6000個のセキュリティポリシーを自動監査！アカツキ流、AWSセキュリティの取り組み紹介

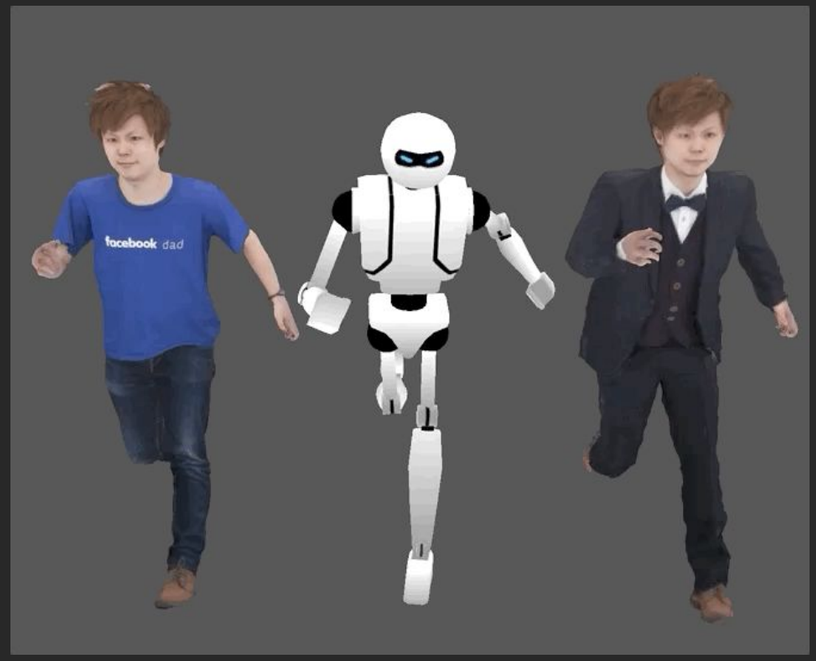
Speaker Name　駒井祐人
Job Title　サーバエンジニア ＆ テクニカル エバンジェリスト
Company Name　株式会社アカツキ

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

自己紹介

氏名 : 駒井祐人　　e__koma



役割 :

- サーバエンジニア
  - 直近5年でアカツキからリリースしている
    全ゲームのサーバサイドに関わっています
- クラウドセキュリティ管理者

去年のAWS Summit Tokyo 2019 にも登壇
(大規模ECS & Docker運用話)

会社紹介

Akatsuki

ビジョン : ”A Heart Driven World.” ハートドリブンな世界へ

他多数

昨今のセキュリティインシデントや脅威の事例

- S3バケット設定ミスによる1億2000万世帯以上の個人情報漏洩 (※ 1)

- 脆弱な部分があればあっという間に狙われる脅威
  - 公開S3バケットのファイルを、マルウェアを仕込んだ状態で上書く攻撃者 (※ 2)
  - GitHubのPublic RepositoryにAccess Keyをpushしたら13分で抜かれた話 (※ 3)
  - URL (/phpMyAdminなど) を推測し、アクセスしてくるbot

※1: https://www.trendmicro.com/vinfo/pl/security/news/virtualization-and-cloud/data-on-123-million-us-households-exposed-due-to-misconfigured-aws-s3-bucket

※2: https://www.skyhighnetworks.com/cloud-security-blog/skyhigh-discovers-ghostwriter-a-pervasive-aws-s3-man-in-the-middle-exposure/

※3: https://qiita.com/saitotak/items/813ac6c2057ac64d5fef

クラウドセキュリティの課題感

可視化と統制

脅威に対する検出と対応

クラウドセキュリティの課題感

今日は主にここの話

可視化と統制

脅威に対する検出と対応

今日お届けすること

40 個以上のAWSアカウントの全リージョンに対して
6000 個以上のセキュリティポリシーの監査を自動化し
自動監査設定を継続的にデプロイしつづけられる仕組み

これらを実現してきたストーリーを中心にお話します。

全体像

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

アカツキのAWS規模 (2020/6時点)

40+

accounts

130+

custom VPC

1000+

EC2 instances

クラウドセキュリティ取り組みの全体像

ロギング
- AWS CloudTrail

監査
- AWS Config
- AWS Trusted Advisor
- Prowler (toniblyx/prowler)

脅威検出
- Amazon GuardDuty

パッチ適用
- AWS Systems Manager Patch Manager

EDR
- CrowdStrike Falcon

ネットワーク監視
- Verizon NDR

可視化と分析
- SumoLogic

アプリケーション脆弱性診断
- FutureVuls

※ 選定していないAWSサービスの話も後ほど出てきます

クラウドセキュリティ取り組みの全体像

ロギング
- AWS CloudTrail

監査
- AWS Config
- AWS Trusted Advisor
- Prowler (toniblyx/prowler)

脅威検出
- Amazon GuardDuty

パッチ適用
- AWS Systems Manager Patch Manager

EDR
- CrowdStrike Falcon

ネットワーク監視
- Verizon NDR

可視化と分析
- SumoLogic

アプリケーション脆弱性診断
- FutureVuls

※ 選定していないAWSサービスの話も後ほど出てきます

AWS Config

AWSの各種設定がルールどおりに設定されているか評価するサービス

例：

- 公開されているセキュリティグループが存在しないか？
- S3バケットが公開設定になっていないか？
- 公開されているEBS / RDSスナップショットが存在しないか？

AWS Config マネージドルール例

マネージドルールは追加され続けている

AWS Config マネージドルール例

有効にするだけで、アカウント内の監査ができる

	ルール名	コンプライアンス
○	incoming-ssh-disabled	1 準拠していないリソース
○	iam-password-policy	準拠
○	iam-root-access-key-check	準拠
○	lambda-function-public-access-prohibited	準拠
○	root-account-mfa-enabled	準拠
○	cmk-backing-key-rotation-enabled	準拠
○	cloud-trail-encryption-enabled	準拠
○	guardduty-enabled-centralized	準拠

AWS Configを使った
マルチアカウント自動監査

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Config 導入前の監査

セキュリティポリシーの項目に対してチェックリストを作り、
これらを監査する運用だった。

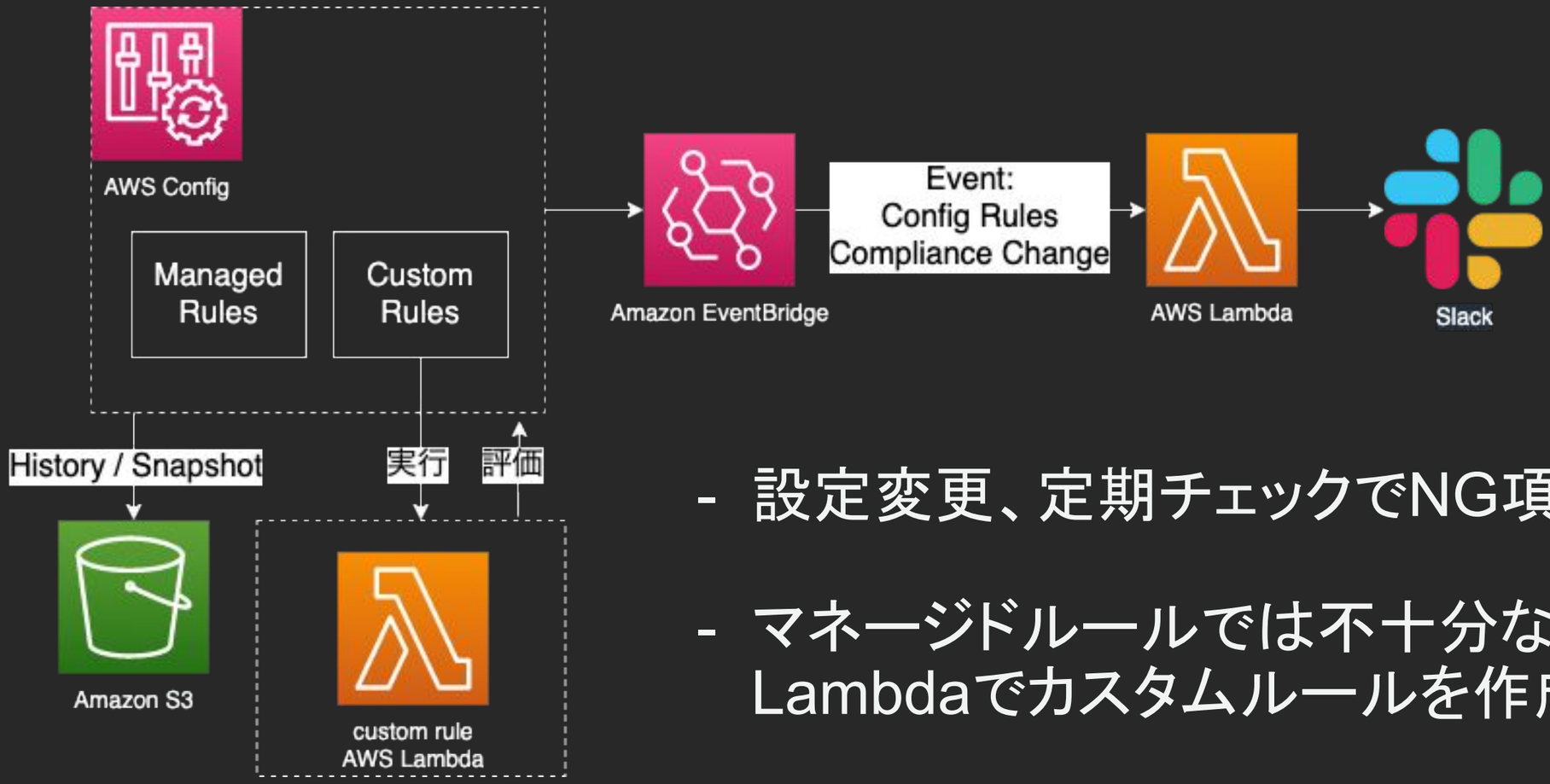
定期監査の課題

- 工数
- チェックの確からしさ
- 抜け漏れ

AWS Config を使った監視の流れ (シングルアカウント)

- 設定変更、定期チェックでNG項目検知

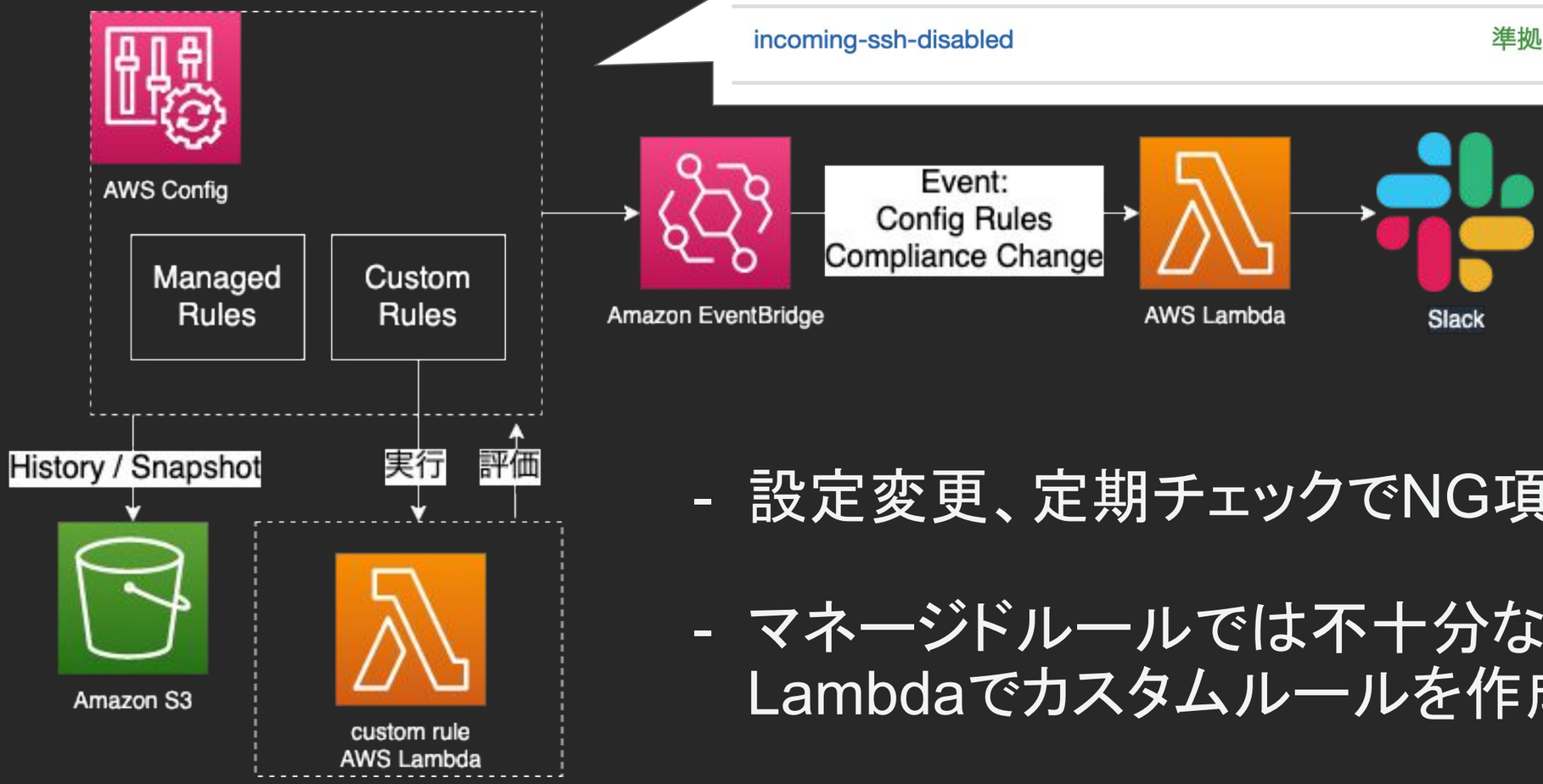
AWS Config を使った監視の流れ (シングルアカウント)



