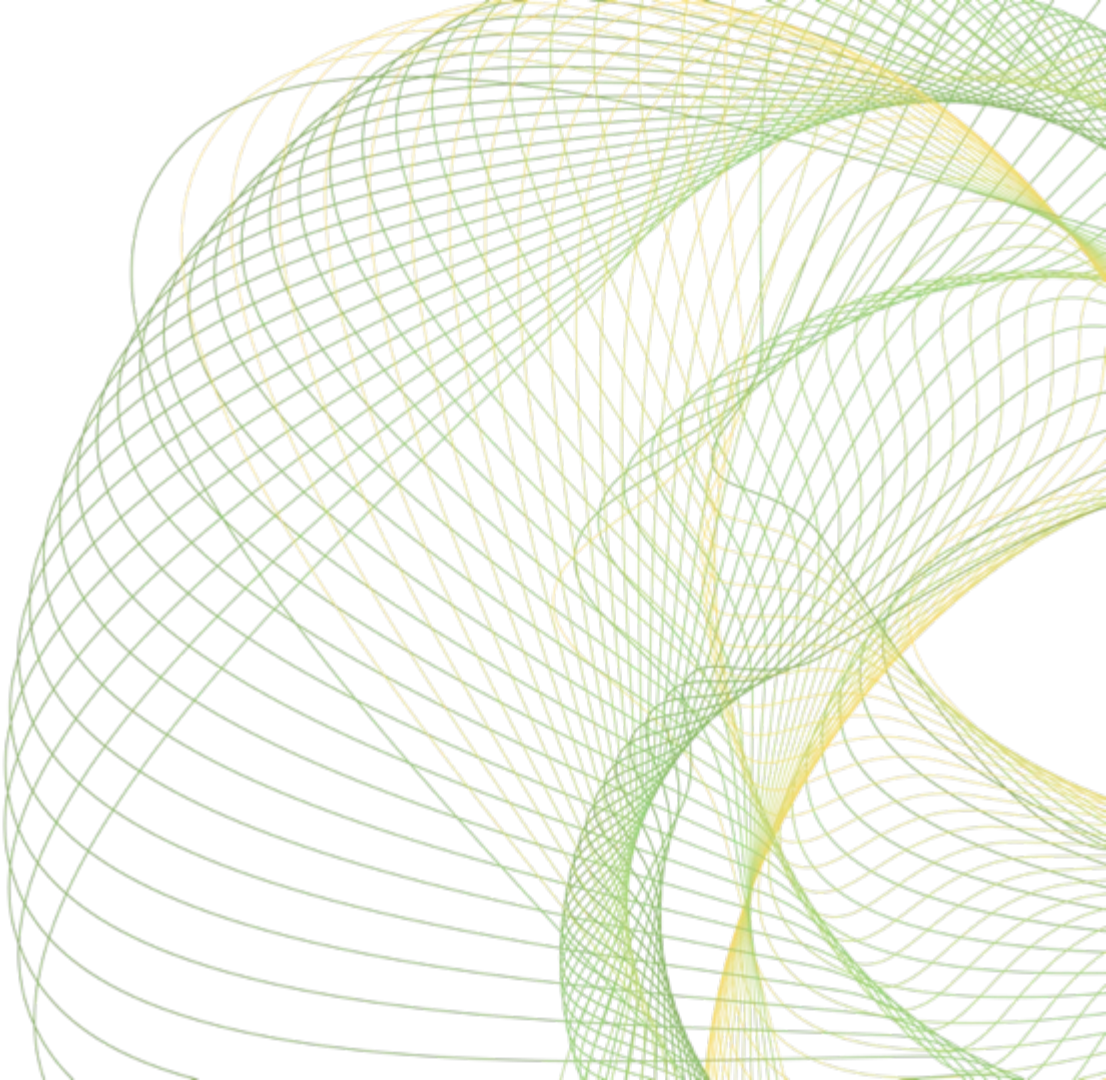
回答例_補足

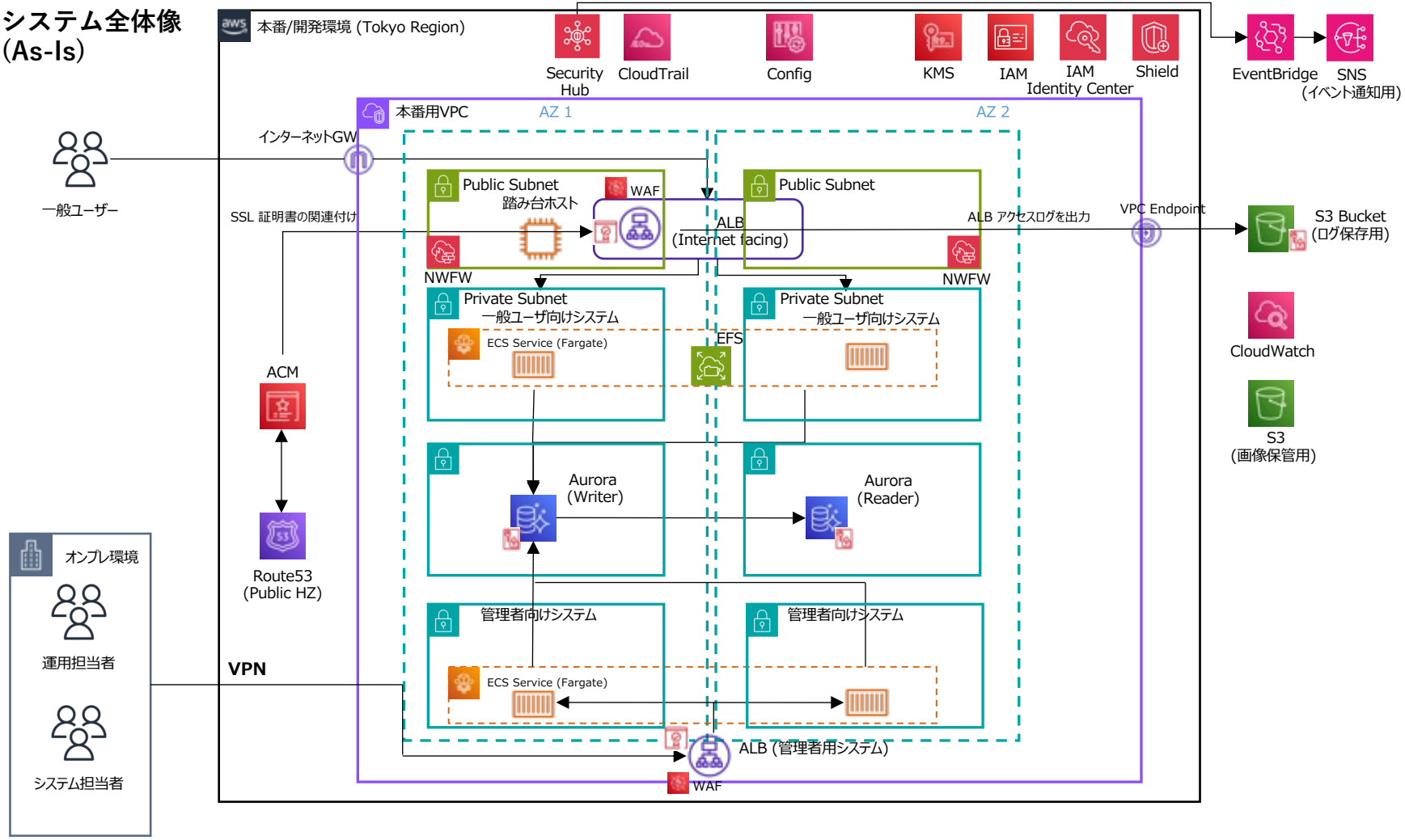


aws financial services

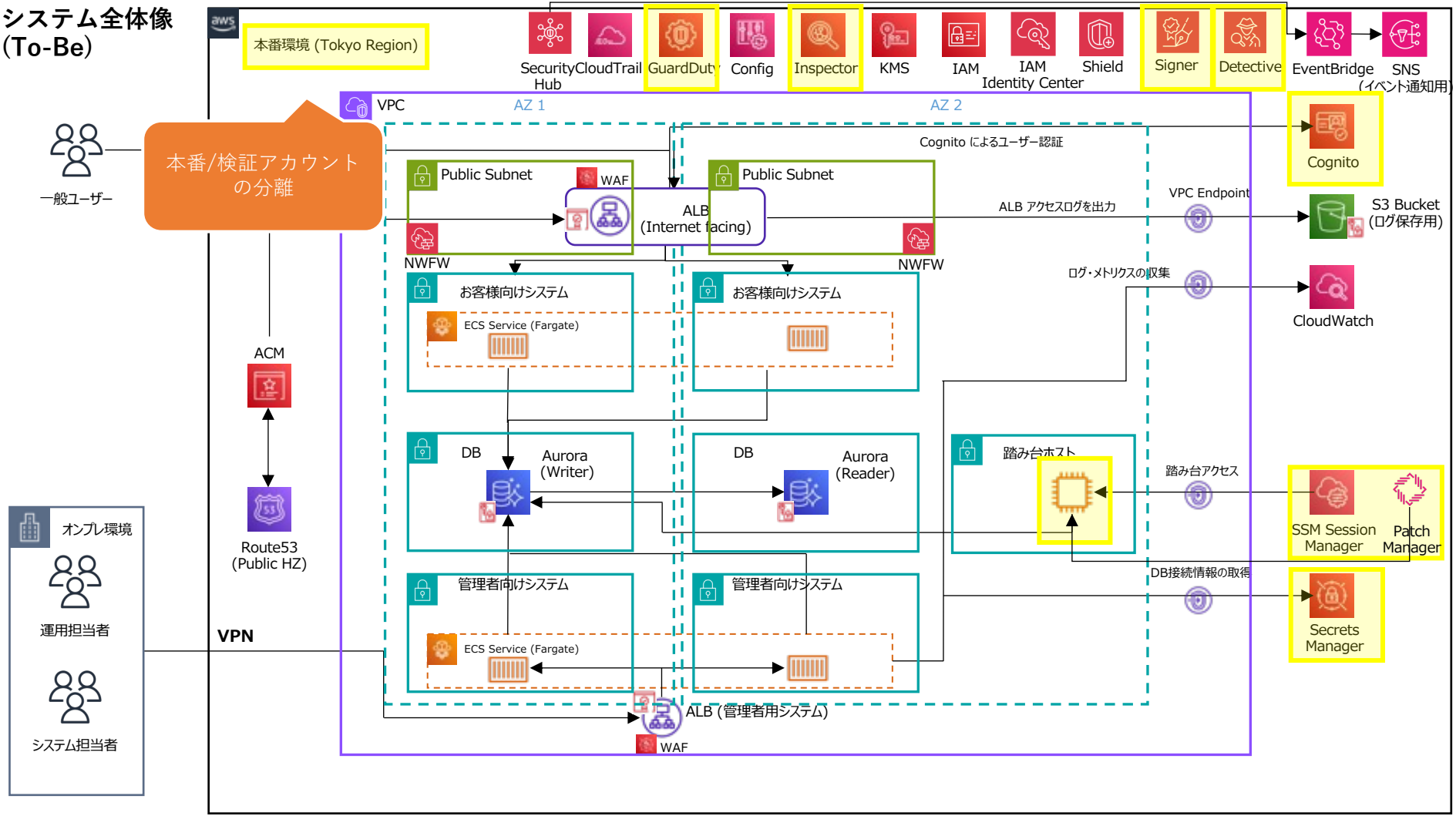
本資料の内容についての注意

- この資料は、ワークショップにて提示された課題の内容に対する解説と、
  一つの回答例を示すものです。
- この資料に記載されている回答はあくまでも一例です。
  本資料に記載の回答例と一致しない構成/設計についても妥当性が認められる場合は考えられます。

システム全体像 (As-Is)



システム全体像 (To-Be)



参考: AWSアカウントが実現するもの

セキュリティ境界の単位

セキュリティ上の境界

- 他のお客様とのセキュリティの境界の役割を果たす (他のAWSアカウントの構成は見えず、アクセスできない)

リソース管理の単位

リソースの管理単位

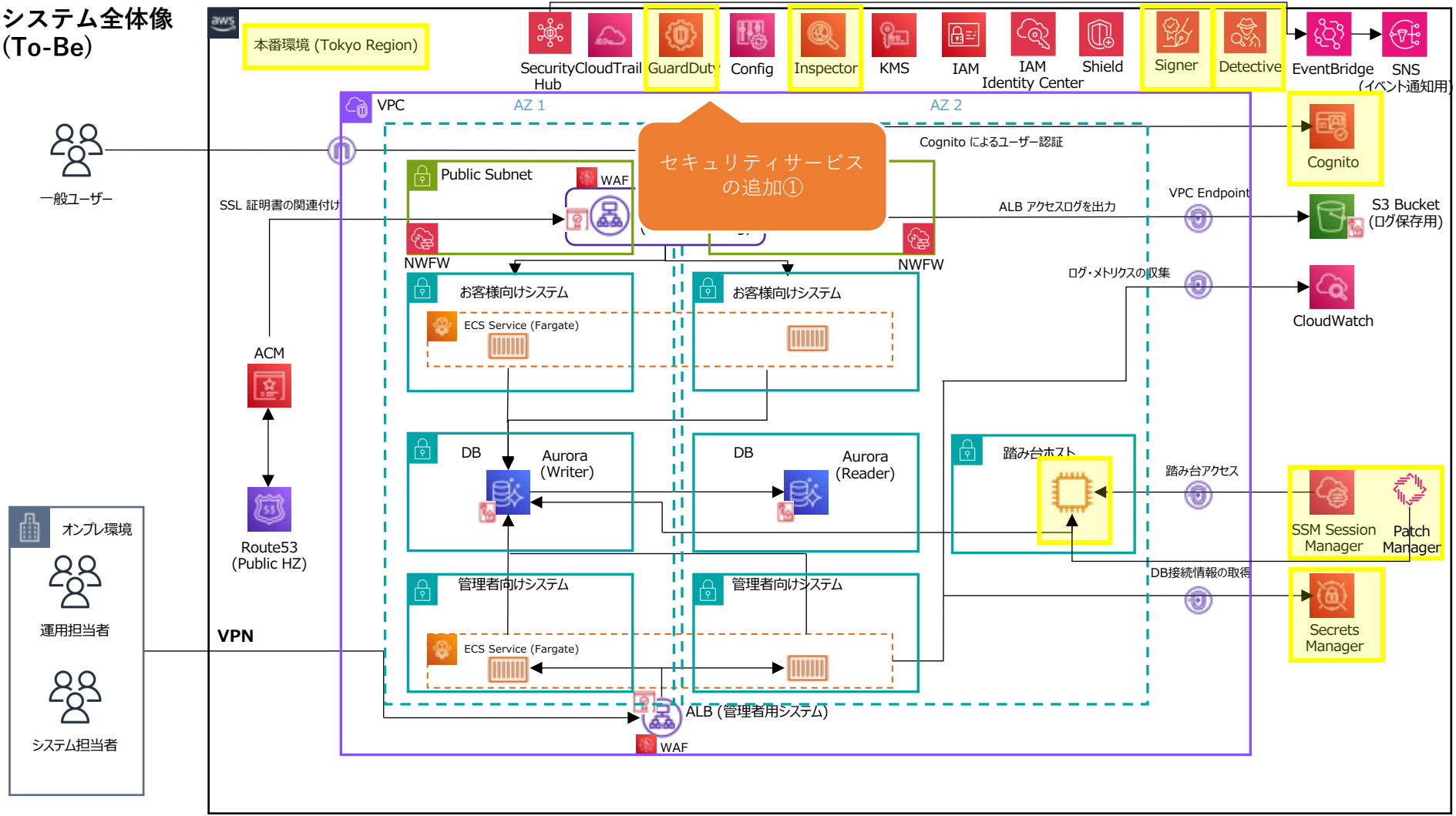
- アカウント単位でAWSのさまざまなリソースを管理する

課金の単位

課金の分離単位

- サービスの利用における課金はAWSアカウントに対して行われる

システム全体像（To-Be）



参考: Amazon GuardDutyの概要

Amazon GuardDuty

悪意のあるアクティビティや不正なアクティビティから保護

サービス概要

- セキュリティの観点から**脅威リスクを検知**するAWSマネージドサービス
- 分析のソースには下記を利用し、メタデータの連続ストリームを分析
  - ✓ VPC Flow Logs
  - ✓ AWS CloudTrail Event Logs
  - ✓ DNS Logs
  - ✓ Kubernetes Audit Logs
- 悪意のあるIPアドレス、異常検出、機械学習などの統合**脅威インテリジェンス**を使用して脅威を認識
- Malware保護機能によりEC2インスタンスまたはコンテナワークロードの疑わしい動作を検出
- 結果は自動的に EventBridgeに送信
  - ✓ S3バケットにエクスポートすることも可能
  - ✓ 出力先のS3バケットは同じアカウントでも、他のアカウントでも対応可

※30日間無料使用あり

参考:
https://aws.amazon.com/jp/guardduty/
Amazon GuardDuty
https://aws.amazon.com/jp/guardduty/faqs/
Amazon GuardDuty のよくある質問
https://d1.awsstatic.com/webinars/jp/pdf/services/20180509_AWS-BlackBelt_Amazon-GuardDuty.pdf
[AWS Black Belt Online Seminar] Amazon GuardDuty

