

DNS をはじめよう

mochikoAsTech 著

2018-04-22 版 好きなコマンドは `dig` です 発行

はじめに

2018年4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。

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のことでも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、現在のWebアプリケーションエンジニアに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す。

ました。

「DNS をはじめよう」はそんな著者が、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。

想定する読者層

この本は、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

この本の特徴

この本では実際にドメインを 1 つ購入します。買ったドメイン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、実際にコマンドをたたいて反復学習するためのドリルもついています。

この本のゴール

この本を読み終わると、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、電子版については必要に応じて適宜改訂を行いますので筆者までお知らせいただけますと幸いです。

目次

はじめに	3
想定する読者層	4
この本の特徴	4
この本のゴール	4
免責事項	5
第1章 ドメインと Whois	11
1.1　ドメインのお店選び	12
1.1.1　お名前.com	12
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の品質は違うのか	17
1.3　ドメイン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　ドメインが生まれてから手元に届くまで	21
1.4.1　ドメイン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 ドメインが原因	24
1.4.4　【ドリル】ネットショップのドメインは○○.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】他のドメインと比べて jp ドメインが高い理由	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	実際にお名前.com でドメインを買ってみよう	29
1.8.1	自動更新はオフにしておこう	35
1.8.2	<トラブル> ドメイン自動更新の落とし穴	41
1.8.3	【ドリル】連絡先メールアドレスは自分でいい?	42
1.9	Whois とは	42
1.9.1	JPRS WHOIS でドメインの所有者情報を見てみよう	44
【コラム】ドメインは大文字小文字の区別をしない	47	
1.9.2	Whois の項目はレジストリごとに微妙に違う	47
1.9.3	Whois を正確に登録しなければいけない理由	50
1.9.4	<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	50
1.9.5	【ドリル】Whois に登録すべきなのはクライアントの情報?	52
1.9.6	プライバシーを守るための Whois 情報公開代行	53
1.10	.co.jp は国内企業しか買えないドメイン	55
1.10.1	【ドリル】.co.jp ドメインは 2 つ買える?	56
1.11	ドメインの有効期限が切れるとどうなるのか?	57
1.11.1	<トラブル> ドロップキャッチされたイオンシネマ	57
1.11.2	【ドリル】ドメインを手放してよい条件は?	58
1.12	ドメインを買ったたらサイトが見られるか?	59
第 2 章	DNS の仕組み	61
2.1	DNS とは	62
2.1.1	ネームサーバ	62
2.1.2	フルリゾルバ	62
2.2	お名前.com のページが表示されるまで	64
2.3	ゾーンと委任	65
2.4	リソースレコード	67
第 3 章	AWS のネームサーバ (Route53) を使ってみよう	69
3.1	AWS とは	70
3.2	AWS アカウント作成	70
3.2.1	AWS のマネジメントコンソールにログイン	79
3.3	ドメインのネームサーバを Route53 に変更	81
3.3.1	Route53 でホストゾーンを作成	81
3.3.2	自分のドメインのネームサーバが何か確認	84

3.3.3	ネームサーバをお名前.com から Route53 に変更	88
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	95
3.3.5	【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	97
第 4 章	dig と whois を叩いて学ぶ DNS	99
4.1	dig と whois のインストール	100
4.1.1	Mac も Linux のサーバ環境もない場合	100
4.2	nslookup はもう卒業！ dig コマンドの便利な使い方	101
4.2.1	host や nslookup じゃダメですか？	102
4.3	Whois を叩いてドメインや IP の持ち主を調べよう	103
4.3.1	【ドリル】ドメインの有効期限が分からぬ	104
4.4	dig を叩いてリソースレコードを確認してみよう	105
4.4.1	A レコード	106
4.4.2	【ドリル】ウェブサーバはどこにある？	106
【コラム】dig のオプションは略せる	108	
4.4.3	MX レコード	108
4.4.4	<トラブル> test@test.co.jp を使って情報漏洩	110
4.4.5	NS レコード	113
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 .	114
4.4.7	SPF レコード (TXT レコード)	115
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？	117
4.4.9	PTR レコード	118
4.4.10	CNAME レコード	118
4.4.11	【ドリル】CNAME の調べ方と使いどころ	120
4.4.12	CNAME と他のリソースレコードは共存できない	121
4.4.13	グルーレコード	123
第 5 章	トラブルシューティング	125
5.1	<トラブル> URL は www ありなしどっち？	126
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	127
5.3	サイト移管の AtoZ	128
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 .	129
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	131
5.5	CAA レコードが原因で SSL 証明書が発行できなかった	137
5.6	<トラブル> AWS で突然ドメイン名が引けなくなった	138
5.6.1	レートリミットを超えると NXDOMAIN を返す	138

目次

5.6.2 勝手に TTL を短くする	138
付録 A　本当の AWS	141
A.1 AWS - 愛はワガママサンシャイン	142
あとがき	143
Special Thanks:	143
レビュー	143
参考書	144
著者紹介	145

第 1 章

ドメインと Whois

自分のドメイン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を買ってみましょう。

1.1 ドメインのお店選び

それではまずどこでドメインを買うのか、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。

1.1.1 お名前.com

お名前.com で example.co.jp を調べてみると、1 年で 3,780 円^{*1}でした。（図 1.2）しかも残念ながら×がついていて「登録済みドメインのため、お申込みいただけません。」と

^{*1} ドメインの値段はすべて税別（ゴンベエドメインのみ税込）表記で、2018年2月に著者が検索した時点の金額です。

なっています。^{*2}

The screenshot shows a search results page for 'example'. A legend at the top indicates that 'x' marks mean the domain is sold out. The grid below lists various domain extensions and their prices:

検索したドメイン	.com	.net	.jp	.work	.xyz	.site	.tokyo	.fun	.biz	NEW .store	.shop	.org	.co.jp
価格 (ローマ字)	1,160円	399円	2,480円	1円	30円	30円	99円	199円	299円	499円	980円	820円	3,780円
example	x	x	x	x	x	x	x	x	x	x	x	x	x

▲図 1.2 お名前.com では 3,780 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)

The screenshot shows the 'NamaZuketeNetto' website's domain selection process. It highlights the 'example.co.jp' option under the 'JP1年プラン' column.

ラベル	プラン
x-example.ac.jp	属性JP1年プラン 7,200円（税別）
x-example.co.jp	属性JP1年プラン 7,200円（税別）

▲図 1.3 名づけてねっとでは 7,200 円

^{*2} 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。

1.1.3 Yahoo! ドメイン

Yahoo! ドメインでも 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か? 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) を確認すると、Yahoo! ドメインではどうやら○○.co.jp というドメイン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

ムームードメインはお名前.com と同額の 3,780 円でした。(図 1.6)



▲図 1.6 ムームードメインでは 3,780 円

1.1.5 VALUE DOMAIN

VALUE DOMAIN もお名前.com やムームードメインと同じく 3,780 円でした。(図 1.7)

The screenshot shows the homepage of VALUE DOMAIN. At the top, there's 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', and 'ログイン/登録'. Below the navigation bar, a breadcrumb trail shows 'ドメイン取得&レンタルサーバー/バリュードメイン > ドメイン一覧・料金表 (gTLD)'.

JPドメイン

JPドメイン = JP Domain は、国や地域割り当てられているccTLDに属する、日本を表すドメインです。種類は、「汎用JP」「属性型JP (CO.JP / OR.JP)」「都道府県型JPドメイン」などがあります。

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

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）



▲図1.8 ゴンベエドメインでは5,616円

金額だけで比べるとお名前.com、VALUE DOMAIN、ムームードメインのいずれかで買えばよさそうです。（表1.1）

▼表1.1 example.co.jp のドメイン代

お店	金額
名づけてねっと	7,200円
ゴンベエドメイン	5,616円
お名前.com	3,780円
ムームードメイン	3,780円
VALUE DOMAIN	3,780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3,780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を取り扱っていないかったのでしょうか？

1.2 値段やお店によってドメインの品質は違うのか

あまりに金額に差があるので、もしかして同じ「ドメイン」という名前がついていても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？

- A. 同じドメイン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はドメインである

答え _____

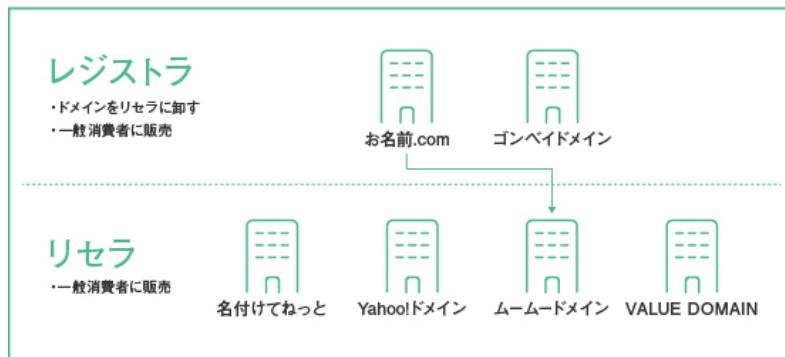
正解は B です。ドメインはどこのお店で買っても品質に差はありません。プロ向けに業務用のドメイン屋さんがあるわけではないので、Web 制作会社でもよくお名前.com でドメインを購入しています。ロボット掃除機のルンバはどこで買っても同じルンバですが、ヤマダ電機とヨドバシカメラで値段は異なりますよね。ルンバと同じようにドメインもどこで買っても同じドメインですがお店によって値段が違うのです。

「なるほど！ どこで買ってもドメインはドメイン。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^{*3}、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）

^{*3} 正確にはムームードメインは jp ドメイン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つた、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い連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、レジストラか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。

それからドメインを買うとそのドメイン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。ドメインの金額が同じなら前者の方がいいですね。

さらにお店によってはドメイン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つたりします。

前述のようにドメイン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。

迷つ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は550種類以上で、累積登録実績1600万件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^{*4}の保有数ランキング(図1.10)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.)	5,824,073	-20,526 ↓	25.66%
2. NameCheap, Inc.	2,657,216	16,350 ↑	11.71%
3. GoDaddy.com, LLC (GoDaddy Group)	1,949,017	4,316 ↑	8.59%
4. GMO Internet, Inc. d/b/a Onamae.com	1,398,405	5,779 ↑	6.16%
5. Chengdu West Dimension Digital Technology Co., Ltd. (www.west.cn)	1,193,945	-2,842 ↓	5.26%
6. Alpnames Limited	724,292	-13,544 ↓	3.19%
7. ALIBABA.COM SINGAPORE E-COMMERCE PRIVATE LIMITED	654,036	277 ↑	2.88%
8. PDR Ltd. d/b/a PublicDomainRegistry.com	539,639	1,102 ↑	2.38%
9. eNom, Inc. (Tucows)	429,273	585 ↑	1.89%
10. Tucows Domains Inc. (OpenSRS / Hover, Tucows)	409,351	998 ↑	1.80%

▲図1.10 新gTLDの保有数が多いレジストラランキング

^{*4} generic Top Level Domain の略。gTLDの一覧は <https://www.nic.ad.jp/ja/dom/types.html> を参照。

【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。（実際、ドメインの金額も同じですし）

渋谷でセルリアンタワー^a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。

^a <https://www.gmo.jp/company-profile/outline/>

1.4 ドメインが生まれてから手元に届くまで

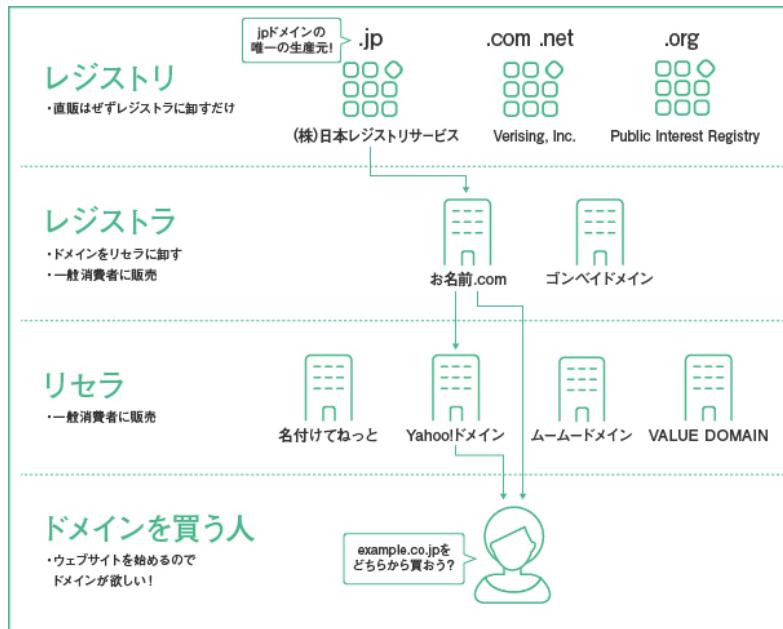
さて折角、レジストラとリセラの話が出てきたので、ドメインが生まれてから私たちの手元にやってくるまでの流れも知っておきましょう。

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を仕入れているのでしょうか？ どこかにドメイン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が収穫できるのでしょうか？

1.4.1 ドメインの卸元はレジストリ

残念ながらドメインは畑からは採れません。レジストラはドメインをレジストリ（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）



▲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 からでも購入できて、実はどちらから買っても御元は同じということです。

ドメインの御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、実際ドメインを買うときは、ドメインを買う人=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。

では御元であるレジストリとは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。^{*5} 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）によっ

^{*5} 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。

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？ 最近は車内広告を出したりしているので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」という会社です。VeriSign Global Registry Services は com だけでなく net や name など複数の TLD を保有しています。

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。



▲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 ドメインのネームサーバが、Mackerel の mackerel.io というドメイン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アドレスなどで、.io のような新興の安い TLD を採用するのはあまりお勧めしません。ドメインを買うときは、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？ と事前に確認しておきましょう。たとえば jp ドメインの JPRS や tokyo ドメインの 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は○○.xxx? ○○.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を○○.com にする予定でしたが、ブランド名に「KISS」を含むため「○○.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は○○.com にする
- B. 提案を受け入れてドメインは○○.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^a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。

