AWS-22

テンプレートによる AWS 環境のガバナンス
～ Baseline Environment on AWS 徹底解説 ～

大村 幸敬
技術統括本部 シニアソリューションアーキテクト
アマゾン ウェブ サービス ジャパン合同会社

大村 幸敬（おおむら ゆきたか）

部長 / シニア ソリューションアーキテクト

- これからクラウドを使いはじめる
  エンタープライズ企業をサポート
- 運用系サービス & DevOps 系サービスをリード
- Baseline Environment on AWS (BLEA) 開発者

好きなAWSのサービス：
AWS Command Line Interface (CLI)
AWS Cloud Development Kit (CDK)
AWS Systems Manager Incident Manager

Agenda

1. クラウド活用に必要なガバナンスの考え方
2. テンプレートによるガバナンス
3. Baseline Environment on AWS (BLEA) の概要
    1. BLEA ベースライン
    2. BLEA ゲストシステムサンプル
    3. なぜ AWS CDK なのか？

セッション視聴にあたって

想定する視聴者

- クラウド環境全体の管理者
- 全社共通のセキュリティ確保とガバナンスを実現したい管理者
- AWS 上でセキュアなシステムを構築したい
  アプリケーション開発者やインフラエンジニア

想定する基礎知識

- AWS のセキュリティサービス群の基礎知識
- AWS CloudFormation や AWS CDK の基礎知識
- 「セキュアでスケーラブルなAWSアカウント統制プラクティス最新動向」（AWS-19）のセッション視聴をお勧めします

クラウド活用に必要な
ガバナンスの考え方

AWSはBuilderを支えるプラットフォーム
- Self Service Platform -

Builderに自由を与え、適切な箇所で適切なツールを使えるようにする
それによってビジネス価値を早期に実現できる

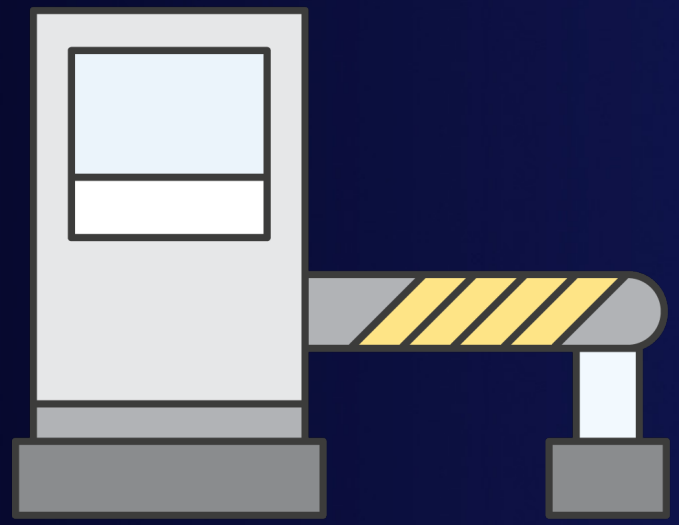
クラウドが提供する価値

- システム開発でビジネスにフォーカスできる
  - 多様かつ継続的に強化されるマネージドサービス
  - クラウドの高い可用性、セキュリティ、スケーラビリティを利用
  - ビジネスの差別化に繋がらない作業をオフロード

- サービス利用により開発および運用のコストと時間を節約できる
  - ソフトウェアのセットアップや運用にかかる要員と作業の低減
  - APIやテンプレートによる構築自動化
  - 使った分だけの課金で後から柔軟に構成変更が可能

Builderに必要なものは？

Gatekeeper

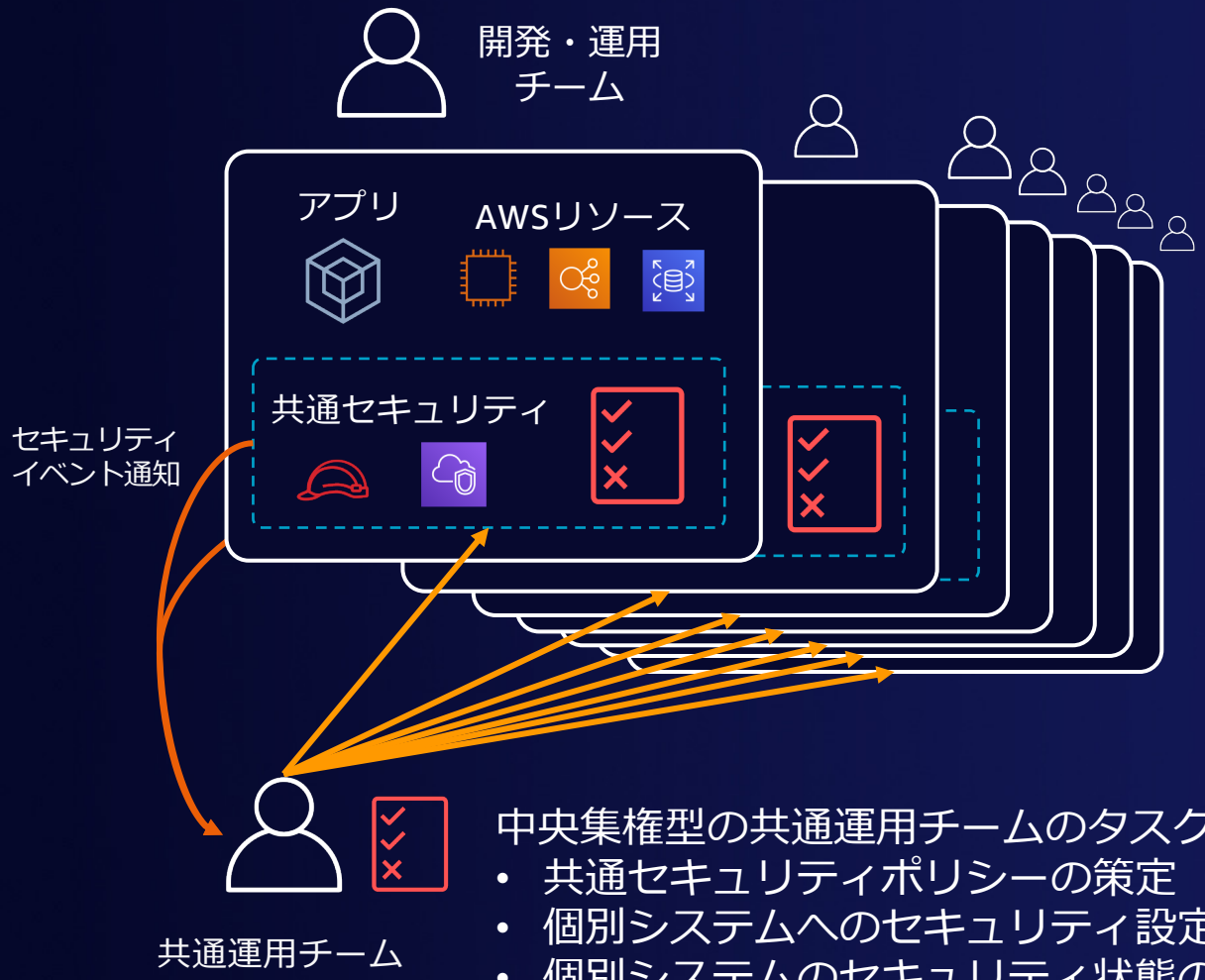


V.S.

Guardrail

中央集権型で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。
分散管理型として、Builderに自由に使わせてガードレール(Guardrail)を設置。
やってはいけない操作を未然に防ぐ予防的ガードレールと、逸脱を検知する発見的ガードレールが必要。

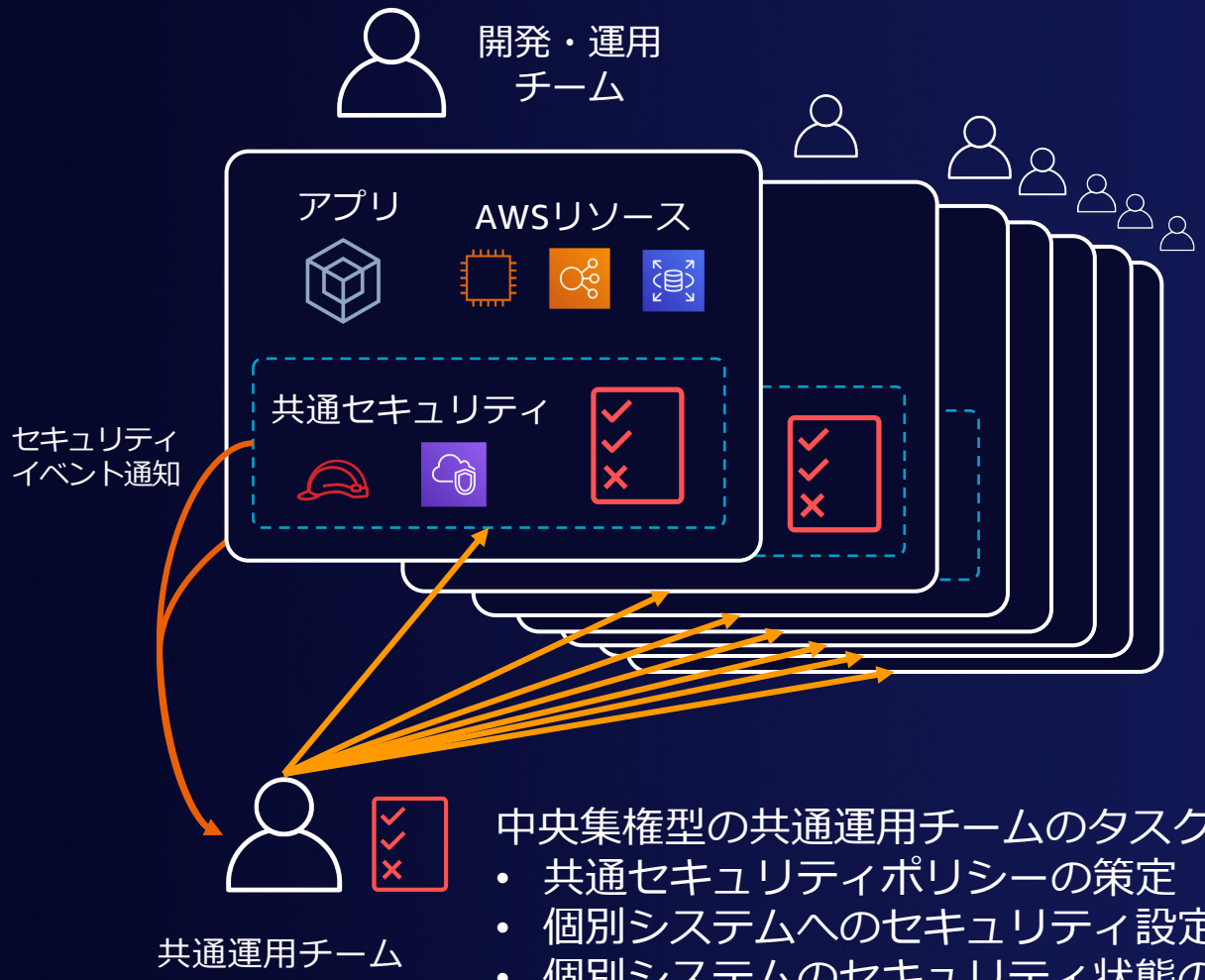
中央集権型の管理はクラウドの価値を活かし切ることが困難



中央集権型の共通運用チームのタスク例

- 共通セキュリティポリシーの策定（利用可能サービスの指定）
- 個別システムへのセキュリティ設定の実施・更新
- 個別システムのセキュリティ状態の監視・イベント対応
- etc.