参考：Amazon Inspector 概要

Amazon Inspector

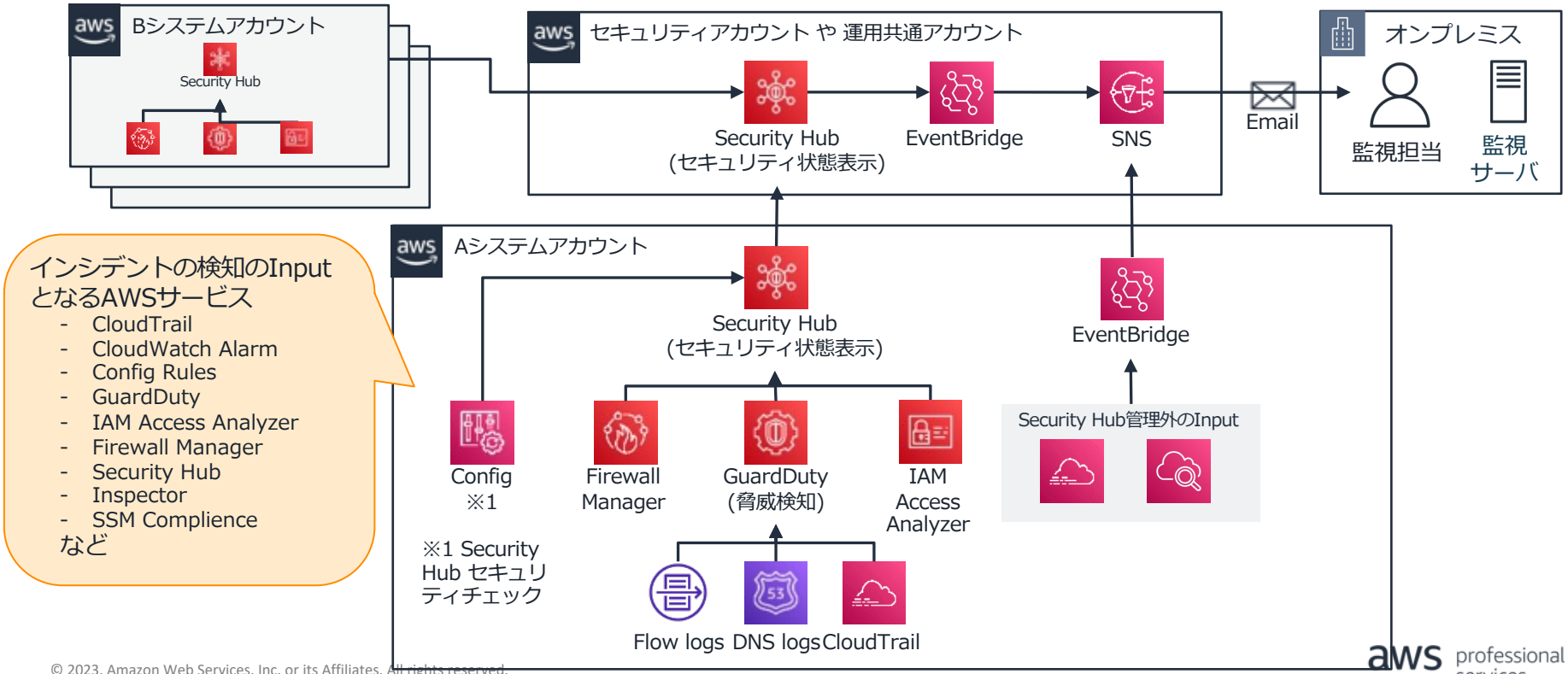
ソフトウェア脆弱性や
意図しないネットワーク露出を検知する
脆弱性管理サービス

サービス概要

- EC2インスタンス(Linux、Windows)、ECRリポジトリコンテナイメージ、Lambda関数・Lambdaレイヤーの脆弱性をスキャンし検出結果(Finding)の評価、管理を行う
  - ✓ パッケージの脆弱性
  - ✓ ネットワーク到達性
- Inspectorを有効化すると存在するEC2、ECR、Lambdaに対するスキャンが開始される
- 以下の要素を提供する
  - ✓システム設定や振る舞いの分析エンジン
  - ✓対象となるすべてのリソースを自動検出し継続的なスキャン
  - ✓検出結果の抑制ルール
  - ✓修復ガイダンスが含まれた詳細レポート
  - ✓コンテキストを踏まえリスクスコアを算出し検出結果の優先順位付け

