マルチアカウント環境のガバナンス実現方法＋BLEA概説

アマゾン ウェブ サービス ジャパン 合同会社
部長/シニアソリューションアーキテクト
大村 幸敬（ohmurayu@amazon.co.jp）

2024/06/20

アジェンダ

1. クラウドを活かすガバナンスの考え方
2. スケールするガバナンスの実現方針ガバナンスの実現方針
3. 中央集権で管理する範囲を広げる場合の考え方と注意点
   - ネットワーク設定に対するガバナンスの例
4. Baseline Environment on AWSの概要
   1. BLEAが提供するガバナンスの具体例
   2. ベースラインの展開パターン
   3. BLEAのカスタマイズ
   4. BLEA利用時に必要なスキルおよび運用タスク
   5. 参考資料

クラウドを活かすガバナンスの考え方

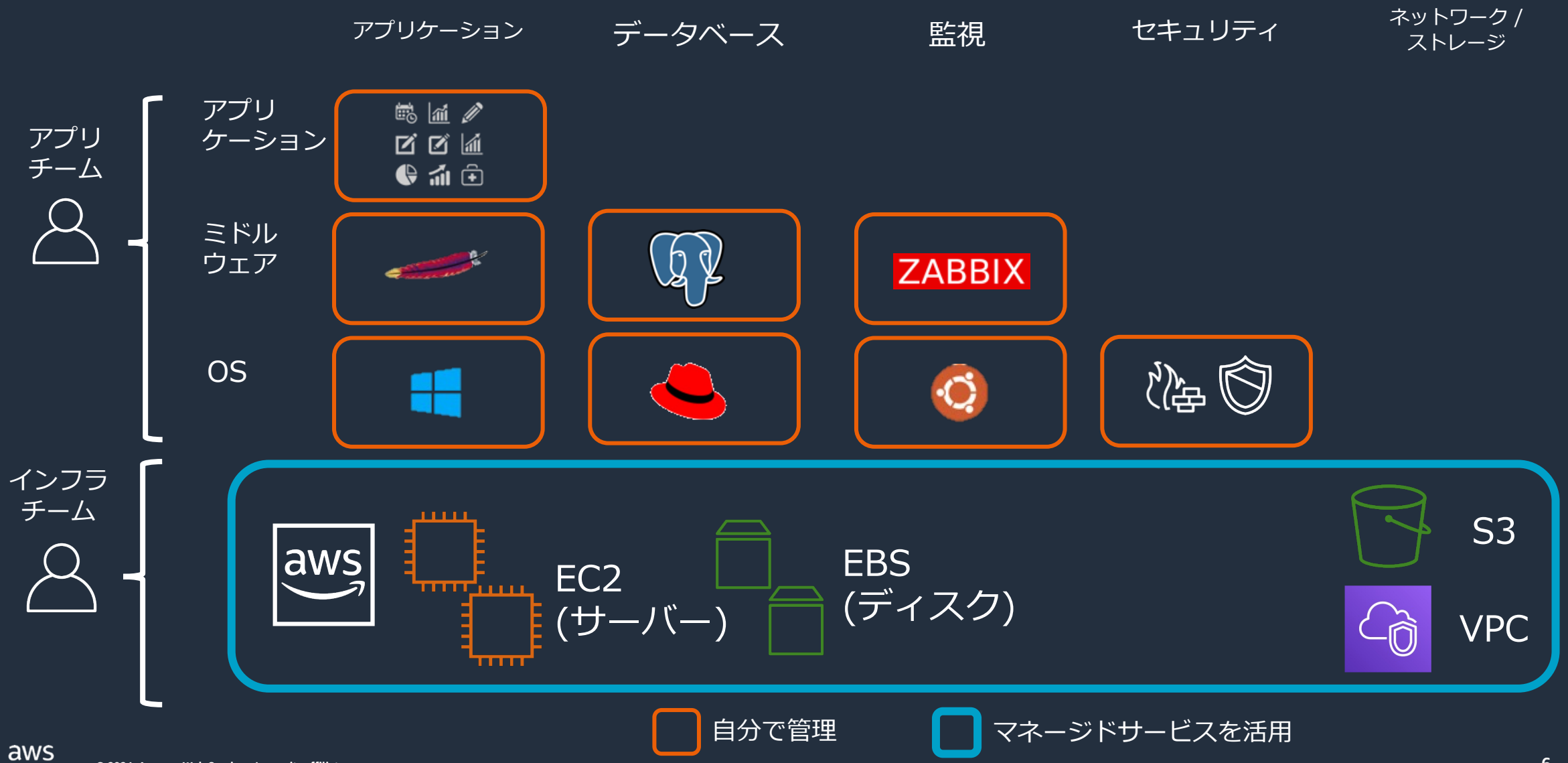
クラウドが提供する価値

Builderを支えるセルフサービスプラットフォーム

Builder（手が動く開発者）に自由を与え、
承認なく適切な箇所で適切なツールを使えるようにする
それによって柔軟・迅速にビジネス価値を実現できる

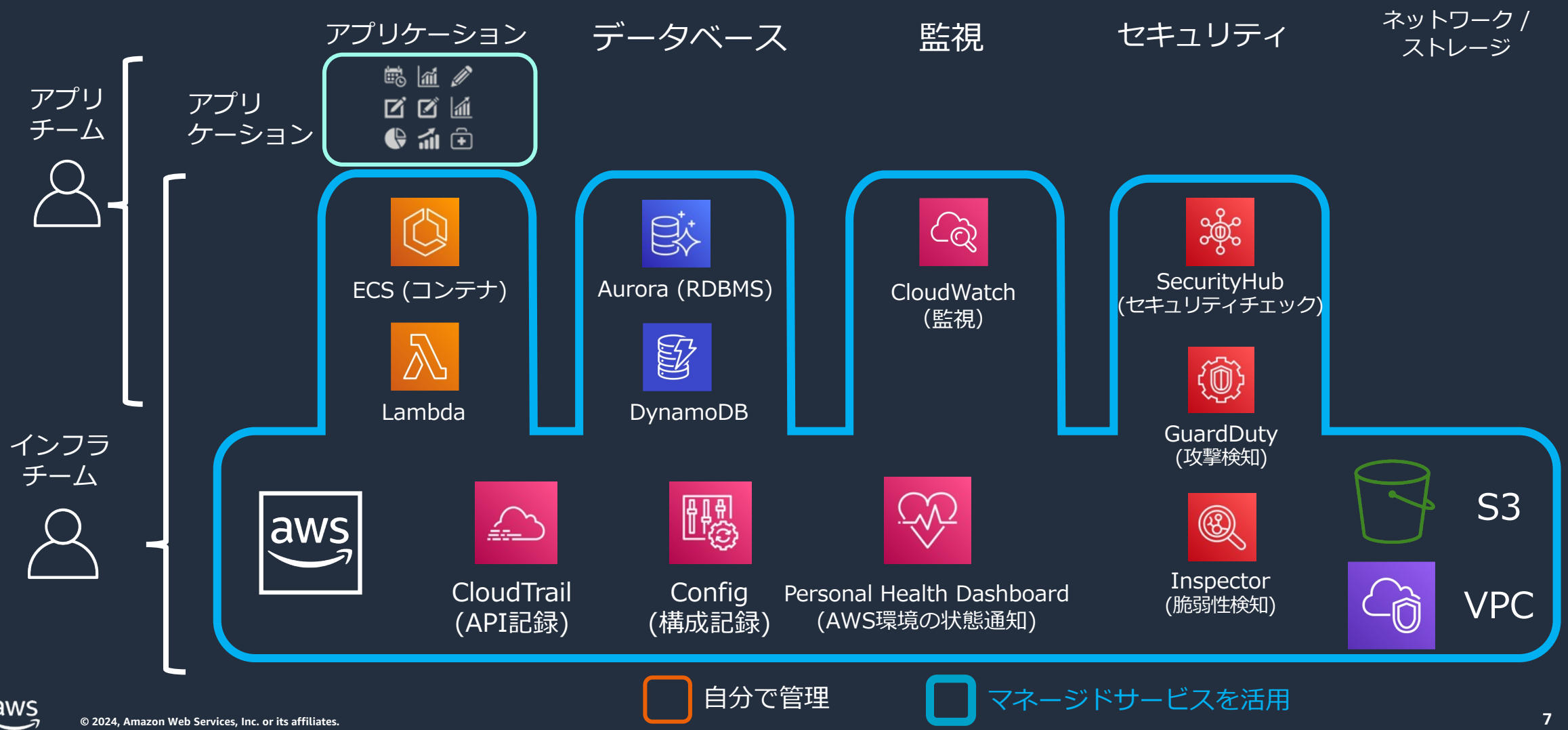
オンプレミスのシステム構成例

クラウドのシステム構成例（Liftパターン）



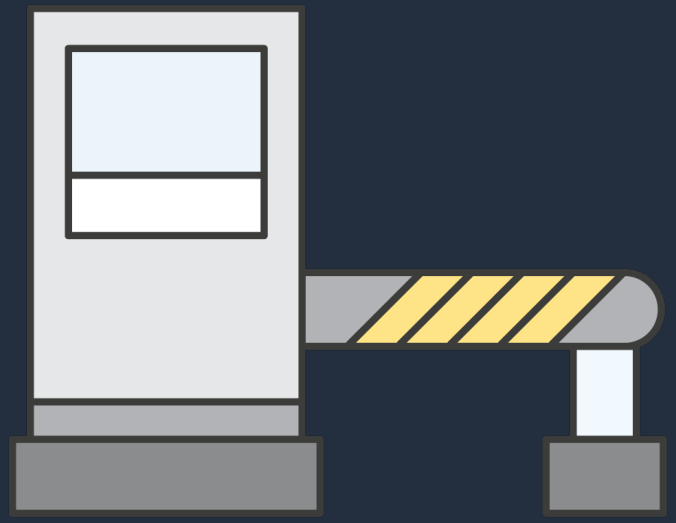
クラウドをフル活用したシステム構成例

マネージドサービス活用により コスト効率よくビジネス価値を実現することに集中



Builderに必要なガバナンス

Gatekeeper

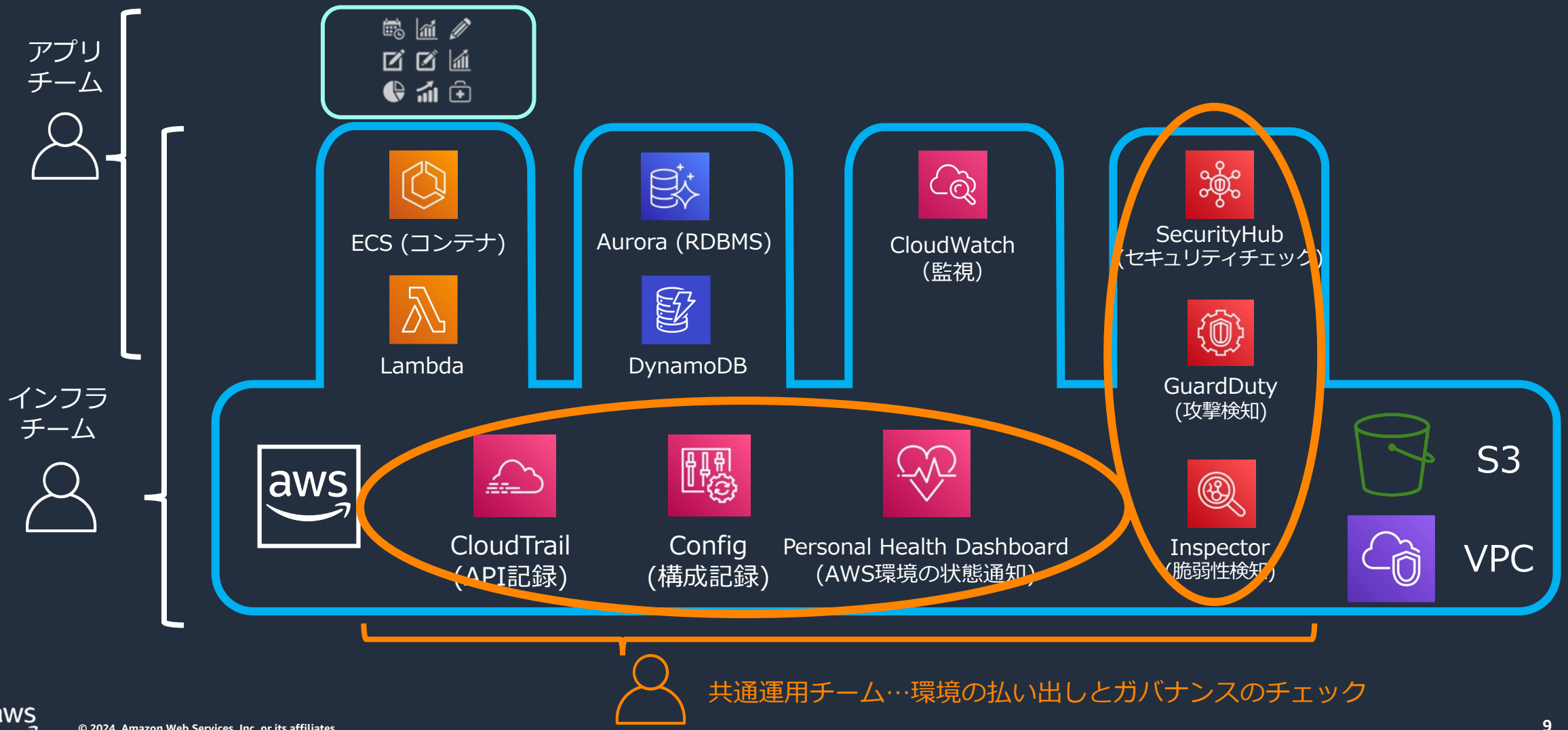


V.S.

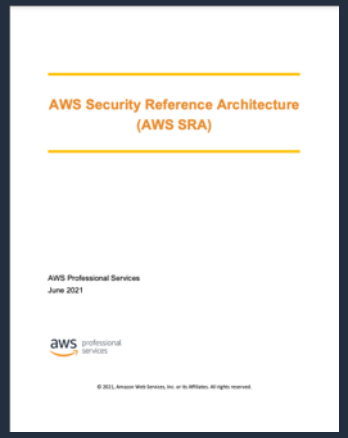
Guardrail

ツールの利用を事前承認(Gatekeeper)すると管理業務がボトルネックになる。
各システムで自由に使わせる一方で、Builderを守るためガードレール(Guardrail)を用意する。
やってはいけない操作を未然に防ぐこと（予防的統制）、逸脱を検知すること（発見的統制）の2種類。

セキュリティサービスを活用したガードレールの実現

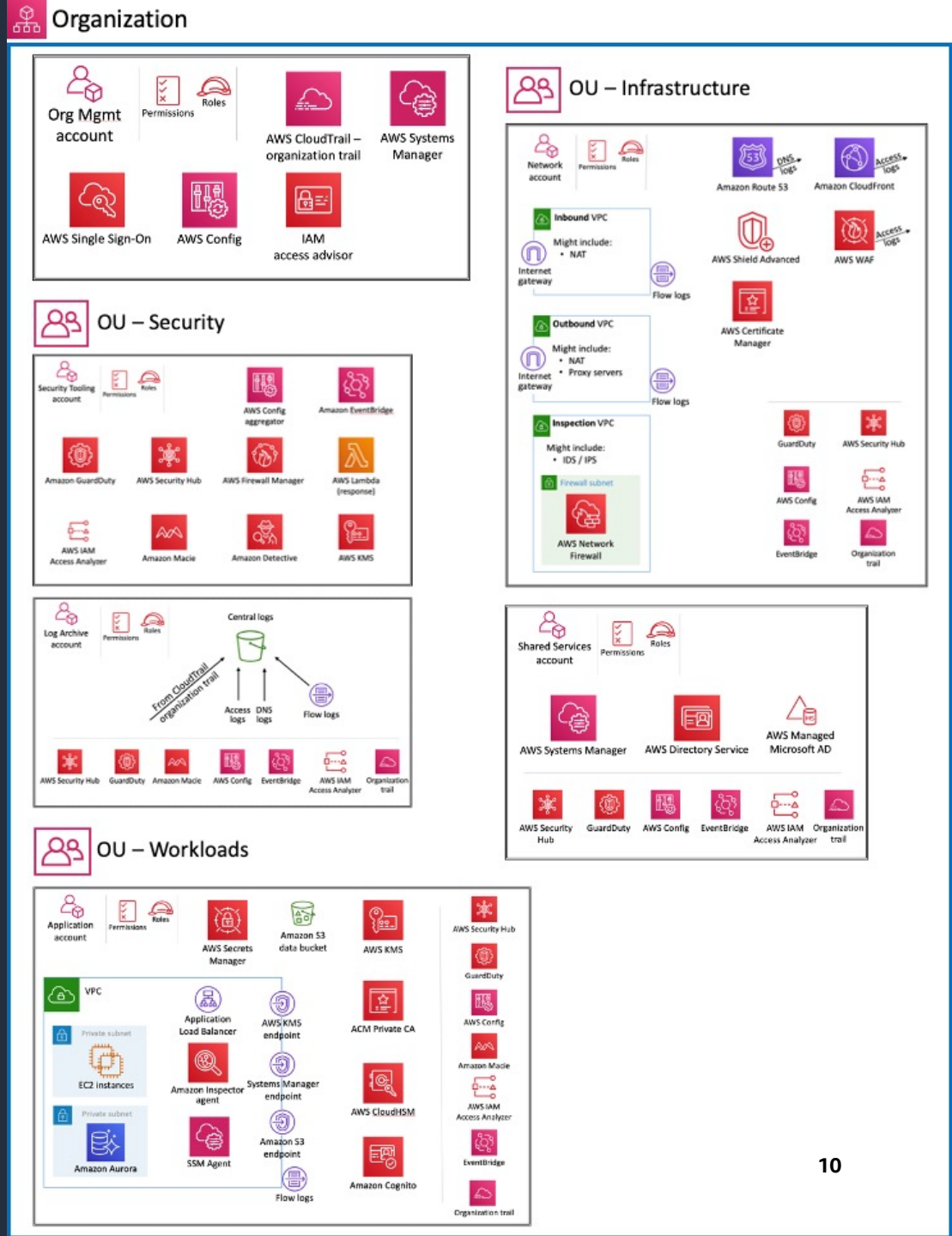


AWS Security Reference Architecture (AWS SRA)



AWSのセキュリティサービスを
フル活用する際のガイダンスと実装例
（多くの場合すべてを実装する必要はない）

- セキュリティガバナンス
- セキュリティ監査
- 脅威検出
- 脆弱性管理
- インフラストラクチャ保護
- データ保護
- アプリケーションセキュリティ
- インシデントレスポンス



https://docs.aws.amazon.com/prescriptive-guidance/latest/security-reference-architecture/welcome.html

