受付メールが迷惑メール扱いされてしまったのは、メールを送っているサーバの IP アドレスを SPF レコードに追加し忘れたことが原因と思われます。ASP サービスにメール送信元の IP アドレスを確認して、SPF レコードに追記しましょう。

4.4.9 <トラブル> SPF を設定したのに時々迷惑メールになる

A さんは猫の譲渡会ボランティアしています。ある日、メンバーの B さんから「猫の様子をネットで公開したいから WordPress でサイトを作ってくれないか」と頼まれました。アクセス数はそろくなさそうなので、さくらインターネットが提供している「さくらの VPS（仮想専用サーバー）^{*32}」でいちばん安いプランを契約した上で、WordPress のセッティングをして B さんに引き渡しました。WordPress が動いているサーバの IP アドレスは 203.0.113.222 で、サイトの URL は <https://cat.example.com> です。

このとき WordPress から管理者宛てに送られてくるメール（送信元のメールアドレスは wordpress@cat.example.com）が迷惑メール扱いされないよう、A さんは次のように SPF レコードを設定しておきました。

^{*32} <https://vps.sakura.ad.jp/>

```
cat.example.com. 600 IN TXT "v=spf1 ip4:203.0.113.222 -all"
```

しかし数日後、B さんから「パスワードを忘れてしまって、WordPress の管理画面から新しいパスワードを発行しようとしたが、パスワードをリセットするためのメールが迷惑メールのトレイに入ってしまう」と相談されました。詳しく聞くと、毎回必ず迷惑メールになってしまふのではなく、何度か試みるとその時によって迷惑メールのトレイに入ったり受信トレイに届いたりするそうです。A さんが B さんの Gmail で迷惑メールのトレイに入ってしまった方のメールを見せてもらったところ、確かに「Received-SPF: fail」となっていました。一方、きちんと受信トレイに届いた方のメールは「Received-SPF: pass」になっていました。

何も設定を変えていないのに、なぜ迷惑メール扱いされたりされなかつたりするんだろう？ と悩みながら、A さんは 2 通のメールのヘッダをよく見てみました。すると、迷惑メールのトレイに入ってしまった方のメールは、送信元が「2001:0db8:0000:0000:0000:0000:0022」になっていました。

さくらの VPS は IPv6 に対応しているため、その時々によってメールの送信元が IPv4 の「203.0.113.222」だったり、IPv6 の「2001:0db8:0000:0000:0000:0000:0022」だったりしていたのです。それなのに SPF レコードでは「203.0.113.222 をメールの送信元サーバとして認める。それ以外は認めない」という設定になっていたため、IPv6 で送られてきたときは迷惑メール扱いされてしまっていたのでした。

A さんが SPF レコードを次のように修正^{*33}すると、迷惑メールになったりならなかつたりする事象は無事に解消しました。

```
cat.example.com. 600 IN TXT "v=spf1 ip4:203.0.113.222 ip6:2001:0db8:0000:0000:0000:0000:0022 -all"
```

4.4.10 PTR レコード

A レコードは前述のとおりドメイン名から IP アドレスを正引きできるレコードです。対して IP アドレスからドメイン名を逆引きできるレコードのことを PTR レコード^{*34}と呼びます。

メールを受信するメールサーバによっては「メール送信元の IP アドレスが SPF レコード

^{*33} ちなみに「ip6:2001:0db8:0000:0000:0000:0000:0022」の部分は省略して「ip6:2001:0db8::22」と書くことも可能ですが、ここでは分かりやすさを優先して省略せずに書いています

^{*34} PoinTeR record の略

4.4 dig を叩いてリソースレコードを確認してみよう

ドに登録されていること」だけでなく「メール送信元の IP アドレスからドメイン名の逆引きができること」という条件も満たさないと迷惑メールと判断することがあります。

PTR レコードは次の dig コマンドで確認できます。

```
$ dig -x IP アドレス +short
```

なお A レコードや MX レコードや SPF レコードといったドメイン名のリソースレコードと、PTR レコードのような IP アドレスのリソースレコードは設定依頼をする先が異なります。筆者は startdns.fun というドメイン名を持っており、ネームサーバは Route53 を使用しているので、startdns.fun の A レコードや MX レコード、SPF レコードは Route53 で設定ができます。

ですが、たとえばさくらインターネットの VPS で借りたサーバの IP アドレスが 203.0.113.222 だったとして、この IP アドレスの PTR レコードを Route53 で設定することはできません。この IP に対して PTR レコードを設定できるのは、IP アドレスの持ち主であるさくらインターネットの管理画面からとなります。

4.4.11 CNAME レコード

ときどきドメイン名から IP アドレスを引こうとして dig で A レコードを調べたのに、こんな風に別のドメイン名と IP アドレスが返ってくることがあります。

```
$ dig aibo.sony.jp a +short  
cs1018.wpc.omicroncdn.net.  
152.195.38.205
```

なぜ aibo のサイト^{*35}の A レコードを調べると cs1018.wpc.omicroncdn.net という全然関係のなさそうなドメイン名が出てくるのでしょうか？ +short オプションを外して、この結果に至るまでの過程を見てみましょう。

```
$ dig aibo.sony.jp a  
; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.68.rc1.el6_10.3 <>> aibo.sony.jp a  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 4163
```

^{*35} <https://aibo.sony.jp/>

```
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 8
;; QUESTION SECTION:
;aibo.sony.jp.          IN      A
;; ANSWER SECTION:
aibo.sony.jp.        270     IN      CNAME   cs1018.wpc.omicroncdn.net.
cs1018.wpc.omicroncdn.net. 3570 IN      A       152.195.38.205
;; AUTHORITY SECTION:
omicroncdn.net.    172770  IN      NS      ns1.omicroncdn.net.
omicroncdn.net.    172770  IN      NS      ns2.omicroncdn.net.
omicroncdn.net.    172770  IN      NS      ns4.omicroncdn.net.
omicroncdn.net.    172770  IN      NS      ns3.omicroncdn.net.
;; ADDITIONAL SECTION:
ns1.omicroncdn.net. 172770  IN      A       72.21.80.5
ns1.omicroncdn.net. 172770  IN      AAAA   2606:2800::1::5
ns2.omicroncdn.net. 172770  IN      A       72.21.80.6
ns2.omicroncdn.net. 172770  IN      AAAA   2606:2800::1::6
ns3.omicroncdn.net. 172770  IN      A       192.229.254.5
ns3.omicroncdn.net. 172770  IN      AAAA   2606:2800::e::5
ns4.omicroncdn.net. 172770  IN      A       192.229.254.6
ns4.omicroncdn.net. 172770  IN      AAAA   2606:2800::e::6
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Aug 30 00:03:49 2019
;; MSG SIZE  rcvd: 333
```

ANSWER SECTIONを見てみましょう。`aibo.sony.jp` の CNAME レコードに `cs1018.wpc.omicroncdn.net` が設定されており、さらに `cs1018.wpc.omicroncdn.net` の A レコードに `152.195.38.205` が設定されていることが分かります。このとき `aibo.sony.jp` を aliases (別名)、`cs1018.wpc.omicroncdn.net` を canonical name (正式名) と呼びます。

フルリゾルバは `aibo.sony.jp` の A レコードを調べに行って、A レコードの代わりに CNAME レコードが見つかった場合、名前解決の対象を正式名である `cs1018.wpc.omicroncdn.net` に置き換えて引き続き A レコードを調べ、最終的に `cs1018.wpc.omicroncdn.net` の A レコードで設定されている IP アドレスを返してきます。

