TECH CON

DeNA TechCon 2021

AWS IAMの属人的な管理からの脱却

Kenta Sato (@karupanerura) DeNA, Co. LTD.

@karupanerura

DeNA - 認証認可基盤周辺サービス開発・運用
Japan Perl Association - 代表理事

Perl/Go/Java/TypeScript/MySQL/etc..

Twitter/Github: @karupanerura

話すこと

- IAM管理のベストプラクティスのおさらい
- IAM管理の難しさとその問題
- IAM管理の部分的な委譲による解決
- それを応用した管理のIaC化

話さないこと

- セキュリティの基礎
- AWSやIAMの基礎
- 各種AWSサービス
- IAM管理の詳しい運用方法
- AWSアカウントのマルチアカウント管理
- AWS CDKなどの仕組みの詳細

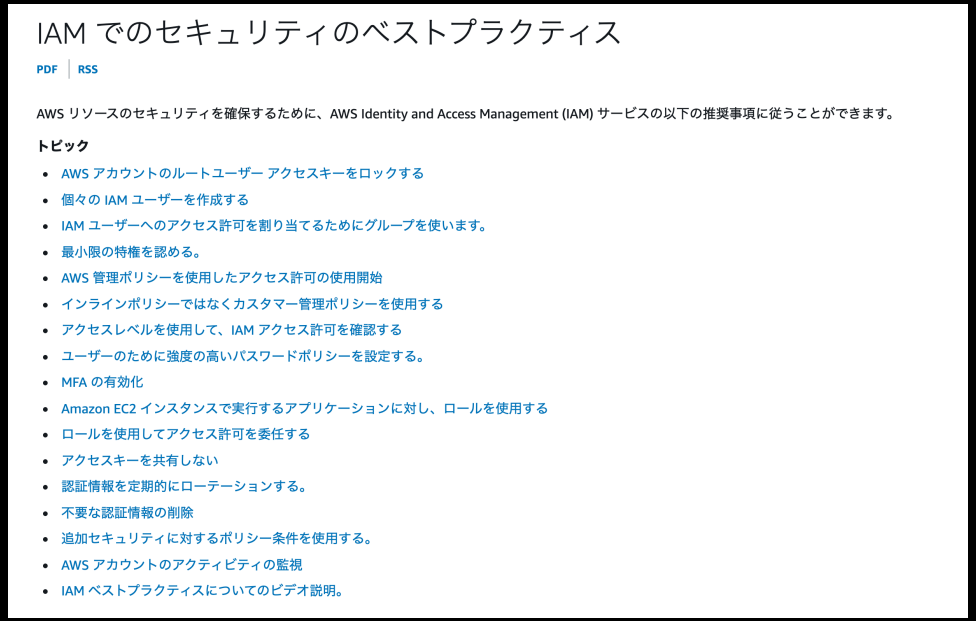
はい

話すこと

- **IAM管理のベストプラクティスのおさらい**
- IAM管理の難しさとその問題
- IAM管理の部分的な委譲による解決方法
- それを応用した管理のIaC化

IAM管理のベストプラクティス

IAM でのセキュリティのベストプラクティス



IAM でのセキュリティのベストプラクティス

PDF | RSS

AWS リソースのセキュリティを確保するために、AWS Identity and Access Management (IAM) サービスの以下の推奨事項に従うことができます。

