

初心者からマニアまで必読

AWS IAM の マニアックな話

佐々木拓郎 著

商業誌でベストセラーになった AWS 本の著者が、商業誌では出せなかった 1 機能に徹底的にこだわって書いた本です。 AWS の認証認可の機能である Identity and Access Management (IAM) の機能・デザインパターン・運用まで秘伝のタレを全公開します。

AWS の薄い本

IAM のマニアックな話

佐々木拓郎 著

2019-09-22 版 発行

はじめに

本書の目的

「AWS の薄い本 IAM のマニアックな話」を手にとっていただき、ありがとうございます。本書は IAM の機能に絞って解説するという機能特化本です。

私は今まで、商業誌で「AWS の基本機能」、「サーバレス」、「業務システム」、「試験対策」をテーマに 4 冊の AWS 本を書きました。テーマに沿って必要な機能を紹介していくというスタイルだったのですが、今回は逆に機能を元に解説するというスタイルに挑戦しています。

それでなぜ、IAMなのでしょうか？ AWS が不正利用されて 100 万円の請求が来たというようなニュースを、ネットで時々目にします。原因の多くが IAM のアクセスキーを GitHub に誤ってコミットしてしまい、そのキーを不正利用されたケースです。そういう事態を防ぐために正しく IAM を知って貰いたいのです。

IAM は、AWS の利用権限を管理する極めて重要な機能です。AWS には多種多様な機能があり、IAM はそれに応じて様々な記述方法で権限を設定できるようになっています。その分設定項目が多く、IAM は難しいと印象を持っている人も多いのではないでしょうか。また、ある程度 IAM に習熟した人にとっては、他の人かどのような方針で IAM の権限設計をしているか気になるという人も多いでしょう。そんな要望に応えるべく、私の考える IAM のベストプラクティスをまとめてみました。

対象読者

- AWS アカウントのルートユーザーで AWS を使っている人
- IAM ユーザーで使ってるけど、Administrator 権限しか付与したことがない人
- 企業内で AWS を導入済みで、異なるロールの様々な人に IAM の権限を振り分ける必要がある人
- お一人様 AWS 利用だけど、AWS を不正利用されないようにちゃんと IAM を使いたい

本書で得られること

- IAM の機能に関する一通りの知識と、権限設計のノウハウ
- 設計する上でのセキュリティの考慮点と運用について
- IAM マニアの称号

お問い合わせ先

本書に関するお問い合わせ：twitter: @dkfj

免責事項

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いた開発、製作、運用は、必ずご自身の責任と判断によって行ってください。これらの情報による開発、製作、運用の結果について、著者はいかなる責任も負いません。

(佐々木 拓郎)

目次

はじめに	3
本書の目的	3
対象読者	4
本書で得られること	4
お問い合わせ先	4
免責事項	4
第1章 AWSとIAM	11
1.1 認証と認可	12
1.2 AWSのアカウント種類	13
1.3 AWSアカウント	13
1.4 IAMユーザー	14
1.5 注意とお願い	14
第2章 IAMの機能	15
2.1 IAMユーザー	16
2.2 IAMグループ	19
2.3 IAMポリシー	20
AWS管理ポリシーとカスタマー管理ポリシーの使い分け	21
2.4 IAMロール	23
2.5 パーミッション・バウンダリー	23

目次

2.6 IAM の機能のまとめ	25
AWS アカウントと IAM の関係	26
第 3 章 IAM チュートリアル	27
3.1 IAM ポリシーの作成	28
すべての権限をもった IAM ポリシーの作成	29
IP アドレス制限の追加	31
反転させて特定 IP アドレス以外のアクションを拒否	32
3.2 IAM グループの作成	34
IAM グループにポリシーのアタッチ	35
IAM の許可・拒否ポリシーの挙動	36
3.3 IAM ユーザーの作成	37
IAM ユーザーに権限の付与	38
3.4 クロスアカウントロールの作成	39
IAM ロールに権限の付与	40
スイッチ用 URL の取得とロールに切り替え	42
IAM ロールにスイッチ元の IAM ユーザー制限の追加	45
3.5 チュートリアルのまとめ	47
第 4 章 IAM ポリシーのデザインパターン	49
4.1 ホワイトリスト・パターン	50
ホワイトリスト・パターンのメリット	51
ホワイトリスト・パターンのデメリット	52
ホワイトリスト・パターンのユースケース	52
4.2 ブラックリスト・パターン	53
ブラックリスト・パターンのメリット	56
ブラックリスト・パターンのデメリット	56
ブラックリスト・パターンのユースケース	56
4.3 ハイブリット・パターン	57

目次

ハイブリット・パターンのメリット	58
ハイブリット・パターンのデメリット	58
ハイブリット・パターンのユースケース	59
1 ポリシーで管理すべきか、グループで組み合わせるか	59
4.4 IAM ポリシーのまとめ	60
最小権限の探求	61
第 5 章 IAM グループのデザインパターン	63
5.1 複数グループに所属	64
5.2 グループ内に複数ポリシー	65
5.3 IAM グループのまとめ	66
IAM グループの階層構造について	67
第 6 章 IAM とセキュリティ	69
6.1 IAM ベストプラクティスの遵守	69
IAM ベストプラクティスのリスト	70
6.2 ルートユーザーを使わない	71
6.3 IAM に関する権限付与	71
ユーザー自身のパスワードと MFA の設定を許可する	72
6.4 Lambda のリソースベースの権限	74
6.5 インターネット公開系の権限	75
EC2 の権限範囲の問題	77
6.6 VPC 内からのアクセス	78
6.7 アクセスキーの原則禁止	80
IAM ロールを利用する	80
6.8 Capital One の情報流出事件に思うこと	81
事件の概要と攻撃手法	81
流出の原因	82
具体的な設定手順	83

目次

6.9	IAM とセキュリティのまとめ	85
	IP アドレス制限の是非とゼロトラストセキュリティ	86
第 7 章	IAM の運用	87
7.1	IAM の運用の目的	87
	IAM 運用の目的	88
	目的の達成のために	88
7.2	役割と責任範囲の明確化	88
7.3	AWS アカウントの管理	90
7.4	IAM ユーザーの管理	90
	AWS マネジメントコンソールから利用する IAM ユーザー (人間用)	90
	CLI やプログラムから利用する IAM ユーザー (プログラ ム用)	91
7.5	アクセスキーの管理と CLI	92
7.6	MFA 未利用時に権限を制限し、MFA 利用を促す	94
7.7	マルチ AWS アカウントでの運用	97
7.8	IAM 運用のまとめ	98
第 8 章	IAM と CloudFormation	99
8.1	IAM と CloudFormation	99
8.2	CFn の分割単位・依存関係	100
8.3	CloudFormation と IP 制限	102
	ライフサイクルで考える	103
第 9 章	IAM のテンプレート集	105
	テンプレートのダウンロード	105
9.1	共通系ポリシー	106
	自身のパスワードと MFA の設定権限	106
	IP 制限と MFA の必須化のポリシー	106

目次

9.2	管理者グループ	108
9.3	ネットワーク管理者グループ	110
9.4	開発者グループ	111
9.5	オペレーターグループ	112
9.6	経理担当者グループ	113
9.7	お一人様 AWS	114
9.8	IAM のテンプレートのまとめ	115
第 10 章 IAM 以外の AWS サービスの活用		117
10.1	AWS Organizations（組織アカウント）	117
10.2	CloudTrail と Config	118
10.3	Amazon GuardDuty	119
10.4	AWS Control Tower と AWS Security Hub	119
10.5	まとめ	120
付録 A アカウント開設時の設定チェックリスト		121
後書き		123

第 1 章

AWS と IAM

セキュリティは、AWS がもっとも重視している項目の一つです。一口にセキュリティといっても、ネットワークの観点であったり、サーバーの観点であったりと、いろいろな観点があります。AWS Identity and Management（以下、IAM）は、AWS 利用に関する認証と認可を司るサービスです。認証認可については後述しますが、AWS の操作権を管理する機能になります。

AWS の利用者が増える裏で、AWS を不正利用されることでの金銭的な被害も発生しています。その原因の多くが、IAM の運用を適切にしていかなかったためです。IAM を乗っ取られると、どれほどネットワークやサーバーセキュリティを固めていても、簡単に無効化されます。そういう意味で、AWS を利用する上でもセキュリティの根幹をなすサービスの一つといえるでしょう。本書では、確かな知識のもとに安心して AWS を運用できるように、IAM を徹底的に解説します。

1.1 認証と認可

IAMの説明にはいる前に、まず認証と認可について確認してみましょう。認証は本人性の確認（Authentication）であり、認可はリソースに対する利用権限の付与（Authorization）です。ファイルサーバの利用で考えると、ユーザー・パスワードの組み合わせで誰が利用しているかを判別するのが認証で、認証したユーザーに対してどのフォルダを参照や書込などどのように利用させるかが認可です。多くのシステムでは、認証と同時に認可を行いますが、本質的には別々の機能です。

認証と認可

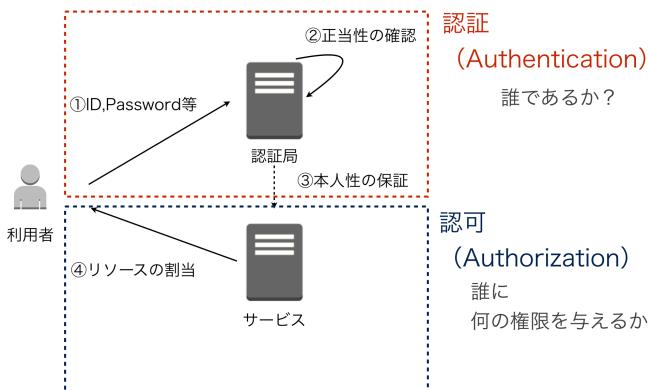


図 1.1: 認証と認可

IAMはAWS自身に対して、IAMユーザーにより認証の機能を、IAMポリシーを介して認可の機能を提供します。なお、AWSを利用して構築したシステム／サービスの認証と認可を担当するのは、Amazon Cognitoという別のサービスです。

1.2 AWS のアカウント種類

認証と認可の次に、AWS のアカウントの種類についてです。AWS には「AWS アカウント」と「IAM ユーザー」と呼ばれる 2 種類のアカウントがあります。AWS アカウントとは、AWS へサインアップする時に作成されるアカウントです。このアカウントは、ルートユーザーとも呼ばれ、AWS のすべてのサービスをどこからでも利用可能です。一方、IAM ユーザーとは、AWS を利用する各利用者向けに作成されるアカウントです。最初は IAM ユーザーは存在していないため、AWS アカウントでログインし、まず IAM ユーザーを作成する必要があります。

また、複数の AWS アカウントを管理する為に AWS Organizations（組織アカウント）という機能も追加されました。組織アカウントを利用することで、従来から存在した複数のアカウントの請求をひとまとめにする一括決済のみならず、サービスコントロールポリシー（SCP）と呼ばれる AWS アカウント単位での利用可能なサービスの制限などができるようになります。AWS Organizations は、本書の対象外です。面白く有用な機能なので、また別の機会で解説します。

1.3 AWS アカウント

AWS アカウントのユーザーはルートユーザーとも呼ばれ、AWS の全サービスに対してネットワーク上のどこからでも操作できる権限を持っていきます。非常に強力なアカウントであるため、取り扱いには十分注意する必要があります。AWS でシステムを構築・運用する場合、**AWS アカウントを利用は極力避け、IAM ユーザーを利用してください**。また、AWS アカウント単体では、ルートユーザーに IP アドレス制限など利用シーンを制限する方法はないです。そのため、**2 要素認証（MFA）の設定**をすることをお勧めします。

1.4 IAM ユーザー

IAM ユーザーは、Web コンソールや API を通じての AWS の操作に使用します。各 IAM ユーザーに対して、操作を許可する（しない）サービスが定義します。各 IAM ユーザーの権限を正しく制限することで、AWS をより安全に使用することができます。たとえば、EC2 インスタンスを起動、停止する権限だけを与えて、終了はできないユーザーを作成したり、ネットワーク（セキュリティグループや VPC、Route53 など）に関する権限のみを持つネットワーク管理者用ユーザーを作ることができます。

IAM ユーザーの管理は、セキュリティの要になります。ネットワークやサーバーに安全対策しても、IAM ユーザーの管理が杜撰な場合、簡単に AWS 自体を乗っ取られてしまいます。IAM ユーザーは、それくらい重要なものだと心得てください。IAM にはユーザー以外にも様々な機能があります。それでは次章で、IAM の具体的な機能をみていきましょう。

1.5 注意とお願い

次章以降で、IAM の機能や設計手法を紹介していきます。その中で IAM のポリシー例なども随時紹介していきますが、ポリシー記述の文法的な部分についてはメインでは扱いません。書式の詳細などは、公式ページ (https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_policies.html) を参照してください。



図 1.2: IAM JSON ポリシーリファレンス

第 2 章

IAM の機能

IAM は AWS 操作をセキュアに行えるように、認証・認可の仕組みを提供します。IAM には大小様々な機能がありますが、まず次の 5 つの機能の認知と理解しておくことが重要です。

- IAM ユーザー
- IAM グループ
- IAM ポリシー
- IAM ロール
- パーミッション・バウンダリー

ユーザーやグループ、ポリシーなど名前から機能が想像できるものもありますが、ロールやパーミッション・バウンダリーと一見しただけで何だろうと思うようなものもあります。それでは、それぞれの役割について確認してみましょう。

2.1 IAM ユーザー

認証を司るのは、IAM ユーザーです。IAM ユーザーの認証は 2 種類あり、ID・パスワードの組み合わせと、アクセスキー ID・シークレットアクセスキーの組み合わせが利用できます。AWS マネジメントコンソールへのログインは ID・パスワードを利用し、API 操作についてはアクセスキー ID とシークレットアクセスキーを利用します。認証をより安全にするために、Multi-Factor Authentication (MFA) をオプションとしてつけることができます。MFA (多要素認証) を利用することで、セキュリティをより強固にできます。MFA は、ハードウェア MFA とか仮想 MFA が利用可能です。

IAM ユーザーの作成時に、ユーザー名を入力しアクセスの種類を選択します。アクセスの種類とは、AWS マネジメントコンソールからアクセスするための ID・パスワードを有効にするのと、プログラムあるいはコマンドラインから利用するアクセスキー ID とシークレットアクセスキーを有効にするの 2 種類があります。次のコンソール画面のキャプチャーの通り、どちらも有効にすることは可能ですが。ただし、**アクセスキーの発行は原則しない**方がよいです。アクセスキーは正しく管理しないと重大な事故の元になります。そのため、必要になった時に最小の権限で発行する方針にします。アクセスキーについては、セキュリティの章で詳しく説明します。

また大切なポイントとして IAM ユーザーは、かならず**利用者ごとに作成**しましょう。共用の IAM ユーザーを作ると、実際に誰が使っていたのか追跡不能になります。またプログラム・ツールから利用するための IAM ユーザーを作る際も、複数のプログラムから共用するのではなく、プログラム・ツールごとに IAM ユーザーを作ることをお勧めします。

ユーザー詳細の設定

同じアクセスの種類とアクセス権限を使用して複数のユーザーを一度に追加できます。 [詳細はこちら](#)

ユーザー名* takuros

別のあるユーザーの追加

AWS アクセスの種類を選択

これらのユーザーから AWS にアクセスする方法を選択します。アクセスキーと自動生成パスワードは前のステップで提供されています。 [詳細はこちら](#)

アクセスの種類* プログラムによるアクセス
AWS API、CLI、SDKなどの開発ツールの アクセスキー ID と シークレットアクセスキーリーを有効にします。

AWS マネジメントコンソールへのアクセス
ユーザーに AWS マネジメントコンソールへのサインインを許可するための パスワードを有効にします。

コンソールのパスワード* 自動生成パスワード カスタムパスワード

* 必須 キャンセル 次のステップ: アクセス権限

図 2.1: IAM ユーザーとアクセスの種類

IAM ユーザーは認証機能だけでなく、AWS のリソースへのアクセス権限を直接付与することも可能です。つまり認可の機能も併せ持つということです。ただし、IAM ユーザーに直接権限を付与すると、管理上の煩雑さが増します。権限の管理については、次で説明する IAM グループで管理し、ユーザーの役割に応じて任意のグループに所属させるというのがお勧めです。



図 2.2: IAM ユーザーへアクセス権限の付与

アクセス権限を記述したものをポリシーと呼びます。ポリシーは、IAM ポリシーと呼ばれる AWS もしくはユーザー定義の権限セット、あるいは IAM ユーザーもしくは IAM グループに紐づくインラインポリシーを使うことができます。ポリシーの使い分けについては、後ほど詳しく説明します。

2.2 IAM グループ

IAM グループは、同一の役割を持つ IAM ユーザーをグループ化する機能です。IAM ユーザー同様にアクセス権限を付与するできます。グループに権限を付与し IAM ユーザーを参加させることにより、役割別グループを作成できます。例えば、全ての権限をもった管理者グループや、インスタンスの操作ができる開発者グループといった具合です。また、IAM ユーザーは複数のグループに所属することもできます

<input type="checkbox"/> グループ名	ユーザー	インラインポリシー	作成時刻
<input type="checkbox"/> admin-group	1		2012-03-01 14:35 UTC+0900
<input type="checkbox"/> developer-group	1		2019-04-20 14:03 UTC+0900
<input type="checkbox"/> handson-group	0		2018-06-27 15:37 UTC+0900
<input type="checkbox"/> viewer-group	0		2019-04-20 14:04 UTC+0900

図 2.3: 役割ごとのグループ

IAM グループを利用することにより、権限を容易に、かつ、正確に管理することができます。IAM ユーザーに直接権限を付与すると、権限の付与漏れや過剰付与など、ミスが発生する確率が高くなります。一般的な IAM の運用として、IAM ユーザーには直接権限を付与せず、IAM グループに権限を付与することを推奨されています。権限の付与方法については、管理ポリシーとインラインポリシーがあります。



図 2.4: グループに権限の付与

ここでオンラインポリシーの説明をしておきます。オンラインポリシーは、管理ポリシーができる前の古い機能です。管理ポリシーは複数のグループやユーザー間で共用できますが、オンラインポリシーは特定のグループ・ユーザーでのみ設定・利用できるポリシーです。オンラインポリシーを使うと、グループごとに同じようなポリシーを何度も記述する必要があり管理上の負荷が高いです。原則使わないようになります。