- 設定変更、定期チェックでNG項目検知
- マネージドルールでは不十分な項目には Lambdaでカスタムルールを作成

AWS Config を使った監視の流れ (シングルアカウント)

ルール名	コンプライアンス
iam-ip-restriction	1 非準拠リソース
incoming-ssh-disabled	準拠



- 設定変更、定期チェックでNG項目検知
- マネージドルールでは不十分な項目には Lambdaでカスタムルールを作成

実装したカスタムルール例

- IAMユーザのAccess KeyにIP制限がかかっているか
  - awslabsにPull Requestを送ってマージされたので誰でも利用可能 (awslabs/aws-config-rules/python/IAM_IP_RESTRICTION)

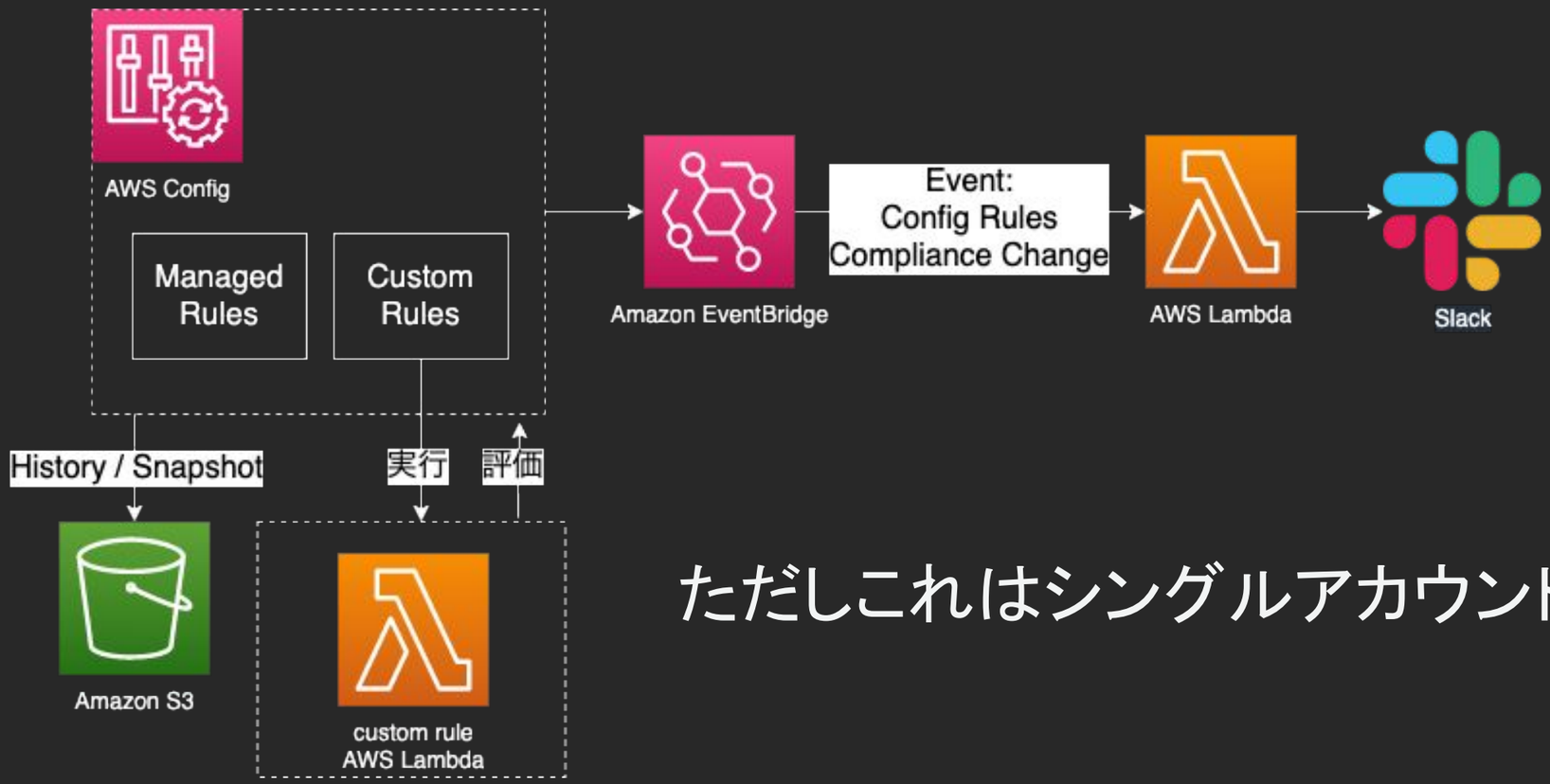
- セキュリティグループに公開IPが存在しないか
  - マネージドルールに vpc-sg-open-only-to-authorized-ports が存在するが、その改良版
  - 例外設定がしづらいので例外設定を追加

- AWSアカウントが利用されているか
  - IAM Roleの最終利用日をチェック

など

AWS Config を導入した結果

セキュリティポリシー監査を自動化できた



ただしこれはシングルアカウントでの話

AWS Config マルチアカウント運用時のポイント

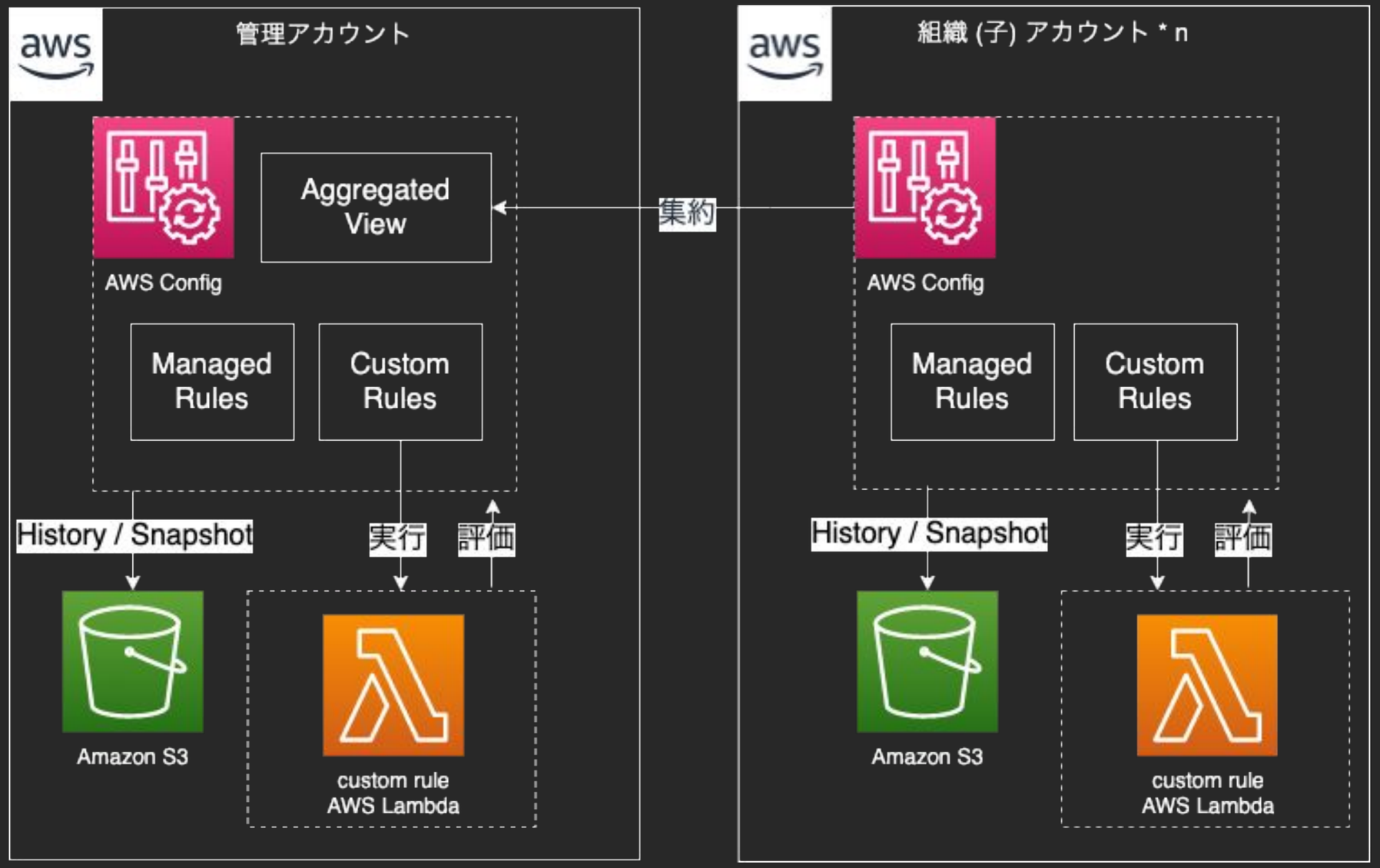
- Aggregator View
  - アグリゲータアカウントに組織全体のAWS Configの結果を集約できる

- Lambdaの一元管理
  - カスタムルールのコードが各アカウントで重複する
  - コードを1アカウントのみで管理して、クロスアカウント実行することで一元管理する