このように CNAME レコードが設定されているときは、A レコードを問い合わせても結果として CNAME レコードと、その正式名の A レコードの両方が返ってきますが、CNAME レコードだけを調べたいときは次の dig コマンドで確認できます。

```
$ dig ドメイン名 cname +short
```

4.4.12 【ドリル】 CNAME の調べ方と使いどころ

問題

キドキド^{*36}のサイト（図 4.5）のドメイン名、kidokid.bornelund.co.jp の CNAME レコードを知りたい場合、どの dig コマンドを叩けばよいでしょうか？



▲図 4.5 キドキド (KID-O-KID) のサイト

- A. dig kidokid.bornelund.co.jp cname +short
- B. dig kidokid.bornelund.co.jp a +short
- C. dig kidokid.bornelund.co.jp txt +short
- D. dig kidokid.bornelund.co.jp mx +short

答え _____

解答

正解は A です。ただし B も CNAME レコードとその A レコードが返ってくるので、B も正解で構いません。CNAME レコードは CDN^{*37}やロードバランサーを使うときに

*36 <https://kidokid.bornelund.co.jp/>

*37 Contents Delivery Network の略。アクセスしてきたエンドユーザに最も近いサーバからサイトのコンテンツを効率的に配信できる仕組みのこと。CDN を使うとエンドユーザからのアクセスが分散されるた

よく利用されます。

```
$ dig kidokid.bornelund.co.jp cname +short  
bornelund-ELB-1960389134.ap-northeast-1.elb.amazonaws.com.
```

CDN を使う場合だけでなく、1台のウェブサーバに大量のサイトが相乗りしているような場合も CNAME レコードを使うと便利です。たとえばウェブサーバにサイト A、サイト B、サイト C…と計 100 サイトが相乗りしていてそれが A レコードを設定していた場合、ウェブサーバを引っ越すとなったら 100 件の A レコードを書き換えなければなりません。ですが、A レコードを設定しているのはサイト A のみで、それ以外のサイトは CNAME でサイト A のドメイン名を指定するという方法にしておけば、サーバ引っ越しに際して書き換えなければならないのはサイト A の A レコードだけとなります。

4.4.13 CNAME と他のリソースレコードは共存できない

一見便利な CNAME レコードですが使用する際は注意点があります。それは「**CNAME レコードを設定したら、他のリソースレコードは設定できない**」ということです。

たとえば次のような CNAME レコードと MX レコードは共存ができません^{*38}。

```
campaign.example.com. IN CNAME cdn.example.jp.  
campaign.example.com. IN MX mail.example.com.
```

campaign.example.com の CNAME レコードで cdn.example.jp を設定すると、A レコードだけでなく MX レコードも TXT レコードも NS レコードも、ありとあらゆるリソースレコードが cdn.example.jp を参照しに行ってしまいます。そのため 2 行目で campaign.example.com の MX レコードで mail.example.com を設定しても、その MX レコードは名前解決に使われることなく無視されてしまうのか、あるいは使われるのか動作がまったく保証されません。

このような理由から Route53 をはじめとするネームサーバのサービスでは、CNAME レコードを設定した場合は他のリソースレコードが設定できないようになっています。

め、TV や Yahoo!ニュース、LINE のプッシュ通知などで一気にアクセスが殺到してもサイトが落ちたり重くなったりしないで済みます

*38 A CNAME record is not allowed to coexist with any other data. <http://www.ietf.org/rfc/rfc1912.txt>

また「ありとあらゆるレコード」には CNAME レコードも含まれるため、次のように CNAME レコードを複数設定することもできません。

```
campaign.example.com.    IN    CNAME    cdn1.example.jp.  
campaign.example.com.    IN    CNAME    cdn2.example.jp.
```

ZONE APEX は CNAME を使えない

前述の「CNAME レコードは他のリソースレコードと共存できない」という理由から、ZONE APEX^{*39}では CNAME を設定することができません。筆者が CDN を使いたいと思っても、次のような CNAME レコードは設定できないのです。

```
startdns.fun.    IN    CNAME    cdn.example.jp.
```

なぜならば CNAME レコードが他のリソースレコードと共存できないのに対して、ZONE APEX には「このドメイン名はこのネームサーバを使うよ」という NS レコード、および「このドメイン名の管理情報はこれだよ」という SOA レコードが必ず存在するからです。（お名前.com で自分のドメイン名を買った後、Route53 でホストゾーンを作成したら SOA レコードと NS レコードが自動生成されていたのを覚えていますか？）

Route53 の Alias レコードなら ZONE APEX でも設定可能

前述の「CNAME レコードは他のリソースレコードと共存できない」「ZONE APEX には NS レコードと SOA レコードが必ず存在する」という 2 つの制限から、たとえ CDN や ELB^{*40}を使いたいと思っても ZONE APEX では使用できなかったのですが、なんと Route53 の Alias レコードという独自拡張を使うと ZONE APEX でも CDN を使うことができるのです。^{*41}

しかも Alias レコードには「ZONE APEX でも使える」だけでなく、「CNAME と違って名前解決が 1 回で済む」という利点があります。

たとえば startdns.fun というドメイン名に紐づく IP アドレスを調べようとしたとき、CNAME レコードの場合は「startdns.fun の CNAME レコードは cdn.example.jp で、

^{*39} startdns.fun や example.jp のように www や stg といったサブドメインを含まないドメインのこと。レジストラやリセラで買ったいちばん短い表記のドメイン名のことを ZONE APEX と呼びます。Apex Domain や Naked Domain、ホスト名なしドメインなどと呼ばれることもあります

^{*40} Elastic Load Balancing の略。AWS のサービスの 1 つでいわゆるロードバランサーのことです

^{*41} Route53 の Alias の他に CloudFlare の CNAME Flattening など類似のサービスはいくつかあります

cdn.example.jp の A レコードは 203.0.113.222」のように名前解決が 2 回発生します。対して Route53 の Alias レコードで「startdns.fun のエイリアス先は cdn.example.jp である」という設定をしておけば、フルリゾルバが startdns.fun の A レコードを問い合わせに来たら「startdns.fun の A レコードは 203.0.113.222」のように一発で IP アドレスを返すので名前解決は 1 回で済みます。

ネームサーバで Route53 を使っていても、参照先の CDN やロードバランサーが AWS 外なのであれば CNAME を使うしかありませんが、参照先が AWS 内のサービスであれば Alias レコードを使わない手はありません。ZONE APEX に限らず積極的に使いましょう。

4.4.14 グルーレコード

ところでドメイン名を買ったとき、お名前.com のネームサーバや Route53 を使う他に自力でネームサーバを立てて使うこともできます。Linux サーバを立てて Apache をインストールすればウェブサーバになるように、Linux サーバを立てて BIND をインストールすればもうそれは立派なネームサーバです。

たとえば筆者が startdns.fun というドメイン名を買って、自分で作ったネームサーバに ns1.startdns.fun という名前を付け、startdns.fun の NS レコードに ns1.startdns.fun を設定したとします。このとき、ブラウザで <http://startdns.fun/> を開こうとすると次のようになります。

1. フルリゾルバに startdns.fun に紐づく IP アドレスを聞きに行く
2. フルリゾルバは自分の中に startdns.fun に紐づく IP アドレスのキャッシュがないことを確認
3. フルリゾルバはルートネームサーバに startdns.fun に紐づく IP アドレスを聞きに行く
4. ルートネームサーバは「.fun のネームサーバは a.nic.fun だよ」と教えてくれる
5. フルリゾルバは a.nic.fun に startdns.fun に紐づく IP アドレスを聞きに行く
6. a.nic.fun は「startdns.fun のネームサーバは ns1.startdns.fun だよ」と教えてくれる
7. フルリゾルバはルートネームサーバに ns1.startdns.fun に紐づく IP アドレスを聞きに行く
8. ルートネームサーバは「.fun のネームサーバは a.nic.fun だよ」と教えてくれる
9. フルリゾルバは a.nic.fun に ns1.startdns.fun に紐づく IP アドレスを聞きに行く
10. a.nic.fun は「startdns.fun のネームサーバは ns1.startdns.fun だよ」と教えてく