トピック

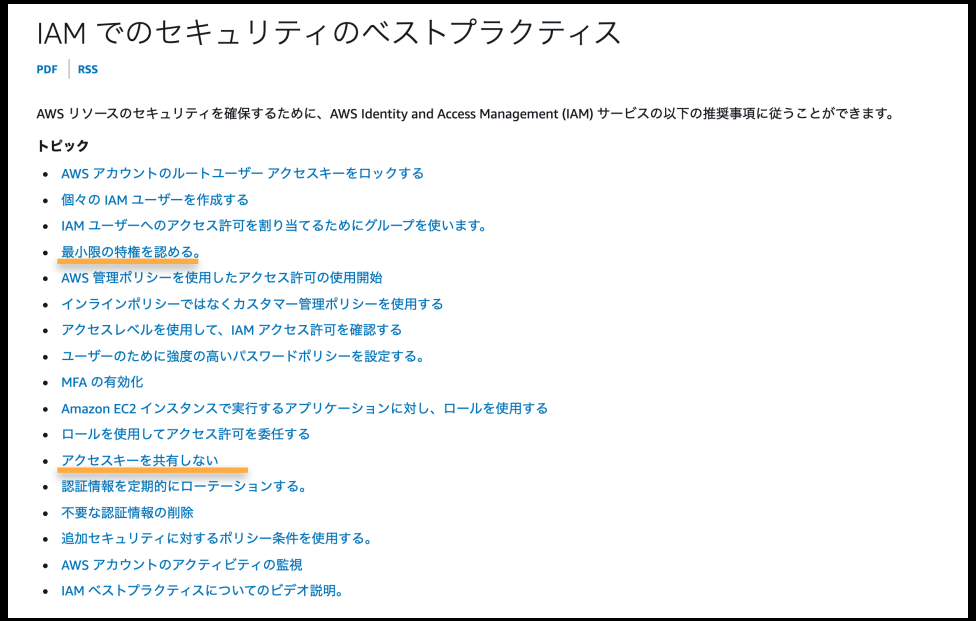
- AWS アカウントのルートユーザー アクセスキーをロックする
- 個々の IAM ユーザーを作成する
- IAM ユーザーへのアクセス許可を割り当てるためにグループを使います。
- 最小限の特権を認める。
- AWS 管理ポリシーを使用したアクセス許可の使用開始
- インラインポリシーではなくカスタマー管理ポリシーを使用する
- アクセスレベルを使用して、IAM アクセス許可を確認する
- ユーザーのために強度の高いパスワードポリシーを設定する。
- MFA の有効化
- Amazon EC2 インスタンスで実行するアプリケーションに対し、ロールを使用する
- ロールを使用してアクセス許可を委任する
- 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。
- 不要な認証情報の削除
- 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のビデオ説明。

https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/best-practices.html

今回の話を理解する上で

特に大事なところ

IAM でのセキュリティのベストプラクティス



IAM でのセキュリティのベストプラクティス

PDF | RSS

AWS リソースのセキュリティを確保するために、AWS Identity and Access Management (IAM) サービスの以下の推奨事項に従うことができます。

トピック

- AWS アカウントのルートユーザー アクセスキーをロックする
- 個々の IAM ユーザーを作成する
- 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います。
- 最小限の特権を認める。
- 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 アクセス許可を確認する
- ユーザーのために強度の高いパスワードポリシーを設定する。
- MFA の有効化
- 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
- ロールを使用してアクセス許可を委任する
- 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。
- 不要な認証情報の削除
- 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のビデオ説明。

https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/best-practices.html

最小限の特権を認める

- 必要以上の権限をつけない
- 最小限のアクセス権限から徐々に育てていく

何故か？

→ 必要以上の権限が付与された状態の常態化を防ぐため

必要以上に権限を持つと認証情報の漏洩時のリスクが高くなる

アクセスキーを共有しない

- 同一IAMユーザーのアクセスキーを複数箇所で共有しない
- IAMロールの利用を推奨する

何故か？

→ アクセスキーを持つシステムを増やさないため

アクセスキーを使う箇所が増えれば漏洩する可能性も増える

つまり

IAM管理のベストプラクティス

- 個々個別の事情に合わせた最小限の権限を持つIAMロールを使う
  - 必要以上の権限をつけない
- IAMユーザーはシステムユースでは原則として使わない
  - GCPから利用する場合などは別

はい

話すこと

- IAM管理のベストプラクティスのおさらい
- **IAM管理の難しさとその問題**
- IAM管理の部分的な委譲による解決
- それを応用した管理のIaC化

IAM管理の難しさ

IAMポリシーは複雑化しがち

- ResourceやConditionによる柔軟で細かい権限管理が可能
  - 単純にActionを付けるだけでは十分に権限を絞れない
  - S3やKMSの利用では実用上Resourceの制限も必要になる
- AWS管理ポリシーにはリソース制限が無い
  - 権限を十分に絞るには自分でポリシーを作るほうが良い
- 結果的にIAMポリシー管理にはノウハウが必要になる

権限を管理する権限が必要になる

- 「IAMを管理する権限」は「権限を管理する権限」である
  - つまり自分自身の権限もコントロールできる
  - 実質的にすべての権限を持つことにほぼ等しい状態
- 無闇に全員がこれを持つと権限管理する意味がなくなってしまう
  - 一部のメンバーだけがIAM管理権限を持つようにすれば解決する
  - 一般的には依頼制(チケット制)となることが多そう