^a <https://www.gmoregistry.com/>

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 の唯一の御元となっています。

たとえば example.jp のように jp で終わるドメインなら、先ほどの JPRS が唯一の御元です。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（あるいはさらにレジストラからリセラに卸して）そこで私たちが買うのです。

ちなみに御元のレジストリから、ドメイン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 はこ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】他のドメインと比べて jp ドメインが高い理由

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 は元々の御値が高いので 2,480 円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 は他の ccTLD^aに比べて高すぎる、独占企業だからって高い値段で売ってないで値下げしようよ」と JPRS に公開質問^b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としているので前述の io ドメイン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 ドメイン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！

^a 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。

^b <https://www.kumagai.com/?eid=718>

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。ここが全ドメインの本当の生産元と言つてもいいかもしれません。

.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 では買うことできません。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）が自由で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つたですよね。これは Yahoo! ドメインが「うちでは.co.jp を扱わない」という判断をしたからです。

これがレジストリ、レジストラ、リセラの関係でした。

1.8 実際にお名前.comでドメインを買ってみよう

では実際に、お名前.com^{*6}でドメイン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 1.15）を開きます。



▲図 1.15 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」ボタンをクリックして、ドメイン検索の画面（図 1.16）を開きます。

*6 <https://www.onamae.com/>

ドメイン検索

ご希望の文字列を入力して「検索」を押してください。

お名前.comでは、550種類以上のドメインからお選びいただけます。

ご希望のドメインの種類がお決まりでしたら下のチェックボックスより選択のうえ、検索をしてください。

特別価格対象ドメイン									
.net 399円 <input checked="" type="checkbox"/>	.com 1,160円 <input checked="" type="checkbox"/>	.work 1円 <input checked="" type="checkbox"/>	.online 30円 <input checked="" type="checkbox"/>	.site 30円 <input checked="" type="checkbox"/>	.xyc 30円 <input checked="" type="checkbox"/>	.club 99円 <input checked="" type="checkbox"/>	.pw 99円 <input checked="" type="checkbox"/>	.tokyo 99円 <input checked="" type="checkbox"/>	
.top 99円 <input checked="" type="checkbox"/>	.website 110円 <input checked="" type="checkbox"/>	.fun 199円 <input checked="" type="checkbox"/>	.space 199円 <input checked="" type="checkbox"/>	.tech 199円 <input checked="" type="checkbox"/>	.biz 299円 <input checked="" type="checkbox"/>	.red 299円 <input checked="" type="checkbox"/>	.info 320円 <input checked="" type="checkbox"/>	.me 360円 <input checked="" type="checkbox"/>	
.cloud 399円 <input checked="" type="checkbox"/>	.asia 499円 <input checked="" type="checkbox"/>	.bio 499円 <input checked="" type="checkbox"/>	.green 499円 <input checked="" type="checkbox"/>	.life 499円 <input checked="" type="checkbox"/>	.organic 499円 <input checked="" type="checkbox"/>	.store 499円 <input checked="" type="checkbox"/>	.world 499円 <input checked="" type="checkbox"/>	.blue 560円 <input checked="" type="checkbox"/>	
.mobi 560円 <input checked="" type="checkbox"/>	.pink 560円 <input checked="" type="checkbox"/>	.host 599円 <input checked="" type="checkbox"/>	.press 599円 <input checked="" type="checkbox"/>	.pro 660円 <input checked="" type="checkbox"/>	.co 699円 <input checked="" type="checkbox"/>	.design 699円 <input checked="" type="checkbox"/>	.gift 799円 <input checked="" type="checkbox"/>	.org 820円 <input checked="" type="checkbox"/>	

▲図 1.16 ドメイン検索

といったん右上の全選択を押してチェックボックスのチェックを外したら、好きな文字^{*7}を入力して欲しい TLD にだけチェック（図 1.17）を入れてください。

今回はちゃんとしたサービスで使う訳ではなく、ドメインや DNS の仕組みを学ぶことが目的なので 1 円～30 円程度の安いドメインで構いません。それから詳細は後述しますが jp ドメインは持ち主の名前が 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。

*7 ドメインには_（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。

お名前.com ドメイン検索

ご希望の文字列を入力して「検索」を押してください。

お名前.comでは、550種類以上のドメインからお選びいただけます。

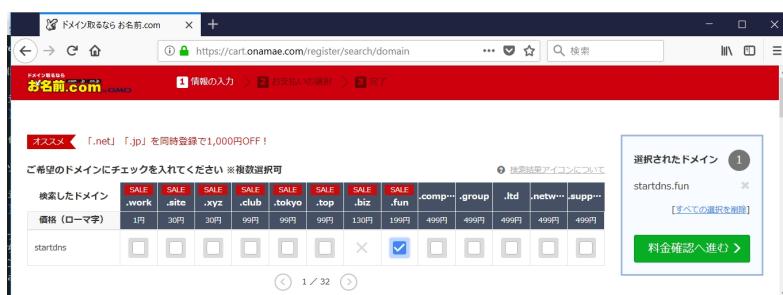
ご希望のドメインの種類がお決まりでしたら下のチェックボックスより選択のうえ、検索をしてください。

The screenshot shows a search interface for Onamae.com. At the top, there's a search bar with placeholder text "まずは、ドメイン名を検索。" and a green "検索" button. Below the search bar, a message says "現在の価格有効期限は、2018年3月2日(金)17:00まで". Underneath, there's a section titled "特別価格対象ドメイン" with a "全選択" checkbox. A grid of TLD options is shown, each with a price and a checkbox. The options include .net (599円), .com (1,160円), .work (1円), .online (30円), .site (30円), .xyz (30円), .club (99円), .pw (99円), .tokyo (99円), .top (99円), .website (110円), .biz (130円), .info (130円), .fun (199円), .space (199円), .tech (199円), .asia (299円), and .red (299円). Some TLDs like .work and .biz have their checkboxes checked.

▲図 1.17 好きな文字を入力して TLD にチェック

筆者は「startdns.〇〇」というドメインが欲しいので、startdnsと記入して比較的安い1円～199円程度のTLDにチェックを入れてみました。TLDを1つしか選ばないとそのドメインが売り切れたったということもあるので、いくつかチェックを入れた上で「検索」ボタンをクリックしましょう。

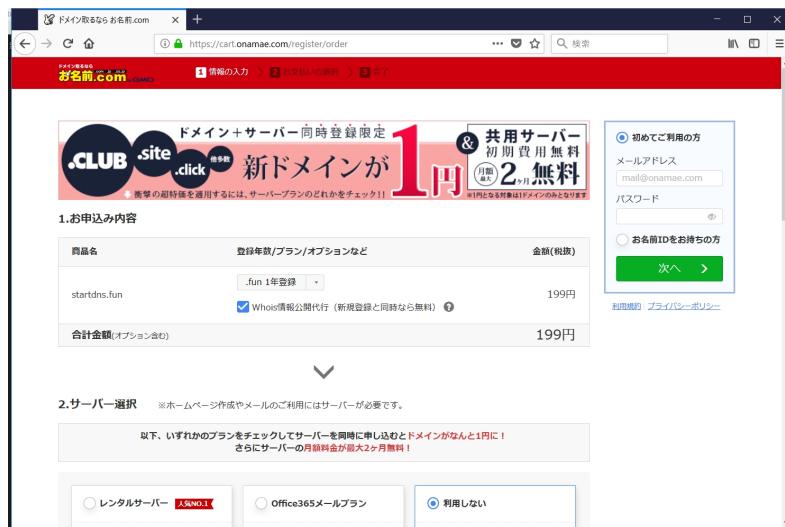
すると次の画面で「これは買えるよ、これは売り切れたよ」と教えてくれます。どうやらstartdns.bizは売り切れたようです。選択肢が幾つかある中で筆者は「startdns.fun」を買うことに決めて、funの下にあるチェックボックスにチェックを入れました。(図1.18) 右側の「選択されたドメイン」に「startdns.fun」だけが表示されています。自分が買おうと決めたドメインだけが右側に表示されていることを確認してください。



▲図 1.18 .funにチェックを入れると「選択されたドメイン」に startdns.fun が出る

なお記載されているのは有効期間1年のドメイン代ですので、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^{*8}も確認しておきましょう。購入するドメインが決まつたら右側の「料金確認へ進む」を押します。

次に表示されるお申込内容の確認画面（図1.19）でとても大切な作業があります。ここでドメイン代を再度確認するとともに「Whois情報公開代行（新規登録と同時なら無料）」にチェックを入れるのを絶対に忘れないようにしてください。チェックを入れ忘れて後からWhois情報公開代行を頼もうとすると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。



▲図1.19 「Whois情報公開代行（新規登録と同時なら無料）」に必ずチェックを入れる

なお画面下の方で、まるでセットのポテトのように「ご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。

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1.20）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

^{*8} <http://www.onamae.com/service/d-price/>

ンを買った直後、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。

▲図 1.20 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 1.21）が表示されます。右側の「お申込み内容」に、選択したドメインの料金および 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。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^{*9}して右側にある緑色の「申込む」を押します。

^{*9} 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。



▲図 1.21 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 1.22）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 1.22 サーバーは不要なので「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が買えたら購入完了画面（図 1.23）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。^{*10}



▲図 1.23 ドメインの購入完了画面が表示された

さてめでたくドメインが買えたのですが、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。

1.8.1 自動更新はオフにしておこう

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 1.24) 「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 1.25)が表示されたら、先ほど発行されたばかりのお名前 ID^{*11}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。

^{*10} なおお名前.comでドメインを購入すると毎日何通も「このドメインもいらない?」「今ならこんないいドメインが買えるよ!」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/

^{*11} 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。



▲図 1.24 右上の「ドメイン Navi ログイン」をクリック



▲図 1.25 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)

1.26) が表示されます。^{*12}

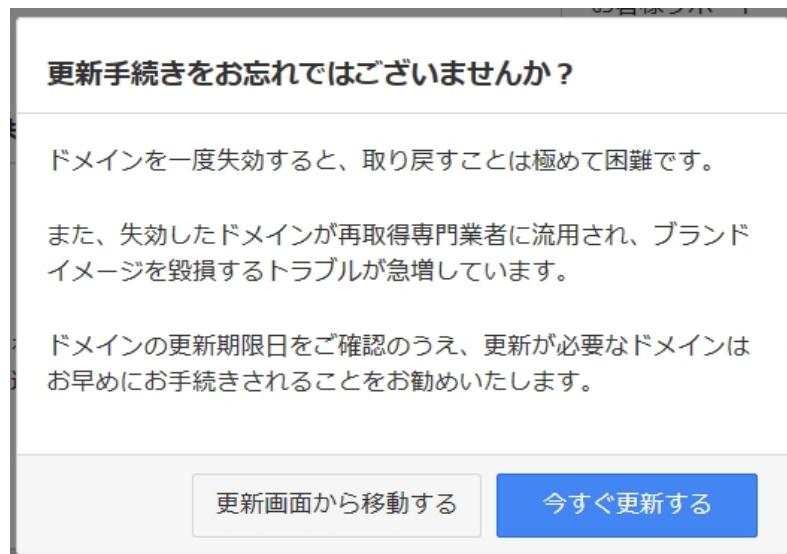


▲図 1.26 ドメイン Navi ドメイン契約更新

「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。先ほど買ったドメイン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。」と警告が出る（図 1.27）のでちょっとドキッときますが、有効期限は1年も先ですので問題ありません。強い気持ちで「このページを離れる」を押してください。

^{*12} 2018年2月時点、お名前.com は AB テストの最中な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ようです。この本では新しい方の画面を用いて説明しますので、人によってお手元で見ている画面と見た目が一致しないかも知れません。ですが新旧問わずにやりたいことは同じで「今買ったドメインで自動更新をオフにしたい」だけです。



▲図 1.27 ドメイン Navi 更新アラート

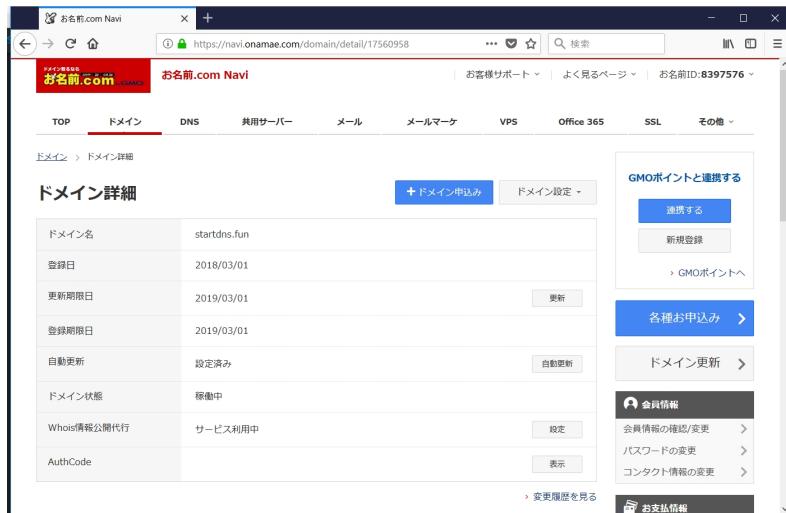
ドメイン一覧（図 1.28）を開くと、今買ったばかりのドメイン（筆者なら startdns.fun）が表示されます。自分が買ったドメインをクリックしてください。



▲図 1.28 ドメイン Navi ドメイン一覧

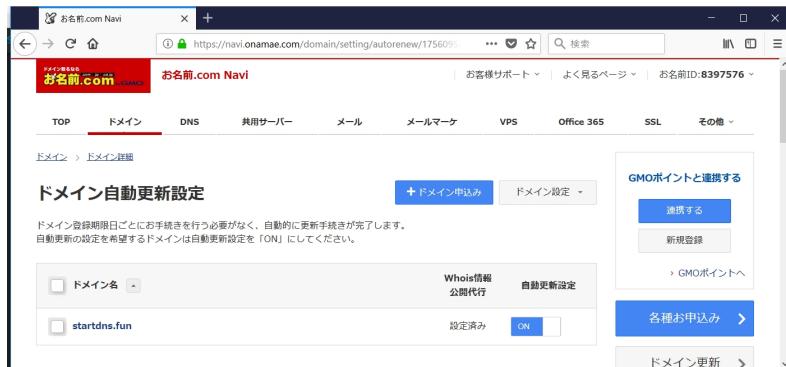
ドメイン詳細（図 1.29）が表示されたら「自動更新」という項目を確認してください。きっと「設定済み」になっていると思います。

