AWSでのセキュリティ対策全部盛り [初級から中級まで]

AWS事業本部　コンサルティング部
臼田佳祐

classmethod

自己紹介

臼田　佳祐

- クラスメソッド株式会社
- AWS事業本部
  ソリューションアーキテクト
  セキュリティチームリーダー
- Security-JAWS運営
- 好きなサービス:
  AWS WAFマネージドルール

AWS WAF

aws CERTIFIED

- Solutions Architect - Associate
- Developer - Associate
- SysOps Administrator - Associate
- Solutions Architect - Professional
- DevOps Engineer - Professional
- Security - Specialty
- Advanced Networking - Specialty

広く浅くではなく
ぼちぼち深くいきます

長い前説

セキュリティ対策って大変ですよね

セキュリティ対策いろいろ

サイバーセキュリティ

機密性/完全性/可用性

ガバナンス/コンプライアンス

発見的統制

リスクマネジメント

脆弱性管理

監視

認証/認可

侵入防御

ログ分析

このセッションの目的

いろんな人に網羅的にAWSのセキュリティを伝えて少しでもたくさん持って帰ってもらう->実践してもらう

対象者

- 開発者
- 運用担当
- セキュリティ担当
- アナリスト
- CISO
- 監査
- などなど
- つまりAWSに関わる人すべて

持ち帰ってほしいこと

- 資料は公開されているから後から熟読・社内展開
- クラウドは武器
- クラウドセキュリティも武器
  - アジリティを落とさない攻めのセキュリティ
- ゲートからガードレールへ
- すべての人がBuilders
- セキュリティはみんなでやるので直接関係ないと思う内容も聞いてみて

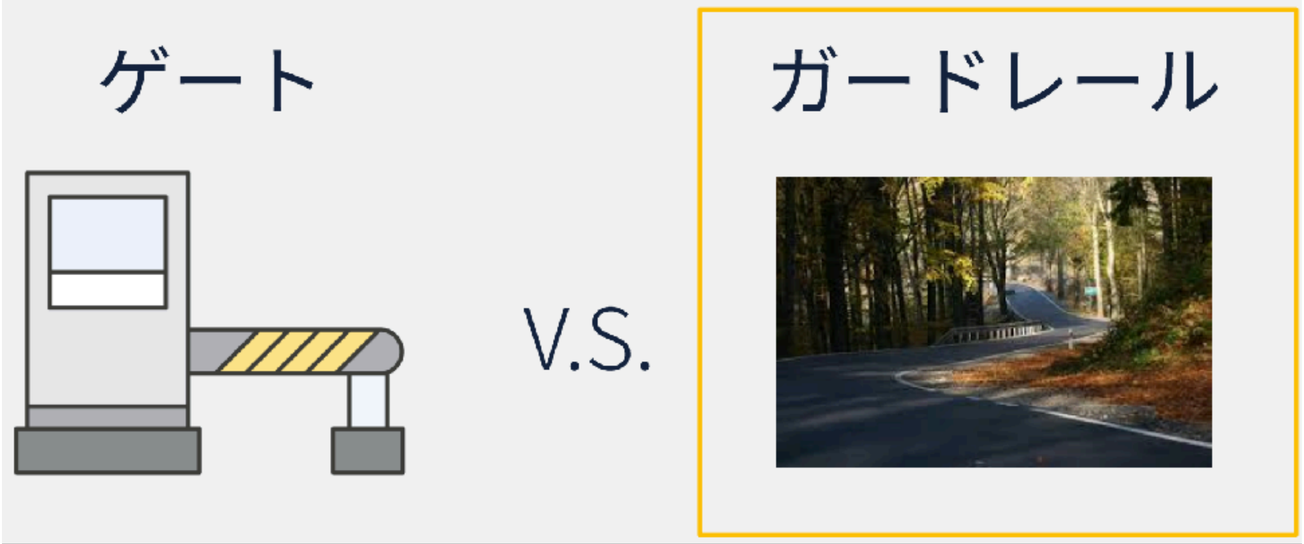
ゲートからガードレールへ

最近AWSがよく言うメッセージ

セキュリティはビジネスにブレーキをかけてはいけない

ガードレールのように道から逸れないように危ない操作を出来ないようにしたりすぐ是正できるよう監視/自動修復する

とりあえず一括で禁止しない

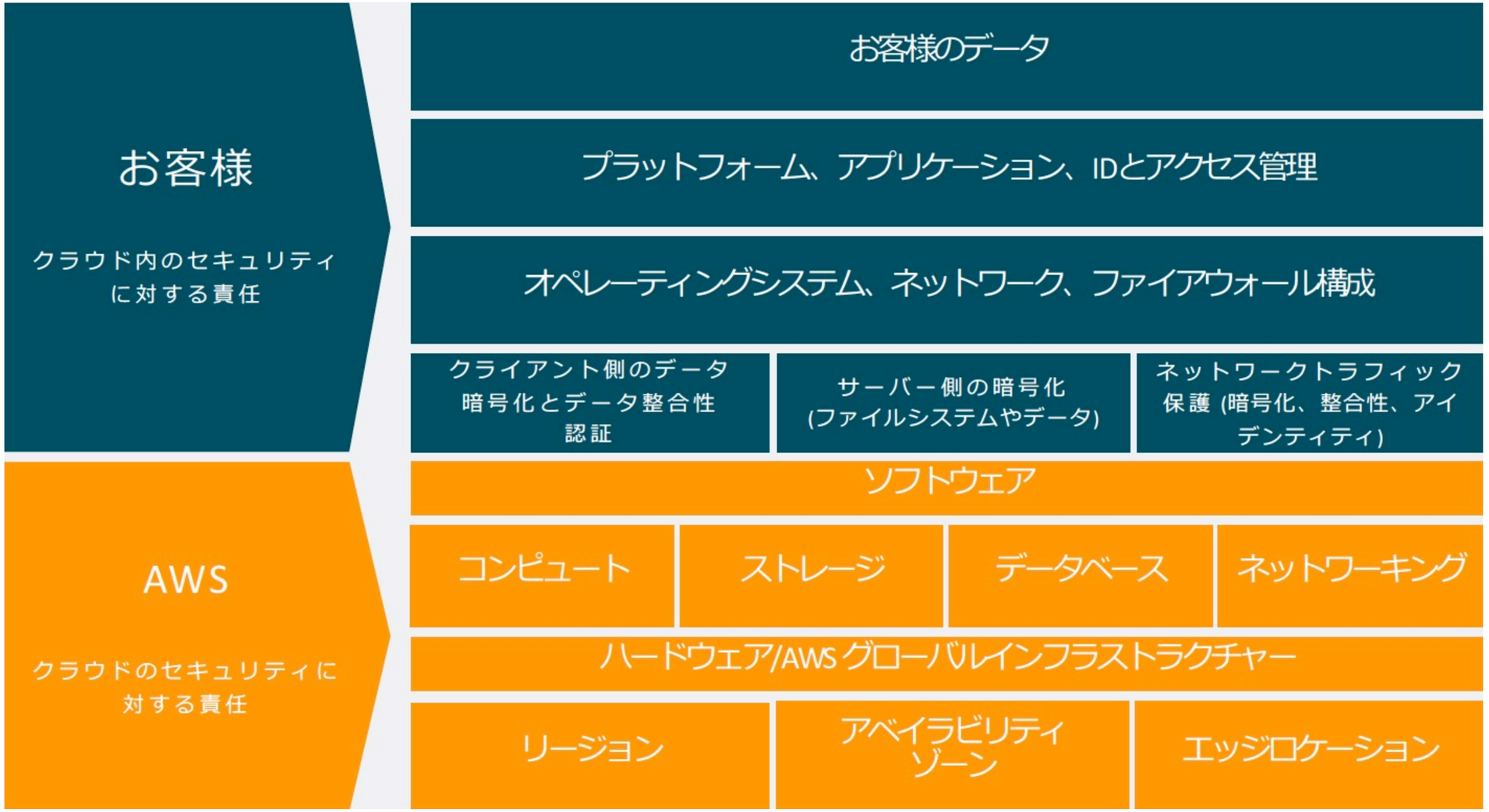


みんなBuilders

- AWSではサービスに関わる全ての人をBuildersと呼ぶ
- 開発者だけではない
- セキュリティ担当者も意思決定者も監査人もAWSのサービスを駆使してセキュリティを高めたりサービスの質を高めたりコンプライアンスチェックを自動化したりできる
- だからみんなBuilders
- BuildersはAWSを使いこなすためにいっぱい学んでください

長い前説
前提情報

AWSセキュリティの前提 - 責任共有モデル

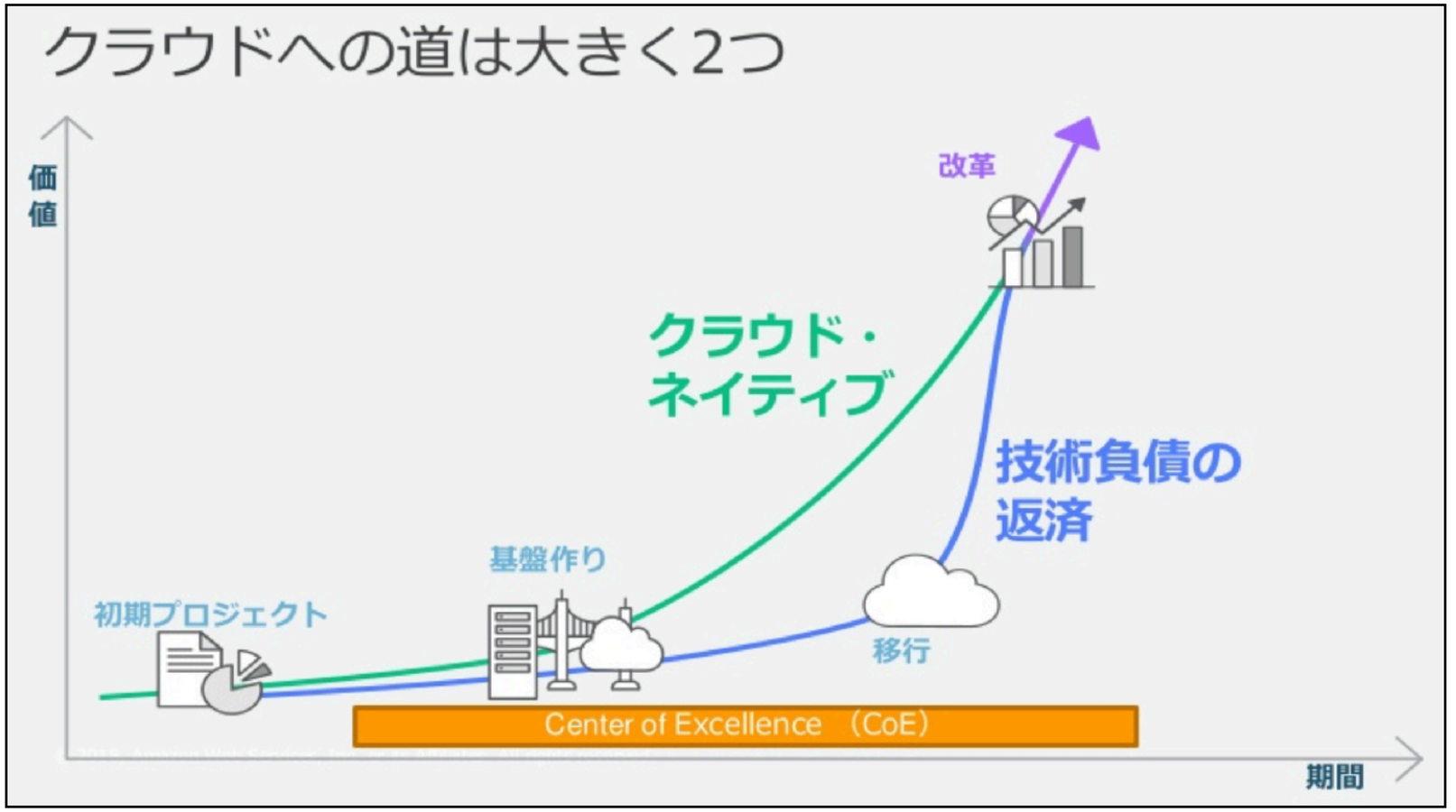


守るべき部分は明確

レベルに合わせた対策 - クラウドジャーニー

クラウド対応の成熟度に合わせて適切な対策を

CCoE等を中心にどこまでやるか決めましょう



参考にするドキュメントいろいろ

- NIST サイバーセキュリティフレームワーク(CSF)
  - https://aws.amazon.com/jp/blogs/news/updated-whitepaper-now-available-aligning-to-the-nist-cybersecurity-framework-in-the-aws-cloud/
- AWSセキュリティベストプラクティス
  - https://d1.awsstatic.com/whitepapers/ja_JP/Security/AWS_Security_Best_Practices.pdf
- Well-Architectedフレームワーク
  - https://aws.amazon.com/jp/blogs/news/aws-well-architected-whitepaper/
- PCI/FISC/HIPPA等各種コンプライアンスのドキュメント
  - https://aws.amazon.com/jp/compliance/programs/

かいつまんで解説

NIST サイバーセキュリティフレームワーク(CSF)

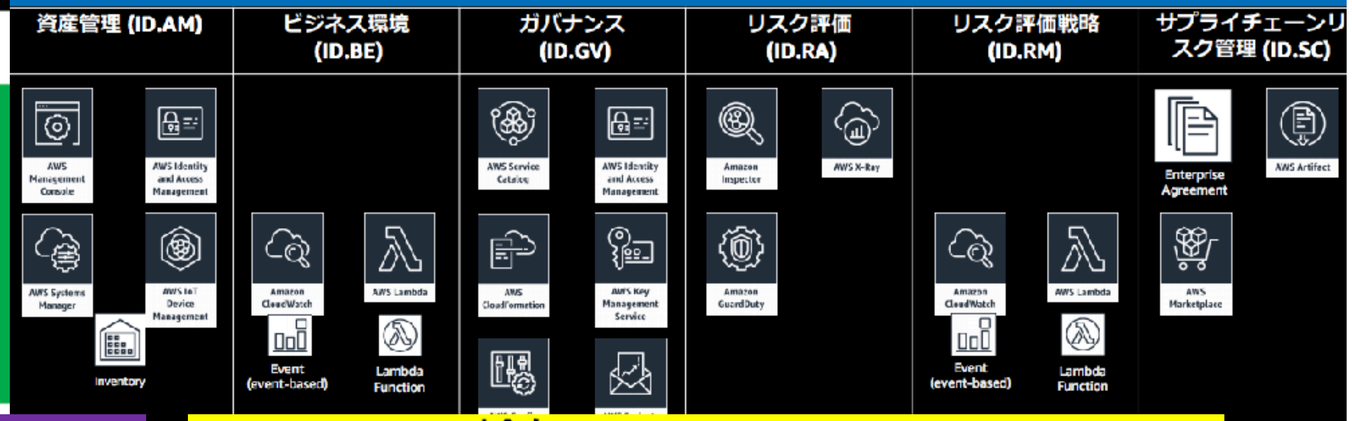
- 世界各地の政府、産業界、組織において参照されているサイバーセキュリティの推奨ベースライン
- ガートナー社によれば、CSF は米国の民間セクターの約30%で利用
- 日本でもIPAのホームページから邦訳版を入手可能
- コア機能は5つ
  - 識別 / 防御 / 検知 / 対応 / 復旧

AWSでのCSF対応表(一例)