それでは IAM ポリシーの機能を見てみましょう。

2.3 IAM ポリシー

IAM ポリシーは、AWS リソースへのアクセス権限を制御権限をまとめたものです。ポリシーは JSON 形式で記述しますが、AWS が提供するビジュアルエディタにより選択式で作成することも可能です。「Action (どのサービスの)」「Resource (どういう機能や範囲を)」「Effect (許可 or 拒否)」という 3 つの大きなルールに基づいて、AWS の各サービスを利用する上で様々な権限を設定します。AWS が最初から設定しているポリシーを AWS 管理ポリシー (AWS Managed Policies) といい、各ユーザーが独自に作成したポリシーをカスタマー管理ポリシー (Customer Managed Policies) と呼びます。作成されたポリシーは IAM ユーザー、グループ、ロールに付与

することで、AWS のリソースの利用制御を行います。

次のコードは、IAM ポリシーの記述例です。logs というリソースに、ログの作成権限と S3 へのフルアクセス権限を許可しています。

リスト 2.1: IAM ポリシーの記述例

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "Action": [
7:                 "logs:CreateLogGroup",
8:                 "logs:CreateLogStream",
9:                 "logs:PutLogEvents",
10:                "s3:*"
11:            ],
12:            "Resource": "arn:aws:logs:*:*:*"
13:        }
14:    ]
15: }
```

AWS 管理ポリシーとカスタマー管理ポリシーの使い分け

前述のとおり、ポリシーには AWS 管理ポリシーとカスタマー管理ポリシーがあります。この 2 つの使い分けの戦略はいろいろありますが、お勧めの方法としては足し算と引き算で組み合わせる方法です。足し算部分が AWS 管理ポリシーにあたり、基本的な権限を付与します。引き算としてカスタマー管理ポリシーで、権限を制限します。例えば、管理者権限を付与する AWS 管理ポリシー "AdministratorAccess" と IP 制限をするカスタマー管理ポリシーを組み合わせて IAM グループに付与することにより、全権限を有するものの特定のネットワークからのみ AWS を操作できない権限を持ったグループを作ることができます。

リスト 2.2: IP アドレス制限

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "Action": "*",
7:             "Condition": {
8:                 "NotIpAddress": [
9:                     "aws:SourceIp": [
10:                         "8.8.8.8/32"
11:                     ]
12:                 }
13:             },
14:             "Resource": "*",
15:             "Effect": "Deny"
16:         }
17:     ]
18: }
```

AWS を安全に利用するには、目的に応じた最小権限を付与することが必要です。それを実現するのが IAM ポリシーです。つまり IAM を極めるということは、IAM ポリシーの書き方をマスターするということです。IAM ポリシーは柔軟な記述が可能で、様々な表現方法があります。3 章以降はそのパターンを詳細に解説します。

2.4 IAM ロール

IAM ロールは AWS サービスやアプリケーションに対して AWS の操作権限を与える仕組みです。EC2 に対して IAM ロールを割り与えることにより、EC2 上で実行するアプリはアクセスキー ID・シークレットアクセスキーを設定することなく、その EC2 に割り当てられた AWS の操作権限を利用することができます。また、サーバレスのコード実行基盤である Lambda や、コンテナサービスである ECS など、実行している個々のタスクに対して IAM ユーザーを割り振れないようなサービスに対しても IAM ロールを利用します。

IAM ユーザーは AWS を利用する上で必須といえる機能ですが、IAM ロールは使わないでも何とかなることが多いです。そういう理由で使わずに過ごす人も多いのですが、IAM ロールを正しく使うことにより、AWS の安全性も利便性も格段に高まります。ついつい苦手意識で敬遠がちにされやすいサービスの一つですが、ぜひ積極的に使っていきましょう。

2.5 パーミッション・バウンダリー

最初に IAM の主要機能は4つと説明しましたが、2018年7月に一風変わった機能が登場しました。IAM の移譲権限を制限する Permissions Boundary (パーミッション・バウンダリー) です。バウンダリーは、IAM ユーザーまたは IAM ロールに対するアクセス制限として動作します。付与した権限と、Boundary で許可した権限と重なりあう部分のみ有効な権限として動作します。概念的に解りにくい部分があるので、次の図を参照してください。

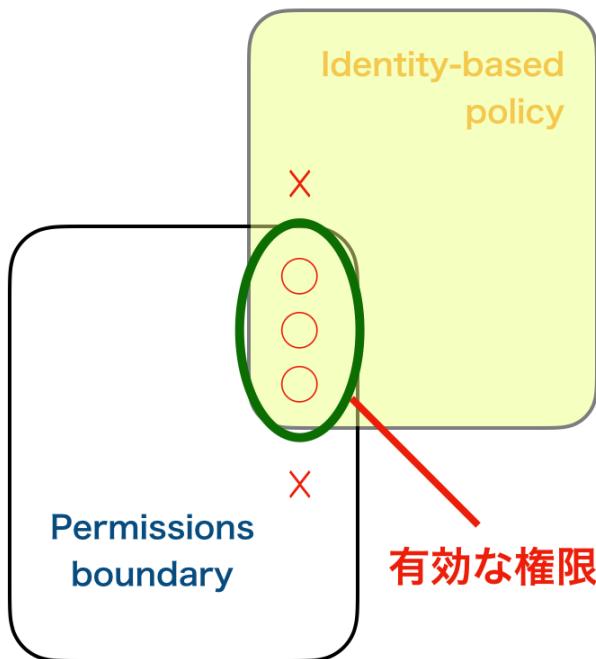


図 2.5: Boundary 利用時の有効権限

バウンダリーで設定していない権限については、IAM ユーザーや IAM ロールでどのように権限を付加しても一切使うことができなくなります。かなり強力な制限のために、使い道についてはよく考える必要があります。一般的な利用例としては、組織外の他者に権限を委任する場合です。この際も、もともと限定した権限の IAM ユーザー等を貸与するのが必須ですが、バウンダリーを利用することにより 2 重の制限となり、意図した以上の権限を渡すことを防ぐことができます。

2.6 IAM の機能のまとめ

この章では、IAM の主要機能である 5 つの機能の解説をしました。

- IAM ユーザー
- IAM グループ
- IAM ポリシー
- IAM ロール
- パーミッション・バウンダリー

IAM ユーザーは、認証を司る機能で必ずユーザーごとに作成しましょう。共有ユーザーの作成は禁止です。IAM グループは、同一の役割を持つ IAM ユーザーをグループ化する機能です。権限の管理は IAM ユーザーではなく、IAM グループにポリシーを付与するとスマートです。IAM ポリシーは、AWS リソースへのアクセス権限を制御権限をまとめたものです。細かい制御をするためには必須となりますので、ぜひマスターしましょう。最後に IAM ロールは、AWS の操作権限を与える仕組みです。ロールを上手く使うと IAM ユーザーのアクセスキーの運用がほぼ不要になります。最初は馴染みにくいですが、大切な機能です。パーミッション・バウンダリーについては、少し使い所を理解するのが難しい機能です。はじめのうちは、意識する必要はないでしょう。

初心者の方には、IAM はとっつきにくい難しいサービスに思えるかもしれません。しかし、IAM の機能はバウンダリーを除いた 4 つを理解すると 9 割方マスターできます。まずその 4 つを理解しましょう。

■コラム: AWS アカウントと IAM の関係

AWS には「AWS アカウント」と「IAM ユーザー」と呼ばれる 2 種類のアカウントがあります。AWS アカウントとは、AWS へサインアップする時に作成されるアカウントです。このアカウントは、ルートユーザーとも呼ばれています。一方、IAM ユーザーは、AWS を利用する各利用者向けに作成されるアカウントです。最初は、IAM ユーザーは存在していないため、AWS アカウントでログインし必要に応じて IAM アカウントを作成します。

AWS の最大のバッドノウハウの一つに、IAM ユーザーを使わずにずっと AWS アカウントで AWS を操作するということがあります。なぜ、これがバットノウハウなのでしょう？ AWS アカウントは、ルートアカウントなので全権を持っていています。そして、IP アドレス制限など利用場所の制限を加えることができません。また 1 つの AWS アカウントに 1 つしか作れないため、複数人で作業する場合は必然的にアカウントを共用する必要があります。つまり通常運用するには向いていないということです。そのため AWS のベストプラクティスとしても、AWS アカウントではなく IAM ユーザーを使って運用しましょうとなっています。

しかし、IAM は AWS の始まりからあった訳ではないのです。IAM の Version を見ても解るように、最初のバージョンは "2008-10-17" です。EC2 や S3 のサービス開始は 2006 年からなので、それまでの 2 年間はどうやっていたのでしょうか？ 実は AWS アカウントを使っていました。そしてサービス開始当初は、画面から操作するマネジメントコンソールは存在しませんでした。つまりルートアカウントのアクセスキーを使って操作していた訳ですね。今だったら絶対ダメーって言われますね。

第3章

IAM チュートリアル

概念的な話が続いたので、ここで IAM に関するチュートリアルをやってみましょう。次の手順の通りに、IAM ポリシー・IAM グループ・IAM ユーザー・IAM ロールと一通りの機能を利用してみます。

1. IP アドレス制限をするカスタマー管理ポリシーを作成する
2. グループを作成する
3. 作成したグループに、全サービスの参照権限と IP アドレス制限をしたカスタマー管理ポリシーを紐付ける
4. IAM ユーザーを作成し、作成したグループをアタッチする
5. クロスアカウントロール（IAM ロール）を作成し、作成した IAM ユーザーからのみスイッチできるようにする

それでは、画面キャプチャーと合わせて確認していきましょう。

3.1 IAM ポリシーの作成

それでは、まず IAM ポリシーの作成をおこないます。IAM のダッシュボードから左メニューのポリシーの作成をクリックします。

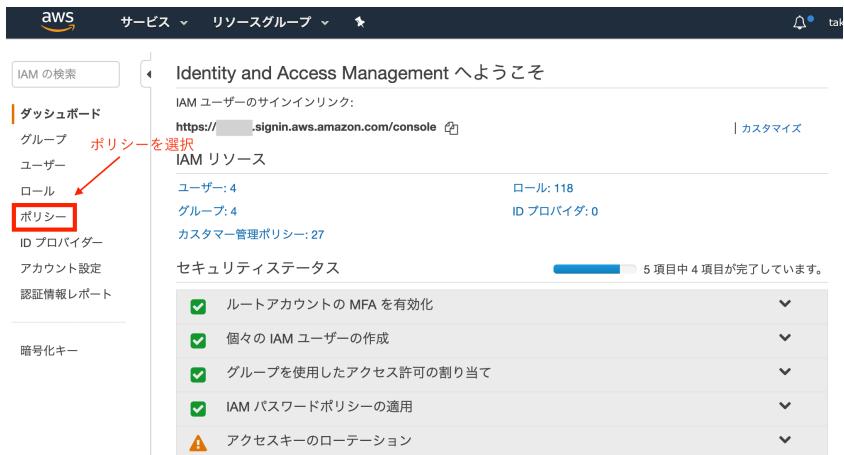


図 3.1: IAM のダッシュボード

既存のポリシー一覧が表示されるので、その左上の「ポリシーの作成」ボタンを押します。ポリシーの作成方法としては、選択式でサービスを追加していくビジュアルエディタと、JSON での記述の 2 パターンがあります。ここでは、JSON で直接記述するので、タブで JSON を選択します。



図 3.2: IAM ポリシーの作成

すべての権限をもった IAM ポリシーの作成

IP アドレスの制限をするポリシーを作成します。その前に、全部の権限をもった IAM ポリシーの作成を見てみましょう。

リスト 3.1: すべての権限を持った IAM ポリシー

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "Action": "*",
7:             "Resource": "*"
8:         }
9:     ]
10: }
```

IAM ポリシーを構成する要素（エレメント）を簡単に説明します。トップレベルの用途として、**Version** と **Statement** の 2 つの要素があります。1 つ目の要素である **Version** は、言語構文ルール（フォーマットのバージョン）の指定をします。IAM ポリシーは 2019 年 5 月現在で 2 つのバージョンがあり、**2008-10-17** と **2012-10-17** があります。記述しない場合はデフォルトの 2008-10-17 が適用されるので、必ず明示的に最新の 2012-10-17 を指定しましょう。2008 年と 2012 年というバージョン番号から解るよう に、IAM ポリシーの記述方法は変更が少ないです。一度覚えると長く使えます。

2 つ目の要素は、**Statement** です。**Statement** はポリシーの主要エレメントで、必須事項です。この中にポリシー式を記載していきます。個々の記載方法については、説明していきます。**Statement** の中には、**Effect**, **Action**, **Resource** の 3 つの要素があります。

Effect は **Statement** の結果を許可する（**Allow**）か、拒否する（**Deny**）のどちらかです。**Action** が利用あるいは否定するサービスを記述します。ワイルドカードが利用できるので、全てのサービスを指定する場合はアスタリスク "*" を使います。**Resource** は利用するリソースの指定をします。例えば利用するリージョンを特定する時や、IAM ユーザー名や特定の S3 バケットに限定する時などに利用します。

IP アドレス制限の追加

次に IP アドレス制限を追加してみましょう。制限を追加する要素は、Condition です。Condition 内には様々な条件式を記述できます。ここでは IpAddress アドレスが一定の範囲内にあるかどうかの判定をしています。

リスト 3.2: IP アドレス制限の追加

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "Action": "*",
7:             "Resource": "*",
8:             "Condition": {
9:                 "IpAddress": {
10:                     "aws:SourceIp": [
11:                         "8.8.8.8/32"
12:                     ]
13:                 }
14:             }
15:         }
16:     ]
17: }
```

IP アドレス部分は、グローバル IP を指定してください。ここでは 1 つの IP アドレスを示す/32 でサブネットを指定しています。サブネットやネットワーク全体を指定することも可能です。なお、SourceIP の部分でプライベート IP の範囲を指定しても意味がないのでご注意ください。VPC 内のリソースを制限するには、VPC ID もしくは VPC Endpoint ID を指定する必要があります。ここについては、6 章の IAM とセキュリティで説明します。