1.8 実際にお名前.com でドメインを買ってみよう



▲図 1.29 ドメイン Navi ドメイン詳細

「設定済み」の右側にある「自動更新」を押してドメイン自動更新設定（図 1.30）の画面に進みます。



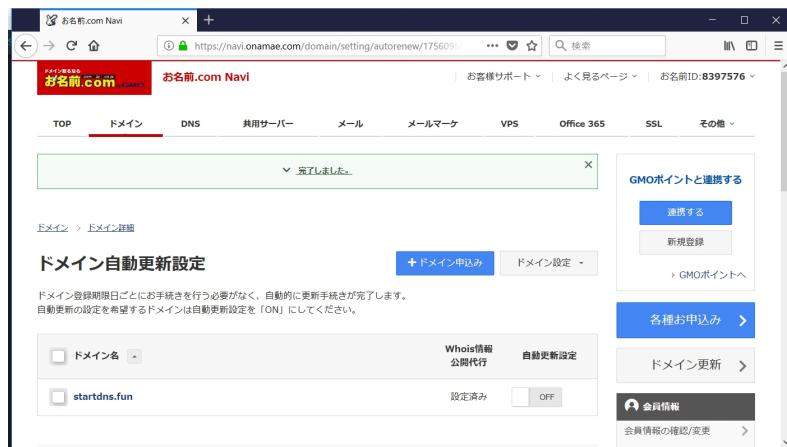
▲図 1.30 ドメイン Navi ドメイン自動更新設定

ドメイン自動更新設定（図 1.30）の画面で「ON」と書かれている箇所をクリックして自動更新を解除します。「自動更新を解除すると、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 1.31）が構いません。OK を押してください。



▲図 1.31 ドメイン自動更新解除の確認

図 1.32 のように自動更新が OFF になったら設定変更は完了です。



▲図 1.32 ドメイン自動更新が OFF になったら完了

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は更新されてクレジットカードからドメイン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。

1.8.2 <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「ドメインは買ついたからこれを使って」と伝えました。^{*13}

ディレクターCさんがドメインを買った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！ 更新しないと期限切れちゃうよ！」というメール^{*14}が届きますが、宛て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のアカウント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、全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を譲つてもらったものの、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

^{*13} 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・名称等は架空であり、実在のものとは関係ありません。

^{*14} ドメインの期限が切れる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。

もドメインの自動更新はオフにしておきましょう。

1.8.3 【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を取ることにしました。お名前.com でドメインを買うとき、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

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のように個人利用のためのドメインであれば、連絡先メールアドレスも個人のもので構いません。

1.9 Whois とは

ところで先ほど購入するとき「Whois 情報公開代行（新規登録と同時なら無料）」というチェックボックスにチェックを入れましたが、もしかしてそこでチェックを入れずに購入手続きを進めていくと「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？

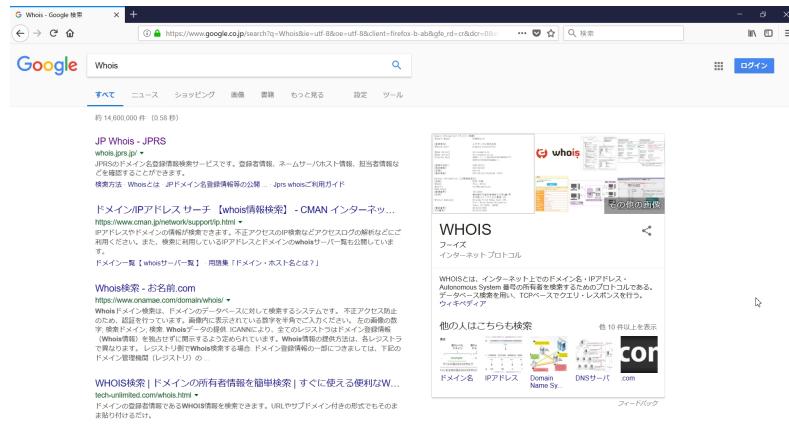
- A. 絶対登録しなきゃいけない！
- B. 任意だから登録しなくてもいい

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？

Whois とは、そのドメイン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 1.33）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？



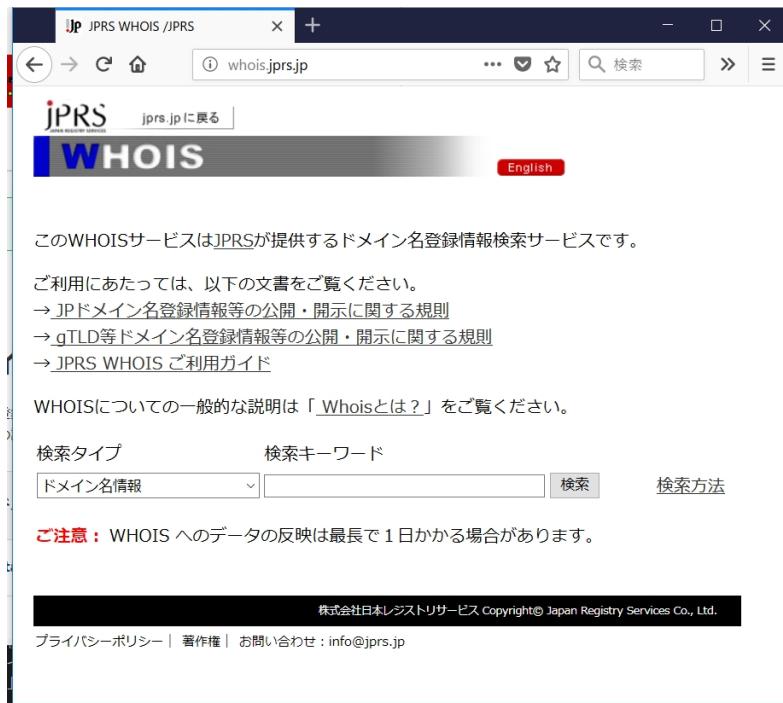
▲図 1.33 Whois で検索するとたくさんのサイトが出てくる

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。

1.9.1 JPRS WHOIS でドメインの所有者情報を見てみよう

たとえば jp で終わるドメインを管理している JPRS が提供する「Whois 情報確認サイト」はこちちら (図 1.34)*15です。



▲図 1.34 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 1.35)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 2018/03/31 が次の更新タイミングなんだな」と分かります。

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



▲図 1.35 cupnoodle.jp の Whois 情報

このように Whois を使えばドメイン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 1.36）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の一覧が出てきました。

The screenshot shows a web browser window for the JPRS WHOIS service. The search term entered is "日清食品株式会社". The results list numerous domain names registered under this entity, such as INSTANT-RAMEN.JP, CUPNOODLE.JP, E-MENSHOP.JP, NISSINSHOKUHIN.JP, INSTANT-NOODLES.JP, NISSIN-NOODLES.JP, NISSIN-RAMEN.JP, DONBEI.JP, CHIKINRAMEN.JP, CHICKEN-RAMEN.JP, NISSIN-UFO.JP, DEMARAMEN.JP, MENTATSU.JP, TONGARASHIMEN.JP, GYORETSU.JP, NISSIN-YAKISOBA.JP, and RAMENYASAN.JP.

登録者名	ドメイン名
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	INSTANT-RAMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	CUPNOODLE.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	E-MENSHOP.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	NISSINSHOKUHIN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	INSTANT-NOODLES.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	NISSIN-NOODLES.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	NISSIN-RAMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	DONBEI.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	CHIKINRAMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	CHICKEN-RAMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	NISSIN-UFO.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	DEMARAMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	MENTATSU.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	TONGARASHIMEN.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	GYORETSU.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	NISSIN-YAKISOBA.JP
日清食品株式会社 (NISSIN FOOD PRODUCTS CO., LTD.)	RAMENYASAN.JP

▲図 1.36 登録者名が日清食品株式会社のドメイン

【コラム】 ドメイン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 1.35）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です。

ためしに大文字小文字混ぜこぜにした <http://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。

1.9.2 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 1.37）



▲図 1.37 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^{*16}に別々（表 1.2）なのです。

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でも調べられる aguse.jp（図 1.38）^{*17}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

^{*16} 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。

^{*17} アグスジェーピー <https://www.aguse.jp/>

▼表 1.2 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/

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 1.38 どのドメインでも Whois が調べられる 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*¹⁸への移行が提唱され

*¹⁸ Registration Data Access Protocol の略。詳しくは <https://tsuchinoko.dmmrlabs.com/?p=4074> を参照。

ています。

1.9.3 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」ものの^{*19}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まになっていたりするとドメイン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。ドメインを買ったら Whois 情報はきちんと登録してください。

1.9.4 <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 1.39）が飛んできま

^{*19} 厳密には TLD によって方針が少しずつ異なります。

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、ドメインが利用停止になってしまいます。

【重要】[お名前.com] ドメイン 情報認証のお願い



▲図 1.39 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^{*20}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 へ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 1.40）が表示されます。URL を踏んでこの画面

^{*20} ドメイン情報認証はメールアドレス 1 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。また全く別のお名前アカウントでドメインを購入した場合も、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。

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認証フォーム
Authentication Form of the Validity of the e-mail address

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すと、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 1.40 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

1.9.5 【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。^{*21}

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、メールアドレスだけは A 社に作ってもらったマーリングリスト（メンバーに B 社や C 社が入っている）にするのが得策でしょう。

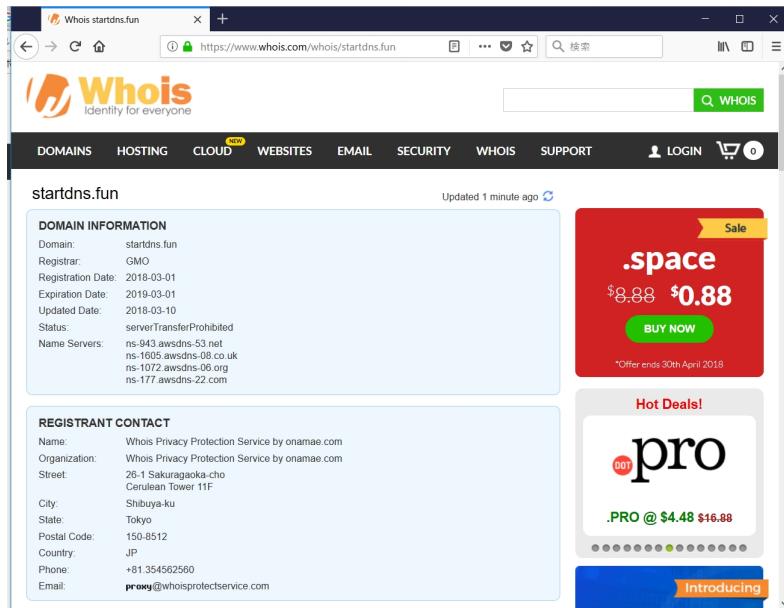
1.9.6 プライバシーを守るための Whois 情報公開代行

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 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「Whois 情報公開代行」というサービスです。

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。

^{*21} 2014 年に NPO 法人が小学 4 年生になりましたて作った「#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

実際に筆者の startdns.fun の Whois 情報を検索図 1.41 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の Whois を確認してみてください。



▲図 1.41 startdns.fun の Whois 情報を見るとお名前.com になっている

ただし.jp はレジストリである JPRS の方針として、ラジストラによる 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。

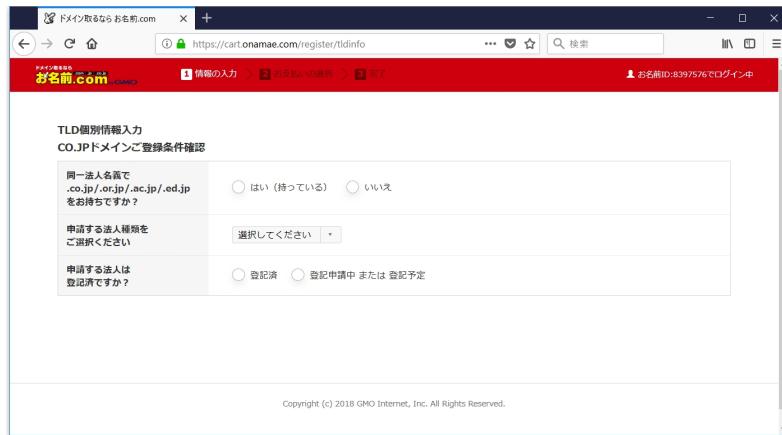
その代わりドメインを買った人がレジストラを通して JPRS に「登録者名を非表示にして欲しい」という申請^{*22}を出せば、JPRS 側で Whois の登録者名を「登録者からの申請により非表示」のようにしてくれるのですが、残念ながらお名前.com ではドメインを買った人から JPRS に対する「登録者名非表示の依頼代行」を受け付けていません。そのため結論としてはお名前.com で.jp のドメインを買った場合、Whois の登録者名に自分の

^{*22} <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、ドメインを買った人から直に JPRS へ申請することはできません。

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 .co.jp は国内企業しか買えないドメイン

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 1.42）でドメインを購入するための条件が表示されます。



▲図 1.42 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

きるドメインなどもあります。よく見る一般的な.com や.net や.biz などの gTLD^{*23}は、どこの国の誰でも無条件で購入できます。

1.10.1 【ドリル】.co.jp ドメインは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 ドメインを買うことは可能でしょうか？

- A. ○○.co.jp を保持したままで△△.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^{*24}を出せば1つの組織で複数の.co.jp ドメインを持てるようになりました。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。

^{*23} generic Top Level Domain の略。gTLD の一覧は <https://www.nic.ad.jp/ja/dom/types.html> を参照。

^{*24} <https://jpdirect.jp/organizational/domain-restriction-relaxation.html>

1.11 ドメインの有効期限が切れるとどうなるのか？

お名前.comでドメインを購入した場合、ドメイン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。

またドメインの持ち主が更新をせずドメインの期限が切れて一定期間が経過すると、そのドメイン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することがあります。このように誰かが落とした値のあるドメインを、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は一度も使われていないまっさらなドメイン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^{*25}すら存在しています。