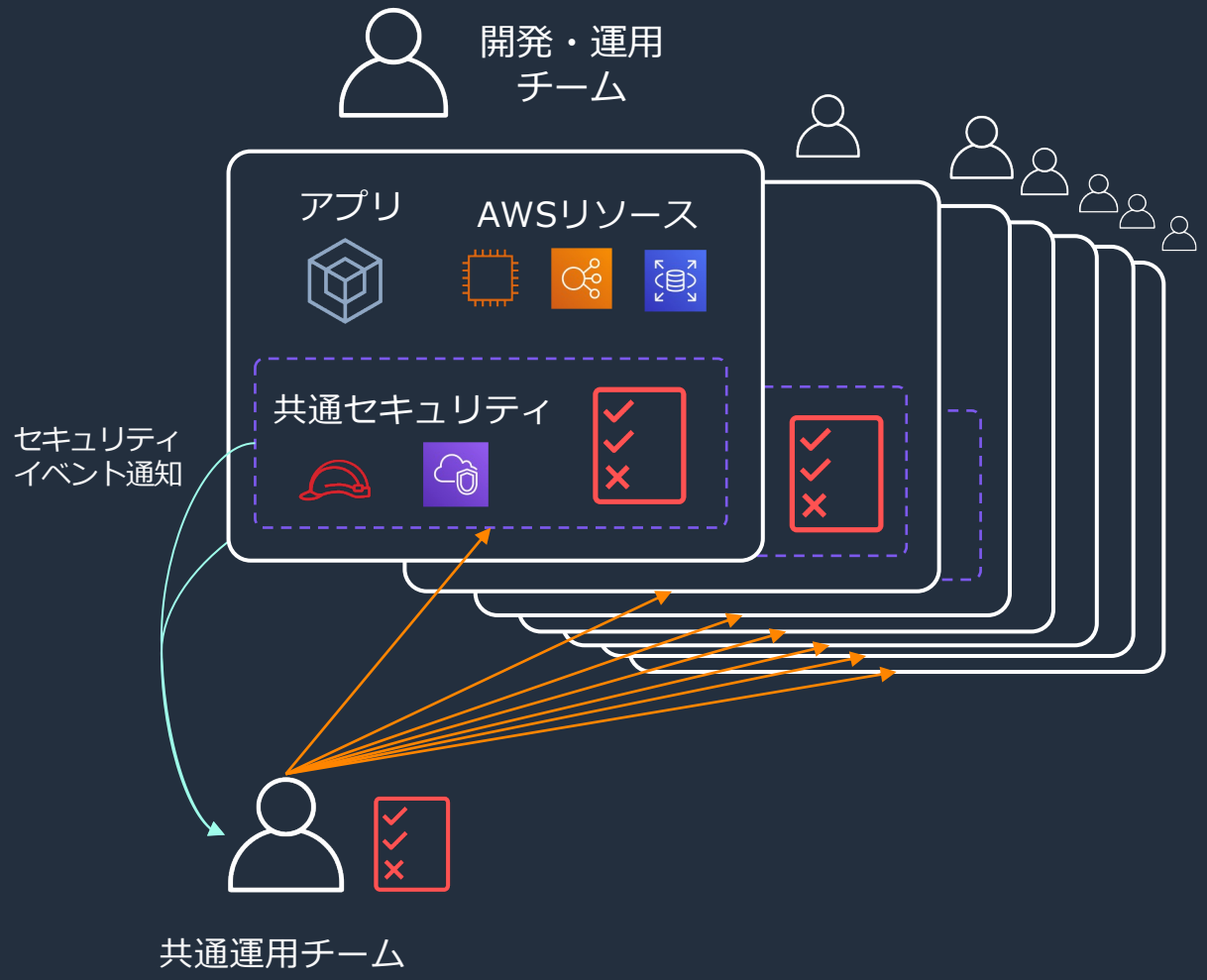
アカウントに対して
ベースラインとして設定を推奨する セキュリティサービス

複雑な設定は不要、コストも小さく、稼働するワークロードに影響を与えない
まずはこれらをONにしてセキュリティレベルのベースを上げることを推奨

スケールするガバナンスの実現方針

よくある落とし穴

よくある課題：中央集権管理がクラウドの価値を下げてしまう



中央集権的な管理作業の例

- 共通運用チームが…
  - 1つのセキュリティポリシーを全体へ適用する
  - 各システムのセキュリティ設定を実施する
  - 各システムのセキュリティイベントを対応する
  - 各チームメンバのアクセス管理を実施する

課題

- マネージドサービスが活用できない
  - 共通セキュリティポリシー変更に時間を要する
  - AWSの機能拡張にルールが追従できない
- 共通運用チームがボトルネック
  - システム数増加に手動作業が追いつかない
  - 多量のセキュリティイベントに対応できない
  - 個々のシステムで認めた例外の管理が困難

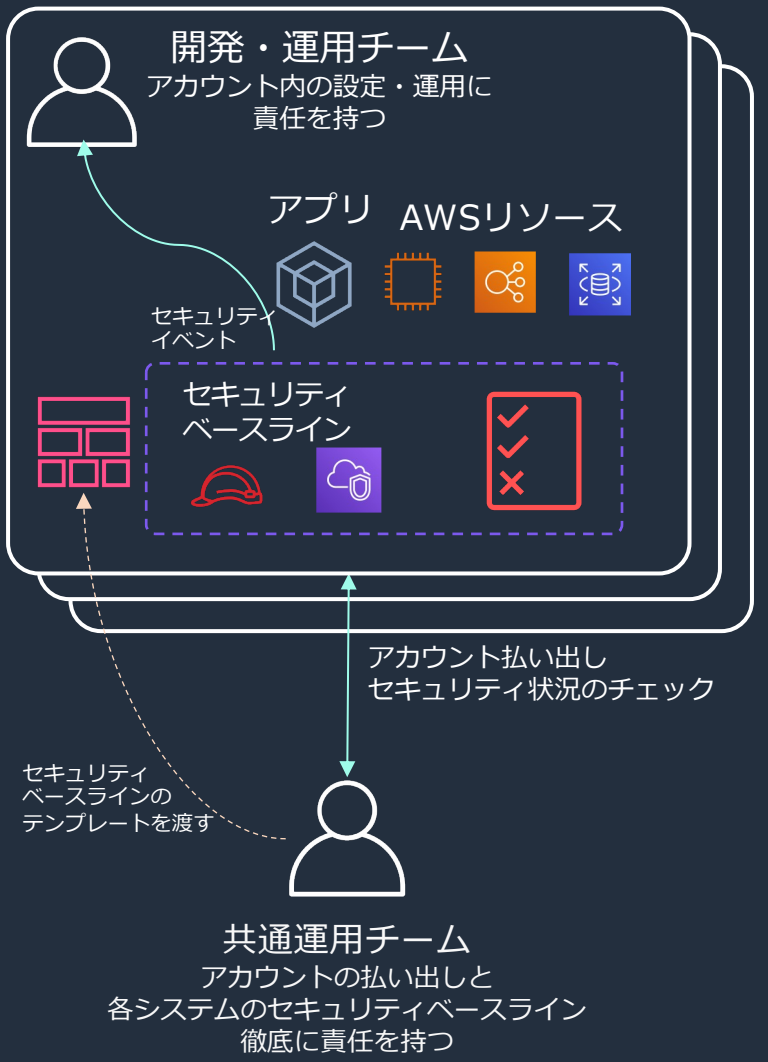
スケールするガバナンスの具体的な実装例
1）アカウントをまるごと渡す

スケールするガバナンスの実現方針

- 環境ごとにアカウントを払い出す
  - アカウント内は自由にAWSを利用可能
    （最低限の制限のみ設定）
  - 他のアカウントに影響を与えない

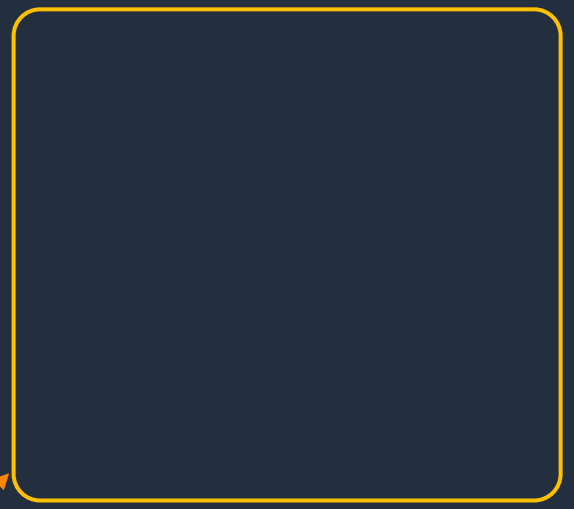
- 払い出したアカウントの管理は開発・運用チームに委ねる
  - 共通運用チームはセキュリティベースラインを用意して、
    開発・運用チームが自分でそれを展開
  - 開発・運用チームはアカウント内の各種設定・運用に責任を持つ

- 各アカウントの状況はセキュリティサービスで集約して閲覧
  - 共通運用チームは自らが望むタイミングでまとめて情報を得る
  - セキュリティイベントは開発・運用チームが自ら受ける



テンプレートを使ったガバナンスの全体像

開発・運用チーム

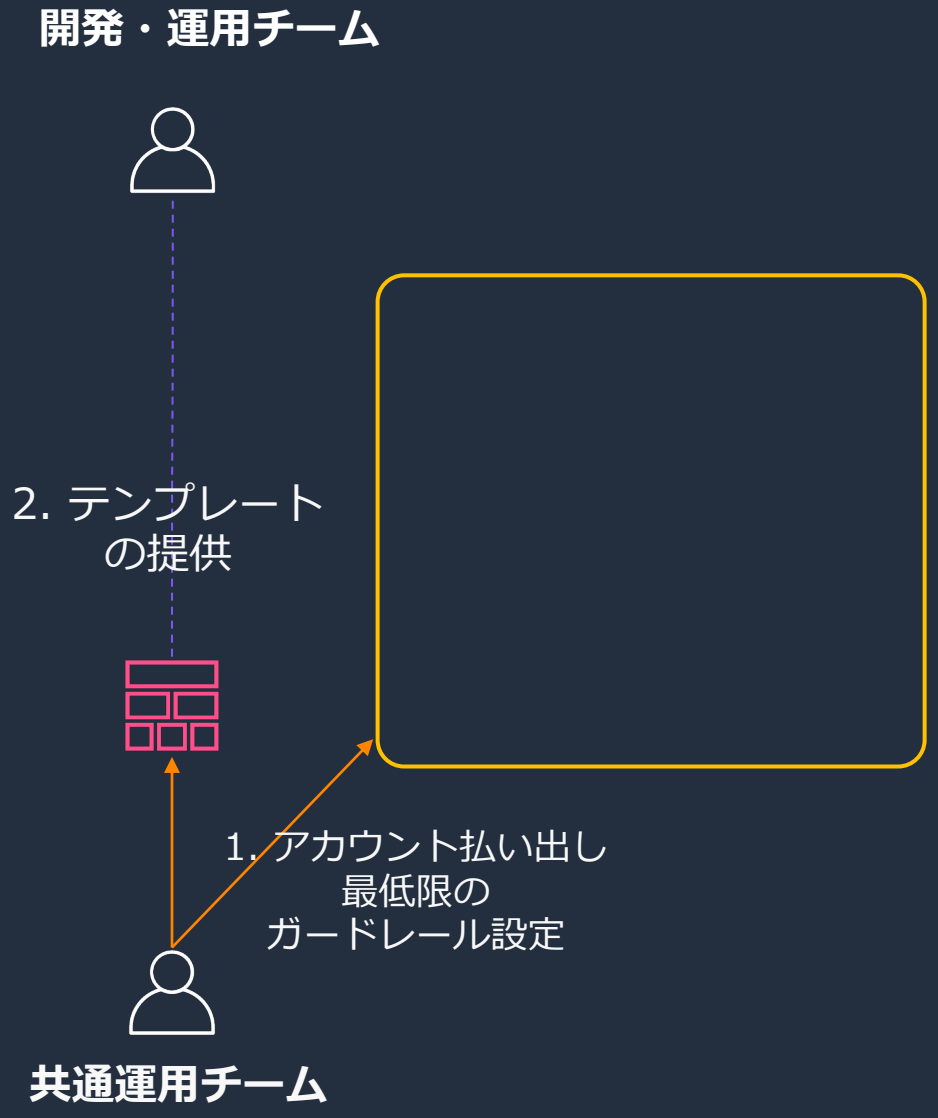


1. アカウント払い出し
最低限の
ガードレール設定

共通運用チーム

1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

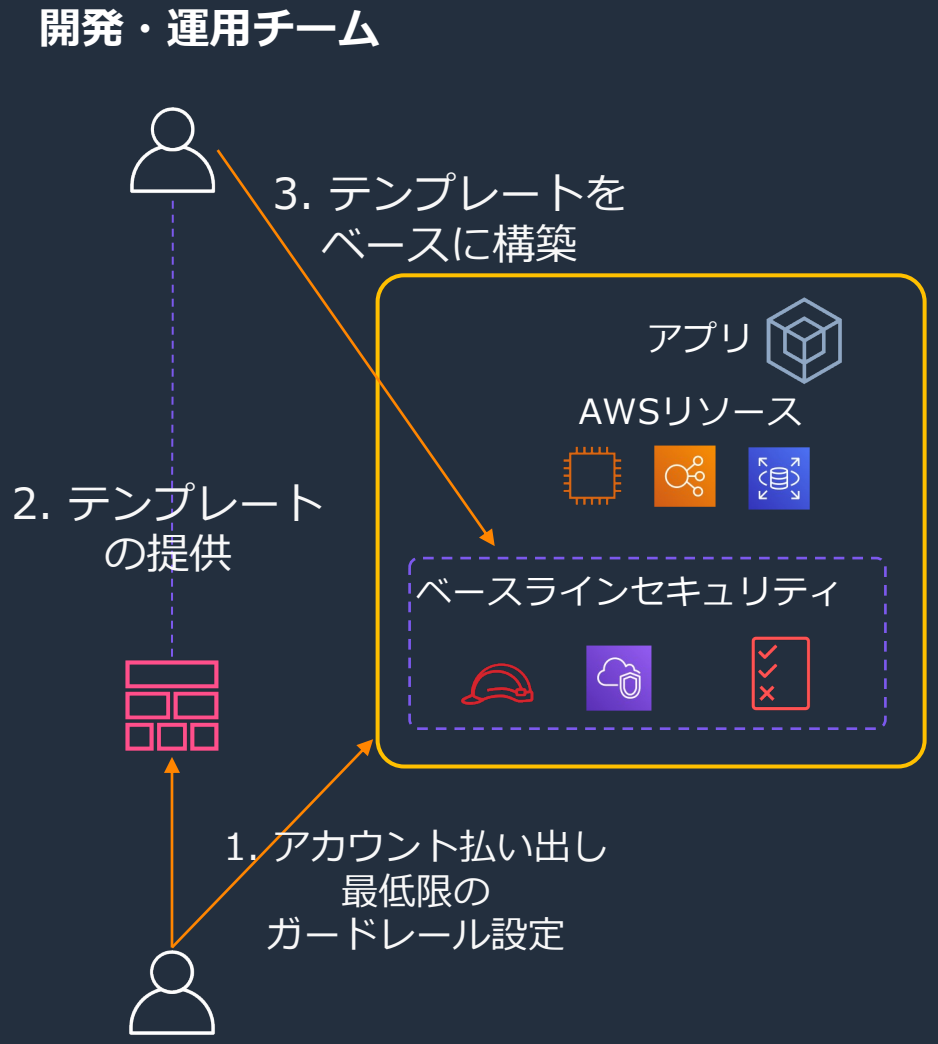
テンプレートを使ったガバナンスの全体像



1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

2. 「実行可能なセキュリティガイド」としてのテンプレートの配布
   - ガイドや手順書ではなく

テンプレートを使ったガバナンスの全体像

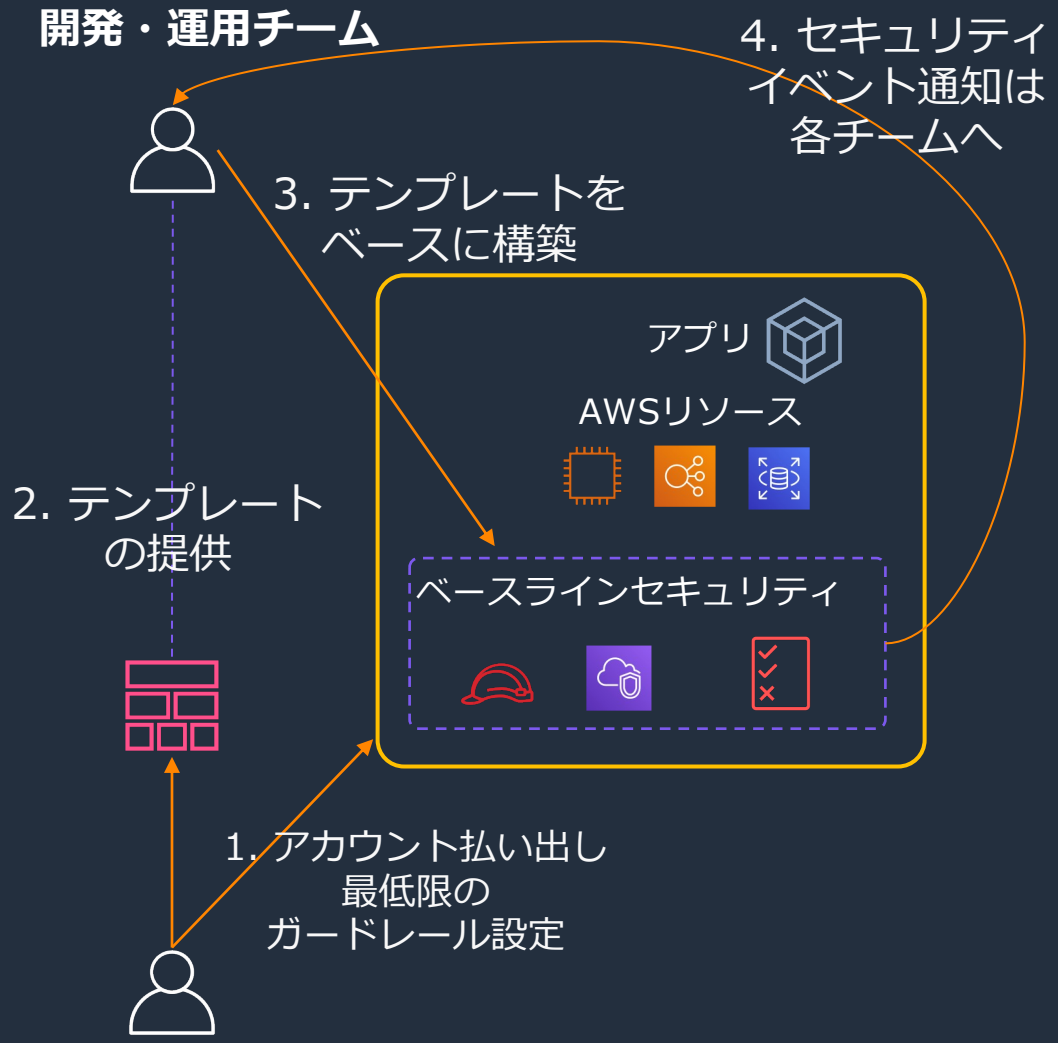


1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

2. 「実行可能なセキュリティガイド」としてのテンプレートの配布
   - ガイドや手順書ではなく

3. 各アカウントの管理責任は各チームが負い、テンプレートも各チームの責任で管理する（カスタマイズやメンテナンスも行う）

テンプレートを使ったガバナンスの全体像



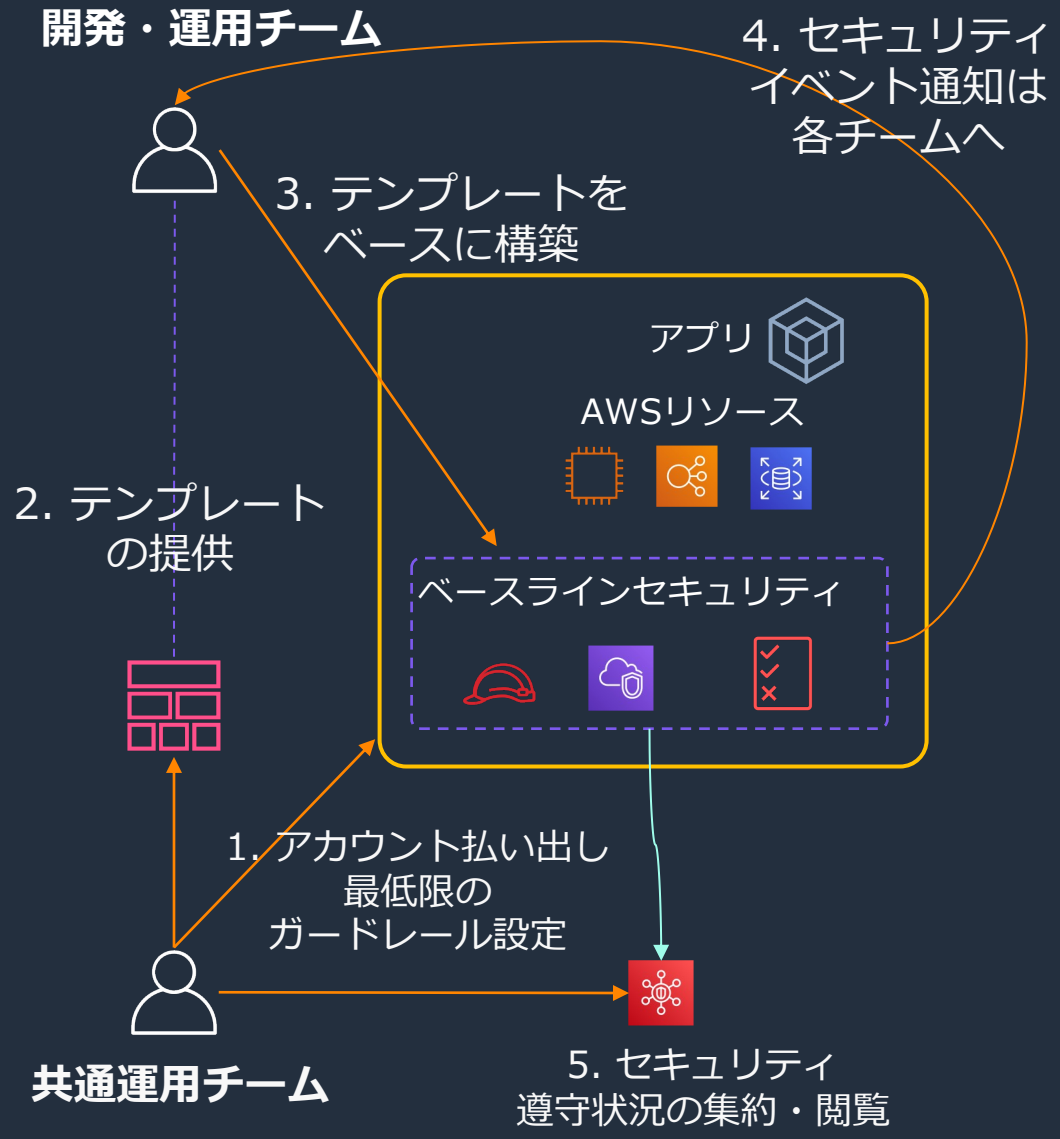
1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