中央集権型の管理はクラウドの価値を活かし切ることが困難



中央集権型管理のよくある課題

- 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

中央集権型の共通運用チームのタスク例

- 共通セキュリティポリシーの策定（利用可能サービスの指定）
- 個別システムへのセキュリティ設定の実施・更新
- 個別システムのセキュリティ状態の監視・イベント対応
- etc.

テンプレートによるガバナンス

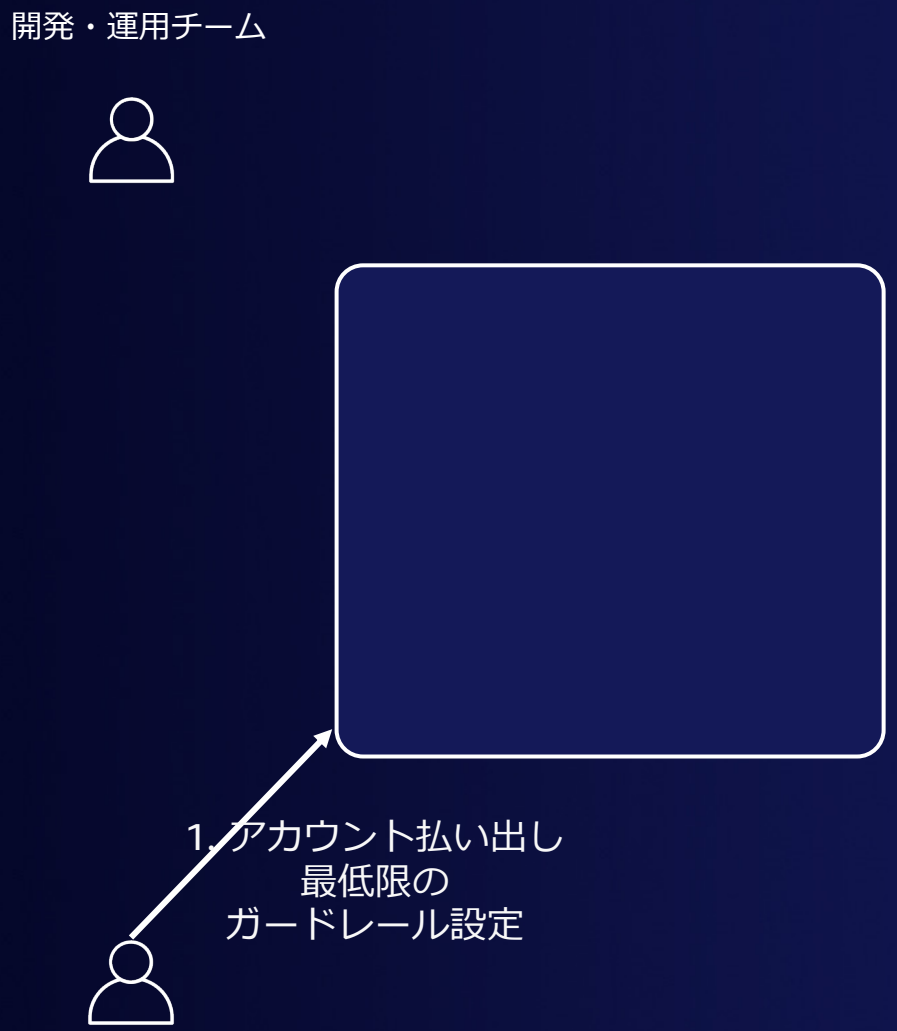
テンプレートを使ったガバナンスの全体像

開発・運用チーム

1. 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

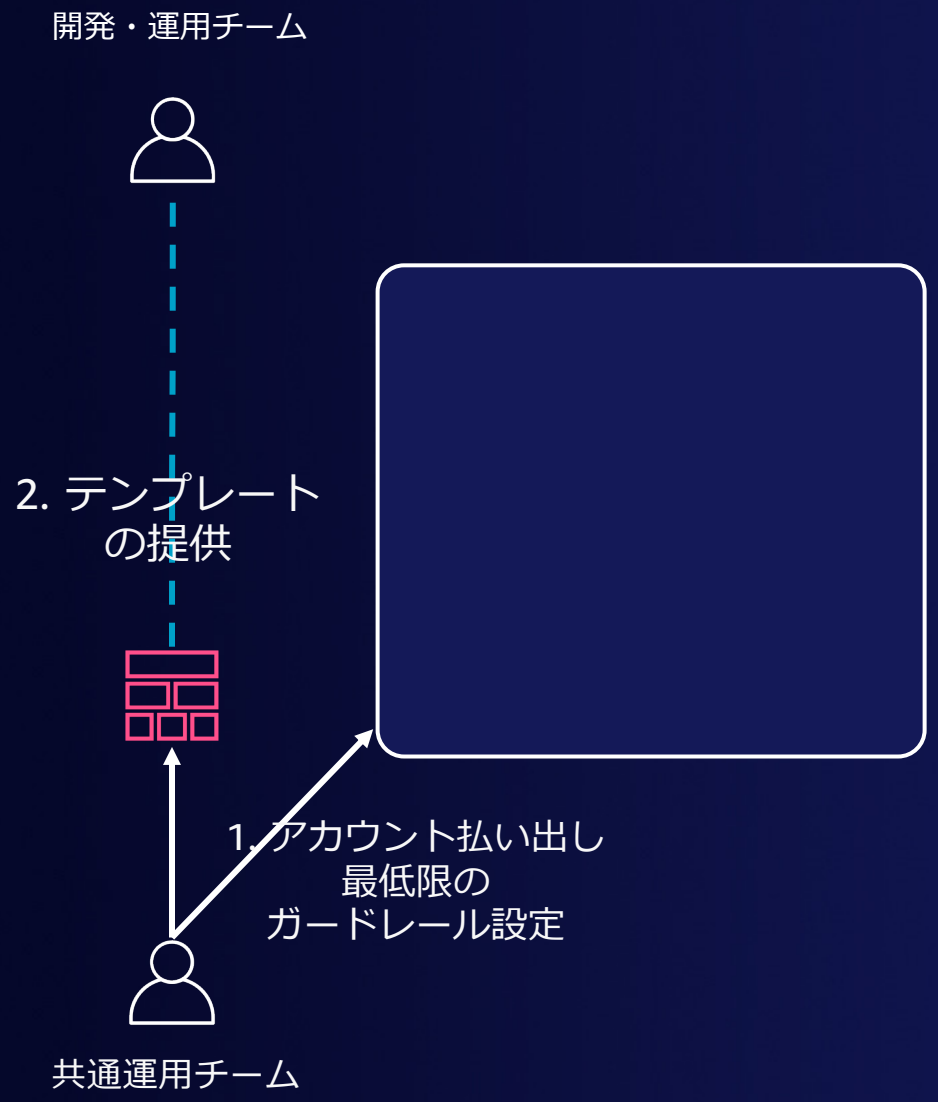
共通運用チーム

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

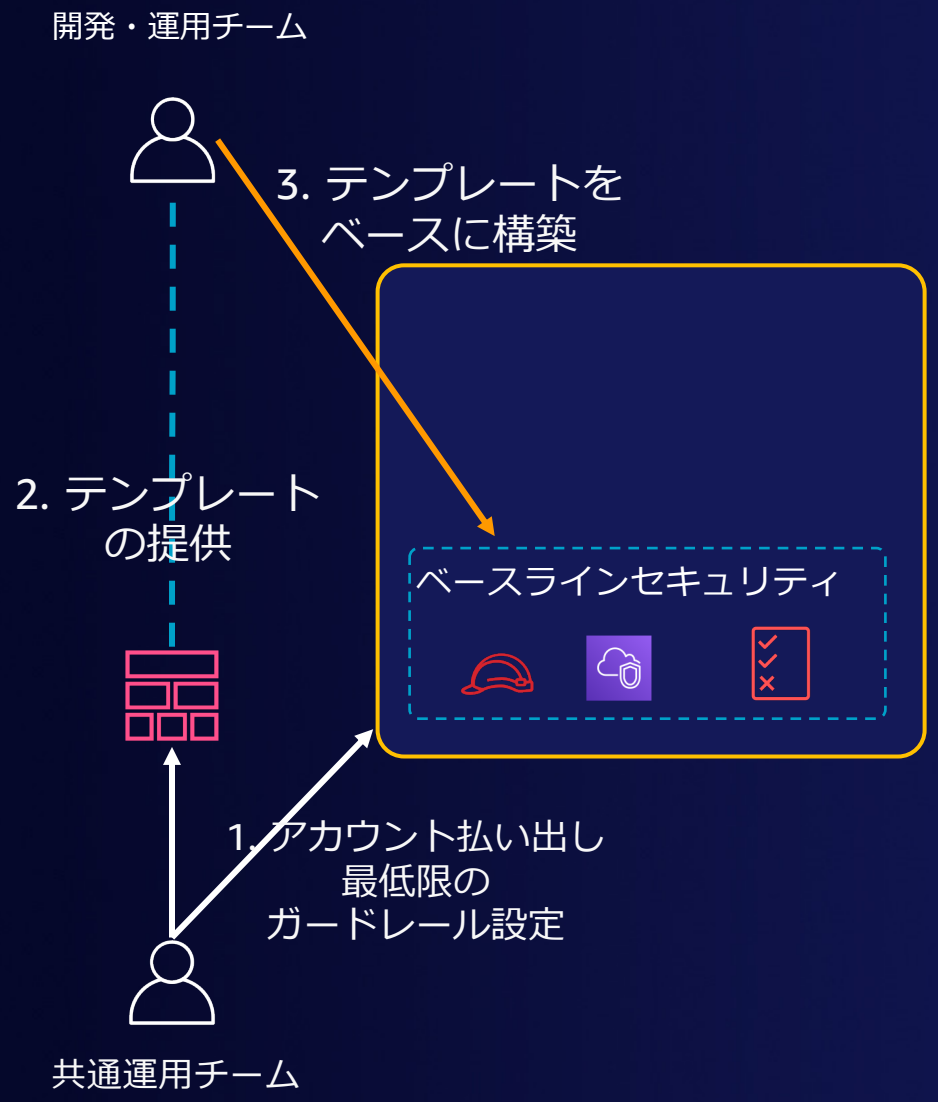
テンプレートを使ったガバナンスの全体像



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

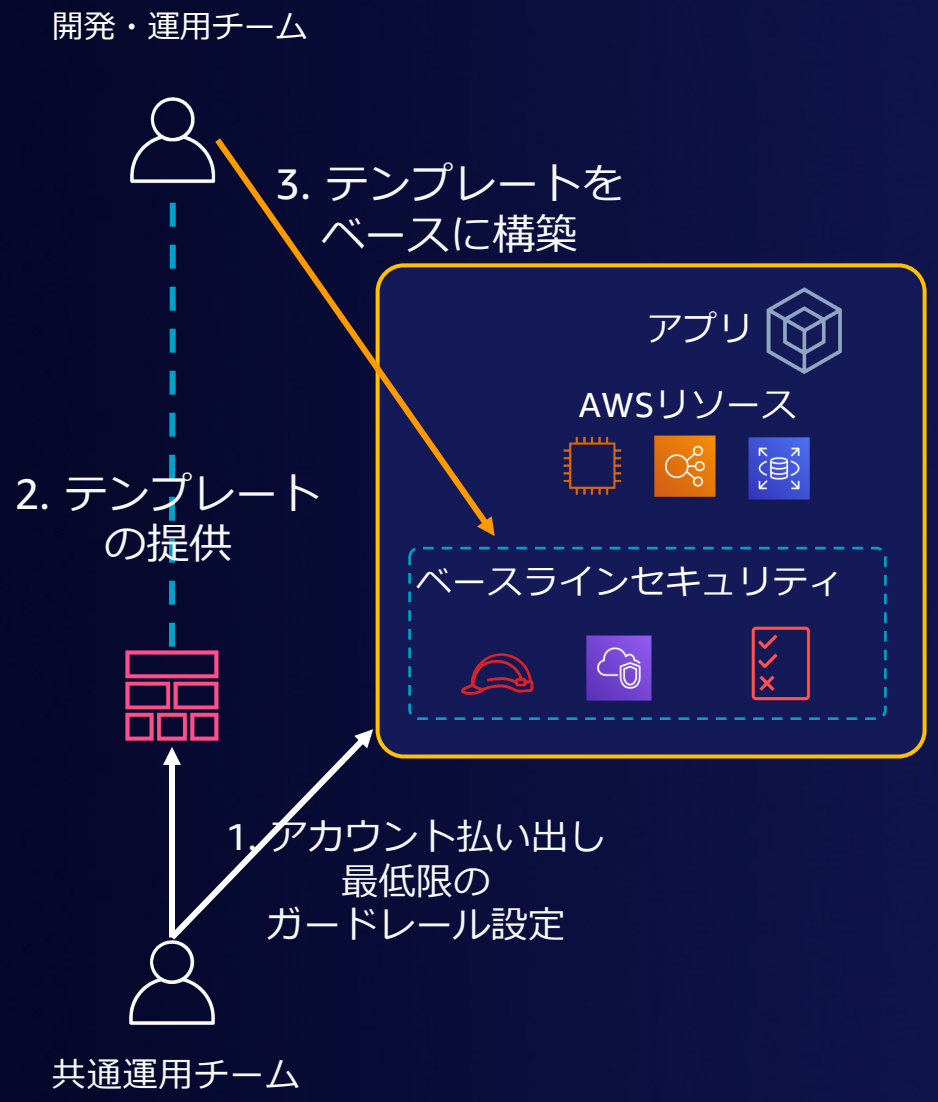