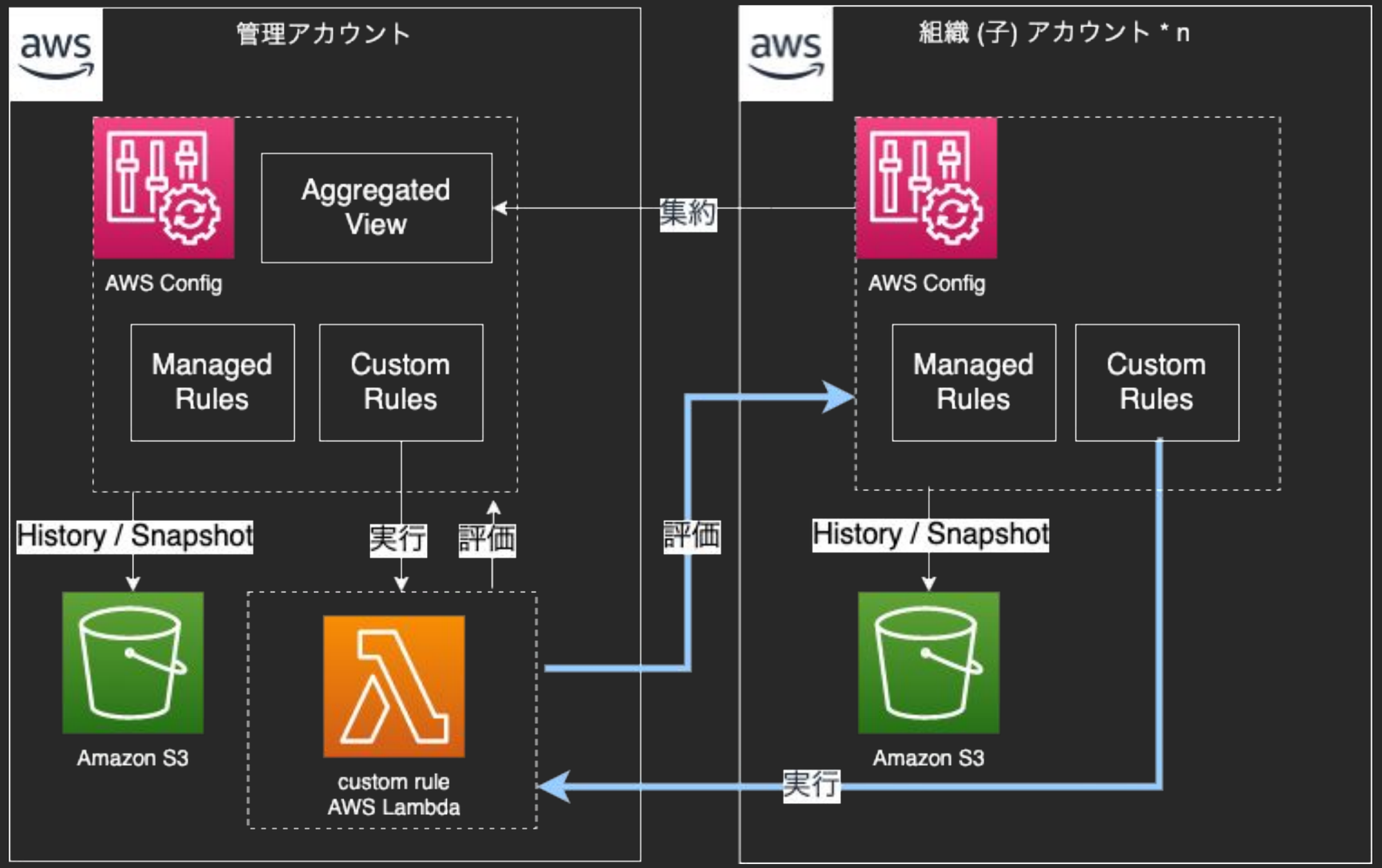
Aggregator View

マルチアカウントマルチリージョンで非準拠ルールを一覧可能

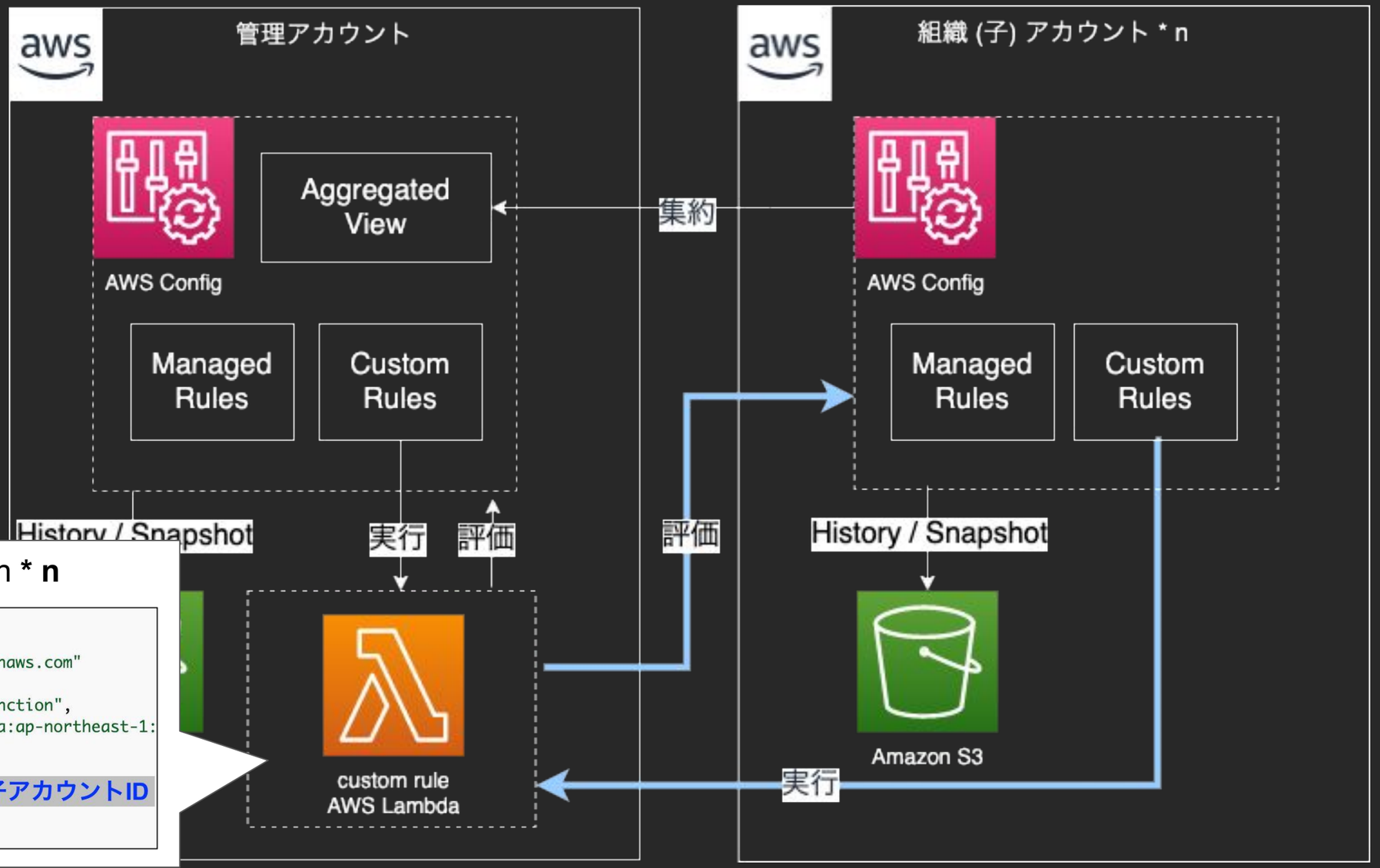
Lambdaカスタムルールの一元管理



Lambdaカスタムルールの一元管理



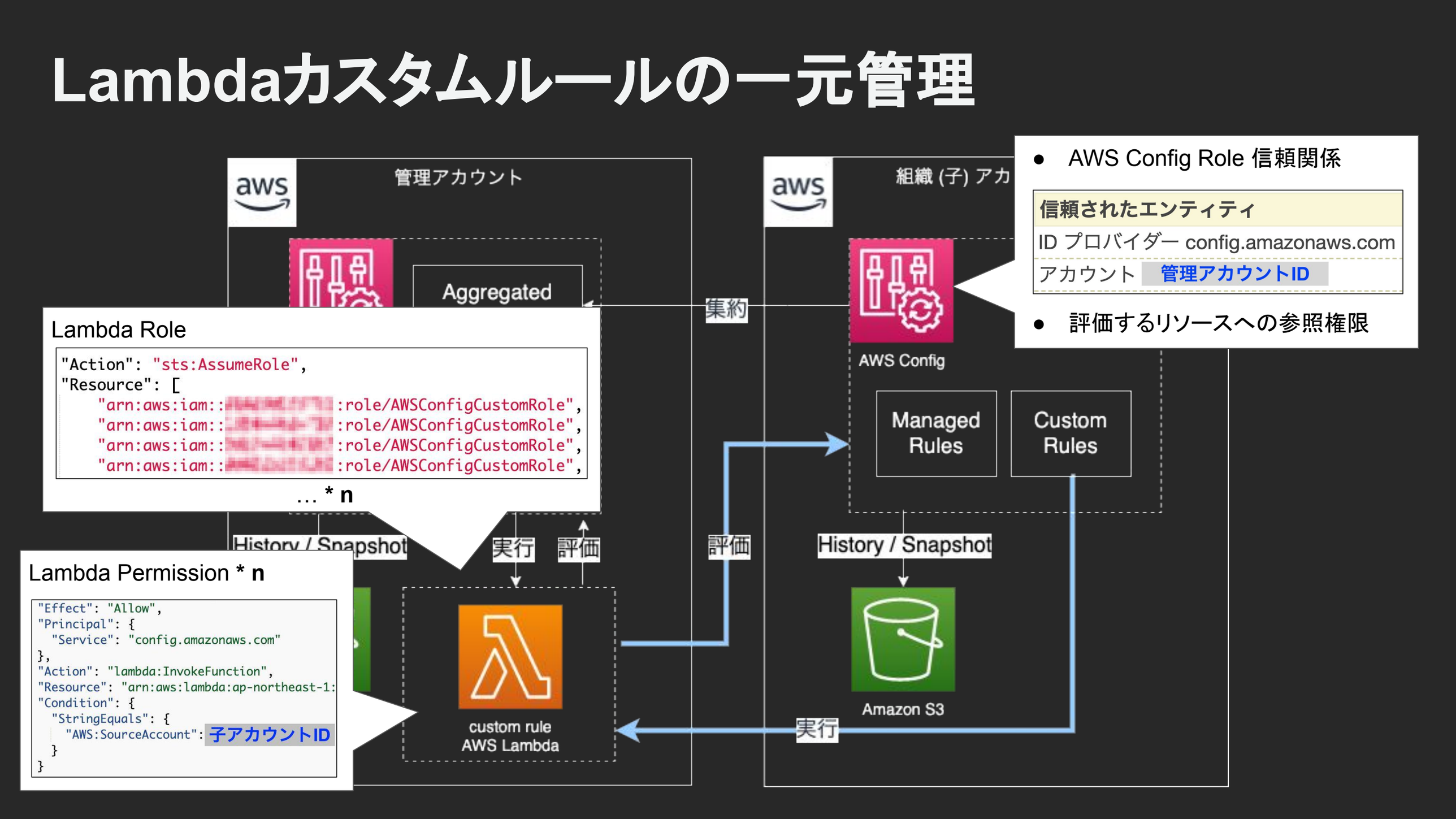
Lambdaカスタムルールの一元管理



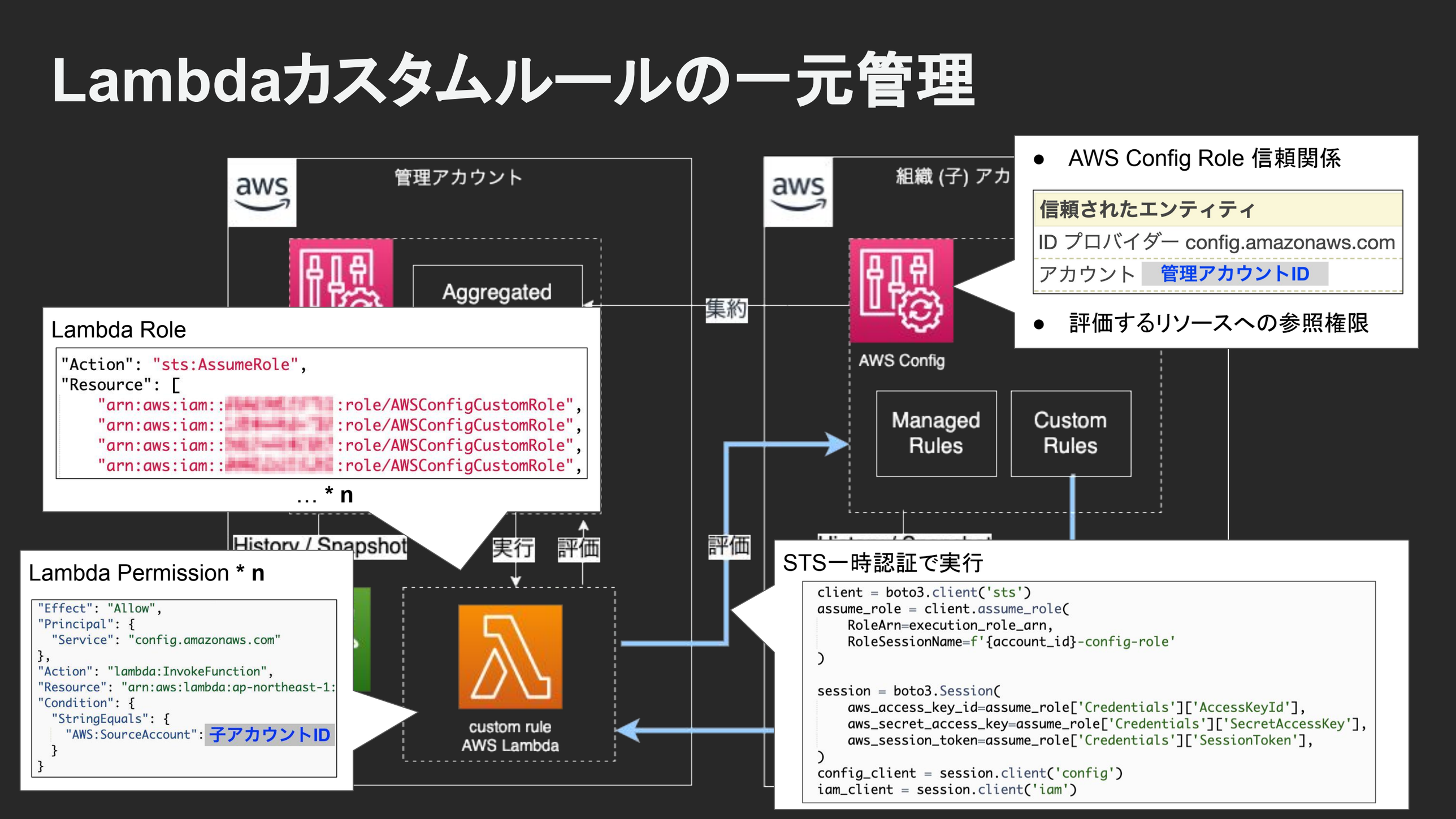
Lambda Permission * **n**

```
"Effect": "Allow",
"Principal": {
  "Service": "config.amazonaws.com"
},
"Action": "lambda:InvokeFunction",
"Resource": "arn:aws:lambda:ap-northeast-1:
"Condition": {
  "StringEquals": {
    "AWS:SourceAccount": 子アカウントID
  }
}
```

Lambdaカスタムルールの一元管理



Lambdaカスタムルールの一元管理



Lambdaカスタムルールの一元管理

- AWS Config Role 信頼関係

信頼されたエンティティ
IDプロバイダー config.amazonaws.com
ト 管理アカウントID

するリソースへの参照権限

Lambda Role

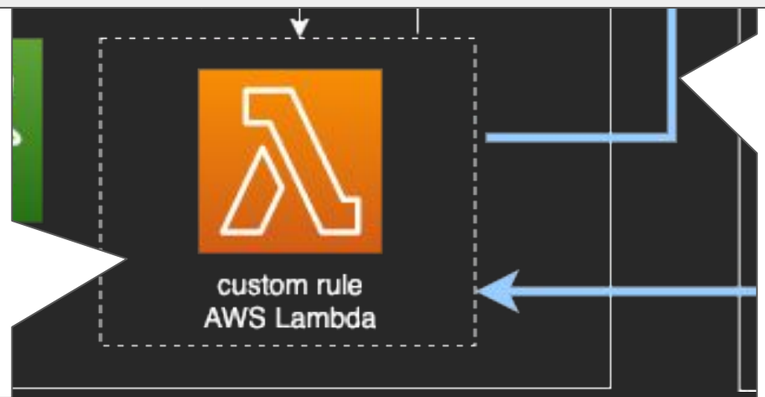
```
"Action": "sts:AssumeRo
"Resource": [
    "arn:aws:iam::
    "arn:aws:iam::
    "arn:aws:iam::
    "arn:aws:iam::
```