1.11.1 <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^{*26}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（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^{*27}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^{*28}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。

^{*25} <http://blogs.itmedia.co.jp/mohno/2014/12/re.html>

^{*26} <http://www.aeoncinema.com/>

^{*27} https://web.archive.org/web/*/http://warnermycal.com

^{*28} <http://www.aeoncinema.com/>

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」と配慮をしていたようです。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う」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に毎年更新料を払うのは馬鹿らしい気もありますが、ドメインを手放した結果、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、状況を調べて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を持ったままの方が安かったのではないか、と思います。

1.11.2 【ドリル】ドメイン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のドメイン有効期限が、まる1年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。また今年の夏向けのキャンペーンサイトは別ドメインで作りました。ドメイン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

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。

その状態でサイトのドメイン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。アクセスもほんないしリンクも残っていなければ、最終的に「ドメイン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。

ドメインの更新代はおよそ数百円から高くても数万円程度です。確認をせざうかつに手放すと、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。

1.12 ドメインを買ったらサイトが見られるか？

ドメインも買ったし 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のサイトを見てみましょう。（図 1.43）



▲図 1.43 http://自分のドメインを開くとお名前の広告が出る

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」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「このドメインに紐づく IP アドレスが知りたいの」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。

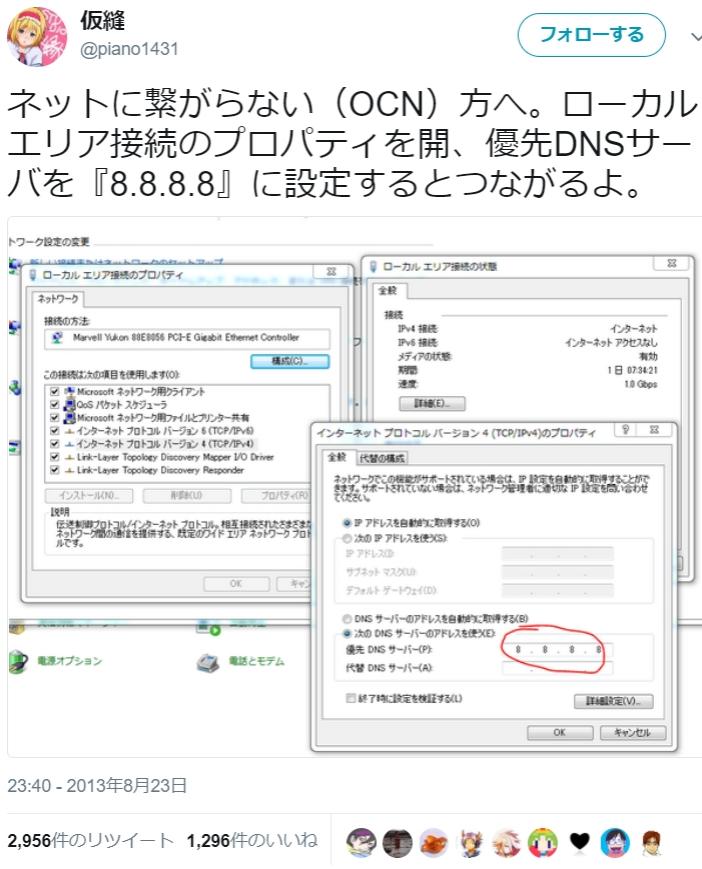
「DNS キャッシュサーバ」「フルサービスリゾルバ」と呼ばれることもありますが、本著では統一してフルリゾルバと呼びます。

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^{*1}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。

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

^{*1} 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。

Public DNS^{*2}のようにだれでも無料で使えるオープンソースのDNSサーバがあります。2013年8月にOCN^{*3}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^{*4}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

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に変更した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