1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

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

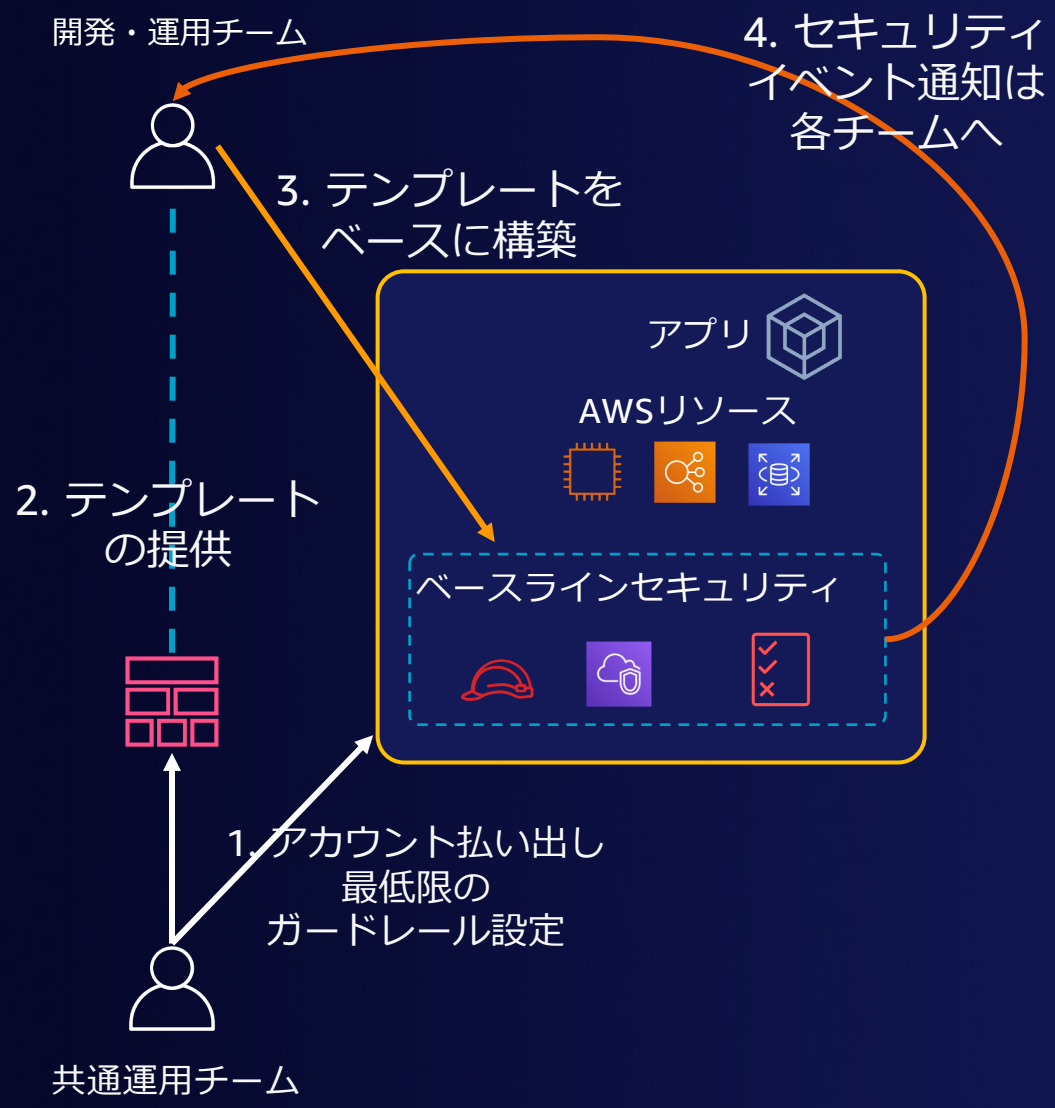


1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

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）

テンプレートを使ったガバナンスの全体像



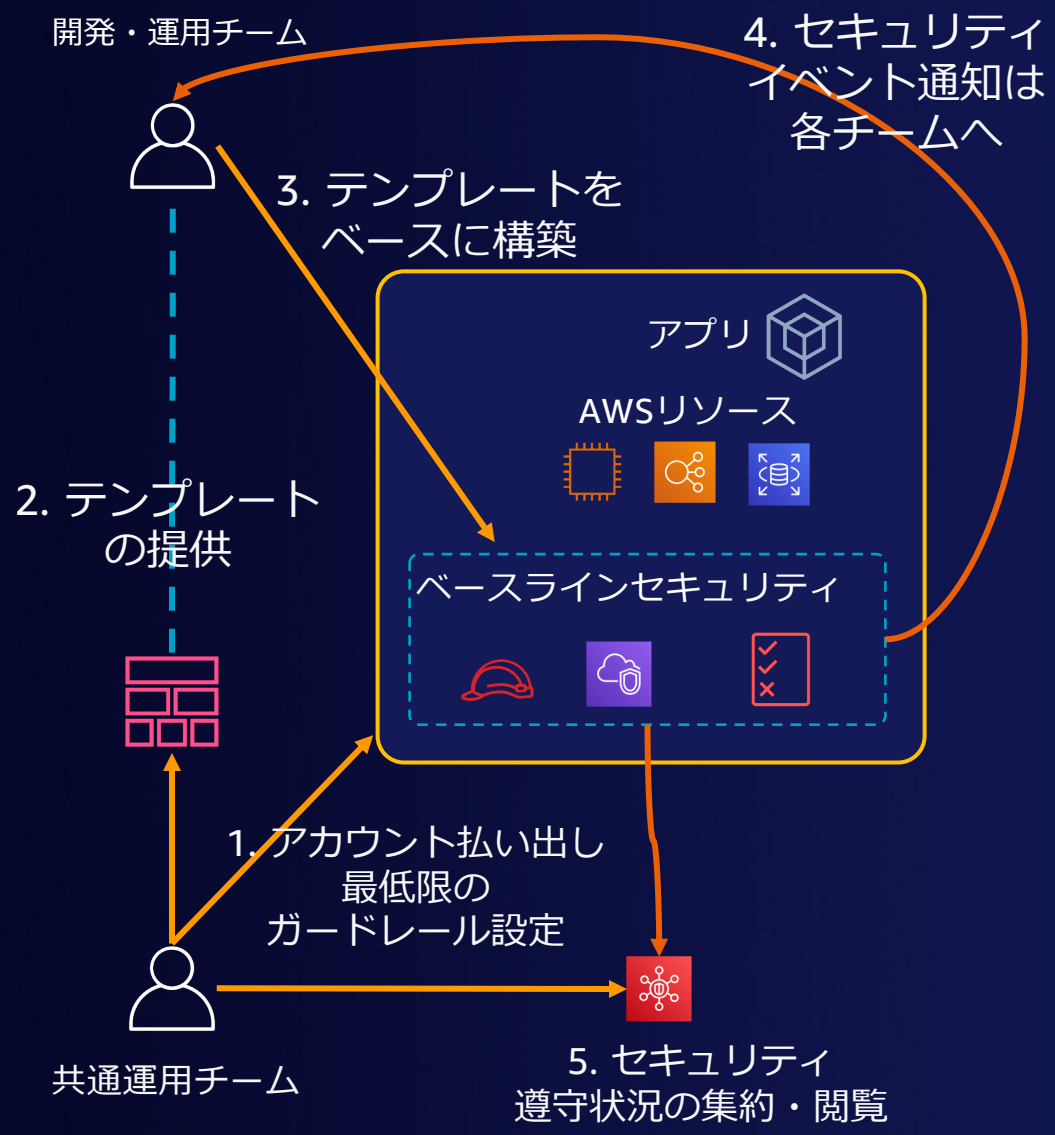
1. 共通運用チームはアカウントの払い出しと最低限のガードレール設定のみを行う

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）

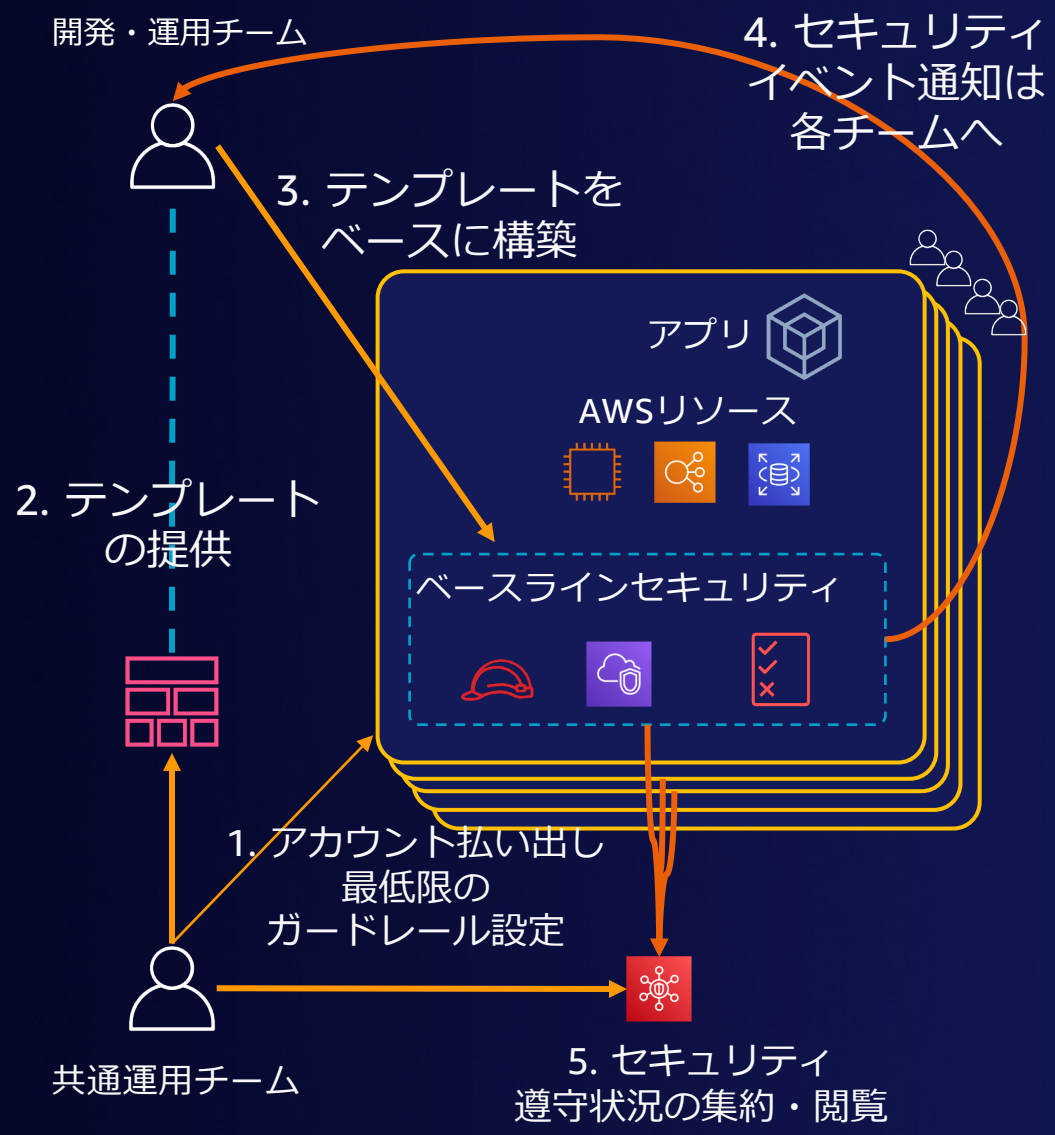
4. セキュリティイベントの即時通知は各チームで対処

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

テンプレートによるガバナンスの考え方

- 許可されたサービスだけを使うより　　アカウントの中で自由にサービスを使える環境を
- 集中管理で設定を強制するより　　テンプレートによる同一設定の展開と分散管理を