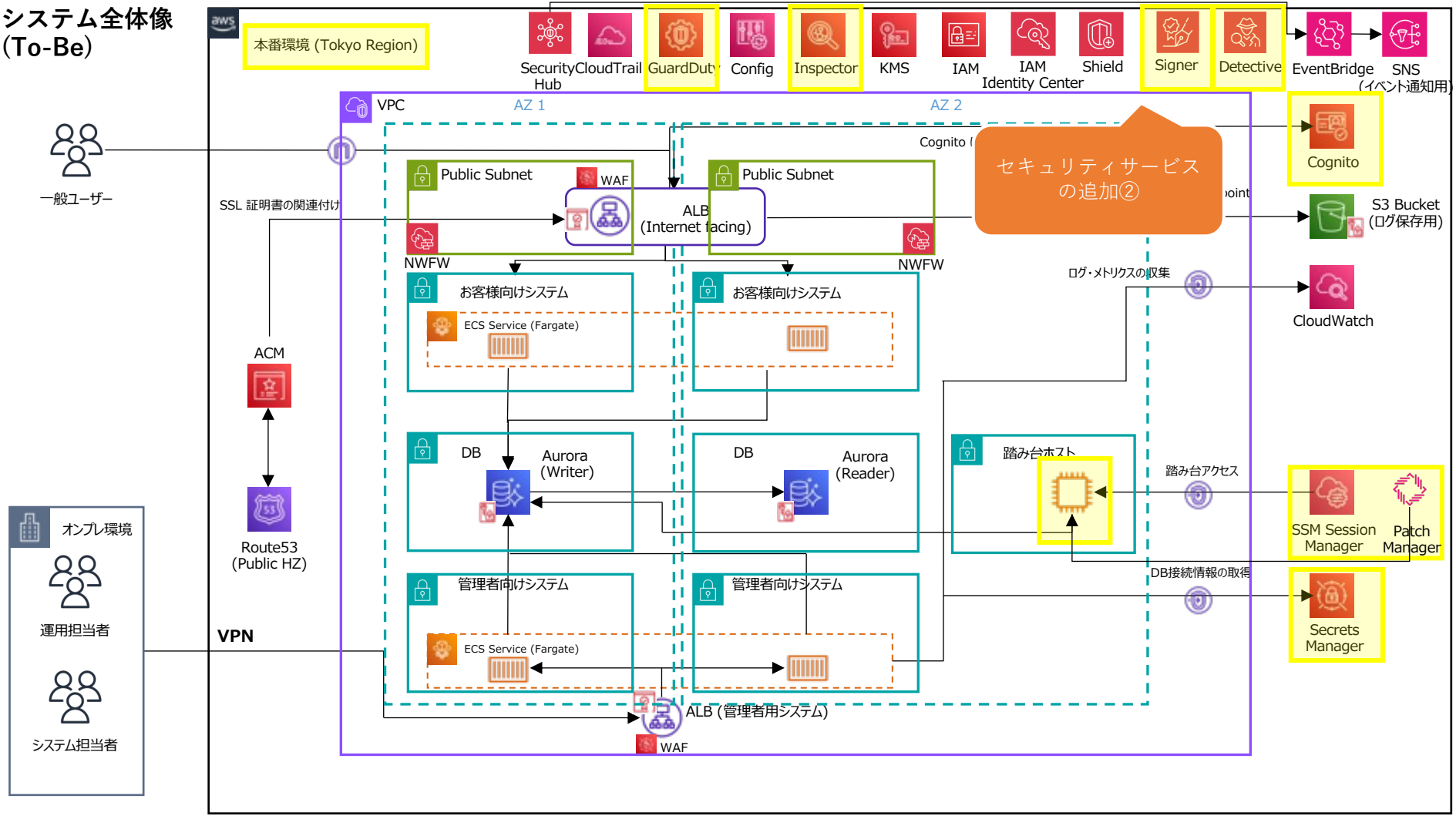
参考：ログおよびモニタリングによる検知ー概要

各種ログおよびモニタリングよりインシデントを検知

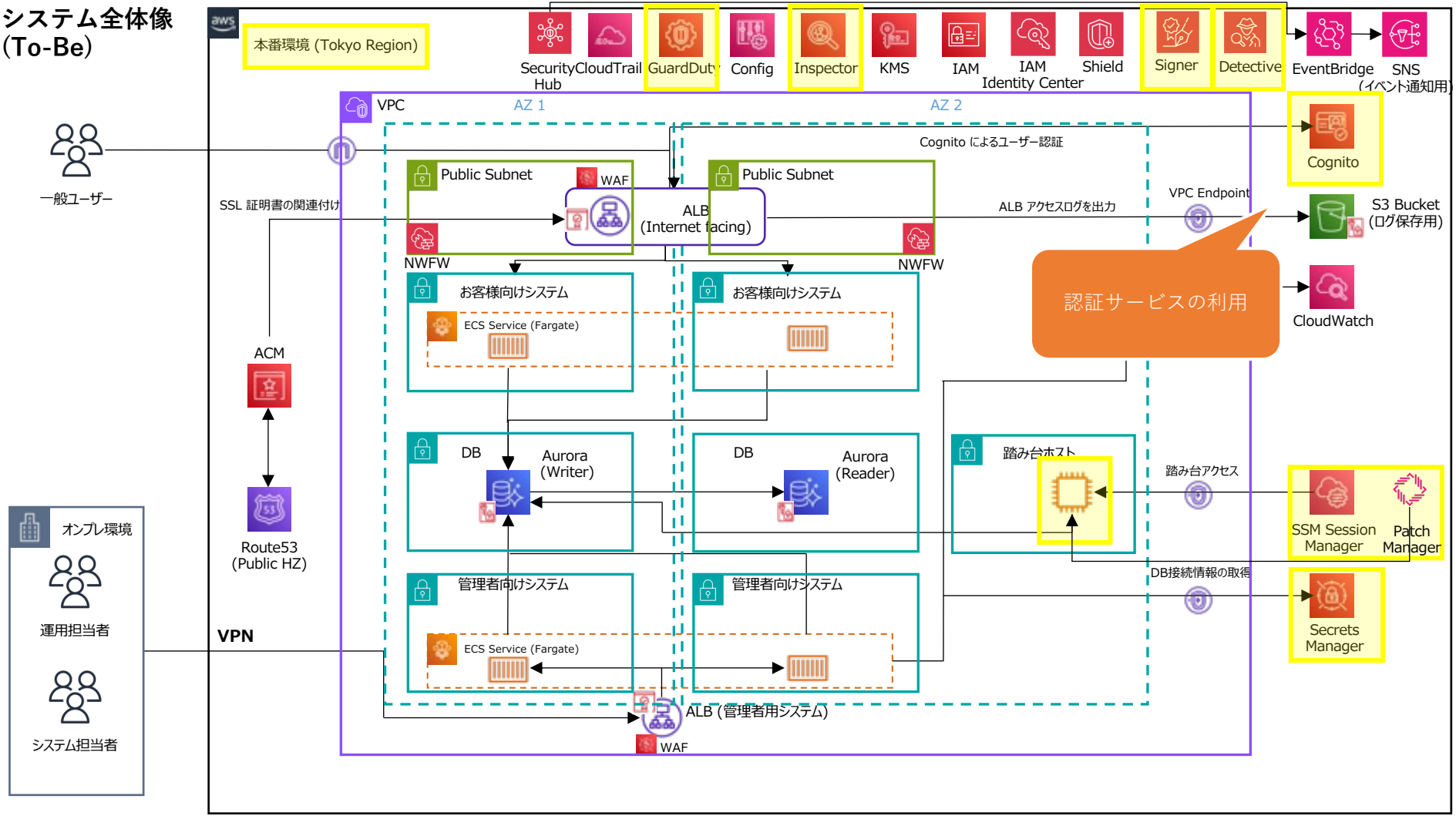


図の簡略化のため、上図は抜粋イメージ

システム全体像
(To-Be)



システム全体像
(To-Be)



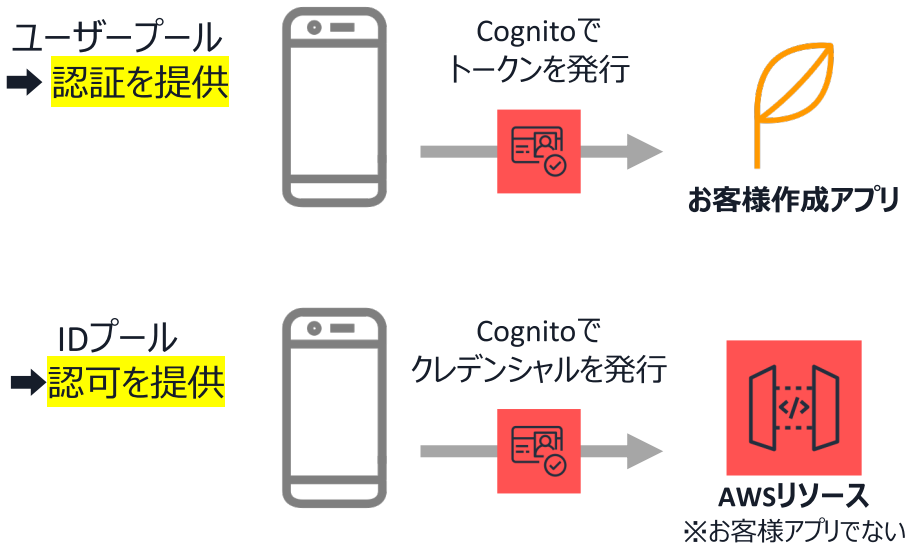
参考: Amazon Cognitoの概要

Cognitoの概要

- **AWSの認証・認可の提供サービス**
  - 完全マネージド型のモバイル、ウェブアプリ向けの認証・認可を提供
  - Amplify、AppSyncなどの別サービスとも統合されており、すぐにWeb、モバイルアプリに認証、認可を統合することができる
  - Cognito自身、もしくは外部IDプールと連携させることもできる
    - SAML、OAuthまたはOpenID Connectでのアクセスを提供
    - 多要素認証（MFA）の利用も可能

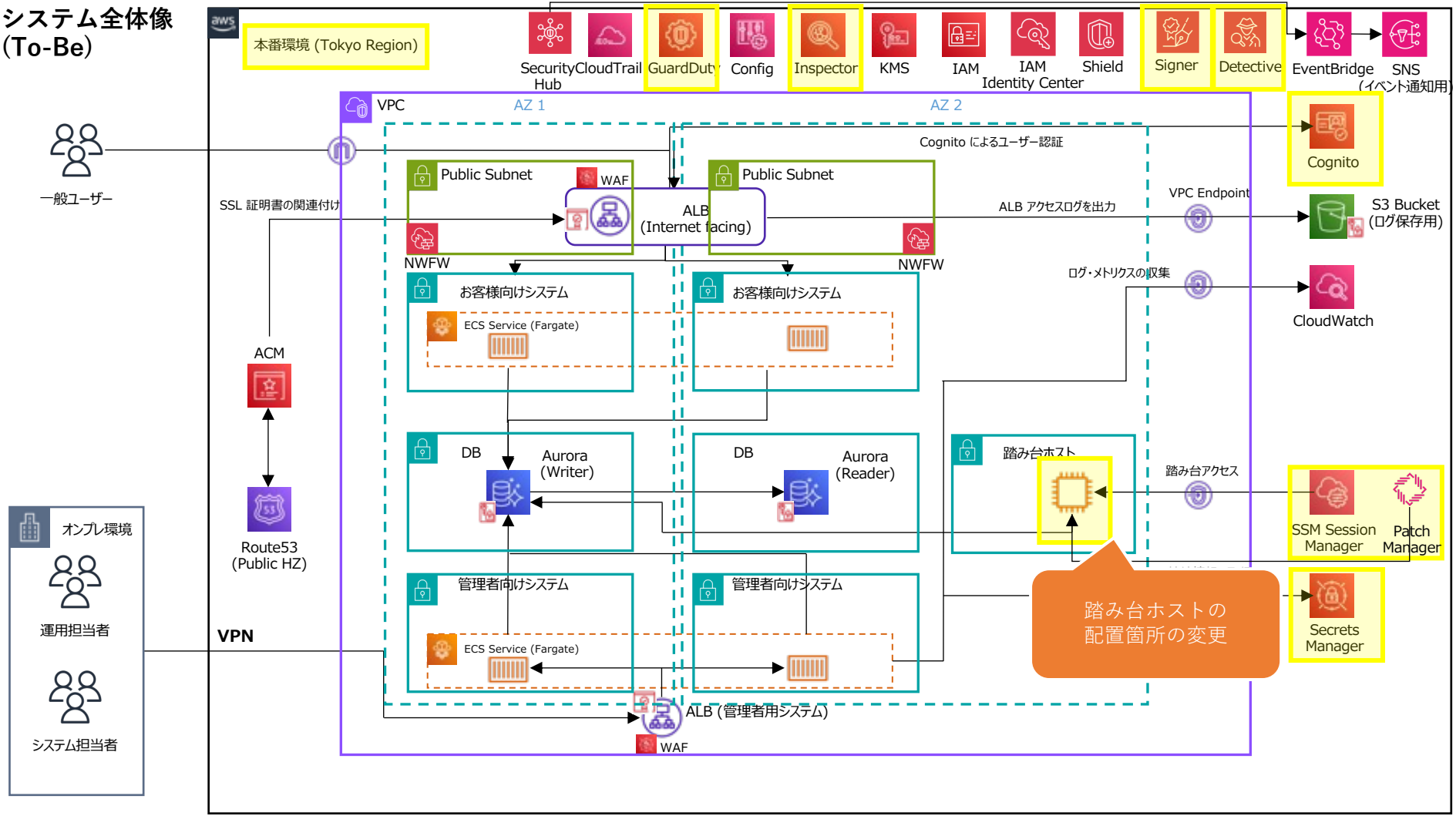
Cognitoで提供する機能

- **モバイル / Webアプリの認証・認可を提供**



aws professional services

システム全体像 (To-Be)