^{*2} <https://developers.google.com/speed/public-dns/>

^{*3} 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。

^{*4} ドメインからIP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。

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
```

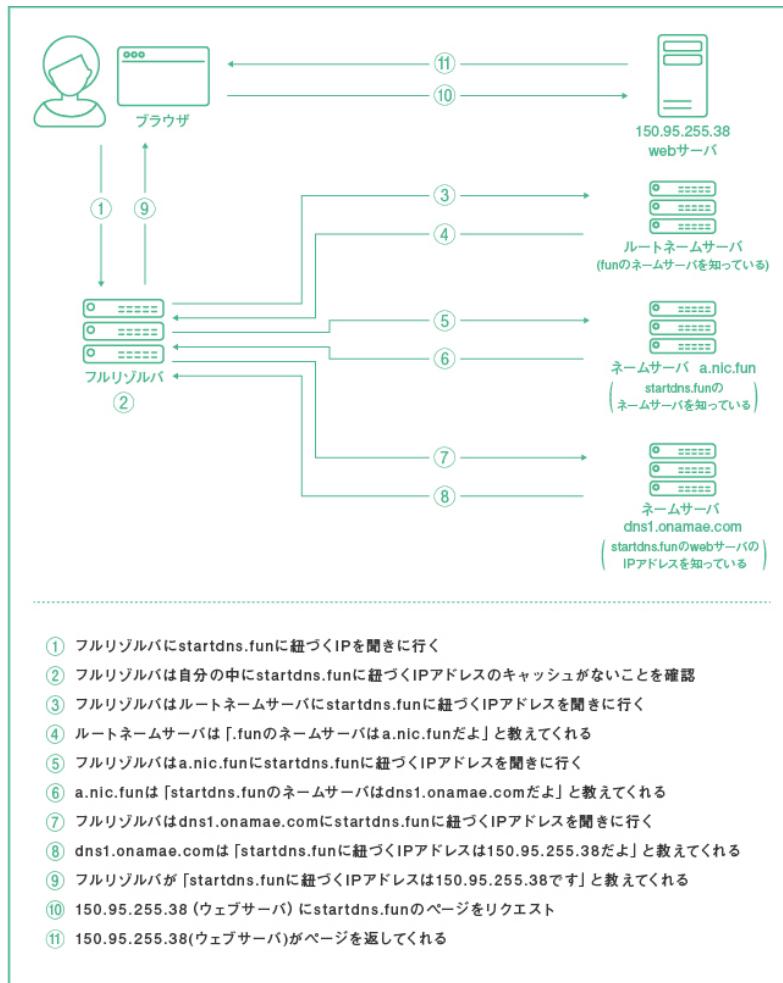
2.2 お名前.com のページが表示されるまで

第1章「ドメインと Whois」の最後で、自分が買ったドメイン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/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/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。

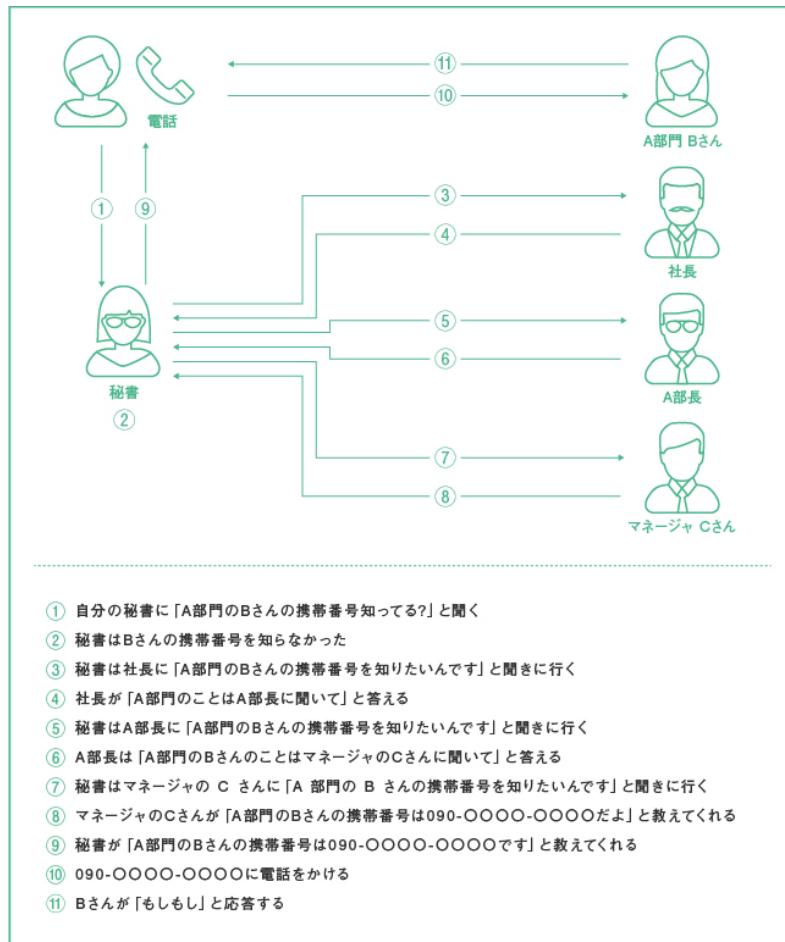


▲図 2.3 ページが表示されるまでのフルリゾルバとネームサーバの仕組み

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つのネームサーバ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たります。

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 に任せしていました。

このように自分が自身が任されているゾーンを分割して、一部のゾーンの管理を他のネームサーバに任せることを委任と呼びます。

つまり社長は A 部門というゾーンを A 部長に委任していた、そして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	ドメインのゾーンを管理するネームサーバ
MX レコード	ドメインに紐づくメール受信サーバ
TXT (SPF)	このドメインのメール送信元サーバ
SOA	ドメインのゾーンの管理情報
CNAME	このドメイン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？ 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していきましょう。

第 3 章

AWS のネームサーバ (Route53) を使ってみよう

第 1 章「ドメインと Whois」で自分のドメインを買って、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 Amazon がやっているクラウドです。

Amazon 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)、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てくる情報も多いので、今回は AWS の Route53 でドメインのゾーンを作成します。

3.2 AWS アカウント作成

まずは 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。

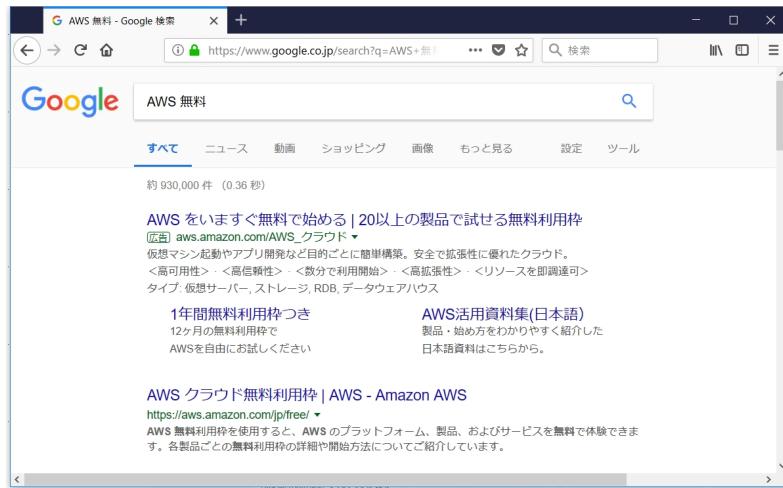
「AWS 無料」（図 3.1）で検索して上から 2 つめの「AWS クラウド無料利用枠 | AWS - Amazon AWS」^{*4}をクリックします。

^{*1} クラウドについて真面目に説明するともう 1 冊本ができそうなので、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} 2017 年時点、AWS はシェア全体の 34% 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/>



▲図 3.1 「AWS 無料」で検索

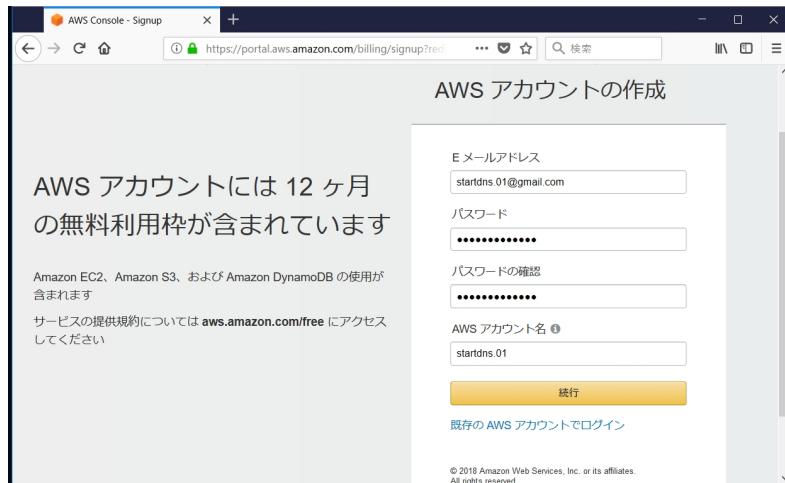
「AWS クラウド無料利用枠」(図 3.2) のページを開いたら、中央にある「まずは無料で始める」をクリックします。



▲図 3.2 「まずは無料で始める」をクリック

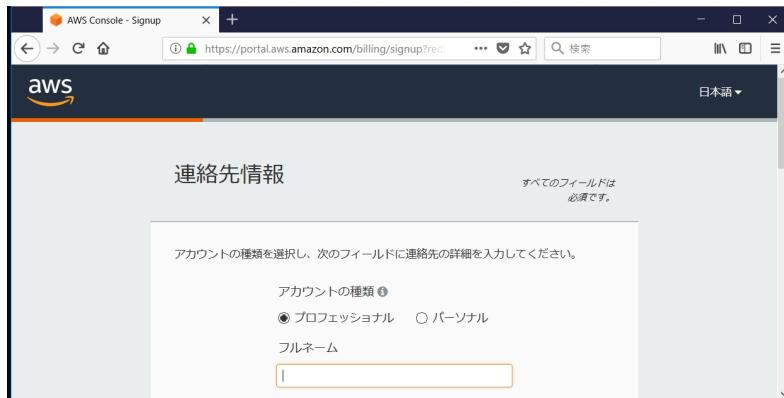
AWS アカウントの作成画面(図 3.3)で次の 4つを入力したら、「続行」をクリックします。後で分からなくなないように、何を登録したのかはメモしておいてください。

- E メールアドレス
- パスワード
- パスワードの確認
- AWS アカウント名

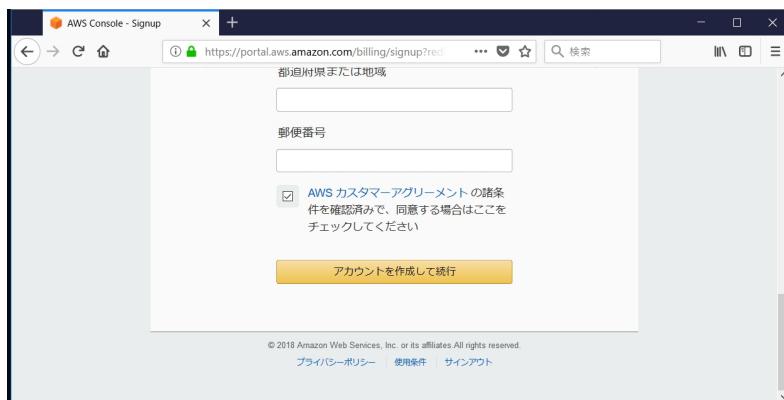


▲図 3.3 メールアドレスとパスワードと AWS アカウント名を記入して「続行」を押す

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

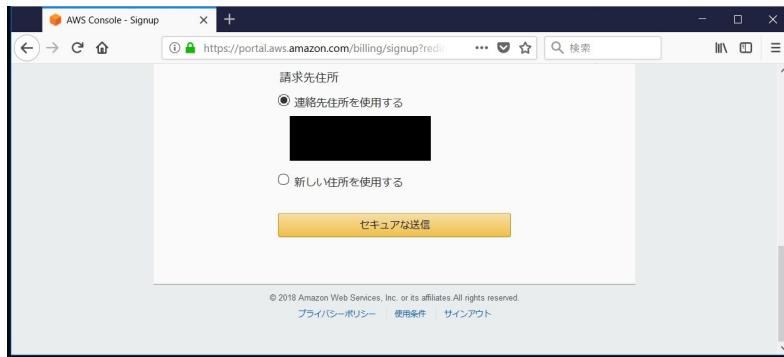


▲図 3.5 連絡先を入力したら「アカウントを作成して続行」を押す

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。

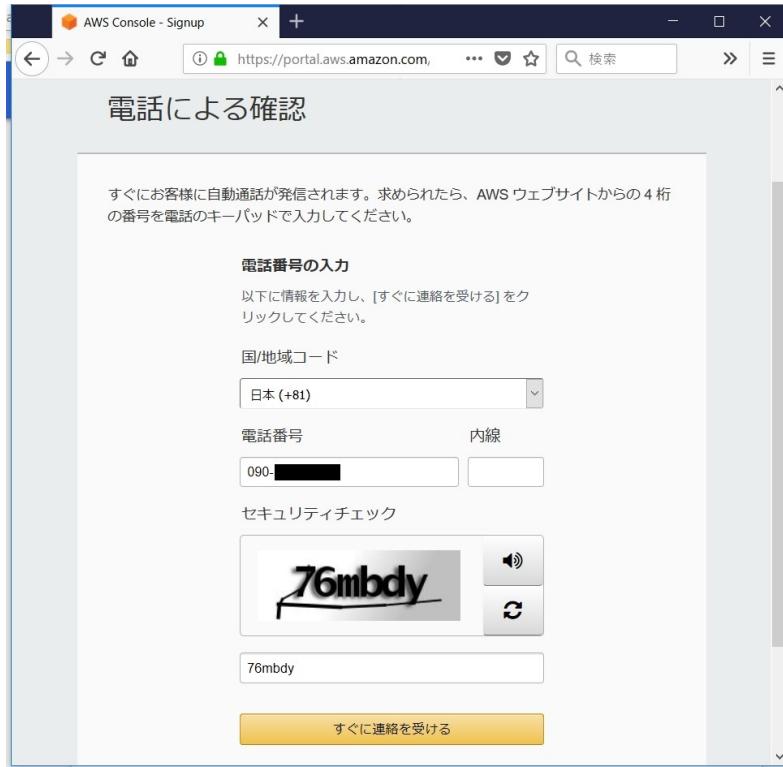


▲図 3.6 クレジットカード情報を記入



▲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の Captcha 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 Captcha 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。



▲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。



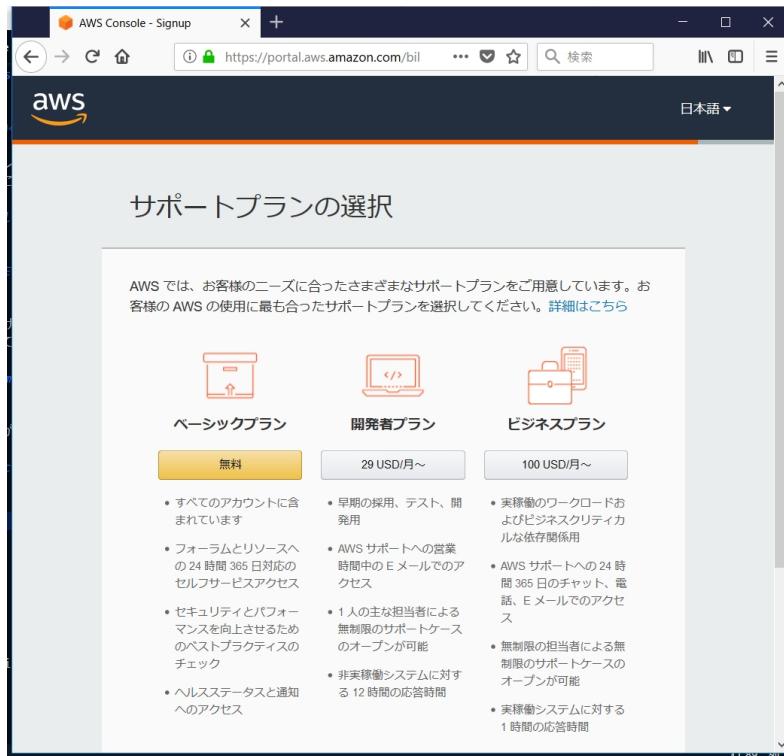
▲図 3.10 パソコンで表示された 4 ケタの番号を電話でプッシュ

確認完了して電話が切れた後、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

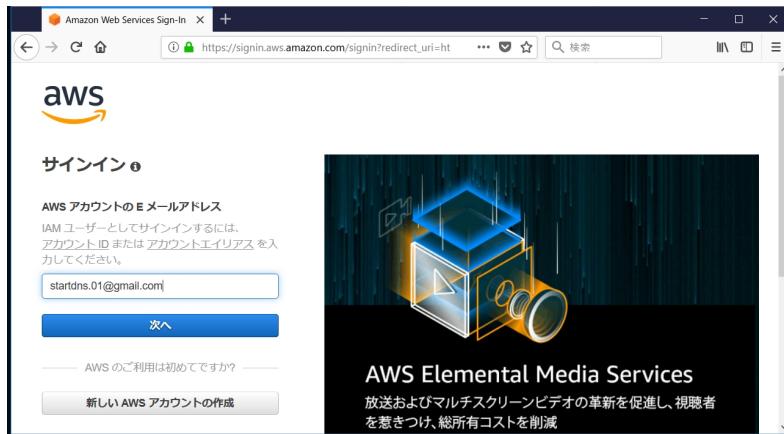
「Amazon ウェブ サービスへようこそ」(図 3.13) と表示されたら AWS のアカウント作成は完了です。おめでとうございます！ 「コンソールにサインインする」ボタンを押して次のステップへ進みましょう。



▲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

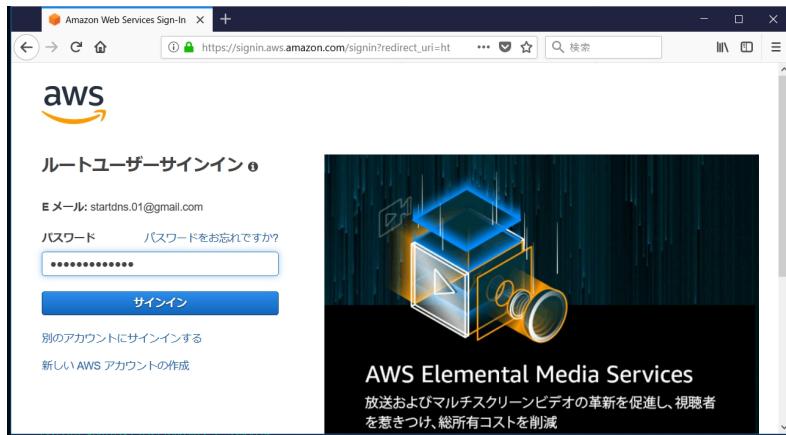
「コンソールにサインインする」を押す^{*5}と、マネジメントコンソールへのログイン画面が表示されます。(図 3.14) 先ほど登録した AWS アカウントの E メールアドレスを入力して「次へ」を押してください。



▲図 3.14 AWS のアカウントの E メールアドレスを入力

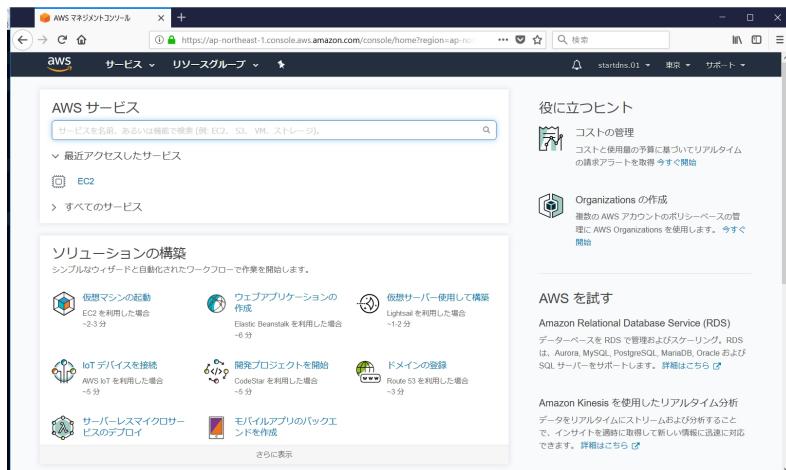
続いてルートユーザー サインインの画面（図 3.15）でパスワードを入力したら「サインイン」を押します。

^{*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。



▲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。



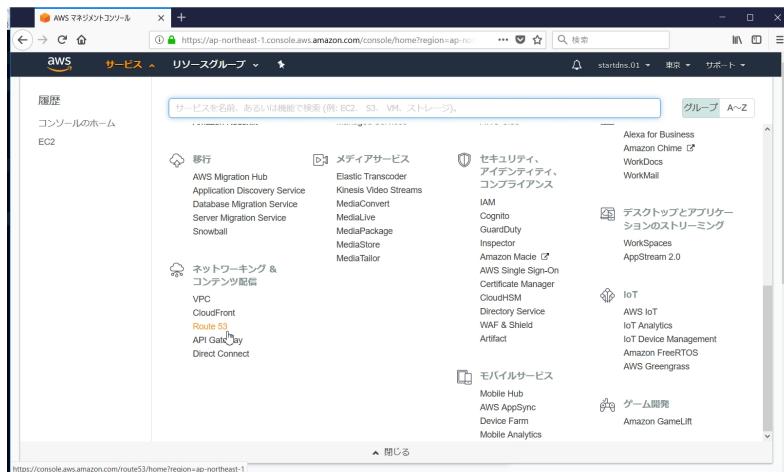
▲図 3.16 マネジメントコンソール (AWS の管理画面)

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 で買ったドメインのネームサーバを Route53 に変更する作業を行いましょう。先ずは Route53 で自分のドメインのゾーンを作成します。筆者は startdns.fun というドメインを使いますので、あなたも自分のドメイン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

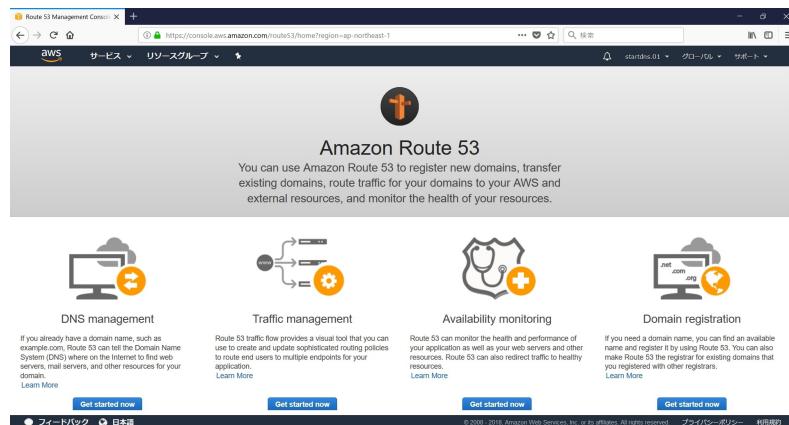
AWS のマネジメントコンソールで、左上の「サービス」を押すと AWS のサービス一覧（図 3.17）が表示されます。^{*6} 「ネットワーキング & コンテンツ配信」というグループにある Route53 を選択してください。



▲図 3.17 サービス一覧から Route53 を選択

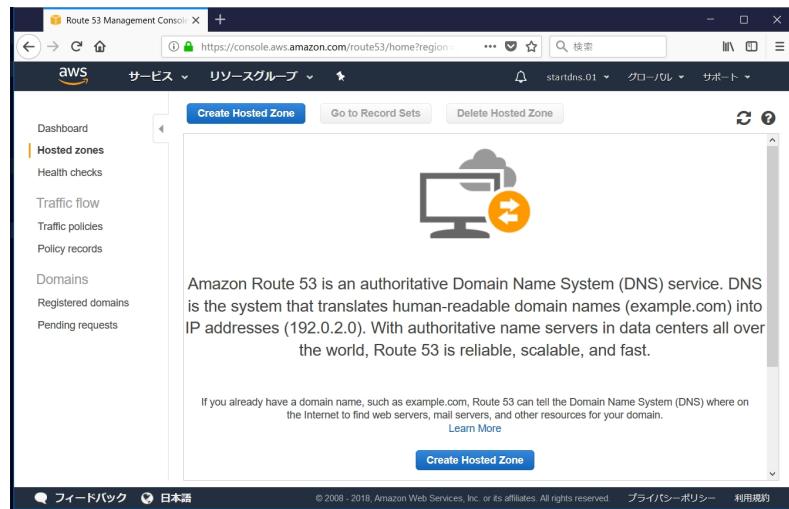
Route53 を押すといくつかメニューが表示される（図 3.18）ので、いちばん左側にある DNS management の「Get started now」を押してください。

^{*6} クラスマソッドの Developers.IO で 2018 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-2018-1/>



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

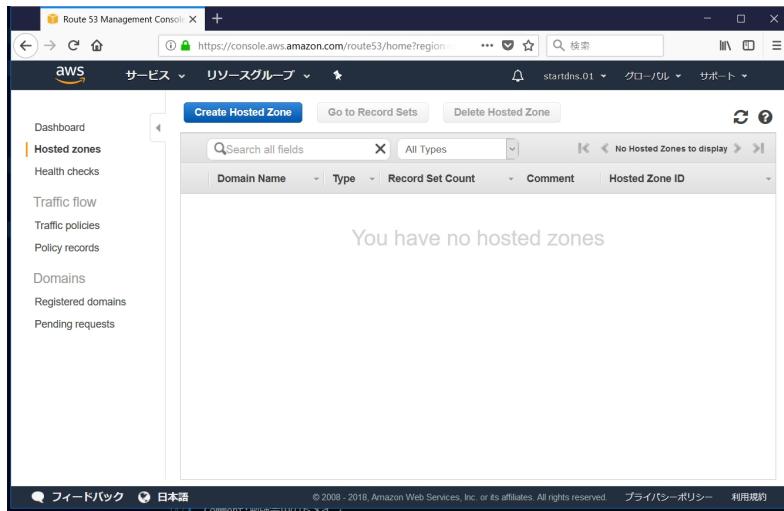
Route53 というネームサーバの中に自分のドメインのゾーンを作りたいので、ホストゾーンの画面（図 3.19）が表示されたら「Create Hosted Zone」を押してください。



▲図 3.19 「Create Hosted Zone」を押す

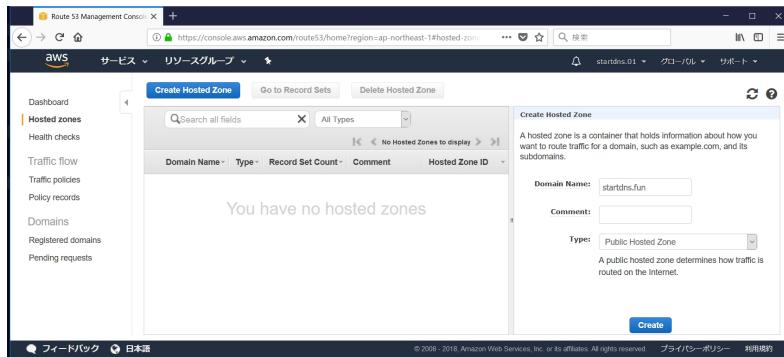
まだゾーンがひとつもないため「You have no hosted zones」と表示（図 3.20）されます。再び「Create Hosted Zone」ボタンを押してください。

3.3 ドメインのネームサーバを Route53 に変更



▲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を書きます。筆者は startdns.fun というドメインを買ったので、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

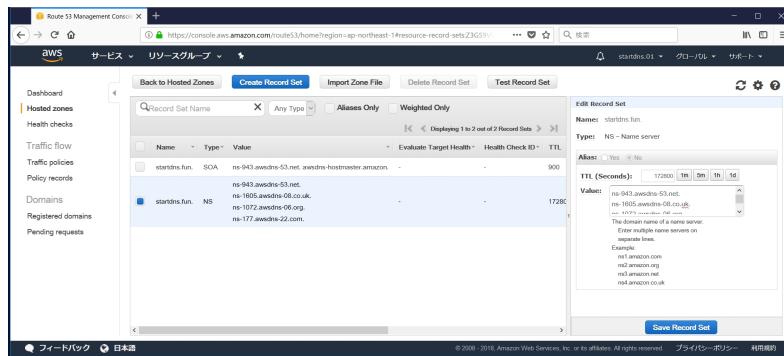
Type は Public Hosted Zone を選択（表 3.1）します。「Public Hosted Zone」にしておけば外から「このドメインに紐づいてる IP はなに？」と聞かれたときに Route53 のネームサーバが「IP は○○だよ」と返事をします。ここを「Private Hosted Zone for

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
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のゾーンが出来て、ゾーンの中にドメインのネームサーバを示す NS レコード（図 3.22）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 自分のドメインのゾーンとその中のリソースレコードが出来た

3.3.2 自分のドメインのネームサーバが何か確認

ではこれで自分のドメインのネームサーバが 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>



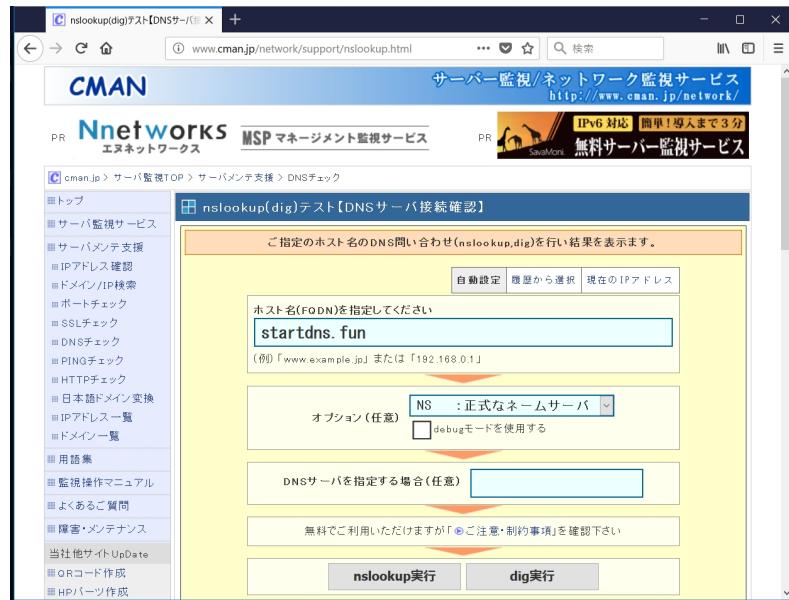
▲図 3.23 「dig」で検索

このサイトでは自分のドメイン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
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。



▲図 3.24 自分のドメインを入力してネームサーバを調べる

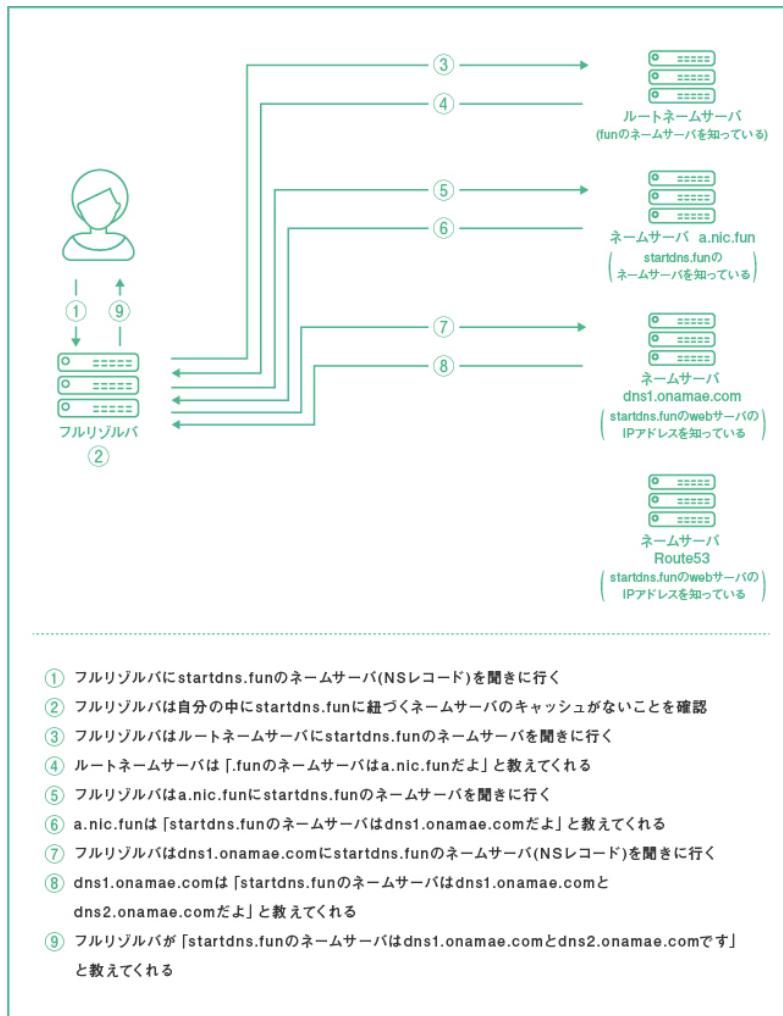
ちょっと下にスクロールして確認結果の ANSWER SECTION というところを見てください。次のように表示されましたか？