- 未然に防ぐより　　逸脱の検知と迅速な修復を
- 初期構築の自動化だけでなく　　コードによる継続的なメンテナンスを
- 自社に最適化した仕組みを作るより　　AWSサービス拡充の柔軟な取り込みを

左の考えを認めつつも右の考えを重視することで
クラウドの価値を最大限活かすガバナンスを実現することを目標とする

Baseline Environment on AWS

Baseline Environment on AWS (BLEA)

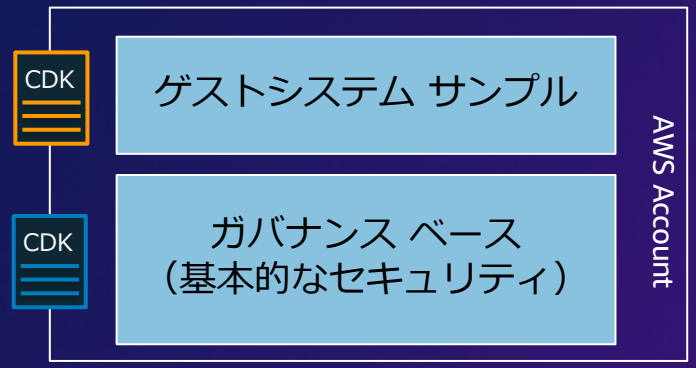
https://github.com/aws-samples/baseline-environment-on-aws

AWSのセキュリティベストプラクティスを実装した
オープンソースのサンプルテンプレート

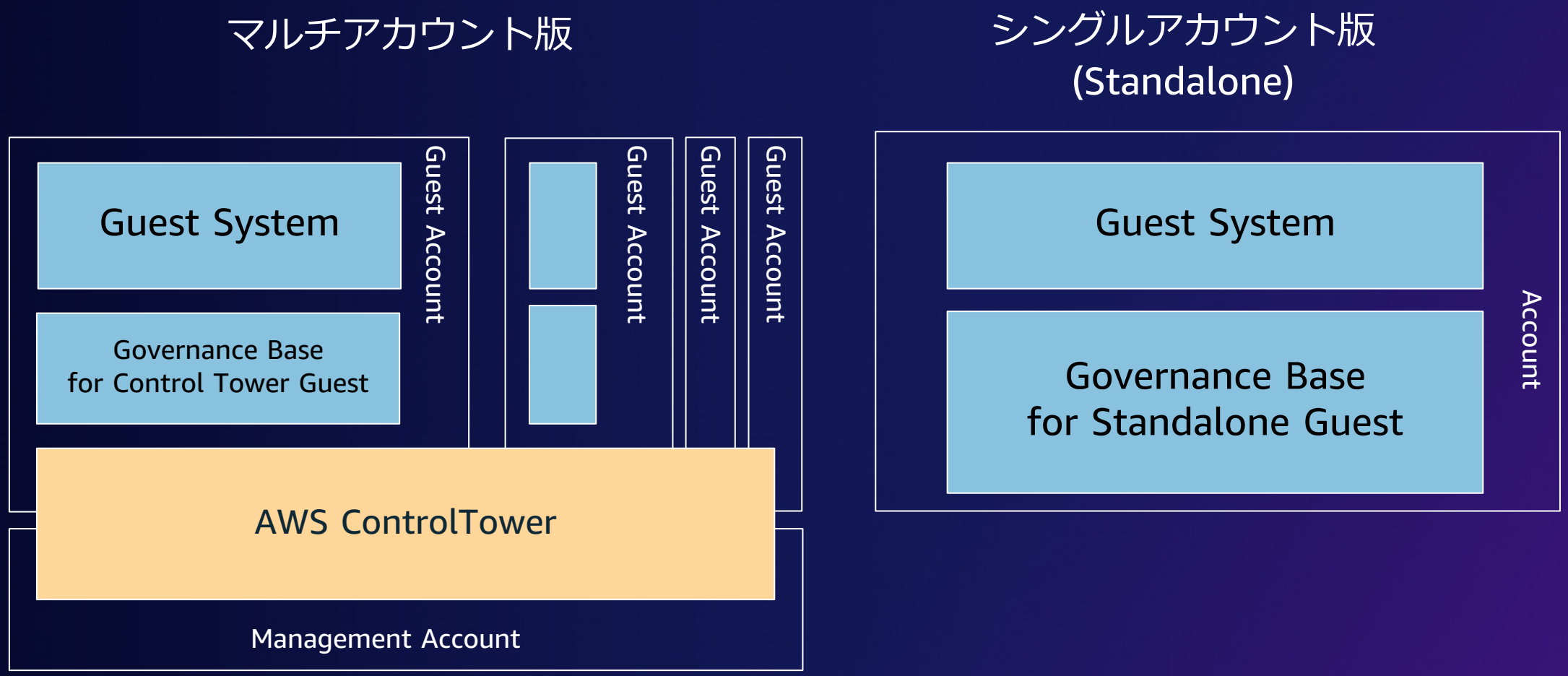
特徴

- 基本的なセキュリティを設定するテンプレートと
  ゲストシステムのサンプルテンプレートを提供
- AWSのセキュリティベストプラクティスに準拠
- AWS Cloud Development Kit (CDK) コード
  参考となるスニペット、コメント、リファレンスを豊富に記載
- チームによる長期的な利用を想定
  CDK標準ライブラリのみを使ったシンプルな実装
  利用者が理解しやすいよう過度な作り込みを避ける