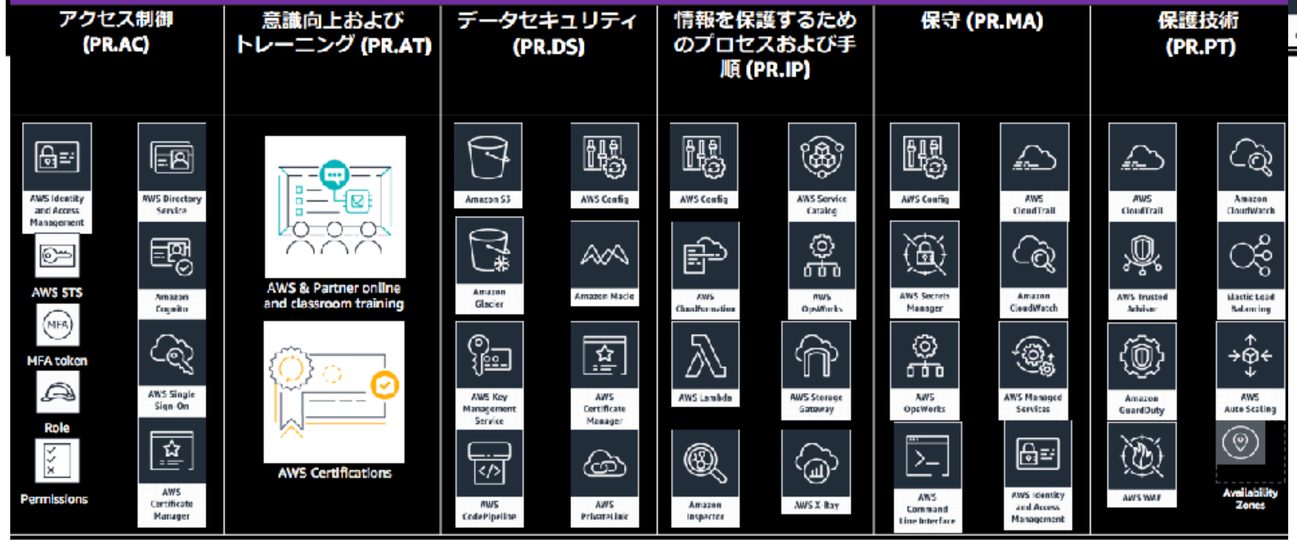
NIST CSFの構成

識別	防御	検知	対応	復旧
資産管理	アクセス制御	異常とイベント	対応計画の作成	復旧計画の作成
ビジネス環境	意識向上および トレーニング	セキュリティの継続的な モニタリング	コミュニケーション	改善
ガバナンス	データセキュリティ	検知プロセス	分析	コミュニケーション
リスク評価	情報を保護するための プロセスおよび手順		低減	
リスク評価戦略	保守		改善	
サプライチェーン リスク管理	保護技術			

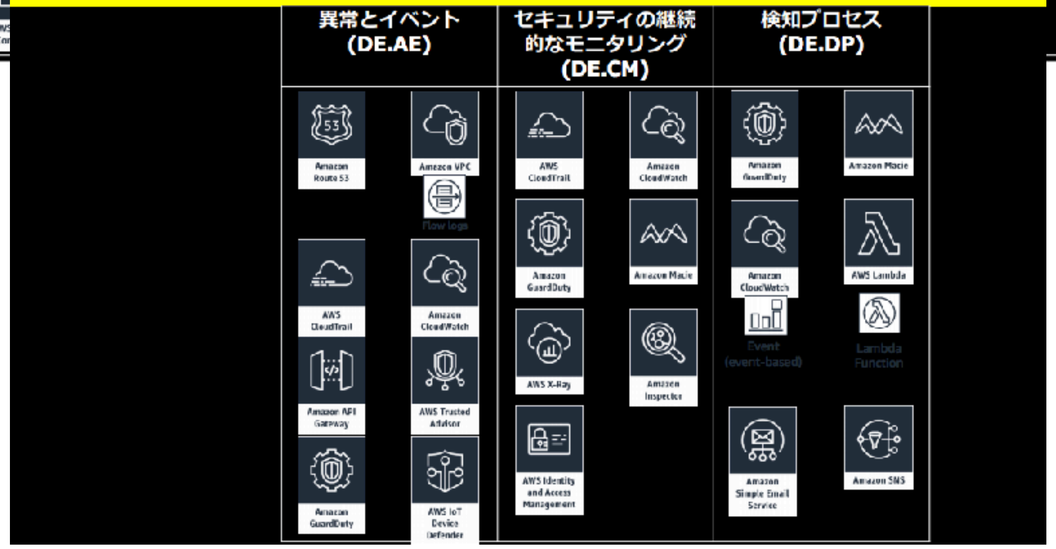
NIST CSF: 識別



NIST CSF: 防御



NIST CSF: 検知



https://pages.awscloud.com/rs/112-TZM-766/images/D2-06.pdf

Well-Architected フレームワーク

- AWSのベストプラクティス集
- 5 つの柱がある(運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化)
- セキュリティの柱の中にも5つの要素
  - アイデンティティ
  - 発見的統制
  - インフラ保護
  - データ保護
  - インシデントレスポンス

前説が長くなりましたが今回の進め方

今回はレベル順: 初級->中級の流れで説明

W-AやCSFなどの参考成分は

ごちゃまぜで説明します

(きれいに分けれないので)

初級

初級の内容

- AWSレイヤー基礎
- OS/アプリレイヤー基礎
- 運用

初級
AWSレイヤー基礎

AWSレイヤー基礎

- IAM
- CloudTrail / AWS Config
- GuardDuty
- AWS Shield
- VPC
- S3
- バックアップ

AWSセキュリティベストプラクティスとその補足

AWSセキュリティベストプラクティスを実践するに当たって適度に抜粋しながら解説・補足した内容を共有します

VPC IAM

2019年09月09日 臼田佳祐（105） 397

https://dev.classmethod.jp/cloud/aws/explanation-aws-security-best-practices/

AWSご利用開始時に最低限おさえておきたい10のこと

https://www.slideshare.net/AmazonWebServicesJapan/day-1-with-amazon-web-services-aws10

IAMの最重要事項

- IAMユーザにはMFA必須
- アクセスキーをコードに埋め込まない(IAMロールを利用する)
- コードを扱う端末すべてにgit-secretsを導入する
- 最小権限を意識する
- AWS利用ユーザすべてにIAMの扱いについて教育する

あわせて読みたい

AWS再入門 AWS IAM (Identity and Access Management) 編

https://dev.classmethod.jp/cloud/aws/cm-advent-calendar-2015-getting-started-again-aws-iam/

AWSマネジメントコンソール上と連動しながら説明されていて、画面を触るイメージが掴めるのでオススメ

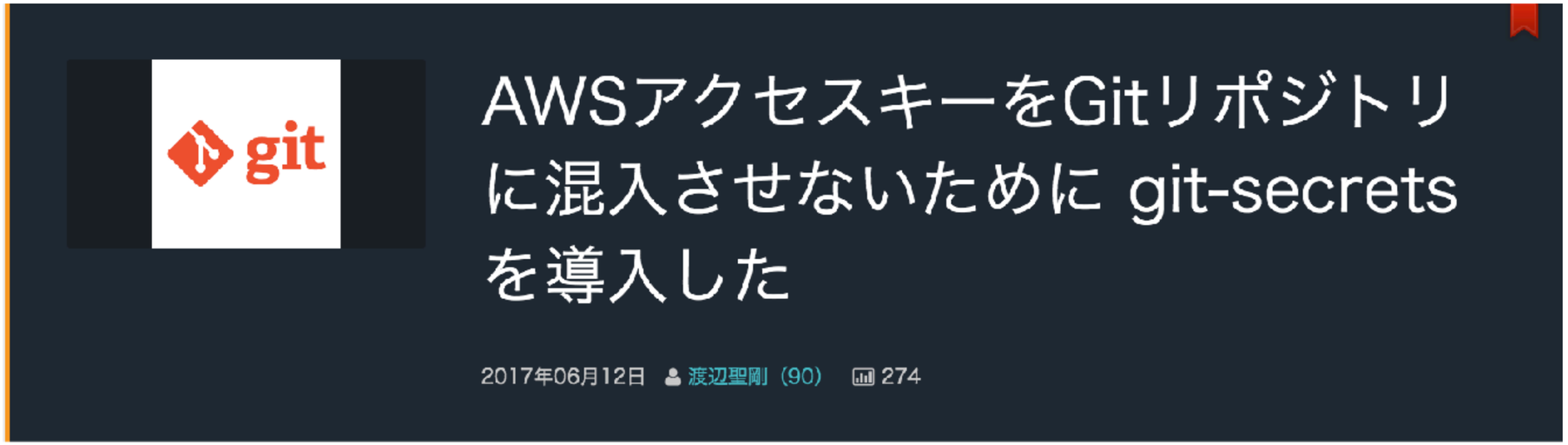
IAM のベストプラクティス

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

AWSが出しているさらにIAMに特化したベストプラクティス
本当にがっしり情報が詰まっているのでたまに振り返ってみるといいです

あわせて読みたい

AWSアクセスキーをGitリポジトリに混入させないために git-secrets を導入した



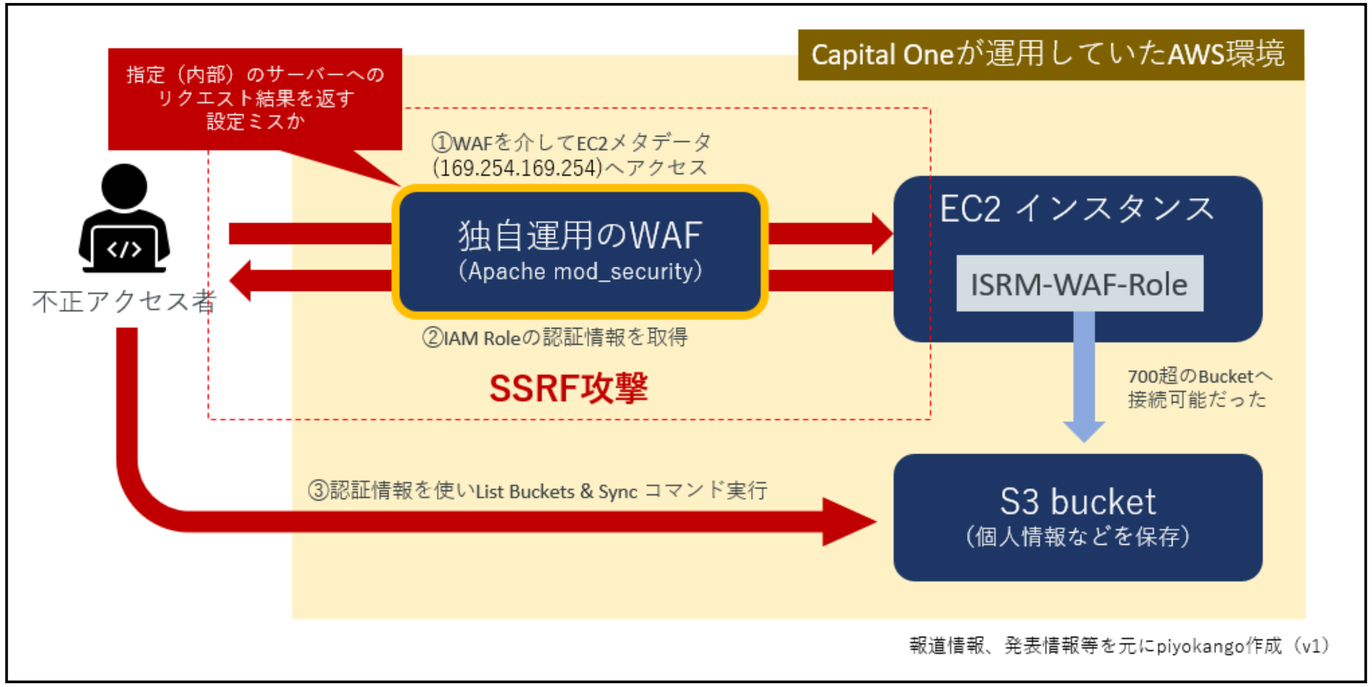
AWSで一番多い事故を防げるので必須

https://dev.classmethod.jp/cloud/aws/startup-git-secrets/

最近の事故事例 Capital One

原因の入り口は独自構築したEC2上のWAF

しかし流出したクレデンシャルに700を超えるS3 Bucketの情報を取得する権限があることが問題



最小権限を意識する

piyologさんから借用

https://piyolog.hatenadiary.jp/entry/2019/08/06/062154

あわせて読みたい

AWSの薄い本　IAMのマニアックな話

佐々木拓郎氏(@dkfj)の同人誌

IAMの基礎からポリシーの秘伝のタレまで惜しみなく紹介されている一冊

ダウンロード版があるので今すぐ購入すべし

https://booth.pm/ja/items/1563844

利用開始時に有効化するサービス

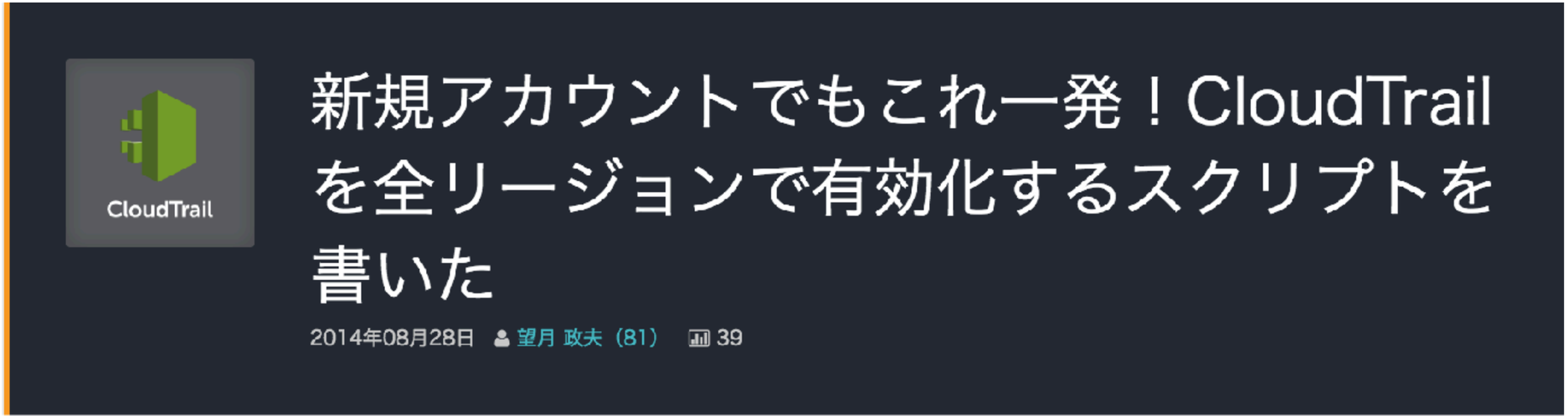
- AWS CloudTrail
  - AWSに対するAPIコールを記録する
- AWS Config
  - AWSリソースベースの変更履歴を記録する
- Amazon GuardDuty
  - AWSに対する不審な動きを検知する
  - 不正なログインやコインマイニングなどの脅威を検知
- これらは多少お金がかかっても必須(何かあったら取り返しがつかないので)

あわせて読みたい

新規アカウントでもこれ一発！CloudTrailを全リージョンで有効化するスクリプトを書いた

https://dev.classmethod.jp/cloud/aws/cloudtrail-activate/

まだ有効になっていないアカウントは一発入れておきましょう



あわせて読みたい

AWS Configはとりあえず有効にしよう

https://dev.classmethod.jp/cloud/aws/aws-config-start/

あわせて読みたい

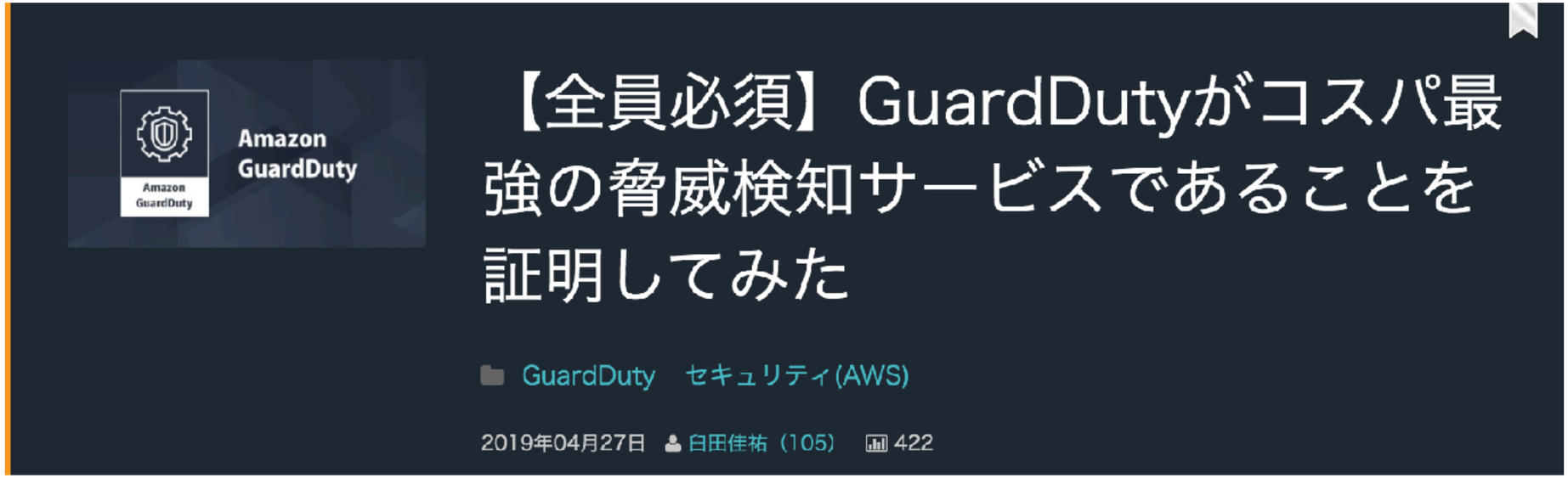
一発でGuardDutyを全リージョン有効化して通知設定するテンプレート作った

https://dev.classmethod.jp/cloud/aws/set-guardduty-all-region/

まずは一発有効化！

GuardDutyはコスパ最強

【全員必須】GuardDutyがコスパ最強の脅威検知サービスであることを証明してみた



もしどうしてもGuardDutyの導入を妨げるものが現れたらこのブログをお使いください

https://dev.classmethod.jp/cloud/aws/guardduty-si-strongest-thread-detection/

AWS Shield

マネージドなDDoS保護サービスで
無料で最初からL3/L4レベルが保護されている
Standardプランと、AWSのDDoS専門チーム
と連携してL7や大規模なDDoSに対抗する
Advancedプランがある