× 全リージョン

(AWS Configから別リージョンのLambdaを実行することはできないので、
リージョン間はLambdaが重複してしまう)

Lambda Permission * n

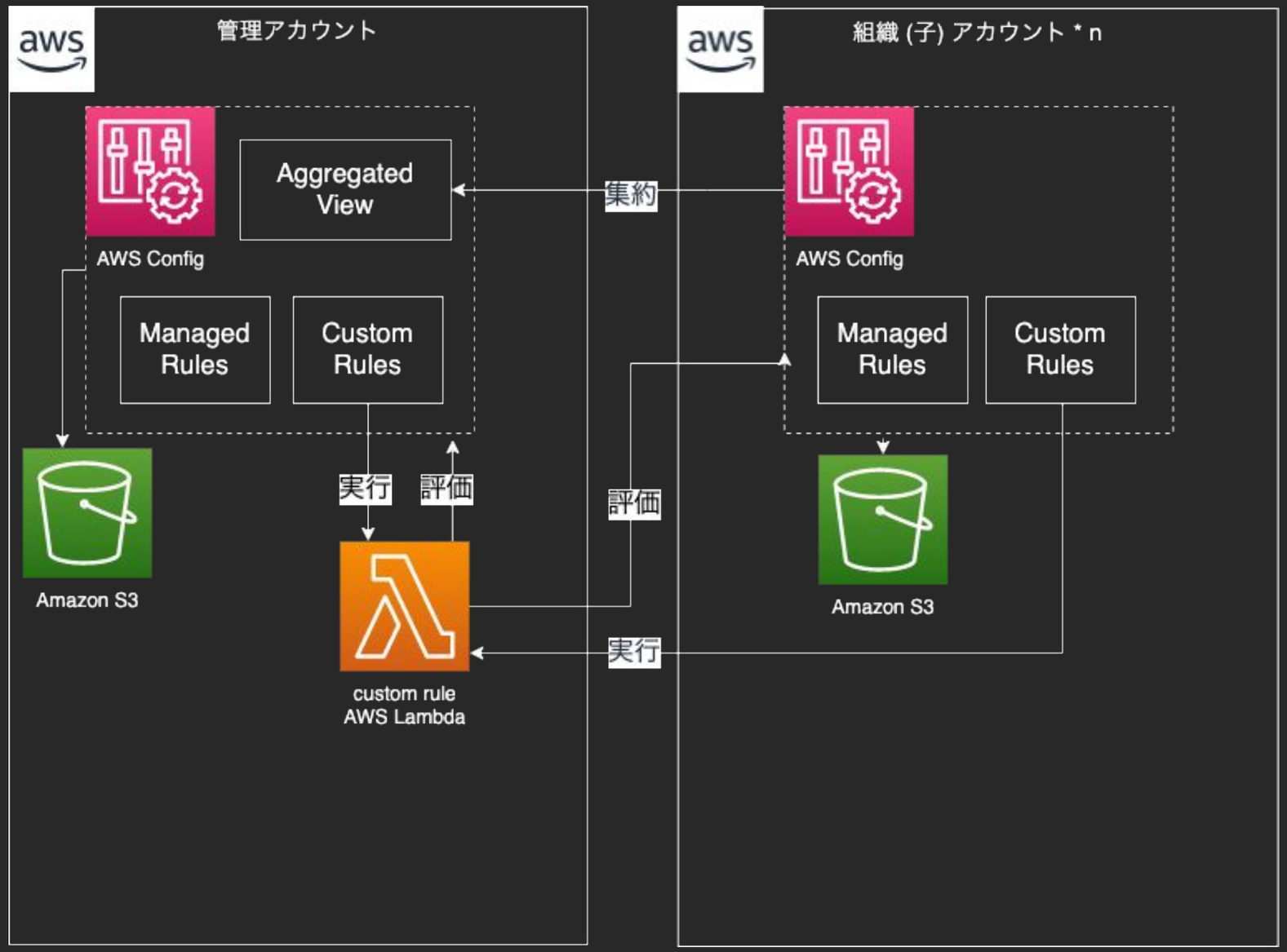
```
"Effect": "Allow",
"Principal": {
  "Service": "config.amazonaws.com"
},
"Action": "lambda:InvokeFunction",
"Resource": "arn:aws:lambda:ap-northeast-1:
"Condition": {
  "StringEquals": {
    "AWS:SourceAccount": 子アカウントID
  }
}
```



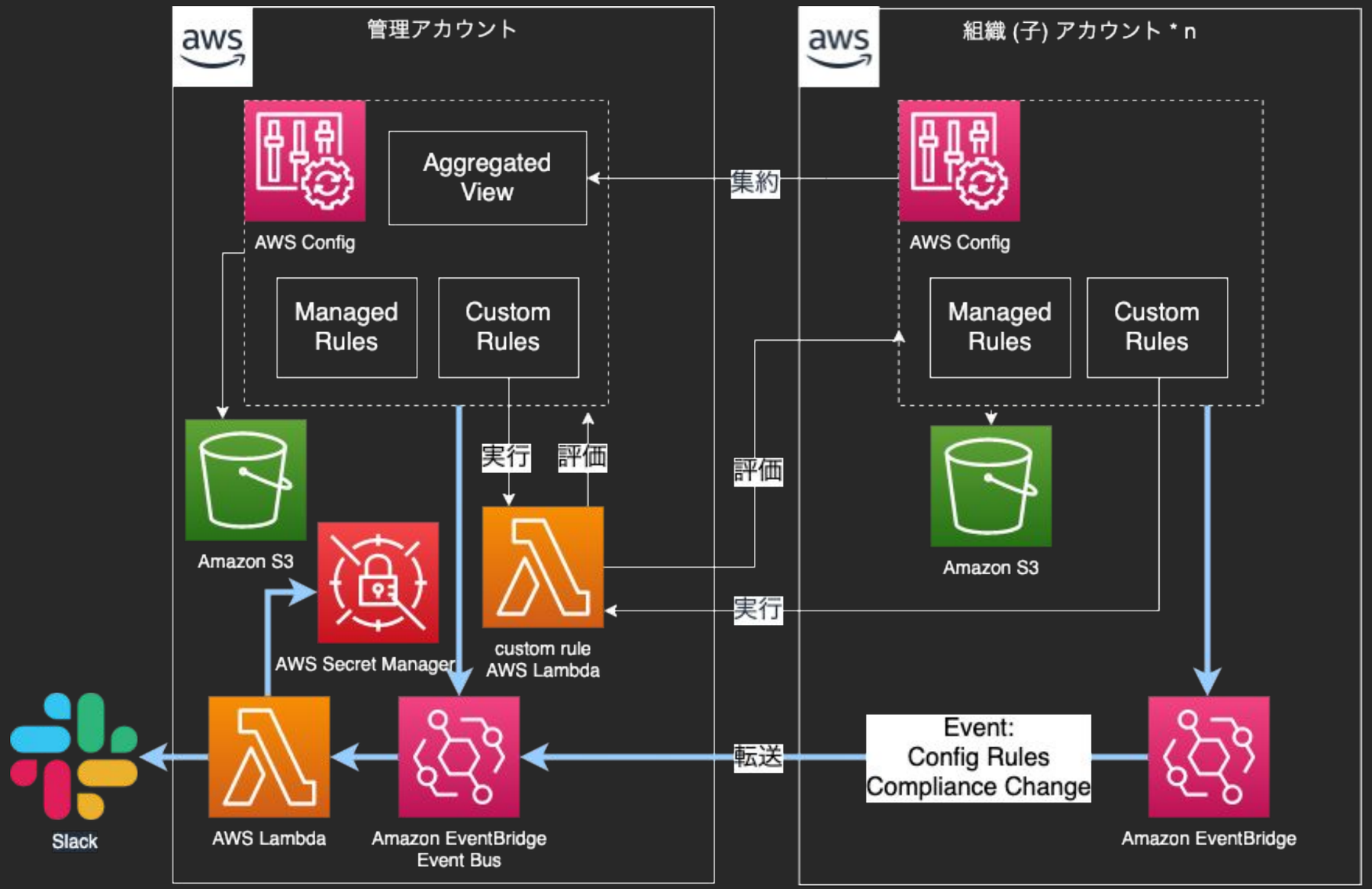

```
client = boto3.client('sts')
assume_role = client.assume_role(
    RoleArn=execution_role_arn,
    RoleSessionName=f'{account_id}-config-role'
)

session = boto3.Session(
    aws_access_key_id=assume_role['Credentials']['AccessKeyId'],
    aws_secret_access_key=assume_role['Credentials']['SecretAccessKey'],
    aws_session_token=assume_role['Credentials']['SessionToken'],
)
config_client = session.client('config')
iam_client = session.client('iam')
```