2. 「実行可能なセキュリティガイド」としてのテンプレートの配布
   - ガイドや手順書ではなく

3. 各アカウントの管理責任は各チームが負い、テンプレートも各チームの責任で管理する（カスタマイズやメンテナンスも行う）

4. セキュリティイベントの即時通知は各チームで対処

テンプレートを使ったガバナンスの全体像



1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

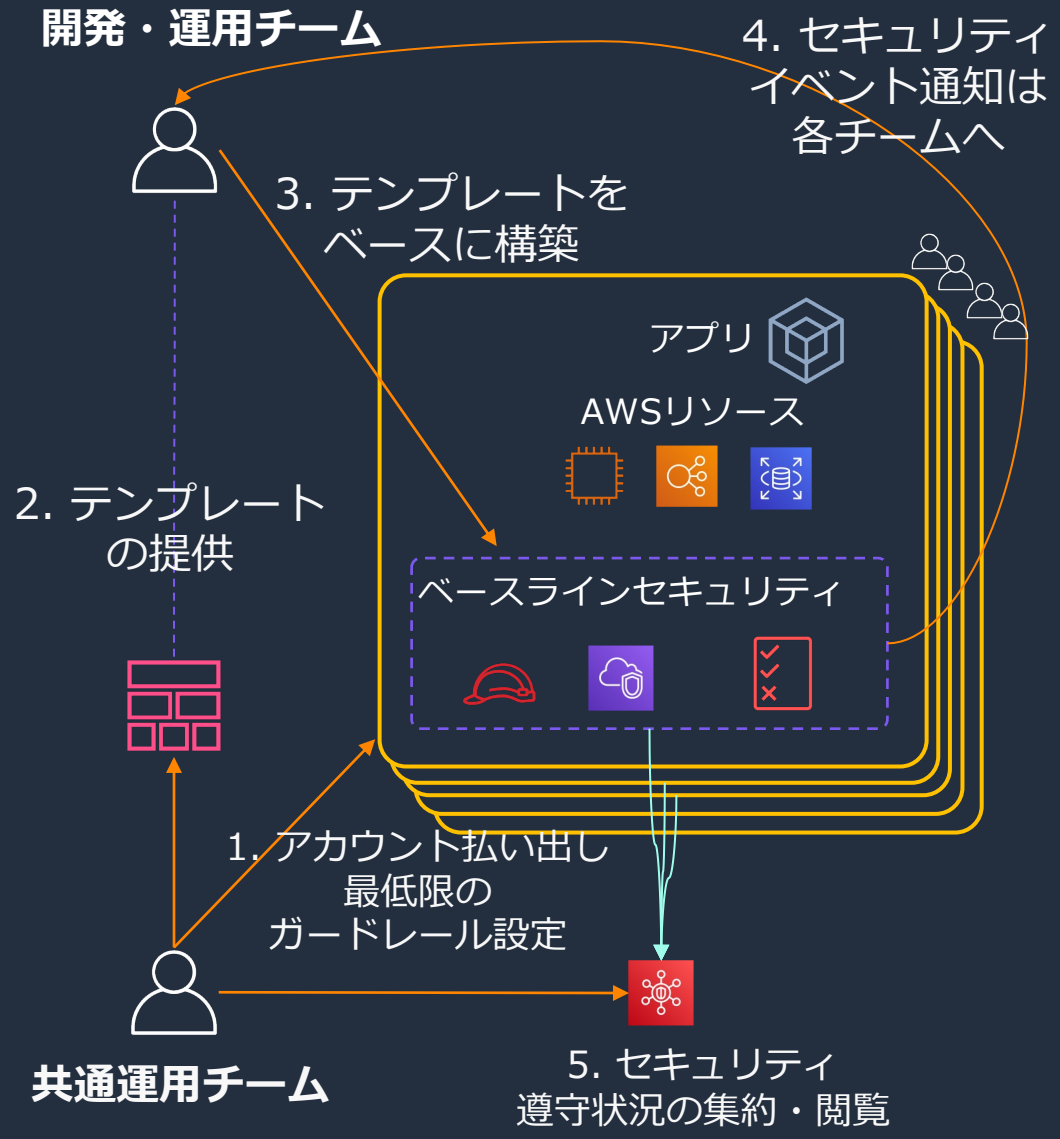
2. 「実行可能なセキュリティガイド」としてのテンプレートの配布
   - ガイドや手順書ではなく

3. 各アカウントの管理責任は各チームが負い、テンプレートも各チームの責任で管理する（カスタマイズやメンテナンスも行う）

4. セキュリティイベントの即時通知は各チームで対処

5. 重要なセキュリティイベントや定期的な遵守状況の確認は共通運用チームが実施

テンプレートを使ったガバナンスの全体像



1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う
2. 「実行可能なセキュリティガイド」としてのテンプレートの配布
   - ガイドや手順書ではなく
3. 各アカウントの管理責任は各チームが負い、テンプレートも各チームの責任で管理する（カスタマイズやメンテナンスも行う）
4. セキュリティイベントの即時通知は各チームで対処
5. 重要なセキュリティイベントや定期的な遵守状況の確認は共通運用チームが実施

中央集権で管理する範囲を広げる場合の考え方と注意事項

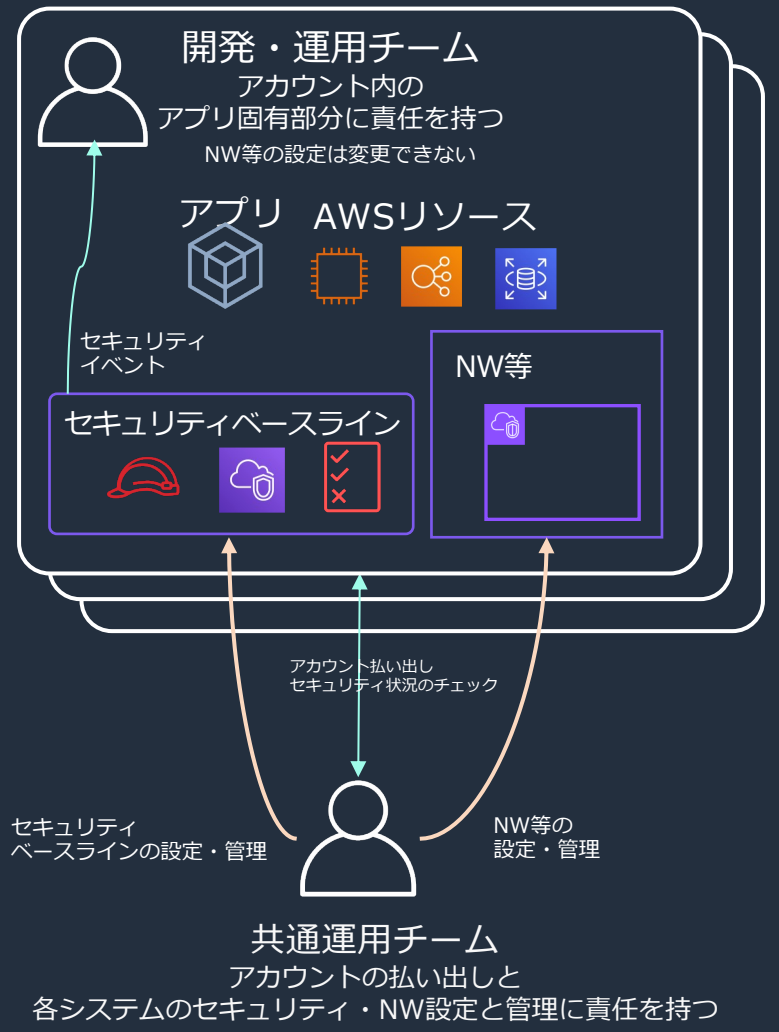
中央集権でベースラインを管理するガバナンスの実現方針（例

アカウントが増えた場合に、管理し切れるか、あとから設定変更可能かよく検討する

- 環境ごとにアカウントを払い出す
  - アカウント内は自由にAWSを利用可能
    ただしNWやセキュリティベースラインの設定は変更不可
  - 他のアカウントに影響を与えない

- 払い出したアカウントのアプリ固有部分のみ
  開発・運用チームに委ねる
  - 共通運用チームがセキュリティベースラインとNW等を設定
    全アカウントの当該設定を共通運用チームで管理
  - 開発・運用チームはアカウント内のアプリ固有部分のみ
    設定・運用に責任を持つ（サーバ、Lambda等を含む）

- 各アカウントの状況はセキュリティサービスで集約して閲覧
  - 共通運用チームは自らが望むタイミングでまとめて情報を得る
  - セキュリティイベントは開発・運用チームが自ら受ける



ネットワーク設定に対するガバナンスの例

ネットワーク設定を集中管理する場合のガバナンス（例）

- 予防的統制
  - ゲストアカウントで、ネットワーク設定を変更できないよう制限
    - SCP （OUまたはアカウント単位） / IAM Policy (boundary) で設定
    - VPC等はタグを付与して、中央管理者のみ変更可能、他のエンティティは変更不可
    - 例）IGWのアタッチ、VPCやサブネットの作成、EIPのアタッチなどを禁止する

- 発見的統制（こちらを強化することを推奨）
  - ゲストアカウントで、ルールを逸脱した設定を検知して通知
    - SecurityHubやConfig rulesを設定（これらの設定を無効化できないようIAM等で制限）
    - ゲストアカウント管理者に通知、悪質なものは中央管理者に通知 or レポーティング
    - 例）許可されないVPCのIGWアタッチ、EIPがアタッチされたインスタンスを検知

ネットワーク設定を集中管理する具体的な設定例

- 予防的統制
  - Amazon VPC ポリシーの例
  - SCPを利用して複数AWSアカウントのEC2へのパブリックIP付与を禁止する

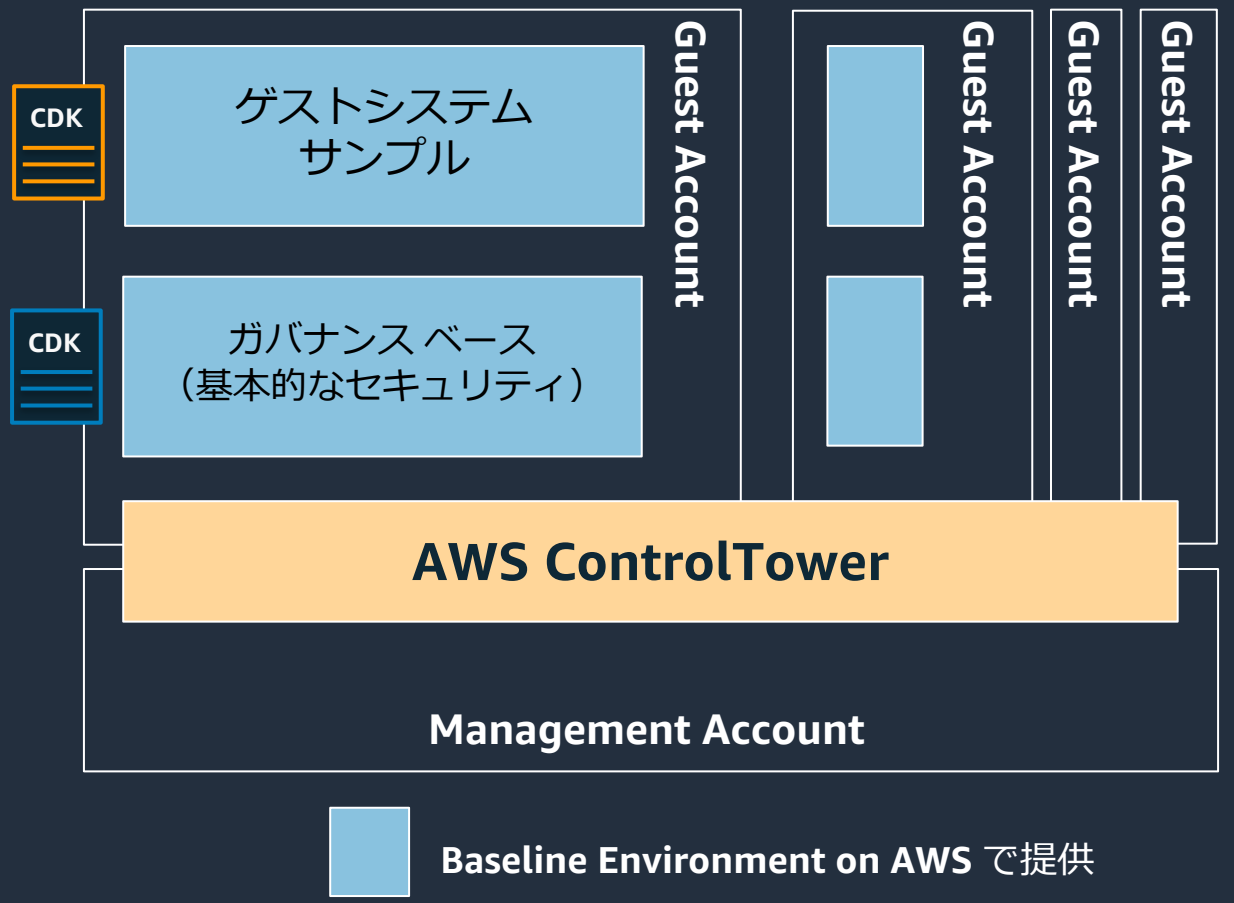
- 発見的統制（こちらを強化することを推奨）
  - SecurityHub
    - [EC2.9] Amazon EC2 インスタンスは、パブリック IPv4 アドレスを未設定にすることをお勧めします
    - [EC2.15] Amazon EC2 サブネットは、パブリック IP アドレスを自動的に割り当てないことをお勧めします
    - [EC2.18] セキュリティグループは、許可されたポートに対して無制限の着信トラフィックのみを許可することをお勧めします
    - [EC2.19] セキュリティグループは、リスクの高いポートへの無制限アクセスを許可してはいけません
  - Config Managed Rules
    - eip-attached
    - internet-gateway-authorized-vpc-only

Baseline Environment on AWS の概要

Baseline Environment on AWS

AWSのセキュリティベストプラクティスを実装したサンプルテンプレート

https://github.com/aws-samples/baseline-environment-on-aws



- セキュリティのベースラインを提供
  - ControlTowerの機能をベースに
    足りない部分を ガバナンスベース で提供
  - AWSのセキュリティサービスをフル利用
- ゲストシステムのサンプルを提供
  - ガバナンスベースを土台として構築する
    Webアプリケーションなどのサンプル
  - AWSセキュリティベストプラクティスを実装
- AWS CDK* で提供 *Cloud Development Kit
  - AWS JapanのSAがオープンソースで公開
  - CDKのリファレンスや学習利用を想定
  - シンプル、引き継ぎやすさを重視

BLEAについてよくある質問

Q: BLEAは公共機関や金融機関など高いセキュリティレベルを実現するための重厚な仕組みなのですか

A: いいえ。逆です。むしろ俊敏性を必要とする環境において最低限のセキュリティを提供するための薄い仕組みです。より高いセキュリティのための第一ステップを容易に実現します。

Q: BLEAを使うためにはBLEAそのものに関する習熟が必要ですか。

A: いいえ。BLEAは極力独自の仕組みを作るのではなく、AWSのセキュリティサービスをそのまま使っています。AWSの標準的な知識だけで理解できるよう作られています。

Q: 開発者がCDKを利用できないと使えませんか。

A: いいえ。管理者が Service Catalog にベースラインを登録し、開発者がそれを使ってベースラインを展開する方法があります。

BLEAが提供するガバナンスの具体例

BLEA で使用している主要 AWS サービス

アカウント統制	マルチアカウント環境統制 アカウント払い出し Control Tower	アカウントを 1つの組織として統合 Organizations	マルチアカウント SSO 環境 Single Sign-On	ID/ポリシー管理 Identity & Access Management	
セキュリティ 担保	証跡ログ Cloud Trail	メトリクス取得 アラート CloudWatch	構成管理 Config	シークレット管理 Secret Manager	脅威検知 GuardDuty	ベストプラクティス チェック Security Hub
BLEA および ゲストシステム 払い出し・管理	コードによるインフラ管理 Cloud Formation	Cloud Development Kit			

BLEAマルチアカウント版 管理タスクと利用サービス対応表

(ver. 2023/03/20)