BLEAが提供するテンプレート

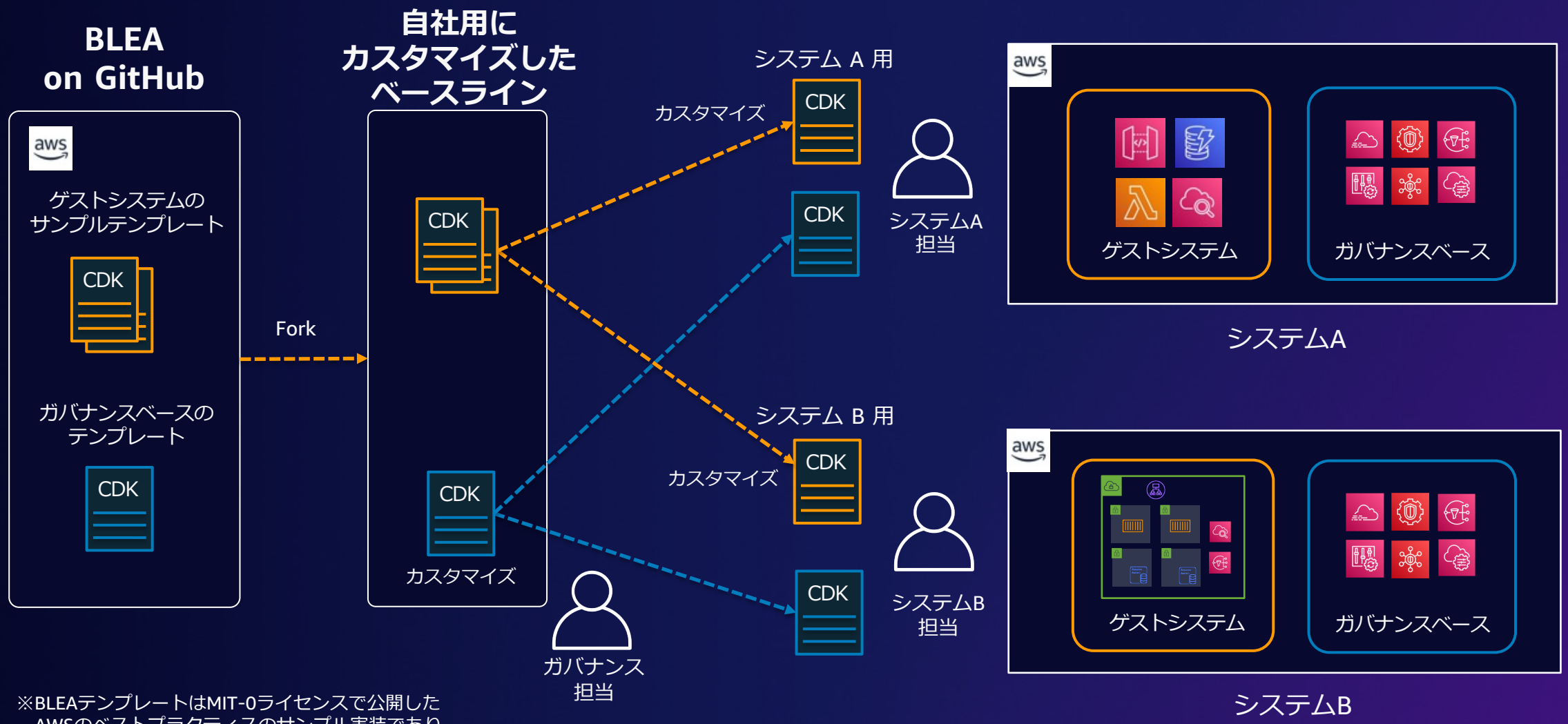


Baseline Environment on AWS の利用パターン



Baseline Environment on AWS の提供範囲
※Governance Baseによって実現されるセキュリティは、マルチアカウント版もシングルアカウント版も同じ
※“Guest system” はマルチアカウント版もStandalone版も同じものが利用可能

Baseline Environment on AWS の利用方法



※BLEAテンプレートはMIT-0ライセンスで公開した
AWSのベストプラクティスのサンプル実装であり、
構築された環境の品質をAWSが保証するものではありません。
実際の構築・運用にあたって、お客様でテンプレートのカスタマイズやテストの実施が必要です。

BLEA ベースライン

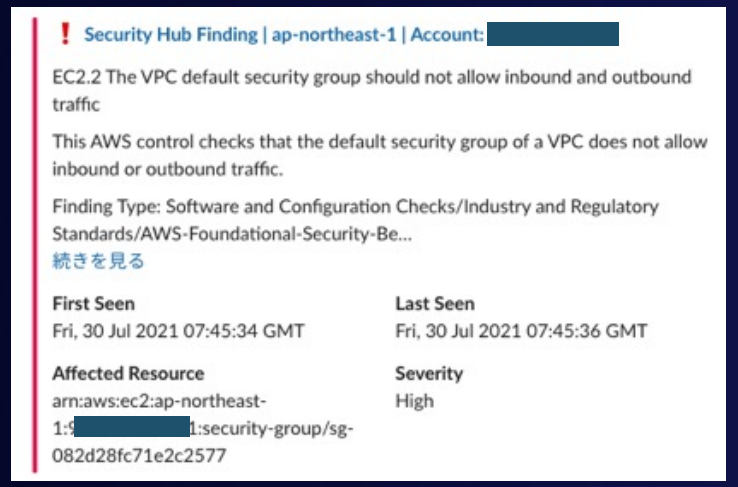
Q. 何をベースライン（基本的な設定）とするか？

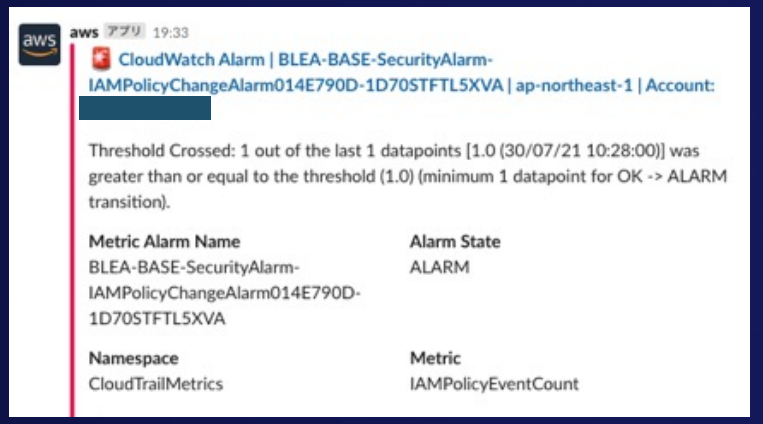
- 実際のシステム構築では要件に合わせた
セキュリティサービスの組み合わせ、設定が必要

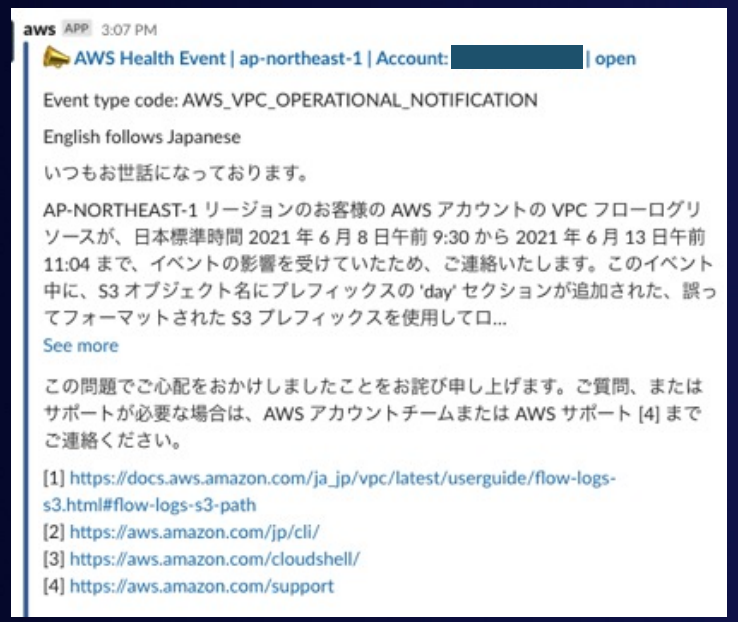
- どのようなシステムでも最低限実施するべき
基本的な設定（ベースライン）がある
  - API呼び出しの記録 / 構成変更の記録
  - セキュリティベストプラクティスに照らしたチェック
  - 不審なAPI呼び出しや通信の検知
  - AWS環境に発生した事象の通知 など

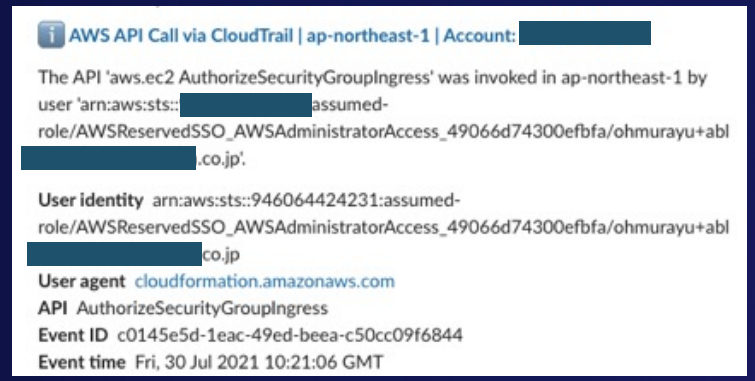
- アラートされたら即時に調査・対応が必要な
セキュリティイベントを通知する

BLEA ベースラインのセキュリティアラート通知









セキュリティサービスの通知

- AWS SecurityHub
  - 以下のCritical/Highを通知
  - CIS Benchmark
  - AWS Foundational Security Best Practices
- Amazon GuardDuty
  - AWSが推奨する Severity を通知
- AWS Config Rules – NON_COMPLIANT
- AWS Health – アカウント固有情報

注意が必要な操作

- セキュリティグループの変更
- Nework ACL の変更
- AWS CloudTrail の変更
- AWS IAM* Policy の変更
- 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

* AWS Identity and Access Management

BLEA シングルアカウント版 アーキテクチャ

(ver. 2022/03/14)

Management Account

AWS Organizations

Guest Account

AWS IAM* role for Admin
(OrganizationAccountAccessRole)

AWS CloudTrail

AWS Config Rules

Amazon GuardDuty

AWS SecurityHub

AWS Systems Manager
QuickSetup

AWS IAM*
AccessAnalyzer

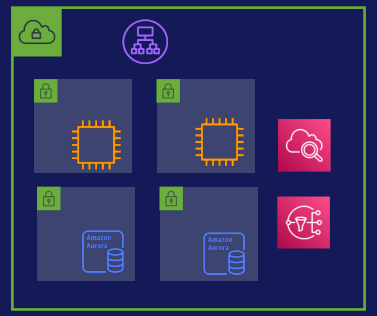
Logging Bucket

AWS Config Rules
+Automation

Amazon EventBridge
+AWS Chatbot

AWS IAM* role
(for Guest)

Guest System



Operation Guide

AWS CDK template

* AWS Identity and Access Management

BLEA マルチアカウント版 アーキテクチャ

(ver. 2022/03/14)

* AWS Identity and Access Management

参考： 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. 2021/10/26)

	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-Organizations	(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-Log Bucket)	CT-CloudTrail/Config	CT-Log Bucket	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			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 yet implemented