Standardはとくに設定の必要なし

- 環境ごとにVPCやアカウントを分割する
  - AWSアカウントごと分割すればIAMも分けれる
- Security GroupとNACLを適切に利用する
  - 基本はSecurity Groupで絞る
  - NACLはサブネット全体で必要なものだけ
- VPC内からAWSサービスを利用する場合はVPC Endpoint(Private Link)を利用する
  - インターネットを経由しないAPI通信を実現

あわせて読みたい

AWSアカウントとVPC、分ける？ 分けない？: 分割パターンのメリット・デメリット

https://dev.classmethod.jp/cloud/aws/account-and-vpc-dividing-pattern/

アカウント分割(戦略)の殿堂入り記事

AWS Trusted Advisorでセキュリティリスクの監視　39

AWS Trusted Advisorは「コスト最適化」、「パフォーマンス」、「セキュリティ」、「フォールトトレーランス」の４つの観点から、利用者のAWS環境をAWSが自動で精査し、推奨設定のお知らせをしてくれる機能

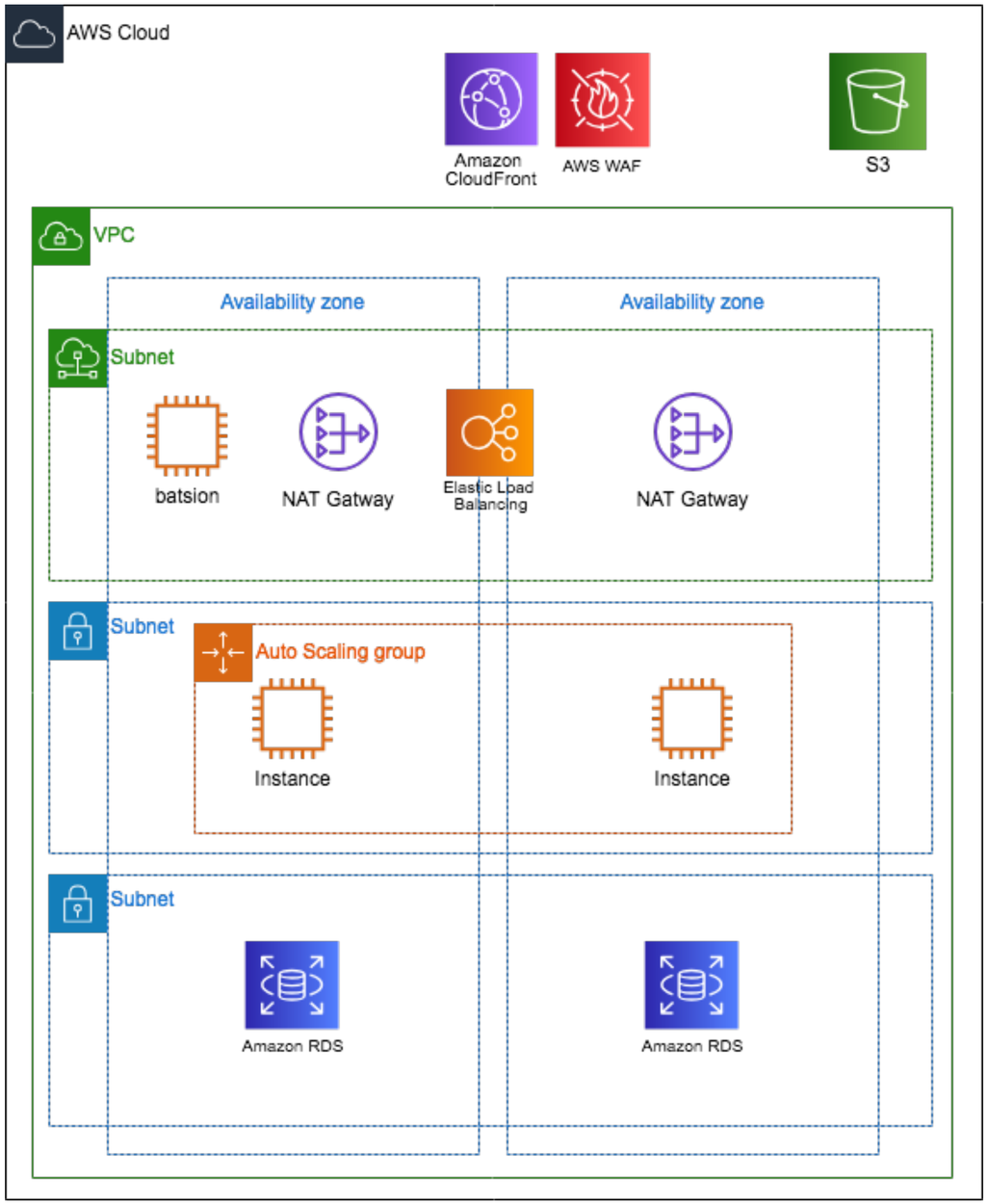
IAMやS3の設定などが問題ないかチェックしてくれる

https://dev.classmethod.jp/cloud/aws/cm-advent-calendar-2015-getting-started-again-aws-td/

JAWS DAYS 2018の登壇資料を参考にしてください

https://dev.classmethod.jp/cloud/aws/jaws-days-2018-i-security/

一般的にセキュアな構成



- 三層ネットワーク
  - フロント以外はプライベート
- マルチAZ
- AutoScaling
- 外部通信はNAT経由
- 必要に応じCloudFrontやWAFを導入

あわせて読みたい

「[初心者向け]AWS環境のセキュリティ運用(設計)をはじめてみよう」JAWS DAYS 2019登壇資料

VPC周りのアーキテクチャやその運用についてはこちら

https://dev.classmethod.jp/cloud/aws/jaws-days-2019-a-security/

- S3のアクセス制御の要素は大きく4つ
  - IAMポリシー
  - バケットポリシー
  - ACL
  - パブリックアクセスブロック

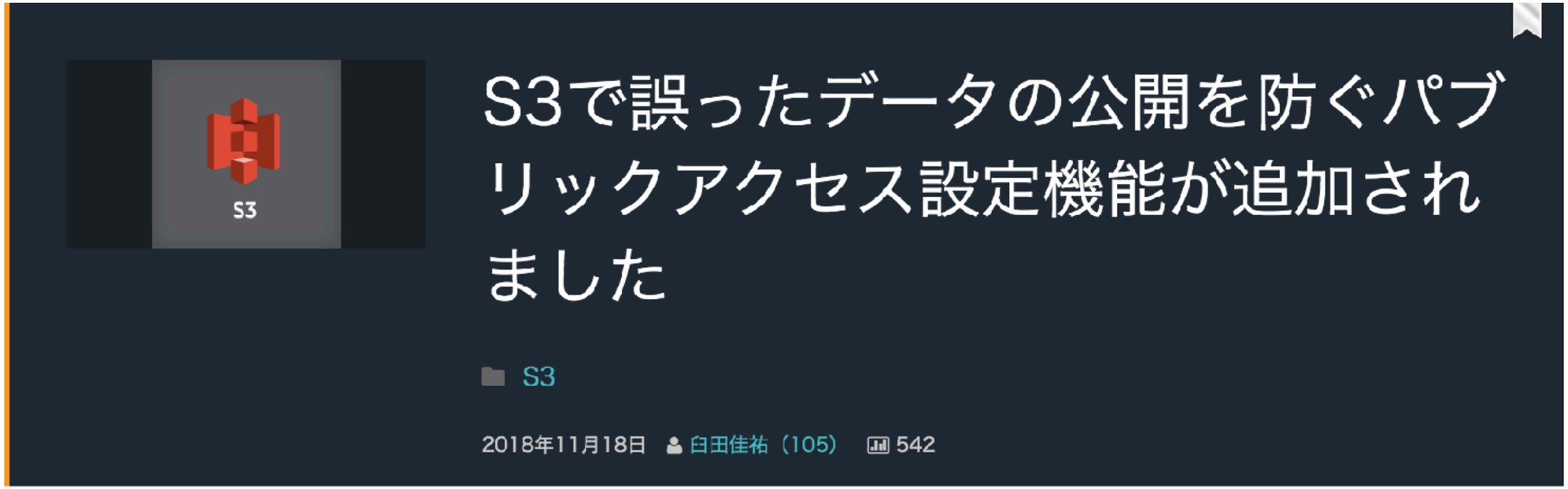
- IAMユーザ/ロールにアクセス権を与える時はResourceでバケットを最小限に限定する
  - 特にEC2 / Lambda
- バケットポリシーとACLで最小権限を意識する
  - IP制限など絞りすぎて自分がアクセスできなくなるのは気をつける
- パブリックアクセスブロックで外部からのアクセスを多重に防ぐ

S3のセキュリティベストプラクティス

https://docs.aws.amazon.com/ja_jp/AmazonS3/latest/dev/security-best-practices.html

- Amazon S3 での予防的セキュリティのベストプラクティス
  - アクセス制御や暗号化など
- Amazon S3 のモニタリングと監査のベストプラクティス
  - ロギングなど

S3で誤ったデータの公開を防ぐパブリックアクセス設定機能が追加されました



https://dev.classmethod.jp/cloud/aws/s3-block-public-access/

バックアップ

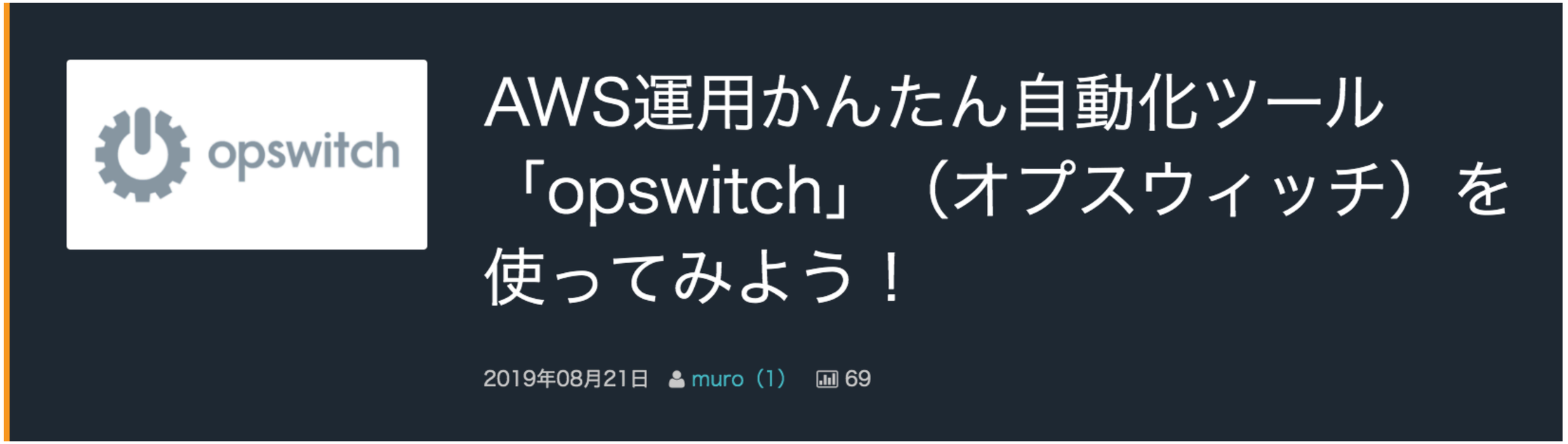
- EC2やRDSなどのバックアップはきちんと取ろう(もちろん冗長化も)
- 先日の障害で復旧しなくなったEBSがあったので、バックアップ設定はユーザ責任であることを強く意識する
- 自動バックアップにはAmazon DLMやAWS Backupなどがあるが、弊社的には弊社提供のOpswitchがオススメ！

Opswitch（オプスウィッチ）

- できること
  - バックアップ作成
  - インスタンスの起動停止
  - リージョン感コピー
  - リソース止め忘れチェック
  - 柔軟なスケジュール実行
- 複数AWSアカウントを統合的に管理できる
- クラスメソッドメンバーズのプレミアムサービス契約済みなら追加費用無しで利用可能

あわせて読みたい

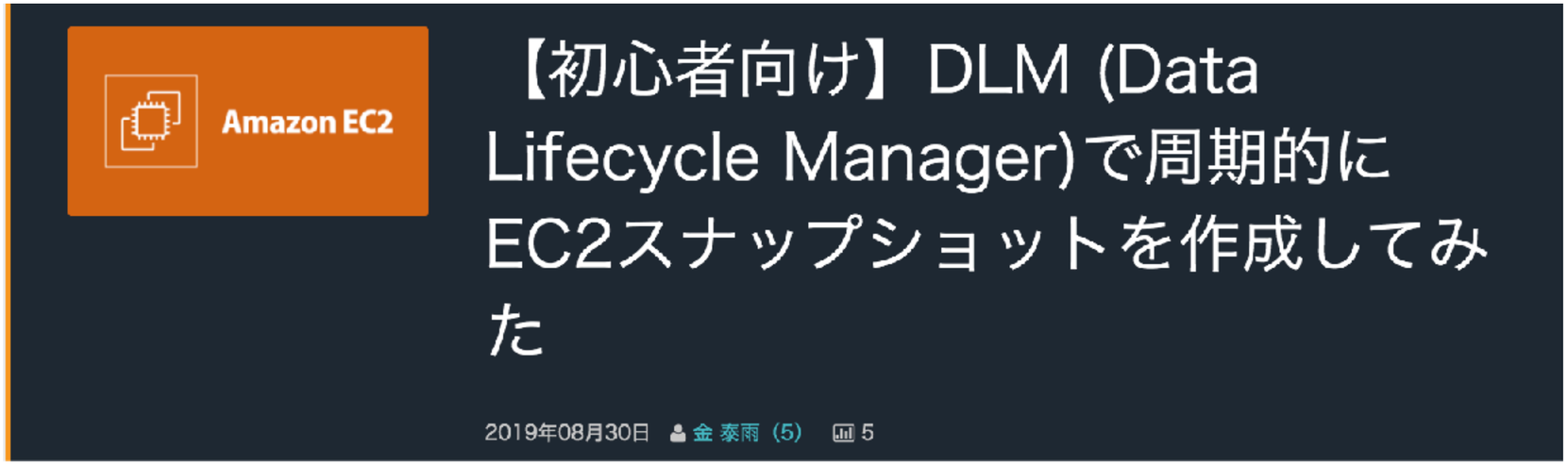
AWS運用かんたん自動化ツール「opswitch」（オプスウィッチ）を使ってみよう！



どんな感じかはこちら

https://dev.classmethod.jp/etc/get-started-with-opswitch/

あわせて読みたい



https://dev.classmethod.jp/cloud/ec2-snapshot-by-amazon-dlm/

https://dev.classmethod.jp/cloud/aws/aws-backup/

初級
OS/アプリレイヤー基礎

- セキュアな実装(コーディング/アーキテクチャ)
- 不正プログラム対策
- 脆弱性対策
- コンテナセキュリティ
- アイデンティティ

セキュアな実装(コーディング/アーキテクチャ)