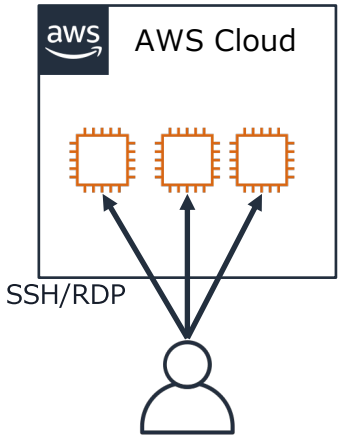


参考: インスタンスのOSへのアクセス

方式の説明

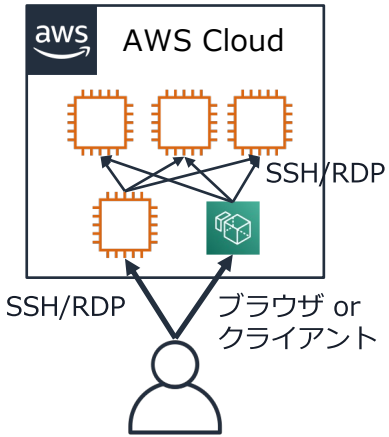
- **インスタンスのOSへのログイン方法は、以下が考えられる**
  - クライアントから直接
  - 踏み台経由
  - ブラウザ経由
    - Systems Manager

クライアントから直接



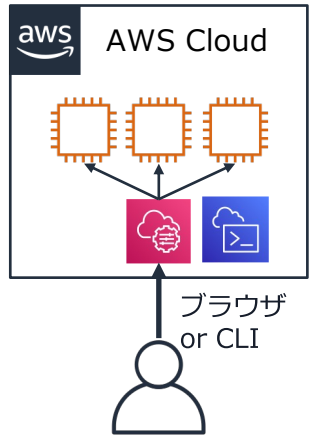
クライアントから対象インスタンスに直接ネットワーク的に到達可能である必要がある

踏み台経由



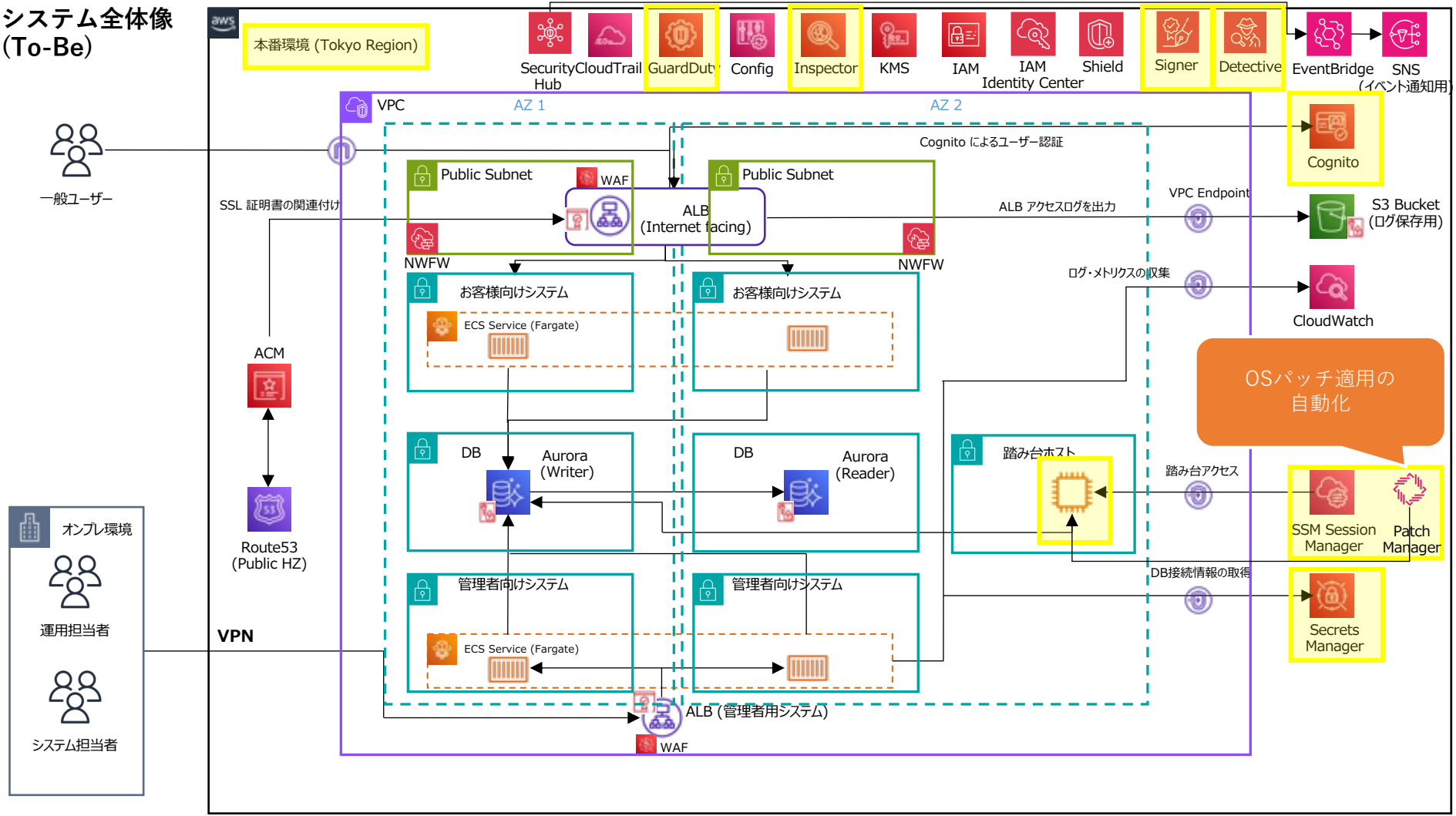
対象インスタンスに到達可能な位置に踏み台を配置する。踏み台の維持管理が必要となる

Session Manager 経由



マネジメントコンソールからSystems ManagerのSession Manager (またはFleet Manager) を使用

システム全体像 (To-Be)



参考：Patch Manager採用における考慮事項

検討にあたって

- Systems Manager／Patch Managerを使用すると簡単にパッチ配布運用を実装できる
- しかし、制約や作りこみが必要な要素も存在する

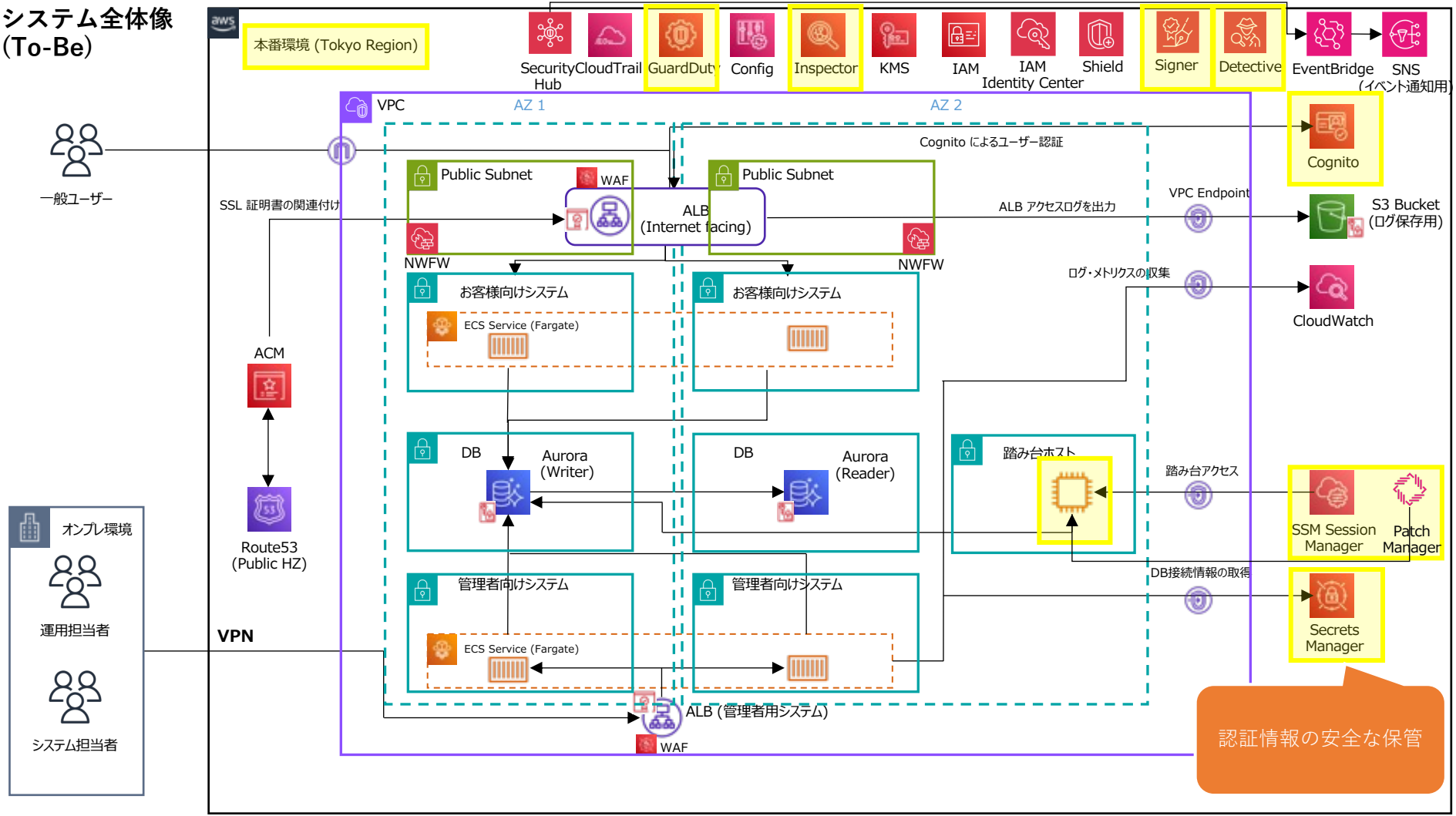
- 機械的に割り切って実施できる場合はPatch Manger、細かな制御をしたい場合は配布サーバを構成いただくことを推奨