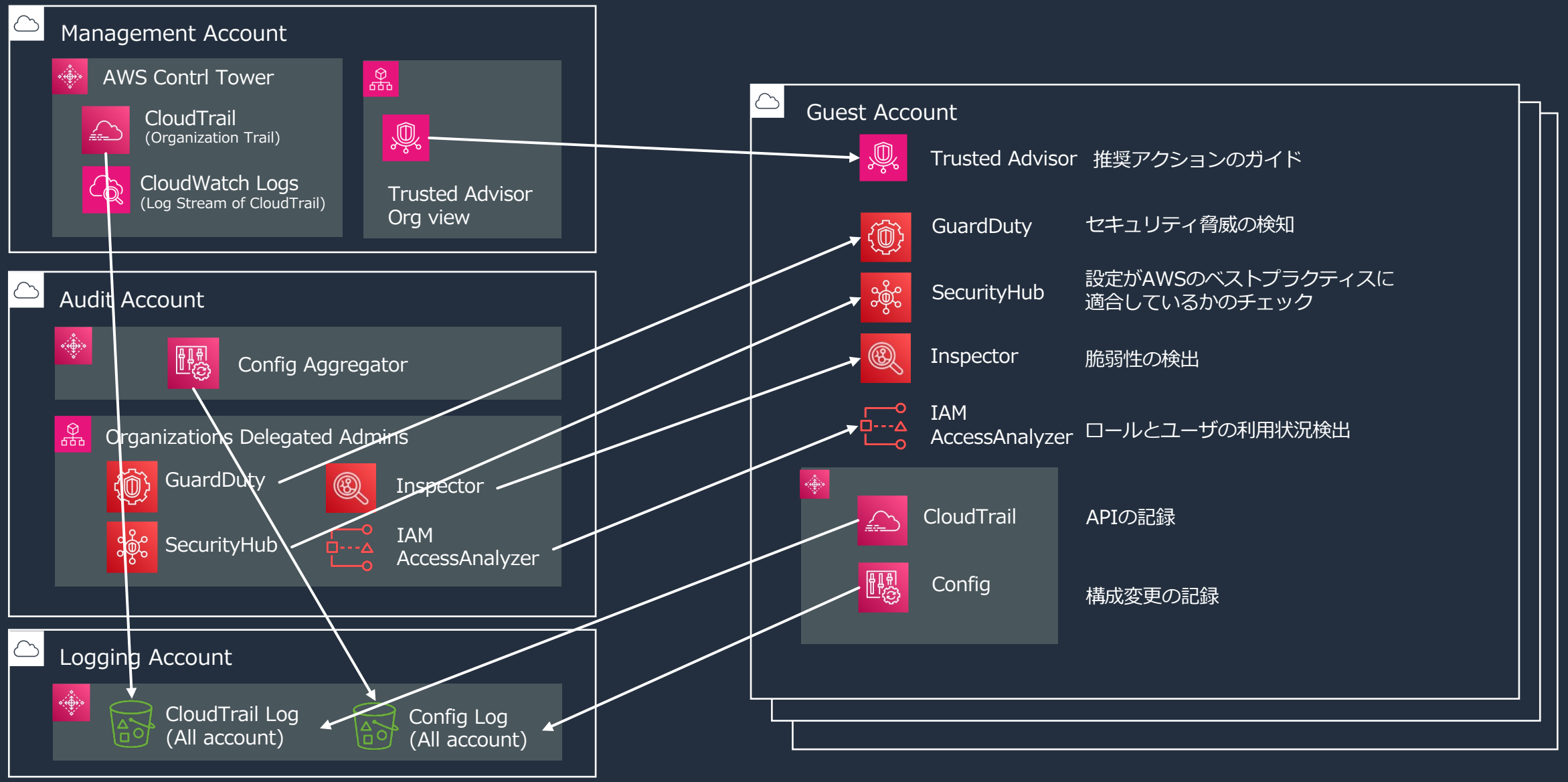
	管理タスク	Management Account	Audit Account	LogArchive Account	Shared Svc Account*	Guest Account
AWS ControlTower	1	アカウント払い出し	CT-Org	(Created by CT)	(Created by CT)	(Created by CT)	(Created by CT)
	2	アクセス制御	CT-SSO/Admin + AD	CT-Admin	CT-Admin	CT-Admin	CT-Admin
	3	予防的統制	CT-SCP	CT-SCP	CT-SCP	CT-SCP	CT-SCP
	4	発見的統制(Config)	CT-ConfigRules作成	CT-ConfigRules	CT-ConfigRules	CT-ConfigRules	CT-ConfigRules
	5	ロギング	CT-Trail for Organization	CT-Config Aggregator	CT-Bucket for Logs	CT-CloudTrail/Config	CT-CloudTrail/Config
	6	通知 (CT)	(CT-Audit Topic)	CT-Audit Topic	(To Audit Topic)	(To Audit Topic)	(To Audit Topic)
Governance Base for CT Guest	7	発見的統制 (挙動)		MNL-SecurityHub MNL-GuardDuty MNL-Inspector			Member-SecurityHub Member-GuardDuty
	8	セキュリティ分析		MNL-IAM-AccessAnalyzer			Member-IAMAccessAnalyzer
	9	共有ネットワーク				TMPL-VPC/DNS/VPCEP *	(Use Shared Svc Account)
	10	サーバ管理					MNL-SSM QuickSetup
	11	通知 (Security)+Chat					TMPL-Security Alarm
	12	アクセス制御 (for Guest)					TMPL-IAM
	13	発見的統制 (for Guest)					TMPL-ConfigRules
Guest System	14	ロギング (for Guest)					TMPL-FlowLogs/ALB Logs etc.
	15	ネットワーク (for Guest)					TMPL-VPC
	16	鍵管理 (for Guest)					TMPL-KMS
	17	通知 (Monitoring)+Chat					TMPL-Monitor Alarm
	18	リソース + バックアップ					TMPL-EC2/Serverless etc.
	19	デプロイメント					TMPL-CI/CD

CT- : Managed by ControlTower / TMPL- : Provide Templates (CDK) / MNL-: Manual / *: Not implemented