ここでは詳細に触れませんがアプリの実装はもちろん
ユーザ責任の範囲です

オンプレミスでもクラウドでも変わらず対策しましょう

とりあえず徳丸本置いときますね

https://www.amazon.co.jp/dp/4797393165

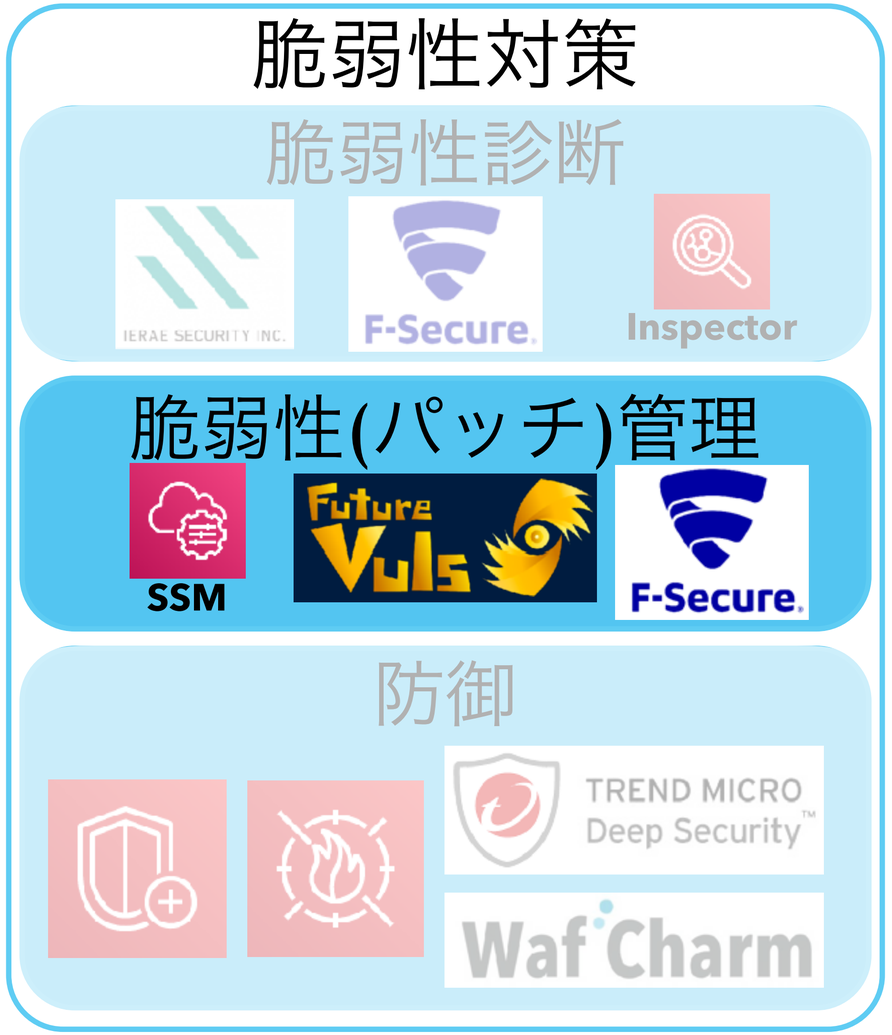
不正プログラム対策

- OSレイヤー以上はお客様責任範囲で、特にサーバ内部の対策はサードパーティ製品が必須
- Webサイトではインターネットに触れるサーバは攻撃を最初に受ける部分のため対策必須
- 弊社の提案としてはOS内で複数レイヤーの機能を持つDeepSecurityが推奨(AWSとの親和性が高い)

脆弱性対策

- 脆弱性診断はプラットフォーム診断とWebアプリケーション診断を両方実施する
  - Amazon Inspectorはプラットフォーム診断のみのため、定常的なチェックには向くがリリース前の全体チェックには力不足
  - 簡易的なWebサイト(個人情報を持たない・複雑なロジックがない)ならF-Secure RADAR(ツールでの診断)でいい
  - 上記に当てはまらない場合にはイエラエセキュリティのように手動で診断するサービス推奨

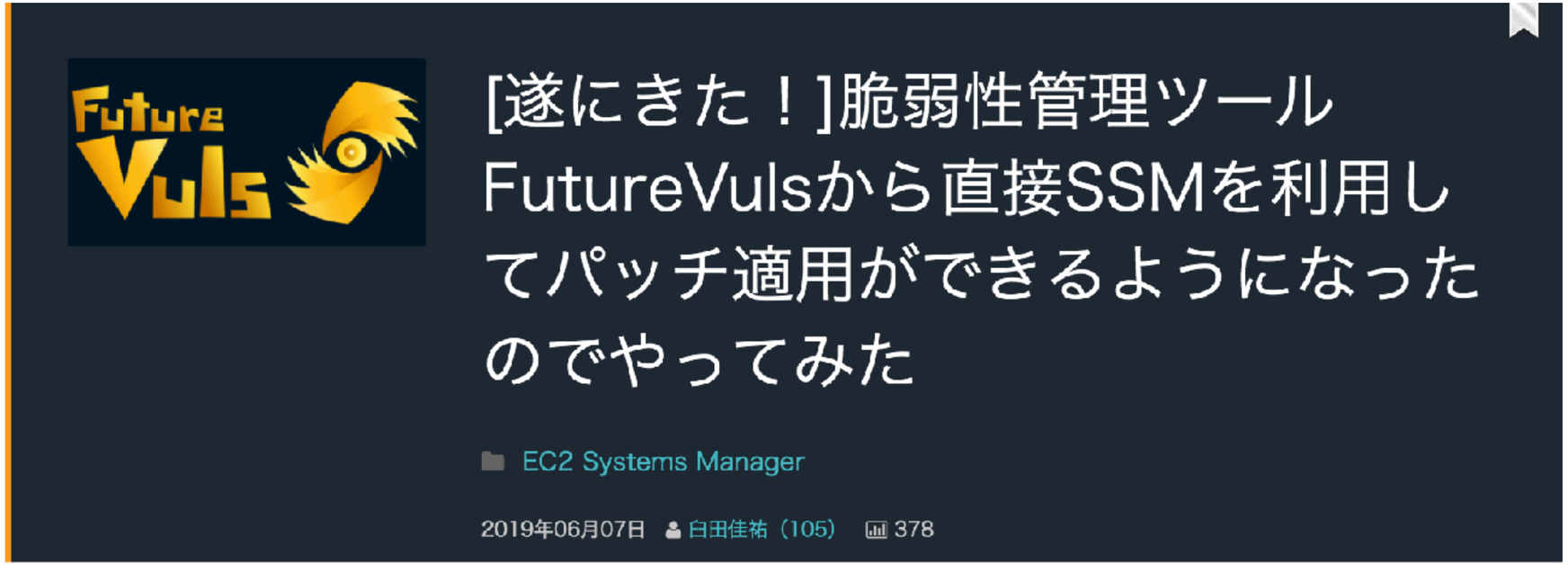
脆弱性対策



- 脆弱性管理はすべてのサーバで必須
  - SSM単体でもできなくないが、パッチ適用のみの利用が向いている
  - OSミドルウェアの脆弱性チェックはFutureVulsがいい、運用に寄り添える
  - Webスキャンと脆弱性管理、チケット機能がありAPI連携でCI/CDサイクルにF-Secure RADARを組み込めるのでそれも検討してみる

あわせて読みたい

[遂にきた！]脆弱性管理ツールFutureVulsから直接SSMを利用してパッチ適用ができるようになったのでやってみた



脆弱性管理から直接パッチの適用までできて運用が段違いで楽になるので活用すべし

https://dev.classmethod.jp/cloud/aws/ssm_integrate_and_patch_application_with_futurevuls/

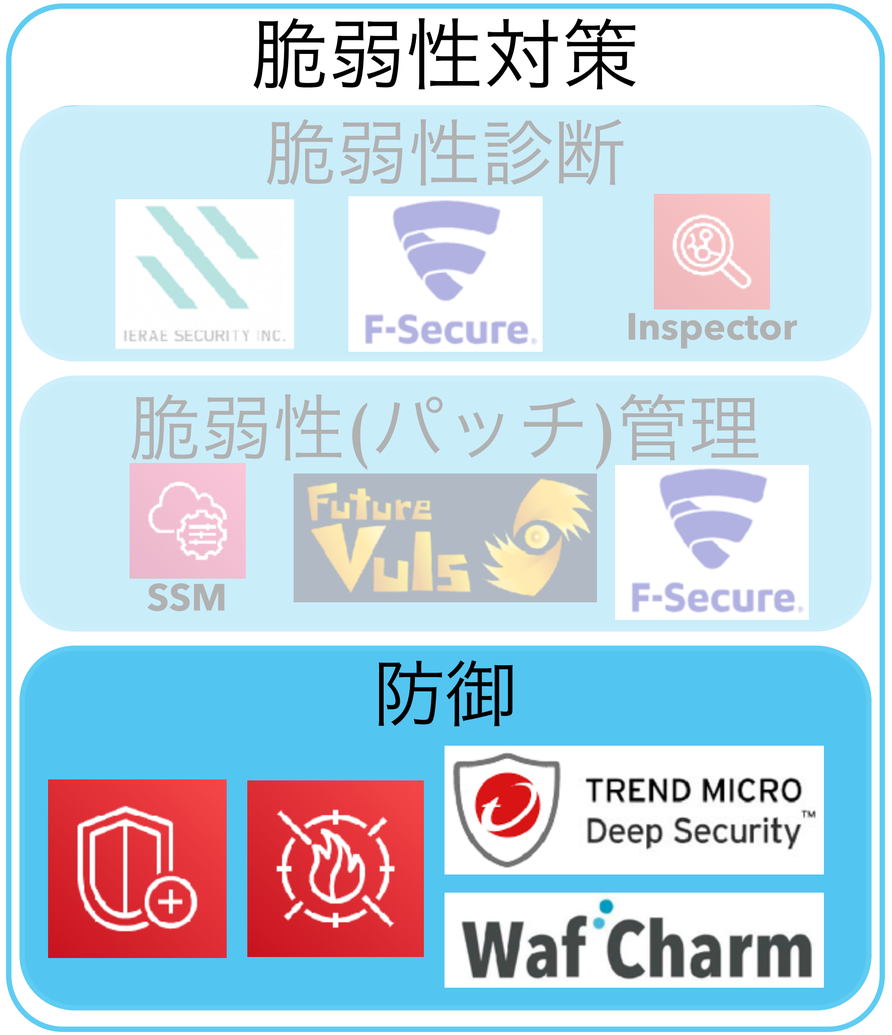
あわせて読みたい

【F-Secure Radar API + CodePipeline】Webスキャンの実行と結果判定を自動化してみた

CI/CDにWebアプリケーションの脆弱性診断を組み込む手法を紹介しています

https://dev.classmethod.jp/cloud/aws/auto-webscan-with-pipeline/

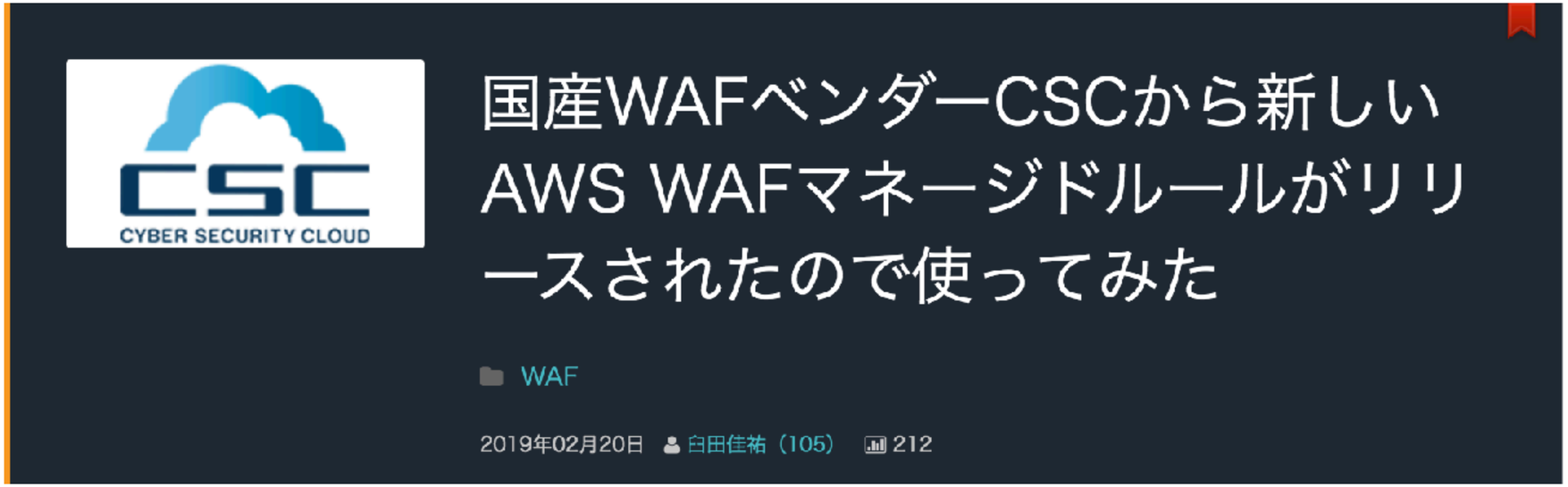
脆弱性対策



- 防御は暫定対処のため診断や管理をした上で活用
  - DeepSecurityでOS上での対策も可能だが、可能なら前段でWAF等で止めたい
  - AWS WAFはアプライアンスWAFよりも機能は限定的だがスケール等相性はいいのでエントリーレベルから検討する
  - AWS WAF運用のナレッジがない場合にはWafCharmによる自動運用も検討

あわせて読みたい

国産WAFベンダーCSCから新しいAWS WAFマネージドルールがリリースされたので使ってみた



マネージドルールどれ使ったら良いかわからなかったらとりあえずこれ

https://dev.classmethod.jp/cloud/aws/csc-waf-managed-rule-release/

コンテナの不正プログラムや脆弱性対策

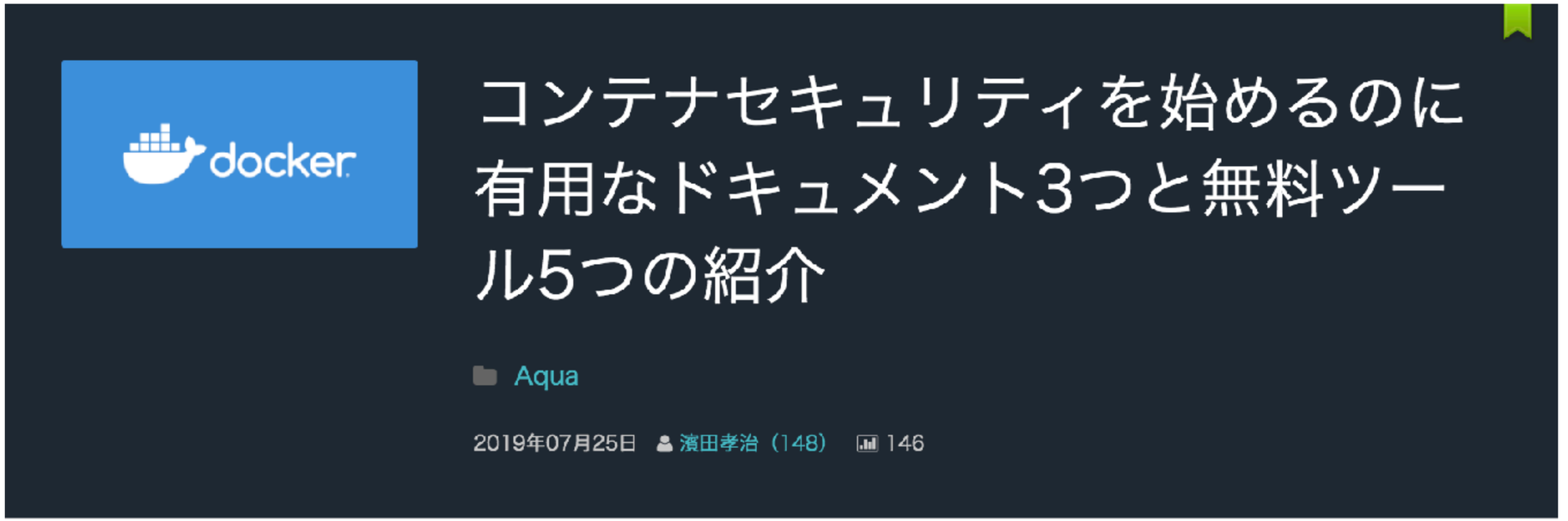
- Fargateなどホストを操作できないコンテナ系では特にセキュリティ製品はまだ市場に出揃っていない