※1 セキュリティに関連するパッチの選択方法
https://docs.aws.amazon.com/ja_jp/systems-manager/latest/userguide/patch-manager-how-it-works-selection.html

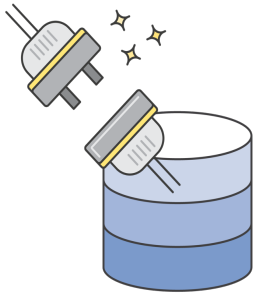
考慮事項

- **対象EC2はリポジトリにアクセスできる必要がある**
  - インターネットあるいは別途構築するリポジトリにアクセスできる必要がある(※1)
  - インターネットとの通信量にも注意
- **選定パッチの細かな制御はできない**
  - 設定条件に合うパッチを機械的に適用
  - Patch Managerは内部的に各OSのパッケージマネージャーを使用しており(yumなど※1)、細かな制御が必要な場合、直接パッケージマネージャーを使用する
- **配布はパッチに限られる**
  - 統合的な構成管理機能ではない
  - 対象OS（製品）はドキュメントを参照のこと
- **前後処理を含めるには作りこみが必要**
  - メンテナンスウィンドウは前後の状況に関係なく起動
  - 事前のAMIバックアップや静止点作成などの業務処理は事前定義済ドキュメントでは提供されない

システム全体像 (To-Be)

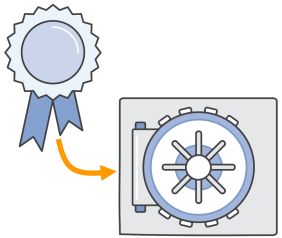


参考: AWS Secrets Manager のユースケース



アプリケーションからデータベースに接続

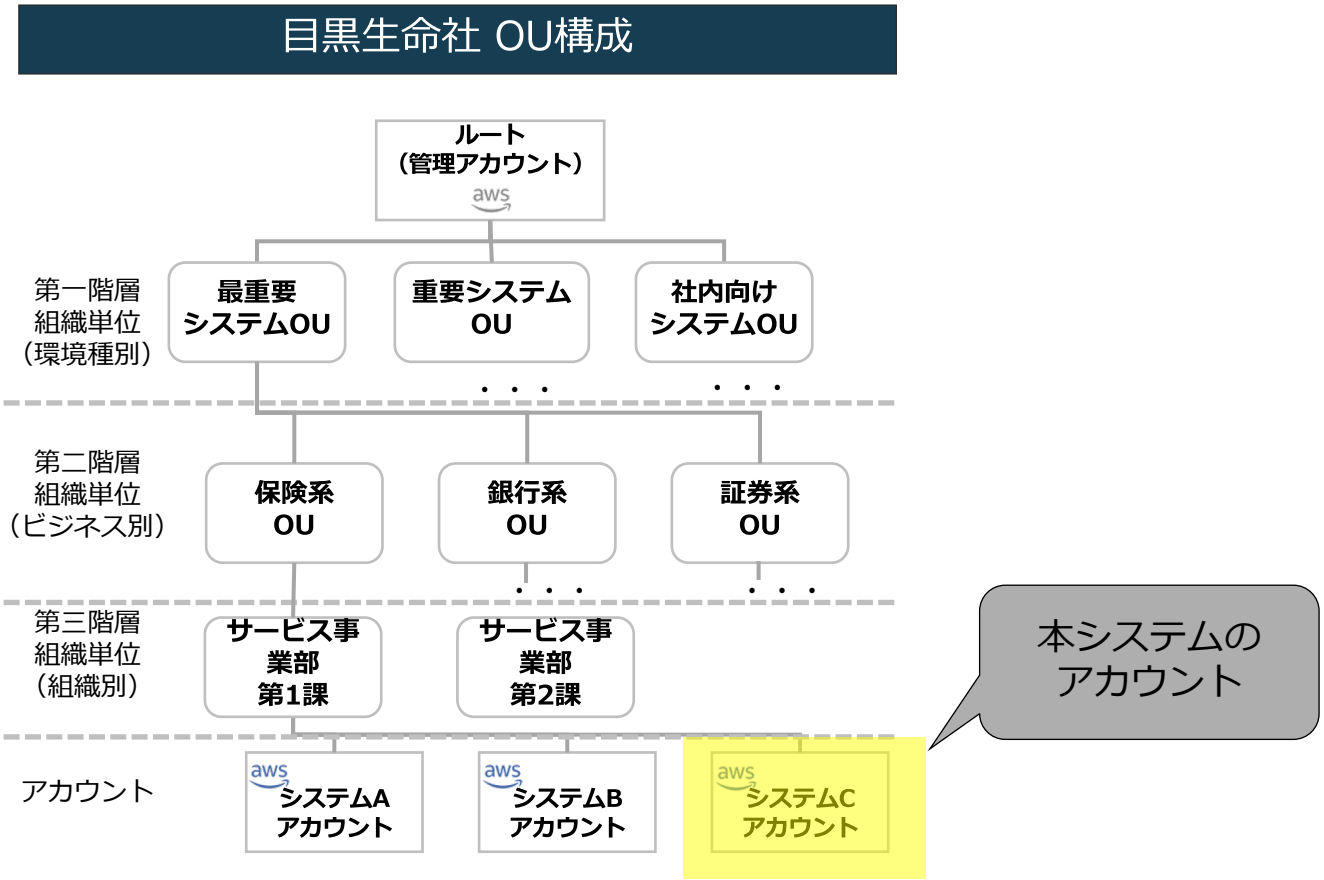
- DBA はアプリケーション用のデータベース認証情報を AWS Secrets Manager に保存
- DevOps エンジニアは AWS IAM ロールと一緒にアプリケーションをデプロイ
- アプリケーション起動処理で IAM ロールで付与された権限で Secrets Manager を呼び出して認証情報を取り出して、データベースにアクセスする