DevOpsの境目が曖昧

- クラウドインフラにおいてはDevとOpsの境目は曖昧
  - IaaSから離れていけばいくほどこれは顕著になっていく
- Devの文脈で新しいインフラストラクチャを作ることになる
  - それに合わせて新しい権限セットが必要になることが多い
  - LambdaやFirehoseに付与するIAMロールなど
- 開発規模が広く多様化すると、依頼制では齟齬が起きやすくなる

おわかり
いただけただろうか…

もう一度ご覧いただこう…

IAM管理のベストプラクティス

- **個々個別の事情に合わせた最小限の権限を持つIAMロール**を使う
  - 必要以上の権限をつけない
- IAMユーザーはシステムユースでは原則として使わない
  - GCPから利用する場合などは別

権限を管理する権限が必要になる

- 「IAMを管理する権限」は「権限を管理する権限」である
  - つまり自分自身の権限もコントロールできる
  - 実質的にすべての権限を持つことにほぼ等しい状態
- 無闇に全員がこれを持つと権限管理する意味がなくなってしまう
  - **一部のメンバーだけがIAM管理権限を持つようにすれば解決する**
  - 一般的には依頼制(チケット制)となることが多そう

無数に発生する
個々個別のIAMロール

それらを全て管理する
一部のメンバー

開発規模に対して
管理がスケールできない

\(^o^)/

どうすれば良いのか

解決方法

話すこと

- IAM管理のベストプラクティスのおさらい
- IAM管理の難しさとその問題
- **IAM管理の部分的な委譲による解決**
- それを応用した管理のIaC化

IAM管理の部分的な委譲

そもそもの問題

- IAMリソース全体を管理するか一切管理しないかの極端な二択
  - 開発規模がスケールしてもIAM管理者がスケールできない
    - 一部の個人が管理担当者としてそれをやるしかない為
- 結果的にIAMユーザーやIAMロールの使いまわしで妥協しがち
  - もちろん潜在的なセキュリティーリスクが付きまとう

そもそもの問題

- **IAMリソース全体を管理するか一切管理しないかの極端な二択**
  - 開発規模がスケールしてもIAM管理者がスケールできない
    - 一部の個人が管理担当者としてそれをやるしかない為
- 結果的にIAMユーザーやIAMロールの使いまわしで妥協しがち
  - もちろん潜在的なセキュリティーリスクが付きまとう

分担はしたいけれど
全権持った人が増えすぎても困る

「このLambdaあたりの権限は
そっちでよしなに管理してね」

みたいに分担したい

All or Nothingは困る!!!

もし部分的な管理を委譲して分担することができたら…？

解決しそう

どうやって？

部分的なIAM権限操作ができる権限なんて作れるのか？

つくれます

Permissions Boundary

- 権限の境界を定めることができる機能
  - 一般的には組織間でIAM管理権限の一部を委譲のために使われる
- IAMユーザーやIAMロールに設定することができる
  - これが設定されているIAMリソースは、どのような権限を持っていても、この境界を超えた権限を行使することはできなくなる
  - エラーは単に権限を持っていないときと同じものが発生する

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html

どのように権限の境界を定めるのか

- 実態としては普通の一般的な管理ポリシー
  - Permissions Boundaryとして付与した場合に特殊な働きをする
- Permissions Boundaryで許可されていないActionは使えなくなる
  - インラインポリシーなどで許可されていてもそれは使えなくなる
  - 単に許可していないときと発生するエラーは変わらない
- (当たり前だが)ここでIAMリソースの操作権限を許可すると権限昇格ができるので本末転倒になってしまう。やってはいけない