管理タスクの主体　管理される側

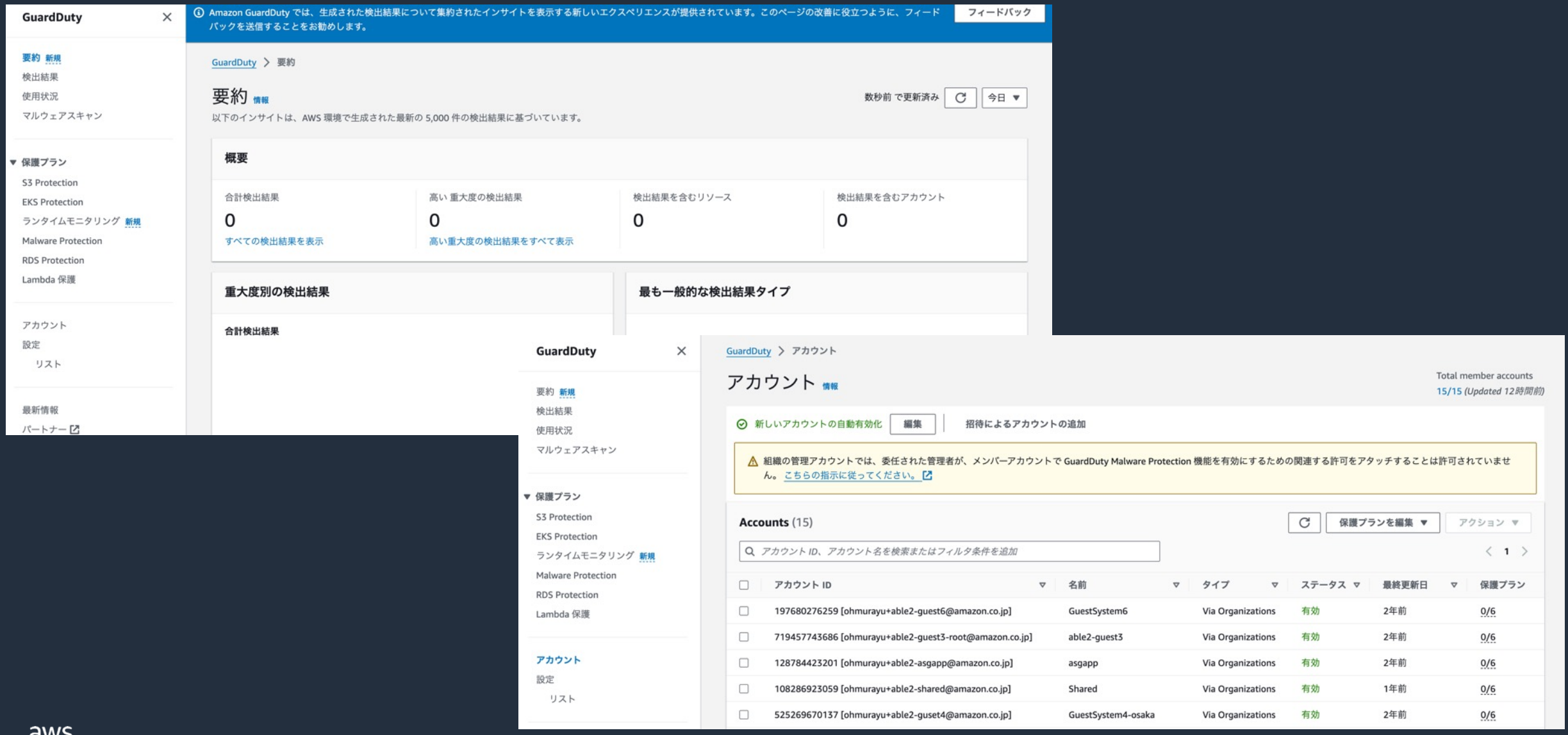
BLEA マルチアカウント版 – セキュリティベースラインの全体像

BLEA マルチアカウント版 – セキュリティベースラインの全体像

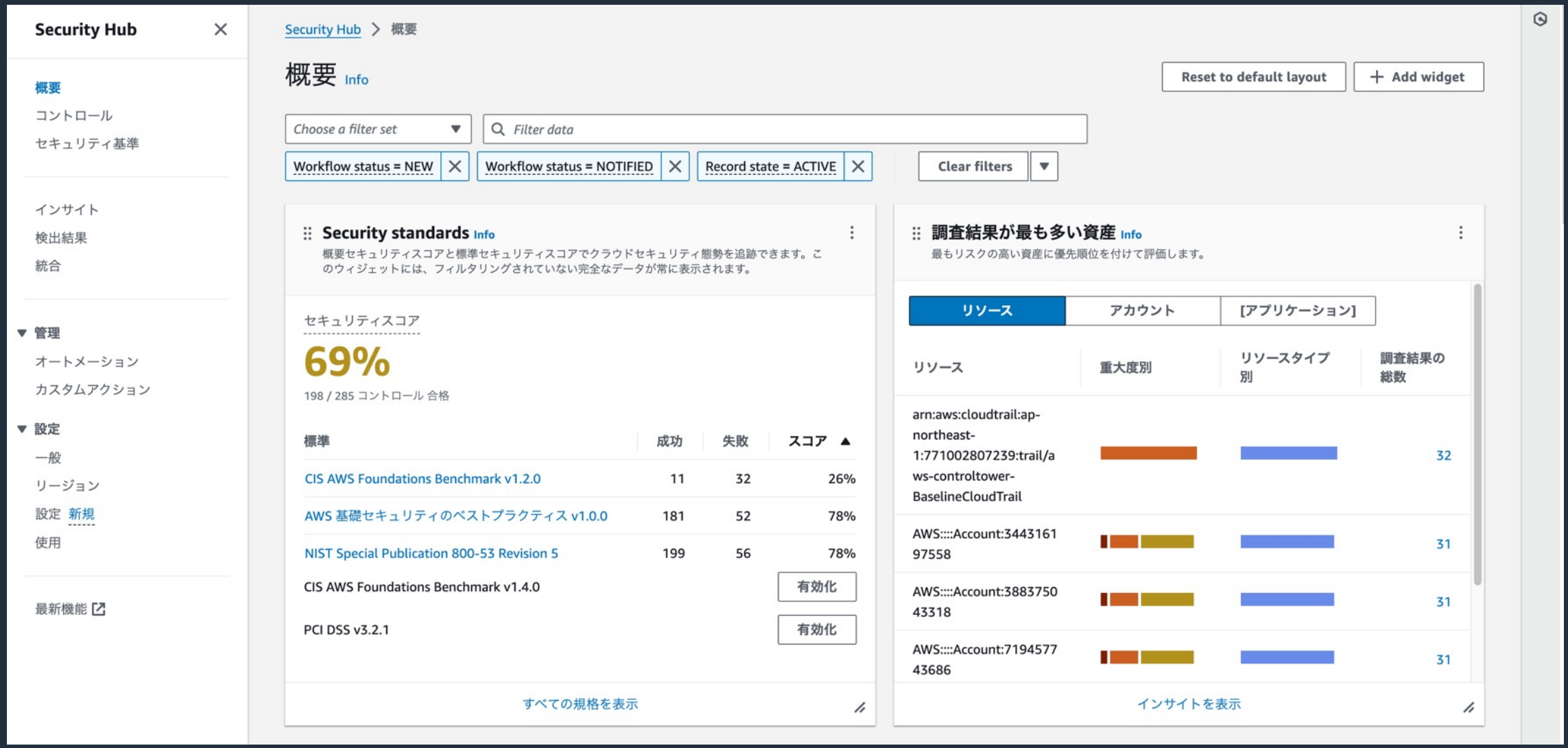


Trusted Advisor – Organizations View

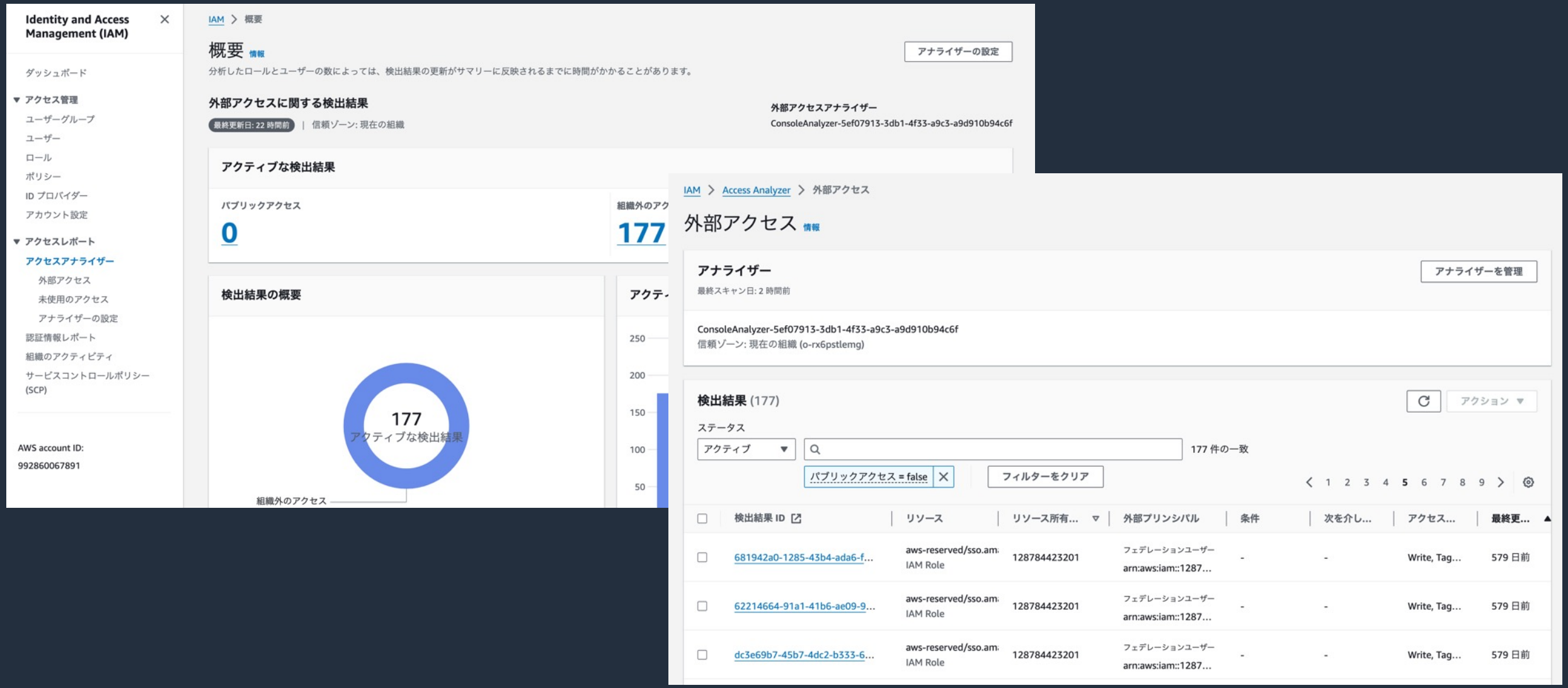
GuardDuty – Organization summary



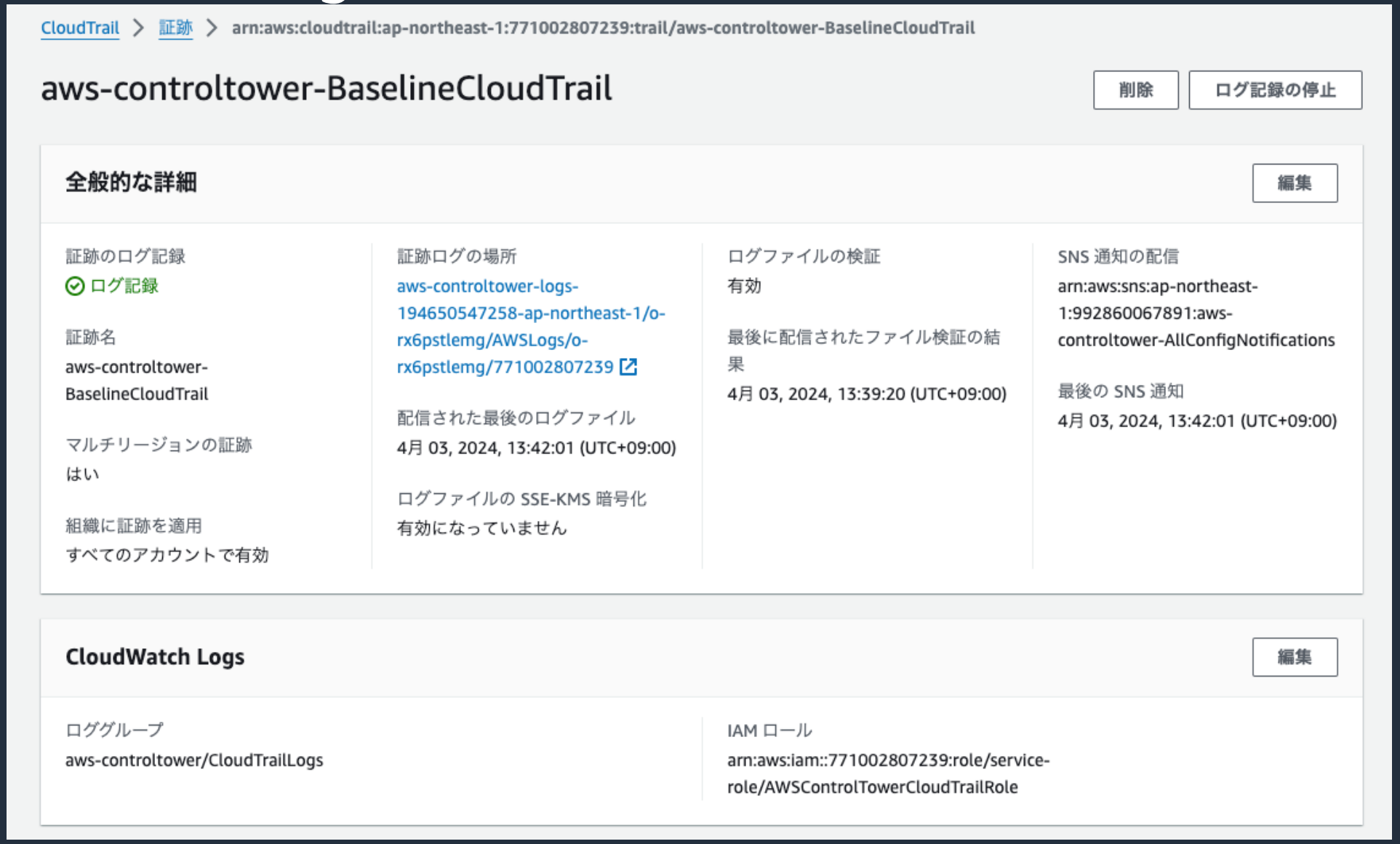
Security Hub – Organization view



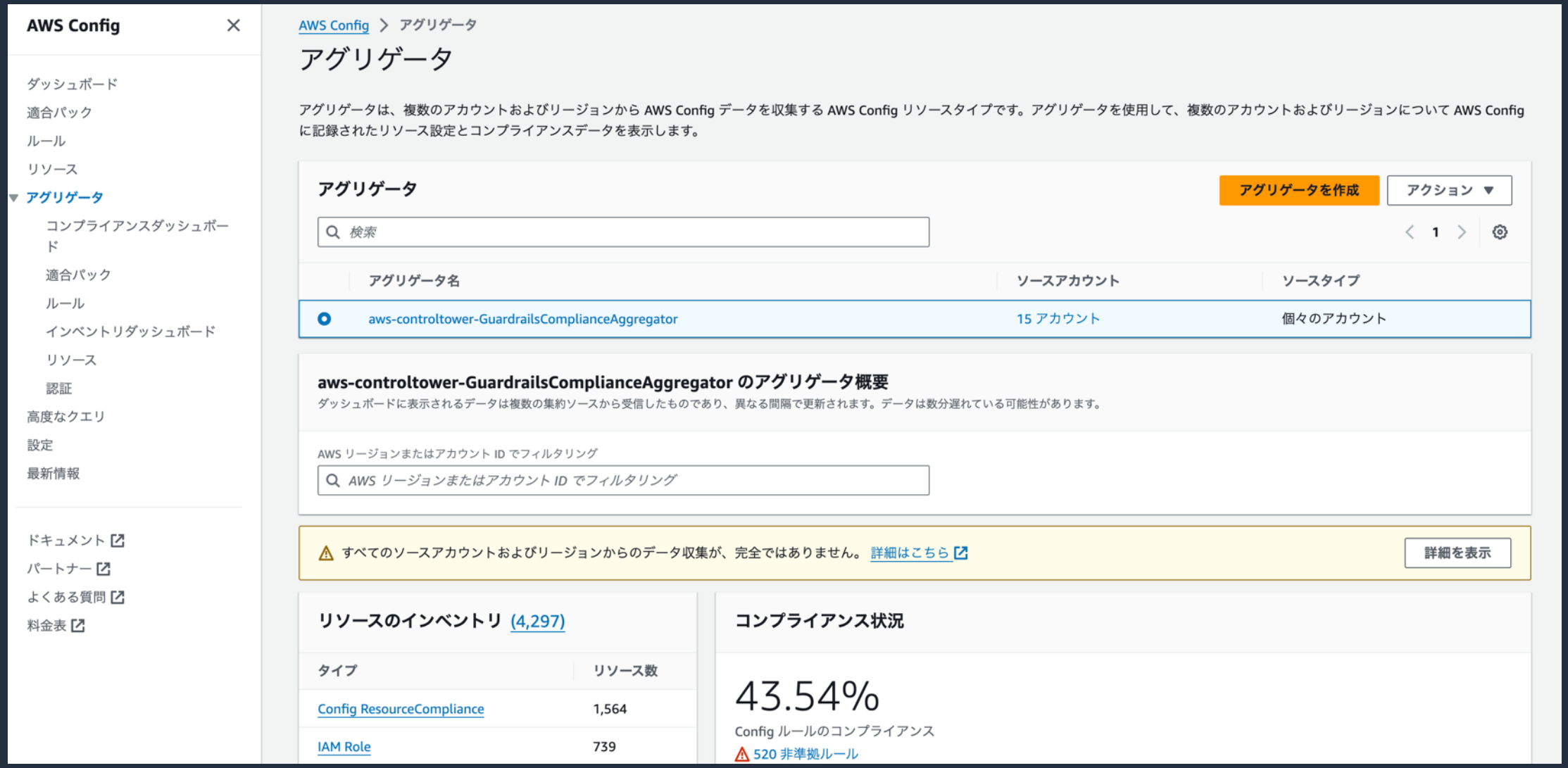
IAM Access Analyzer – Organization activity



CloudTrail – Organization Trail



AWS Config – Organizations Aggregator

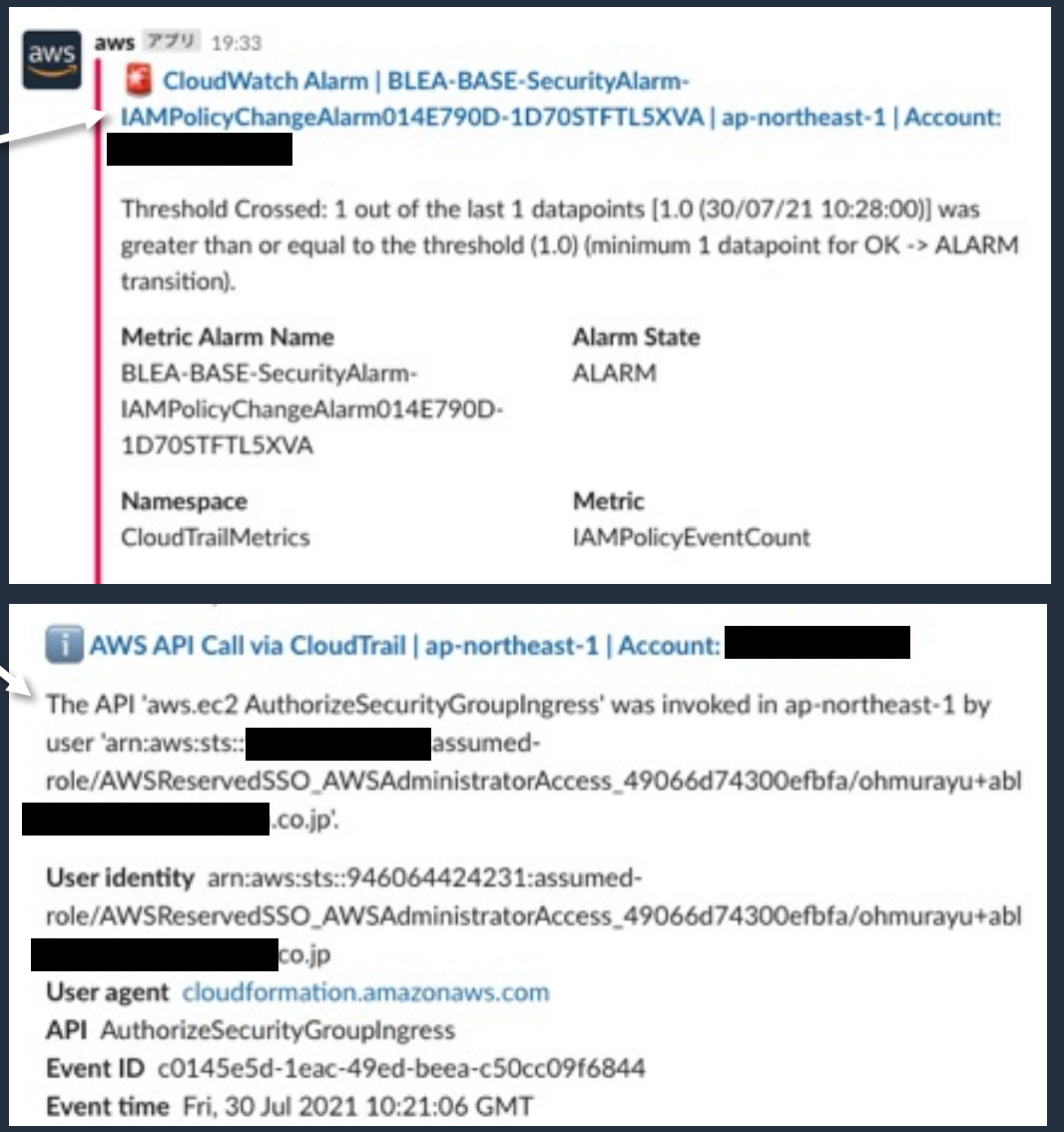


BLEA セキュリティ通知　1) 注意が必要な操作

BLEAベースラインにより以下の
注意が必要な操作を検知

- IAM Policyの変更
- セキュリティグループの変更
- NeworkACLの変更
- CloudTrailの設定変更
- API認証エラー
- アクセスキーの作成
- Root ユーザーによる操作

Slackへの通知例

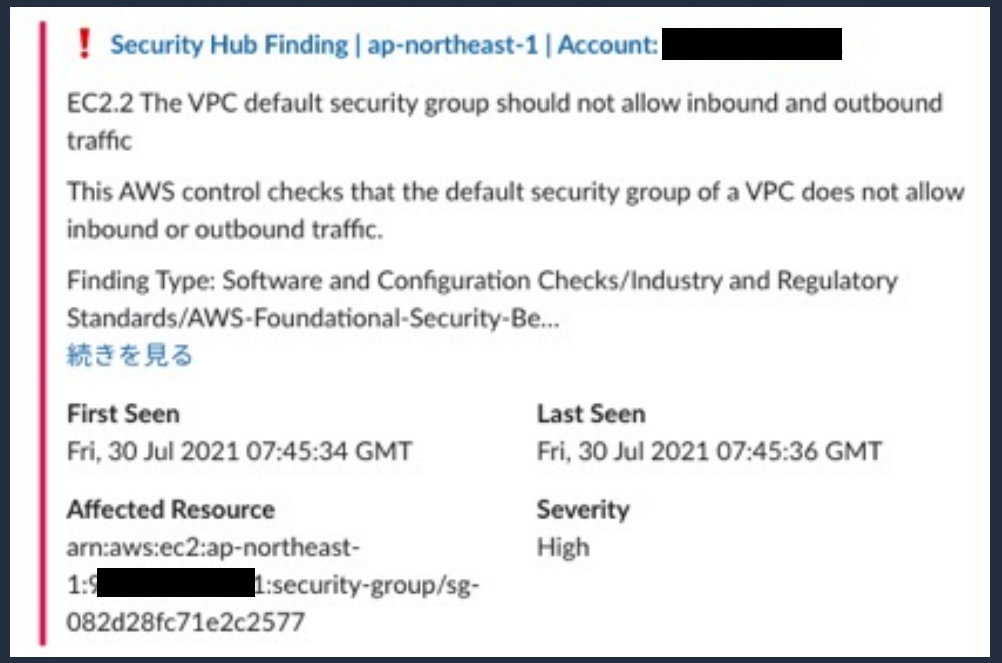


BLEA セキュリティ通知　2) セキュリティサービスの通知

BLEAベースラインにより以下の
AWSセキュリティサービス通知を受ける

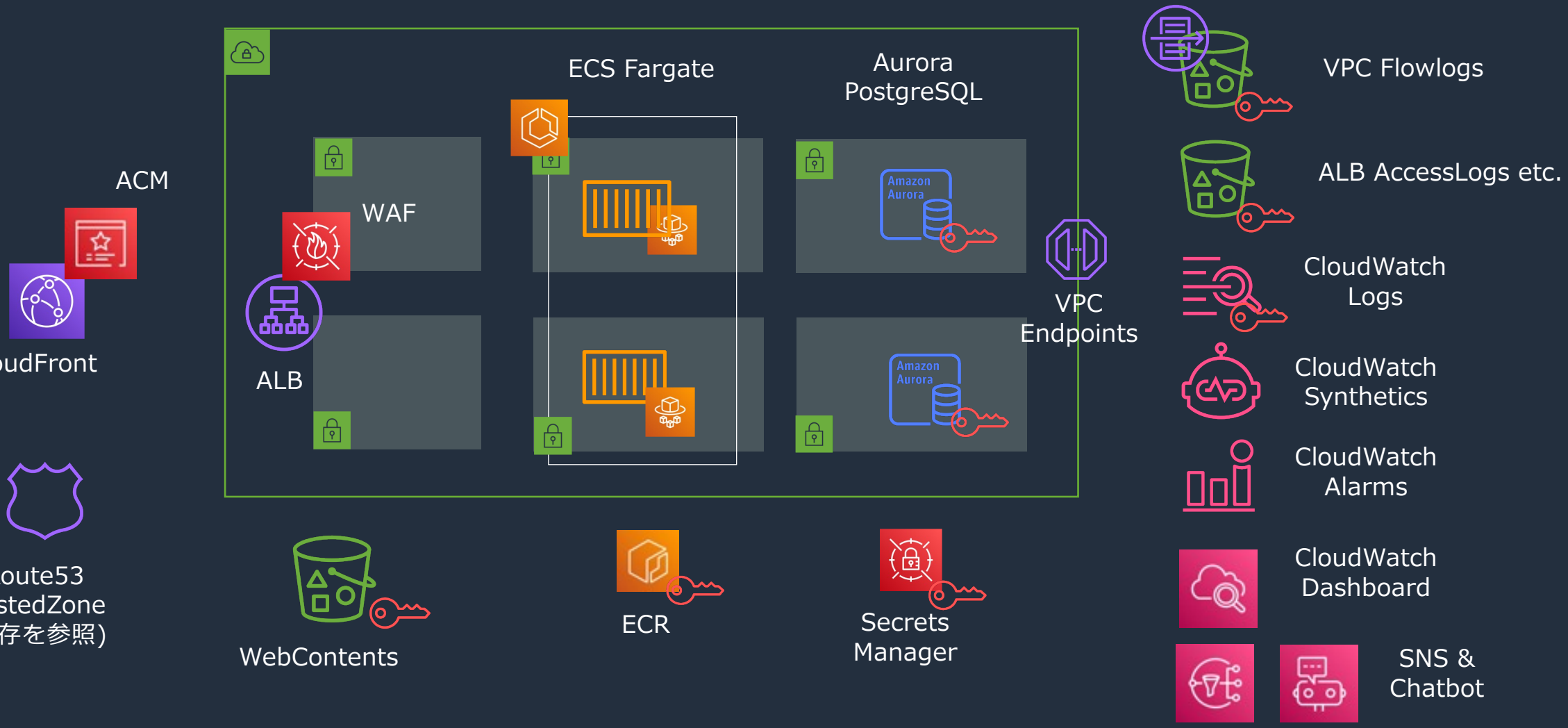
- SecurityHub
  - CIS Benchmark および
    AWS Security Foundational Best Practices
    の Critical/Highレベル検知
- GuardDuty
  - Medium～Highの検出結果
- AWS Health
  - 稼働中リソースに影響する
    AWSイベントの発生

Slackへの通知例

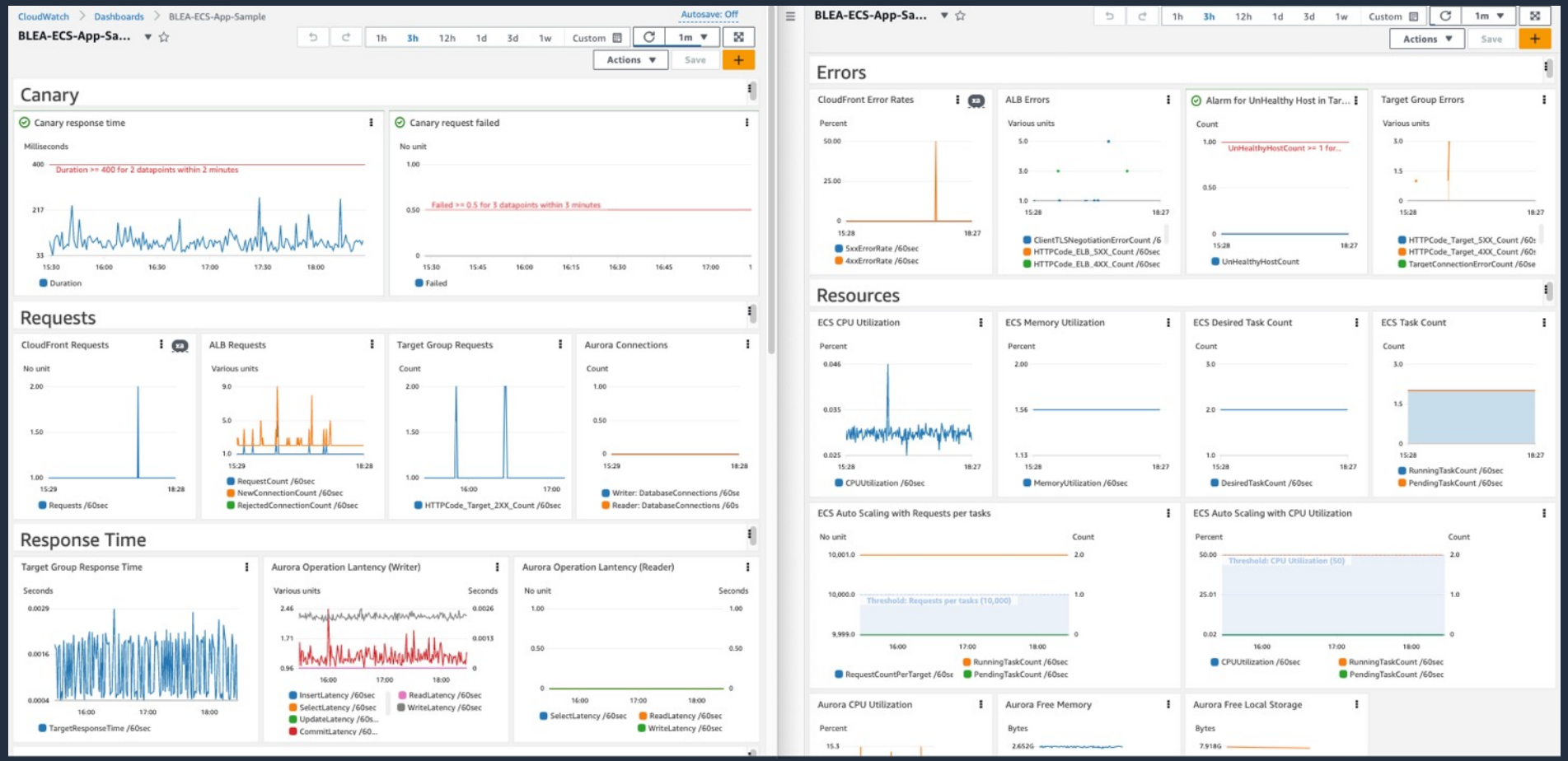


BLEA ゲストシステム例： Webアプリケーション (ECS+SSL)

(ver. 2021/10/26)

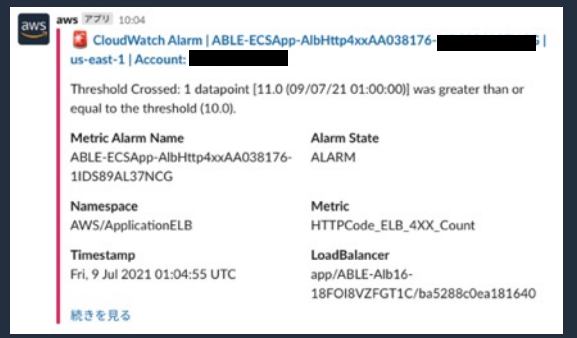


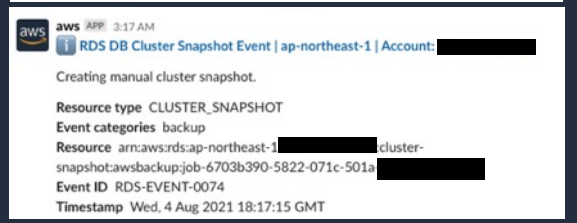
モニタリングアラートの通知とダッシュボード



サンプルアプリケーションの通知実装

- ECS/Fargate Service 平均CPU使用率
- ECS/Fargate Service タスク数
- ALB レスポンスタイム
- ALB HTTP 4XXエラー数
- ALB HTTP 5XXエラー数
- ALB HealthyHost数
- Aurora CPU使用率
- RDS イベント