コンテナ 

- Aquaならリリース前のコンテナの静的スキャンや稼働中の動的なスキャンにも対応している

あわせて読みたい

コンテナセキュリティを始めるのに有用なドキュメント3つと無料ツール5つの紹介

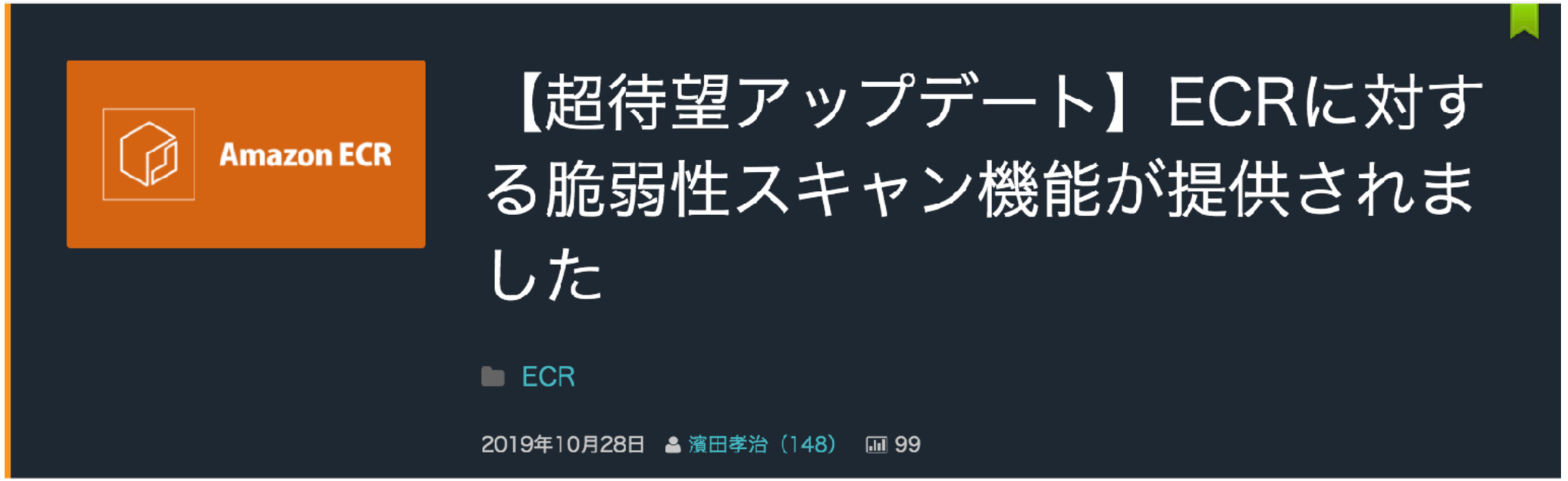


コンテナセキュリティはNISTのドキュメントを読むといい

https://dev.classmethod.jp/tool/docker/container-security-tools-and-docs/

あわせて読みたい

【超待望アップデート】ECRに対する脆弱性スキャン機能が提供されました



コンテナの自動スキャンができるようになっています

https://dev.classmethod.jp/cloud/aws/ecr-repository-scan/

アイデンティティ

- アプリケーションでのアイデンティティ管理の実装は大変
- Cognitoで認証機能を利用するのも一つの手
- CognitoならAdvanced Securityでリスクベースの認証強化なども可能
- より手軽に強力な機能を利用するならAuth0

初級
運用

運用設計した？(セキュリティ以外も含めて)

お願いだから運用設計してください(切実)

運用設計

幅が広すぎるのでかいつまんでトピックあげます

- CloudWatchをベースにEC2の死活監視やRoute53の外形監視を使ってアラート設定をして異常を検知する
- その他使っているサービスのメトリクスで何を見るか、アラートトリガーにするかを決めておく(何を障害とするか定義する)
- アプリ側のメトリクス取得やログの外部出力(S3やCloudWatch Logs)を行う
- 検知後の1次切り分けや復旧フローを作っておく
- 何回もフローを実際に流しておく(クラウドだからもう一個環境作って簡単にできます)
- できるだけ自動化する(例: AutoScaling使うだけでEC2の復旧を自動化できる)

バックアップ復旧の試験はやりましょう

- 簡単にバックアップイメージから復旧できますが、わかってないと簡単ではないです
  - EC2の場合、AMIからの復旧かEBSスナップショットからの復旧か？
  - IPアドレス変わることもあるから大丈夫か
  - RDSなら無くなるデータのことも考えて

復旧計画もしっかり作る

- AZ障害が起きたらどうするか
- リージョンレベルも考えるのか
- 環境を作り直しやすいようにCloudFormationなどでテンプレート化するという手もある
- いわゆるInfrastructure as Code(IaC)
- 何でも組み込みすぎないように目的に合わせた作り方をするように(変更の反映が手間になりすぎたら本末転倒)

監視・ロギング・分析

監視 mackerel CloudWatch

ロギング・分析


CloudWatch


Config


CloudTrail

sumo logic®

- 各種リソースのメトリクスとログの取得は必須
- ログ分析は単純なWebサイトなら優先度は低め。パフォーマンスやセキュリティ要件が高い場合には検討する
- ログについて保存は基本S3保存、分析を検討するならCloudWatch Logs/ElasticSearch/Sumo Logicが選択肢になる
- 分析を初めて行う場合や分析に集中したい場合、スケールが読めない場合にはSumo Logicが初期ダッシュボードとインテグレーションに優れているのでいい

あわせて読みたい

CloudWatchダッシュボードを利用してオートスケール環境の稼働状態を可視化してみた

ネ申のダッシュボードを活用しましょう

https://dev.classmethod.jp/cloud/aws/autoscalling-with-cloudwatch-dashboard/

その他監視要素

- **Auroraのパフォーマンスインサイトは可視性がちょー上がる**

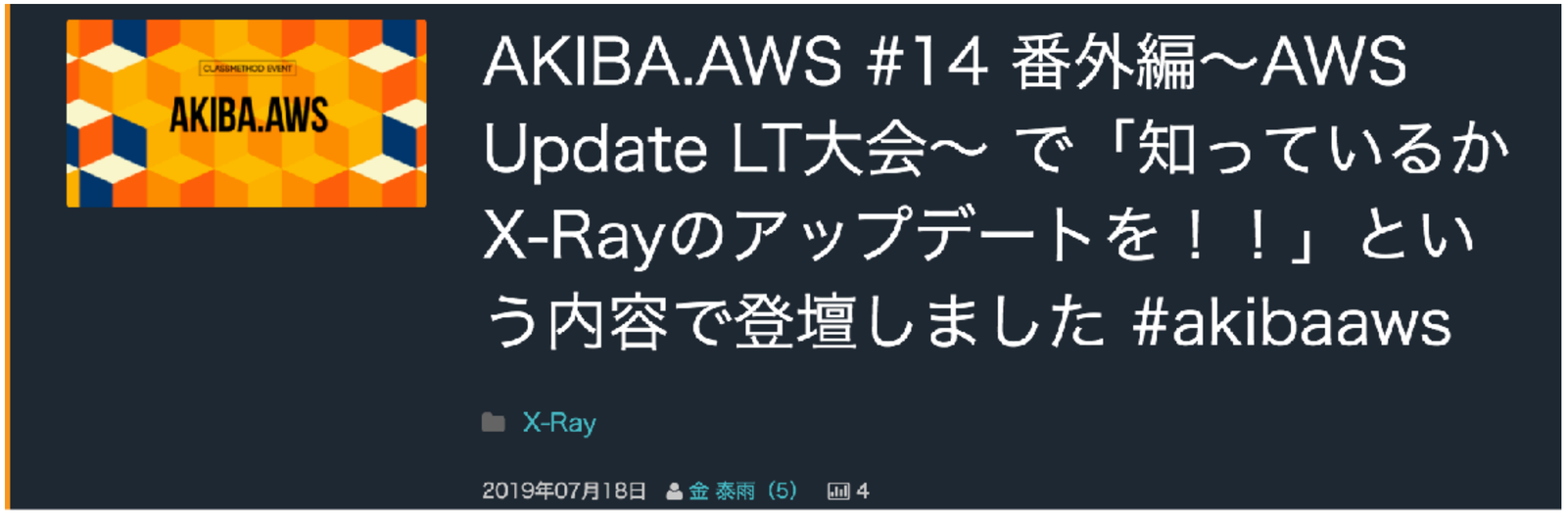
- https://dev.classmethod.jp/cloud/aws/wordpressdb-performance-insight/

- **ECSやEKSのコンテナインサイトもいっぱい見える**

- https://dev.classmethod.jp/cloud/aws/container-insights-ga/

その他監視要素

- AKIBA.AWS #14 番外編〜AWS Update LT大会〜 で「知っているかX-Rayのアップデートを！！」という内容で登壇しました #akibaaws



- アプリ側はX-Rayを使うとよく見える
- https://dev.classmethod.jp/cloud/aws/akiba-aws-14-xray/

セキュリティ系サービスの運用

- アラートが出たときの対応はぼちぼち大変だけど確認するポイントだけでもまとめておく
- GuardDuty / AWS WAF / DeepSecurityは参考情報載せておきます

Amazon GuardDutyによる疑わしいネットワーク通信の検知と初動対応の振り返り

運用時の対応の一例

https://dev.classmethod.jp/cloud/aws/guardduty-firstaction/

AWS WAFマネージドルール導入と運用の勘所

AWS WAF運用の参考例

https://dev.classmethod.jp/cloud/aws/howto-waf-managedrule/

はじめてのDeep Security運用に必要な情報まとめ

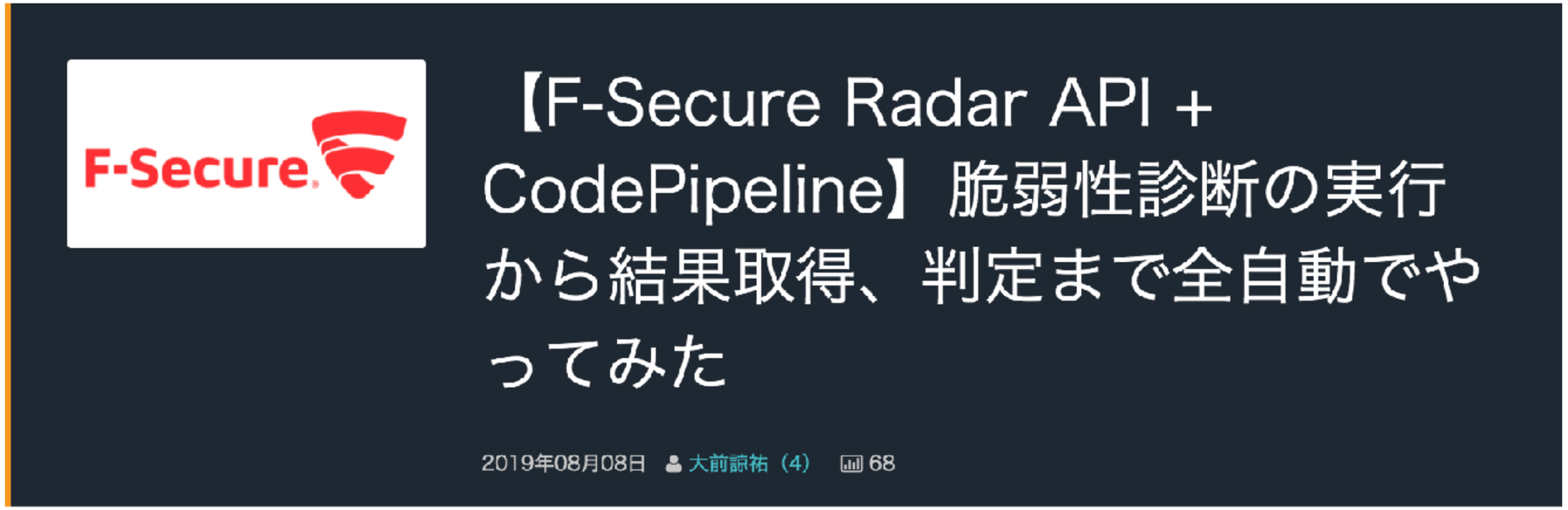
https://dev.classmethod.jp/cloud/aws/deepsecurity-first-step-with-classmethod/

CI/CDついでにセキュリティ対策

- CI/CDのフローがあるならやりましょう
- なくてもやりたいなら文化づくりなので要相談
- 「自分たちでやっていくぞ！」という意思が必要

あわせて読みたい

【F-Secure Radar API + CodePipeline】脆弱性診断の実行から結果取得、判定まで全自動でやってみた



EC2のOS/ミドルウェアの自動診断に有効

https://dev.classmethod.jp/cloud/aws/all-auto-vulscan-with-pipeline/

初級まとめ

- AWSレイヤーのセキュリティはオンプレミスに無いものなのできちんと覚える
- OS/アプリレイヤーは従来どおりだけど、これまで出来てなかったなら気にする(オンプレミスよりはやりやすい！)
- 運用を最初からちゃんと考える

全部をやる必要はないかもしれないけど検討はしましょう

中級

中級の内容

- 発見的統制 / インシデントレスポンス
- GRC(ガバナンス / リスク / コンプライアンス)

視野を広く持って欲しいって話です
一人じゃなく組織で対応していく話

中級

発見的統制 / インシデントレスポンス

発見的統制とは

- Detective Control
- 予防的コントロールと対比して使われる事が多い
- リスクマネジメントの用語
- WAFやパッチ適用でインシデントを予防するだけでなく、インシデントが起きた時に気付けるようにしておく
- 日本だと発見的統制の体制が弱いと言われますね

インシデントレスポンス(IR)とは

- インシデント(障害とかセキュリティ事故とか顕在化した脅威など)の対応(レスポンス)をすること
- AWSにおけるインシデントレスポンスは各種アラート等から検知して、ログ等を見て事象を確認、設定変更や環境隔離、IAMの削除等を行っていく

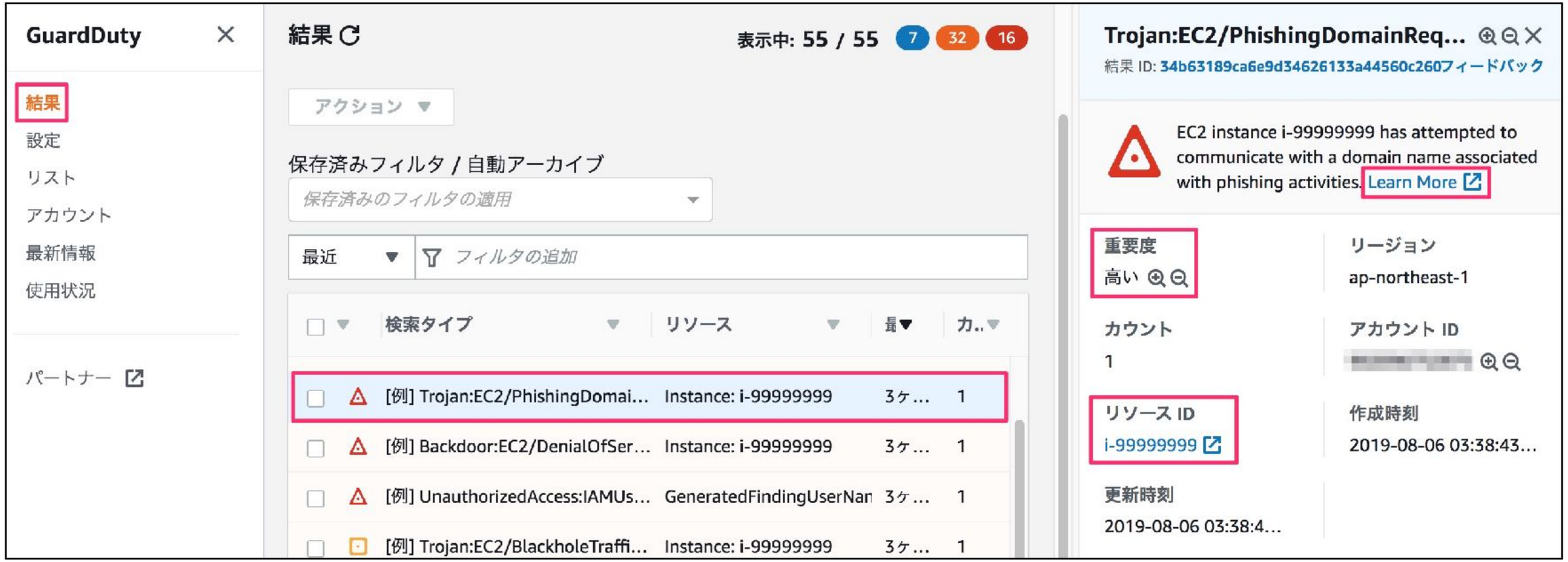
発見的統制の第一歩: GuardDuty

- GuardDutyはAWS上で起こるあらゆる脅威を検知してくれる
- まずはこのアラートから対応する体制を作る
- どんなアラートが上がるかの例
  - IAMのアクセスキーが漏洩して不正利用されている
  - EC2がマルウェアに感染している