どのように権限の境界を定めるのか

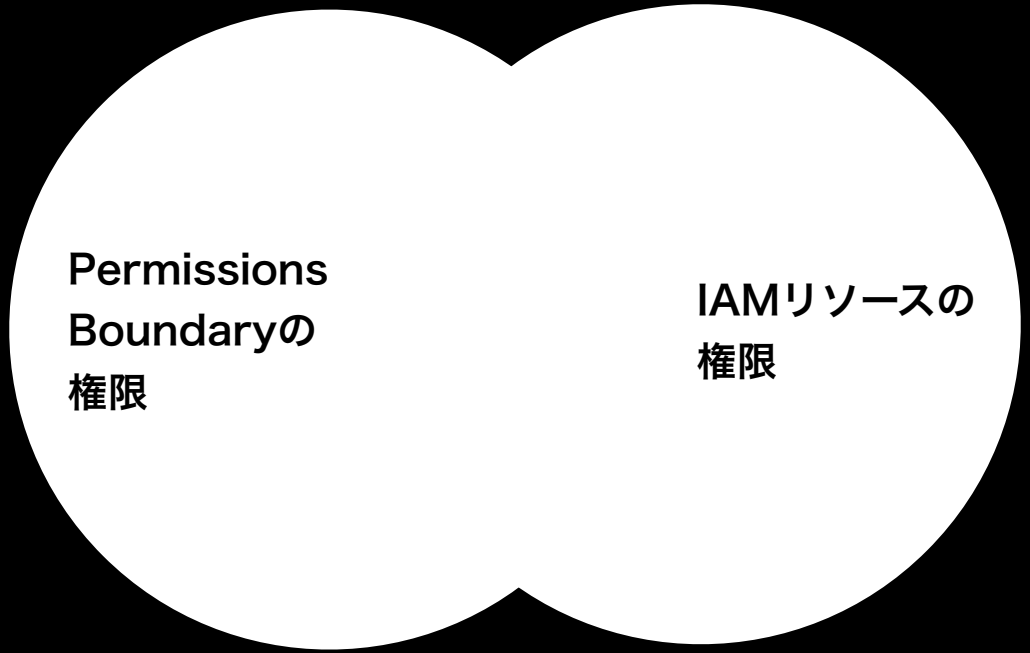
- 一般的にはサービス単位で指定することが多い
- 右の例のポリシーをPermissions Boundaryに指定した場合は、S3とCloudWatchに関係する権限だけが機能する