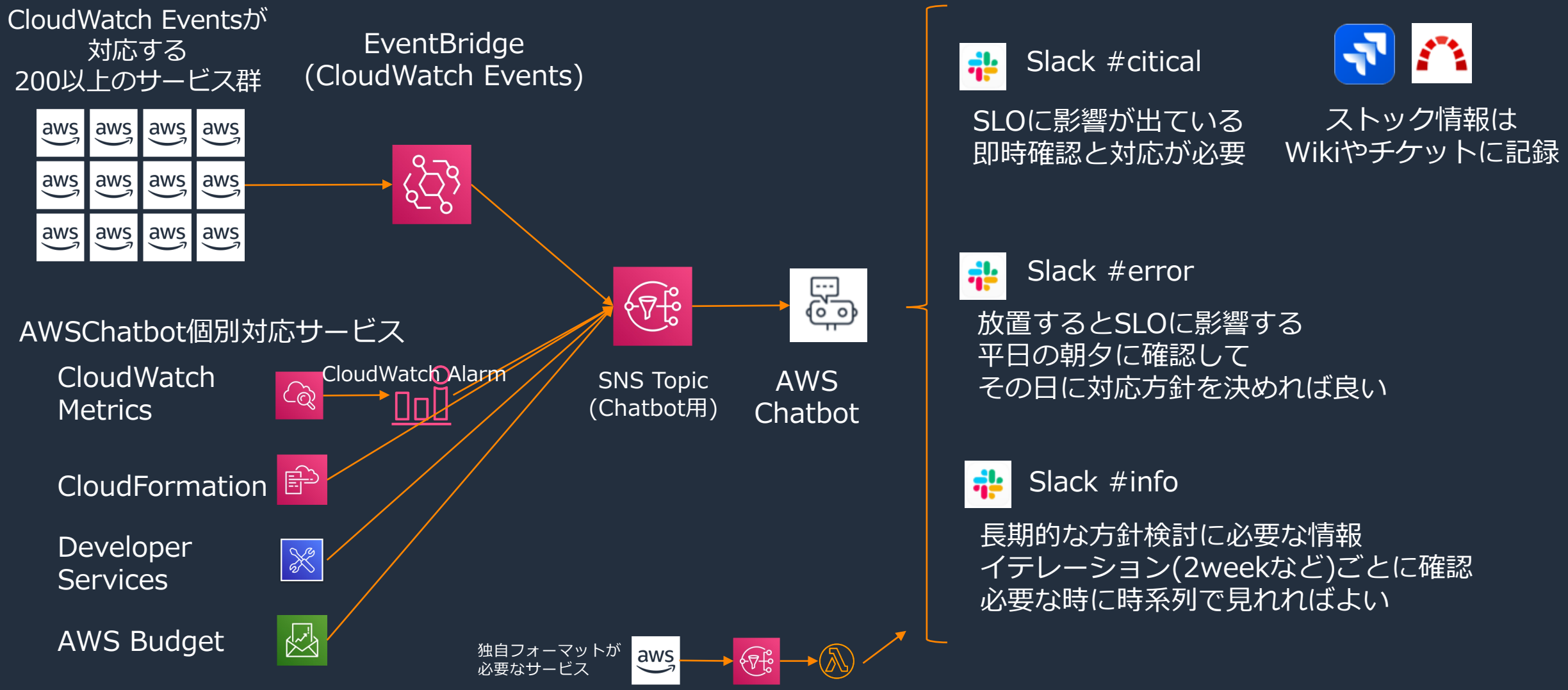


※リソースの変更にCloudWatchダッシュボードを追従させるのは IaCでないとかなり煩雑

ゲストシステムのセキュリティ実装

- アプリケーション用KMSキーによる各ストレージ暗号化（データ消去管理）
- ALBアクセスログバケットの暗号化（SSEのみ）と権限設定の最小化
- Webコンテンツバケット暗号化（SSEのみ）
- AWS外部で取得した独自ドメインによるHTTPS通信
- ECS FargateクラスタをProtected サブネットに配置
- ECRリポジトリのプライベートアクセス（Pull through cache 対応）とイメージスキャン有効化
- ECSサービスおよびデプロイメントイベントの通知
- Aurora クラスタおよびインスタンスの重要イベント通知
- Aurora 監査ログの出力
- Aurora ログインパスワードのSecret Manager登録
- Aurora Performance Insightの有効化とデータの暗号化

参考：Chatへのフロー情報集約の例



ベースラインの展開パターン

ベースラインの展開パターン

A. テンプレート配布方式（Pull型）

- テンプレートのみを提供。開発・運用チームがコードから展開する

B. サービスカタログ方式（Pull型）

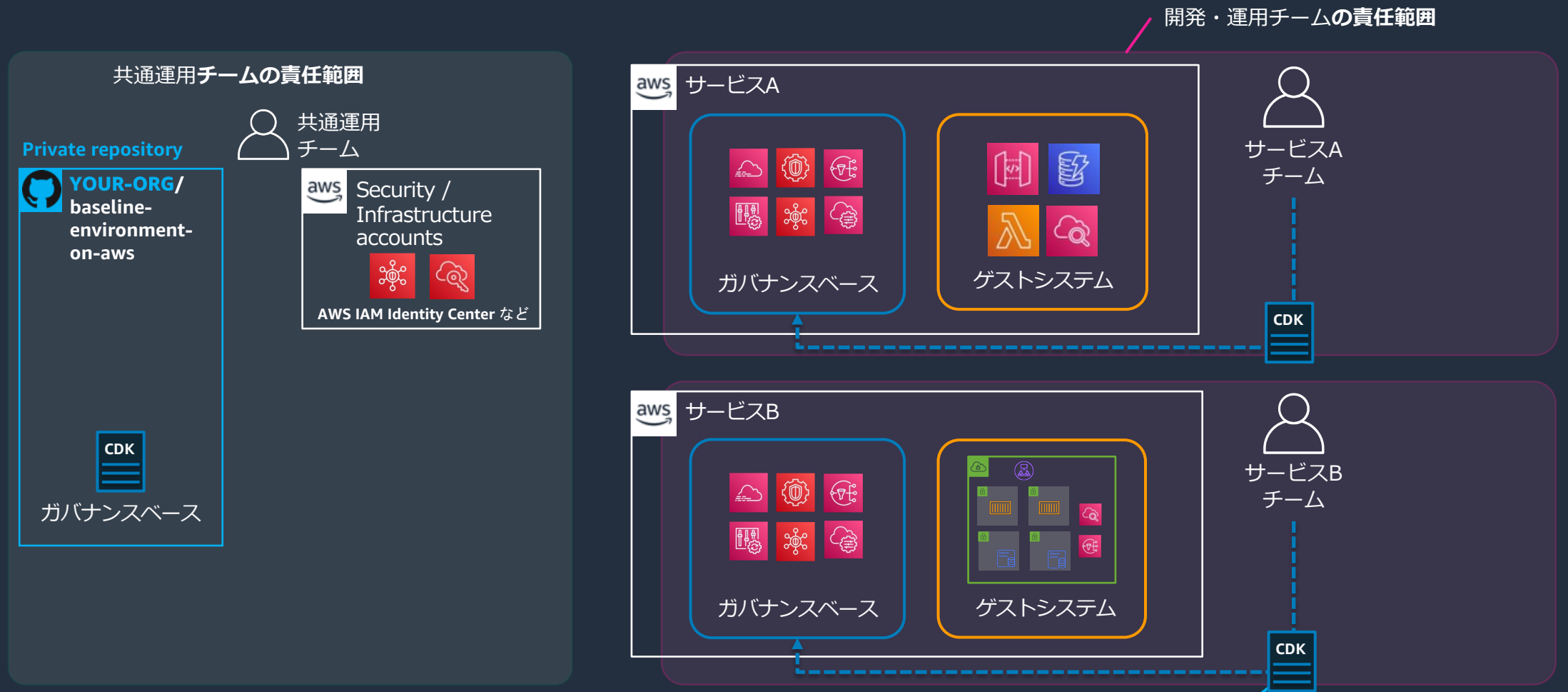
- Service Catalogを提供。開発・運用チームが自らデプロイする

C. AFC方式（Push型）

- 中央集権で共通運用チームがベースラインをデプロイ・管理する

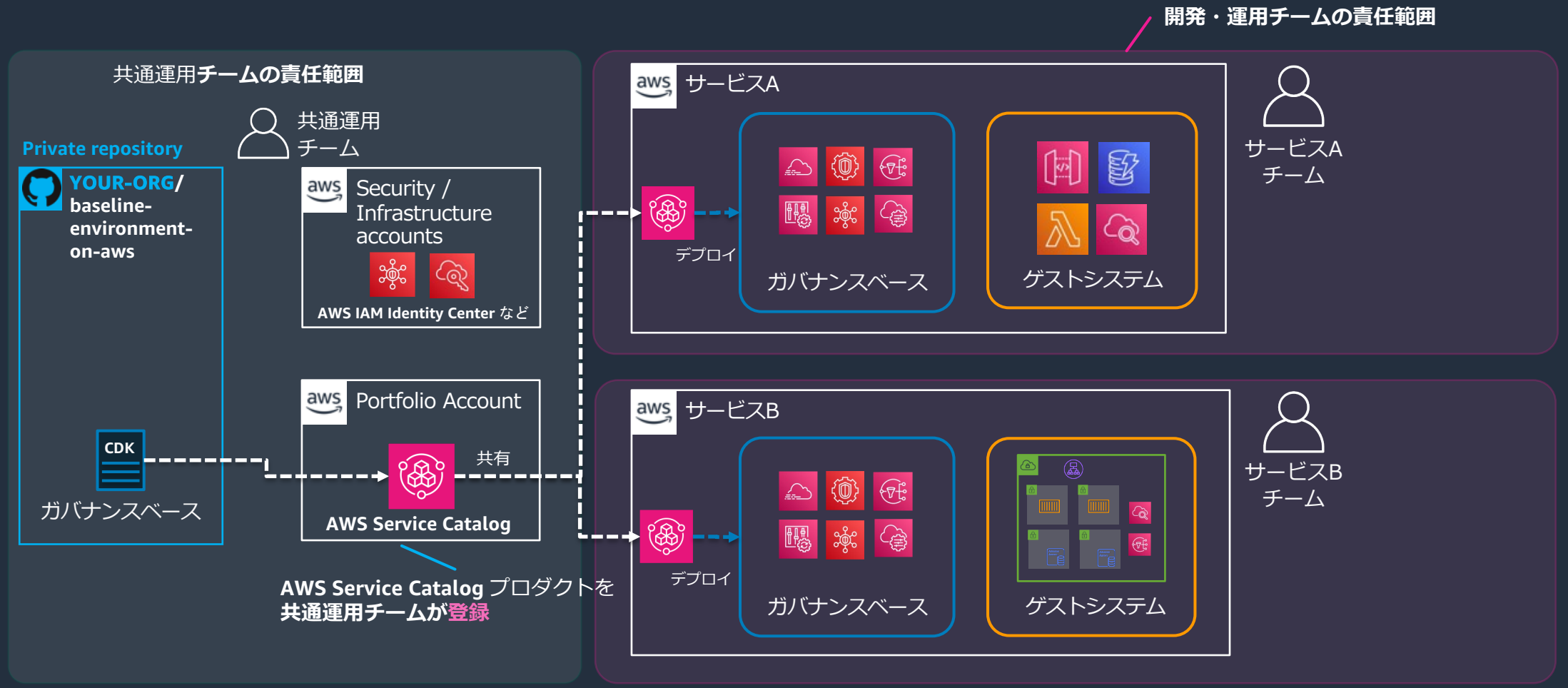
ベースラインの展開パターン A. テンプレート配布方式

- ガバナンスベースはテンプレート（コード）で提供し、開発・運用チーム自身がデプロイする基本の運用モデル
- 共通運用チームはガバナンスベースのコードをメンテナンス・配布する



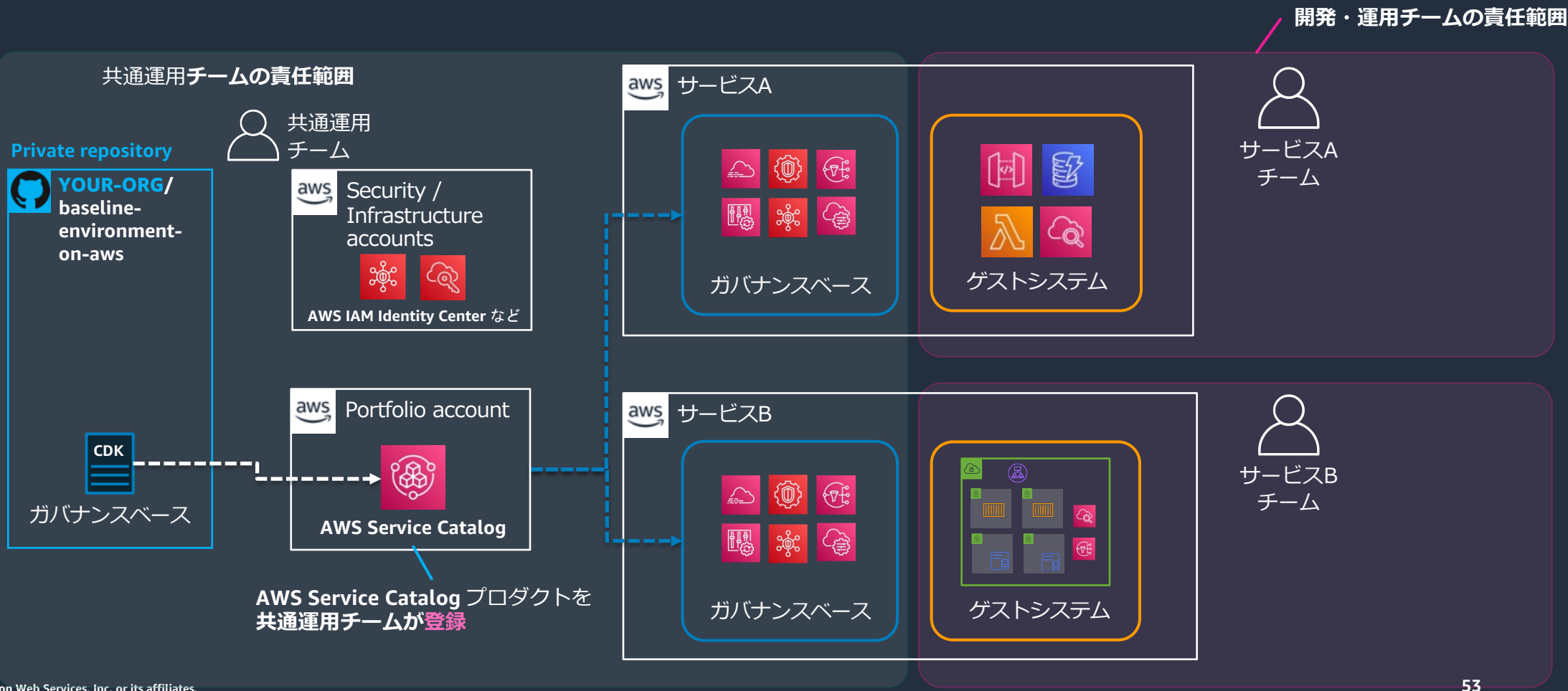
ベースラインの展開パターン B. サービスカタログ方式

- 共有されたService Catalogを使って、
  開発・運用チームがガバナンスベースを自分のアカウントへデプロイする（Pull型）



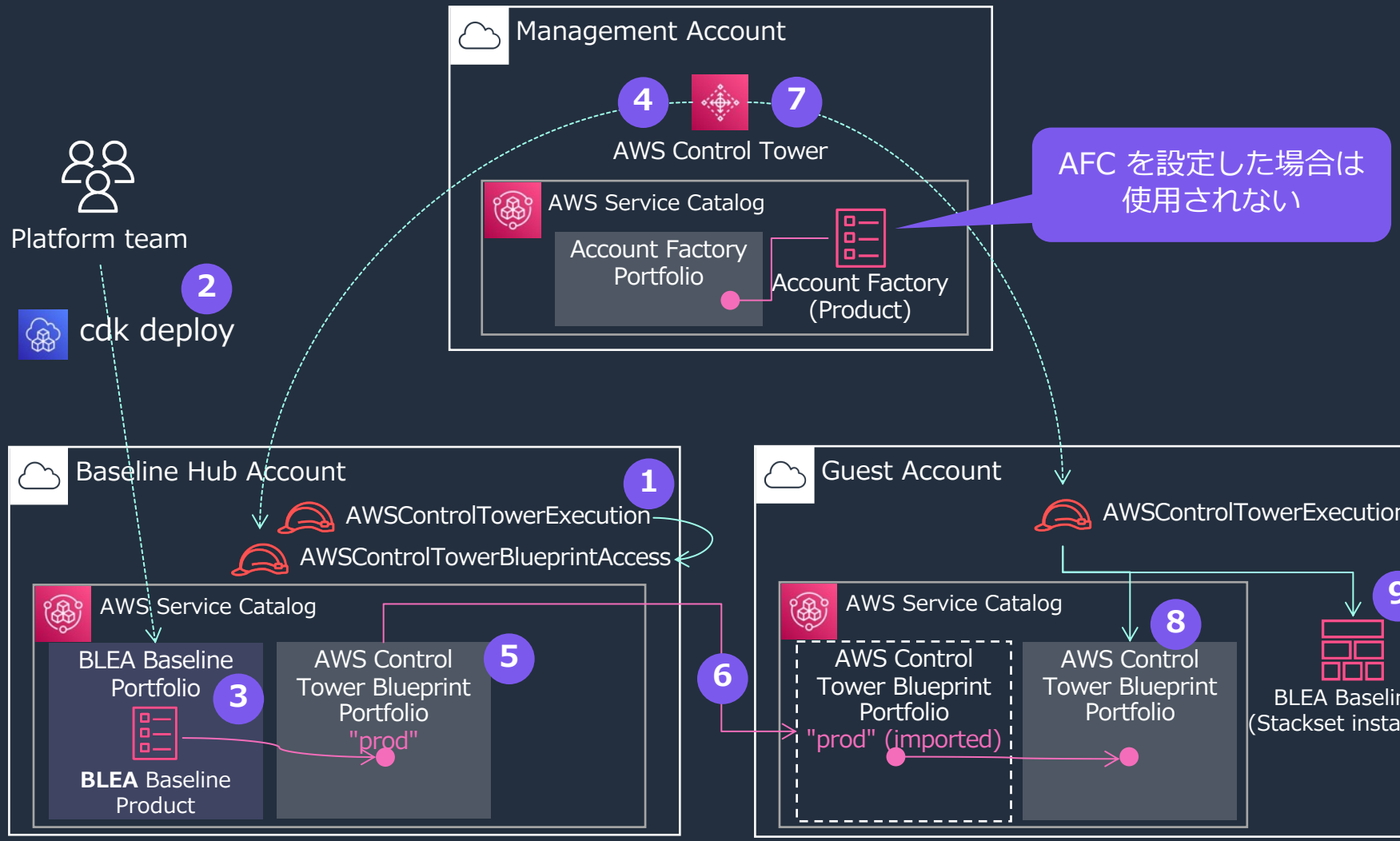
ベースラインの展開パターン C. AFC方式

- AWS Control Tower Account Factory Customization (AFC) でアカウント発行時に自動適用（Push型）
- ガバナンスベースの更新も、共通運用チームが Control Towerから実施（Push型）