変更イベント、通知先の一元管理

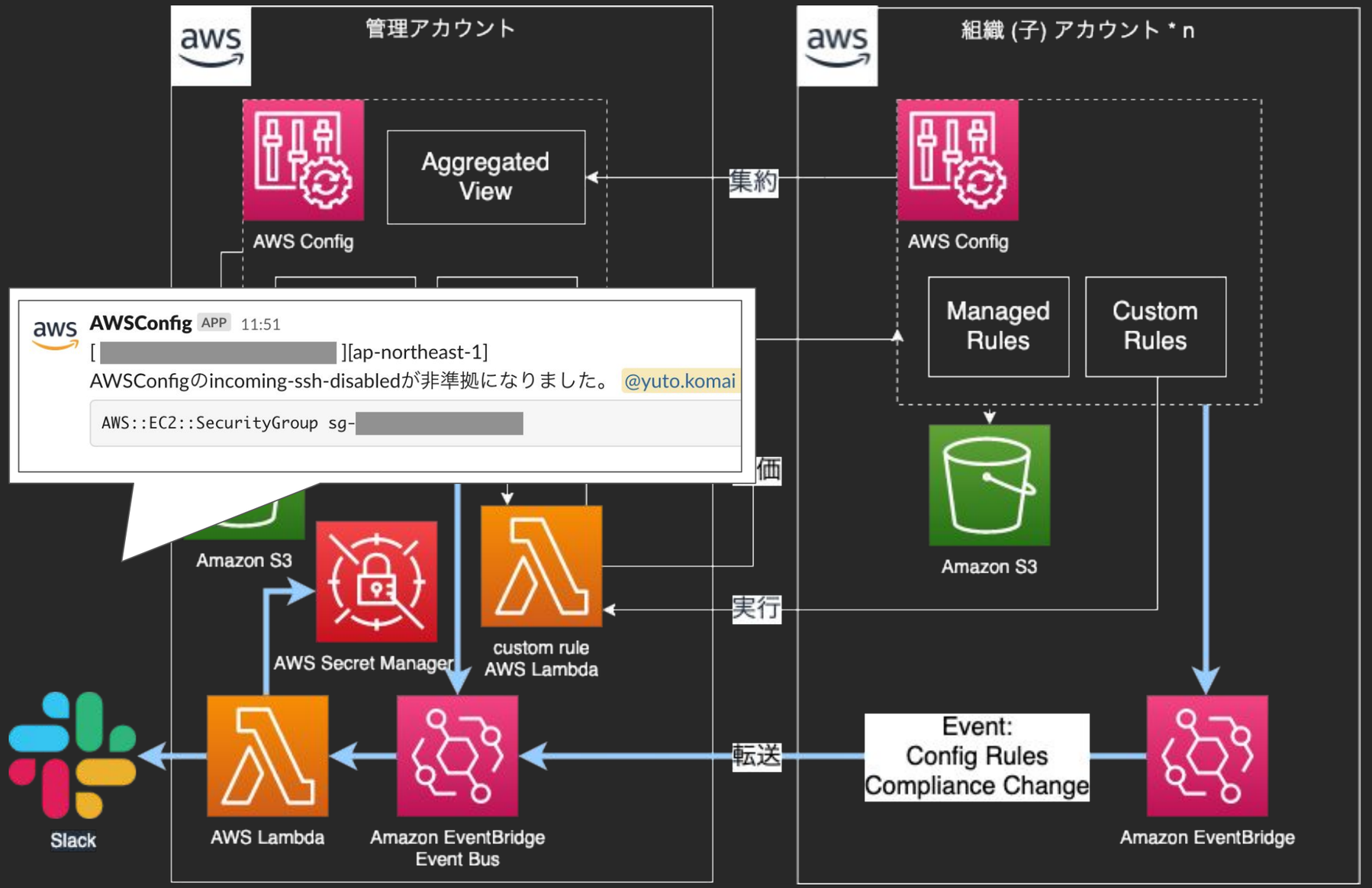


変更イベント、通知先の一元管理



- 変更イベント(Compliance Change)を1アカウントに集約
- 各AWSアカウント管理者のSlack member IDをSecret Managerに登録し、AWS Config NGの場合はアカウント管理者に通知

変更イベント、通知先の一元管理



- 変更イベント(Compliance Change)を1アカウントに集約
- 各AWSアカウント管理者のSlack member IDをSecret Managerに登録し、AWS Config NGの場合はアカウント管理者に通知

AWS Configで監視できているルール数

6000+

rules

※ 全アカウント全リージョン合計。
グローバルリソースのチェックは1リージョンのみ、リージョナルリソースは全リージョンにデプロイ

AWS Configで実現できたこと

- 40 以上のAWSアカウントの全リージョンに対して、
  6000 以上のセキュリティポリシーの監査を自動化

- カスタムルール作成により、マネージドルールに存在しない項目も監査。結果的にawslabsにも貢献。

- カスタムルールや変更イベントの一元管理により、管理負担を軽減

一見、これで
うまくいきそう

ですが、
AWS Configを
マルチアカウントで運用するには
まだまだ課題がありました
(次章で説明します)

AWSマルチアカウントへの自動デプロイ

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

課題

AWS Configルールを含むセキュリティ統制の運用管理が膨大

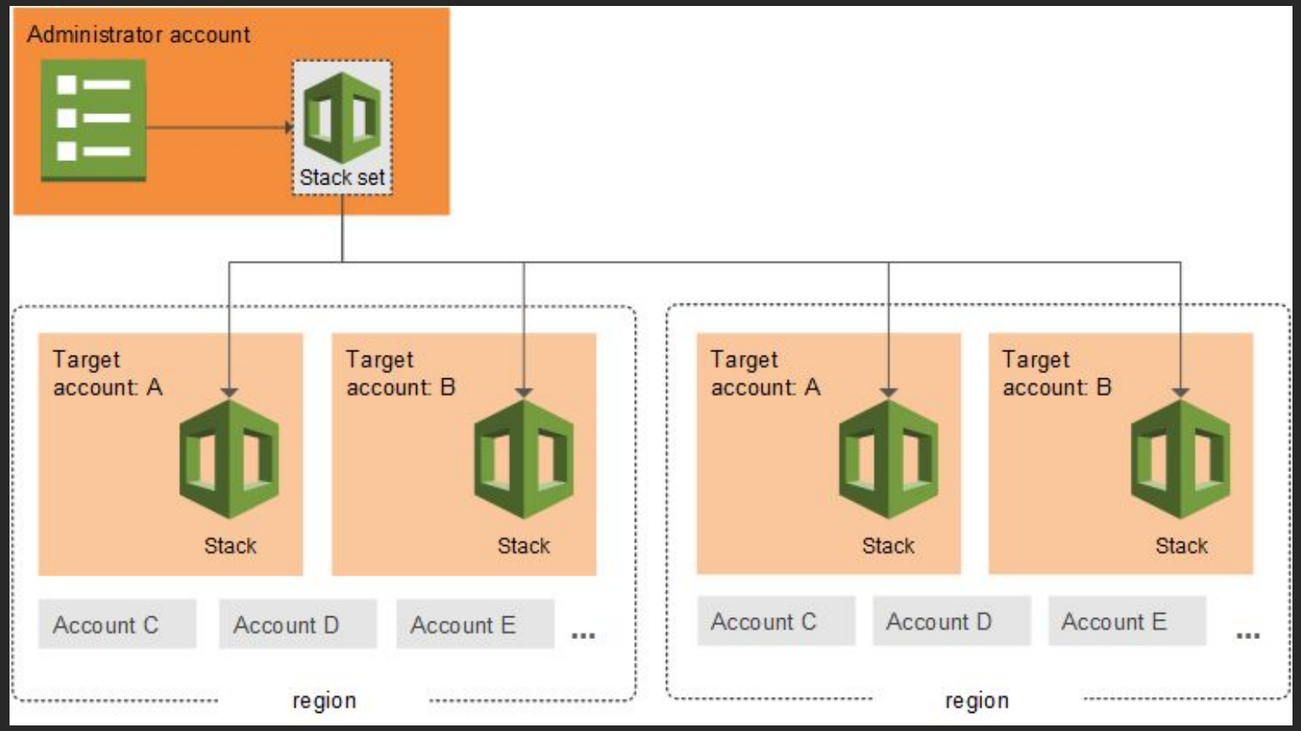
- CloudFormationでAWS Configを管理していたが、
  当時は各アカウントに個別で適用していた。