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*"
            ],
            "Resource": "*"
        }
    ]
}
```

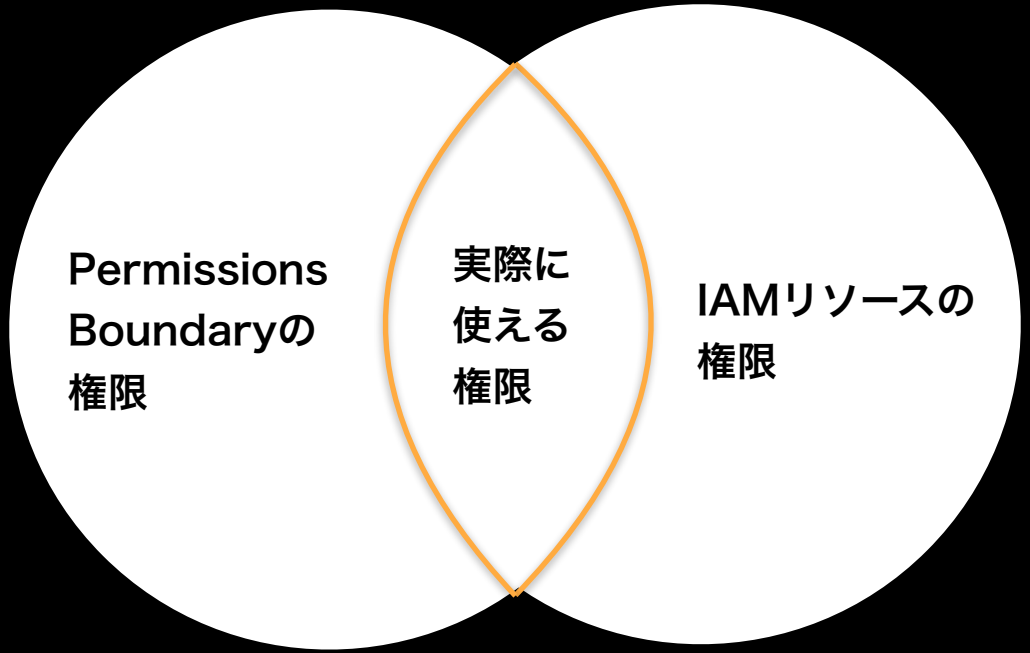
つまり

単にIAMリソースの権限だけがある状態

そこにPermissions Boundaryがつくと



実際に使える権限を制限できる



Q.付けなかったり外されたりしない？

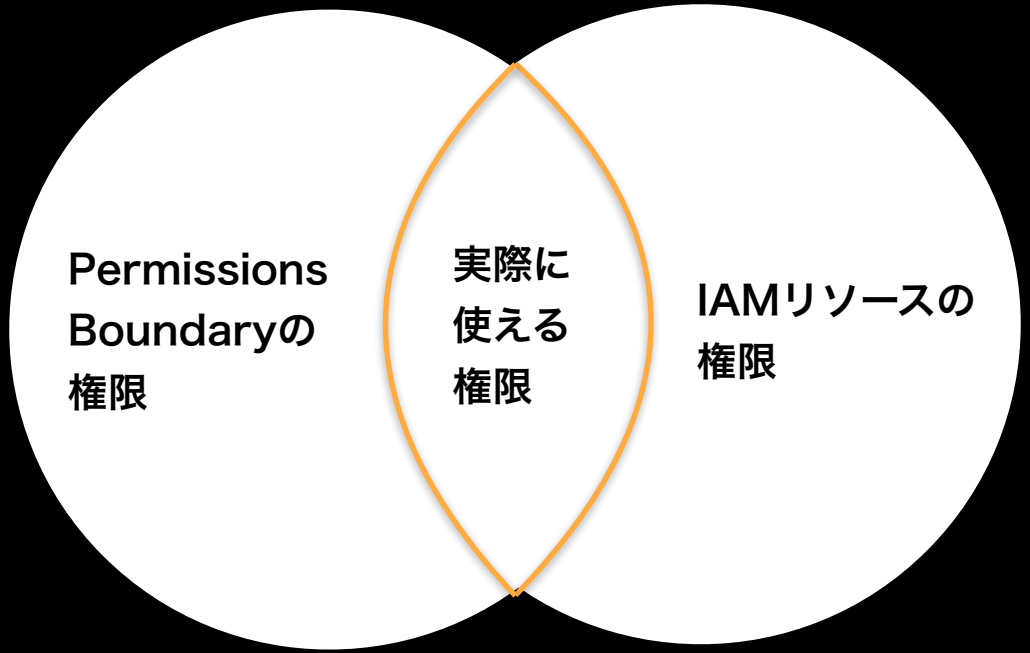
A.付与を強制すればよい

Permissions Boundaryの付与を強制

- IAM Conditionである **iam:PermissionsBoundary** を使う
  - 指定したPermissions Boundaryを付与したIAMリソースのみに対してのみ利用できる権限を表現できる
- 実質的に特定のPermissions Boundaryの付与を強制できる
  - 指定したPermissions Boundaryを付与しないリソースに対しての操作を拒否できる

実際に使える権限の制限を強制できる

外せない！



どのように使うか

IAM権限管理の部分的な委譲の実現

1.権限範囲を表現するPermissions Boundary用管理ポリシーを作る

2.部分的なIAM管理権限を与えたいIAMリソースに対し..

2.1. iam:PermissionsBoundaryで上記のPermissions Boundaryで制限したIAM操作権限を付与する

2.2. (必要なら)自分自身のIAMリソースに対する操作も禁止する

できました

これでIAM管理を分担しやすくなった

やりましたね

しかし

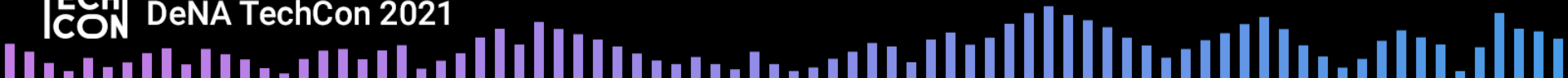
まだ問題は残っている…

思い出してみましょう

IAMポリシーは複雑化しがち

- ResourceやConditionによる柔軟で細かい権限管理が可能
  - 単純にActionを付けるだけでは十分に権限を絞れない
  - S3やKMSの利用では実用上Resourceの制限も必要になる
- AWS管理ポリシーにはリソース制限が無い
  - 権限を十分に絞るには自分でポリシーを作るほうが良い
- **結果的にIAMポリシー管理にはノウハウが必要になる**

個人でなく組織として IAM管理ノウハウを 蓄積したい



理想