GuardDutyのアラートを受け取る

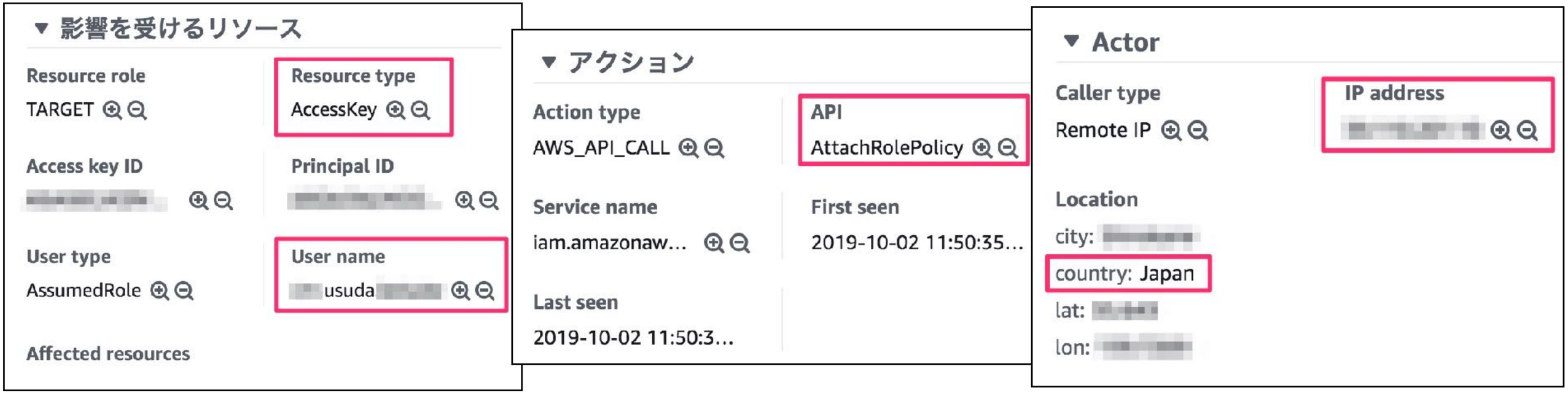
- GuardDutyは有効化するだけだとアラートが飛ばない
- CloudWatch Eventsを経由してSNSでメールやSlackへのチャットやBacklogへのチケットを飛ばす
- メールを受け取ったらAWSマネジメントコンソールにログインして見る(jsonより見やすい)

GuardDutyの見るところ



- GuardDuty画面の結果から選んで詳細画面
- 重要度と対象のリソースをチェック
- 脅威の内容はLearn Moreから詳細ドキュメントへ飛べる

GuardDutyの見るところ



- リソース / API / Actor(操作元)などの情報も
- ユーザ名や操作しているIP、国はよく使う判断材料

GuardDutyの対応方法

- 検知した内容によるがクリティカルなものは2種類
- IAM不正利用はIAMを無効化して影響調査
  - CloudTrailで該当クレデンシャルでの操作をすべて確認
  - Admin権限が乗っ取られていたら最悪アカウント破棄
- EC2が乗っ取られていたら隔離・フォレンジック
  - in/outすべて制限したSecurity Groupで隔離
  - ssm-aquire等のツールを使うと楽
  - 重要なデータにアクセスしていないかとか確認

あわせて読みたい

- フォレンジックツールのSSM Acquireを使ってみた #reinvent

- 事前に準備しておけば安全に、専門家じゃなくても必要なデータを取得できる
- https://dev.classmethod.jp/cloud/aws/reinvent2018-ssmacquire/

実際のGuardDutyの対応

- ちゃんとやっていれば重要度: 高のアラートはない
- 重要度: 中以下は過検知の場合もある
- でも内容は確認してね
- よくある内容
  - UnusualAccess: 普段とは違う場所からのアクセス
  - UserPermissions: IAMの権限変更
- これらは環境によってはよく上がるが、ただの過検知なのかはしっかり確認する
- 検知結果とその対応法
  - https://docs.aws.amazon.com/ja_jp/guardduty/latest/ug/guardduty_finding-types-active.html

過検知が多い場合の対応方法

- まず1ヶ月程度待つ
  - パターンを学習するためよく行われる操作なら検知されなくなる
- IPアドレスのホワイトリスト登録
  - オススメしない
  - 全く検知結果が残らないため
- 自動アーカイブ
  - GuardDutyの仕組みを使う
  - もしくはBacklog等チケットシステム側と連携してアーカイブする仕組みを作る

次の発見的統制: Config Rules

- 変更管理を行うAWS Configの機能の一つ
- 変更された内容が定めたルールから逸脱したらアラートを出す
- 例: Security GroupをSSHフルオープン(0.0.0.0/0)を設定する
- アラートはSNS

Config Rulesのルール設定

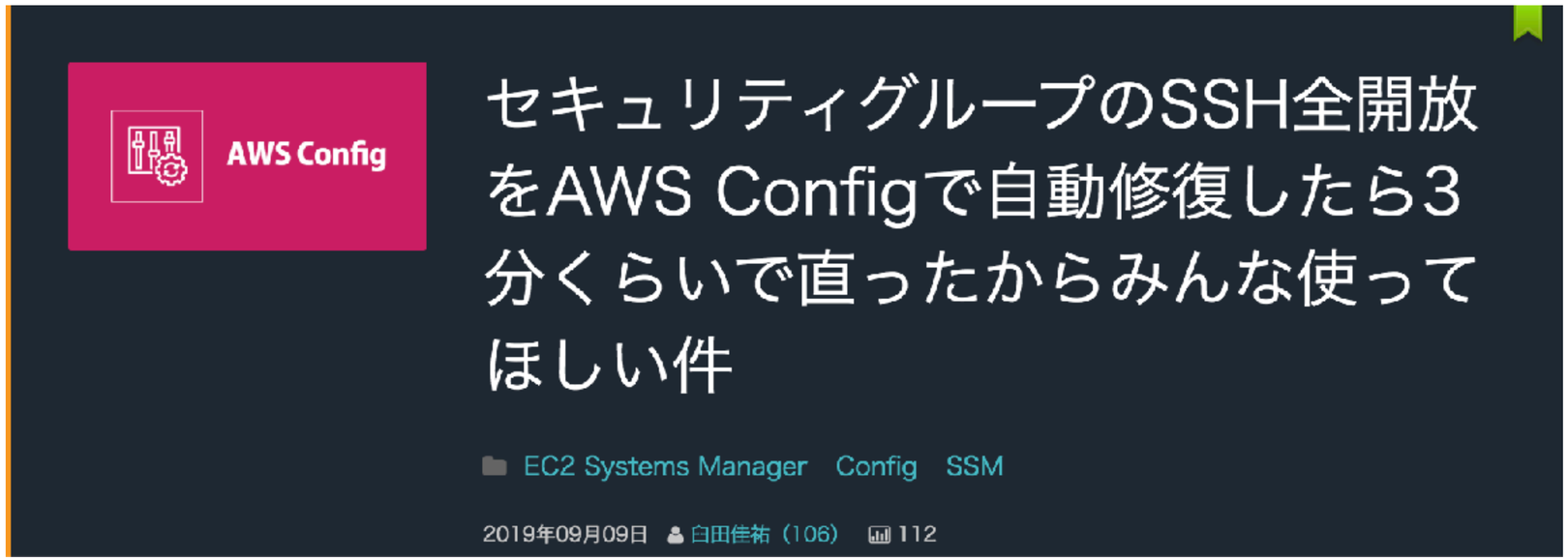
- ルールは2種類
  - AWSが用意しているマネージドルール
  - ユーザでLambdaを作成して作るカスタムルール
- マネージドルールを眺めて必要なものを導入するところから始めるのがオススメ

あわせて読みたい

推奨ルール

https://dev.classmethod.jp/cloud/aws/recommend-config-rules-for-all-user/

自動修復



https://dev.classmethod.jp/cloud/aws/auto-recovery-restricted-ssh-without-lambda/

Security Hub

- GuardDuty / Inspector / Macieやサードパーティツールのアラートをまとめて管理する仕組み
- コンプライアンスチェック機能もある

- https://dev.classmethod.jp/cloud/aws/reinvent2018-security-hub/

Security Hubの使い方

- インサイトとしていろんな観点でのグラフが見れる
- アラートが多いAWSアカウントやリソース等

- 見つかっている内容に対してカスタムアクションを実行可能
  - チケット起票したりEC2を隔離したり(Lambda経由)

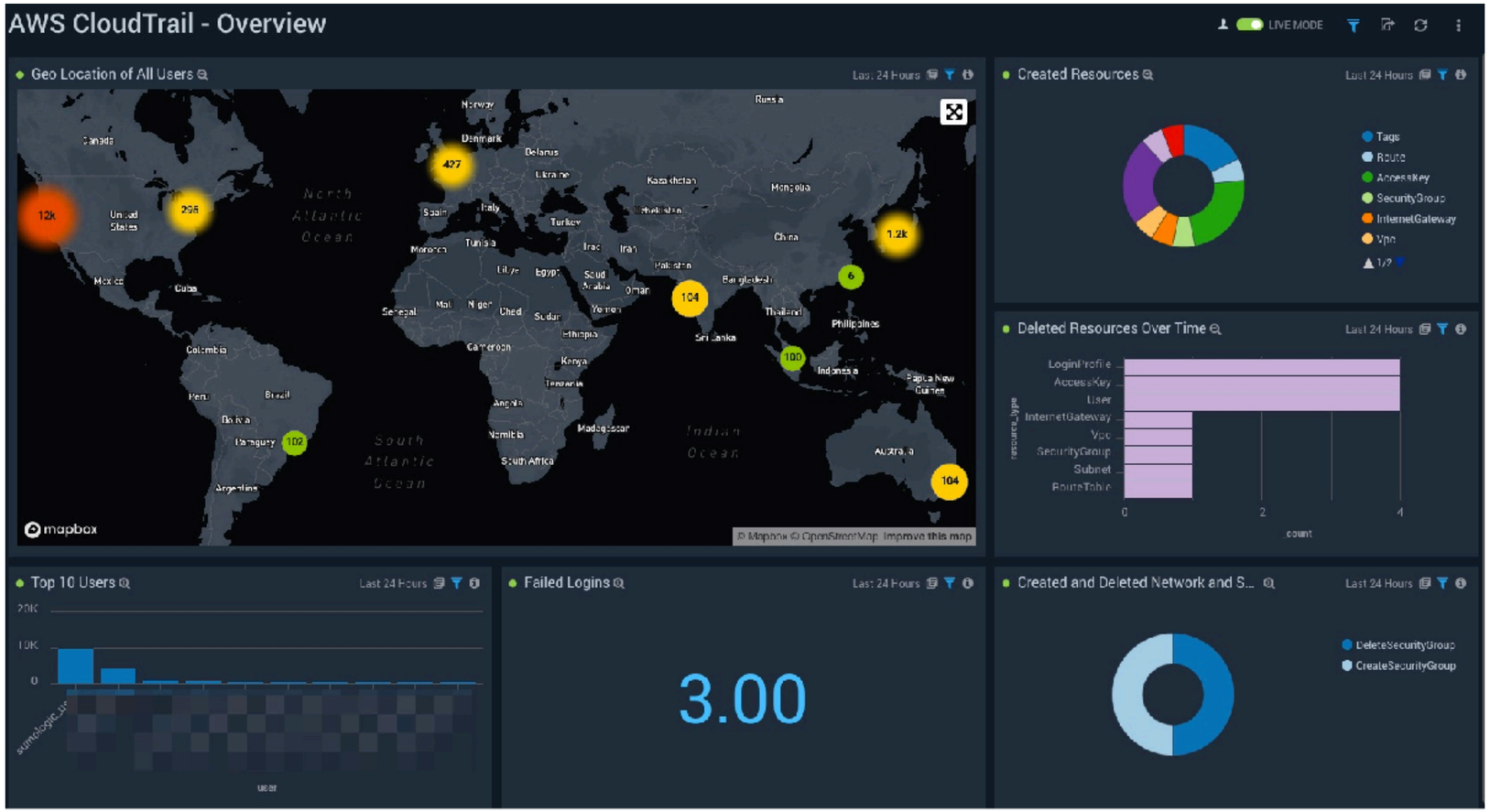
Security Hubの用途

- アカウントをまたがって情報を集められるところはいい
- コンプライアンスチェックとして使ってもいい
- ただ現状の機能では見通しがそんなに良くならない

- https://dev.classmethod.jp/cloud/aws/security-hub-usecase/

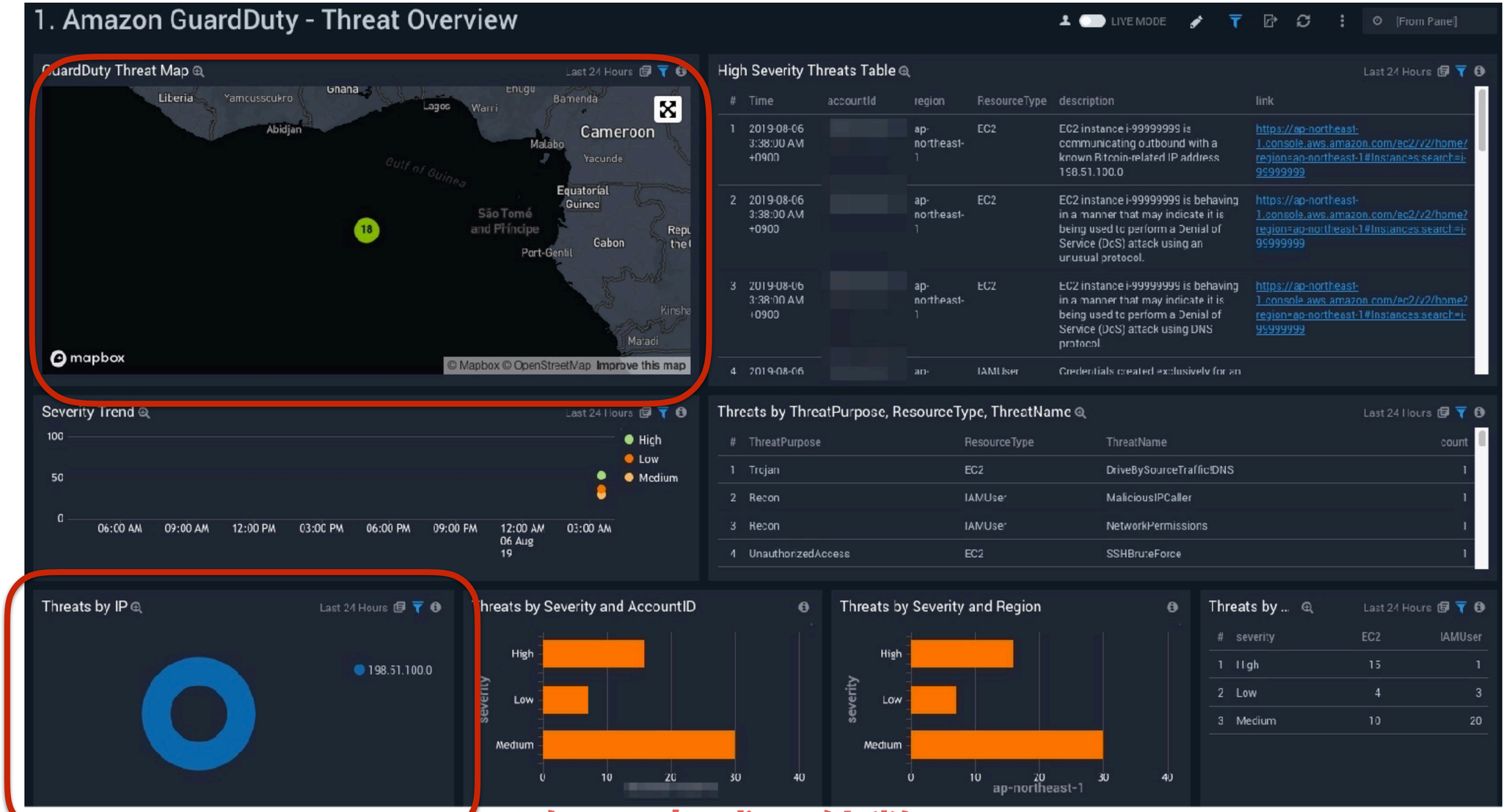
Sumo Logicによるログ分析

- 少しアプローチが変わるが何が起きているかを見る場合にはログ分析SaaSのSumo Logicの活用がオススメ
- CloudTrailログなどから不審な動きをわかりやすく可視化