- AWS Configルールの追加オペレーション1回で、
  40アカウント * 15リージョン = 600 stackの修正が必要だった。

CloudFormation StackSetsに移行

AWS CloudFormation StackSets とは

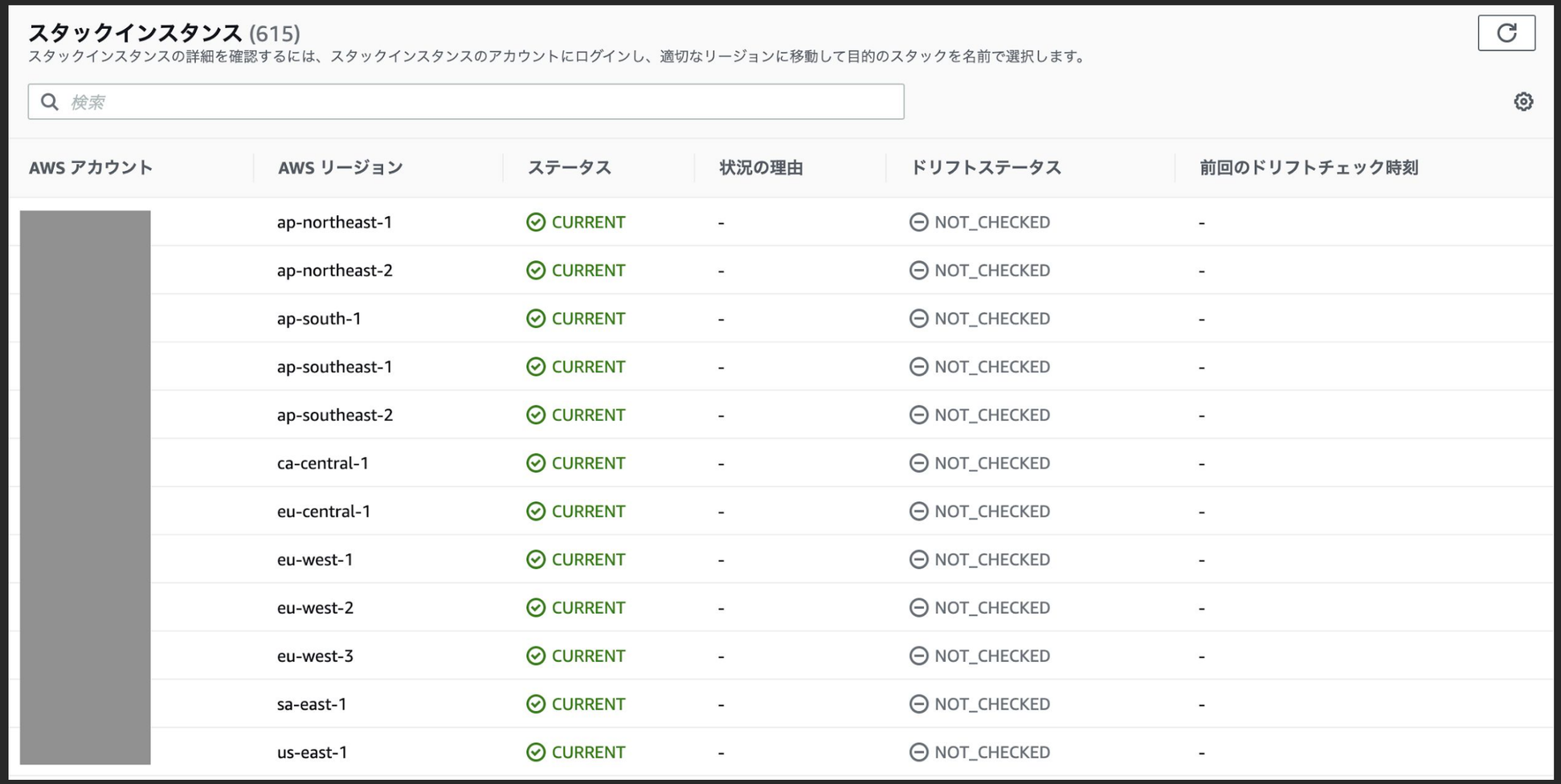
- マルチアカウント、マルチリージョンのStackを一括更新できる



※ 画像は公式ドキュメントから抜粋
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

CloudFormation StackSetsに移行

AWS CloudFormation StackSets とは



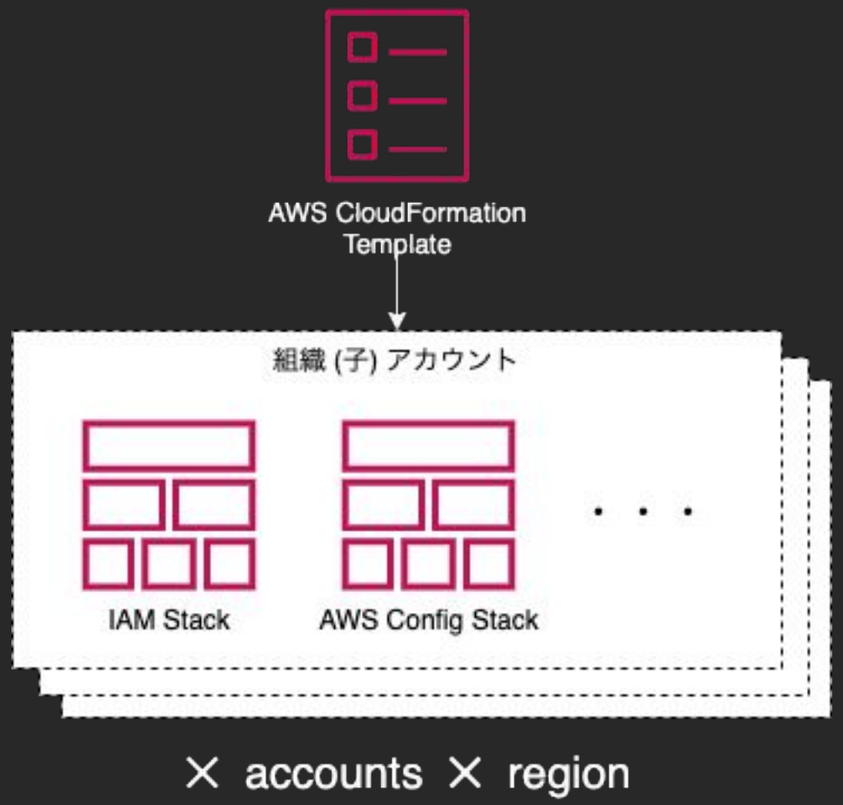
スタックインスタンス (615)

スタックインスタンスの詳細を確認するには、スタックインスタンスのアカウントにログインし、適切なリージョンに移動して目的のスタックを名前で選択します。

検索

AWS アカウント	AWS リージョン	ステータス	状況の理由	ドリフトステータス	前回のドリフトチェック時刻
	ap-northeast-1	CURRENT	-	NOT_CHECKED	-
	ap-northeast-2	CURRENT	-	NOT_CHECKED	-
	ap-south-1	CURRENT	-	NOT_CHECKED	-
	ap-southeast-1	CURRENT	-	NOT_CHECKED	-
	ap-southeast-2	CURRENT	-	NOT_CHECKED	-
	ca-central-1	CURRENT	-	NOT_CHECKED	-
	eu-central-1	CURRENT	-	NOT_CHECKED	-
	eu-west-1	CURRENT	-	NOT_CHECKED	-
	eu-west-2	CURRENT	-	NOT_CHECKED	-
	eu-west-3	CURRENT	-	NOT_CHECKED	-
	sa-east-1	CURRENT	-	NOT_CHECKED	-
	us-east-1	CURRENT	-	NOT_CHECKED	-

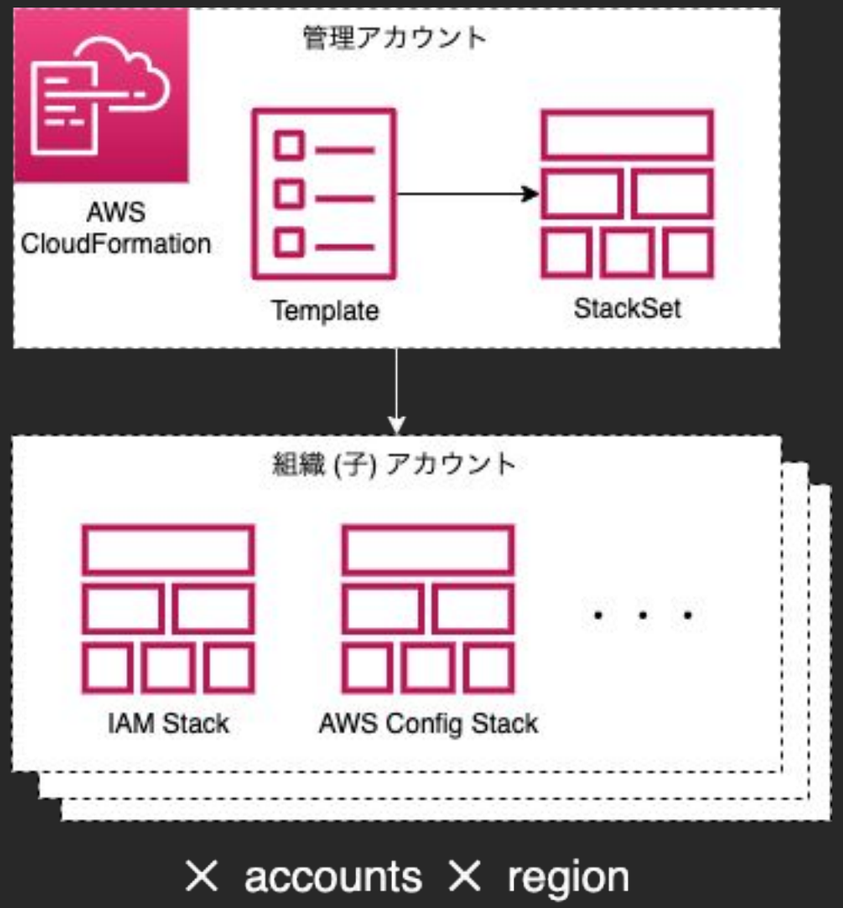
CloudFormation StackSetsに移行



StackSets 移行前の状態

- 全アカウント全リージョンに対して CloudFormationを個別実行していた (600 stack 以上)

CloudFormation StackSetsに移行



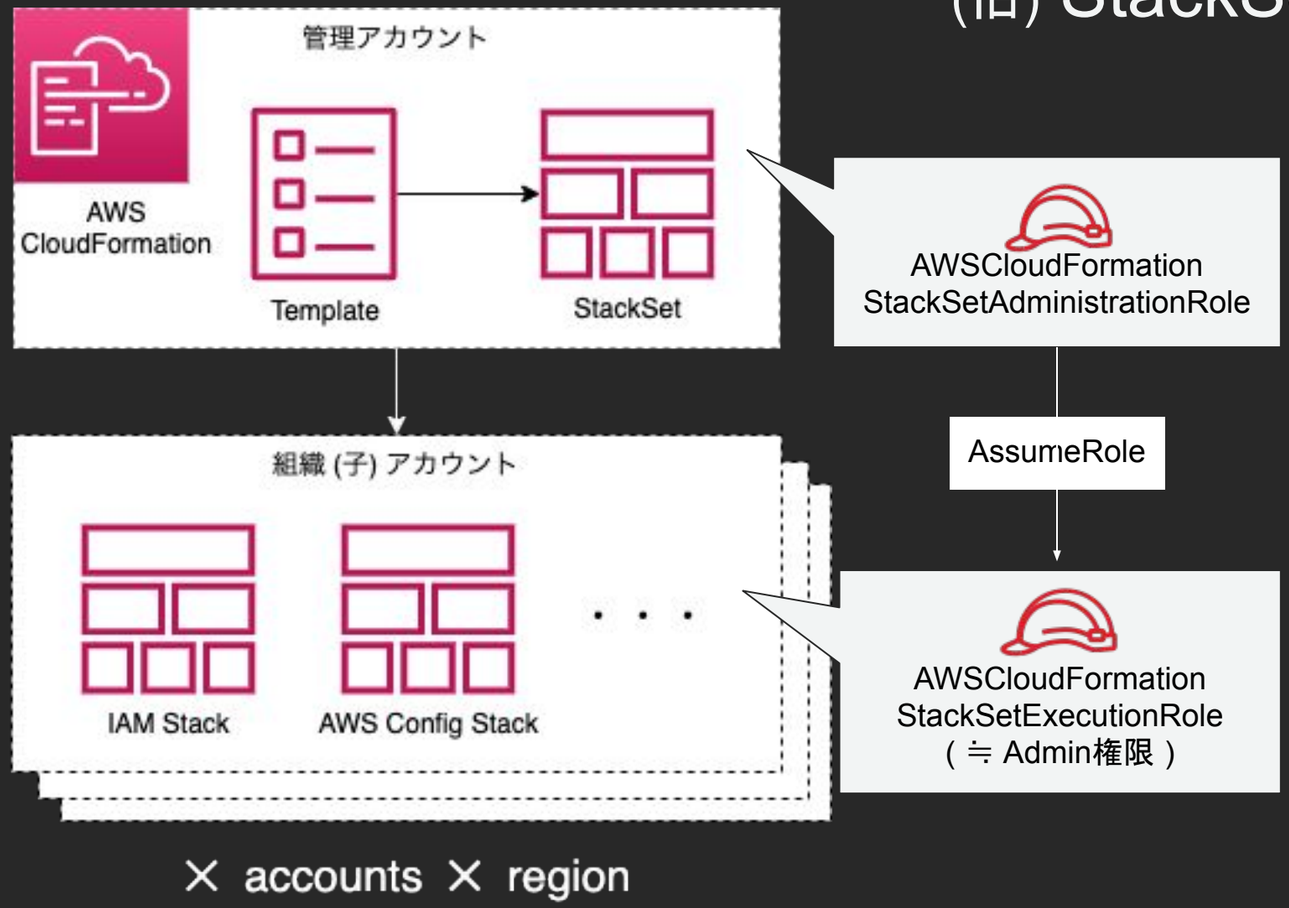
StackSets に移行

- Stack作り直し

  (S3バケットは中身が入ってると壊せないので
  頑張って移行）

CloudFormation StackSetsに移行

(旧) StackSets の仕組み



(旧) CloudFormation StackSetsの課題

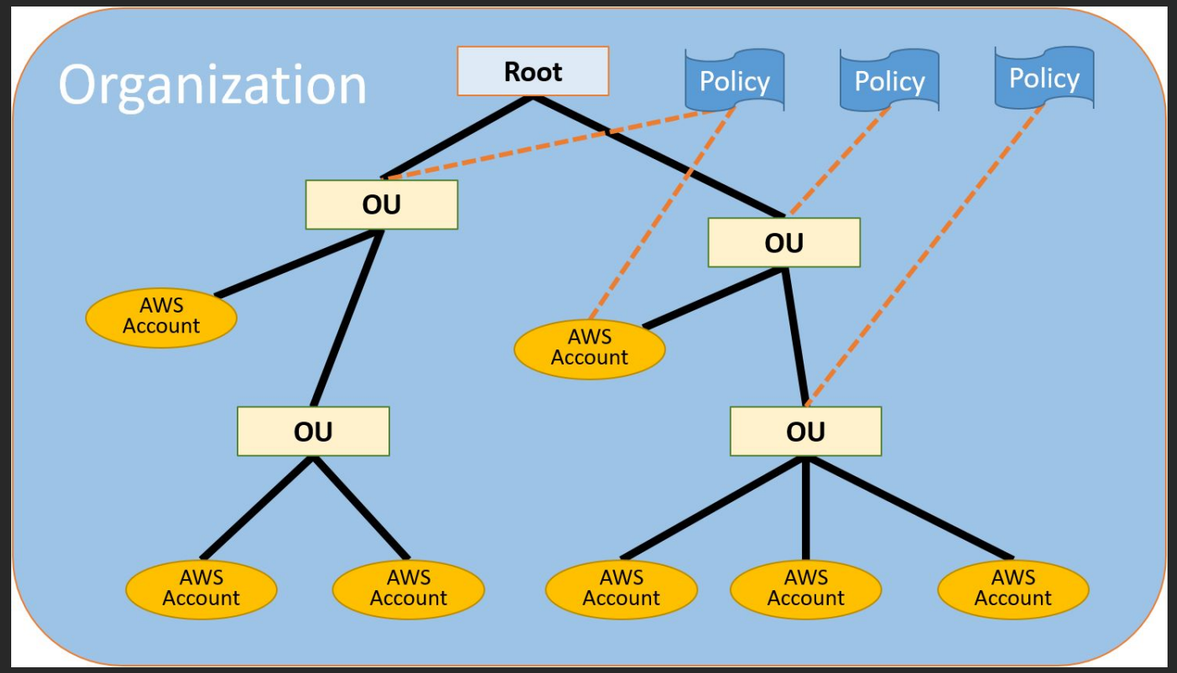
- 権限が強すぎるIAM Role
  - 全アカウントのAdmin権限を持っているようなもの
  - このRoleがあればStackSetsを使わずとも、AWS CLIで簡単に同じことを実現できてしまう…