れる

11. ns1.startdns.fun の IP アドレスを知っているのが ns1.startdns.fun なのでフルリグルバはいつまでも ns1.startdns.fun にたどり着けない

「田中さんの住所は田中さんの家に行って田中さんに聞いて」みたいなもので、このままだといつまでも startdns.fun に紐づく IP アドレスが分かりません。

これを解消するため、自分でネームサーバを立てるときは上位のネームサーバ（ここでは a.nic.fun のこと）に、ネームサーバの IP アドレスも一緒に登録しておかなければいけません。これをグルーレコードと呼びます。上位の a.nic.fun から ns1.startdns.fun への道をちゃんと繋げてくれるレコードだから glue（接着剤のこと）レコードなんですね。

お名前.com であればドメイン Navi の中に「ネームサーバー名としてのホストを設定する」という設定画面がありますので、そこで自分が立てたネームサーバのドメイン名(ns1.startdns.fun) とその IP アドレス(203.0.113.222) を登録すれば OK です。これでいつまでも ns1.startdns.fun にたどり着けない無限ループが回避できるようになります。

【コラム】この IP アドレスはグローバル IP？ プライベート IP？

ところで dig コマンドを叩いて「203.0.113.222」という応答が返ってきたとき、この IP アドレスがグローバル IP アドレス^{*42}なのか、プライベート IP アドレス^{*43}なのか、ぱっと見て判別できますか？

グローバル IP は「03-〇〇〇〇-〇〇〇〇」のような外線番号で、プライベート IP は「法務部の片岡さんは 4322 で、総務部の茂木さんは 2652」のような内線番号だと思ってください。A 社から B 社に電話をかけるときは外線番号でないと繋がりませんが、A 社のオフィス内で片岡さんが茂木さんにかけるときは内線番号を押せば繋がります。これと同じように、インターネットを介して通信するときはグローバル IP を使いますが、オフィスのように同じネットワーク内であればプライベート IP で通信ができます。

そして全世界のどこを探しても同じ電話番号を使っている人がいないのと同じように、インターネット上でグローバル IP は一意です。対してプライベート IP は同一ネットワーク内でのみ一意です。A 社で法務部の片岡さんが内線番号 4322 を使っているときに、まったく別の会社で 4322 という内線番号を使っていても何の問題もないですよね。

そして実はプライベート IP アドレスの範囲は RFC1918^{*44}で次のように定められています。

- 10.0.0.0/8 (10.0.0.0～10.255.255.255)
- 172.16.0.0/12 (172.16.0.0～172.31.255.255)
- 192.168.0.0/16 (192.168.0.0～192.168.255.255)

ですので、dig コマンドを叩いて IP アドレスが返ってきた場合、ぱっと見てこの範囲外であれば「これはグローバル IP だな！」と判断できます。203.0.113.222 もプライベート IP アドレスの範囲に含まれないのでグローバル IP ですね。

*⁴² グローバル IP は AWS 上だとパブリック IP という名前で呼ばれています

*⁴³ プライベート IP はローカル IP と呼ばれることもあります

*⁴⁴ <https://tools.ietf.org/html/rfc1918#section-3>

第5章

トラブルシューティング

ドメイン名や DNS の周りには「知ってさえいればすぐ解決できるけど、知らないとどうにもならない」という落とし穴がいくつかあります。この章ではそんなありがちトラブルとその解決方法、あるいは未然に防ぐための対策をご紹介していきます。

5.1 <トラブル> URL は www ありなしどっち？

サイトのドメイン名を www ありかなしかはっきり決めておかなかったことで、商品パッケージでは www なし、店頭で配るチラシには www ありの URL を記載してしまい、www ありとなしの URL があちこちで混在してしまった…というトラブルはよくあります。

「え、別に www あるかないかくらい些細なことじゃないの？ 何かだめなの？」と思われるかも知れませんが、これは意外と重要な問題です。

そもそもドメイン名を www.example.co.jp のように頭（この例だと第 4 レベルドメイン）から最後（トップレベルドメイン）までしっかり全部書いたものを FQDN^{*1}と呼びます。

- www : 第 4 レベルドメイン
- example : 第 3 レベルドメイン
- co : 第 2 レベルドメイン
- jp : トップレベルドメイン

startdns.fun と www.example.co.jp はまったく別の FQDN です。それと同じように www.example.co.jp と example.co.jp もまったく別の FQDN です。

名前が途中まで一緒だとなんだか近いような気がしてしまいますが、北海道と奄美大島がまったく別の場所であるのと同様に、地名が似ていても奄美大島と大島もまったく別の場所^{*2}ですよね。

たとえば実際に用意したウェブサイトは www.example.co.jp なのに、LINE のプッシュ通知で「今すぐ example.co.jp にアクセス！」と告知してしまったとします。これはイベント会場が奄美大島にあるのに「大島（東京）でイベントやります！」と告知してしまったようなものなので、LINE を見たユーザが URL をクリックして example.co.jp にアクセスしてもこのままではサイトは表示されません。

気の利いた Web 制作会社であれば、依頼側が何も言わなくてもドメイン名やリダイレクトの設定をしておいてくれて、example.co.jp を訪れたユーザを取りこぼさずに www.example.co.jp にリダイレクトさせておいてくれるかも知れませんが、そうでなければ折角の告知が台無しです。（「リダイレクトの設定」というのは、大島に「奄美大島への自動転送装置」を設置しておくようなものだと思ってください）

^{*1} 第1章「ドメイン名と Whois」でも出てきましたが、FQDN は Fully Qualified Domain Name の略で、日本語にすると完全修飾ドメイン名です

^{*2} 大島は伊豆諸島にあり東京から船で 30 分、対して奄美大島は沖縄の近くにあります

また「うちのサイトは www ありとなしのどちらでアクセスしても同じ画面が出るよ！リダイレクトはされないけどね」という場合もやはり問題です。検索エンジンは FQDN が異なるサイトをまったく別のサイトとして認識するため、www ありのサイトとなしのサイトで価値が分散されてしまい、その結果として検索時に上位表示されにくくなります。

サイトを作るときは、最初に FQDN を www ありなのかなしなのかはっきり決めましょう。もし既にばらばらの状態でサイトが公開されてしまっていたら、今からでも「メインを www ありにする」のように決めて、片方にちゃんとアクセスを寄せるようにしましょう。www ありでもなしでも正直どっちでもいい、という場合は、前述の「ZONE APEX では CNAME が使えない」問題があるため、将来 CDN を使いたくなつた時に弊害がないよう www ありにしておくのがお勧めです。

5.2 <トラブル> .dev で終わるテストサイトが見られなくなった

2017 年 12 月、.dev で終わるドメイン名のサイトを Chrome で開くと、強制的に HTTP から HTTPS にリダイレクトされるようになり、あちこちで「テストサイトが見られなくなった！」「開発環境が急に使えなくなった！」という悲鳴が上がりいました。