管理タスクの主体　管理される側

BLEA ゲストシステムサンプル

Q: なぜ BLEA はゲストシステムサンプルを含むのか？

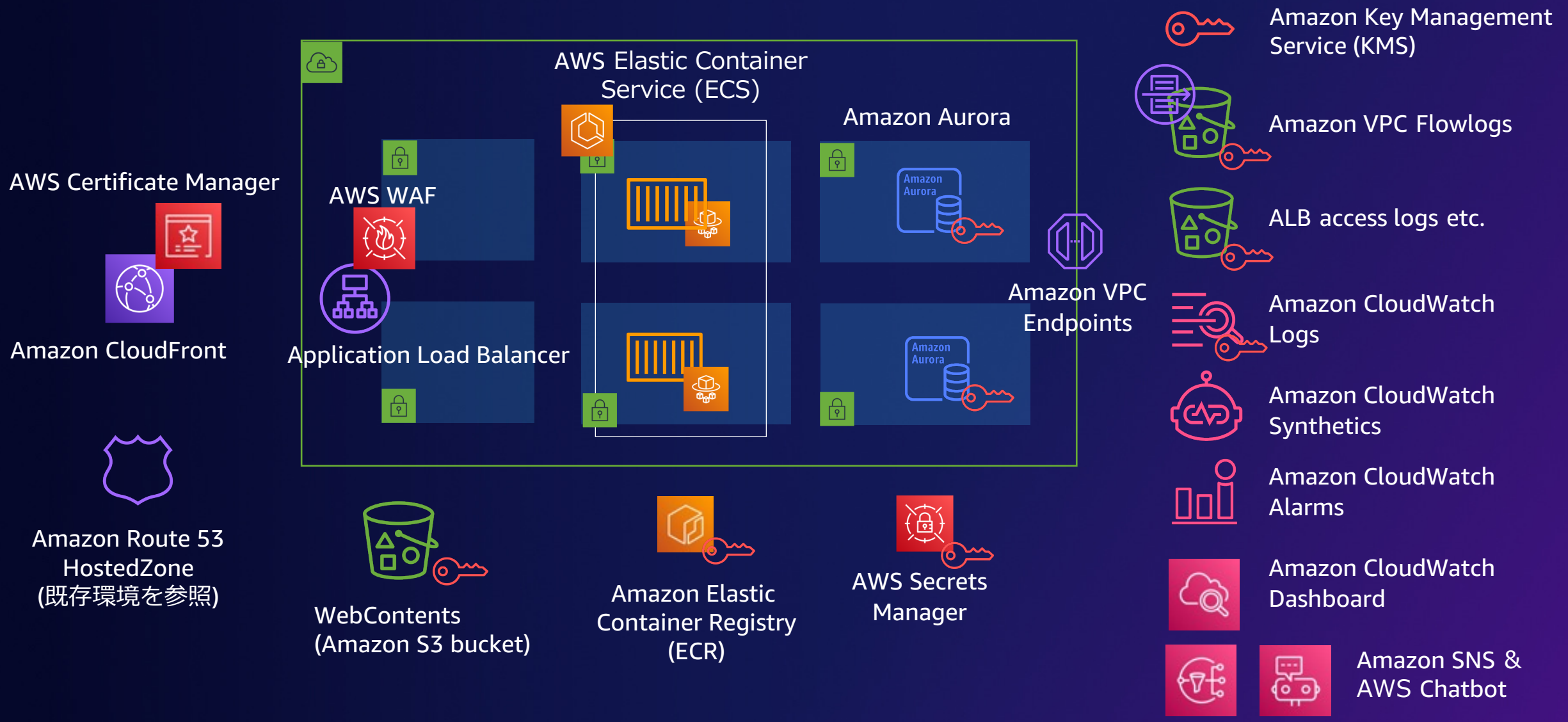
- セキュリティはゲストシステムを含む全てのレイヤに必要
  - ベースラインは AWS の環境を保護する最低限の設定
  - ベースラインだけではセキュリティを担保できない

- 実装の参考としてそのまま使えるレベルの具体的な設定が必要
  - よく利用されるユースケースや機能を選定
  - Security Hubの CIS ベンチおよびAFSBP* の CriticalおよびHighの対処方法が実装済

- サンプル実装をリファレンスとしつつ拡張していく使い方を想定
  - よく利用する CDK コードの書き方やコメントやリファレンスを多く記載
  - BLEA 独自クラスは極力作らず、CDK の基礎知識と標準 API リファレンスを読めば、必要な部分をコピペしつつ開発を開始できるように実装

* AWS Foundational Security Best Practice

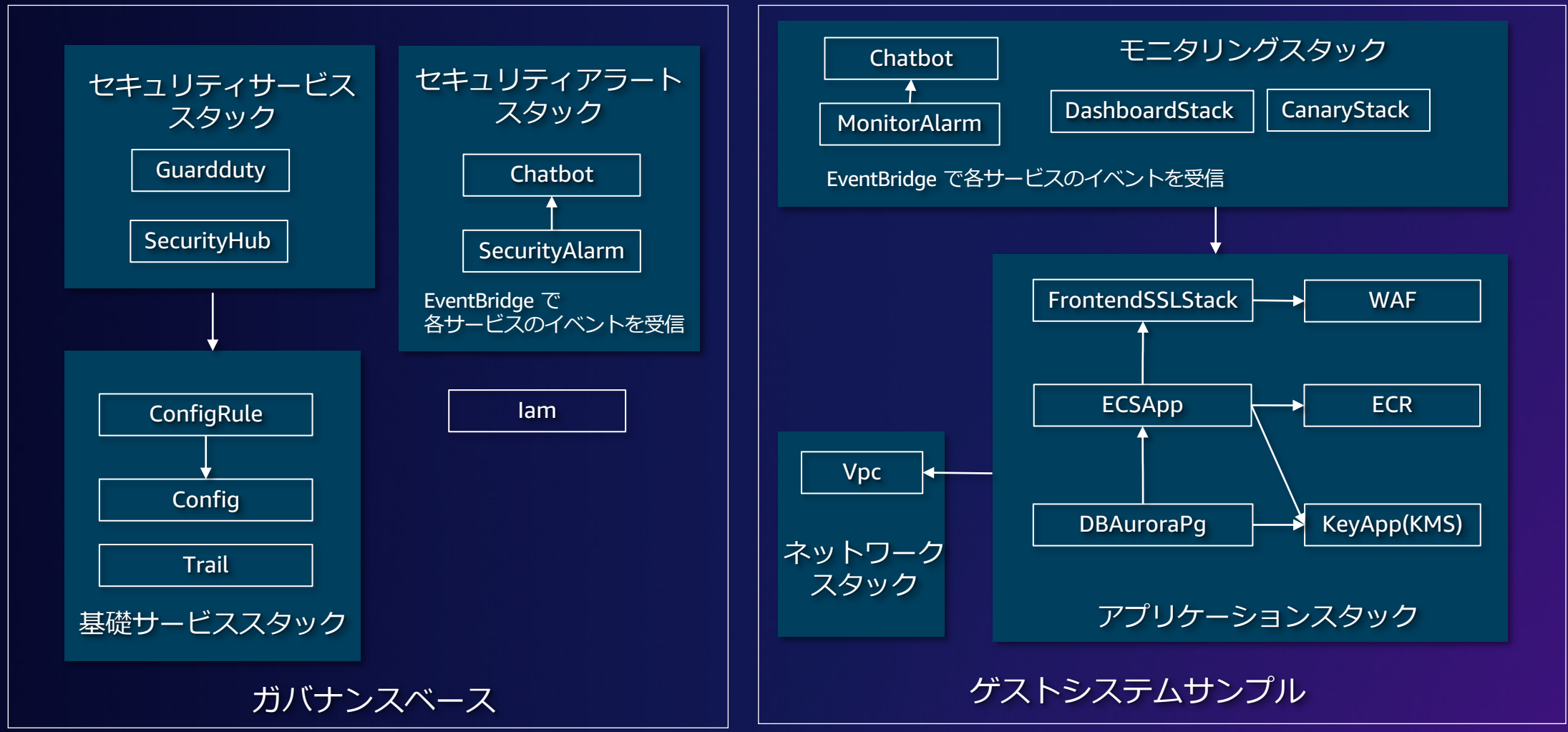
BLEA ゲストシステム例： Webアプリケーション (ECS+HTTPS) (ver. 2022/03/14)



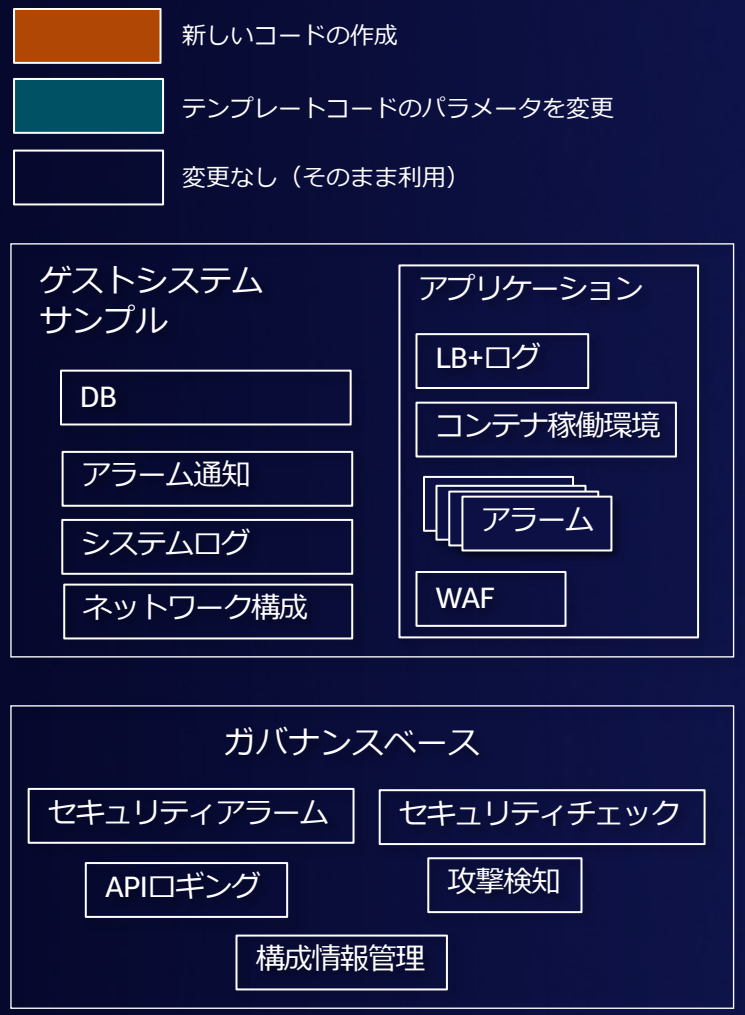
Baseline Environment on AWS - スタック依存関係

ガバナンスベースとゲストシステムサンプルには依存関係がない

(ver. 2022/03/14)