反転させて特定 IP アドレス以外のアクションを拒否

最後にこれまでの記述を反転させて、特定 IP アドレス以外のアクションを拒否するポリシーを書いてみましょう。

リスト 3.3: 特定 IP アドレス以外のアクションを拒否

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Deny",
6:             "Action": "*",
7:             "Condition": {
8:                 "NotIpAddress": [
9:                     "aws:SourceIp": [
10:                         "8.8.8.8/32"
11:                     ]
12:                 }
13:             },
14:             "Resource": "*"
15:         }
16:     ]
17: }
```

ここでのポイントが、Condition の指定が NotIpAddress で反転させていることです。そうすることで、今まで 8.8.8.8/32 という IP からのアクセスという条件だったのが、それ以外という条件に変わっています。併せて、Effect を Allow (許可) から Deny (拒否) に変更しています。これにより、8.8.8.8/32 以外の IP アドレス以外でアクセスした場合は一切拒否するというポリシーに変わっています。少し解りにくいところがあるので、図で確認してみましょう。

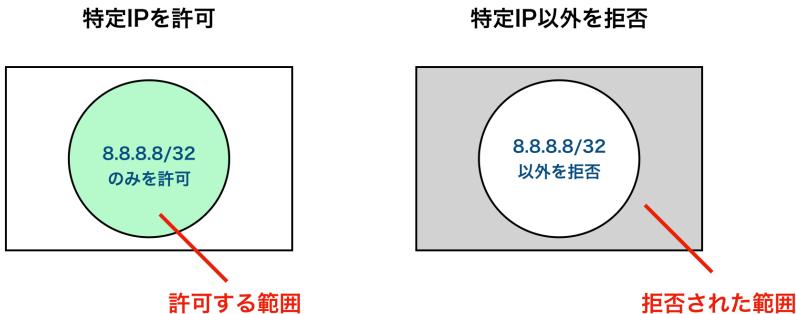


図 3.3: 条件指定の反転

ステートメント一つ一つに IP アドレス制限を追加するという方法もありますが、記述量が多くミスもしやすくなり、また範囲の変更時にも修正箇所も多くなります。その際に有効になるのが、反転させた条件指定を持つ IAM ポリシーです。この反転させた IAM ポリシーの活用は、次の IAM グループの作成の時に詳しく説明します。

最後に、この"特定 IP アドレス以外のアクションを拒否"を IPAddressRestriction として保存しておきましょう。

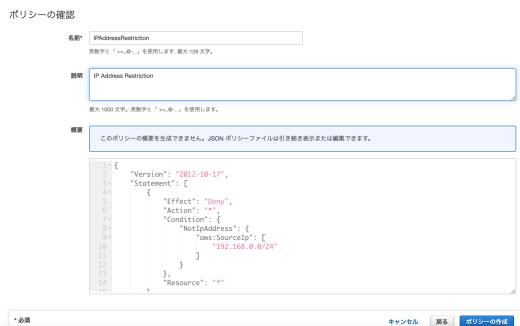


図 3.4: IP アドレスの制限

3.2 IAM グループの作成

IAM ポリシーが作成できたので、次に IAM グループを作成します。前章の IAM の機能で説明しましたが、権限は IAM ユーザーではなく IAM グループに付与して管理すべきです。ここでは、AWS 管理ポリシーである `ReadOnlyAccess` を付与して、更に先ほど作成した IP アドレス制限ポリシーを付与してみましょう。

グループの作成は、グループ名を指定の上でポリシーをアタッチして作成します。ここでは、`ViewOnlyUsers` というグループ名にします。



図 3.5: IAM グループの作成

グループ名の命名規則については、特に標準的なものはないので独自で作れば大丈夫です。一般的に多いケースとしては、`Admins` や `Operators` といった、最後に `s` をつけて複数形にする命名規則です。不規則な変化をするものについて綴を間違えると恥ずかしいので、入力前に英単語として正しいか確認しておくことをお勧めします。それ以外には、`admin-group` のように最後に `group` をつけるケースも多いです。どちらでも良いので、一貫性のある名前をつけましょう。一方で AWS 管理ポリシーの名前をみていると、一貫性がある名前を付けるのは難しいのかなという気持ちになってきます。

IAM グループにポリシーのアタッチ

次にポリシーをアタッチします。AWS 管理ポリシーは非常に数が多いので、条件句で絞り込んで指定しましょう。ここでは、AWS 管理ポリシーである ReadOnlyAccess と、先程作成したカスタマー管理ポリシーの IPAddressRestriction を選択しましょう。

The screenshot shows the 'Policy Attachments' section of the AWS IAM Groups page. On the left, there's a sidebar with navigation links: '新しいグループの作成ウィザード', '手順 1: グループ名', '手順 2: ポリシーのアタッチ', and '手順 3: 対象'. The main area has a title 'ポリシーのアタッチ' and a note: 'アタッチするポリシーを 1 個以上選択してください。グループは、それぞれ 10 個までのポリシーをアタッチできます。' Below this is a table titled '結果件数: 106' with a filter dropdown 'フィルター: ポリシータイプ' set to 'ReadOnly'. The table lists six policies:

ポリシー名	アタッチされたエンティティ	作成時間	最終更新時間
AmazonS3ReadOnlyAccess	2	2015-02-07 03:40 UTC+0900	2015-02-07 03:40 UTC+0900
AmazonEC2ReadOnlyAccess	1	2015-02-07 03:40 UTC+0900	2015-02-07 03:40 UTC+0900
<input checked="" type="checkbox"/> ReadOnlyAccess	1	2015-02-07 03:39 UTC+0900	2019-04-03 01:07 UTC+0900
AlexaForBusinessReadOnlyAccess	0	2017-12-01 01:47 UTC+0900	2018-06-26 08:52 UTC+0900
AmazonAppStreamReadOnlyAccess	0	2015-02-07 03:40 UTC+0900	2016-12-08 08:00 UTC+0900

At the bottom right are buttons: 'キャンセル', '戻る', and '次のステップ'.

図 3.6: IAM ポリシーの付与

これで 2 つのポリシーを付与したグループの作成が完了です。効率的な権限管理には、グループにポリシーを組み合わせることが効率的になります。では、相矛盾する権限が付与した場合、どのような動作になるのでしょうか？ここで権限を重ねがけした場合の動作について説明しましょう。

IAM の許可・拒否ポリシーの挙動

IAM ポリシーの基本的な動作は、次のようになっています。

明示的な Deny > 明示的な Allow > 暗黙的な Deny (デフォルト)

何も権限を与えていない状態が暗黙的な Deny となります。そこに権限を付与 (明示的な Allow) していきます。では、それに対して拒否 (明示的な Deny) をした場合、どうなるのでしょうか？ 実は、この明示的な Deny が一番強くなります。いくら重ねて許可していても、一つでも拒否されるとその権限は拒否されます。

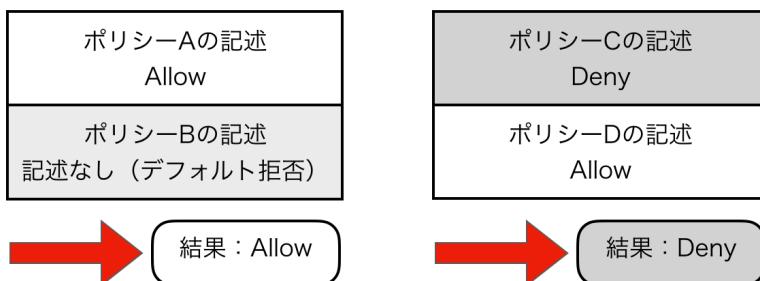


図 3.7: 許可・拒否ポリシーの判断

上記の考え方を踏まえると、ポリシーを組み合わせの有効性が解ります。基本的な権限を AWS 管理ポリシーで付与して、除きたい権限をカスタマー管理ポリシーで否定すると、最小限の設定で望みの権限を作ることができます。よくある利用例としては、先程のような IP アドレス制限であったり、全権限を持った Administrator ポリシーから IAM に関する権限のみを除くといったものがあります。これらのパターンについては、IAM のデザインパターンで紹介します。

3.3 IAM ユーザーの作成

それでは、IAM ユーザーを作成し、先程作成した IAM グループに属させてみましょう。IAM ダッシュボードの左メニューのユーザーを選択し、IAM ユーザー画面から **ユーザーを追加** ボタンから作成ウィザードを開きます。

ここでは、test-user01 という名前でユーザーを作成します。IAM ユーザーは必ず利用者ごとに作る必要があり、アカウントの共用はしてはいけません。アクセスの種類としてプログラムによるアクセス、AWS マネジメントコンソールへのアクセスの 2 つが選べます。両方とも選択することもできますが、原則的にはプログラム用のアカウントと人間が使うアカウントで分けた方が良いでしょう。人間が使うアカウントは、大きめの権限を付与することが多いので、プログラム・CLI との共用は避けましょう。プログラム用アカウントはプログラムごとに最小権限を付与しましょう。



図 3.8: IAM ユーザーの作成

IAM ユーザーに権限の付与

次に権限の付与です。幾つかの付与方法がありますが、原則はグループに追加することにより権限を付与しましょう。ここでは、先程作成した viewer-group に参加させます。なお、1 ユーザーが複数のグループに参加することも可能です。



図 3.9: IAM グループのアタッチ

タグの設定や確認の後で、IAM ユーザーの作成が行われます。パスワードの自動生成した場合に、そのパスワードを取得するのは作成後の確認画面のみなのでご注意ください。

ユーザーを追加

1 2 3 4 5

✓ 成功
以下に示すユーザーを正常に作成しました。ユーザーのセキュリティ認証情報を確認してダウンロードできます。AWS マネジメントコンソールへのサインイン手順を E メールでユーザーに送信することもできます。今回が、これらの認証情報をダウンロードできる最後の機会です。ただし、新しい認証情報はいつでも作成できます。

AWS マネジメントコンソールへのアクセス権を持つユーザーは「」でサインインできます

[.csv のダウンロード](#)

	ユーザー	パスワード	ログイン手順を E メールで送信
▶	test-user01	***** 表示	E メールの送信

図 3.10: パスワードの取得

ここでダウンロード可能な CSV にはアクセスするための URL と ID・パスワードが記載されているので、取扱には注意しましょう。

3.4 クロスアカウントロールの作成

それでは、最後にこのユーザーからのみスイッチできるクロスアカウントロール (IAM ロール) を作成しましょう。クロスアカウントロールの作成は、IAM ダッシュボードの左メニューのロールから、ロールの作成ボタンから行います。ロールの作成画面では、最初に信頼されたエンティティの種類を選びます。ここでは、別の AWS アカウントを選びます。

別の AWS アカウントという所で違和感も持つかもしれませんが、自分の AWS アカウントからスイッチすることも可能です。アカウント ID の部分に 12 桁のアカウント ID を入力しましょう。アカウント ID は、IAM ユーザーでログインしている場合は右上にユーザー名の横に表示されます。エイリアスログイン URL でログインしている場合は、グローバルメニューからマイアカウントに遷移して確認しましょう。

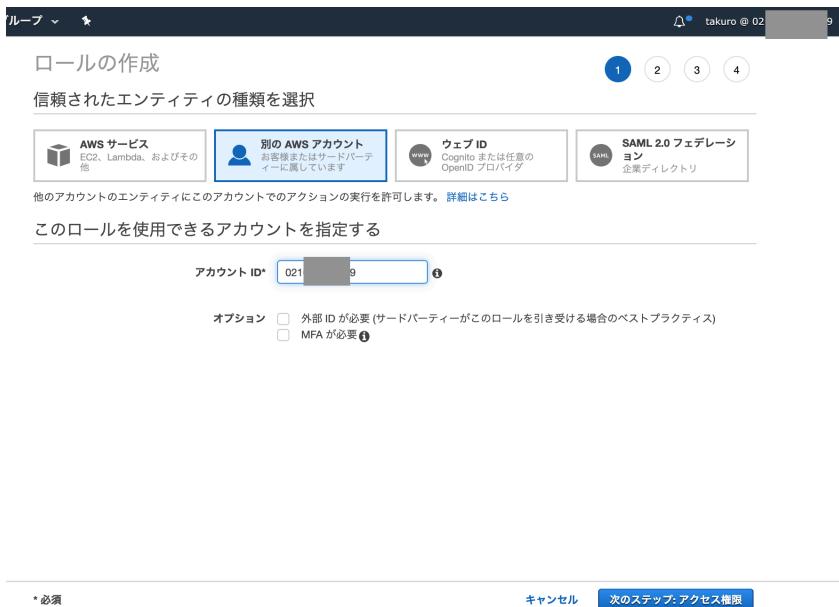


図 3.11: クロスアカウントロールの作成

オプションとして、スイッチ元のユーザーの MFA の必須化が可能です。セキュリティの観点から、IAM ユーザーには MFA の設定を推奨します。その際には、ロールスイッチの際に MFA を必須化するのが良いでしょう。

IAM ロールに権限の付与

次にアクセス権限の設定を行います。IAM ロールへの権限付与は、IAM グループ同様に IAM ポリシーをアタッチすることで権限を付与します。ここでは、全件をもった AdministratorAccess 権限を付与しましょう。

ロールの作成

1 2 3 4

▼ Attach アクセス権限ポリシー

新しいロールにアタッチするポリシーを 1つ以上選択します。

ポリシーの作成

検索

538 件の結果を表示中

	ポリシー名	次として使用	説明
<input checked="" type="checkbox"/>	AdministratorAccess	Permissions policy (8)	Provides full access to AWS services an...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	なし	Provide device setup access to AlexaFor...
<input type="checkbox"/>	AlexaForBusinessFullAccess	なし	Grants full access to AlexaForBusiness r...
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	なし	Provide gateway execution access to AI...
<input type="checkbox"/>	AlexaForBusinessNetworkProfileServicePolicy	なし	This policy enables Alexa for Business to...
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	なし	Provide read only access to AlexaForBu...
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	なし	Provides full access to create/edit/delete...
<input type="checkbox"/>	AmazonAPIGatewayInvokeFullAccess	なし	Provides full access to invoke APIs in A...

▶ アクセス権限の境界の設定

* 必須

キャンセル

戻る

次のステップ: タグ

図 3.12: アクセス権限の設定

タグを設定して、ロール名・ロールの説明を設定してクロスアカウントロールの作成は完了です。ここでは、ロール名に SwitchRoleForTest01 と設定しています。

ロールの作成

確認

以下に必要な情報を指定してこのロールを見直してから、作成してください。

ロール名* 英数字と「+=, @_」を使用します。最大 64 文字。

ロールの説明 最大 1000 文字。英数字と「+=, @_」を使用します。

信頼されたエンティティ アカウント ()

ポリシー  AdministratorAccess 

アクセス権限の境界 アクセス権限の境界が設定されていません

追加されたタグはありません。

* 必須

キャンセル 戻る ロールの作成



図 3.13: アクセス権限の設定

スイッチ用 URL の取得とロールに切り替え

それではいよいよ、作成したロールに切り替えてみましょう。初回のロールの切り替えは、URL を利用します。切替用の URL は、ロール詳細画面の"コンソールでロールを切り替えることができるユーザーに、このリンクを知らせます。"から取得します。

第3章 IAM チュートリアル

3.4 クロスアカウントロールの作成

The screenshot shows the AWS IAM 'Roles' page with the 'SwitchRoleForTest01' role selected. The role ARN is listed as `arn:aws:iam::02...:role/SwitchRoleForTest01`. The role's name is `Switch role for test01`. It has no inline policies. A policy named `AdministratorAccess` is attached. The role was created on 2019-05-01 11:40 UTC+0900. A note at the bottom says: '最大 CLI/API セッション期間 1時間' and provides a URL: <https://signin.aws.amazon.com/switchrole?roleName=SwitchRoleForTest01&account=02...>. The 'Permissions policies' tab is selected, showing one policy named 'AdministratorAccess'. The 'Permissions boundary' section is noted as '(not set)'.

図 3.14: 切替用 URL の取得

この URL を IAM ユーザーでログイン済みのブラウザで開くと、切替名の設定画面が表示されます。任意の表示名を設定ください。また、表示名と同時にメニューの色の設定もできます。本番系であれば黄色、開発系であれば青色等のルールを作れば、切替間違えを防ぐ効果も期待できます。ぜひ、自身の運用実態に沿った使い方を考案してください。

ロールの切り替え

同一ユーザー ID とパスワードを使用している AWS アカウント全体にわたって、リソースの管理を許可します。AWS 管理者がロールを設定してアカウントとロールの詳細が提供されると、ロールを切り替えることができるようになります。[詳細はこちら](#)。

The dialog box for switching roles. It contains fields for 'アカウント*' (Account) with a dropdown menu, 'ロール*' (Role) with a dropdown menu set to 'SwitchRoleForTest01', '表示名' (Display Name) with the value 'SwitchRoleForTest01' in a blue input field, and a '色' (Color) section with a color palette showing a gradient from red to black. At the bottom are buttons for '*必須' (Required), 'キャンセル' (Cancel), and a large blue button labeled 'ロールの切り替え' (Switch Role).

図 3.15: スイッチロールの表示名の設定

なお、未ログインの場合は、切替元である IAM ユーザーへのログイン画

面が表示されます。

切替後の画面は、メニュー右上のユーザー表示が背景色付きになっている以外の違いはありません。ユーザー名の表示は、切替前に設定した文字列が表示されます。なお、このロールの切替設定は、IAM ユーザーが保持している訳ではなく、ブラウザに保存されています。そのため、違う PC あるいはブラウザで利用する場合は、再度設定する必要があります。

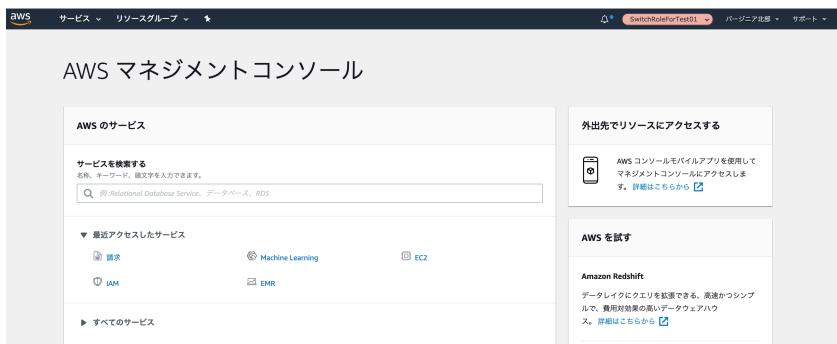


図 3.16: 切替後の画面

元のユーザーに戻るには、メニュー右上のユーザー情報をクリックして、○○に戻るをクリックしてください。



図 3.17: 元の IAM ユーザーに戻る

これで IAM ユーザーから IAM ロールにスイッチすることができました。ただし、このままでは問題があります。どの IAM ユーザーからスイッチできるかという設定をしていないため、同一アカウントの誰からもスイッチできる状態になっています。次は、スイッチ元の IAM ユーザーの設定をしてみましょう。

IAM ロールにスイッチ元の IAM ユーザー制限の追加

IAM ロールのスイッチ元の制限は、Condition (条件) ではなく Principal (信頼関係) を利用します。該当のロールの詳細画面から、信頼関係のタブを見てみましょう。

リスト 3.4: 変更前の Principal

```
1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Principal": {
7:         "AWS": "arn:aws:iam::999999999999:root"
8:       },
9:       "Action": "sts:AssumeRole",
10:      "Condition": {}
11:    }
12:  ]
13: }
```

99999999999 の部分は AWS アカウント ID です。Principal というのは信頼する AWS のサービスで、arn というリソース ID で指定します。変更前のデフォルトの状態だと、AWS アカウント全体に対して信頼関係が設定されています。そのため、この AWS アカウント内のどの IAM ユーザーでも切り替える事ができます。それでは、これに制限を加えてみましょう。

リスト 3.5: 変更後の Principal

```
1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Principal": {
7:         "AWS": "arn:aws:iam::999999999999:user/test-user01"
8:       },
9:       "Action": "sts:AssumeRole",
10:      "Condition": {}
11:    }
12:  ]
13: }
```

ユーザーの指定には、user/ユーザー名という形で指定します。複数のユーザーを指定する場合は、次のように角括弧 [] 内に列挙します。残念ながら IAM グループでの指定はできません。

リスト 3.6: 複数ユーザーの指定

```
1: "Principal": {
2:   "AWS": [
3:     "arn:aws:iam::999999999999:user/user-name-1",
4:     "arn:aws:iam::999999999999:user/user-name-2"
5:   ]
6: }
```

変更後に保存して、制限したユーザーでスイッチできる事、それ以外のユーザーでスイッチできない事を確認してください。なお、今回はクロスアカウントロールとして IAM ロールを使いましたが、実際の使い方としては EC2 や Lambda など AWS のサービスに付与することの方が一般的です。

3.5 チュートリアルのまとめ

この章の IAM チュートリアルでは、IAM ポリシーの作成から、IAM グループ・IAM ユーザー・IAM ロールと一通りの機能の作成・利用をしました。IAM ポリシー・IAM グループ・IAM ユーザーを利用して、IP アドレス制限付きの参照権限を持ったユーザーを作成しました。そして、IP 制限なしの全権限を持った IAM ロールを作成し、スイッチ元を作成したユーザーのみに限定しました。誤操作防止の観点からも、普段使いするユーザーには参照権限のみというのは良い設計です。そして、設定変更時のみスイッチするというオペレーションが簡単にできるようになりました。

この章以降も IAM の使い方・テクニックを紹介しますが、IAM ポリシー・IAM グループ・IAM ユーザー・IAM ロールの 4 つの機能が使えれば、あとは応用するだけです。それでは、IAM のディープな世界に飛び込んでいきましょう。

第4章

IAM ポリシーのデザインパターン

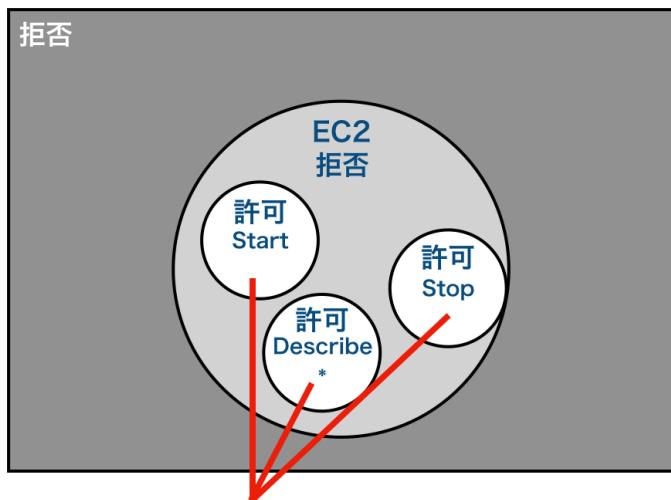
チュートリアルにて IAM の基本的な機能を理解したでしょうか？ 機能の理解の次は、IAM の権限設定の基本的な IAM ポリシーのデザインパターンを理解しましょう。デザインパターンには幾つかあり、代表的なものは次の3つのパターンに分類できます。

- ホワイトリスト・パターン
- ブラックリスト・パターン
- ハイブリット・パターン

ホワイトリスト・パターンは、許可する権限のみ付与していくパターンです。ブラックリスト・パターンは、逆に許可してはいけない権限のみを剥奪するパターンです。ハイブリット・パターンは、両者の組み合わせです。それぞれ見ていきましょう。

4.1 ホワイトリスト・パターン

ホワイトリスト・パターンは、許可する権限のみ付与していくパターンです。権限の付与方法として、EC2 や S3 といったサービス単位に付与するパターンや、更に細かくアクション単位まで指定するパターンがあります。



特定のサービス・アクションのみ許可

図 4.1: ホワイトリスト・パターン

次の例は、EC2 に関しては参照権限すべてとインスタンスのストップ・スタートの権限を付与し、S3 に関しての全権限を付与しています。

リスト 4.1: ホワイトリストパターン