アプリケーションのシークレットを保存

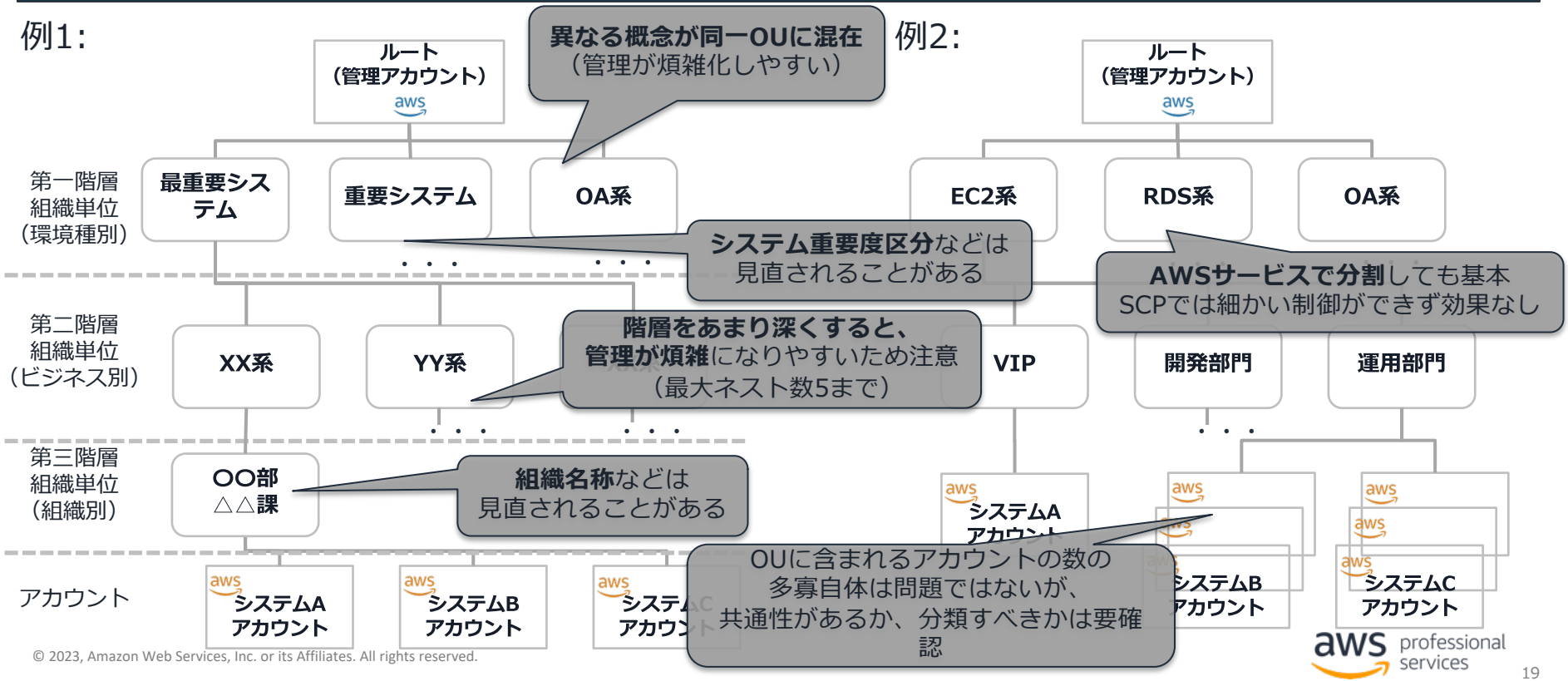
- アプリケーションは Open ID Connect を使用してユーザーを認証し、アクセストークンを使用してユーザーの代わりに API 呼出す
- アプリケーションは OAuth 更新トークンを Secrets Manager に保存する
- アプリケーションはアクセストークンの有効期限が切れた時にそれを使って新しいアクセストークンを取得する

Organizations組織構成(As-Is)