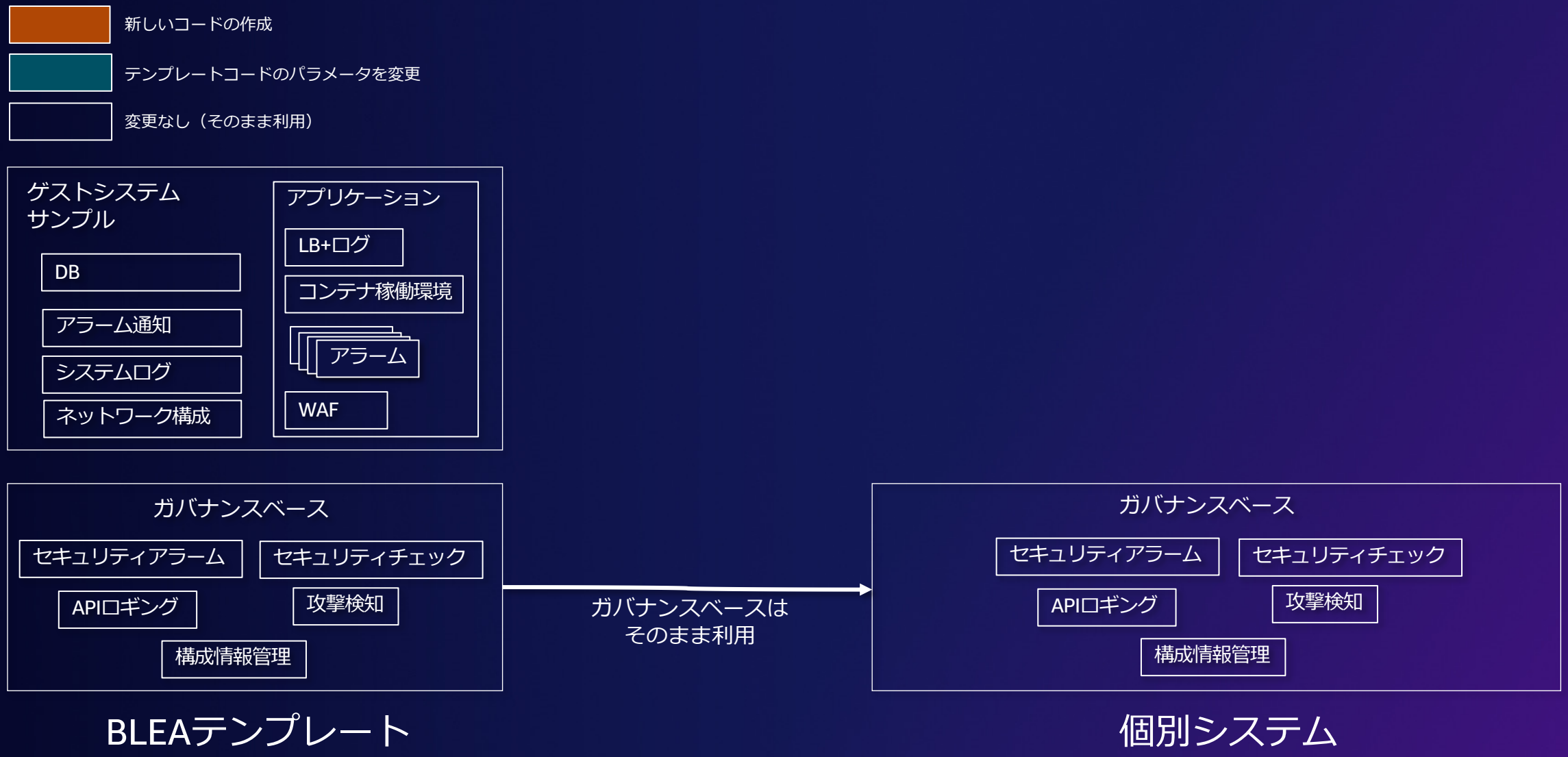
テンプレートを利用した開発の例



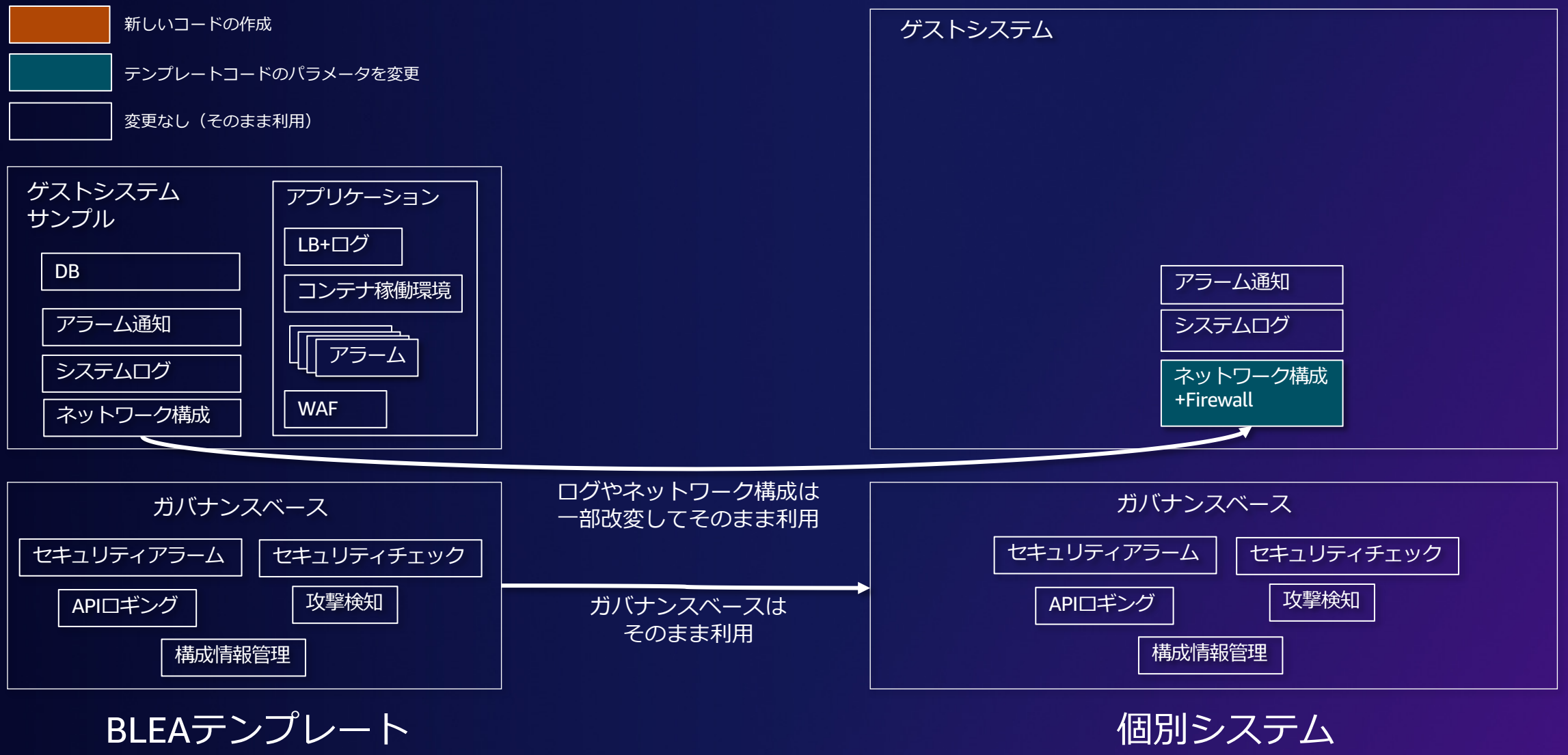
BLEAテンプレート

個別システム

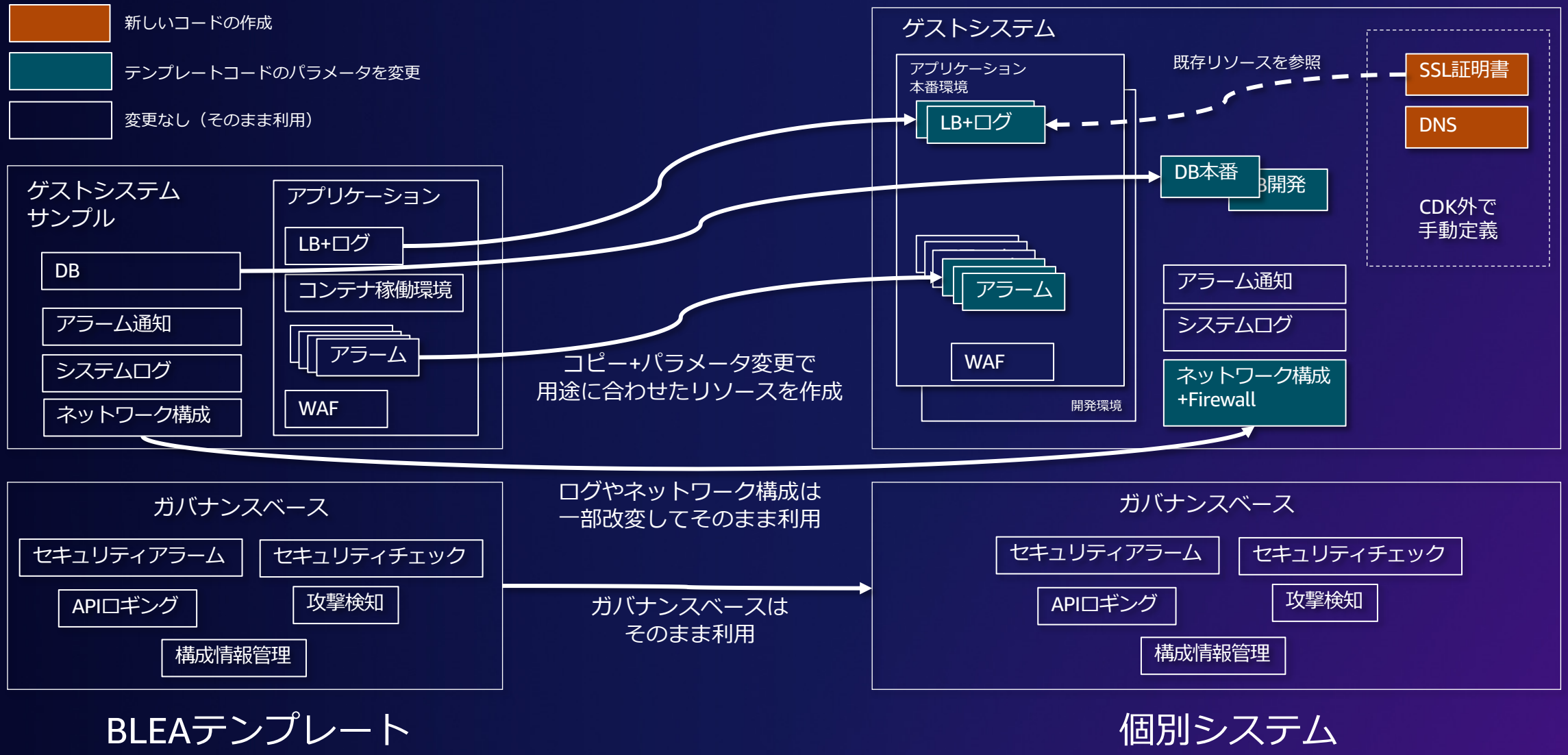
テンプレートを利用した開発の例



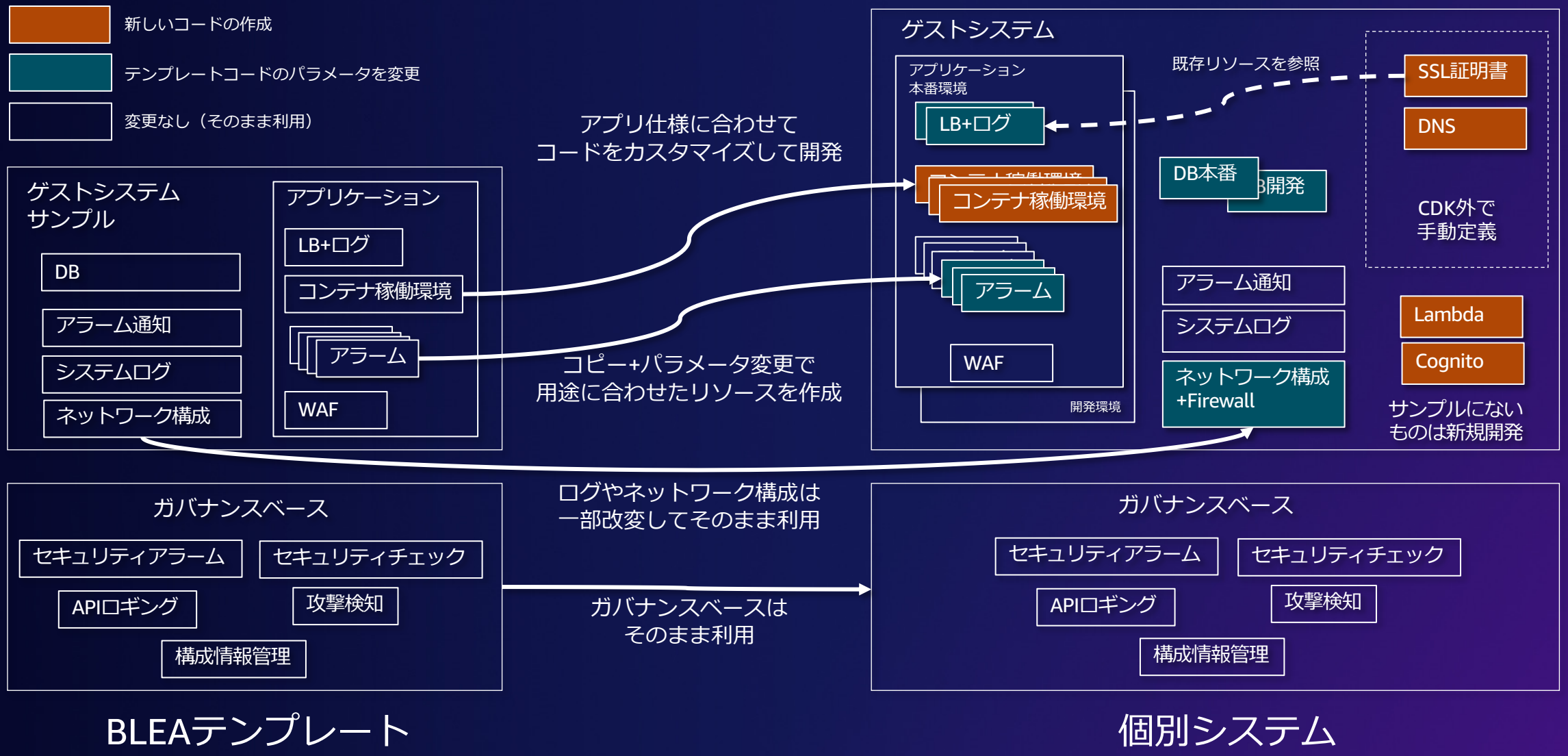
テンプレートを利用した開発の例



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



なぜ AWS CDK なのか？

インフラのプロビジョニングおよび設定を
信頼性高く一貫した手順で実施することは
DevOps と
迅速なソフトウェアデリバリの基盤

手動のインフラ操作では
プロビジョニングの際
手順の一貫性、設定の正常性、
エラーの検知/修復
が実現できない

Infrastructure as code によって
自動化と一貫した手順によるリソース管理を実現

信頼できる唯一の情報源
バージョン管理された
リポジトリ

必要に応じて
変更を前のバージョンに
ロールバック

コードによる
ベストプラクティスの共有
一貫性を強化

AWS の Infrastructure as Code サービス

AWS CloudFormation

- テンプレートを YAML または JSON で定義
- テンプレートを迅速かつ一貫性を持って展開
- ライフサイクル全体にわたって
  テンプレートを管理

AWS CDK

- AWSベストプラクティスに基づく
  Python, Java, .NET, やTypeScript の
  クラスライブラリやコンストラクト
- カスタムコンストラクトの作成と共有
- リソースを一貫的に展開

Q: なぜ BLEA は AWS CDK を使うのか？

- 記述量が少なくシンプルで理解しやすいコード（CloudFormation 比）
  - 長期にわたる安全な運用には引き継ぎしやすいコードが必要
  - テンプレートをブラックボックスにしない

- 開発上の利点が多い
  - AWSのベストプラクティスがライブラリに含まれておりコードの記述量が少なく済む
  - 一般のプログラミング言語（TypeScript）であるためオブジェクト参照が明快
  - エディタによる強力なサジェストがあるため実装中の誤り混入を少なくできる
  - 実績のある CloudFormation によりデプロイされる

- CDKの学習コンテンツとしての利用
  - インフラエンジニアは TypeScript 開発未経験者であることも多いため、
    環境構築も含めた 1st step ガイドやドキュメントを提供する
  - メンテナンスおよび可読性を第一に考え過度な自動化（複雑な実装）をしない
    （場合によっては手作業を許容し、後にサービスアップデートの取り込みを想定）

コードサンプル (ALB アクセスログの設定)

ALB のアクセスログ設定（やや特殊な書き方）

特殊な設定の理由（security best practice準拠）、
一般的な設定のやり方、およびリファレンス

ログ出力に必要な特殊権限とその書き方
(RegionInfo の使い方)

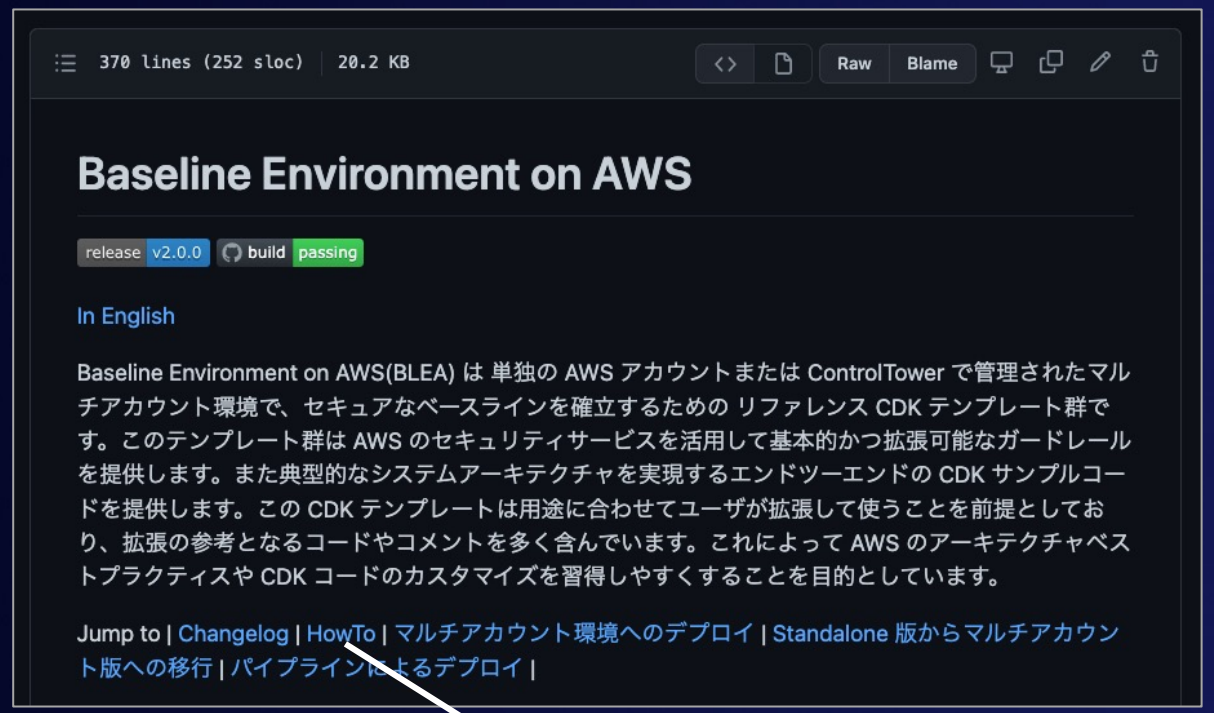
```
lbForApp.setAttribute('access_logs.s3.enabled', 'true');
lbForApp.setAttribute('access_logs.s3.bucket', albLogBucket.bucketName);

// Permissions for Access Logging
//    Why don't use bForApp.logAccessLogs(albLogBucket); ?
//    Because logAccessLogs add wider permission to other account (PutObject*). S3 will become Noncompliant on Security Hub [S3.6]
//    See: https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-standards-fsbp-controls.html#fsbp-s3-6