```
1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Action": [
6:         "ec2:Describe*",
7:         "ec2:StartInstances",
8:         "ec2:StopInstances",
9:         "s3:*"
10:      ],
11:      "Effect": "Allow",
12:      "Resource": "*"
13:    }
14:  ]
15: }
```

サービス単位で指定するのであれば、サービスごとに FullAccess, Read-OnlyAccess といった予め用意されたポリシーが AWS 管理ポリシーとして用意されています。そのため、カスタマー管理ポリシーとして単体で作成する必要は殆どありません。ホワイトリスト・パターンとしての一般的な使い方は、運用者の用途に応じてアクション単位まで細かく指定するケースが多いです。また、ホワイトリスト・パターンは、必然的に最小権限を付与することになります。

ホワイトリスト・パターンのメリット

ホワイトリスト・パターンのメリットは、必要最小限の権限のみ付与するのでセキュリティ面での信頼性が高い点です。EC2 のインスタンスの操作する担当の例に挙げると、EC2 の参照権限と起動・停止の権限付与するので、誤って、或いは悪意を持って、ネットワークの設定を変えて本来許されない通信を許可してしまうということが起こり得なくなります。

また、ホワイトリスト・パターンで設計・運用するには、事前に充分な設計が必要になります。その事が逆説的に、構築しようとしているシステムが充分に考慮されたシステムにつながるでしょう。

ホワイトリスト・パターンのデメリット

ホワイトリスト・パターンのデメリットは、事前に役割が決まっていないと権限が付与できることになります。本番運用等で定型のオペレーションが決まっている場合は問題がないのですが、探索的な開発の場合、事前に必要な権限は決まっていません。その際に、最小権限を前提とするホワイトリスト・パターンは効率が悪くなります。また、アクション単位で指定する場合、AWS のサービスが拡張して新たなアクションが増えた場合、都度それに対応して増やしていく必要があります。このように、ホワイトリスト・パターンは厳密に管理されているが故に、IAM ポリシー管理の負荷が高いです。

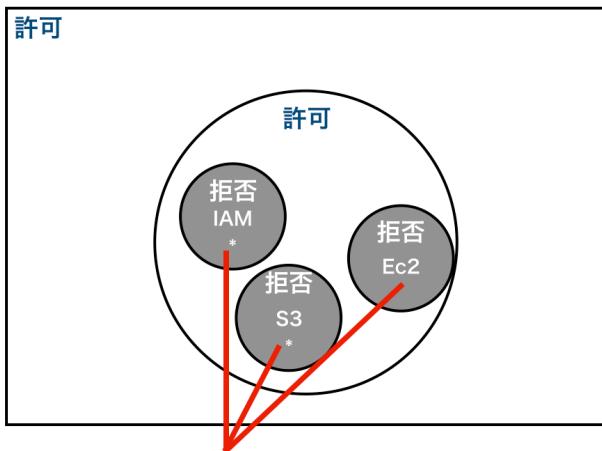
ホワイトリスト・パターンのユースケース

ホワイトリスト・パターンの適用は、業務・運用設計が確立している本番環境に適用すべきです。また本番環境の中でも、役割分担がはっきりしていて、かつ高いセキュリティーを求められるエンタープライズ向けです。柔軟な対応を求められる環境でホワイトリスト・パターンを適用しても、運用の実態にそぐわなく破綻するでしょう。一方で、監査証跡など求められるようなシステムであれば、必須とも言えます。

なお AWS 管理ポリシーも、ほぼホワイトリスト・パターンの実装になっています。

4.2 ブラックリスト・パターン

ブラックリスト・パターンは、許可してはいけない権限のみを剥奪するパターンです。IAM のユーザー作成やポリシー付与を与える事は、実質的にその AWS アカウントを自由に操作する権限をに等しいです。またネットワークに関する権限を付与すると、悪意があればネットワークに穴を開け情報漏えいをするといったことも可能となります。ブラックリスト・パターンは、防がないといけない事項のみ禁止し、あとは比較的自由に権限を付与します。



特定のサービス・アクションのみ拒否

図 4.2: ブラックリスト・パターン

次のポリシーでは、IAM に関する権限のみ拒否しています。付与する範囲 (Resource) を全部 (*) ではなく、サービス単位で指定するといった方法も有効です。この例だと、IAM に関する全権限を否定しているため、自分の IAM ユーザーのパスワード変更すらできなります。IAM ロールに付与する分には問題ないですが、IAM ユーザーに付与すると問題がでてきます。

リスト 4.2: ブラックリスト・パターン

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Action": "iam:*",
6:             "Effect": "Deny",
7:             "Resource": "*"
8:         }
9:     ]
10: }
```

先程の否定の仕方だと、自分自身のパスワードの変更すらできなくなります。運用上、IAM ユーザー自身に自分のパスワードが変更できないと困ります。パスワード変更を除いてそれ以外の IAM をブラックリストに入れる記述も可能ですが、記述方法が難しくなります。そういう場合、IAM のデフォルト拒否を使う方法も選択肢としていれておきましょう。

次のポリシーの記述例は、NotAction を使い IAM 以外の権限を許可し、IAM に関しては自身のパスワード変更と一覧取得の権限を明示的に付与しています。

リスト 4.3: NotAction を利用した記述例

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "NotAction": [
7:                 "iam:*"
8:             ],
9:             "Resource": "*"
10:            },
11:            {
12:                "Resource": [
13:                    "arn:aws:iam::999999999999:user/${aws:username}"
14:                ],
15:                "Action": [
16:                    "iam:ChangePassword",
17:                    "iam:GetAccountPasswordPolicy"
18:                ],
19:                "Effect": "Allow"
20:            },
21:            {
22:                "Resource": "*",
23:                "Action": [
24:                    "iam:Get*",
25:                    "iam>List*"
26:                ],
27:                "Effect": "Allow"
28:            }
29:        ]
30: }
```

なお自分自身のパスワード変更には、上記で示した以外の権限も必要になってきます。第6章のIAMとセキュリティにて、詳細の説明をします。

ブラックリスト・パターンのメリット

ブラックリスト・パターンのメリットは、禁止事項のみを定義すればよいので、IAM ポリシーの設計・設定が最小で済むことです。また、事前に必要な権限が確定していない場合でも、禁止事項のみ定義すればよいので、その段階でも始めるすることができます。つまりブラックリスト・パターンは AWS の利用者に高い自由度を与えるながら、組織として許容するセキュリティレベルを保つポリシー戦略になります。

ブラックリスト・パターンのデメリット

ブラックリスト・パターンのデメリットは、予期せぬ機能が突然使えるようになるという点です。Admin 権限にブラックリストを加えて運用している場合、AWS に新しいサービスが始まると自動的に新しい機能も使えるようになります。つまり想定していないサービスが突然使えるようになるということです。その前提を受け入れる必要があります。

ブラックリスト・パターンのユースケース

AWS のサービスの進化の速度は速く、新しいサービスがどんどん追加されます。現実問題、AWS のスピードに合わせてホワイトリストを定義していくのは難しいです。そのため、一般的な IAM の運用としては、ブラックリスト・パターンの利用が多くなります。ブラックリスト・パターンの適用環境としては、前述のような探索的な開発環境もしくは、比較的大きな権限が必要な AWS アカウントの管理者のように向いています。

また AWS の権限のデフォルトは暗黙的拒否です。そのため、どこかで許可のステートメントを作った上でブラックリストで拒否するという形になります。厳密にはブラックリストのみで運用することは不可能です。実際の運用は次で紹介するハイブリット・パターンになります。

4.3 ハイブリット・パターン

ハイブリット・パターンは、ホワイトリスト・パターンとブラックリスト・パターンの組み合わせです。ここでは、IAM グループを利用した AWS 管理ポリシーによる許可ポリシーとカスタマー管理ポリシーで作成したブラックリスト・ポリシーの組み合わせを紹介します。

リスト 4.4: 管理者権限と IP アドレス制限の組み合わせ

```
1: 管理者権限をあらわす AWS 管理ポリシー AdministratorAccess
2: {
3:     "Version": "2012-10-17",
4:     "Statement": [
5:         {
6:             "Effect": "Allow",
7:             "Action": "*",
8:             "Resource": "*"
9:         }
10:    ]
11: }
12:
13: 特定 IP アドレス以外の否定
14: {
15:     "Version": "2012-10-17",
16:     "Statement": [
17:         {
18:             "Effect": "Allow",
19:             "Action": "*",
20:             "Condition": {
21:                 "NotIpAddress": [
22:                     "aws:SourceIp": [
23:                         "8.8.8.8/32"
24:                     ]
25:                 ]
26:             },
27:             "Resource": "*",
28:             "Effect": "Deny"
```