```
;; 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 というドメインのゾーンはお名前.com のネームサーバの中にあるよ」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のゾーンを作ったのに、なぜかドメインの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 で「startdns.fun については dns1.onamae.com に委任する」という設定になっているため、お名前.com のネームサーバが startdns.fun の権威を持っており、startdns.fun の NS レコードを問い合わせたときに誰も Route53 のネームサーバに聞きに来ない状態なのです。

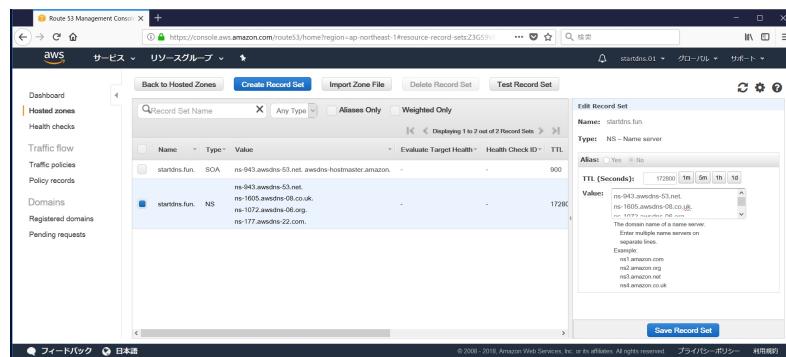
そもそもお名前.com でドメインを買うと、デフォルトの設定でネームサーバには次の 2つが設定されています。

- 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「ゾーンはお名前.com にある」という設定を「ゾーンは Route53 にある」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

それではドメインのネームサーバを 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^{*10}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。あなたのドメインの NS レコードをメモしてください。

^{*10} <https://www.onamae.com/>



▲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、ドメインを買ったときに発行されたお名前 ID^{*11}とパスワードを入力し、「ログイン」を押して管理画面の Domain Navi にログインします。

*¹¹ 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。



▲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。^{*12}



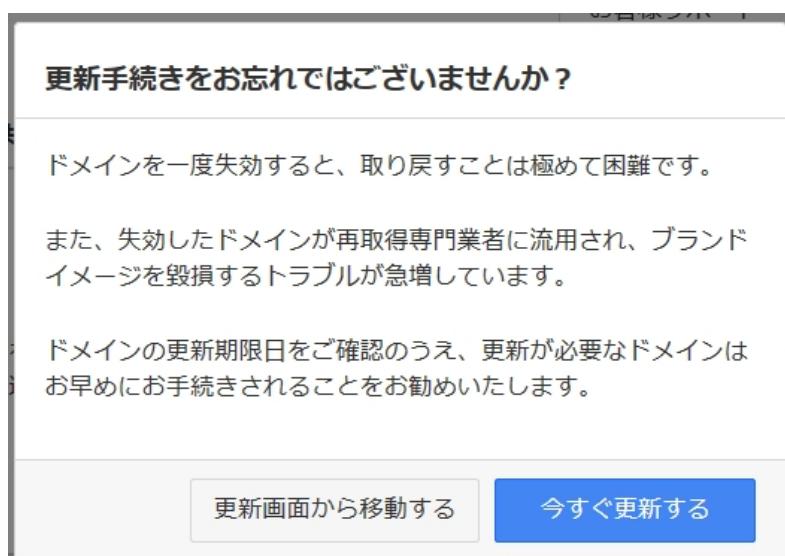
▲図 3.29 ドメイン Navi ドメイン契約更新

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

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

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「このページを離れる」を押してください。



▲図 3.30 ドメイン Navi 更新アラート

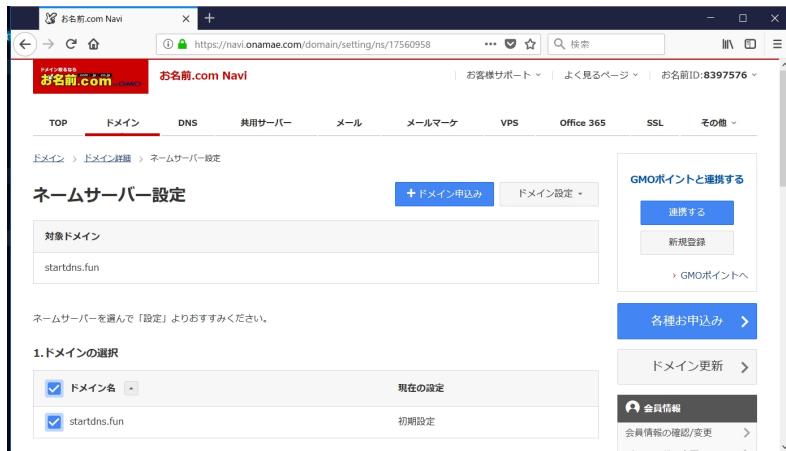
ドメイン一覧（図 3.31）には自分が買ったドメイン（筆者なら startdns.fun）があります。買ったばかりのドメインはネームサーバが「初期設定」になっているので、そこをクリックして「ネームサーバー設定」の画面に進みます。