GuardDutyも見れる

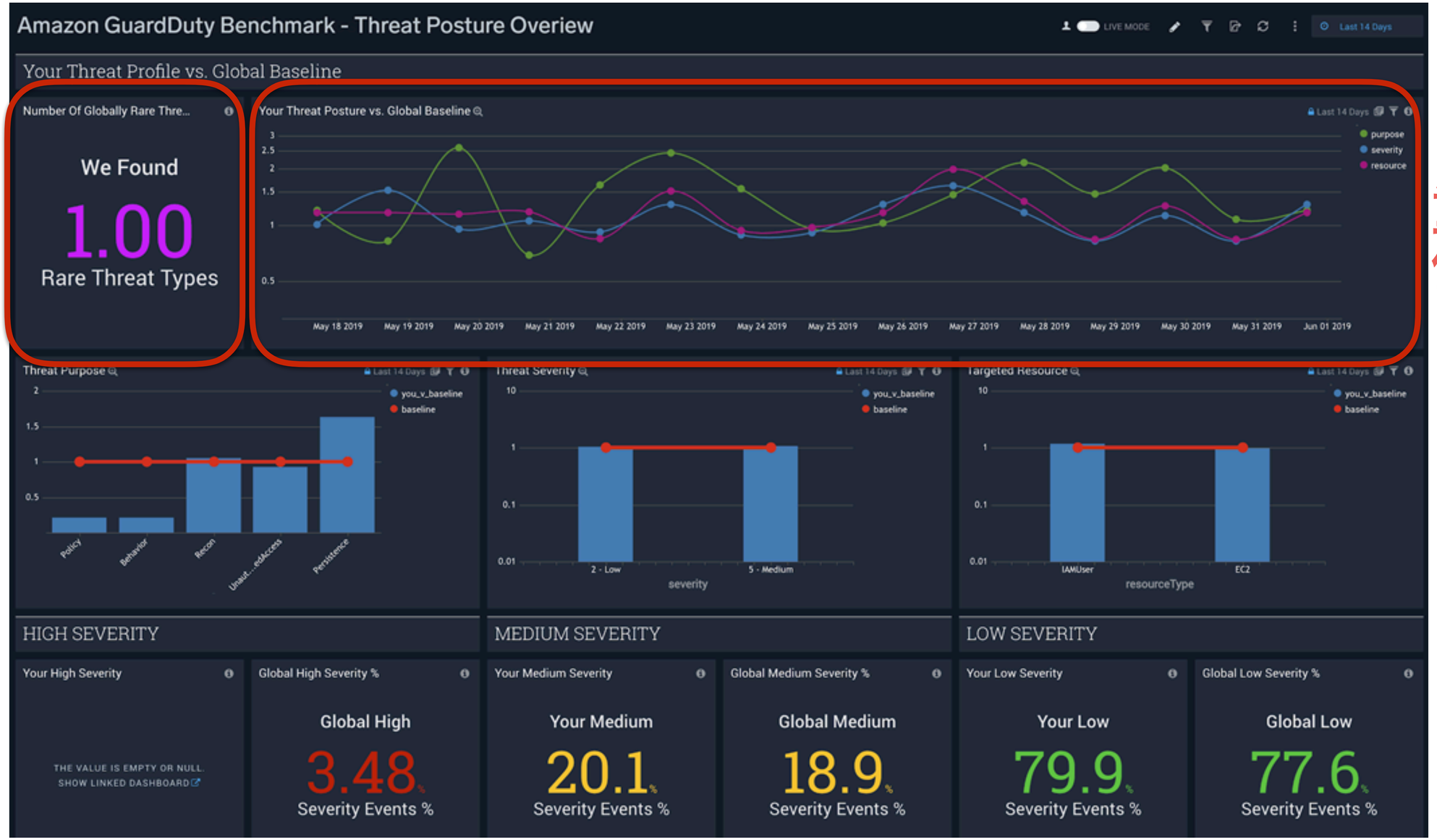
どこで起きているかのグローバルマップ



既知の脅威IP件数

GuardDutyも見れる

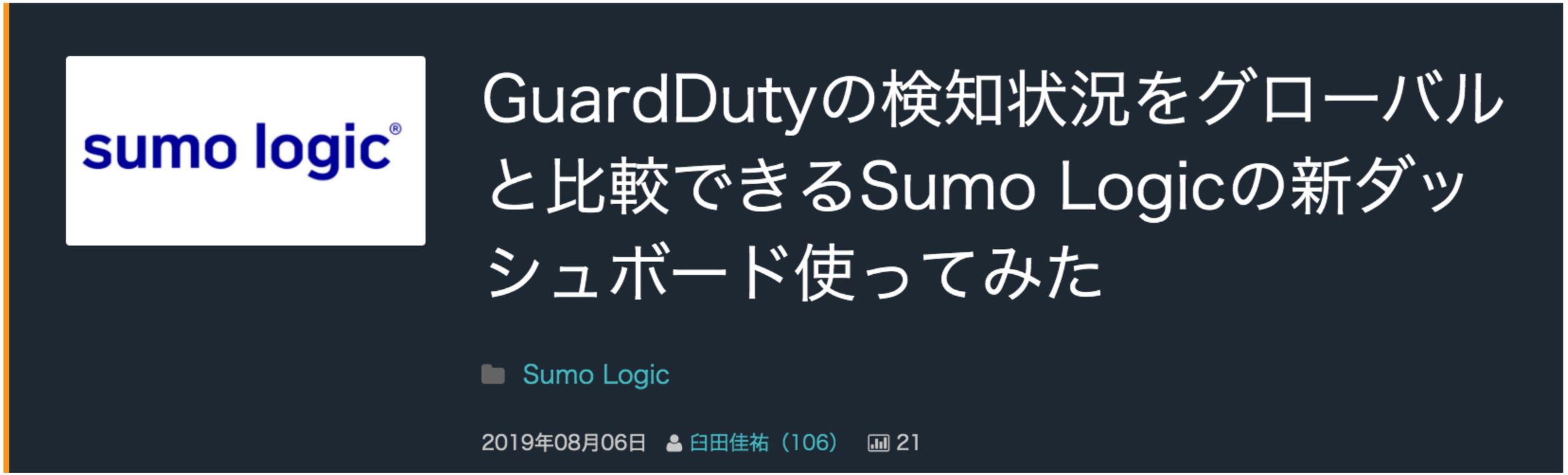
特に危ない脅威の件数



重要度毎の
件数の推移

あわせて読みたい

GuardDutyの検知状況をグローバルと比較できるSumo Logicの新ダッシュボード使ってみた

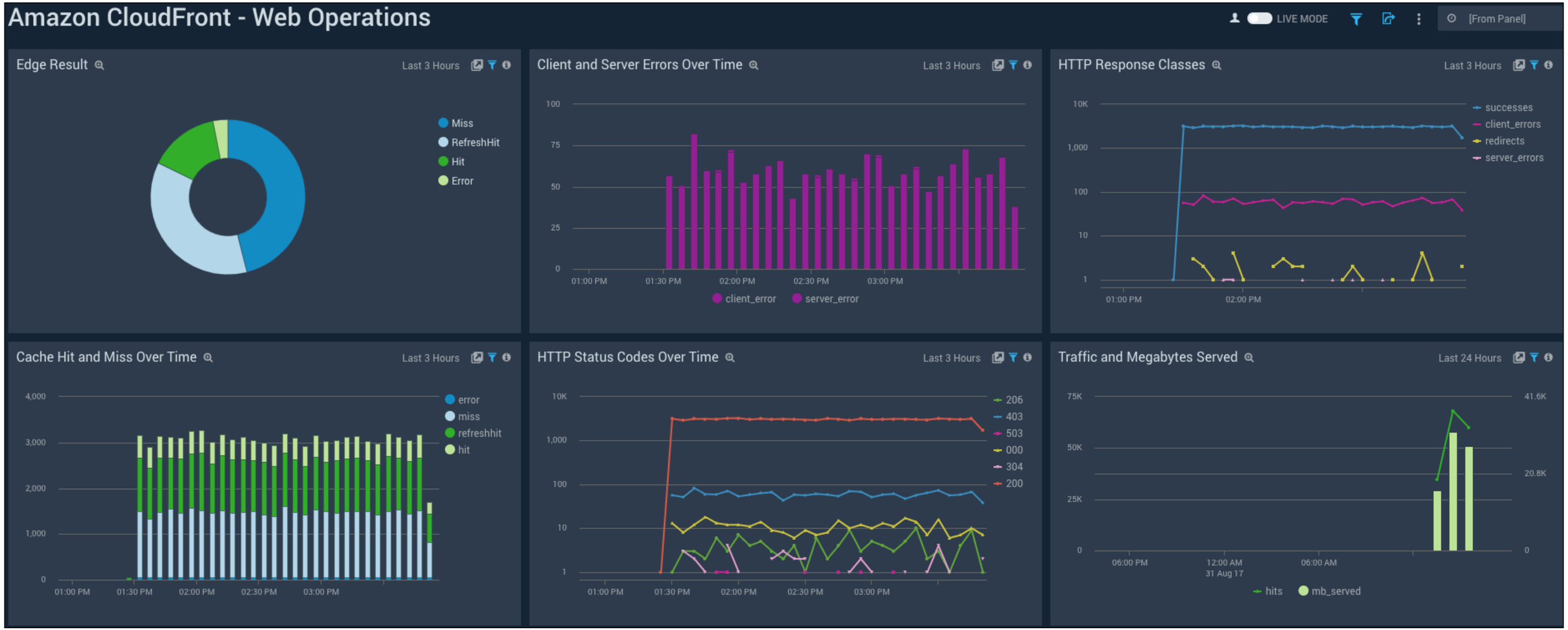


https://dev.classmethod.jp/cloud/aws/sumologic-guardduty-gis/

Sumo Logicのススメ

- AWSのログを簡単に強力に可視化(デフォルトのダッシュボード)
- セキュリティ以外にも運用で活用できる
  - CloudFront
  - ALB
  - Lambda
  - RDS等々
- ミドルウェアのログなどもダッシュボードあり

- Webオペレーション


Amazon CloudFront - Web Operations
LIVE MODE
[From Panel]
Edge Result
Last 3 Hours
Miss
RefreshHit
Hit
Error
Client and Server Errors Over Time
Last 3 Hours
100
75
50
25
0
01:00 PM
01:30 PM
02:00 PM
02:30 PM
03:00 PM
client_error
server_error
HTTP Response Classes
Last 3 Hours
10K
1,000
100
10
1
01:00 PM
02:00 PM
successes
client_errors
redirects
server_errors
Cache Hit and Miss Over Time
Last 3 Hours
4,000
3,000
2,000
1,000
0
01:00 PM
01:30 PM
02:00 PM
02:30 PM
03:00 PM
error
miss
refreshhit
hit
HTTP Status Codes Over Time
Last 3 Hours
10K
1,000
100
10
1
01:00 PM
01:30 PM
02:00 PM
02:30 PM
03:00 PM
206
403
503
000
304
200
Traffic and Megabytes Served
Last 24 Hours
75K
50K
25K
0
41.6K
20.8K
0
06:00 PM
12:00 AM
31 Aug 17
06:00 AM
hits
mb_served

- 訪問者統計

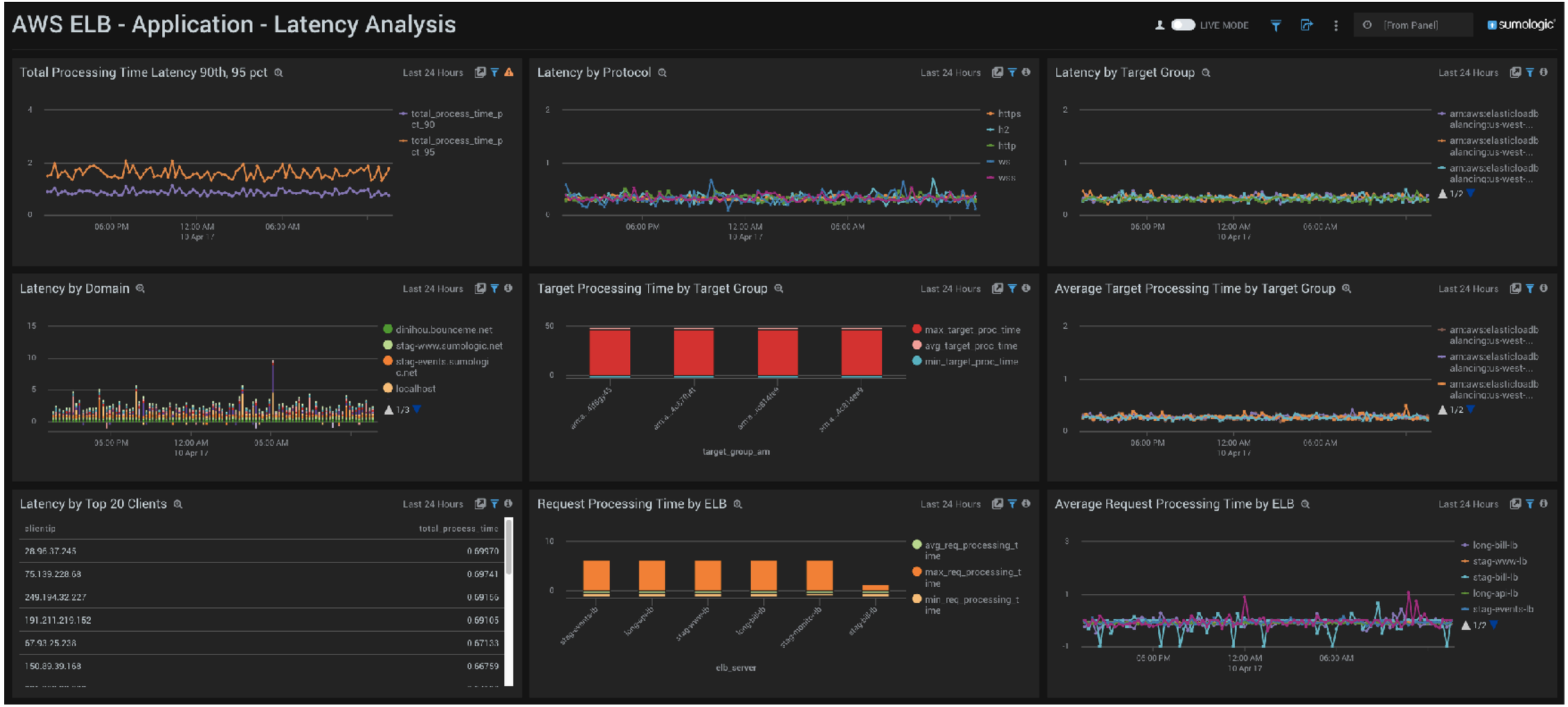

Amazon CloudFront - Visitor Statistics
LIVE MODE
[From Panel]
Client Geo Distribution
Last 24 Hours
7.3k
44k
7.5k
815
mapbox
Requests Served by Edge Location
DFW53-M2
ATL53-M3
IAD51-M3
ATL52-M2
SEA19-M2
ORD51-M1
1/6
Visitor Access Platforms
Last 3 Hours
unknown
mobile
PC
Mac
Visitor Session Duration Distribution Histogram
150
100
50
0
_count
175
180
bucketsize
Unique Visitors Over Time
Last 3 Hours
150
100
50
0
labs/aws/cloudfront
09:00 AM
09:30 AM
10:00 AM
10:30 AM
11:00 AM
Visitor Browsers and Devices
150K
100K
50K
0
Series 1
other
Other
os

Application Load Balancer(ALB)

- ダッシュボード

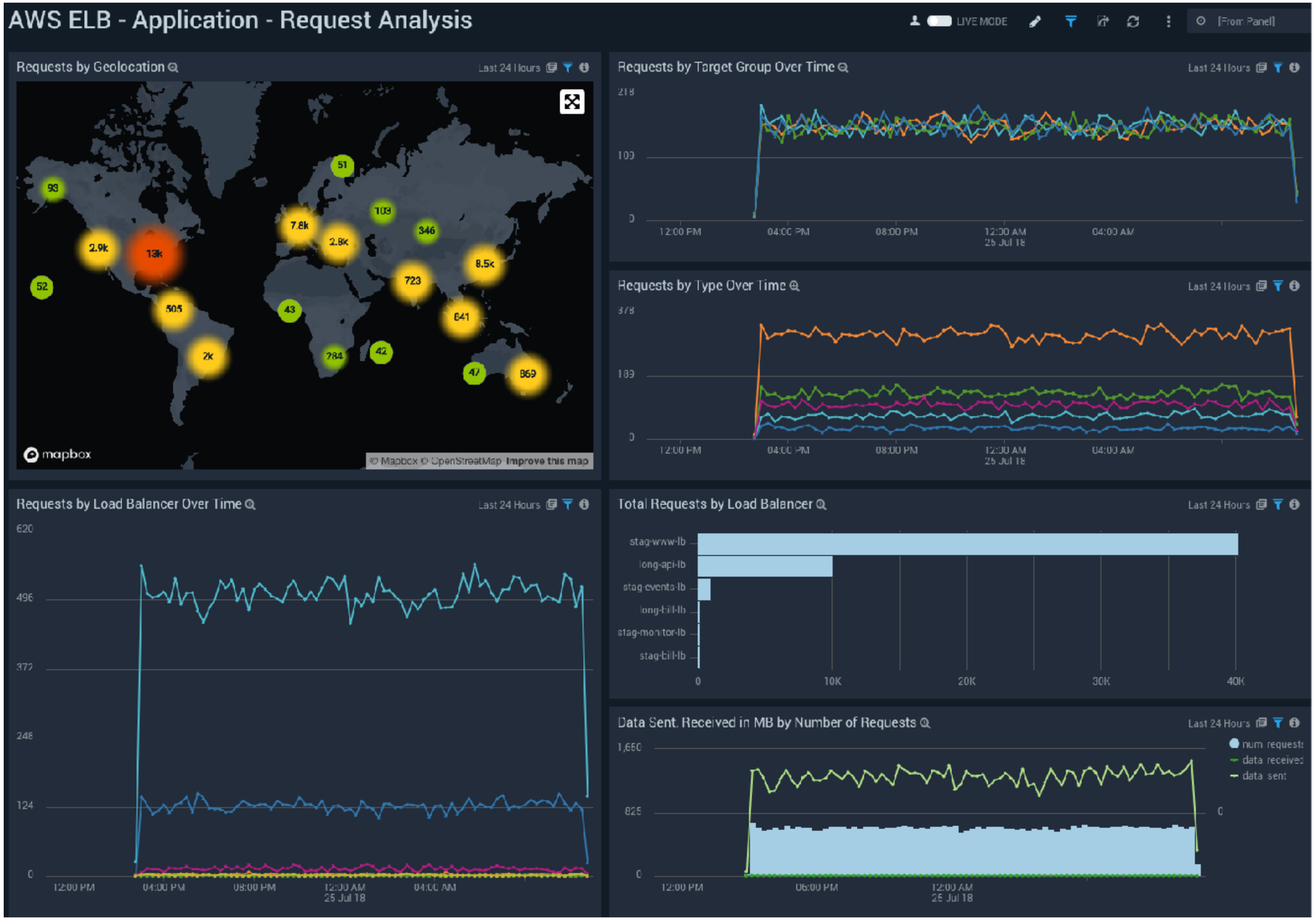
Application Load Balancer(ALB)