```
29:      }
30:    ]
31: }
```



図 4.3: IAM Group で 2 つのポリシーを組み合わせる

ハイブリット・パターンのメリット

ハイブリット・パターンは、AWS の定義済みのポリシーと自分で作ったブラックリストを組み合わせて利用することにより、最小の労力で実用的なポリシーを作ることができます。また、ポリシーを組み合わせて権限を実現するため、個々のポリシーはシンプルに作ることができます。運用要件等で詳細の権限までホワイトリストとして定義する必要がある場合を除いて、ほとんどのケースがハイブリット・パターンを適用することができます。

ハイブリット・パターンのデメリット

ハイブリッド・パターンのデメリットは特にありません。強いていふと許可の IAM Policy と拒否の IAM Policy をグループで組み合わせる前提で作っておいて、拒否をつけ忘れる可能性があるというくらいですが、それは

作業面でのミスなので設計上の問題ではありません。

ハイブリッド・パターンのユースケース

ハイブリッド・パターンは、あらゆる環境で適用可能です。ホワイトリスト方式は AWS に習熟した上で業務要件が確定している必要があります。そのため現実的に多用するのが、AWS 管理ポリシーです。AWS 管理ポリシーは機能ごとに全ての権限、参照専用といったパターンで提供されています。最初に利用する際は FullAccess が多いのですが、権限が大きくなりがちという問題があります。その対策に、ハイブリッド・パターンで一定の制限をつけることをお勧めします。

1 ポリシーで管理すべきか、グループで組み合わせるか

一口にハイブリッド・パターンと定義しましたが、権限の付与の仕方としてポリシー内に許可の権限と拒否の権限を両方記述する方法と、それぞれ別のポリシーとして作成した上で IAM グループに付与することで組み合わせる方法があります。どちらの方法がよいのでしょうか？

1 つのポリシー内にすべて書くことのメリットは、視認性が高まりそれを付与すれば完結するという簡潔さです。一方で再利用性は低いです。複数のポリシーをグループ等で組み合わせる方法は、再利用性が高いので結果として管理するポリシーは少なくてすみます。

どちらが良いかは状況次第ですが、一般的な傾向としては少人数・小規模の利用であれば 1 つのポリシーで完結させる方法でも充分です。しかし大規模利用であれば、ポリシーを組み合わせて使う方法をお勧めします。

4.4 IAM ポリシーのまとめ

この章では、IAM ポリシーのデザインパターンを学びました。ポリシーの付与パターンとして、許可（ホワイトリスト）か拒否（ブラックリスト）のどちらかです。複雑にみえるポリシーも、この2つの組み合わせになります。その上でどういった方針で記述していくかが IAM のデザインポリシーになります。

付与のパターンは、許可と拒否のどちらかです。しかし、条件文を駆使することにより表現のパターンは多様化します。特に、○○以外といった反転のパターンを使うことにより、ほぼあらゆる表現をすることができるようになります。柔軟な頭で考えてみましょう。

IAM ポリシーは権限設定の最小単位であり、AWS を使いこなす上では必須の機能の一つといえます。長い付き合いになるので、ぜひマスターしていきましょう。要件ごとの具体的な記述方法については、後の章でも順次説明していきます。

■コラム：最小権限の探求

ホワイトリスト方式の基本は、必要最小限の権限を付与することにあります。AWS が提供する IAM のベストプラクティスの一つにも、"最小権限を付与する"と明記されています。しかし、これを実践しようとするとなかなかに難しい事が解ります。権限はアクセスレベルとして、「リスト」、「読み込み」、「タグ付け」、「書き込み」の 4 つに大別されています。多くのサービスについては、実際のリソースを操作する「書き込み」の権限付与を注意していると、大半のリスクは制御できます。

一方で、最小権限を追求するには、それだけでは済みません。例えば EC2 の権限の場合、書き込みに関するアクションだけで、140 以上あります。この中には、ネットワークを操作といった影響の大きいアクションも含まれています。そういったアクションを回避しつつ、運用に必要な権限を追加していくという作業が必要になります。これを完璧に追求するのは、かなりの知識を持った上で試行錯誤する必要があります。本来実施したかった構築作業より IAM の設計の方が時間が掛かるといったことも、往々にしてよく発生します。

つまり最小権限を完全に追求するのは難しいということです。ある程度は見切りをつけて、どこかで折り合いをつける必要があるということを認識してください。また、こういった点を考えると、NG の項目のみ洗い出すブラックリスト方式の優位性が出てきます。

人が作業する場合は、完全に定型の作業に終始し続けることはないです。そのため、必ず防ぎたいことをブラックリスト方式で防ぎ、ある程度自由度を残すのが良いでしょう。そして、プログラムが使う権限については動作のゆらぎがないので、必要な権限が限られています。こちらをホワイトリスト方式で管理するのが良いでしょう。

第5章

IAM グループのデザインパターン

前章では、IAM ポリシーの設計の考え方であるデザインポリシーを紹介しました。この章では、IAM グループについて考えてみましょう。筆者は、複数のポリシーを束ねて使う方法を好みます。また、束ねる先としては、IAM ユーザーではなく IAM グループが適切と考えています。そういう意味で、IAM ポリシーと IAM グループを併用する設計を多用しています。

IAM グループ自体の機能はシンプルなので、デザインパターンもシンプルになります。そのパターンとは、グループにユーザーがどのように所属するかです。IAM ユーザーは複数のユーザーに所属できます。そのため、デザインパターンとしては、IAM ユーザーが権限の異なる複数のグループに所属する方法と、1 つのグループに必要なポリシーを全て集めて、ユーザーはそのグループに所属するのか、どちらかの方法が考えられます。それでは、それぞれ見ていきましょう。

5.1 複数グループに所属

複数グループに所属するパターンは、ユーザーが複数のグループに属することを前提に権限設定します。次の図の例は、全ユーザーが所属するグループに IP アドレス制限やパスワード変更権限を付与し、それとは別に開発者向けやネットワーク担当者向けの権限を付与したグループを作成し所属させます。

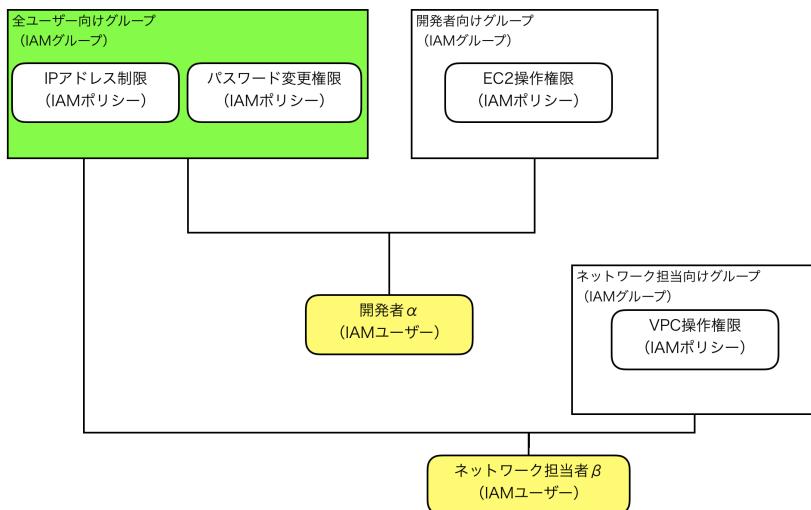


図 5.1: 複数グループに所属

この方法のメリットは、ポリシーが複数のグループから利用されることが殆どないので、変更の際の影響範囲が解りやすいということがあります。また必須の権限・制約を全ユーザー向けのグループに付与して、抜け漏れを防ぎやすくできるです。

5.2 グループ内に複数ポリシー

グループ内に複数ポリシーは、ユーザーが1つのグループに属することを前提に権限設定します。次の図の例は、IP アドレス制限やパスワード変更権限など全ユーザーに必要なポリシーを、それぞれのグループに付与しています。

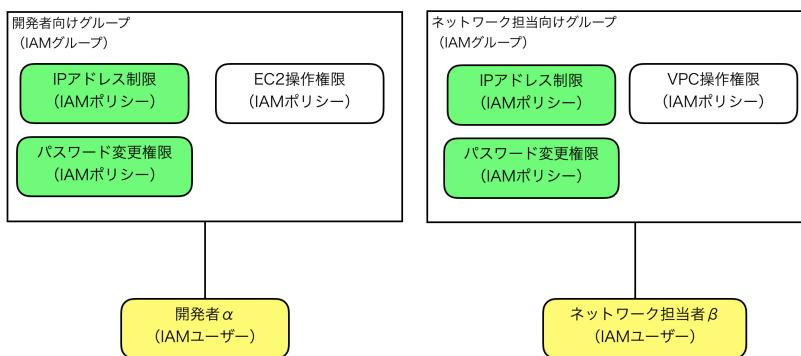


図 5.2: グループ内に複数ポリシー

この方法のメリットは、ユーザーは基本的に1つのグループにしか所属しないため、ユーザーからみるとシンプルな構造になることです。また、ポリシーも多数のグループから利用される事が前提になるため、自然とシンプルで使いやすいポリシー設計につながります。

5.3 IAM グループのまとめ

この章では、IAM グループのデザインパターンとして、複数グループに所属するパターンと、グループ内に複数ポリシーを配置するパターンを紹介しました。この2つのパターンについては、機能的な優劣がないので、どちらを選択するかは完全に好みの問題となります。あるいは両者を組み合わせたハイブリッドパターンも考えられます。

著者としては、グループ内に複数のポリシーを配置するデザインパターンを利用することが多いですが、対象となる組織の規模や利用形態によっては複数グループに所属というパターンも充分有効です。それぞれの特徴を掴んだ上で、使いやすい方を選んでください。

■コラム: IAM グループの階層構造について

IAM グループの設計をしていると、グループを階層構造にして管理したくなります。ここでいう階層構造とは上位の権限を引き継ぎ、下の階層の場合に差分のみ記述するといった塩梅です。AWS の IAM グループで、階層構造を表現することは可能なのでしょうか？

AWS の機能としては、Yes です。IAM グループにはグループ名の他に、path というオプション設定が可能です。これを使うと、/my-company/division/のような階層表現が可能です。FAQ にも次のように記載されています。

LDAP のように、ユーザーの集合を階層的に構成できますか？

はい。ユーザーおよびグループは、Amazon S3 のオブジェクトパスと同様に、パスを使って編成できます。例えば、/my-company/division/project/joe などです。

一方で、この path による階層は、どのような機能があるのでしょうか？ 実は、グループを検索する時に利用できるだけです。権限の継承のような機能はありません。ということで、IAM グループを階層構造にできないのと聞かれると、筆者は No と答えています。

なお、この path オプションは、GUI の AWS マネジメントコンソールでは設定できないので、CLI 等を使って指定する必要があります。また、AWS マネジメントコンソールでの IAM グループの検索は、グループ名のみ対象となっているので、path の階層で検索しようとしても効果ありません。

第 6 章

IAM とセキュリティ

4、5 章で IAM のデザインパターンを見てきました。以降は、具体的な設計・運用について踏み込んでいきます。まず 6 章で扱うのはセキュリティです。AWS を扱う上でセキュリティは最重要項目の一つであり、AWS のセキュリティの半分は IAM の適切な設計と運用に掛かっていると言っても過言ではありません。権限を付与する際に考慮すべき点をまずおさえた上で、実際に付与する際に特に注意すべき点を実例とともに見ていきます。

6.1 IAM ベストプラクティスの遵守

AWS では、IAM のベストプラクティスという形で、守るべきポイントを公開しています。https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/best-practices.html)



図 6.1: IAM のベストプラクティス

IAM ベストプラクティスのリスト

- AWS アカウントのルートユーザーのアクセスキーをロックする
- 個々の IAM ユーザーの作成
- IAM ユーザーへのアクセス許可を割り当てるためにグループを使用する
- 最小権限を付与する
- AWS 管理ポリシーを使用したアクセス許可の使用開始
- インラインポリシーではなくカスタマー管理ポリシーを使用する
- アクセスレベルを使用して、IAM 権限を確認する
- ユーザーの強力なパスワードポリシーを設定
- 特権ユーザーに対して MFA を有効化する
- Amazon EC2 インスタンスで実行するアプリケーションに対し、ロールを使用する
- ロールを使用したアクセス許可の委任
- アクセスキーを共有しない
- 認証情報を定期的にローテーションする
- 不要な認証情報を削除する
- 追加セキュリティに対するポリシー条件を使用する
- AWS アカウントのアクティビティの監視
- IAM ベストプラクティスについてビデオで説明する

ベストプラクティスは、IAM のみならず AWS アカウントの管理を含めて記載されています。内容については既に説明したものも多く、また残りの部分は後ほど説明するので割愛します。一点だけ補足すると、特権ユーザーに対して MFA を有効化するについては、これはより厳しくした方がよいと考えています。人間が使うことを想定している、全ての IAM ユーザーに対して MFA を有効化を原則にするのがよいでしょう。なぜならば現時点では特

権ユーザーでなくても、将来的に特権ユーザーになる場合もあるからです。その際に MFA を設定し忘れるという事もあります。IAM の権限管理は IAM グループで行うのが原則で、IAM ユーザーは認証のみにしておくべきです。であれば、認証は原則 MFA 付きというルールにしておいた方がシンプルで解りやすいです。一方で、プログラムから利用するための IAM ユーザーというのもあります。これについては、次の運用の章で説明します。

次に IAM のベストプラクティスで公開されている以外の部分で、権限設計時に考えるべきポイントを紹介します。

6.2 ルートユーザーを使わない

AWS アカウントを作ったら最初にすることは、Administrator 権限を付与した IAM ユーザーを作成し AWS アカウントを使わずに済むようにすることです。原則、AWS アカウントを使いません。また、今では AWS アカウントに対してアクセスキーを発行するのはご法度です。AWS アカウント開設時にすべきことは、巻末付録の AWS アカウント開設時のチュートリアルを参照してください。

6.3 IAMに関する権限付与

IAM の権限設計で、考えるべきは IAM に関する権限です。洒落のような文言ですが、IAM 権限は自他に AWS の権限を付与できる権限なので、実質 Administrator 権限と同義になります。事実、多くの AWS 不正利用のツール・ボットも、最初に操作用の IAM ユーザーを作り出すことから始めるようです。そのため、まず一番最初の原則として、AWS のアカウント管理者以外には IAM 権限を与えないということを覚えてください。

ユーザー自身のパスワードと MFA の設定を許可する

IAM 権限の剥奪がセオリーとしても、単純に全ての IAM 権限を拒否すると困ったことになります。ユーザー自身でパスワード変更や MFA の設定が自分でできなくなるからです。それでは、まず自身のパスワード変更・MFA 設定・アクセスキーの設定のポリシーを作ってみましょう。考え方としては、必要な権限を付与するだけでよいのですが、重要なポイントとしてリソースで自分自身のみという制限を追加することです。そうしないと、人のパスワードを自由に変更し自分でログインすることができるという大きなセキュリティの穴を作ることになるからです。

リソース制限の 999999999999 の部分は、AWS アカウント ID です。自身のものに変えましょう。AWS アカウント ID に限らず汎用的に使う場合は、ワイルドカード "*" での代用も可能です。また \${aws:username} の部分は、IAM ポリシーエレメントと呼ばれる変数で、ログインしている IAM ユーザー名、つまり自分自身に置き換えられます。残念ながら、AWS アカウント ID の変数は、IAM ポリシーエレメントにはありません。代わりに CloudFormation の擬似パラメーター参照に AWS::AccountId として存在します。

リスト 6.1: 自身のパスワードと MFA の設定を許可

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Resource": [
6:                 "arn:aws:iam::999999999999:user/${aws:username}",
7:                 "arn:aws:iam::999999999999:mfa/${aws:username}"
8:             ],
9:             "Action": [
10:                 "iam:ChangePassword",
11:                 "iam>CreateAccessKey",
12:                 "iam>CreateVirtualMFADevice",
13:                 "iam:DeactivateMFADevice",
14:                 "iam>DeleteAccessKey",
15:                 "iam>DeleteVirtualMFADevice",
16:                 "iam:EnableMFADevice",
17:                 "iam:GetAccountPasswordPolicy",
18:                 "iam:UpdateAccessKey",
19:                 "iam:UpdateSigningCertificate",
20:                 "iam:UploadSigningCertificate",
21:                 "iam:UpdateLoginProfile",
22:                 "iam:ResyncMFADevice"
23:             ],
24:             "Effect": "Allow"
25:         },
26:         {
27:             "Resource": "*",
28:             "Action": [
29:                 "iam:Get*",
30:                 "iam>List*"
31:             ],
32:             "Effect": "Allow"
33:         }
34:     ]
35: }
```

6.4 Lambdaのリソースベースの権限

Lambdaのアクションの中には、アクセス権限の管理に関するものがあります。通常、AWSのリソースのアクセスについては、IAMポリシーで管理されています。Lambdaはこれを、Lambda関数の実行ロールとして紐付けることで利用できます。実はLambdaはこれ以外にもリソースのアクセス制御をすることができます。AWS Lambdaのリソースベースのポリシーというものです。リソースベースのポリシーを利用すると、リソースごとに他のアカウントに使用許可を付与することができます。[\(https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/access-control-resource-based.html\)](https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/access-control-resource-based.html)



図 6.2: lambda のリソースベースポリシー

ここで問題になるのが、この権限を利用すると IAM 権限を利用せずとも AWS のリソースのアクセス権を自由にコントロールすることができる点です。Lambda のアクセス権限の付与は、AddPermission や AddLayerVersionPermission で行えます。Lambda に権限を付与する際は、FullAccess ではなく必要な権限を選択して付与しましょう。

6.5 インターネット公開系の権限

IAM 権限の次に気をつける必要があるのが、AWS のリソースをインターネットに公開することができる権限です。ユーザー情報やクレジットカード番号のような機密情報を、設定の間違いや悪意を持ってインターネットから直接アクセスできるような状態にしてしまうと、重大なインシデントになります。このため、リソースのインターネット公開につながる権限は特に注意して制御する必要があります。

AWS の難しいところは、このインターネット公開系の権限が多岐に渡ることです。インターネット公開系の権限としてパッと思いつくのが、EC2 のインスタンスへの通信許可だと思います。また、S3 のファイルをパブリック・アクセスにするといったことも考えられます。実は、それ以外にも沢山あります。例えば、EC2 のルートデバイスピリュームに機密情報を入れたままマシンイメージ（AMI）を作成し、それを一般公開化（パブリック・イメージ）にしてしまうと、他のアカウントから簡単にその情報を抜き出せます。EBS のスナップショットについても同様です。必要がない限りそういった権限は、拒否しましょう。

リスト 6.2: ネットワーク系の操作の禁止

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Deny",
6:             "Action": [
7:                 "ec2:*Vpc*",
8:                 "ec2:AttachVpnGateway",
9:                 "ec2:DetachVpnGateway",
10:                "ec2>CreateInternetGateway",
11:                "ec2:ModifySnapshotAttribute",
12:                "ec2:ModifyImageAttribute"
13:            ],
14:            "Resource": "*"
15:        }
16:    ]
17: }
```

EC2 のように権限が多岐に渡るものについては、ホワイトリストで管理するのはなかなか大変です。そういう際はネットワーク系の操作を明示的拒否する方法と併用しましょう。しかし、ここでは EC2 の代表的な権限を拒否しているだけで、同様の危険性のあるものは AWS のサービスの中で多数あり全てをカバーしている訳ではありません。原則として、最小権限を付与するということは忘れないでおきましょう。

■コラム: EC2の権限範囲の問題

EC2のactionのリストをみると、思いのほか権限の範囲が広い事に気がつくでしょう。インスタンス操作の他に、VPCやセキュリティグループの操作といったネットワーク系の操作の権限も含まれています。この事がEC2に関する権限設定を難しくします。単純にEC2:"*"といった権限を付与すると、ネットワークの操作までされてしまうからです。EC2権限にはネットワーク操作系の権限が含まれているということを認識しておいてください。

そもそも何故EC2権限の中にネットワーク系の権限が含まれているのでしょうか？歴史的な経緯として、初期のAWSにはVPCの機能はありませんでした。インスタンスとそれに対するセキュリティグループしかなかったのです。そのためにEC2のactionとしてネットワーク操作が含まれることとなったのでしょう。この際にnetwork:といったアクショングループを作っていてれば、もう少しスッキリしたでしょうね。なお、ネットワーク系のポリシーは、AWS管理ポリシーであるNetworkAdministratorとしてまとめられています。これを使えば楽に設定できるので、是非活用してください。

のことからも、先を見通して設計するのはAWSですら難しいということが解ります。そして、例え設計に失敗したとしても何とかなるものです。だから、我々も設計に悩みすぎずポジティブに捉えていきましょう。

6.6 VPC 内からのアクセス

IAMの権限設定にある程度慣れてくると、より細かく利用元のリソースを制限するようになります。その時、一度はハマるであろうがVPC内からAWSリソースへのアクセスです。ここではEC2のCLIからS3バケットへのアクセスを、特定のIPで制限する方法とVPCによって制限する方法の2つを見てみましょう。

次の図のようにPublic Subnet内にEC2インスタンスが配置されたVPCのネットワークがあります。EC2インスタンスには、Global IP(パブリックIP)が付与され、VPCにはInternet GatewayとS3用のVPC Endpointがあります。VPC Endpointについて少しだけ補足すると、VPC内からInternet Gatewayを介さずに直接S3のようなAWSリソースにアクセスするための経路です。

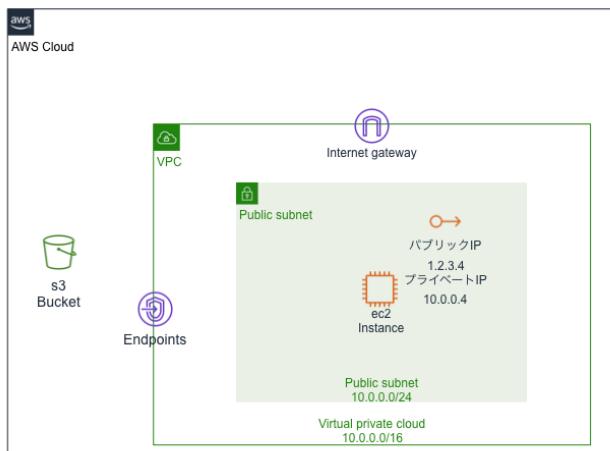


図 6.3: VPC からの AWS リソースへのアクセス

まずルーティングの設定で、VPC Endpoint を通らずに Internet Gateway を経由して S3 にアクセスする場合を考えてみましょう。この場合、S3 に接続するのを VPC 内の EC2 に限定したい場合は、パブリック IP を指定すれば良いです。

次にルーティングを切り替えて、Internet Gateway を通らずに S3 にアクセスする場合を考えてみましょう。VPC Endpoint を利用の場合は、プライベート IP のまま S3 にアクセスすることになります。となると、先程の制限をプライベート IP に変えるだけでいいのでしょうか？ ここがハマリポイントなのですが、IAM でリソース制限する際にプライベート IP は使えません。VPC の ID を使います。

なお、S3 へのアクセス元制限は、IAM ポリシーを使うより S3 側のバケットポリシーを使うのが一般的です。IAM のポリシーによる制限で複数のバケットに対して複雑な制御をするのは困難ですし、そもそも全てのユーザーに対して設定する必要があります。それであれば、S3 のバケットポリシーで制限する方が合理的です。
(https://docs.aws.amazon.com/ja_jp/AmazonS3/latest/dev/example-bucket-policies-vpc-endpoint.html)



図 6.4: S3 へのアクセスを VPC からのみに制限

6.7 アクセスキーの原則禁止

IAM ユーザーを作成する際の原則は、アクセスキー・シークレットアクセスキーを作らないとしましょう。昨今の AWS のアカウント乗っ取りの原因を整理すると、アクセスキー・シークレットアクセスキーが起因するのが大半です。アクセスキーの流出防止策や流出時の被害を最小限にする方法は沢山ありますが、そもそもアクセスキーを作らないに越したことがないです。今では殆どのケースで、アクセスキー無しで代替できるようになっています。

IAM ロールを利用する

アクセスキーを利用しようとする場合の代表的なケースとしては、プログラムからの利用になります。そのプログラムの実行場所が EC2 インスタンスの場合、そもそもアクセスキーを利用する必要はありません。EC2 インスタンスに付与された IAM ロールはインスタンスプロファイルと呼ばれ、プログラムはそのインスタンスに付与された権限を利用できるようになっています。プログラムからは SDK に定められた認証情報の検索順序に従い、自動的に取得して利用するようになっています。EC2 インスタンス内へのアクセスキーの配置は、原則禁止と覚えておいてください。

CLI についても、自分のローカル PC でアクセスキーを使って実施するより、作業用の EC2 を用意するのがより安全で良いでしょう。

6.8 Capital Oneの情報流出事件に思うこと

2019年7月末に、米大手金融機関であるCapital Oneが不正アクセスによる1億人を超える個人情報の流出を発表しました。規模もさることながら、Capital OneはAWSから何度も事例に取り上げられるような先進的な企業だったので、衝撃な事件でした。また、攻撃手法についても、従来の単純な設定ミスを狙ったものではなく、より一步踏み込んだ攻撃でした。事件の概要と攻撃手法を振り返ることで、IAMをどのように設計すべきだったか確認してみましょう。

事件の概要と攻撃手法

攻撃手法を簡単にまとめると、WAFの脆弱性を利用してIAM Roleのクレデンシャル情報を取得し、それを手元から利用して情報を取得したという形です。piyologさんの図を、再描画させて貰いました。

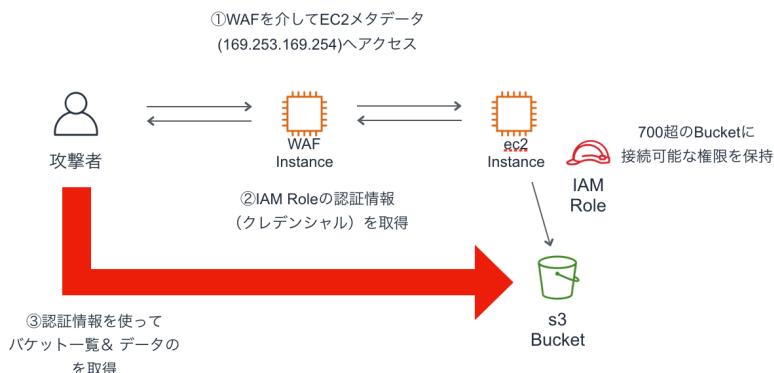


図 6.5: 攻撃手法

第6章 IAMとセキュリティ 6.8 Capital Oneの情報流出事件に思うこと

<https://piyolog.hatenadiary.jp/entry/2019/08/06/062154>



図 6.6: piyolog SSRF 攻撃による Capital One の個人情報流出についてまとめてみた

流出の原因

情報流出の一次的な原因としては、WAFです。セキュリティ製品については、もう独自の運用でカバーできる範囲ではないので、AWS WAF等のマネージド・サービスもしくは、専門の会社が提供する製品を利用するがあ良いでしょう。ただ、あくまで WAFについては一次的な原因で真の問題については、別のところにあります。それは、不用意に大きな権限を持った IAM Role を利用した点です。

対策としては、2点考えられます。付与する IAM ロールの権限を最小にすることと、S3 のバケットポリシーです。

IAM ロール側の対策

IAM ロール側の対応としては、権限の絞り込みです。今回の例ですと、バケットの一覧取得権限を付与しないで、特定の S3 Bucket のみ接続できるような権限のみ付与します。また、アクションも絞り込みます。ログやファイルを置くだけであれば、Put のアクションのみ権限を付与するようにします。

S3 Bucket 側の対策

機密性が高い情報を格納する S3 バケットについては、IAM 側の制限のみではなくバケット側にも制限を加える事が必須です。制限の例としては、バケットポリシーで EC2 インスタンスが属する VPC のみ接続できるように制限するなどが有効です。

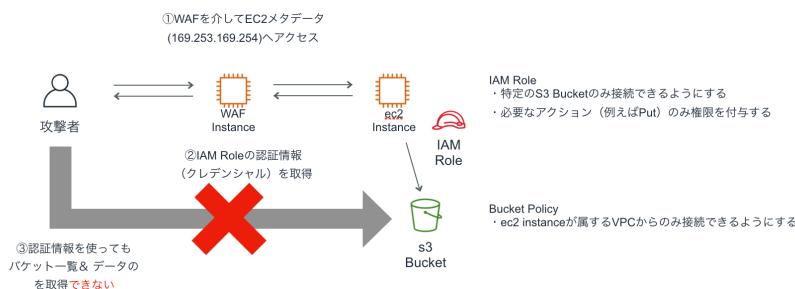


図 6.7: IAM の制限と S3 の BucketPolicy による防御

具体的な設定手順

それでは、IAM ロールとバケットポリシーでの制限の具体的な手順をみてみましょう。

IAM ロールの制限

IAM ロールの制限は、特定のバケットのみ接続できる IAM ポリシーを作成し、アタッチしましょう。次の例は、バケットの操作が全てできるようにアクションをワイルドカードで指定しています。例えば、ログを置くだけであれば s3:PutObject のみに絞る方がより良いでしょう。

第6章 IAMとセキュリティ 6.8 Capital Oneの情報流出事件に思うこと

リスト 6.3: 特定バケットのみ接続するポリシー

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Action": "s3:*",
6:             "Effect": "Allow",
7:             "Resource": [
8:                 "arn:aws:s3:::<bucketname>",
9:                 "arn:aws:s3:::<bucketname>/*"
10:            ]
11:        }
12:    ]
13: }
```