参考：Account Factory Customization (AFC) の動作

* CT = AWS Control Tower



1. (事前準備) CT 用のロールを作成
2. CDK コマンドで ガバナンスベースをデプロイ
3. CDK から Service Catalog 製品と ポートフォリオを作成
4. CT でブループリントを設定して Account Factory を実行
5. CT 用のポートフォリオを作成し 指定された製品を**紐付け**
6. CT がポートフォリオを ゲストアカウントに**共有**
7. CT がゲストアカウントで ポートフォリオの共有を**受入れ**
8. ゲストアカウント用の ポートフォリオを作成して 共有済みの製品を**紐付け**
9. Service Catalog 製品を **プロビジョニング**（StackSet）

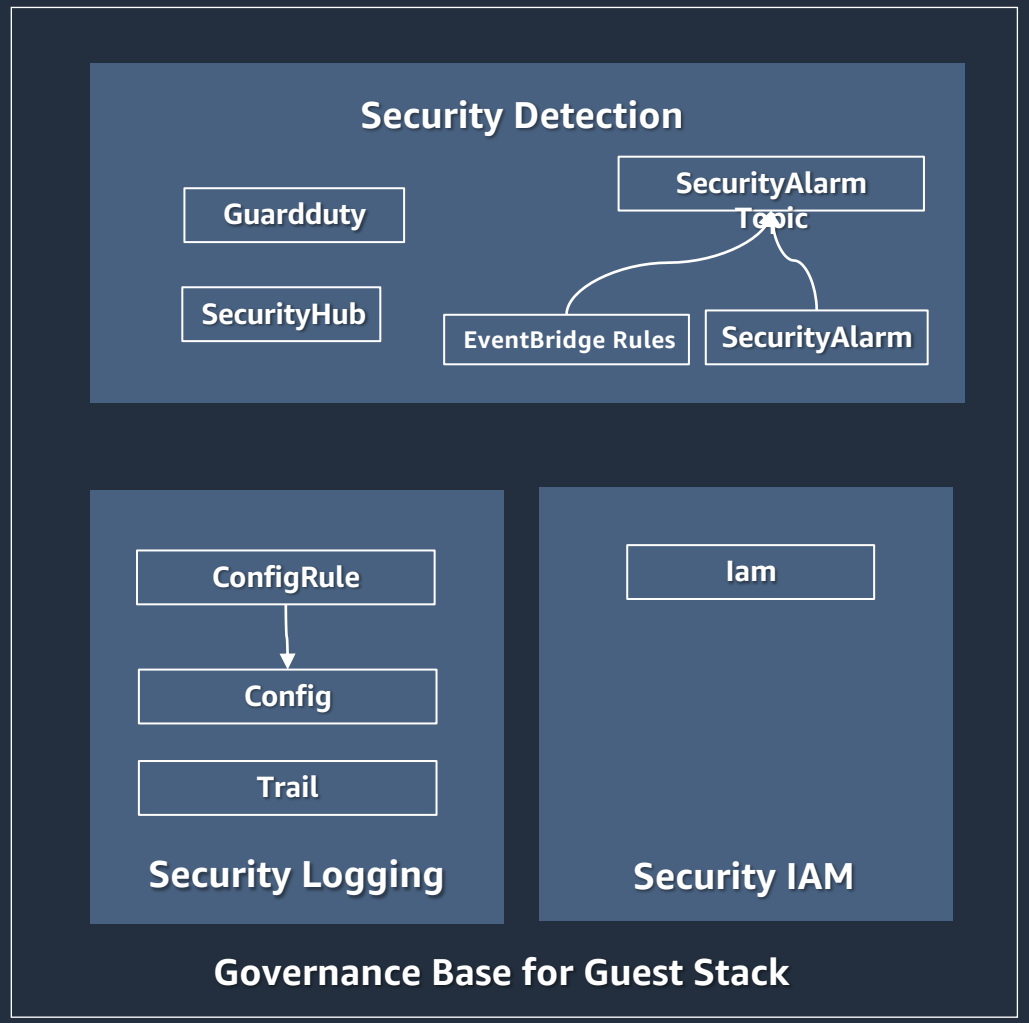
BLEA のカスタマイズ

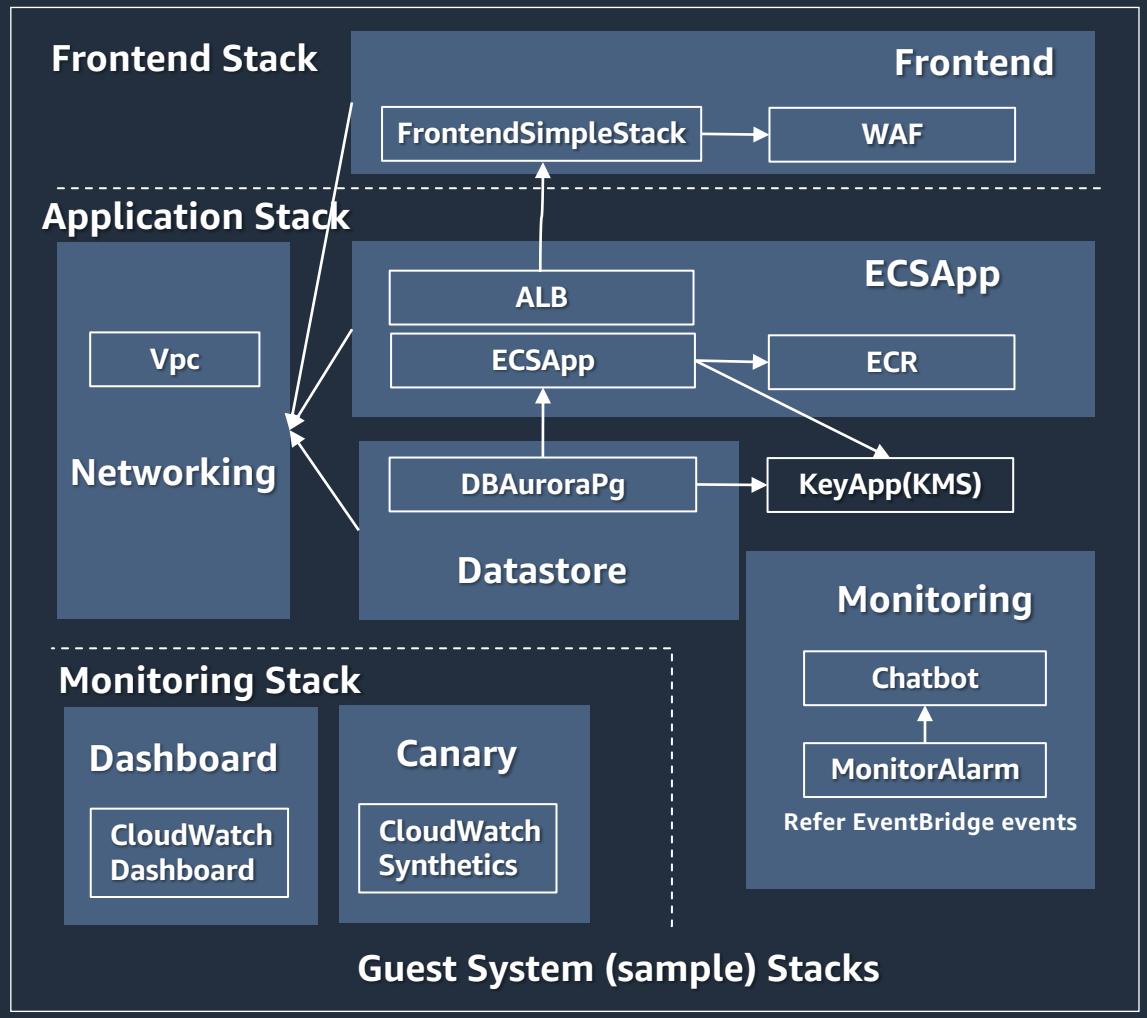
BLEAのカスタマイズガイド

- ガバナンスベースは基本的に修正不要
  - 最低限必要なセキュリティ設定を有効化
- ゲスト基盤は現状の設定をベースに必要な機能を追加する
  - セキュリティ要件に合わせてSecurityHubやAWSConfigのCompliance状態を確認
- ゲストシステムは用途に合わせて選択、カスタマイズが前提
  - よく使いそうなコード素材がサンプルの中に散りばめられている
  - コピーしてパラメータ変更でできる部分を多く
- フラットなコード構成
  - クラス化による隠蔽より、設定のわかりやすさを優先
  - 自社用には独自の仕様を埋め込んだクラスを作ってもよい
- サンプルがどこまで実装しているか
  - サンプルテンプレートは
    SecurityHubの「AWSのベストプラクティス」および「CISベンチマーク」で、
    「すぐに対処が必要 (CRITICAL)」「優先して対処が必要 (HIGH)」がないよう実装

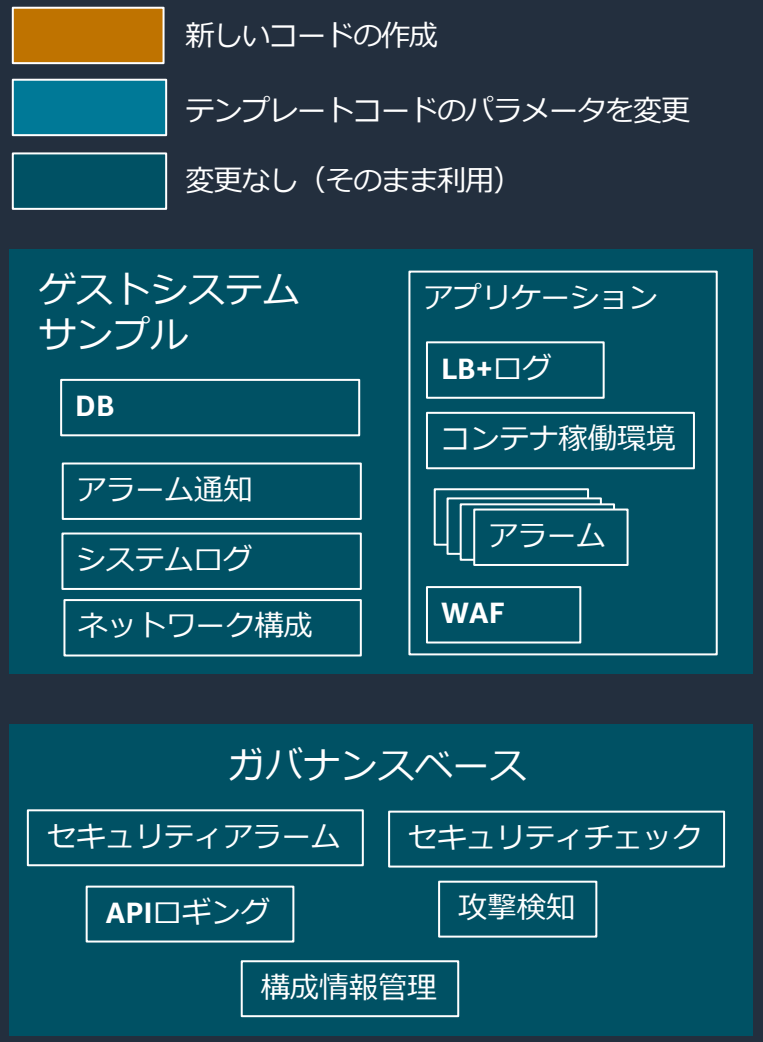
Baseline Environment on AWS – Component dependency