もともと gTLD は.com や.net などの 22 種類しかなく、.dev という TLD は存在しなかつたため、IT 系の各社が組織内のネットワーク（開発環境など）でオレオレ TLD として勝手に.dev を使っていた、という背景がありました。しかし gTLD の種類が段々増えてきてとうとう.dev という TLD もできてしまい、色々なところで「まずいぞ、名前衝突（Name Collision）する…」という話題が出たのが 2014 年ごろのことでした。

ちなみに余談ですが ICANNWiki^{*3}によると、この.dev は当初 Amazon と Google で「俺がレジストリになる！」「いいや、俺がなる！」と奪い合ってたけれど、.you と.talk を Amazon に譲って、代わりに.dev を Google がゲット、というような経緯があったようです。

詳しくは第 1 章「ドメイン名と Whois」でお話ししたとおりですが、レジストリとはその TLD の唯一の卸元ですので、あちこちで「Google が.dev を買った」と言われているのは、この「Google が.dev のレジストリになったこと」を指しているものと思われます。

そして Google が「Chrome で.dev を開くと強制的に HTTPS にリダイレクトする」という暴挙に出たのは「Google は 2014 年に dev のレジストリ（生産元）になった。でもまだ 2017 年 12 月の時点では.dev のドメイン名は一般発売していない。ということはその

^{*3} <https://icannwiki.org/.dev>

時点での.dev を使っている人は全員、ドメイン名を買わずに勝手に使ってる人なので迷惑が掛かっても知ったことではない。だから.dev は HTTPS に強制リダイレクトさせるね」ということだったのだと推察しています。その後、2019年2月に.dev のドメイン名は登録が開始されました。

このトラブルについては「.dev が使えなくなったからとりあえず.test にした！ 解決！」のような記事も見ますが、なんで.test にしたら直ったのか？ .test でいいならたとえば.dev2 でもいいのか？ といった根本原因と解決策が分かっていないとまた同じトラブルが起きてしまいます。

ざっくり言うと、.test はいいけれど.dev2 はだめです。社内ネットワークにしてもテスト用のメールアドレスにしても第4章「dig と whois を叩いて学ぶ DNS」で書いたとおり例示やテストで使っていいドメイン名が用意されているので、基本的にはそれを使いましょう。「自分の持ち物でないドメイン名」を勝手に使うのは、今回のようなトラブルの元です。

5.3 サイト移管の AtoZ

自社サイト (example.net) は今まで Web 制作会社の A 社にお任せしていたが、サイトリニューアルを機に運用を A 社から B 社に移管することになった。でも移管と言っても何をすればいいのかふんわりしていてよく分からない…。そんなときは以下のような移管前後の表（表5.1）を作って埋めていきましょう。

▼表 5.1 サイト移管前後表

	移管前	移管後
1. ドメイン名の管理	A 社	B 社
2. お店（レジストラ・リセラ）	お名前.com	お名前.com
3. ネームサーバ	お名前.com	Route53
4. ウェブサーバ	A 社のクラウドサーバ (203.0.113.111)	EC2 (203.0.113.222)

表5.1 のようにドメイン名の管理は A 社から B 社に変わるので、レジストラはお名前.com のまま変わらないのであれば、お名前.com の「お名前 ID 付け替え」という機能で A 社のお名前 ID から B 社のお名前 ID にドメイン名を移動することができます。

ドメイン名の管理が B 社に移ったらお名前.com のネームサーバにあったリソースコードを Route53 にコピーして、ドメイン Navi (お名前.com の管理画面) でネームサーバを Route53 に変更します。仮にサイトリニューアルが 5月15日だったとしたら、これらの作業は 4月中に済ませておきましょう。この時点ではまだ移管前のウェブサーバを

使っていきます。

EC2 上でリニューアル後のサイトが完成してテストも完了したら、5月 15 日に B 社が Route53 で example.net の A レコードの値を 203.0.113.111 から 203.0.113.222 に書き換えてやればサイト移管は完了です。

このようにドメイン名の管理→ネームサーバ→ウェブサーバの順で移管することで、移管元の A 社の負担が少なくなり、移管先である B 社が主体となって移管作業を進められるようになります。逆の順番（ウェブサーバ→ネームサーバ→ドメイン名の管理）で移管を進めようとすると、5月 15 日の A レコード書き換えも B 社から A 社に頼まなければならぬし、お名前.com のネームサーバにあるリソースレコードの情報もいちいち A 社が B 社に渡してあげないといけません。

5.3.1 【ドリル】リニューアル後のサイトが人によって表示されない

問題

ドメイン名の管理やネームサーバの変更は 4 月上旬に済ませたし、EC2 上で動いている新サイトもテストはばっちりです。5 月 15 日のサイトリニューアル当日、B 社の C さんは Route53 で example.net の A レコードの値を 203.0.113.111 から 203.0.113.222 に書き換えた後、手元のスマホでサイトを開いてリニューアル後のページが正しく表示されることを確認してからクライアントの D 社に「リリース完了」の電話をしました。ですが電話の向こうのクライアントは「え、古いサイトが表示されるんですけど…？」と困惑気味です。

慌ててパソコンのブラウザでサイトを見ると、確かにリニューアル前の古いサイトが表示されました。別のオフィスにいるデザイナーにチャットで「ちょっとサイト見てくれない？ 古いやつが出てる？」と聞いたところ「こっちではちゃんと新しいサイトが表示されてますよ？」と返事が来ました。

先ほど C さんが書き換えたリソースレコードは以下のようになっていました。

変更前	
example.net.	604800 IN A 203.0.113.111
変更後	
example.net.	300 IN A 203.0.113.222

人や端末によって古いサイトが表示されたり、リニューアル後の新しいサイトが表示されたりする現象の原因は次のどれでしょうか？

- A. 変更前の TTL (604800) が長いことが原因

- B. 変更後の TTL（300）が短いことが原因
- C. 変更後の IP アドレス（203.0.113.222）が間違っていることが原因

答え _____

解答

正解は A です。TTL は Time To Live の略でキャッシュ保持時間のことです。TTL に書いてある秒数が過ぎるまで、フルリゾルバにはこのリソースレコードがキャッシュとして残ります。

example.net の変更前の A レコードは TTL が 604800 秒、つまり 7 日間になっているため、C さんが A レコードを書き換えてからも最大 7 日間はフルリゾルバに古い IP アドレスがキャッシュされており、そのフルリゾルバを使っているユーザには古いサイトが表示されてしまうのです。

人や端末によって異なるフルリゾルバを使っているため、C さんのスマホでは新しいサイトが見られますが、クライアント D 社のパソコンではまだ古いサイトが表示されてしまう、といった現象が起こります。古いウェブサーバでリダイレクトの設定をすればいいんじゃない？と思われるかもしれません、古いウェブサーバに来たアクセスを example.net にリダイレクトしたところでまた古いウェブサーバにリクエストが飛んでくるだけですので、何の意味もないどころか無限リダイレクトでブラウザがエラーになってしまいます。

キャッシングが消えるまでの時間を逆算して考えると、そもそも C さんはリニューアル日である 5 月 15 日の 8 日以上前に example.net の A レコードの TTL を 300 程度まで短くしておいて、当日 IP アドレスを書き換えたたらすぐに反映されるようにしておくべきでした。

どうしてもキャッシングを今すぐ消したい！ という場合、もし D 社の情シスが自社のフルリゾルバを管理しているのであれば「フルリゾルバのキャッシングをクリアして！」と頼めばキャッシングがなくなって、ネームサーバへ改めてリソースレコードを問い合わせに行ってくれますが、それでリニューアル後のサイトが表示されるようになるのは D 社内だけの話です。世界中にあるフルリゾルバすべてに対してキャッシングクリアをお願いしてまわるわけにはいかないので、クライアントの D 社も C さんもキャッシングが消えるまで大人しく 7 日待つしかありません。