- 自動化が中途半端
  - 事前に組織(子)アカウントにRoleを作っておく必要がある

StackSets + Organizations連携

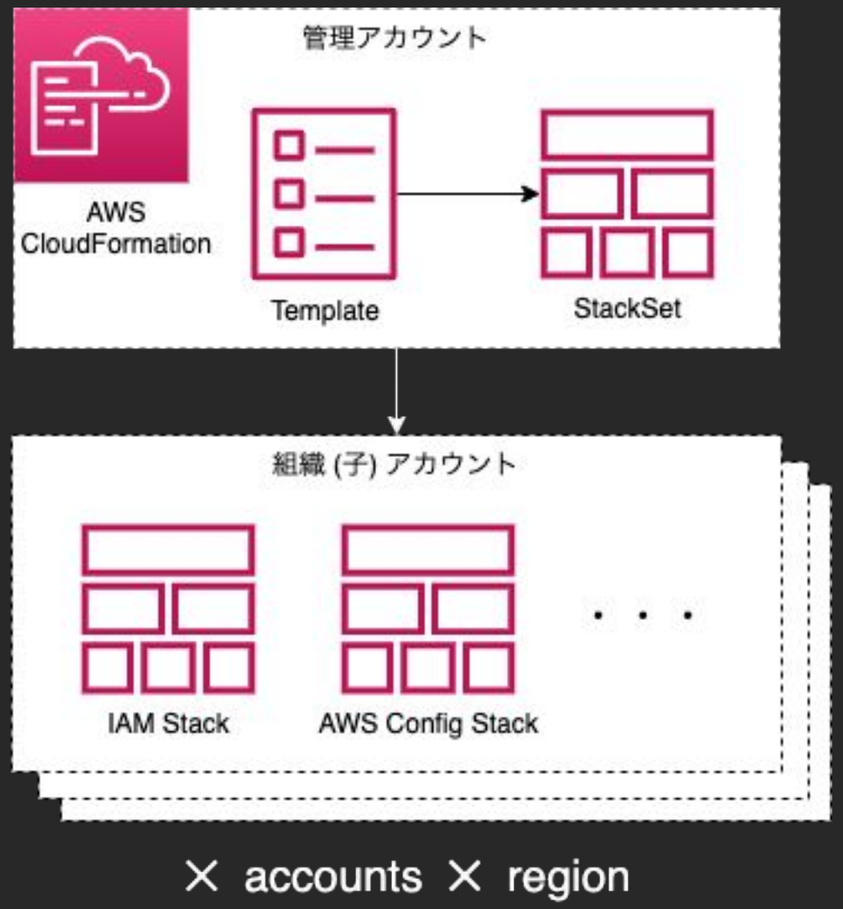
2020/02、AWS CloudFormation StackSetsとAWS Organizationsが連携
旧StackSetsの課題がまとめて解消。現在はOrganizations連携がオススメ

- 強力なIAM Roleが不要
- 組織単位のデプロイ
- 自動デプロイ機能



※ 画像は公式ドキュメントから抜粋
https://docs.aws.amazon.com/ja_jp/organizations/latest/userguide/orgs_getting-started_concepts.html

StackSets + Organizations連携

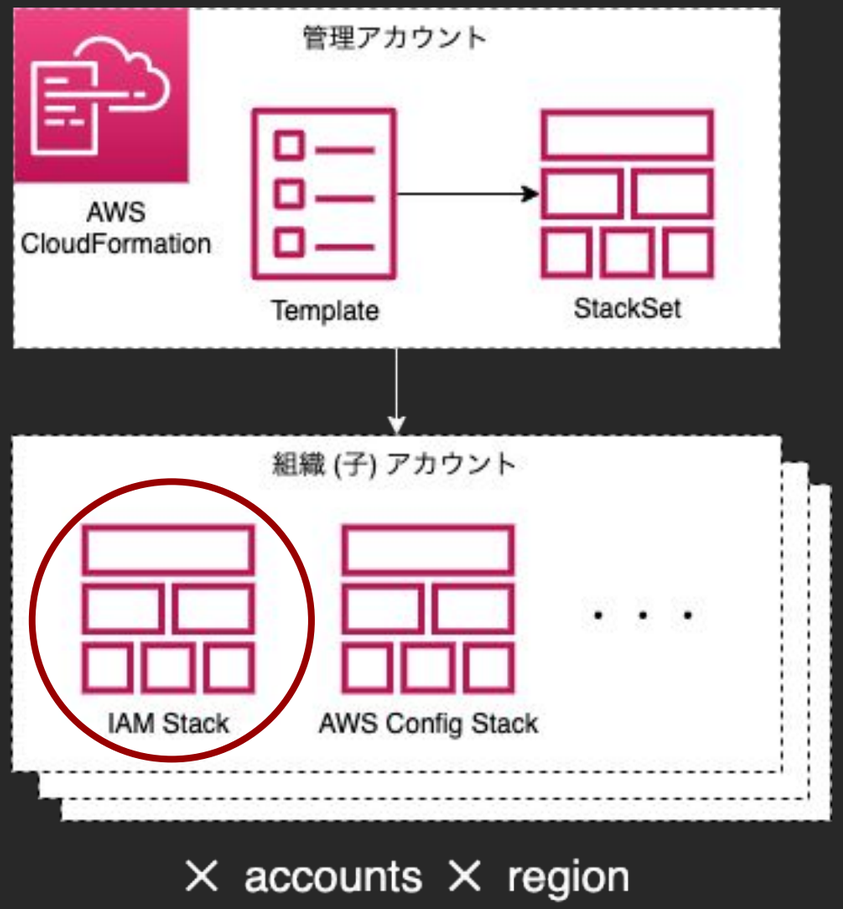


Organizations連携に移行

- 初期設定で有効にする必要があるため、またしても全て作り直し

- これで新規AWSアカウント発行時に、セキュリティ設定のCloudFormationが自動で適用される？

StackSets + Organizations連携



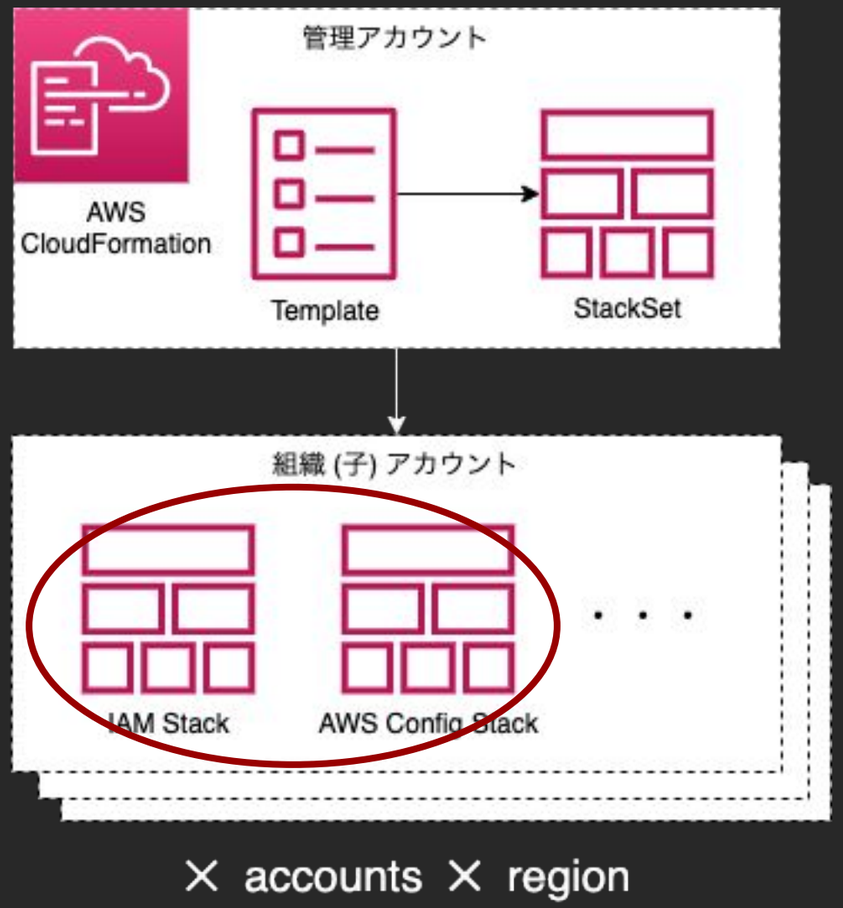
課題

自動デプロイに失敗する

- IAMを個別Stackで作成しているため
  - IAM Stack は東京リージョンのみ作成
  - AWS Config Stackは全リージョン作成

- 自動デプロイではStack間の順序性や依存関係を持たせられないため、AWS Config Stack作成時にIAMが作成されている保証がない

StackSets + Organizations連携



対処

- IAM用Stackを他Stackと統合
  - IAM + AWS Config
  - IAM + Lambda など

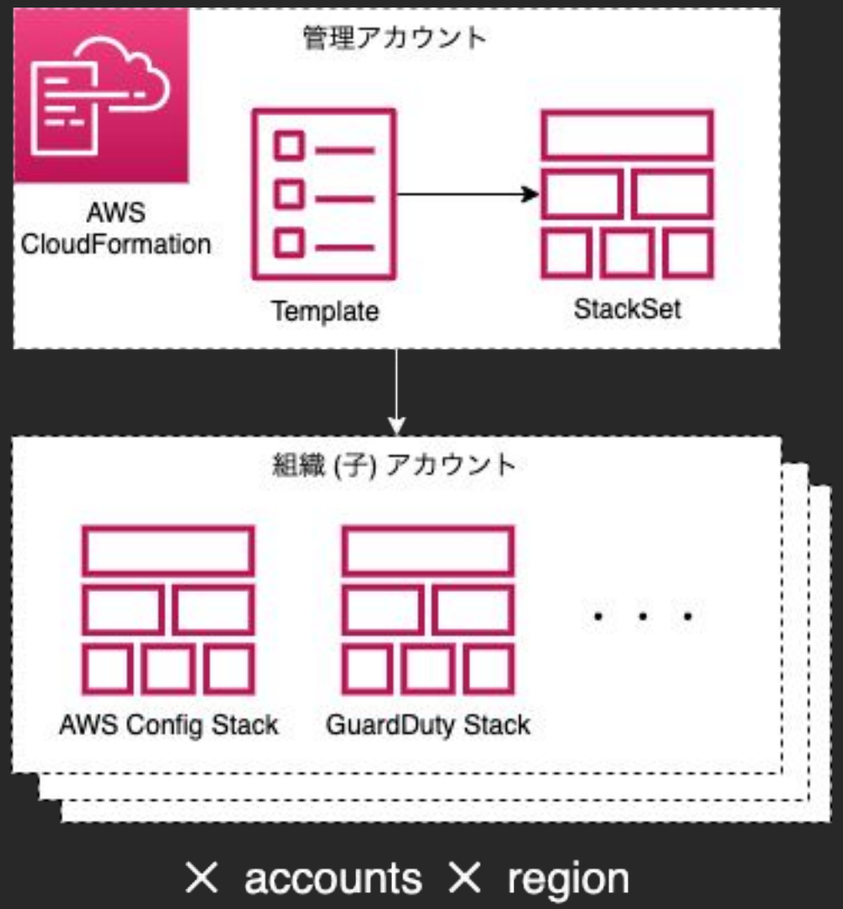
- 一方、Stack統合すると、各リージョンでIAMが重複してしまう問題も発生

StackSets + Organizations連携

StackSetsのリージョンごとの実行順序性を利用。
例 : バージニア北部 (初回実行リージョン) のみIAM作成

```yaml
Conditions:
  VirginiaRegion: !Equals [ !Ref AWS::Region, us-east-1 ]
Resources:
  AWSConfigRole:
    Type: AWS::IAM::Role
    Condition: VirginiaRegion
    Properties:
      RoleName: AWSConfigCustomRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
```