- Governance Base and Guest System can deploy independently



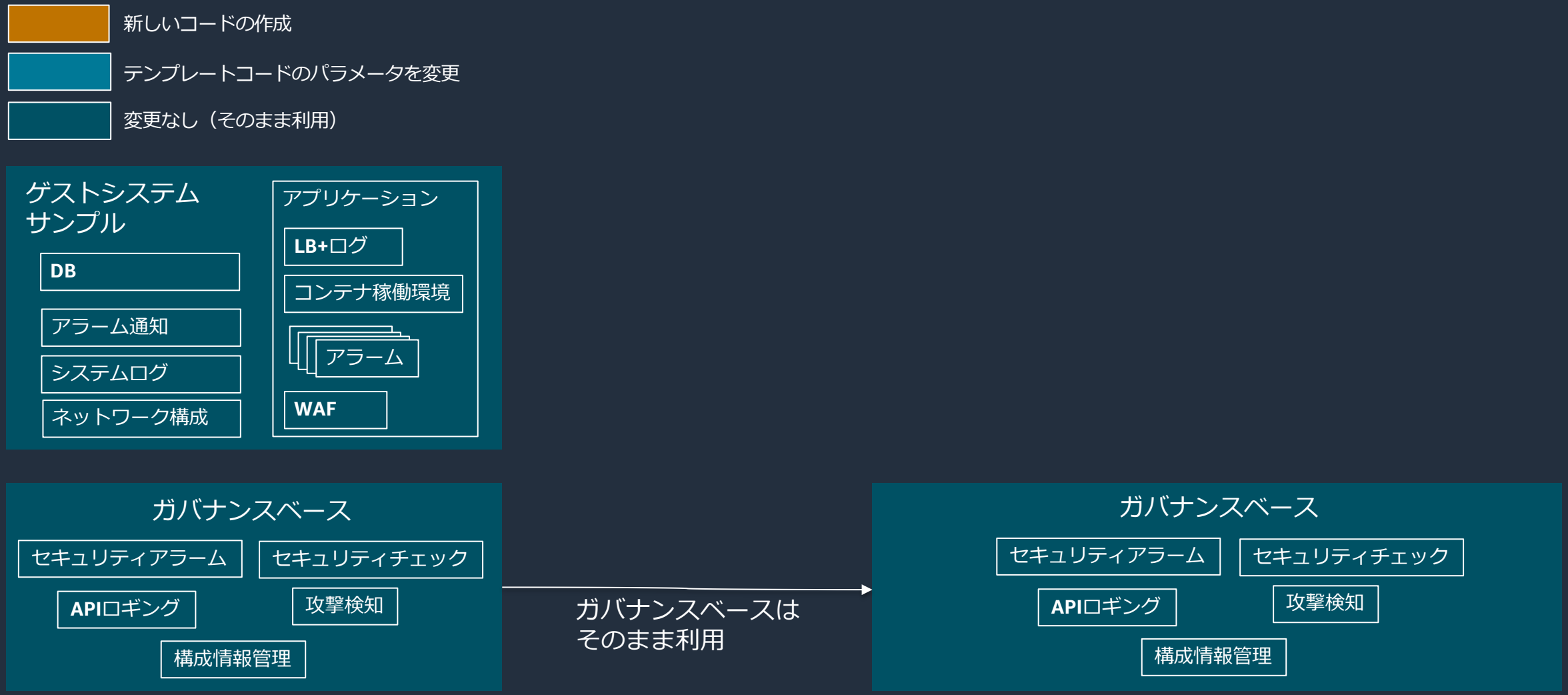


テンプレートを利用した開発の例

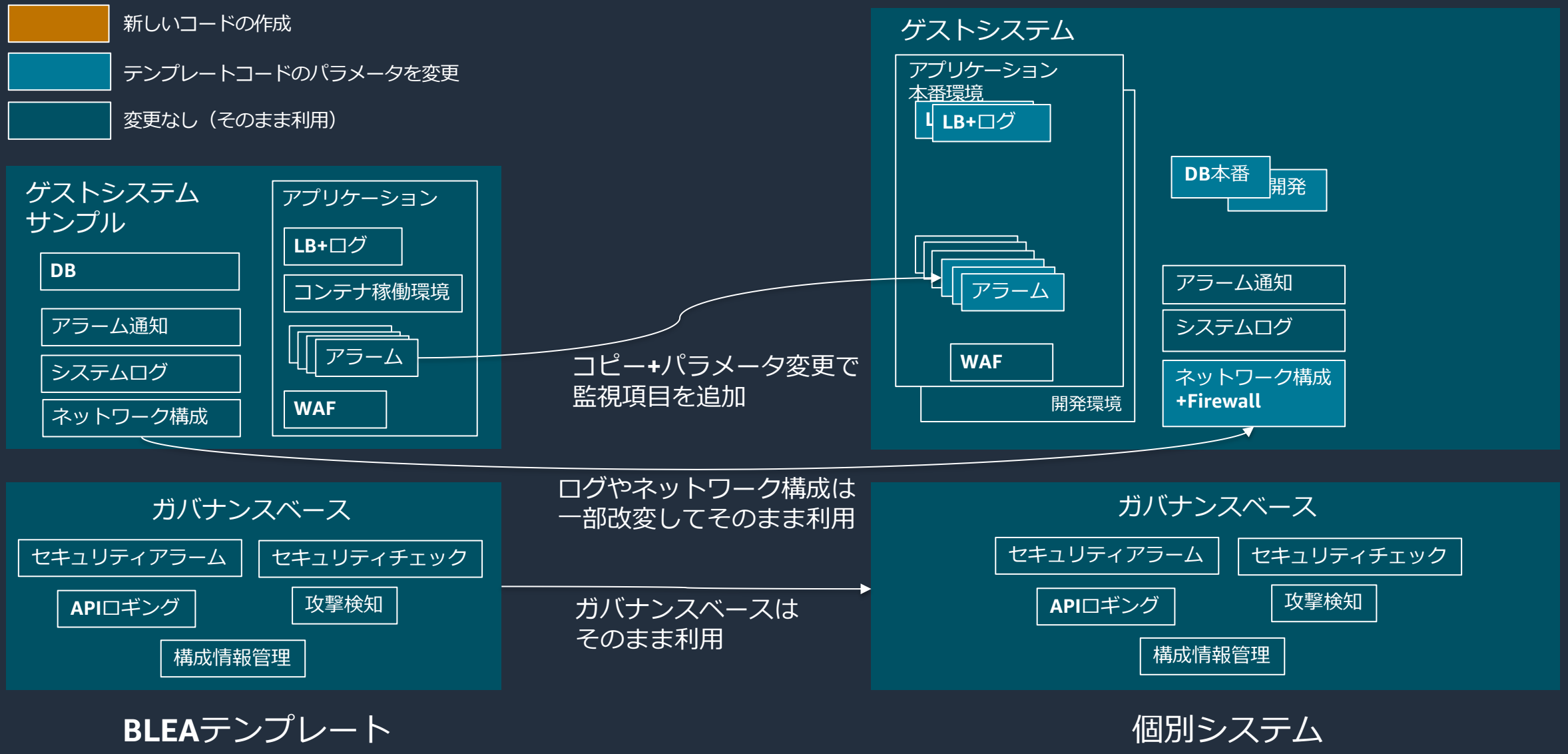


BLEAテンプレート

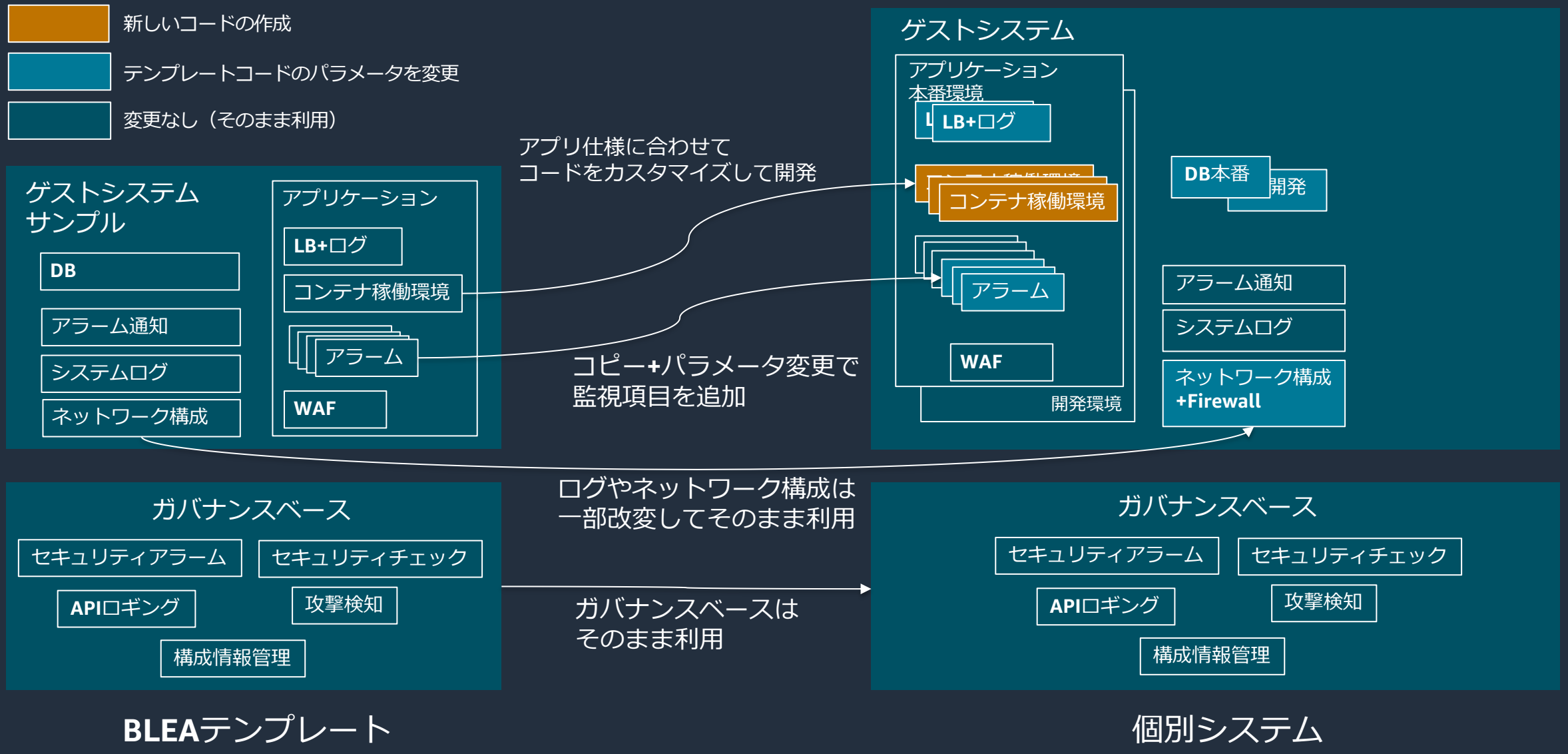
テンプレートを利用した開発の例



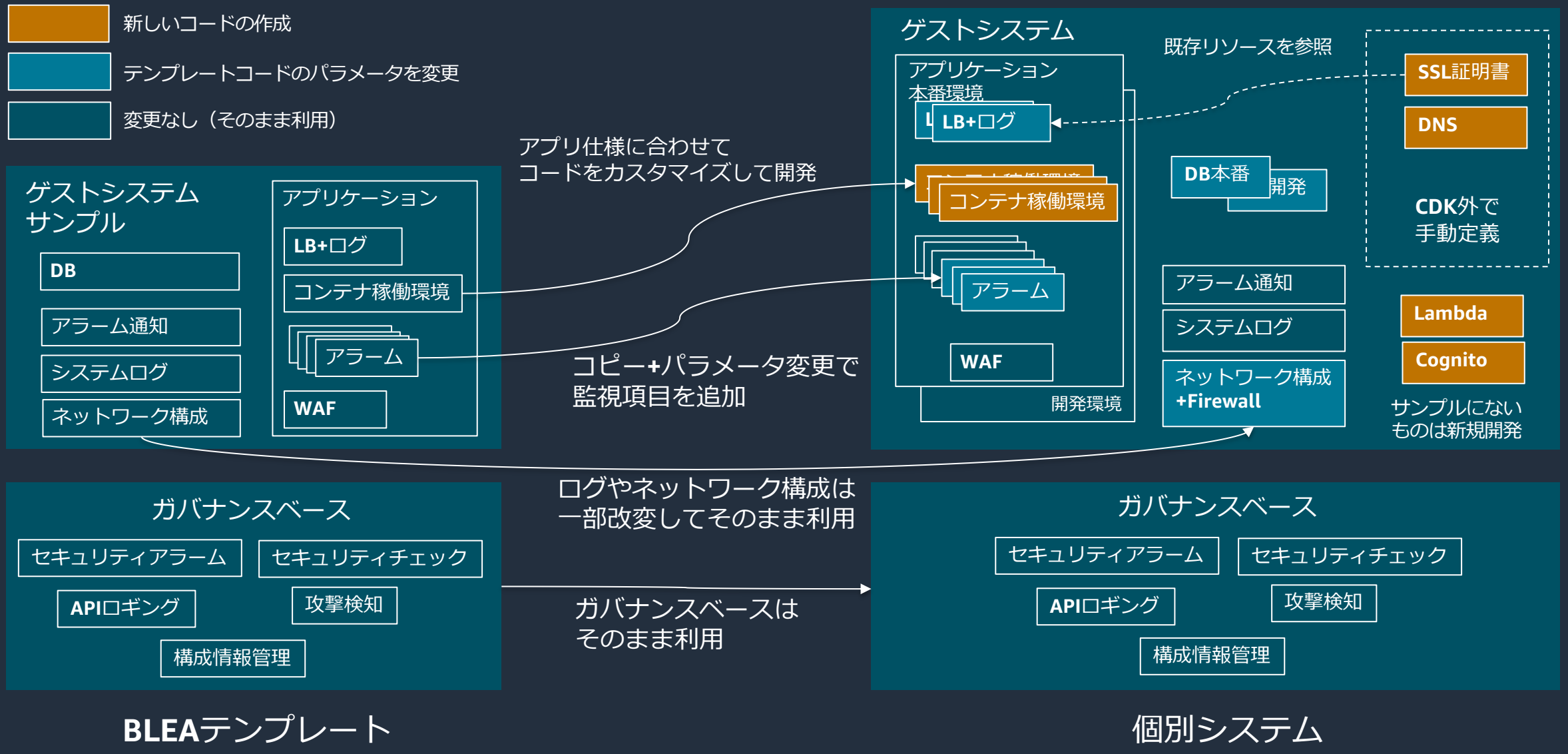
テンプレートを利用した開発の例



テンプレートを利用した開発の例



テンプレートを利用した開発の例



参考資料

BLEA（関連）公開資料

- BLEA
  - https://github.com/aws-samples/baseline-environment-on-aws

- AWS資料
  - テンプレートによるAWS環境のガバナンス（AWS Summit 2022 発表資料）
  - BLEA 開発チームが学んだ AWS CDKの開発プラクティス (CDK Conference 2023 発表資料)

- ユーザー様資料
  - みずほリサーチ＆テクノロジーズが AWS CDK で実装したマルチアカウント管理の仕組み
  - オンプレシステムのクラウドリフトにおいてBLEAを利用してセキュリティのベースラインを確保した話

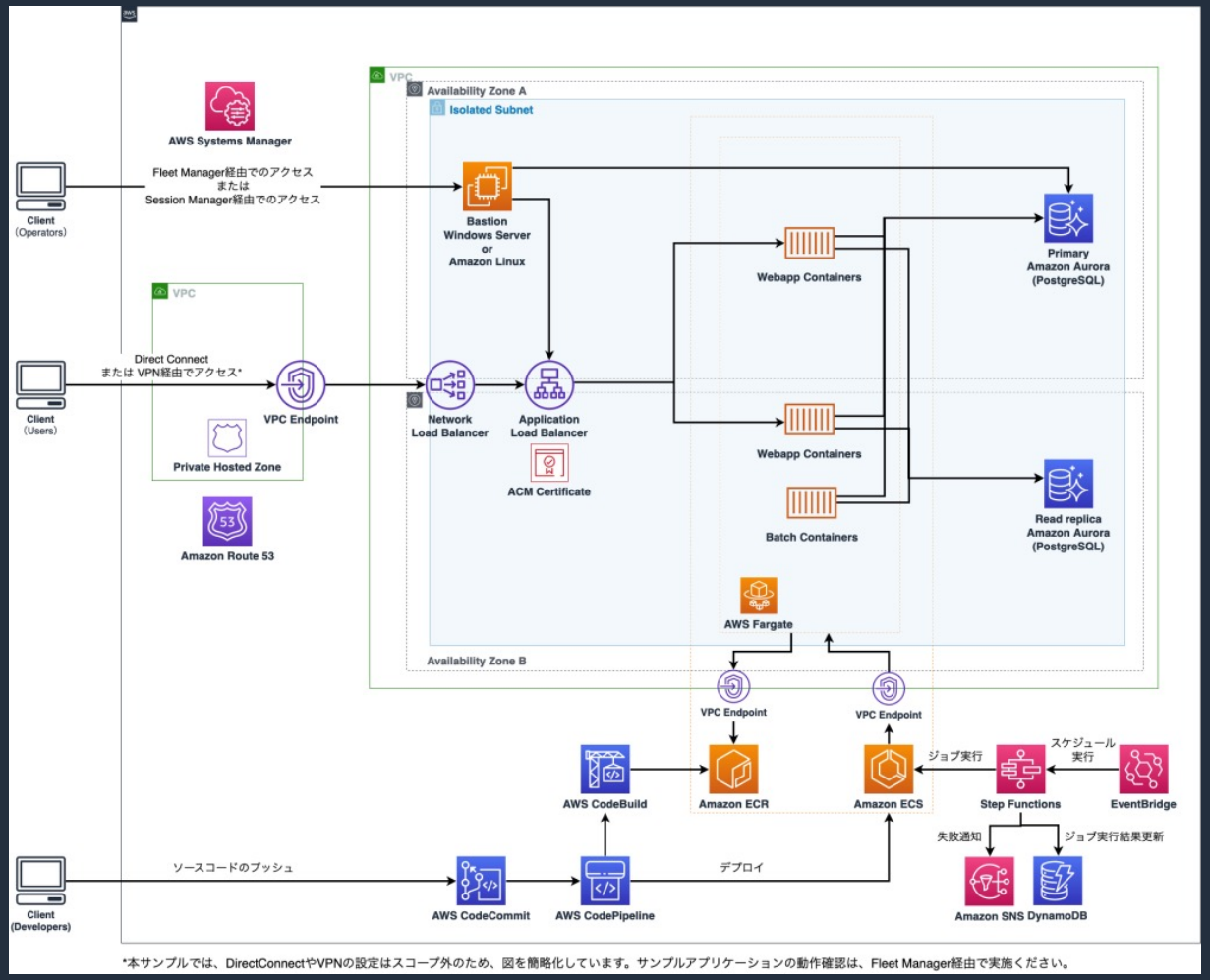
閉域網での利用を前提としたCDKのサンプルテンプレート

Amazon Web Services ブログ

閉域網での利用を前提としたCDKのサンプルテンプレートを公開しました

by Yozo Suzuki | on 10 3月 2023 | in General, Government, Public Sector, State Or Local Government, Technical How-To | Permalink | Share

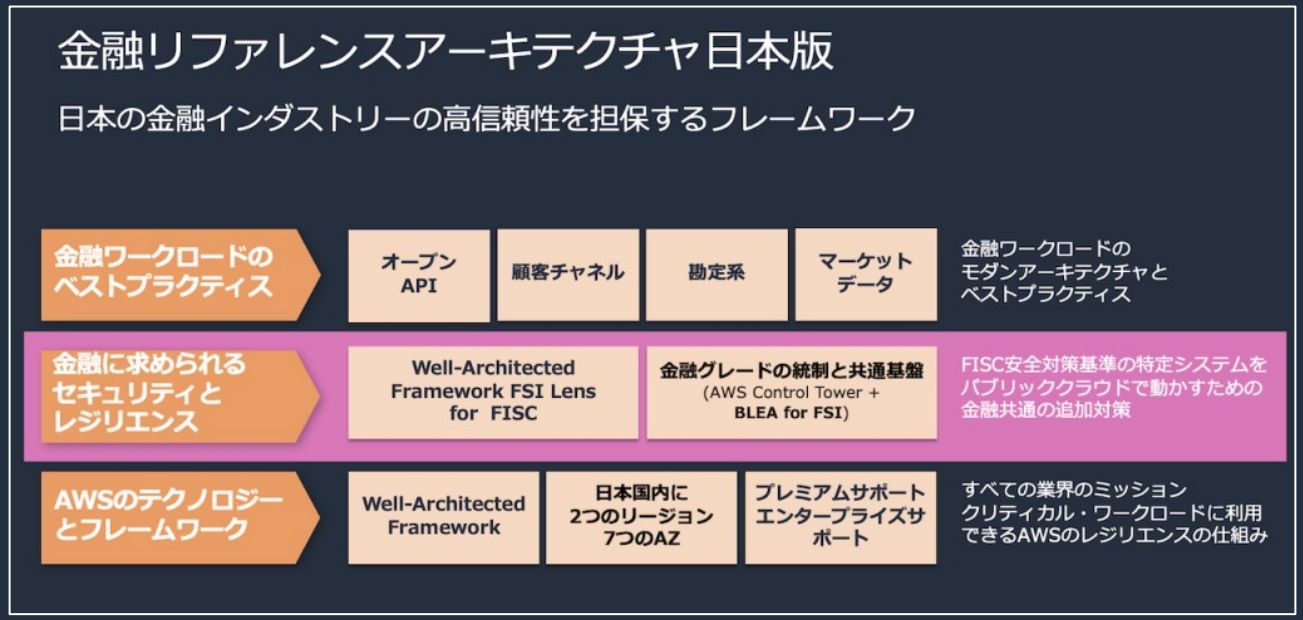
こんにちは、公共部門でプロトタイプSAをしている鈴木です。
デジタル庁が整備するガバメントクラウドではAWSが採択されており、中央省庁や地方自治体等でAWSをご利用いただくお客さまが増えてきました。
このブログではこれからAWSを利用し始めるお客さまやアプリケーションのモダナイズをご検討されているお客さま向けに開発した、閉域網での利用を前提としたサンプルテンプレートについてご紹介します。



BLEAのゲストシステムサンプルとして利用可能

https://aws.amazon.com/jp/blogs/news/announcing-template-for-closed-network-system-workloads-on-aws/
https://github.com/aws-samples/template-for-closed-network-system-workloads-on-aws

金融リファレンスアーキテクチャ



BaseLine Environment on AWS for Financial Services Institute (BLEA for FSI)

BLEA for FSI は BLEA をベースとして、FISC 準拠が求められる 金融グレードの統制と共通基盤のベースラインとなる CDK のテンプレートを提供します。マルチアカウント AWS 環境をセキュアに構築・管理するための AWS Control Tower環境を前提とし、ゲストAWSアカウントに FISC 実務基準に対応したガードレールを組み込むことを強力にサポートします。BLEA for FSI は共通の統制基盤を構築する Control Tower ベースのガバナンスベースと金融ワークロード サンプルアプリケーションから構成されます。※ BLEA for FSI で提供するCDK コードはサンプル実装となるため、実際の構築・運用にあたってはお客様側でのカスタマイズおよび検証が必要となります。

https://aws.amazon.com/jp/blogs/news/financereferencearchitectureproductionseminar202210/

https://github.com/aws-samples/baseline-environment-on-aws-for-financial-services-institute

日本が目指すデジタル社会の姿と、それを実現するために必要な考え方と取り組みについて(SP-01)

ガバメントクラウド テックブログ

♡ 76

デジタル庁 ガバメントクラウド
2022年1月24日 17:11

「政府共通のクラウドサービスの利用環境です。クラウドサービスの利点を最大限に活用することで、**迅速**、**柔軟**、かつ**セキュア**で**コスト効率**の高いシステムを構築可能とし、利用者にとって利便性の高いサービスをいち早く提供し改善してくことを目指します」

- **Cost（コスト効率、最適化）**：投資対効率
- **Elasticity（弾力性）**：伸張性、拡張性を支える性能
- **Performance（パフォーマンス）**：システム全体のパフォーマンス
- **Agility（俊敏性）**：変化への追従性
- **Velocity（ベロシティ）**：実行の加速度
- **Security（セキュリティ）**：セキュリティ性能
- **Resiliency（レジリエンシー）**：復旧や回復性能
- **Observability（可観測性）**：システムの内部状況の把握性能
- **Transparency（透明性）**：目的に対する貢献度合いの可視化性
- **Improvability（改善性）**：改善のしやすさ
- **Automation（自動化）**：自動化のしやすさ

「政府情報システムにおける
クラウドサービスの適切な利用に係る基本方針」
の改定について

2022年5月24日
ガバメントクラウドチーム

参考
- ガバメントクラウド テックブログ
- 「政府情報システムにおける クラウドサービスの適切な利用に係る基本方針」 の改定について

改定の概要

- 旧方針（2018年6月に初版）は、クラウドファースト（先ずはクラウドの利用を検討する）だったが、本方針では**クラウドスマート**（クラウドを賢く適切に利用する）を目的とする。
- スマートとは**モダン技術**の利用であり「**マネージドサービス**」と「**IaC** (Infrastructure as Code)」が中心。

「3.4　アプリケーションとシステム刷新」より抜粋

- アプリケーションとインフラを分離した調達は、アプリケーションのモダン化とスマートなクラウド利用を阻害する要因となるため、クラウドでは見直しが必要となる。（中略）クラウド移行に向けた刷新においては、**インフラとアプリケーションを同時に刷新する**ことが合理的である。また、事業者や調達についても**インフラとアプリケーションを原則として分離するべきではない。**
- オンプレミスにおけるアプリケーションとクラウド上のアプリケーションでは、以下で大きく異なるので、新規開発時やアプリケーション刷新時には特に留意されたい。
  - **モダンアプリケーションとする**
  - **オンプレミス時代の旧来技術・運用を単純に踏襲しない**
  - **オンプレミス時代の人海戦術的な方式を踏襲せず自動化する**
  - **単なるシステム監視ではなく定量的計測を行う**
  - **セキュリティ対策もクラウドに最適化させる**
  - **開発プロセスをクラウドに最適化させる**
  - **稼働日で完成ではなく日々の運用で改善していく**

ガバメントクラウドにおけるIaC(Infrastructure as Code)の考え方

96

デジタル庁 ガバメントクラウド
2022年1月24日 17:12

デジタル庁クラウドチーム

Cloud Architect 山本教仁

ガバメントクラウドでは、コスト効率、迅速性、柔軟性、セキュリティを基本的指標にしています。第1回では、この指標を実現するための技術要素の1つであるInfrastructure as Code（以下、IaC）についてガバメントクラウドにおける考え方を説明します。

IaCのメリット

IaCのコードはいったん作成すればそれを何回でも別の環境でも流用可能です。典型的なインフラ構成をコード記述し、それをマスターテンプレートとして使い回すことで、(1)インフラ構成を迅速に作れ、(2)設計工数と期間を短縮することでインフラ構築コスト削減につながり、(3)手元のコードに記述された内容がインフラ構成のすべてなのでブラックボックス化を防ぎ頻繁なプリケーションの改善リリースに対しても柔軟に対応でき、(4)マスターテンプレートの一部を強制することでガバナンスを効かせてセキュリティレベルを保持でき、(5)IaCを徹底すればインフラ（OS）へのログインがなくなるためセキュリティ機能や運用もシンプルになる、という5つのメリットがあります。

https://cloud-gov.note.jp/n/na2ea9a24e3a1

Thank you!