事前に TTL を確認した上で短くしておくことで、いわゆる「浸透」を無為に待たなく

て済みますし、TTL を過ぎても新サイトに切り替わらなければ何か他に問題があるので
は？ と気づくことができます。サイトリニューアル時は変更対象のリソースレコードの
TTL を事前に確認しておきましょう。

5.4 <トラブル> サブドメインを追加したのにサイトが見 られない

たとえば次のように 3 つの A レコードがあったとします。

```
example.net.      300    IN  A    203.0.113.222
www.example.net. 600    IN  A    203.0.113.222
stg.example.net. 900    IN  A    203.0.113.222
```

この状態で example.net の A レコードを dig で引いてみると、TTL が 300 なのでフル
リゾルバにはキャッシュが 5 分間保持されます。www.example.net なら TTL が 600 な
ので 10 分間、stg.example.net なら 15 分間保持されます。

ですが、存在しない test.example.net の A レコードを dig で引いたら「そんなレコー
ドなかったよ」というキャッシュはフルリゾルバに残るのでしょうか？ そして残るとし
たらキャッシュ保持時間は何秒になるのでしょうか？

実際に自分のドメイン名で試してみましょう。筆者は startdns.fun を使いますので、あ
なたもお名前.com で買ったドメイン名を使って試してみてください。

まずはリソースレコードが存在しない test.startdns.fun を dig で引いてみましょう。
status が「そのドメイン名のリソースレコードは存在しない」という意味の NXDOMAIN
になって返ってきます。該当するレコードが存在しないため ANSWER SECTION は存
在せず、代わりにドメイン名の管理情報を表す SOA レコードが付加情報として返ってき
ました。

```
$ dig test.startdns.fun a
; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.el6_9.5 <>> test.startdns.fun a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 16872
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;
;; QUESTION SECTION:
;test.startdns.fun.          IN      A
;
;; AUTHORITY SECTION:
```

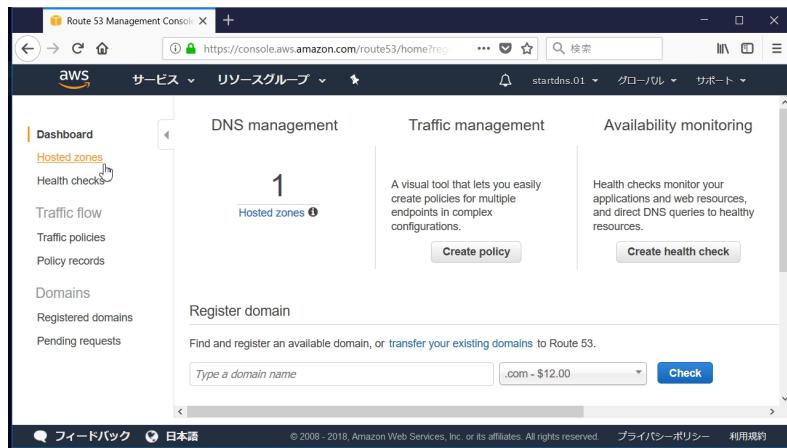
```
startdns.fun.          900      IN      SOA
    ns-943.awsdns-53.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400

    ; Query time: 139 msec
    ; SERVER: 127.0.0.1#53(127.0.0.1)
    ; WHEN: Wed Mar 21 17:59:28 2018
    ; MSG SIZE  rcvd: 119
```

このとき「そんなレコードは存在しない」というネガティブキャッシュがフルリゾルバに残ります。ネガティブキャッシュの保持時間は SOA レコードの TTL、もしくは SOA の MINIMUM (Negative cache TTL) の値のいずれか小さい方となります。

今回の test.startdns.fun の場合、SOA レコードの TTL は 900 で、SOA の MINIMUM は 86400 ですので、ネガティブキャッシュの保持時間は 900 秒 (15 分) となります。ではネガティブキャッシュが残っている 15 分の間に Route53 で test.startdns.fun の A レコードを作ってみましょう。

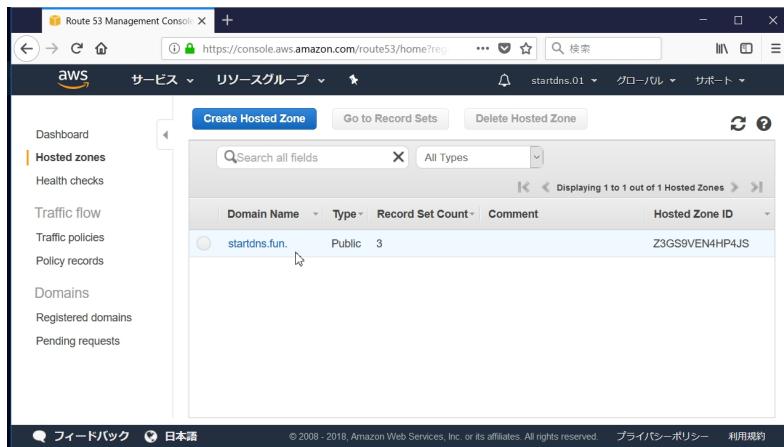
<https://aws.amazon.com/> からマネジメントコンソールにログインしたら、左上の「サービス」を押して Route53 を選択します。Route53 の画面 (図 5.1) を開いたら左メニューの Hosted Zones を選択します。



▲図 5.1 左メニューの Hosted Zones を選択

表示されている自分のドメイン名（筆者なら startdns.fun）を選択（図 5.2）します。

5.4 <トラブル> サブドメインを追加したのにサイトが見られない



▲図 5.2 自分のドメイン名を選択

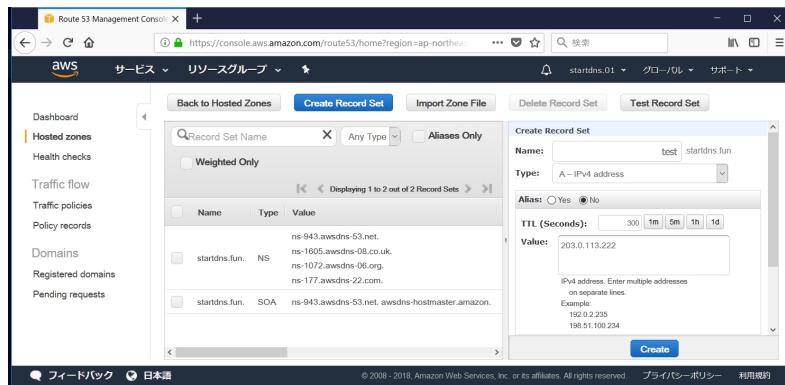
上部にある青い「Create Record Set」を押したら、右側の Name に test、Value に 203.0.113.222^{*4}と入力して右下の青い「Create」を押します。

▼表 5.2 レコード作成時の設定

項目	入力するもの
Name	test
Value	203.0.113.222

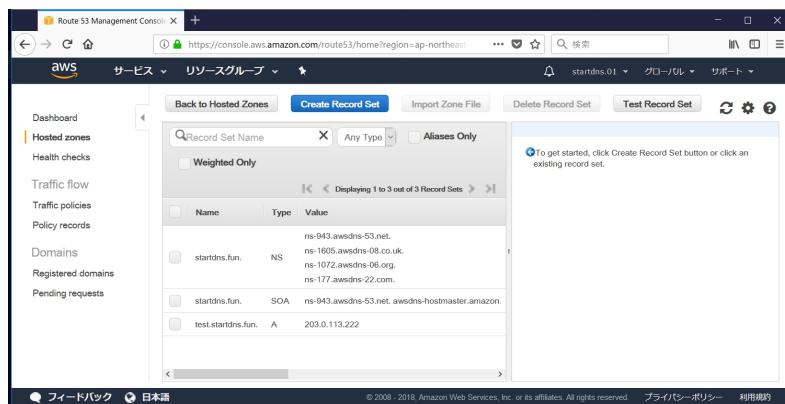
^{*4} この IP アドレスは例示用として定められている IP アドレスです。<https://tools.ietf.org/html/rfc5737>