S3にバケットポリシーの追加

S3のバケットの方にも、バケットポリシーを追加することで接続元を制限しましょう。Condition句で、特定のVPCエンドポイント以外からの接続を拒否するようなバケットポリシーを作ります。

リスト 6.4: 特定のVPCからのみ接続できるバケットポリシー

```
1: {
2:     "Version": "2012-10-17",
3:     "Id": "ListrictAccess",
4:     "Statement": [
5:         {
6:             "Sid": "Allow-from-specific-VPC-only",
7:             "Effect": "Deny",
8:             "Principal": "*",
9:             "Action": "s3:*",
10:            "Resource": [
11:                "arn:aws:s3:::kindle-iam-test",
12:                "arn:aws:s3:::kindle-iam-test/*"
13:            ]
14:        }
15:    ]
16: }
```

```
13:           ],
14:           "Condition": {
15:             "StringNotEquals": [
16:               "aws:sourceVpce": "vpce-id1234"
17:             ]
18:           }
19:         ]
20:       ]
21:     }
```

接続条件として、VPC エンドポイント以外に VPC 全体を示す VPC ID なども使えます。しかし、VPC エンドポイントの方が、VPC エンドポイントのポリシーと併用することでより細かい制御ができます。そのため、VPC の制限の場合は、VPC エンドポイントの ID で制限することをお勧めします。

バケットポリシーの制御としては、他にもグローバル IP なども利用できます。AWS 以外の特定の拠点からのみアクセスする場合などに使うと良いでしょう。重要なデータを格納するバケットの場合、バケットポリシーの制限も必ず使うようにしましょう。

6.9 IAM とセキュリティのまとめ

この章では、IAM のセキュリティの考え方と具体的な設定について踏み込みました。まず一番最初に押さえておくべきは、AWS が公式に出している IAM ベストプラクティスの遵守についてしっかりと理解した上で実践することです。その上で、AWS 内のリソースをインターネットに公開が可能にする権限であったり、VPC からの AWS リソースへのアクセスについて学びました。また Lambda の FullAccess の意外な危険なについても認識できたと思います。そして、Capital One の情報流出事件を元に、IAM ロールの最小権限付与の必要性と、S3 バケット側の制限による二層防御の必要性が

解ったのではないかと思います。AWSを利用する上で、セキュリティは最重要事項の一つです。しっかりと理解して使っていきましょう。それでは、次はセキュリティが保たれて安全な状態を継続するために、IAMの運用について考えてみましょう。

■コラム: IPアドレス制限の是非とゼロトラストセキュリティ

IAMの不正利用防止の対策としてよく挙げられる手法の一つがIPアドレスによる制限です。防御のカテゴリーとしては、ネットワークによる防御にあたります。一方で2010年代以降、ゼロトラストセキュリティ（ネットワーク）という概念が提唱されています。これは、社内（ネットワーク内）は安全であるという前提を捨て、多層でセキュリティ施策を検討しましょうというアプローチになります。

これまでセキュリティ施策の一貫として、IP制限の方法について説明をしてきました。ゼロトラストセキュリティの流れの中で、この手法を使うのは問題ないのでしょうか？結論としては、問題ありません。ゼロトラストセキュリティは、ネットワークによる防御のみを信じないと言っているだけあって、それぞれの層で防御することは否定していません。むしろ、それぞれの層で可能な限りの施策をうつように推奨しています。その観点から考えると、IPアドレスの制限は有効な手段の一つです。なぜならば、IAMに関わる事故の原因としては、アクセスキーなどの認証情報の漏洩から不正利用されている事が大半だからです。それであれば、例え流出しても実害を防ぐIPアドレス制限は有効な手段です。また、それ以外にも必ず二要素認証を行うようにしておきましょう。運用形態によっては、二要素認証のみで充分な場合もありますが、2つの防御方法を講じておくことでより安全になります。

第 7 章

IAM の運用

この章では、IAM の運用について考えてみましょう。運用といっても、人によって頭に思い浮かべることが違う可能性があります。そこで、まずは IAM の運用の定義と目的を整理した上で、何が必要かというのを考えていきましょう。またその次に、AWS を日常的に利用する上で直面する課題についても考えてみます。

7.1 IAM の運用の目的

セキュリティを考慮して正しく設計したとしても、それだけでは充分ではありません。安全な状態を保ち続けることが必要です。それが IAM の運用です。ただし、セキュリティのために手間が 100 倍掛かるようになるのであれば、その手順を守る人は誰もいなくなります。机上の空論ではなく、現実に即した運用を考える必要があります。また、AWS を組織で利用する場合は、当然複数の人が関係することになります。そうなると利用者によってセキュリティレベルの差異がないようにする必要があります。技術レベルが高い人だけ実現できる手法ではなく、誰でも達成できるにはどうすれば良いかの観点が必要です。

IAM 運用の目的

- AWS を安全な状態に保ち続ける

目的の達成のために

- 維持するために手間が掛からない（現実に許容できるコスト）
- 利用者によってセキュリティレベルの差異がでないようにする（標準化）

7.2 役割と責任範囲の明確化

それでは IAM の運用を設計する上で、最初に何をすれば良いのでしょうか？ それは、AWS アカウントを利用する登場人物の洗い出しだけではありません。個人で利用している場合は、関係者は 1 人です。しかし、実はそれだけではないのです。例えば、サードパーティのツールから利用する IAM ユーザーやサーバー付与する IAM ロールなども出てきます。自分一人で使うから、管理者権限を付与した IAM ユーザーを一つ作ればよいと考えるのではなく、ツール・プログラムを含めて AWS の利用者を洗い出す必要があります。

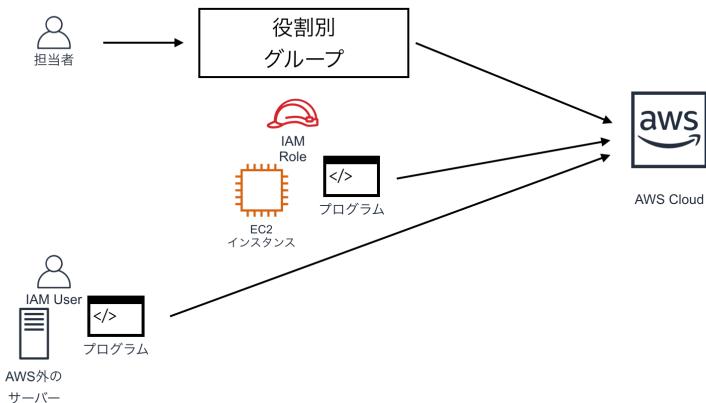


図 7.1: AWS 利用者の洗い出し

次に複数人で AWS を利用する場合です。会社など組織で利用するケースの場合は、ほとんどこの形態になることでしょう。複数人で利用する場合では、特に役割と責任範囲の明確化が必要です。一般的な役割としては、AWS アカウント管理者、開発者、運用担当者などがあります。大きな組織の場合、ネットワーク担当者や、経理担当者が請求金額などを確認する事や、AWS が正しく利用されているか監査担当がいるかもしれません。つまり組織によっては、役割分担の仕方が変わることです。最低限必要なのは AWS アカウント管理者で、後は組織の形態に応じて分けていきましょう。注意点としては、無理に役割を細分化する必要はありません。担当範囲から役割を決めて、それぞれが責任を持つ範囲を決めます。その領域を超える場合や不明瞭な場合に、役割を分けていけばよいでしょう。

7.3 AWS アカウントの管理

IAM の範疇から少し離れますが、まず AWS アカウント（ルートユーザー）の管理と運用から考えましょう。ルートユーザーは、アカウント作成後に原則使ってはいけません。AWS アカウントを作った後は、まずルートユーザーを使わずに済むための設定を肃々とすべきです。具体的な手順としては、チュートリアルとして巻末に載せているので是非実施してください。

7.4 IAM ユーザーの管理

IAM ユーザーの管理としては、AWS を利用する人ごとに適切な最小権限を付与し、必要でなくなったら権限をなくすということに尽きます。ただ、それを手作業で継続的にし続けるというのも大変です。そのため、運用ルールを設定の上で、AWS のサービスを併用しつつ運用手順に落とし込みましょう。

- CloudTrail で AWS コンソール、API 利用履歴のログを取得する
- Config を利用し、IAM 関係の設定の変更を監視する
- Config Rules で設定値を監視
- (月次等の) 定期的な IAM ユーザーの棚卸し

IAM ユーザーの運用をする上では、人が利用する IAM ユーザーとプログラムが利用する IAM ユーザーと分けて考えることが必要です。

AWS マネジメントコンソールから利用する IAM ユーザー（人間用）

人間用の IAM ユーザーは、ユーザーごとに作成し共用アカウントの作成は禁止です。異動等で人の入れ替わり時を考慮して、棚卸しルールを作りま

しょう。不要になった時に即時に削除、月次で棚卸し確認が理想的です。また、人間は必ずミスをします。そのため、設計とセキュリティサービスでカバーすることが大事です。

設計でのカバーとしては、最小権限の付与を前提としてミスをしても影響を最小限にできるようにしましょう。次にサービスでのカバーです。Config や CloudTrail を使うことにより、IAM の設定に関することも監視できます。MFA の設定の有無や Admin 権限を付与しているかの確認が自動的に行なえます。サービスを上手く活用することで、設定ミスをしたとしても気付ける・防止することができます。個人の注意に依存するのではなく、仕組みとしてカバーできるように目指しましょう。

サービスによるカバーの詳細については、AWS Config マネージドルールのリストを確認してください。ここにあるリスト以外にも、カスタムで自作のチェックツールを作ることもできます。

https://docs.aws.amazon.com/ja_jp/config/latest/developerguide/managed-rules-by-aws-config.html



図 7.2: AWS Config マネージドルール リスト

CLI やプログラムから利用する IAM ユーザー（プログラム用）

盲点になりがちのが、CLI やプログラムから利用する IAM ユーザーです。人の出入りについては解りやすいので、プログラムから利用されなくなった IAM ユーザーというのは外から観測しにくいです。その結果、ゾンビのような野良 IAM ユーザーが残り続ける事になります。これを防ぐため

には、次の2点を実施しましょう。

- IAM ユーザーごとの担当者を明確化する
- 最終利用日から一定時間が過ぎた IAM ユーザーは継続の確認をする

1つ目の担当者の明確化です。野良 IAM ユーザー化を防ぐために、プログラムから利用する IAM ユーザーに関しては用途を明確にした上で担当者を決めましょう。IAM ユーザーにも任意のタグをつけることができるので、タグに担当者を書くというルールにしておけば解りやすいでしょう。

次に最終利用日から一定時間が過ぎた IAM ユーザーについてです。一定期間使われていない IAM ユーザーについては、先程決めた担当者に存続の可否を確認しましょう。使われていないようであれば、まずはキーの無効化をしましょう。無効化して暫く問題がなければ、そのキーは使われていないということです。消しましょう。問題があれば、再度有効化することで復活させることができます。

ユーザー名	グループ	アクセスキーの古さ	パスワードの古さ	最後のアクティビティ	MFA
<input type="checkbox"/> Amazon-Product-Ad...	なし	1 767 日間	なし	767 日間	有効でない

図 7.3: 最終利用日（アクティビティ）の確認

7.5 アクセスキーの管理と CLI

AWS の不正利用の多くは、IAM が起因です。そしてその中でも、IAM ユーザーのアクセスキーの流出によるものが特に多いです。そのため、アクセスキーとシークレットアクセスキーについては、特に注意をする必要があります。

まず大前提として、アクセスキーの発行は必要な IAM ユーザーのみとします。使うかどうか決まっていないのに、IAM ユーザー発行時に必ずアクセスキーを作るということは止めましょう。次に、アクセスキーを発行する

IAM ユーザーの権限についてです。付与する権限を最小にする必要があります。特に注意が必要なのが、管理者権限を持っているユーザーのアクセスキーです。管理者権限付きのアクセスキーが流出すると一大事です。そこでお勧めなのが、CLI でのスイッチロールです。

あまり知られていませんが、CLI を利用時にも IAM ロールへの切り替えが可能です。IAM ユーザー側に IP アドレスによる利用制限をした上で、必要な場合にスイッチロールで権限を切り替えるという運用をしておけば、万が一アクセスキーを流出した場合にも直接の被害を防ぐことが可能となります。

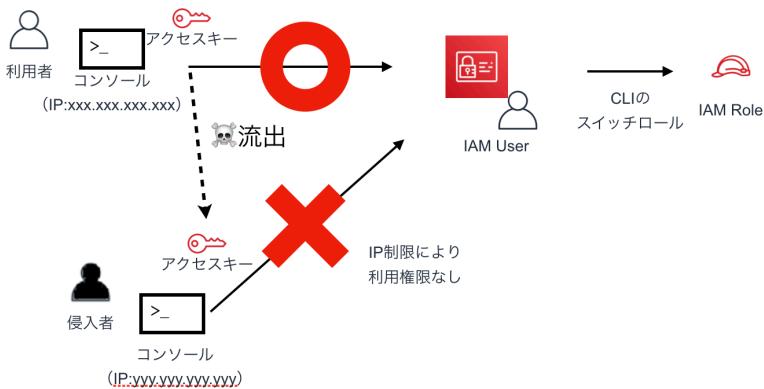


図 7.4: CLI でのスイッチロールの活用

CLI での IAM ロールの切替方法は、次の MFA 利用を促す部分で一緒に説明します。

7.6 MFA 未利用時に権限を制限し、MFA 利用を促す

お勧めなのが MFA を利用していない時は殆どの権限を剥奪し、CLI でも MFA を促すという方法も有効です。まずユーザー側で MFA 認証をしていない場合は、IAM 以外の権限を拒否する IAM ポリシーをアタッチします。これにより、CLI から MFA 無しでコマンドを発行した場合に殆どの権限がなくなります。次に、`~/.aws/config` に切替元のユーザー名 (`mfa_test_user`)、切替元のユーザー NOMEFA の ARN、切替先のロール名を記載したプロファイル (`mfa_test_role`)。こうすることにより次のリストのように、ユーザーでコマンド実行時は権限不足で拒否され、ロールで実行時は、MFA のワンタイムパスワードの入力を求められ、合っていれば実行できるようになります。なお、デフォルトでは一度認証に成功すると 1 時間有効なので、2 回目以降は都度ワンタイムパスワードを入力する必要はありません。