第3章 AWS のネームサーバ (Route53) を使ってみよう



▲図 3.31 ドメイン Navi ドメイン一覧

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。



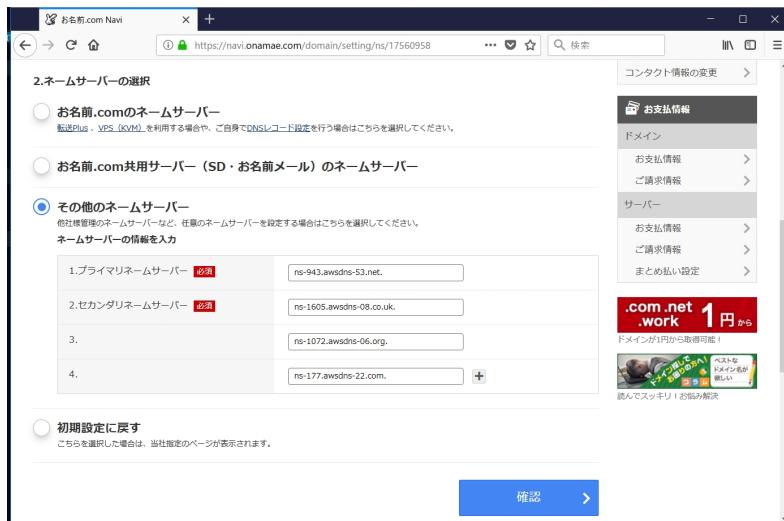
▲図 3.32 ドメイン Navi ネームサーバー設定

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「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。

3.3 ドメインのネームサーバを Route53 に変更



▲図 3.33 ネームサーバーの選択 変更前



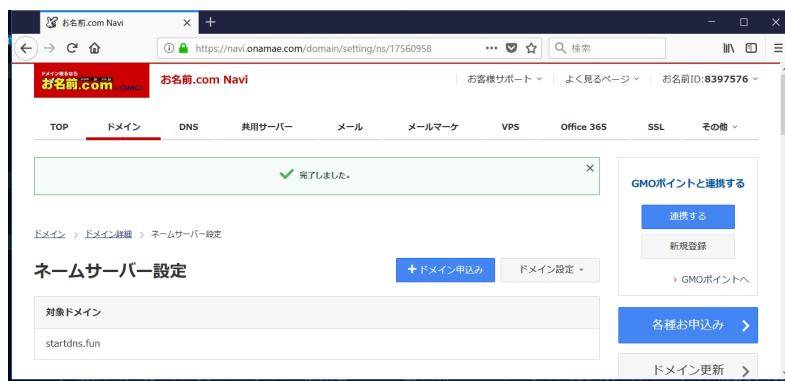
▲図 3.34 さきほどメモした Route53 のネームサーバを 4 つ書く

「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、少し経つてから「[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

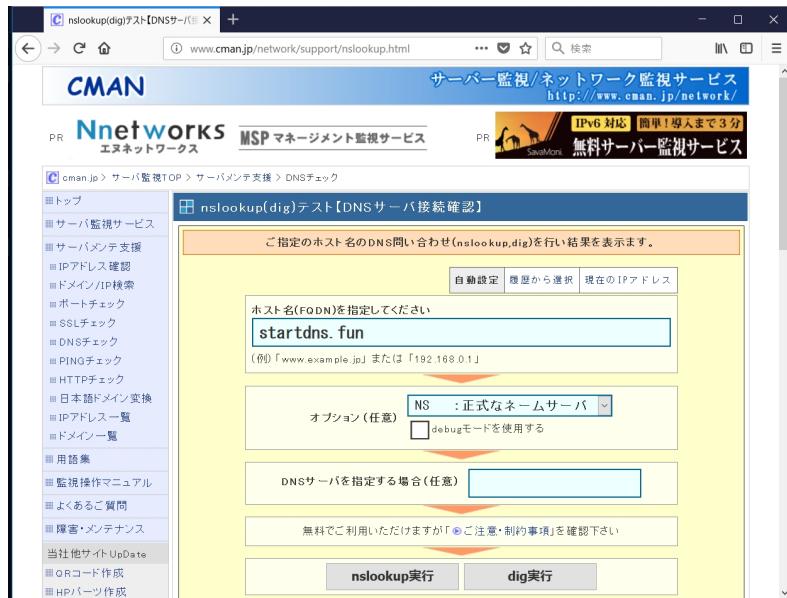
ところで今、ネームサーバをお名前.com から Route53 に変更しましたが、古い設定の TTL が過ぎるまでフルリゾルバにはキャッシュが残っています。

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】^{*13}」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、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。

^{*13} <https://www.cman.jp/network/support/nslookup.html>



▲図 3.37 再び自分のドメイン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。

```
;; 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 の変更は「浸透に時間がかかる」と言われますが、これは 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、新しいリソースレコードの情報が反映されない、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」はただ闇雲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分かりますし、事前に TTL を短くしておくことで反映にかかる時間を短くすることもできま

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げるので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、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。

3.3.5 【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「ドメインがもうすぐ有効期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. AWS の 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（ドメインを購入するお店）をお名前.com から AWS の 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。

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インストールされていなければ使えません。Mac ユーザであれば元々インストールされているので、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 12 か月間は毎月 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、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ぼちぼ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。^{*3}

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】」^{*4}というページや各レジストリの Whois 情報検索サイトで疑似的に 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} 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 12 か月間ずっと無料ということです。

^{*4} <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

ドメインに紐づく IP を調べたいとき、あるいは IP に紐づくドメインを調べたいときは dig コマンド^{*5}を使用します。

たとえば本著を初めて頒布する技術書典 4 のウェブサイトは <https://techbookfest.org/event/tbf04> 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-9.8.2-0.62.rc1.el6_9.5 <>> techbookfest.org a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48654
;; 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      216.239.32.21
techbookfest.org.    300     IN      A      216.239.34.21
techbookfest.org.    300     IN      A      216.239.36.21
techbookfest.org.    300     IN      A      216.239.38.21

;; AUTHORITY SECTION:
techbookfest.org.   75009   IN      NS      ns-cloud-b1.googledomains.com.
techbookfest.org.   75009   IN      NS      ns-cloud-b4.googledomains.com.
techbookfest.org.   75009   IN      NS      ns-cloud-b2.googledomains.com.
techbookfest.org.   75009   IN      NS      ns-cloud-b3.googledomains.com.

;; ADDITIONAL SECTION:
ns-cloud-b1.googledomains.com. 334209 IN A      216.239.32.107
ns-cloud-b1.googledomains.com. 334209 IN AAAA   2001:4860:4802:32::6b
ns-cloud-b2.googledomains.com. 334209 IN A      216.239.34.107
ns-cloud-b2.googledomains.com. 334209 IN AAAA   2001:4860:4802:34::6b
ns-cloud-b3.googledomains.com. 334209 IN A      216.239.36.107
ns-cloud-b3.googledomains.com. 334209 IN AAAA   2001:4860:4802:36::6b
ns-cloud-b4.googledomains.com. 334209 IN A      216.239.38.107
ns-cloud-b4.googledomains.com. 334209 IN AAAA   2001:4860:4802:38::6b

;; Query time: 148 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Mar 11 11:51:13 2018
```

^{*5} 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。grope 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。

```
;; 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^{*6}のようなコマンドなのでちょっと聞いていただけで「調べてきた結果を教えまし！ まずキミの質問はこれでし！ このドメインに紐づく IP はこれとこれとこれとこれなのでし！ ちなみにこれを教えてくれたネームサーバの名前はこれでし、IP はこっちなのでし！ ちなみにフルリゾルバは 127.0.0.1 で調査には 148msec 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  
Server:      172.31.0.2  
Address:     172.31.0.2#53  
  
Non-authoritative answer:  
Name:  techbookfest.org
```

^{*6} 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い可愛い。

```
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^{*7}」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 っていぢれなくなるんでしょ？」という認識かと思いますが、実際は BIND^{*8}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^{*9}」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。

```
ドメインの持ち主を調べる
$ whois ドメイン
```

```
IP アドレスの持ち主を調べる
$ whois IP アドレス
```

^{*7} 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。

^{*8} 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 DNS サーバへの乗り換えもよく聞かれます。

^{*9} nslookup is no longer to be treated as deprecated. Remove "deprecated" warning message.

4.3.1 【ドリル】ドメイン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^{*10}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

答え _____

^{*10} <http://tigerkyoro.jp/>

解答

正解は B です。whois コマンドを叩くと「Expires on」という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、有効期限が 2019/01/31 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]           2019/01/31
[Status]                Active
[Last Updated]          2018/02/01 01:05:08 (JST)

Contact Information:
[Name]                  Clarivate Analytics (Japan) Co.,Ltd.
[Email]                 admin@thomsonbrandy.jp
[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
```

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。^{*11}するとそのドメインに紐づく IP アドレスが表示されるはずです。

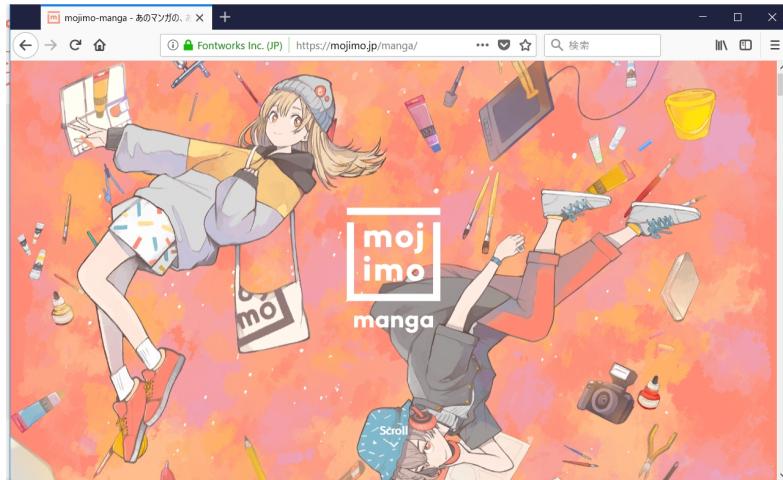
```
$ dig ドメイン名 a +short
```

4.4.2 【ドリル】ウェブサーバはどこにある？

問題

あなたは「mojimo-manga」（モジモマンガ）というサイト (<https://mojimo.jp/manga/>) の運用担当になりました。^{*12}（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。



▲図 4.2 mojimo-manga - あのマンガの、あのアニメの、あのフォントが使える！

^{*11} 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。

^{*12}もちろん例えです。本著及び著者は例え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。

ウェブサーバはどこのクラウド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 B です。

先ず「dig ドメイン名 a +short」でドメインから IP を引いてみましょう。サイトの URL が <https://mojimo.jp/manga/> なので、確認すべきドメインは mojimo.jp です。

```
$ dig mojimo.jp a +short  
46.51.255.231  
54.248.233.139
```

IP アドレスは分かりましたが、IP だけではまだど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 の持ち主を調べてみましょう。^{*13}

「whois 46.51.255.231」を叩いてみると「descr: Amazon Web Services, Elastic Compute Cloud, EC2」と出ますので、この IP は AWS の EC2 のものであることが分かります。これで mojimo-manga のサイトが AWS の EC2 上で動いていることが分かりました。

```
$ whois 46.51.255.231  
(中略)  
descr:      Amazon Web Services, Elastic Compute Cloud, EC2  
country:    JP  
admin-c:    ADSI2-RIPE  
tech-c:     ADSI2-RIPE  
status:     ASSIGNED PA  
mnt-by:    MNT-ADSI  
mnt-by:    NN20150720  
created:   2017-11-07T20:48:28Z  
last-modified: 2017-11-07T20:49:00Z  
source:    RIPE
```

^{*13} IP アドレスは 2 つありますがどちらでも構いません。

(後略)

【コラム】dig のオプションは略せる

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、それぞれ後の何文字かを省略して「+shor」や「+norec」でも実行可能です。

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 - ブラブル事例とその解決のポイント -」^aという資料をお勧めです。

^a <https://dnsops.jp/event/20140626/dns-beginners-guide2014-mizuno.pdf>

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。

```
$ 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^{*14}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 までたどり着くことができます。

```
$ dig nintendo-co-jp.mail.protection.outlook.com a +short  
23.103.139.138  
23.103.139.170
```

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つたらどうなるのでしょうか？ メールを送ろうとしたとき、そのドメインの MX レコードが存在しなかつ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。^{*15}

つまり「今回は <http://example.co.jp/> というサイトを開くだけでメールを送受信する要件は全くないので 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。Postfix^{*16}は CentOS 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うっかりウェブサーバで 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^{*17}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つてよいかも知れません。

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

*14 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。

*15 <https://tools.ietf.org/html/rfc5321#section-5.1>

*16 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。

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

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メール^{*18}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した。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 社ではなく 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

^{*18} 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つ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。

やら GMO ペパボがやっているヘテムル^{*19}というクラウドサービスのメールサーバのようです。

○○@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 というドメインは誰の持ち物なのでしょうか？ 今度は whois でドメイン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 (2019/02/28)
[Registered Date]          2000/02/14
[Connected Date]            2000/02/16
[Last Update]              2018/03/01 01:05:12 (JST)
```

Whois で調べてみるとこのドメインは「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 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。

^{*19} <https://heteml.jp/>



▲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」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、これらのドメインにはいずれもちゃんと持ち主^{*20}がいます。

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ー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

^{*20} たとえば「test.com」というドメインは 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 というドメインは有限会社教育評価研究所という会社が持ち主です。

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しまって、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。

ありがとうございま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 と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ばつ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、他人のドメインも勝手に使わないようにしましょう。

例示やテスト専用のドメインを使おう

他人が持っているドメイン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を取得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」というものが定められて^{*21}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を使いましょう。

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

これらのドメイン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、リソースレコードは MX レコードも A レコードも設定されていないため、予期せぬ第三者へうっかりメールが飛んでしまうことは避けられます。