第5章 トラブルシューティング



▲図 5.3 Name と Value を記入したら「Create」を押す

これで test.startdns.fun の A レコードが出来上がりました。(図 5.4)



▲図 5.4 test.startdns.fun の A レコードが出来た

それでは再度 test.startdns.fun を dig で引いてみましょう。今さっき A レコードを作ったので、今度は存在しているはずです。

```
$ dig test.startdns.fun a
; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.el6_9.5 <>> test.startdns.fun a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 2854
```

5.4 <トラブル> サブドメインを追加したのにサイトが見られない

```
; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;test.startdns.fun.      IN      A
;; AUTHORITY SECTION:
startdns.fun.          759      IN      SOA
    ns-943.awsdns-53.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Mar 21 18:01:49 2018
;; MSG SIZE  rcvd: 119
```

A レコードは存在しているのに再度 NXDOMAIN が返ってきました。先ほど「そのドメイン名のリソースレコードは存在しないよ」と言わされてから、まだ 900 秒（15 分）が経過していないためネガティブキャッシュが返ってきたようです。このとき SOA レコードの TTL を見ると 759 と表示されています。これは SOA レコードの残りキャッシュ保持時間が 759 秒であることを示しています。

ではフルリゾルバの中にあるネガティブキャッシュを返すのではなく、もう一度ルートネームサーバから順にちゃんと聞きに行ってもらいましょう。dig コマンドに +trace オプションを付けて叩いてみます。

```
$ dig test.startdns.fun a +trace
; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.el6_9.5 <>> test.startdns.fun a +trace
;; global options: +cmd
.          514824  IN      NS      m.root-servers.net.
.          514824  IN      NS      e.root-servers.net.
.          514824  IN      NS      d.root-servers.net.
.          514824  IN      NS      g.root-servers.net.
.          514824  IN      NS      f.root-servers.net.
.          514824  IN      NS      a.root-servers.net.
.          514824  IN      NS      l.root-servers.net.
.          514824  IN      NS      k.root-servers.net.
.          514824  IN      NS      b.root-servers.net.
.          514824  IN      NS      h.root-servers.net.
.          514824  IN      NS      c.root-servers.net.
.          514824  IN      NS      j.root-servers.net.
.          514824  IN      NS      i.root-servers.net.
;; Received 508 bytes from 127.0.0.1#53(127.0.0.1) in 1682 ms

fun.          172800  IN      NS      b.nic.fun.
fun.          172800  IN      NS      c.nic.fun.
fun.          172800  IN      NS      d.nic.fun.
fun.          172800  IN      NS      a.nic.fun.
;; Received 279 bytes from 2001:7fe::53#53(2001:7fe::53) in 1538 ms

startdns.fun.      3600      IN      NS      ns-1605.awsdns-08.co.uk.
startdns.fun.      3600      IN      NS      ns-177.awsdns-22.com.
```

```
startdns.fun.      3600   IN      NS      ns-1072.awsdns-06.org.
startdns.fun.      3600   IN      NS      ns-943.awsdns-53.net.
;; Received 175 bytes from 108.59.161.12#53(108.59.161.12) in 74 ms

test.startdns.fun. 300    IN      A       203.0.113.222
startdns.fun.     172800  IN      NS      ns-1072.awsdns-06.org.
startdns.fun.     172800  IN      NS      ns-1605.awsdns-08.co.uk.
startdns.fun.     172800  IN      NS      ns-177.awsdns-22.com.
startdns.fun.     172800  IN      NS      ns-943.awsdns-53.net.
;; Received 191 bytes from 205.251.198.69#53(205.251.198.69) in 99 ms
```

+trace を付けたらちゃんと「test.startdns.fun に紐づく IP アドレスは 203.0.113.222 です」という返答が返ってきました。でも +trace を付けて引いた結果はフルリゾルバのキャッシュを上書きしないため、+trace を消すとやはり NXDOMAIN の状態に戻ります。ネガティブキャッシュの残り時間はあと 334 秒です。

```
$ dig test.startdns.fun a

; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.el6_9.5 <>> test.startdns.fun a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 19871
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;test.startdns.fun.           IN      A

;; AUTHORITY SECTION:
startdns.fun.      334    IN      SOA
    ns-943.awsdns-53.net. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Mar 21 18:08:54 2018
;; MSG SIZE  rcvd: 119
```

さらに 5 分ほど待ってようやくネガティブキャッシュの保持時間が過ぎたら、もう一度 test.startdns.fun を dig で引いてみましょう。status は NXDOMAIN から NOERROR に変わり、ANSWER SECTION のところに Route53 で設定した A レコードが表示されました。

```
$ dig test.startdns.fun a

; <>> DiG 9.8.2rc1-RedHat-9.8.2-0.62.rc1.el6_9.5 <>> test.startdns.fun a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64301
```