//    See: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html#access-logging-bucket-permissions
albLogBucket.addToResourcePolicy(
  new iam.PolicyStatement({
    effect: iam.Effect.ALLOW,
    actions: ['s3:PutObject'],
    // ALB access logging needs S3 put permission from ALB service account for the region
    principals: [new iam.AccountPrincipal(ri.RegionInfo.get(cdk.Stack.of(this).region).elbv2Account)],
    resources: [albLogBucket.arnForObjects(`AWSLogs/${cdk.Stack.of(this).account}/*`)],
  }),
);
```

Next step...

BLEA GitHub リポジトリ
https://github.com/aws-samples/baseline-environment-on-aws/blob/main/README_ja.md

AWS環境にセキュアなベースラインを提供するテンプレート
「Baseline Environment on AWS」のご紹介（解説ブログ）
https://aws.amazon.com/jp/blogs/news/announcing-baseline-environment-on-aws/



HowTo にいろいろ情報あります

手元環境構築から導入まで
詳しい手順あります

デプロイの流れ

デプロイするステップについて記載します。デプロイだけ行う場合はエディタ環境の構築は必ずしも必要ありませんが、コードの変更が容易になりミスを減らすことができるため、エディタも含めた開発環境を用意することをお勧めします。

前提条件

a. ランタイム

以下のランタイムを使用します。各 OS ごとの手順に従いインストールしてください。

- Node.js (>= `14.0.0`)
  - `npm` (>= `8.1.0`)
- Git

npm は workspaces を使用するため 8.1.0 以上が必要です。最新バージョンは以下のようにしてインストールしてください。

```
npm install -g npm
```

b. 開発環境

CDK コードを安全に編集するため、本格的な開発を行わない場合であっても開発環境のセットアップを推奨します。以下に VisualStudioCode のセットアップ手順を示します。

- [手順]: VisualStudioCode のセットアップ手順

典型的な導入手順

BLEA を使う場合の最も典型的な導入手順は次の通りです。ここでは単独のアカウントにガバナンスベースとゲストアプリケーションを導入する手順を示します。

1. 関連ライブラリのインストールとコードのビルド
2. AWS CLI の認証情報の設定
3. デプロイ対象のアカウント作成
4. ガバナンスベースをデプロイ
5. ゲストアプリケーションサンプルをデプロイ

> NOTE: ここでは 単独アカウントに Standalone 版ガバナンスベースと Web アプリケーションサンプルの ECS 版を導入します。 ControlTower を使ったマルチアカウント版の導入手順については、Deploy to ControlTower environmentを参照してください。

導入手順

ここでは最もシンプルな、単一アカウントへの Standalone 版導入を例にとって解説します。

1. リポジトリの取得とプロジェクトの初期化

1-1. リポジトリの取得

```
git clone https://github.com/aws-samples/baseline-en
cd baseline-environment-on-aws
```

まとめ

1. クラウド活用に必要なガバナンスの考え方
   →集中型管理はクラウドの価値を活かし切るのが難しい

2. テンプレートによるガバナンス
   →テンプレートを使った分散管理の実現

3. Baseline Environment on AWS (BLEA) の概要
   1. BLEA ベースライン
   2. BLEA ゲストシステムサンプル
   3. なぜ AWS CDK なのか？

Thank you!