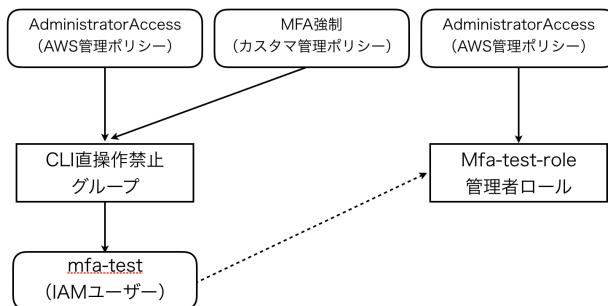


図 7.5: MFA の強制と CLI でのスイッチロール

第7章 IAM の運用 7.6 MFA 未利用時に権限を制限し、MFA 利用を促す

リスト 7.1: CLI での MFA 認証

```
1: $ aws s3 ls --profile mfa_test_user
2:
3: An error occurred (AccessDenied) when calling
4: the ListBuckets operation: Access Denied
5:
6: $ aws s3 ls --profile mfa_test_role
7: Enter MFA code for arn:aws:iam::021010746129:mfa/mfa-test:
```

MFA を強制するポリシーは、次のとおりです。

リスト 7.2: MFA 認証していない場合は、IAM 以外の権限拒否の IAM ポリシー

```
1: {
2:   {
3:     "Effect": "Deny",
4:     "NotAction": [
5:       "iam:*",
6:       "sts:AssumeRole"
7:     ],
8:     "Resource": "*",
9:     "Condition": {
10:       "BoolIfExists": {
11:         "aws:MultiFactorAuthPresent": "false"
12:       }
13:     }
14:   }
15: }
16: }
```

プロファイルの記載には、.aws 以下の credentials ファイルと config ファイルに記載します。CLI 利用時にも MFA を利用する方法としては、`~/.aws/config` にスイッチ先のロールを記載します。

第7章 IAM の運用 7.6 MFA 未利用時に権限を制限し、MFA 利用を促す

リスト 7.3: クレデンシャルファイルに記載されたユーザー

```
1: [mfa_test_user]
2: aws_access_key_id = AKIZZZZZZZZZZZZZZZZZ
3: aws_secret_access_key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

リスト 7.4: コンフィグファイルに記載されたロール

```
1: [default]
2: output = json
3: region = ap-northeast-1
4:
5: [profile mfa_test_role]
6: source_profile = mfa_test_user //切替元のユーザー名
7: mfa_serial = arn:aws:iam::999999999999:mfa/mfa-test //切替元の
   ユーザーの MFA デバイス
8: role_arn = arn:aws:iam::999999999999:role/mfa-test-role //切替先のロール名
```

7.7 マルチ AWS アカウントでの運用

最後にマルチアカウントでの運用です。マルチアカウントで問題になるのが、複数の IAM ユーザーの管理の問題です。グループやロール・ポリシーについては、CloudFormation で管理しておけば複数のアカウントに用意するのは、それほど手間ではありません。一方で、IAM ユーザーをそれぞれのアカウントに作るのは管理の手間が大きいです。そこでお勧めの方法が、踏み台 AWS アカウントです。IAM ユーザーは踏み台 AWS アカウントのみに作成し、各アカウントはロールとポリシーのみを作成します。これにより、アカウントの管理は一箇所で済みます。

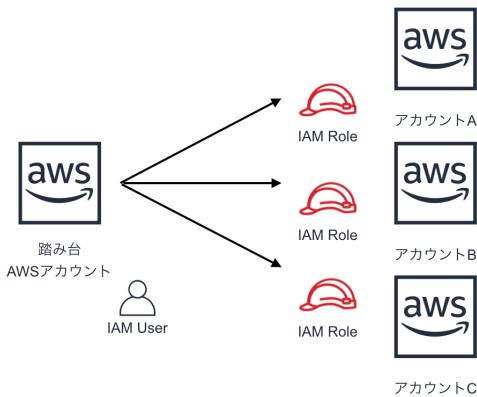


図 7.6: マルチ AWS アカウントでの運用

別の方法として、AD 等のディレクトリ・サービスと連携する方法も考えられます。しかし、認証の方は解決できるが権限である認可をどこで管理するのかという問題が出てきます。なかなか複数 AWS アカウントの場合、AD だけで管理するのは難しい印象があります。

7.8 IAM 運用のまとめ

IAM 運用の章では、そもそも運用の目的から役割と責任範囲の整理を行いました。IAM 運用の目的は、あくまで AWS を安全な状態に保ち続けるためです。そのために、現実的なコストで利用者全員が守れる方法で AWS を使っていく方法を考え実践していくのが運用です。

そのためには幾つか具体策がありますが、運用を始める前に利用者をしっかりと洗い出すことが大事です。ここでいう利用者とは人間に限りません。プログラムから使う IAM ユーザーもしっかりと把握しておきましょう。また実運用面では、実運用で悩む事が多いのがアクセスキーの管理です。

アクセスキーは CLI から使われる事が多く、AWS の不正利用の原因となることが多いです。そのため不要な場合はアクセスキーを発行しないという原則にするのが良いでしょう。そのうえで必要な場合は、制限と最小権限の付与です。制限方法としては、IP アクセス制限の他に、スイッチロールや MFA の活用がお勧めです。今回説明していませんが、Terraform や Ansible などのツールでアクセスキーが必要な場合は、IAM ロールを利用し、その一時的なクレデンシャルを利用するというのも良い手法でしょう。

運用方法については、環境によっては課題は変わりますが、共通する部分も多いので参考にしてください。

第8章

IAM と CloudFormation

ここまで IAM の設計の定石からセキュリティ・運用まで一通り理解できたかと思います。IAM をしっかりと使いこなしていくと、用途に応じて細かく IAM ユーザーやポリシー・ロールを作っていくことになります。では、それをどのように作っていけば良いのでしょうか？お勧めは CloudFormation (CFn) です。ここでは、CloudFormation で IAM を管理する利点と具体的な手法をお伝えします。

8.1 IAM と CloudFormation

CloudFormation は、テンプレートの記述を元に AWS のリソースを作成するサービスです。コードでインフラを管理する Infrastructure as Code を体現していると言えるでしょう。複数の AWS アカウントを管理する場合、それぞれのアカウントに同じポリシーやロールを作ることになります。CloudFormation 化しておけば、アカウント作成時に IAM 作成の CloudFormation を実行することで、手間を省き人為ミスを排除できます。

しかし、CloudFormation は便利なもの、最初のとっつきにくさから敬遠される事も多いです。IAM を管理するテンプレートは非常にシンプルなので、まず IAM で CloudFormation に入門することをお勧めします。

8.2 CFnの分割単位・依存関係

CloudFormation の機能の詳細の説明はしませんが、設計の上で重要なポイントを 1 つ挙げておきます。それは CloudFormation の分割単位です。初めて CloudFormation を作ろうとすると、それ一つで全部完結するものを作ろうとしてしまう事が多いようです。例えば、VPC のネットワークを作ってセキュリティグループなどの通信設定、ELB やその中の EC2 インスタンスまで全部作るといった長大な CloudFormation をイメージです。一つの環境を一つのコードで管理するのは、それはそれで素晴らしいです。例えばサービスの提供者が環境を提供するためにオールインワンの CloudFormation を作るというのは正しいです。一方で、自分あるいは自組織のために使う CloudFormation の場合にそこまでする必要があるのでしょうか？ 結論としては、できるだけ小さな複数の CloudFormation を作った方が使い勝手はよくなります。なぜならば、長大な CloudFormation は読むのも難しいし、更にメンテナンスをするのが難しくなります。難易度が高まると扱える人も少なくなります。そこで、できるだけ小さく作ることを心がけましょう。

それでは、実際に CloudFormation で IAM を管理する場合は、どのようにしたら良いのでしょうか？ IAM の主な機能としては、IAM ポリシー、IAM ロール、IAM グループ、IAM ユーザーがあります。IAM ポリシーがまずあって、その後に IAM ロール、IAM グループを紐付けて最後に IAM ユーザーと関連付けるという形です。

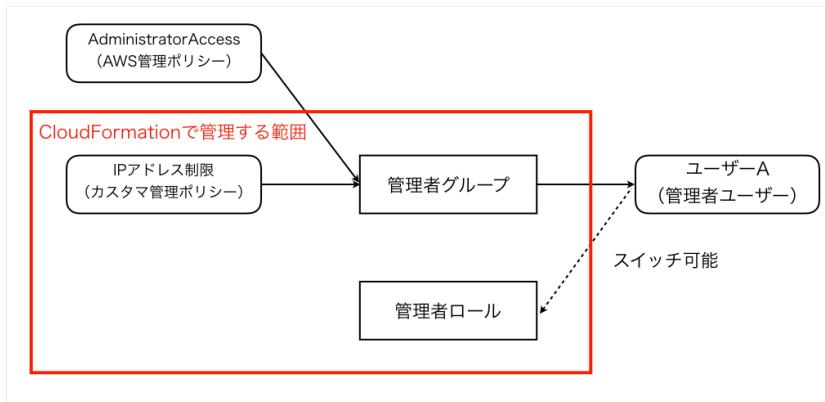


図 8.1: CloudFormation での管理する範囲

この中で、IAM ユーザーは CloudFormation の管理対象から外しましょう。IAM ユーザーは基本的に一度作ってお終いです。CloudFormation のライフサイクルで管理するメリットが殆どないです。後ほど解説しますが、CLI で IAM グループに紐付けるなどで充分です。では残った IAM ポリシー、IAM ロール、IAM グループです。CloudFormation からみると IAM ロールと IAM グループは同列の存在で、IAM ポリシーが先行関係にあります。IAM ポリシーと IAM ロールもしくは IAM グループを 1 つの CloudFormation で作るかどうか、少し悩ましい問題です。個人的な方針としては、一つの CloudFormation で管理しても、それほど複雑性はあがらないのでセットで管理してしまってもよいと考えています。

8.3 CloudFormationとIP制限

CloudFormationを使う上で、かなりの確率で引っかかるのがIP制限です。IP制限をしているアカウントでCloudFormationを実行しようとすると、権限を付与していても権限無しのエラーで失敗します。これは、CloudFormationがAWSの内部的なリソースで実行し、そのリソースのIPを元に権限評価されるためです。

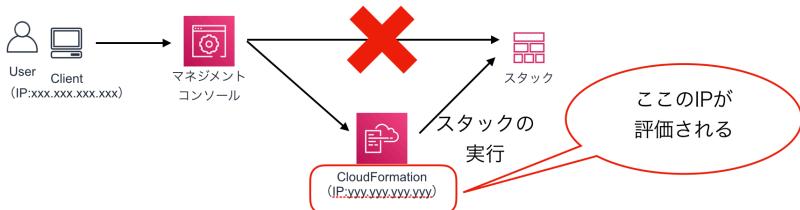


図 8.2: CloudFormation 実行時に IP 制限を受ける理由

IP制限が必要な場合での対応策としては、次の2つがあります。

- IP制限していないロールにスイッチしてから実行する
- CloudFormationの実行時ロール指定を利用する。

なお AWS コンソールから実行しても AWS の内部的なリソースで実行されるサービスは、CloudFormation以外にもいくつかあります。全てのサービスで CloudFormation のように実行時のロールを指定できる訳ではありません。ですので、IP制限なしのロールにスイッチできるように準備しておくのが良いでしょう。

具体的なポリシーについては、次章「IAMのテンプレート集」で見ていくましょう。

■コラム：ライフサイクルで考える

CloudFormationが敬遠される理由の一つとしては、初期の設計・学習コストが高いためにメリットを感じる前に挫折してしまう事が多いということがあります。CloudFormationは一度作ってしまえば、ずっと使えます。事実、CloudFormationの書式を表すテンプレート形式バージョンは、唯一"2010-09-09"があるだけです。つまり、書式は一度も変わっておらず、2010年を作ったものがそのまま2019年でも基本的には動くということを意味しています。AWSがそれだけ継続性を重視していると言えるでしょう。サードパーティ一製で対比される事が多いTerraformに対して、その点のメリットが大きいと言えるでしょう。

では、CloudFormationを挫折しないためにはどうすれば良いのでしょうか？それは、シンプルに割り切りです。次の3点を守っていれば、比較的簡単に使えるようになります。

- 全てのAWSリソースをCloudFormationで管理しようとする
- 1つのCloudFormationで完結させようとする
- 変更頻度・利用者が違うものを一緒に管理しない

1つ目の全てを管理しようとするという点ですが、CloudFormationのも得意とする範囲とそうでもない範囲があるからです。IAMやVPCの管理など比較的変更が少ないものについては、CloudFormationで管理しやすいです。一方で個々のEC2インスタンスのようなものは、CloudFormationで管理するには向いていません。ELBとAutoScaleのセットなどより上位の中で管理すべき項目です。なので、まずどの範囲をCloudFormationで扱うかの検討が必要です。最初は自分の手の届く小さなところから始めるのがよいでしょう。その点、IAMはお勧めです。

次に1つのCloudFormationで完結させようとしないという点です。

ある程度 CloudFormation に慣れてくると、複数のテンプレートに依存関係を作ってネストして実行するようなものを作りたくなります。自分の思い通り順序よく CloudFormation が実行され、次々と AWS リソースを作られていく様をみるのは万能感が出てきます。しかし、そのテンプレートを作り上げるコストに見合うものなのか冷静に判断する必要があります。CloudFormation を扱うのはプログラマーだけではありません。自分だけ理解できるものだと、そのメンテナンスは自分がし続ける必要があります。また、業務の中で CloudFormation を書き続けるということも少ないとと思うので、次に使うのが数ヶ月先ということもあるでしょう。その際に過去に自分が書いたテンプレートを理解できるかという問題がでてきます。割り切って実行する順番を書いた手順書と併用、或いは CLI との併用をお勧めします。

最後に変更頻度・利用者が違うものを一緒に管理しないという点です。先に例を挙げましたが、ネットワークの変更頻度と、その中の EC2 インスタンスの変更頻度は全然違います。また恐らく、それを管理する人も違うでしょう。これを一つのテンプレートとして管理してしまうと管理が難しくなります。なので、ネットワークならネットワークを管理するテンプレートとして作っていきましょう。これをしっかりと考えていくと、CloudFormation はとても使いやすくなります。

第9章

IAM のテンプレート集

この章では、具体的な IAM の構成をみていきましょう。テンプレートとしては、AWS 管理ポリシーと独自に作成したカスタマー管理ポリシーをグループに付与するという形で提供します。ポリシーとグループをそのまま使えるように、CloudFormation として提供します。単純化のために、ポリシーとグループを作成する 1 つのテンプレートとして作成しますが、運用実態に合わせて分割・修正して利用してください。

テンプレートのダウンロード

テンプレートは、次の URL (<https://books.takuros.net/aws-iam/templates.zip>) でダウンロードできます。自由に改変して利用してください。再配布もご自由にどうぞ。



図 9.1: CloudFormation テンプレート

9.1 共通系ポリシー

各グループの説明の前に、共通で使うポリシーを解説します。

自身のパスワードと MFA の設定権限

管理者グループを除き IAM の権限は与えません。しかし、パスワードや MFA の設定の権限は必要です。そこで、IAM のポリシー変数を利用して自分自身のパスワードを変更する権限を付与するポリシーを作成します。具体的なポリシーについては、前章の IAM とセキュリティで紹介しているので割愛します。

IP 制限と MFA の必須化のポリシー

次に IP 制限と MFA の必須日のポリシーです。IP 制限の方は比較的単純で、特定の IP 以外だったら拒否というポリシーです。IAM では明示的拒否が一つでもあれば、他のポリシーでいくら許可しても上書きされることはありません。必ず防ぎたいことは、明示的拒否で記述するようにしましょう。

次に MFA の必須化です。こちらも "Effect": "Deny" という明示的拒否を利用しています。そして、MFA を利用しているかどうかという判定に aws:MultiFactorAuthPresent というグローバル条件コンテキストキーを利用しています。またここで注目すべき点としては、NotAction を利用することで、iam 以外を指定します。この指定をしないと、初回に自身で MFA を設定することができません。

リスト 9.1: IP 制限と MFA の必須化

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Deny",
6:             "Action": "*",
7:             "Condition": {
8:                 "NotIpAddress": [
9:                     "aws:SourceIp": [
10:                         "8.8.8.8/32"
11:                     ]
12:                 }
13:             },
14:             "Resource": "*"
15:         },
16:         {
17:             "Effect": "Deny",
18:             "NotAction": [
19:                 "iam:*"
20:             ],
21:             "Resource": "*",
22:             "Condition": {
23:                 "BoolIfExists": [
24:                     "aws:MultiFactorAuthPresent": "false"
25:                 ]
26:             }
27:         }
28:     ]
29: }
```

9.2 管理者グループ

管理者グループは、全権を付与します。ただし作業場所を制限するためには、IP 制限のみ加えます。IP 制限があると、CloudFormation 等の一部の機能が使えない場合があります。そこで Switch Role を利用して IP 制限を解除できるようにしておきます。管理者権限ロールは、誰からでもスイッチできないようにユーザー名指定でスイッチの可否を制御します。

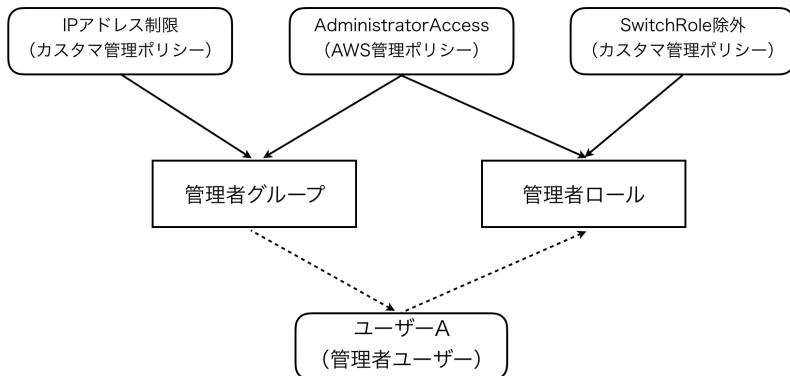


図 9.2: 管理者グループ

管理者グループ特有のポリシーとしては、SwitchRole の抑制のポリシーです。管理者ロールにつけておかないと、全てのユーザーから管理者ロールにスイッチされることになります。対応策としては、次の 2 点があります。

- ロールに、スイッチ可能なユーザーを記載する。
- ユーザーにスイッチロールの権限を拒否し、必要なユーザーのみスイッチロールの権限を付与する

ユーザー記載の場合、対象ユーザーが増えるたびにロールに付与している

ポリシーを変更する必要があります。スイッチロール原則拒否の場合、全ユーザーにその権限を付与する必要があります。どちらも一長一短ありますので、自身の運用ポリシーと照らし合わせた上で選択してください。