4.4.5 NS レコード

ドメイン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わかります。

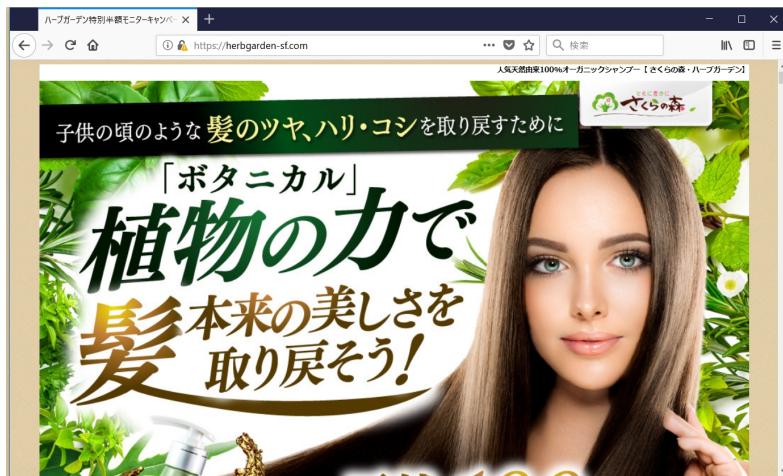
^{*21} RFC2606 で定められている example.com や example.net、もしくは JPRS が定めている example.jp や example.co.jp などを使いましょう。 <https://jprs.jp/faq/use/>

```
$ dig 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)



▲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-390.awsdns-48.com.
ns-1487.awsdns-57.org.
ns-1984.awsdns-56.co.uk.
ns-692.awsdns-22.net.
```

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^{*22}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」なのか否かを確認する手段として、SPF^{*23} 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
```

^{*22} 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。

^{*23} Sender Policy Framework の略。

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。^{*24}

たとえばポケモンだいすきクラブ^{*25}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 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"
```

include は引数で渡しているドメインの 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"
```

一番最初の「+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すぎるので省略しますが、要はここに書いてある IP 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、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。

^{*24} <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。

^{*25} ポケモンの情報サイト。ミミッキのうたは名曲です。 <http://www.pokemon.jp/>

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 (<http://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」という追加希望が出てきました。

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、SPF レコードにメールを送っているサーバの IP アドレスを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 PTR レコード

A レコードは前述のとおりドメインから 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を逆引きできるレコードのことを PTR レコード^{*26}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 が SPF レコードに登録されていること」だけでなく「メール送信元の IP からドメインの逆引きができる」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のリソースレコードと、PTR レコードのような IP 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 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 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。

4.4.10 CNAME レコード

ときどきドメインから IP を引こうとして dig で A レコードを調べたのに、こんな風にドメインと IP 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^{*27}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が出てくるのでしょうか？ +short オプションを外して、この

^{*26} Pointer record の略。

^{*27} <https://aibo.sony.jp/>

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-9.8.2-0.62.rc1.el6_9.5 <>> aibo.sony.jp a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58537
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;aibo.sony.jp.           IN      A

;; ANSWER SECTION:
aibo.sony.jp.          300     IN      CNAME   cs1018.wpc.omicroncdn.net.
cs1018.wpc.omicroncdn.net. 3091 IN      A       152.195.38.205

;; AUTHORITY SECTION:
omicroncdn.net.        172518  IN      NS      ns2.omicroncdn.net.
omicroncdn.net.        172518  IN      NS      ns1.omicroncdn.net.

;; ADDITIONAL SECTION:
ns1.omicroncdn.net.    172518  IN      A       72.21.80.5
ns1.omicroncdn.net.    172518  IN      AAAA   2606:2800:1::5
ns2.omicroncdn.net.    172518  IN      A       72.21.80.6
ns2.omicroncdn.net.    172518  IN      AAAA   2606:2800:1::6

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Mar 19 22:09:15 2018
;; MSG SIZE  rcvd: 209
```

ANSWER SECTION を見てみましょう。aibo.sony.jp の CNAME 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 レコードに紐づく IP アドレスを返します。

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
```

4.4.11 【ドリル】CNAME の調べ方と使いどころ

問題

キドキド^{*28}のサイト（図4.5）のドメイン「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 _____

^{*28} <https://kidokid.bornelund.co.jp/>

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 でも構いません。CNAME レコードは CDN*²⁹を使うときによく利用されます。

```
$ 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ぞれ A レコードを設定していた場合、ウェブサーバを引っ越しとなったら 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 レコードのみです。

4.4.12 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^{*30}。

```
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

*²⁹ Contents Delivery Network の略。アクセスしてきたエンドユーザー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ーからのアクセスが分散されるため、TVCN や LINE 砲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。

*³⁰ A CNAME record is not allowed to coexist with any other data. <http://www.ietf.org/rfc/rfc1912.txt>

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全く保証されません。

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。

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^{*31}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したか？）

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^{*32}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。^{*33}

^{*31} 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。

^{*32} 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。

^{*33} 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。

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、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。

4.4.13 グルーレコード

ところでドメインを買ったとき、お名前.com のネームサーバや Route53 を使う他に自分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 なのでフルリゾルバはいつまでも 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 DNS コンテンツサーバ（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。

第 5 章

トラブルシューティング

ドメインや 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？

サイトのドメインを www ありかなしかはつきり決めておかなかつたことで、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。

そもそもドメインを www.example.co.jp のように頭（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 は全く別の FQDN です。それと同じように www.example.co.jp と example.co.jp も全く別の FQDN です。

名前が途中まで一緒だとなんだか近しいような気がしてしまいますが、北海道と奄美大島が全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 プッシュで「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。

気の利いたサイト制作会社であれば、依頼側が何も言わなくてもドメイン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）

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！

^{*1} Fully Qualified Domain Name の略で、日本語にすると完全修飾ドメイン名。正確を期するのであれば FQDN には TLD である jp までではなく、さらにルートを表す（ドット）も含むべきですが、ここでは TLD まで表記されていれば FQDN とします。余談ですが FQDN に対して、www や www.example のように一部を抜き出した表記は相対ドメイン名、あるいは不完全なドメイン名と呼びます。

^{*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。

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 が異なるサイトを全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ドメインとしての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つきり決めましょう。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、色んなところで「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。

そして今回 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.dev のドメインは一般発売していない。ということは今.dev を使っている人は全員、ドメイン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」ということなんだと推察してい

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

ます。

.dev だけなく.app という TLD も同様のようですので、これらのドメインがもう HTTPS でしか使えないということは、Google は.dev を.com や.net のように普通に売るのではなく、何か特別な用途で使わせるつもりなのではないか、と思われます。

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」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 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ぬ・・・。そんなときは以下のようないくつかの移管前後の表（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 のようにドメインの管理は A 社から B 社に変わることで、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 にドメインを移動することができます。

ドメインの管理が 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レコードを 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 1 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1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→ドメインの管理）で移管を進めようすると、5月1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1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やない？と思われるかもしれません、古いウェブサーバへのアクセスを example.net にリダイレクトしたところでまだ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 1 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 C 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 C 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、クライアントも C 社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、無駄に「浸透」を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で試してみましょう。筆者は startdns.fun を使ってみますので、あなたもお名前.com で買ったドメイン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

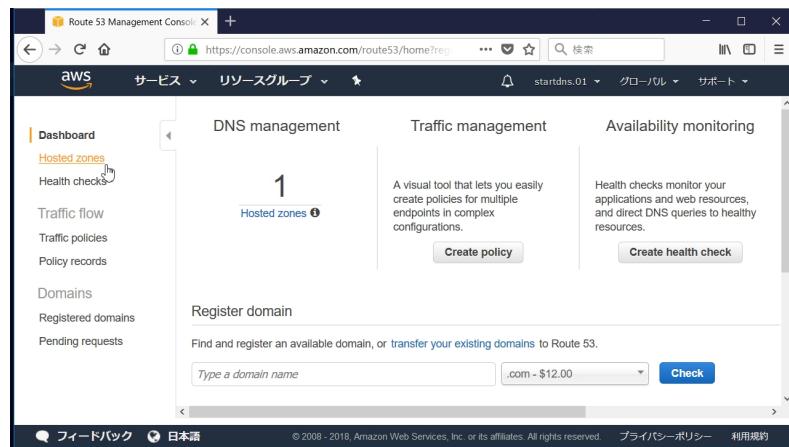
;; AUTHORITY SECTION:
startdns.fun.        900    IN      SOA     ns-943.awsdns-53.net. awsdns-hostma
                               2018032100000
                               3600
                               1800
                               604800
                               86400

;; 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。

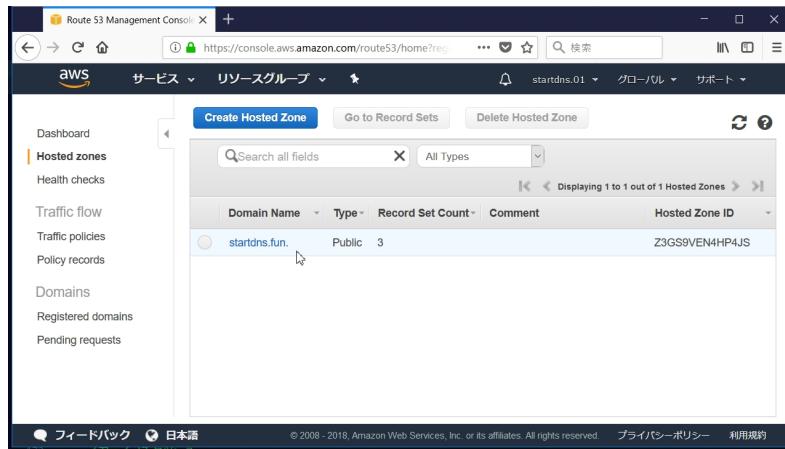
今回の test.startdns.fun の場合、SOA レコードの TTL は 900 で、SOA の MINIMUM は 86400 ですので、ネガティブキャッシュの保持時間は 900 秒 (15 分) となります。ではネガティブキャッシュが残っている 15 分の間に Route53 で test.startdns.fun の A レコードを作つてみましょう。

<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（筆者なら startdns.fun）を選択（図 5.2）します。



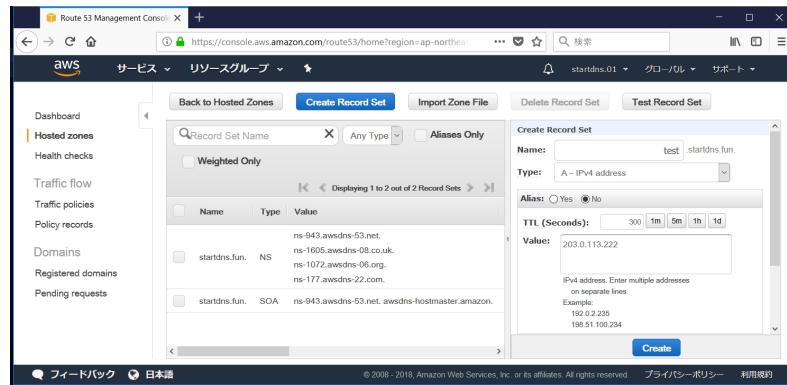
▲図 5.2 自分のドメインを選択

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

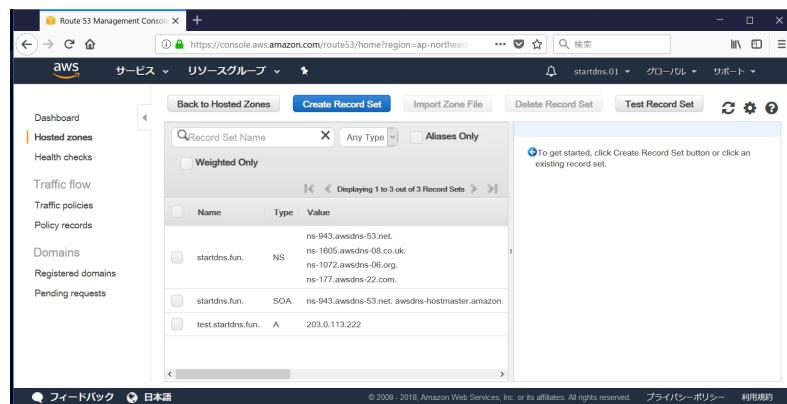
項目	入力するもの
Name	test
Value	203.0.113.222

^{*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>



▲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
;; 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-hostmas
;; 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「そのドメインのリソースレコードは存在しないよ」と言われてから、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引いてみます。

```
$ 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-hostmas
;; 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
```

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
;; 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。「もう A レコード用意できたかな？」と確認するときは、ネガティブキャッシュが残っても困らない環境で dig を叩いてみるのがお勧めです。

5.5 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 レコードの略で、CAA レコードに「そのドメイン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つても認証局は証明書を発行せず、ドメイン管理者に対して意図しない発行申請があつ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 レコードまで遡って順に確認していきます。^{*5}

そのため証明書を発行しようとした FQDN では CAA レコードは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、そのドメインの TLD までさかのぼって 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。

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^{*6} 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全く別物です。

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。^{*7}

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

^{*5} 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。

^{*6} 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。

^{*7} <https://dev.classmethod.jp/cloud/aws/amazon-dns-threshold-exceeded-action/>

秒にして返す^{*8}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、AWS で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。

^{*8} <https://dev.classmethod.jp/cloud/aws/dig-route53-begginer/>

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。そんな気持ちで作りました。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っ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
つながっていきの 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歩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まったので、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。

この本が 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。Twitter やブログなどで「買ったよ！」「読んだよ！」などと気軽につぶやいてくださると飛び跳ねて喜ぶと思います。

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。

2018 年 4 月
mochikoAsTech

Special Thanks:

レビュー

- 深澤俊

参考書

- ワンストップ！ 技術同人誌を書こう 親方@oyakata2438
- 技術書をかこう！ ～はじめての ReVIEW～ Techbooster

著者紹介

mochiko / @mochikoAsTech

ウェブ制作会社のシステムエンジニア。モバイルサイトの Web アプリケーションエンジニア、SIer とソーシャルゲームの広報を経て、2013 年よりサーバホスティングサービスの構築と運用を担当。現在は再び Web アプリケーションエンジニアとしてシステム開発に従事。「分からぬ気持ち」に寄り添える技術者になれるように日々奮闘中。

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

2018年4月22日 技術書典4版 v1.0.0

著者 mochikoAsTech

デザイン Hikaru Wakamatsu / Shinya Nagashio

発行所 好きなコマンドは dig です

(C) 2018 mochikoAsTech