組織で管理する理想のIAM管理環境

- 必要最小限の人が管理権限を持っている
  - この人数を規模に合わせて増減できる
- 作業者は誰でも変更をリクエストできる
- 変更内容は客観的に齟齬の無い表現で記載できる
- いつだれがなんのためにどのような変更をしたのかを追跡できる
- 変更の際に気をつけるべきノウハウが蓄積できる

おや

Infrastructure as code で解決しそう

組織で管理する理想のIAM管理環境

- 必要最小限の人が管理権限を持っている → Github Review
  - この人数を規模に合わせて増減できる → Github Team
- 作業者は誰でも変更をリクエストできる → Pull Request
- 変更内容は客観的に齟齬の無い表現で記載できる → Code
- いつだれがなんのためにどのような変更をしたのかを追跡できる
- 変更の際に気をつけるべきノウハウが蓄積できる → Git

Pull-Request出して
レビューしてマージして
CIで自動同期

したい

やりましょう

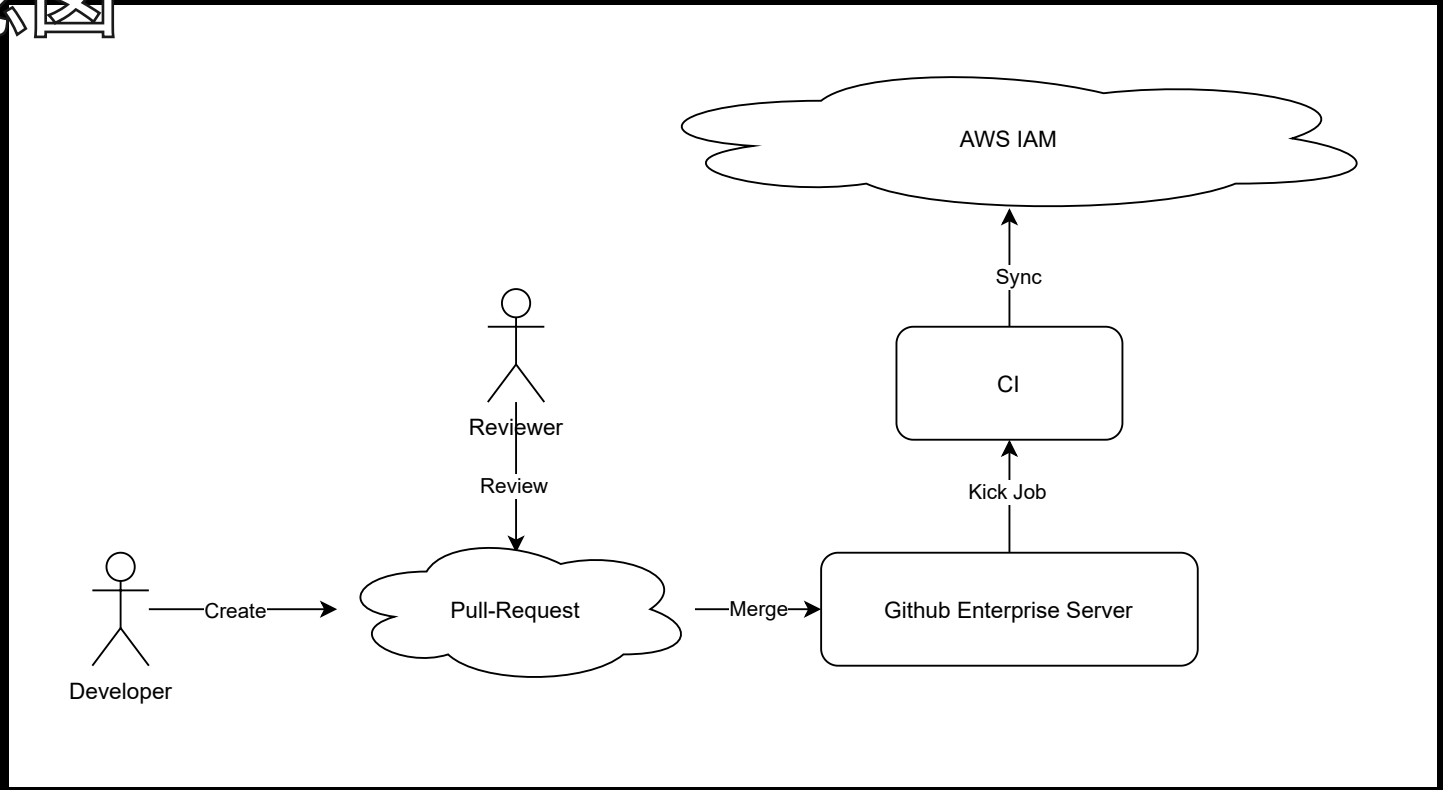
話すこと

- IAM管理のベストプラクティスのおさらい
- IAM管理の難しさとその問題
- IAM管理の部分的な委譲による解決
- **それを応用した管理のIaC化**

IaC

Pull-Request出して
レビューしてマージして
CIで自動同期

概念図


AWS IAM
Sync
CI
Kick Job
Github Enterprise Server
Merge
Pull-Request
Create
Developer
Reviewer
Review

よっしゃ
やっていくぞ！！！！！

その前に…

IAM変更適用を自動化するリスク

- CI JobがIAM管理権限を持つことになる
  - IAMの全権を持つと内部不正行為者に悪用された際のリスクが高い
  - 部分的なIAM管理権限を持つ場合はそのリスクは限定的
- リポジトリにアクセス権限があれば誰でも変更できる
  - レビューがザルだと悪意のある権限を見落としかねない
- CIの構成自体に脆弱性が無い限りは外部から攻撃する余地はない

リスクに対する考え方

- 内部不正行為者の余地しかない && 自動化せずとも常にあるリスク
  - IAM管理権限を持っている人が居る限りは起こり得る
- 不正行為をしにくくする仕組みを作ることでリスクを下げられる
  - CloudTrailのIAM APIイベントをSlackに通知
    - IAMリソースの変更などを検知/周知する