5.4 <トラブル> サブドメインを追加したのにサイトが見られない

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 8
;; QUESTION SECTION:
;test.startdns.fun.      IN      A
;; ANSWER SECTION:
test.startdns.fun.      300     IN      A      203.0.113.222
;; AUTHORITY SECTION:
startdns.fun.          3599    IN      NS      ns-1605.awsdns-08.co.uk.
startdns.fun.          3599    IN      NS      ns-177.awsdns-22.com.
startdns.fun.          3599    IN      NS      ns-1072.awsdns-06.org.
startdns.fun.          3599    IN      NS      ns-943.awsdns-53.net.
;; ADDITIONAL SECTION:
ns-1072.awsdns-06.org. 168458  IN      A      205.251.196.48
ns-1072.awsdns-06.org. 168458  IN      AAAA   2600:9000:5304:3000::1
ns-1605.awsdns-08.co.uk. 168458 IN      A      205.251.198.69
ns-1605.awsdns-08.co.uk. 168458 IN      AAAA   2600:9000:5306:4500::1
ns-177.awsdns-22.com.   168459  IN      A      205.251.192.177
ns-177.awsdns-22.com.   168459  IN      AAAA   2600:9000:5300:b100::1
ns-943.awsdns-53.net.   168459  IN      A      205.251.195.175
ns-943.awsdns-53.net.   168459  IN      AAAA   2600:9000:5303:af00::1
;; Query time: 163 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Mar 21 18:19:03 2018
;; MSG SIZE rcvd: 367
```

新たにサイトをオープンするとき、まだリソースレコードが用意できていないのにブラウザでサイトを開いてしまい、リソースレコードが用意できた後で再度確認しようとしたらなぜか見られない…という現象はこのネガティブキャッシュが原因です。「インフラ担当に頼んでおいた A レコード、もう用意できたかな？」と確認するときは、フルリゾルバに問い合わせてネガティブキャッシュが残ってしまうことのないよう、次のように⁶対象ネームサーバを指定して直接問い合わせる⁵方法がお勧めです。

1. 先ずは自分のドメイン名のネームサーバを確認する
\$ dig startdns.fun ns +short
ns-1605.awsdns-08.co.uk.
ns-177.awsdns-22.com.
ns-1072.awsdns-06.org.
ns-943.awsdns-53.net.
2. そのネームサーバに対して直接問い合わせる
\$ dig test.startdns.fun a +norecurse +short @ns-1605.awsdns-08.co.uk

⁵ dig コマンドはデフォルトで「再帰的な名前解決の要求」が有効になっています。しかしネームサーバに対して直接問い合わせるときは、フルリゾルバのようにあちこち聞き回ってきてほしい訳ではなく、単に自分自身が知っていることを答えて欲しいだけなので、「+norecurse」というクエリオプションを付けて「再帰的な名前解決の要求」を無効にしています

(問い合わせに対して答えが何も返ってこなければ「まだ設定されていない」と分かる)

5.5 <トラブル> CAA レコードが原因で SSL 証明書が発行できなかった

最近、SSL 証明書を取得するときに「CAA レコード」という言葉を聞くようになってきました。

CAA レコードの CAA は Certification Authority Authorization の略で、CAA レコードに「そのドメイン名の証明書を発行できる認証局」を書いておくことで、意図しない認証局による証明書の発行を防ぐ仕組みになっています。

たとえば毎年 DigiCert で startdns.fun の SSL 証明書を発行をしており、それ以外の認証局に発行を依頼する予定はない、という場合は startdns.fun の CAA レコードで digicert.com を指定しておくことで、DigiCert 以外の認証局に対して証明書の発行依頼があっても認証局は証明書を発行せず、ドメイン管理者に「意図しない発行申請があったこと」を報告してくれます。

CAA レコードは必須ではありませんので、CAA レコードを設定していなければ従来通り証明書の発行に際してチェックや制限は一切行われません。ただし CAA レコードが見つからない場合の挙動には注意が必要です。

たとえば www.example.co.jp の証明書を取ろうとした場合、最初は www.example.co.jp の CAA レコードがないか確認し、なければ example.co.jp の CAA レコードを確認、それもなければ co.jp の CAA レコード、最終的には jp の CAA レコードまでさかのぼって順に確認していきます。⁶

そのため証明書を発行しようとした FQDN では CAA レコードを設定していなかったのに、その親ドメインで CAA レコードを設定していたため、予期しない形で証明書の発行が制限されてしまった、というトラブルが予想されます。SSL 証明書の発行が思いがけず失敗してしまったときは、そのドメイン名の TLD まで順にさかのぼって CAA レコードが設定されていないか確認してみるのがよいでしょう。

⁶ ちなみに JPRS では co.jp や jp の CAA レコードは作成しておらず、当面作成する予定もないそうです

5.6 <トラブル> AWS で突然ドメイン名が引けなくなった

AWS の EC2 でインスタンスを立てると、自動的にネットワーク内に Amazon DNS^{*7}と呼ばれるフルリゾルバが用意されます。Amazon DNS はフルリゾルバ、Route53 はネームサーバですのでまったく別物です。

この Amazon DNS は、次のように若干変わった挙動をするので注意が必要です。

5.6.1 レートリミットを超えると NXDOMAIN を返す

Amazon DNS に対する名前解決の問い合わせのパケット数は「ネットワークインターフェイスあたり最大 1024 パケット/秒」に制限されています。つまり EC2 のインスタンスで大量に dig を叩いて、パケット数が秒間 1024 パケットを超えると、制限に引っかかってきちんと応答が返ってこなくなるのです。^{*8}

5.6.2 勝手に TTL を短くする

AWS の EC2 で dig コマンドを叩いてみたところ、TTL を 1000 にしているはずのリソースレコードで TTL が 60 になって返ってきました。公式ドキュメントでは記載がないのですが、リソースレコードの TTL が 60 秒以上の場合、Amazon DNS はすべて 60 秒にして返す^{*9}ようです。(逆にリソースレコードの TTL が 60 秒より短い場合はそのままです)

ネームサーバが死んでしまったときのために TTL を長くしておいたつもりでも、DNS に問い合わせる側が AWS の場合は他の環境より早くキャッシュが切れて異なる挙動になると思われますので、注意が必要です。

^{*7} 正確にはインスタンスを立てたらではなく、VPC を作成すると Amazon DNS は VPC の範囲で第 4 オクテットを 2 にした IP アドレスで自動的に作成されます。たとえば VPC の IPv4 ネットワークの範囲が 172.31.0.0/16 の場合、Amazon DNS の IP アドレスは 172.31.0.2 となります

^{*8} <https://dev.classmethod.jp/cloud/aws/amazon-dns-threshold-exceeded-action/>

^{*9} <https://dev.classmethod.jp/cloud/aws/dig-route53-begginer/>

5.7 <トラブル> ラブライブ！ のオフィシャルサイトが乗っ取られた

2019年4月5日、「ラブライブ！」^{*10}のオフィシャルサイトが乗っ取られる事件がありました。^{*11}事象発生時、ブラウザでサイト (<http://www.lovelive-anime.jp/>) を開くと、可愛いスクールアイドルたちではなく、こんな画面^{*12}が表示されていました。(図5.5)

ラブライブは我々が頂いた！

我々がラブライブを入手する際、
手の込んだプログラミングを行なったり、
こっそりとデータを傍受したりする必要はなかった

我々の方法は、移管オファーを行い元所有者が移管オファーを承認しただけだった
元所有者はこれだけであっさりと、ラブライブを、我々へと移管してしまった

このサイトは60秒で艦隊これくしょん-艦これに転送されます
<http://kancole-anime.jp/>

幅ないッスね

by.キュアメロディ好きのポケモントレーナー

▲図5.5 乗っ取られたラブライブ！ のオフィシャルサイト

これは「ウェブサーバに侵入されてコンテンツを差し替えられた」のではなく、攻撃者が `lovelive-anime.jp` というドメイン名をレジストラの A 社から B 社へ移管することによって成立した乗っ取りのようです。

事象発生当時のツイート(図5.6)^{*13}に貼られていた Whois 情報などを見ると、B 社は恐らくネットオウル株式会社が運営するスタードメインと思われます。攻撃者はスタード

*¹⁰ アニメ制作会社の株式会社サンライズが手がける人気アニメ

*¹¹ Wayback Machineを見る限り、2019年4月4日の17:39時点で既に改ざんされたサイトが表示されていたようです。<https://web.archive.org/web/20190404173940/http://www.lovelive-anime.jp/>