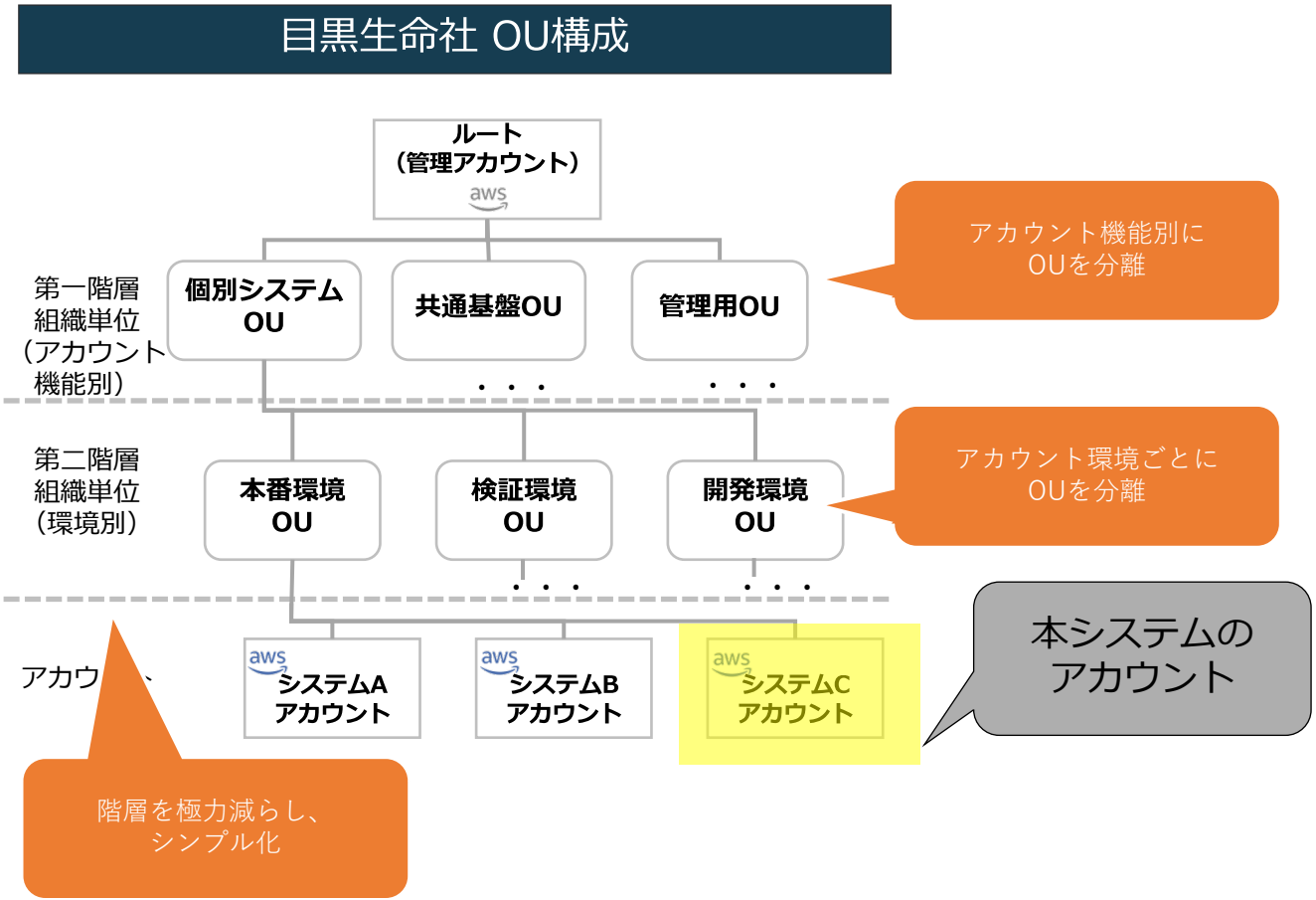


参考: AWS Organizations アンチパターン

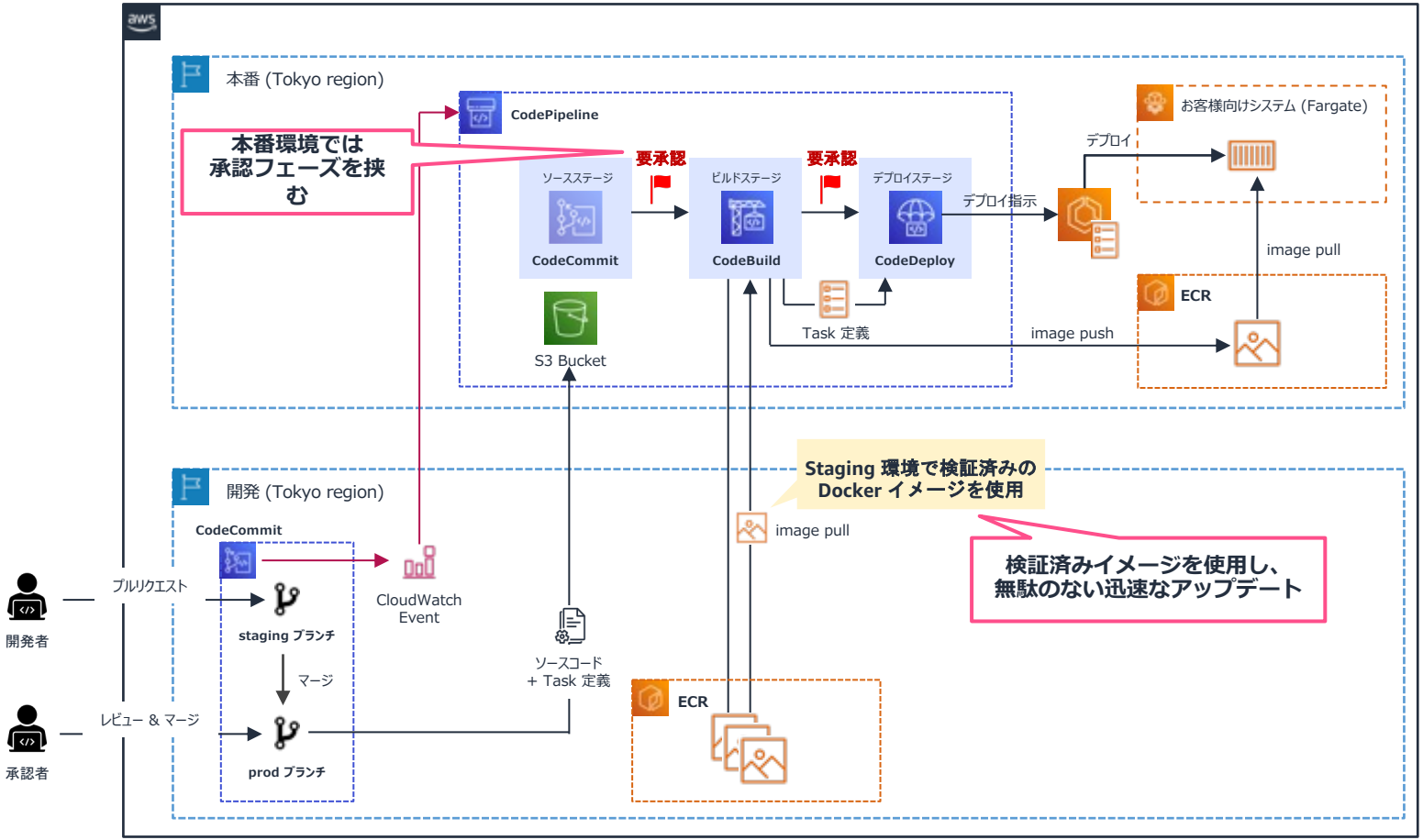
OUアンチパターン



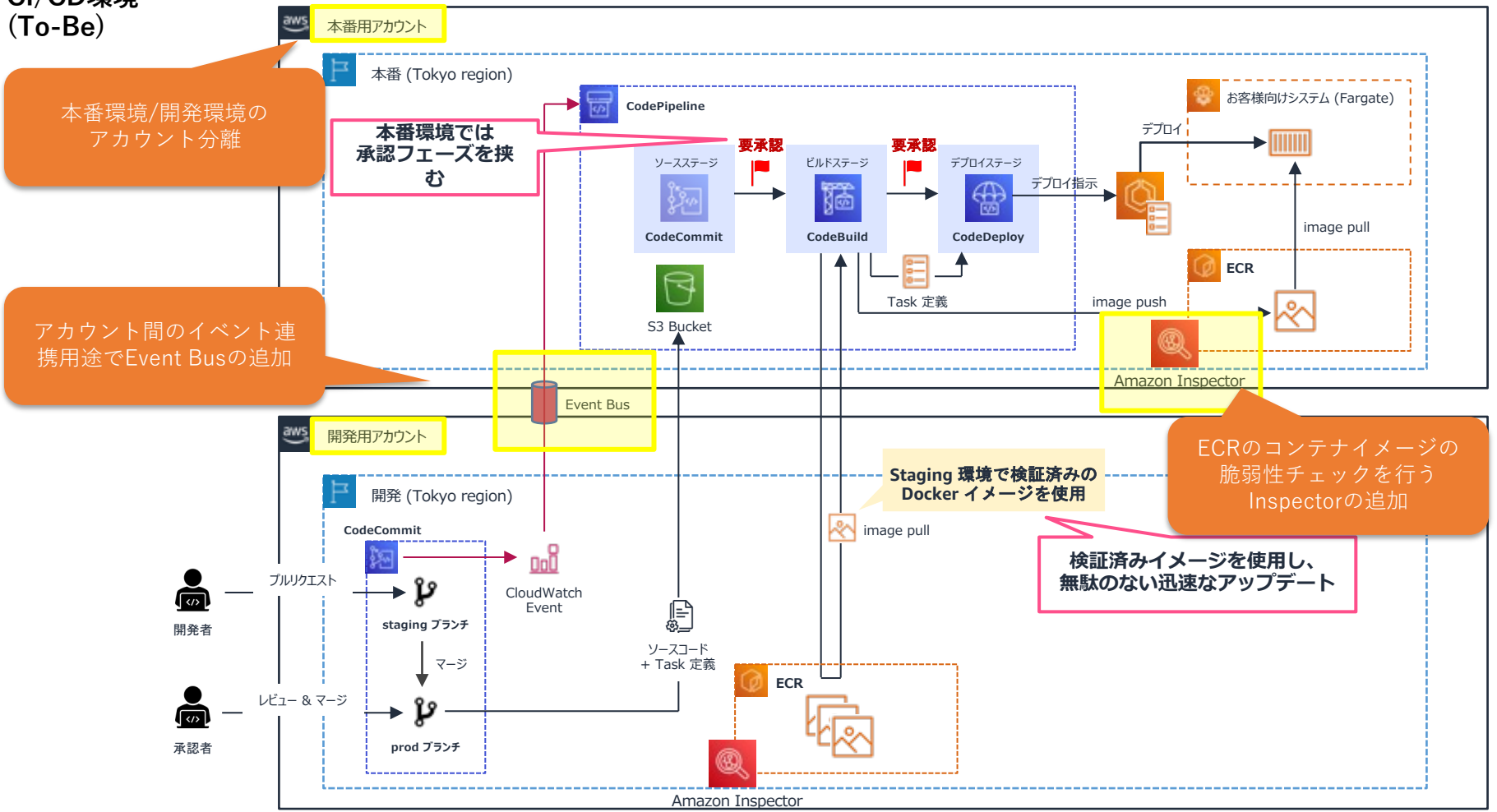
Organizations組織構成(To-Be)



CI/CD環境
(As-Is)


本番 (Tokyo region)
CodePipeline
本番環境では
承認フェーズを挟
む
ソースステージ
要承認
ビルドステージ
要承認
デプロイステージ
CodeCommit
CodeBuild
CodeDeploy
デプロイ指示
デプロイ
お客様向けシステム (Fargate)
image pull
ECR
S3 Bucket
Task 定義
image push
開発 (Tokyo region)
CodeCommit
CloudWatch
Event
ソースコード
+ Task 定義
image pull
Staging 環境で検証済みの
Docker イメージを使用
検証済みイメージを使用し、
無駄のない迅速なアップデート
ECR
プルリクエスト
開発者
staging ブランチ
マージ
レビュー & マージ
承認者
prod ブランチ

CI/CD環境
(To-Be)


本番環境/開発環境の
アカウント分離
本番用アカウント
本番 (Tokyo region)
本番環境では
承認フェーズを挟
む
CodePipeline
ソースステージ
CodeCommit
要承認
ビルドステージ
CodeBuild
要承認
デプロイステージ
CodeDeploy
デプロイ指示
お客様向けシステム (Fargate)
デプロイ
image pull
ECR
S3 Bucket
Task 定義
image push
Amazon Inspector
アカウント間のイベント連
携用途でEvent Busの追加
Event Bus
開発用アカウント
開発 (Tokyo region)
Staging 環境で検証済みの
Docker イメージを使用
ECRのコンテナイメージの
脆弱性チェックを行う
Inspectorの追加
CodeCommit
image pull
検証済みイメージを使用し、
無駄のない迅速なアップデート
プルリクエスト
開発者
staging ブランチ
CloudWatch
Event
ソースコード
+ Task 定義
マージ
レビュー & マージ
承認者
prod ブランチ
ECR
Amazon Inspector

参考：Inspectorによるスキャン

パッケージの脆弱性
- EC2
- ECRコンテナイメージ
- Lambda関数・レイヤー

ソフトウェアのCVE曝露をチェック
- 環境内のソフトウェアパッケージをスキャンして検出した脆弱性に該当するCVE(Common Vulnerabilities and Exposures)を示す。

スキャンのタイミング

EC2	ECR	Lambda

• Inspectorがインスタンスを検出する時
• 新しいインスタンスを開始した時
• 既存のインスタンスに新しいソフトウェアをインストールする時
• Inspectorがデータベースに新しい脆弱性 (CVE) を追加した時
	• リポジトリレベルでいずれかを選択
	  • オンプッシュスキャン：イメージをプッシュしたとき
	  • 継続スキャン:オンプッシュスキャン+自動連続再スキャン
	• Inspectorがデータベースに新しい脆弱性 (CVE) を追加した時
		• InspectorがLambdaを検出する時
		• 新しいLambdaをデプロイする時
		• 既存のLambdaを更新する時
		• Inspectorがデータベースに新しい脆弱性 (CVE) を追加した時

ネットワーク到達性
- EC2のみ

VPCの外から到達可能なポートをチェック

インターネットゲートウェイ、VPCピアリング、VPNなどVPC 外から到達可能なEC2インスタンスのTCP/UDPポートをチェックする(※Transit Gateway未サポート) 。検出結果では、管理が誤っているセキュリティグループ、NACL、IGWなど過度に許容されたネットワーク設定や、潜在的に悪意のあるアクセスを許可する可能性のあるネットワーク設定について説明する

スキャンのタイミング：24時間に一回実行

検出結果のステータス

Inspectorは発見した脆弱性とネットワーク露出を自動的に追跡・保存する
- ステータスはactive, suppressed, closedの3種類
- 環境内を継続的にスキャンしてアクティブな検出結果を監視する。脆弱性への対応が完了すると、Inspectorは自動的に検出結果のステータスをclosedに変更