    - 変更機会は比較的少ないので変な操作があっても気づきやすい
- Permissions Boundaryで範囲を限定し被害リスクを下げられる

仕組み次第で
十分にリスクを下げられる

よっしゃ
やっていくぞ！！！！！

理想のIaCによるIAM管理

- GithubにPull-Request(PR)を作りマージされたらCIで自動同期
  - コード(設定)を正として差分同期したい
    - 全同期ではSDKなどのスロットルが掛かり時間がとても掛かる
  - 定期的にAWS側を正とした差分同期を取り込みたい
    - 緊急時などに行った直の変更が意図せず巻き戻されないように
- Linterを用意してポリシーなどの不備を自動レビューしたい
  - ノウハウをPRだけではなくコードとしても蓄積できる

どうやって同期するか？

同期手段の選択肢

- Terraform
  - pros: importも可能 / 独自DSLで自由度が高すぎない
  - cons: stateの管理が必要
- AWS CDK (Cfn=Cloud Formation)
  - pros: stateの管理が不要
  - cons: drift発生時に手で検知・修正が必要 / 自由度が高すぎる

記述の自由度の影響

- AWS側の状態を正とした差分取り込みのしやすさ
  - 記述の自由度が低いほどテキストでの差分が出にくい
  - フォーマッターなどがあればマシだが…
    - たとえばTypeScriptのコードに落とされても差分が大変
- 純粋なJSONのような単なる構造体が一番良い

既存IaCソリューションのstate管理

- CfnやTerraformなどのIaCではstateを持ってそれとの差分を取る
  - 実態とstateは乖離し得る(この乖離をdriftと呼ぶ)
- 一般的な構成管理ではdriftはどの状態が正しいか都度判断が必要
- AWS側の状態を正としたIAMを表現するコード(設定)への差分取り込みをするためにはstateは邪魔になる

どうするか？

自作しました

実装

- 差分検出しAWS SDKでIAM APIを適切に叩く感じの実装
- JSONでIAMユーザーやIAMロールの設定内容を記述
  - インラインポリシーなどもそのまま記載できる
- TypeScriptで実装
  - AWS CDKを導入済みだったので開発言語を合わせたかった
  - io-tsでバリデーションすることで簡単なtypoなども検出可能

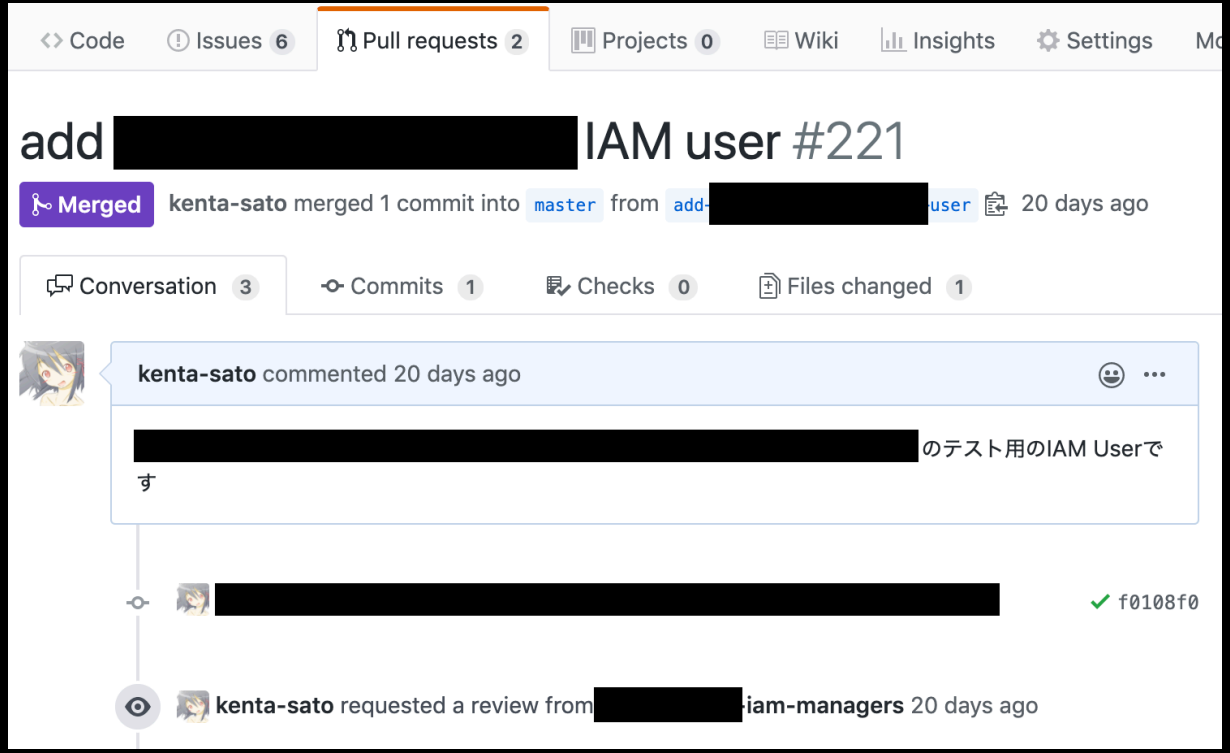
定義例