*¹² ちなみに「艦これ」の他に、リダイレクト先が「ハイスクール・フリート」や「蒼き鋼のアルペジオ」というパターンもあったようです。攻撃者の好みが垣間見えます

*¹³ <https://twitter.com/nsutngo/status/1113953668783456256>

5.7 <トラブル> ラブライブ！のオフィシャルサイトが乗っ取られた

メインの管理画面から `lovelive-anime.jp` というドメイン名の移管を申請し、申請を受けた JPRS は A 社に対してドメイン名の移管申請があったことを通知します。そして A 社から JPRS へ「ドメイン名の登録者（株式会社サンライズ）は移管を承認しない」という回答が 10 日以内^{*14}に行われなかったことで、JPRS が「承認する」という回答を得たものとみなし、その結果移管が行われてしまったものと推察されます。

^{*14} JPRS の「汎用 JP ドメイン名登録申請等の取次に関する規則 第 11 条（取次にかかる登録申請等に対する決定の伝達業務）第 2 項」より。<https://jprs.jp/doc/rule/toritsugi-rule-wideusejp.html>

猫好き@フォロバ100%
@nsutng0

返信先: @LoveLive_staffさん
恐らくドメイン乗っ取られてますね。
グローバルIP直打ちならアクセス出来るのでは?

ドメイン再取得出来ると良いですね...

検索タイプ 検索キーワード
[ドメイン名情報] lovelive-anime.jp [検索] [検索]

Domain Information: [ドメイン情報]
[Domain Name] LOVELIVE-ANIME.JP

[Name Server] ns1.star-domain.jp
[Name Server] ns2.star-domain.jp
[Name Server] ns3.star-domain.jp
[Signing Key]

[登録年月日] 2012/07/27
[有効期限] 2019/07/31
[状態] Active
[最終更新] 2019/04/05 01:18:00 (JST)

Contact Information: [公開連絡窓口]
[名前] Whois公開代行
[Name] Whois Privacy
[Email] whois@sky-domain.jp
[Web Page]
[郵便番号] 604-8006
[住所] 京都府京都市中京区河原町通三条上る下丸屋町
[Postal Address]
[電話番号] 075-256-8553
[FAX番号]

株式会社日本レジストリサービス Copyright© Japan Registry Service

プライバシーポリシー | 著作権 | お問い合わせ : info@jprs.jp

▲図 5.6 不正に移管された際の Whois 情報

しかしこの「10日以内に回答がなければ移管を承認したとみなす」というのは、あくまでレジストリとレジストラの間での話であり、レジストラとドメイン名を買った人との間も同じ動きになっているとは限りません。

たとえばお名前.comでは、ドメイン名の移管申請の通知を受けると、当該ドメイン名でWhoisの管理担当者に登録されているメールアドレスに対して「【重要】トランスマ

申請に関する確認のご連絡」というメールを送り、移管の意志を確認します。そしてメールに記載された期日までに承認あるいは拒否の手続きが行われなければ、自動的に承認しないものとみなします。

ここは完全に推測ですが、株式会社サンライズが lovelive-anime.jp を登録していたレジストラの A 社は、この「期限までに回答がなかった場合」に何もしない、あるいはレジストリである JPRS と同様に「承認する」とみなした動きをする実装になっていたものと思われます。同様の事象発生を防ぐために、あなたも普段利用しているレジストラが、移管の通知を受けたときにどのような動きをするのか確認しておきましょう。

5.7.1 移管できないようロックしておくべき安心？

そもそも JPRS から通知を受けたレジストラの A 社が、ドメイン名の登録者（株式会社サンライズ）に対して移管を承認するか拒否するかの確認メールを出した際に、登録者がきちんと「移管しない」と回答していればこの乗っ取りは防げていました。しかし重要なメールや通知をうっかり見落としてしまう失敗は、誰しも一度は経験があると思います。大切なドメイン名で勝手に移管が進んでしまわないよう、ロックしておくことはできないのでしょうか？

実は JPRS では、jp ドメイン名の登録者情報やレジストラの変更申請そのものを制限できる「レジストリロックサービス」^{*15}という機能を提供しています。これを有効にしていれば、そもそも申請自体を受け付けなくなるため安心です。しかしこのレジストリロックサービスは、「1 組織 1 ドメイン名制限緩和申請」や「Whois 登録者情報非表示設定機能」^{*16}と同様に、レジストラが対応していないければ使うことができません。

「大切なドメイン名だから勝手に移管されないようロックしておきたい！」という場合は、今利用しているレジストラが JPRS の「レジストリロックサービス」に対応しているか、もしくはレジストラが自前で同様のロックサービス^{*17}を提供しているかを確認してみましょう。

^{*15} JPRS のレジストリロックサービス <https://jprs.jp/about/dom-rule/registry-lock/>

^{*16} 第 1 章「ドメイン名と Whois」を参照

^{*17} たとえば「お名前.com」には、他レジストラへの意図しないドメイン移管を防ぐための「ドメイン移管ロック」というサービスがあります。しかしこの「ドメイン移管ロック」では jp ドメイン名はサービスの対象外です。また「名づけてネット」にも不正な移管を阻止するための「レジストラロック」というサービスがありますが、こちらも jp ドメイン名は対象外です

付録 A

本当の AWS

何を言っているのかわからないと思いますが「AWS 用語だけでアイドルソングを作つて架空のアイドルに歌わせたい」そんな気持ちで作詞をしてしまいました。原稿の〆切に追われている時ほど才能の無駄遣いをしてしまう傾向にあります。あまりによくできてしまつたのでなぜか本著に収録します。それでは聞いてください。

A.1 AWS - 愛はワガママサンシャイン

はじめの駆け引き SecurityGroup
Elastic なこのキモチ みちびいてね Route53
『やめときな』って声は Gracier に
そだてて Beanstalk わたしたちのキズナ
一挙一動 CloudTrail
つながってみたいの Direct Connect
わたしの恋は Autoscale
IAM でわたしだけを見つけてね
ふたりの思い出 Redshift において
CloudFront 世界に届けてこの AI を

あとがき

ドメイン名と DNS を巡る旅はいかがでしたか？ 少しはこれから開発や運用にお役に立ちそうでしょうか？

筆者は新卒で Web アプリケーションエンジニアになったものの、色々な偶然が重なってその後は SIer やソーシャルゲームの広報としてキャリアを重ね、インフラエンジニアとして再スタートを切ったのは 29 歳の時でした。

今からでもちゃんとエンジニアになりたい！ けどもう遅すぎるんじゃないかな？ そんな泣きそうな気持ちでいたときに「まだ 29 歳でしょ。ケツの青いのが何を言う」と笑い飛ばしてくれた夫（上から下までなんでも知ってるチートみたいなフルスタックエンジニア）のお陰で、エンジニアとしての新たな一步を踏み出すことが出来ました。

自分自身が「Apache ってなんだっけ…？ なんとか d で終わる名前のやつ…？」みたいな状態から始だったので、今でも「分からない人の気持ち」、そして「技術を学びたいけどどこから始めたらいいのか途方に暮れる人の不安な気持ち」は痛いほど分かります。

本著が DNS が分からなくて困っている誰かの役に立てば、それ以上に嬉しいことはありません。

数ある技術本の中から「DNS 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。

2019 年 9 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/813317>
- パスワード : *****

Special Thanks:

- 初めての本作りに奮闘する姿を見守ってくれた猫と旦那様と息子
- soushi @_so4
- きやすね @casnail

レビュアー

- 深澤俊
- Takeshi Matsuba

参考書籍

- ワンストップ！ 技術同人誌を書こう - 親方@oyakata2438
 - <https://oyakata.booth.pm/items/708196>
- 技術書をかこう！ ～はじめての Re:VIEW～ 改訂版 - Techbooster
 - <https://techbooster.booth.pm/items/586727>
- プロフェッショナル IPv6 - 小川 晃通
 - <https://www.lambdanote.com/products/ipv6>
- DNS がよくわかる教科書 - 株式会社日本レジストリサービス (JPRS)
 - <https://www.amazon.co.jp/dp/B07KQSRZ1S/>
- DNSDNS Resolution - Cryptic Command
 - <https://cryptic-command.booth.pm/items/1317266>

著者紹介

mochiko / @mochikoAsTech

無職。元 Web 制作会社のシステムエンジニア。モバイルサイトのエンジニア、SIer とソーシャルゲームの広報を経て、2013 年よりサーバホスティングサービスの構築と運用を担当したのち、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。「分からぬ気持ち」に寄り添える技術者になれるように日々奮闘中。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。本著の名付け親。

Shinya Nagashio

挿絵デザインを担当。

DNSをはじめよう

基礎からトラブルシューティングまで 改訂第2版

2019年9月22日 標準書典7 改訂第2版

著 者 mochikoAsTech

デザイン Hikaru Wakamatsu / Shinya Nagashio

発行所 mochikoAsTech

印刷所 日光企画

(C) 2019 mochikoAsTech