StackSets + Organizations連携



- 自動デプロイが失敗する問題も解決
- セキュリティ統制で設定したい全ての設定をStackSetsに移行できた

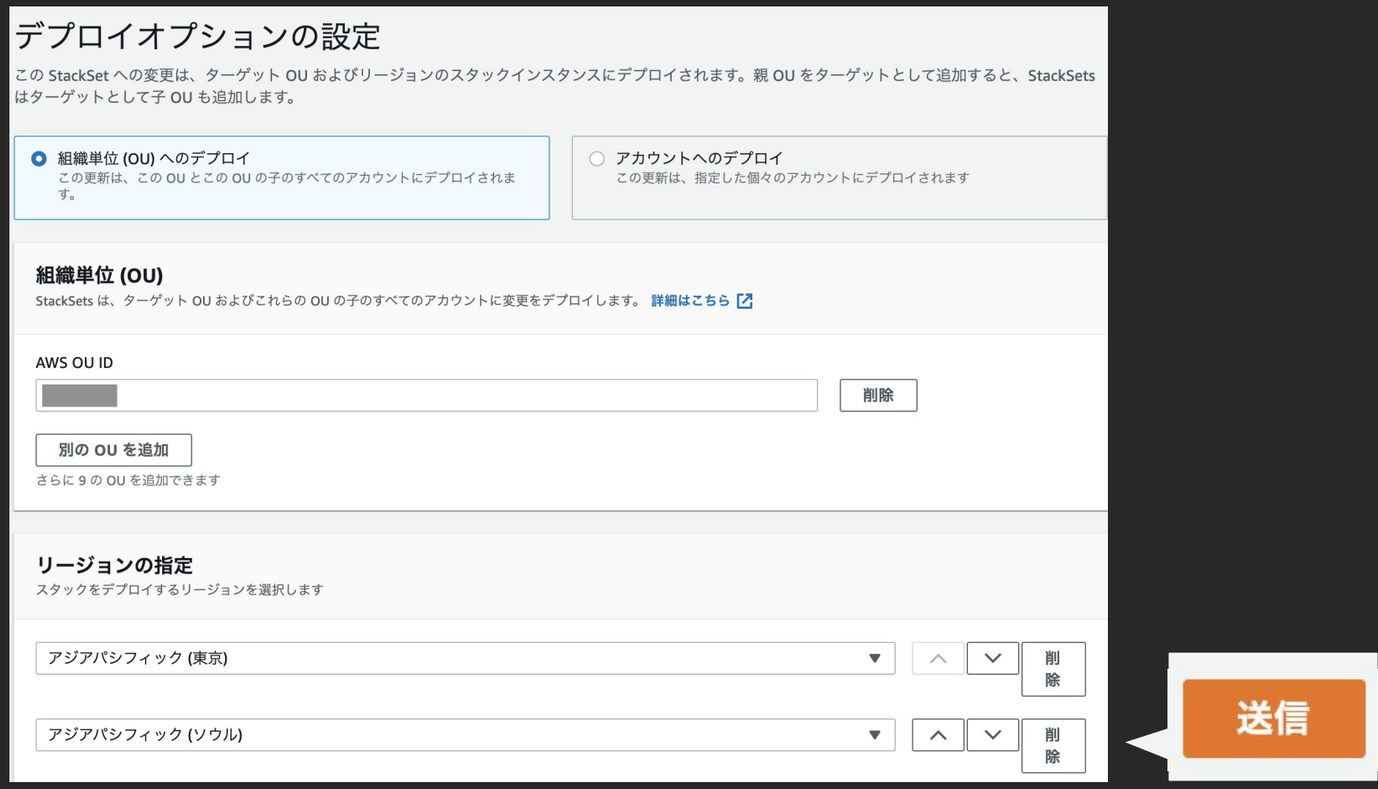
どれくらい簡単か

Templateを用意してポチッとするだけで、全アカウント全リージョン(※)に対して設定が追加 / 更新できる (※ 正確にはStackSets対応リージョン)

Template

```yaml
AWSConfigRulesROOTAccountMfaEnabled:
  Type: AWS::Config::ConfigRule
  Condition: TokyoRegion
  Properties:
    ConfigRuleName: root-account-mfa-enabled
    Source:
      Owner: AWS
      SourceIdentifier: ROOT_ACCOUNT_MFA_ENABLED
```

GUI



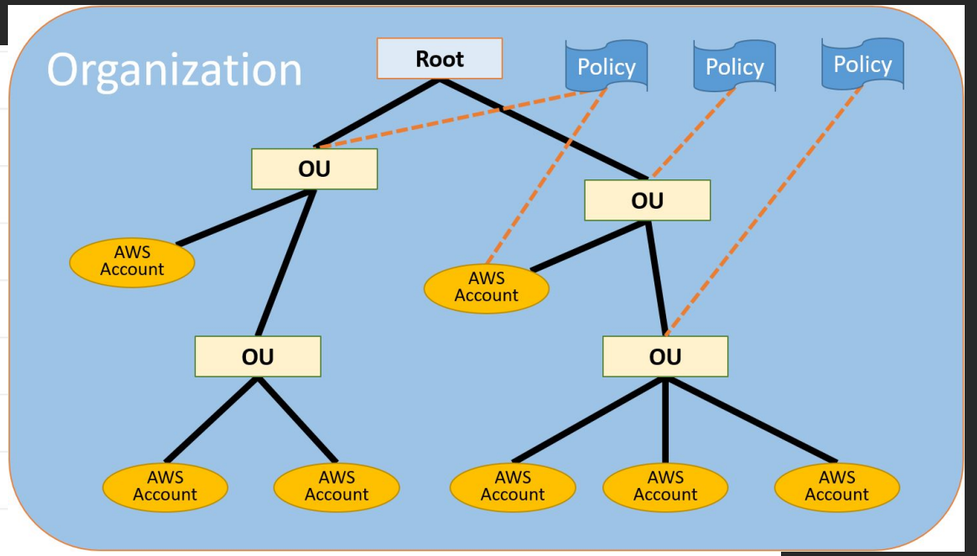
AWS CloudFormation StackSets + AWS Organizationsで実現できたこと

- 1オペレーションで全ての組織アカウントに対してCloudFormation更新が可能になった。各アカウントにログインも不要

- 自動デプロイ機能により、新規AWSアカウント作成時のセキュリティ設定を自動化

おまけ

組織単位 (OU) を整理すると、毎回自動デプロイが走ってカオスなことに



○	5ec83eeaff06d1837e58d91e5421f048	CREATE	FAILED		
○	674302ddc1ed9f1603b7da3cf1ba24d5	CREATE	FAILED		
○	9411b8826f0e69c7b8b1c7c1ed2a73df	CREATE	FAILED		
○	b6196ce06085e7c5c766808c79f5772d	CREATE	FAILED		
○	0d75c2abda61271aa606111bc295f7fa	CREATE	FAILED		
○	bffd1cfb188aee4598dd15db367b5328	CREATE	FAILED		
○	8fc6c59c93320831a134903ebbb6bf1a	CREATE	FAILED		
○	1e3e0646b9ebbcf024df37e18dfc02f5	CREATE	FAILED		
○	ec37dd4a7890f804683adb0793a4e4f6	CREATE	FAILED		
○	ade4cb7cb11795439702e4c5ce31e332	CREATE	FAILED	2020-04-15 18:17:58 UTC+0900	2020-04-15 18:23:11 UTC+0900
○	a2ec1f96cc57ae2b5f715a497aae4a6e	CREATE	FAILED	2020-04-15 18:17:57 UTC+0900	2020-04-15 18:23:42 UTC+0900
○	3c0e3206daa3fc15151fd183b23a2dbe	CREATE	FAILED	2020-04-15 18:17:56 UTC+0900	2020-04-15 18:24:12 UTC+0900
○	8cf913f4b4e581994b56d9a7b9699813	CREATE	FAILED	2020-04-15 18:17:55 UTC+0900	2020-04-15 18:24:43 UTC+0900
○	74e8051c0f3c418be3eb3bcfa5b5a1e1	CREATE	FAILED	2020-04-15 18:17:53 UTC+0900	2020-04-15 18:25:14 UTC+0900
○	782c3dd534d515fa10667a07ad9f1c59	CREATE	FAILED	2020-04-15 18:17:52 UTC+0900	2020-04-15 18:25:44 UTC+0900
○	f8762af158095f6718a52be988276acd	CREATE	FAILED	2020-04-15 18:17:51 UTC+0900	2020-04-15 18:26:15 UTC+0900

Tips

停止状態のAWSアカウントが存在する場合は、停止アカウント数 * リージョン分失敗して処理がロールバックするため、障害耐性 (FailureToleranceCount) オプションを使って回避する

us-west-1	CURRENT	-
us-west-2	CURRENT	-
ap-northeast-1	OUTDATED	Account 043533898082 should have 'stacksets-exec-70f042e3a59f596617c649ca9c96207a'
ap-northeast-2	OUTDATED	Account 043533898082 should have 'stacksets-exec-70f042e3a59f596617c649ca9c96207a'

その他の取り組み紹介

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

その他の取り組みを一部ご紹介

- セキュリティ設定が削除されない仕組み
  - サービスコントロールポリシー(SCP)で、初期統制の削除を禁止

- AWS Config以外で利用している監査サービス
  - Trusted Advisor
  - Prowler

  ※ Prowlerをサーバレスに全アカウント一括チェックしている話なども
  がっつりしたかったのですが、時間の関係上割愛

選定してないAWSサービス

- Amazon Inspector
  - UI、フィルタ機能が使いづらいので、FutureVuls + SSM Patch Managerを採用
- ECRスキャン
  - 有効化はしているが、やや結果が微妙。こちらもFutureVulsを採用
- AWS Security Hub
  - 統合UIのみなので使っていない。統合UIが欲しい方は使うと良いと思います
- AWS Control Tower
  - 選定時は、Control Towerを有効化する前に作られたアカウントへの統制ができなかった
  - CFn StackSets + SCPで代替
- Amazon Macie / Amazon Detective
  - Macieは安くなったが、それでも絶対値としてめっちゃ安いわけではないので必要になれば

全体まとめ

全体まとめ

- AWS Configのマルチアカウント運用により 6000個以上のセキュリティポリシー監査を自動化

- CloudFormation StackSetsにより 全AWSアカウントのセキュリティ設定更新を自動化。さらに自動デプロイ機能により、新規AWSアカウント作成時のセキュリティ設定も自動化

- SCPによりこれらの初期統制が削除されない仕組みを導入

セキュリティポリシーの見直しや、様々なご助言をいただいた
AWSプロフェッショナルサービスのみなさまにも感謝いたします

最後に

アカツキのエンジニア組織

情熱が偉大なゲームを生み出す

課題解決のプロ集団

技術を使って感情を動かす

自ら課題を見つけ、考え、動きたい、
楽しく働きたいという方、Welcomeです !!
お待ちしております！

アカツキのエンジニア組織

ゲーム事業部特設サイト

https://game.aktsk.jp/about/engineer/