ここでは、ロールにユーザーを記載するパターンを採用します。記載範囲が少なく済むという点と、管理者ユーザーはそれほど増えないという前提にたっています。

スイッチロール可能なユーザーの絞り込みには、Principal を利用します。Principal で信頼するリソースを指定することが可能です。sts:AssumeRole はロールを引き受けられる権限で、sts は Security Token Service (STS) という機能を指しています。STS は IAM のサブセットのような機能で、一時的セキュリティ認証情報を管理するという重要な役割を果たします。ロールや Cognito は、内部的にはこの STS を利用しています。

リスト 9.2: スイッチロール可能なユーザーの絞り込み

```
1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Principal": {
7:         "AWS": [
8:           "arn:aws:iam::999999999999:user/user-name-1",
9:           "arn:aws:iam::999999999999:user/user-name-2"
10:        ]
11:      },
12:      "Action": "sts:AssumeRole",
13:      "Condition": {}
14:    }
15:  ]
16: }
```

9.3 ネットワーク管理者グループ

ネットワーク管理者グループは、VPC の作成や VPN の設定などのネットワーク、セキュリティグループやネットワーク ACL などの通信制御の権限を持ちます。それに加えて、全てのサービスの参照権限のみ付与します。

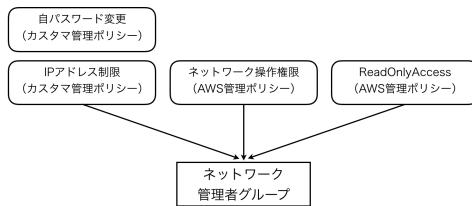


図 9.3: ネットワーク管理者グループ

このグループの肝は、ネットワーク管理系の権限を包括したポリシーを作るところです。コラム「EC2 の権限範囲の問題」で記載したとおり、EC2 の権限の中にネットワークに関係する権限が含まれています。その部分のみピックアップするのが、正直難しいです。

そこで、職務機能の AWS 管理ポリシーと呼ばれる AWS 管理ポリシーの利用をお勧めします。その中の NetworkAdministrator を利用すると必要な権限がほぼ網羅されています。https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/access_policies_job-functions.html



図 9.4: 職務機能 AWS 管理ポリシー

9.4 開発者グループ

開発者グループは多くのリソースの制御が必要なので、一番悩ましい権限セットになります。ここでの開発者の想定は、主に EC2 インスタンスを立ち上げて細かな調整をしつつインフラとシステムを作り上げていくということです。悩みの根本の部分は、今や EC2 に IAM ロールを付与するというのが必須であるということです。IAM ロールの作成・更新に関する権限を付与すると実質的に何でもできます。現実的な対応としては、開発環境では管理者権限を付与した上でしっかり監査証跡を残すという形になりますが、ここでは PowerUserAccess とネットワーク権限と Lambda の権限付与機能を剥奪したポリシーを組み合わせて提供します。

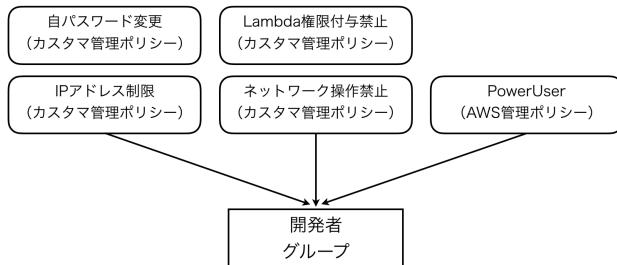


図 9.5: 開発者グループ

ネットワーク権限の剥奪は、AWS 管理ポリシーである NetworkAdministrator を参考に、Deny で反転させることによりポリシーを作成します。併せて、Lambda の権限付与機能も剥奪したポリシーを作成します。ネットワーク権限については、非常に広範囲に渡るため、章冒頭で紹介しているテンプレートをダウンロードして、CloudFormation を参照してください。

9.5 オペレーターグループ

オペレーターグループに必要な権限は、どこまで担当するかに依存します。ここでは AWS の状態を確認し、場合によっては EC2 インスタンス等の再起動を行うという前提で権限を付与します。また、状況によっては AWS サポートに問い合わせが必要なので、その権限 (AWSSupportAccess) も付与します。AWS 管理ポリシーとして SupportUser という support 権限と参照のみの権限を付与した職務機能の AWS 管理ポリシーもありますが、参照範囲が ReadOnlyAccess より少ないので、AWSSupportAccess と ReadOnlyAccess を組み合わせて使います。

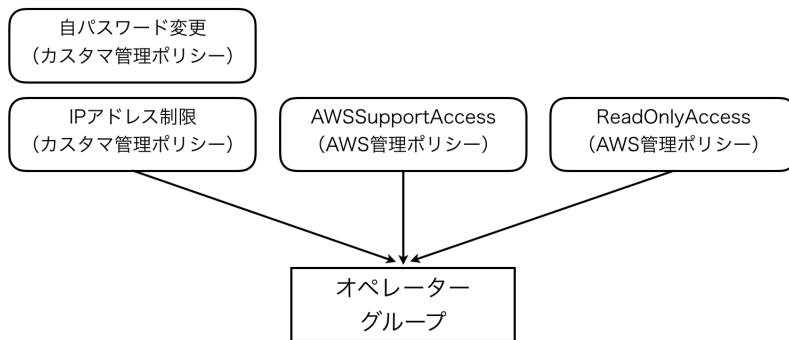


図 9.6: オペレーターグループ

オペレーターについては、担当する範囲によって権限が大きく変わっていきます。業務の担当範囲を決めた上で、このテンプレートに対して、権限を足していくという形が良いでしょう。

9.6 経理担当者グループ

ここで一風変わって経理担当者のグループ権限を考えてみましょう。経理担当者は、請求画面を見る必要があります。

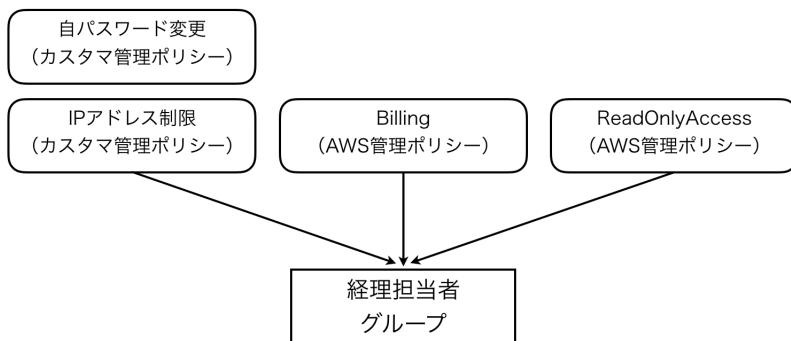


図 9.7: 経理担当社グループ

こちらも Billing という職務機能の AWS 管理ポリシーを利用しましょう。ここでは実際のリソースを確認する必要性を考慮して、参照権限も付与しております。運用の実態に即して変更してください。

9.7 お一人様 AWS

個人で AWS を利用する場合のテンプレートです。この場合は管理者グループと同一でも問題ないのですが、Admin 権限を普段使いするのも危険です。そこで、普段はグループには参照権限を付与して、必要な時にスイッチロールするというパターンがお勧めします。

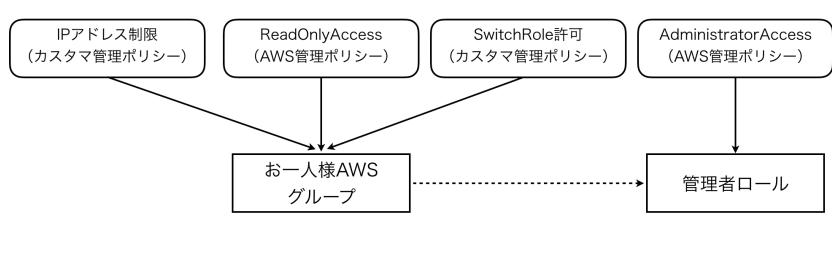


図 9.8: お一人様 AWS

リスト 9.3: グループ側にスイッチ可能なロールの指定

```
1: {  
2:   "Version": "2012-10-17",  
3:   "Statement": [  
4:     {  
5:       "Effect": "Allow",  
6:       "Action": "sts:AssumeRole",  
7:       "Resource": "arn:aws:iam::999999999999:role/admin-role",  
8:     }  
9:   ]  
10: }
```

このテンプレートの特徴としては、ロール側ではなくグループ側に AssumeRole できるリソースを指定しています。このやり方だと、対象ユーザーが増えた場合に都度ロールの信頼関係を変更する必要が無いのがメリットです。

なお CloudFormation を使うとアカウント ID を環境変数として取得できます。仮に 999999999999 と置いている部分は自動的に自分の AWS アカウント ID に置き換えられます。

9.8 IAM のテンプレートのまとめ

代表的なケースをテンプレートとしてまとめてみました。AWS の利用方針が決まると、それに応じた権限のセットが決まります。一度 CloudFormation でテンプレートかしておこうと、AWS アカウントが増えたとしても簡単に使いまわしができます。CloudFormation の入門にもつながるお勧めの利用方法なので、是非挑戦してみてください。

また AWS 管理ポリシーを上手く使うことにより、カスタマー管理ポリシーは最小限にできます。例に挙げたネットワーク管理者権限は、初期には提供されておらず EC2 の権限を一生懸命読み解いて対応してたものです。どんどん使いやすくなっているので、自作のポリシーを作成する前に一度調べて見ましょう。昔は無かったものが増えているかもしれません。

第 10 章

IAM 以外の AWS サービスの活用

これまでに IAM を利用する上で、必要な事を書き連ねてきました。単一アカウントで運用する際に遭遇する検討事項の 7 割くらいのケースはカバーできているのではないかと思います。一方、これだけをやっていれば完璧という訳ではありません。IAM は AWS を安全に使う上で要となる機能ですが、それ以外にも様々なサービスを併用しておく必要があります。この章では、代表的なサービスの紹介をします。

10.1 AWS Organizations（組織アカウント）

AWS を本格的に使っていくと、単一 AWS アカウントでの運用では無理が生じることが多いです。そこで多くの企業では、複数の AWS アカウントを利用して運用しています。AWS でも、複数の AWS アカウントを前提としたサービスを矢継ぎ早にリリースしています。その代表格の一つが、AWS Organizations です。

AWS Organizations の機能としては、主に 2 つあります。従来からあった一括請求機能を内包して、かつ組織内の AWS アカウントに対して IAM

のようなポリシーコントロールができます。AWS Organizations には、ルートと呼ばれるマスター アカウントと、それに紐づく子アカウントがあります。また、子アカウントを階層化するための組織単位があります。また、アカウントに対するポリシーという形で、ホワイトリスト／ブラックリスト形式で権限の管理ができます。上位で決めたポリシーは、個々のアカウント単位で打ち消すことはできません。この機能があるために、AWS アカウントに対して強力な統制を掛けることができます。

10.2 CloudTrail と Config

AWS を安全に使うには、監査可能な状態にする事が大切です。ここでいう監査とは、誰が AWS を操作したのか、また AWS のリソースがどのような状態になったのかということです。その為のサービスとして、AWS には CloudTrail と Config というサービスがあります。

CloudTrail は、AWS マネジメントコンソールやコマンドラインツール、その他の AWS サービスから実行履歴をログとして取得します。このログをみれば、誰がどの操作を行ったか一目瞭然となります。一方で、ルートユーザー や IAM ユーザーを共用で使っていると、ログ上からは誰が利用していたのか解らなくなります。そういう事もあり、ルートユーザーの利用禁止 や IAM ユーザーを一人 1 アカウント化が大切な事が解ります。

CloudTrail と対となるようなサービスが Config です。Config は、リソースの状況を断面として評価します。操作ログからリソースがどのような状況になっていったかを追うのは難しいですが、Config を使うとどういう状態になったのか、またそれに対してどうアクションするのかといった設定ができます。例えば、運用ルールとして EC2 インスタンスにタグで名前をつけることを必須としていてたとすると、Config を使うことによりそれを満たしていない場合にインスタンスを強制的にシャットダウンするといったようなことができるようになります。

10.3 Amazon GuardDuty

AWSを利用する上では、IAMやCloudTrail、Configなどを使って開発・運用を正しくしていくだけでは足りません。不正侵入や脆弱性をついた攻撃など、外部からの脅威に対する備えが必要です。攻撃者は機械的に24時間絶え間なくやってくるので、対応する方も24時間の防御が必要です。AWSではGuard Dutyと呼ばれる脅威検出と継続的なモニタリングサービスが提供されています。上手く使うことにより、脅威の自動検出と、それに対応した防御というのを機械的にすることも可能になります。

10.4 AWS Control TowerとAWS Security Hub

この章の始めに説明したとおり、今や組織でのAWS利用は複数アカウントが主戦場です。そうなると、複数アカウントで同じレベルでセキュリティを保つ必要があります。1箇所でも不備があると、そこから重大なインシデントが発生する可能性がでてきます。では、場合によっては何十、何百に及ぶアカウントを手動で管理して同じレベルを保持できるのか？中々難しいですよね。そういういたケースを考慮したサービスとしてControl TowerとSecurity Hubがあります。

Control Towerは複数のアカウントの設定および管理をするサービスです。そしてSecurity Hubは、複数のアカウントの状況を一括してモニタリングするサービスです。まだ一般公開されたばかりのサービスで馴染みは少ないですが、今後重要度が増してくるサービスです。

10.5 まとめ

このように AWS は、より安全に AWS を使えるようなサービスをどんどんと出してきています。サービスを上手く使うことにより、より簡単により安全に AWS アカウントを運用することができるでしょう。しかし、そのためには我々ユーザー側としては、サービスを踏まえた上で体系的にどのように運用していくかの知見が必要です。またそのサービスを組み合わせたベストプラクティスも刻々と状況が変わるので、アップデートしていく必要があります。

AWS のサービスのアップデートに追随し続けるのは中々難しいのは事実です。一方で、セキュリティの基本的な考え方というものはドラスティックに変わるものではありません。セキュリティの高めるためにどうすれば良いのか、例えばどこに穴があるのかそれをどうすれば埋めることができるのかという考え方は、一度身につけると長い間使えます。その上で AWS のサービスをどう使っていけばよいかという思考ができれば、きっと安全なシステムが作れるでしょう。これを機会に皆さんも考えてみてください。

私も今回 IAM の使い方を整理することにより、より深く考える必要性を痛感しました。次回、この章で紹介したサービスを使いながら、AWS をどのように運用していくべきかまとめる本を作成することにしました。またお会いできる機会あれば幸いです。それでは、AWS の世界を楽しんでください！！

付録 A

アカウント開設時の設定 チェックリスト

最後に、アカウント開設時に設定すべき項目のチェックリストを掲載します。

1. ルートアカウントの MFA 設定
2. IAM Group と IAM ユーザーの作成
3. IAM パスワードポリシーの適用
4. CloudTrail の有効化
5. Config の有効化
6. GuardDuty の有効化
7. Trusted Advisor の E メール通知設定
8. Cost Usage Report の出力
9. ルートアカウントで IAM User への請求情報へのアクセス許可
10. 支払い通貨を日本円に変更
11. コスト配分タグの設定
12. 代替連絡先の設定

付録 A アカウント開設時の設定チェックリスト

本書は IAM をメインのテーマとしているため、解説していない部分も多いです。設定のポイントとしては、次の 3 つに分かれています

- セキュリティの強化
- 状況の出力・見える化
- 機能の有効化

1~3 番からは主にセキュリティの強化で、本書でも解説した部分です。4 ~8 番は状況の出力・見える化です。CloudTrail により AWS 操作ログの出力、Config で設定状態の管理、GuardDuty は AWS に対する脅威の検出をします。Trusted Advisor は、AWS の利用の仕方に対してアドバイスです。E メール通知をしておくと、定期的にレポートを送ってくれます。Cost Usage Report はコストレポートを設定すると、詳細のレポートが見られるように設定できます。

9~12 番は機能を使うための設定です。まず請求情報へのアクセス許可を与えておかないと、IAM で権限を与えても請求情報を見ることができません。必ず設定しましょう。支払い通貨を日本円に設定しておくと、クレジットカード会社より有利なレートで日本円に算出してくれる事が多いので、設定をお勧めします。またコストの状況を詳しく把握したい場合は、コスト配分タグを利用します。また、AWS アカウント作成に利用した E メールアドレス以外に通知したい場合は、代替連絡先を設定しましょう。

後書き

毎年年初の目標を立てて、2015 年から Kindle 本を出すと宣言しつつ、一度も達成したことがありません。その間、10 冊近い本を出しているのですが、締切が無いと全く筆が進みません。これは自分の意志の力では無理だと悟り、なにか強制力の必要を感じていました。

そんな折に参加したのが、技術書典 6 です。それまで、技術同人誌というジャンルがあることは知っていたのですが、実物も見たこともなく、どれくらいの盛り上がりなのか肌感覚として知りませんでした。そんな中で 14 時過ぎにフラッと参加してみました。事前の予測では、昼過ぎになると空いてきて余裕を持ってみれると聞いていましたが、そんなことはなく入場待ちの長蛇の列でした。入ってみても、大変な混雑！！

その熱気を体験して、これは私もやらなければと思い、今回の技術書典 7 に初参加です。執筆の経験はあるので余裕と思っていましたが、これが大間違い。執筆だけするのと、自分で全部するのは全然違いました。校正や製本といったプロセスを体験し、締め切りの大変さを実感することができました。いや、これを毎回やってる同人界の皆さん、本当に凄いですね。

さて、今回は初の参加ということで、手堅いテーマで挑戦しています。しかし、書いているうちに、やっぱり IAM だけでは語り尽くせないことが解りました。次はその宿題をやります。また、AWS 以外のジャンルの開拓も目指します！！ 引き続きのご愛顧、宜しくおねがいします。

(2019 年 9 月 15 日 佐々木 拓郎)

AWS の薄い本 IAM のマニアックな話

2019 年 9 月 22 日 初版第 1 刷 発行

2020 年 3 月 5 日 初版第 2 刷 発行

著 者 佐々木拓郎

デザイン 佐々木さやか

印刷所 株式会社 日光企画