```
{
 "UserName": "foo",
 "Tags": {},
 "PermissionBoundary": "arn:aws:iam::012345678901:policy/OurPermissionsBoundary",
 "InlinePolicies": {
  "Bar": {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Effect": "Allow",
     "Action": [
      "s3:ListBucket"
     ],
     "Resource": "arn:aws:s3:ap-northeast-1:012345678901:foo-bucket"
    },
    {
     "Effect": "Allow",
     "Action": [
      "s3:GetObject",
```

定義例

```
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:ListBucket"
        ],
        "Resource": "arn:aws:s3:ap-northeast-1:012345678901:foo-bucket"
      },
      {
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:ap-northeast-1:012345678901:foo-bucket/*"
      }
    ]
  }
 },
 "AttachedPolicies": []
}
```

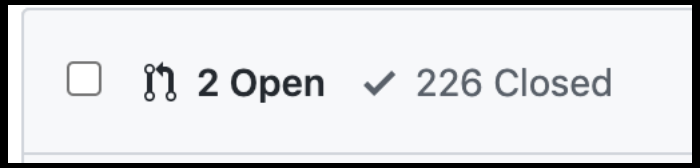
PR例


Code
Issues 6
Pull requests 2
Projects 0
Wiki
Insights
Settings
add [redacted] IAM user #221
Merged
kenta-sato merged 1 commit into master from add-[redacted]user 20 days ago
Conversation 3
Commits 1
Checks 0
Files changed 1
kenta-sato commented 20 days ago
[redacted]のテスト用のIAM Userです
[redacted]
f0108f0
kenta-sato requested a review from [redacted]iam-managers 20 days ago

運用したその後

喜びの声

- 「IAM周りを担当者に依頼する必要がなくなって楽になった」
- 「便利」
- がっつり使われています



起きたトラブル

トラブル

- IAM APIの障害で同期が止まる
- 存在しないActionを指定して同期エラー
- 存在しないユーザーをAssumeRolePolicyで信頼しようとして同期エラー
- Permissions Boundaryで許可していないものを追加してハマる

いずれも修正やリトライで解決済

めでたし

まとめ

まとめ

- IAM管理は難しい
  - ベストプラクティスはあるが実践には労力的な課題が多い
  - Permissions Boundaryをうまく使うと解決できる
- IAM管理ノウハウは組織に蓄積する仕組みを作ると吉
  - GithubでPull-RequestベースでIAMリソースを管理する道もある
  - リスクが受け入れられそうなら検討の価値アリ

TECH CON

DeNA TechCon 2021

ありがとうございました

質疑応答: Youtubeコメント欄にて