- レイテンシ分析



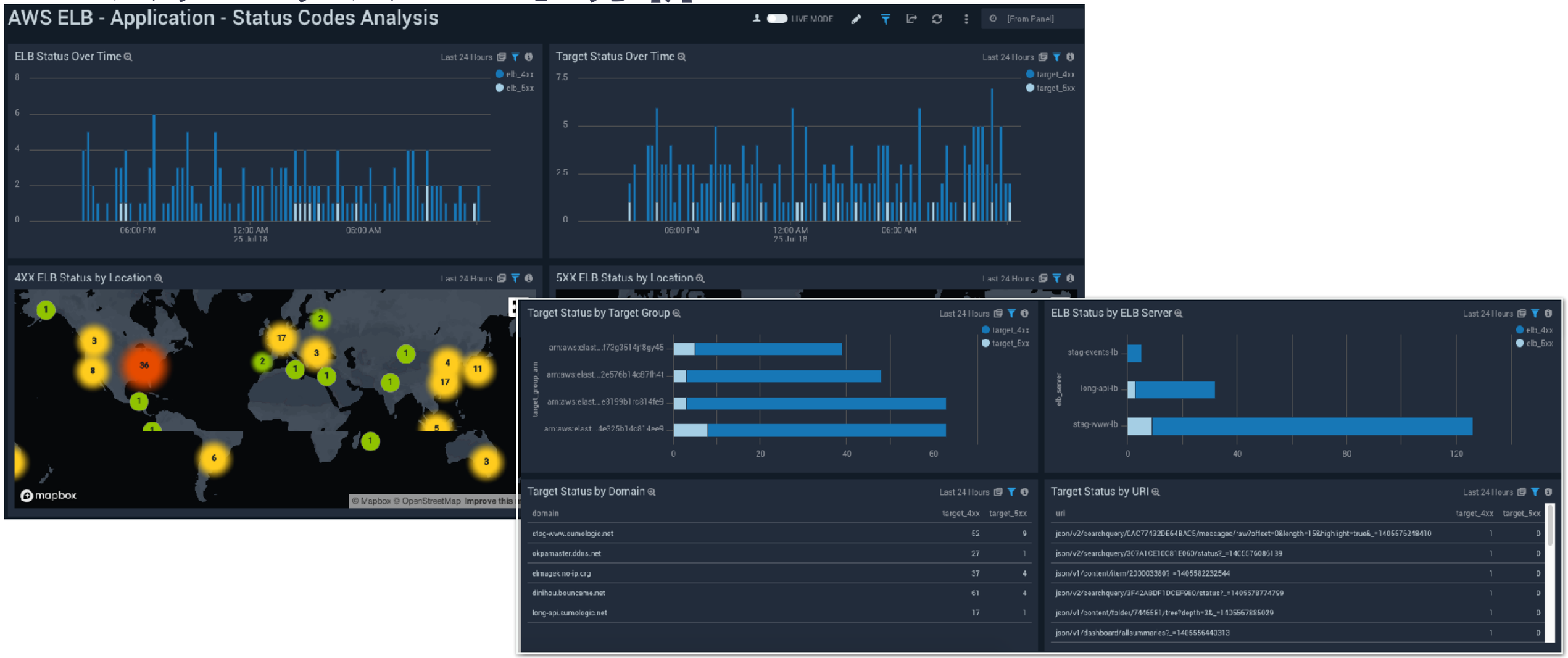
Application Load Balancer(ALB)

リクエスト分析



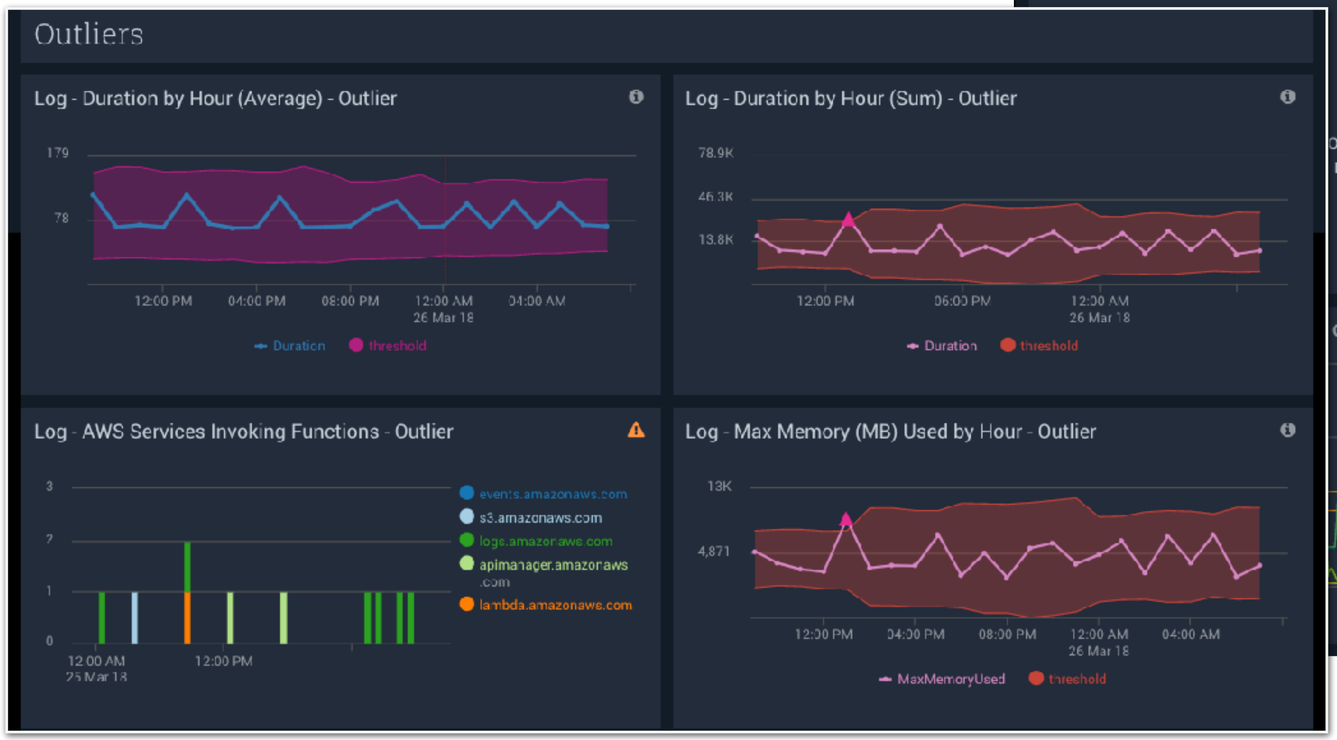
Application Load Balancer(ALB)

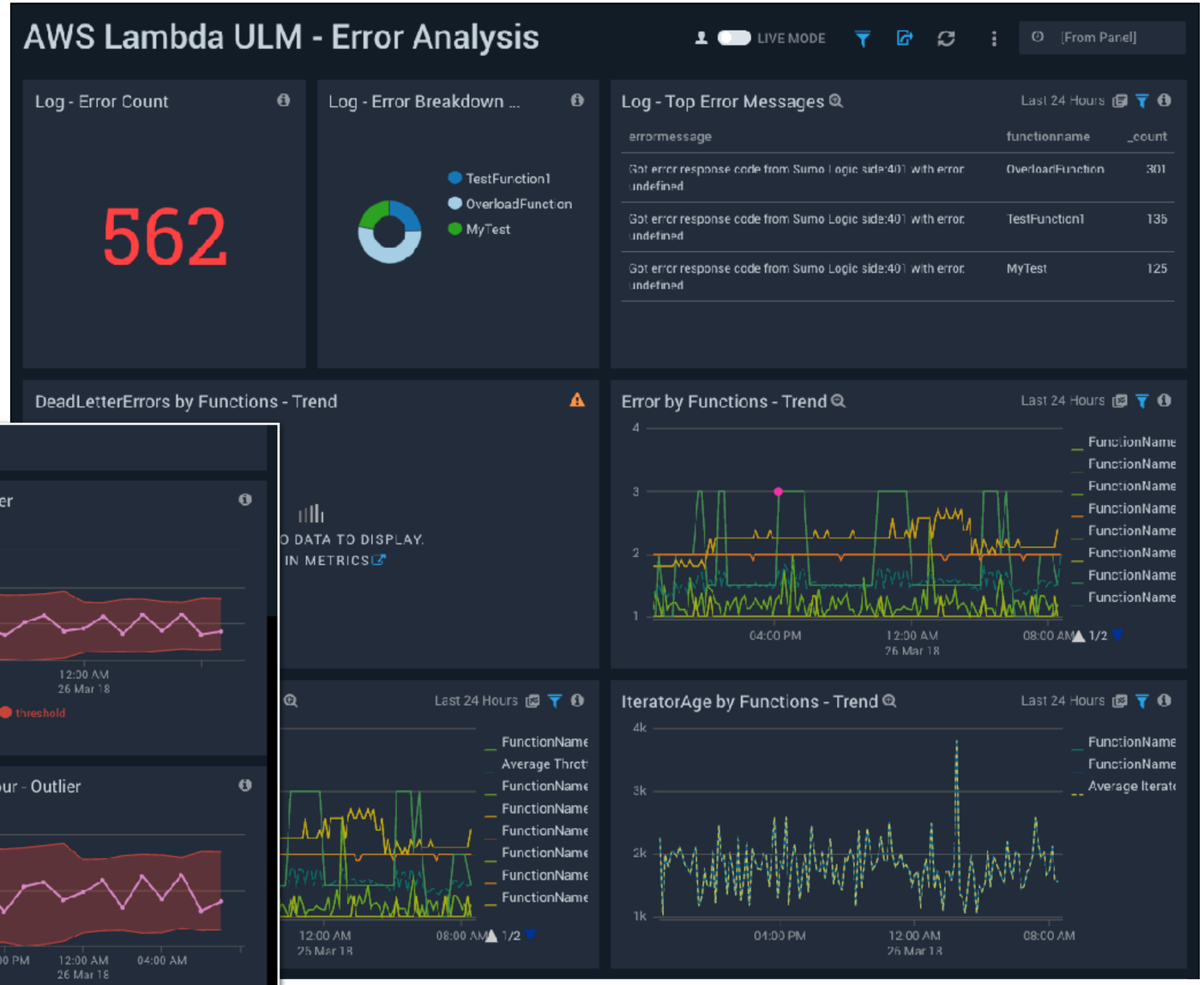
- ステータスコード分析



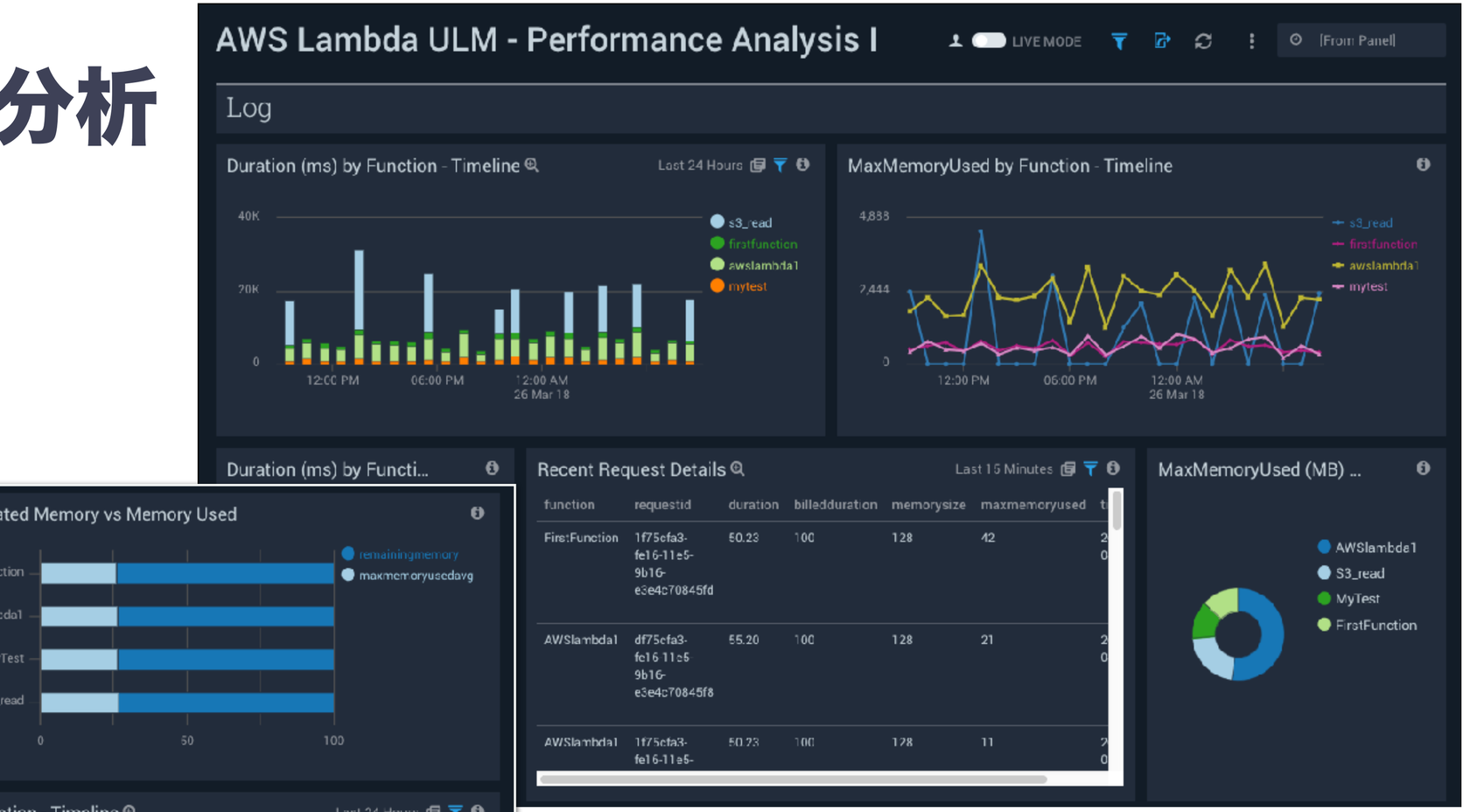
AWS Lambda

- エラー分析





- パフォーマンス分析


AWS Lambda ULM - Performance Analysis I
LIVE MODE
[From Panel]
Log
Duration (ms) by Function - Timeline
Last 24 Hours
40K
20K
0
12:00 PM
06:00 PM
12:00 AM
26 Mar 18
s3_read
firstfunction
awslambda1
mytest
MaxMemoryUsed by Function - Timeline
4,888
2,444
0
Duration (ms) by Functi...
Recent Request Details
Last 15 Minutes
function
requestid
duration
billedduration
memorysize
maxmemoryused
FirstFunction
AWSlambda1
MaxMemoryUsed (MB) ...
Average Duration vs Average BilledDuration
remainingduration
durationavg
functionname
MyTest
FirstFunction
AWSlambda1
S3_read
0
50
100
Log - Allocated Memory vs Memory Used
remainingmemory
maxmemoryusedavg
Compute Usage (GB-s) by Function
Compute Usage (GB-s) by Function - Timeline
Last 24 Hours
7.5
5
2.5
0
12:00 PM
05:00 PM
12:00 AM
26 Mar 18
AWSlambda1
S3_read
FirstFunction
MyTest

より進んだ発見的統制

リクルートテクノロジーズ様の例は一つの参考になる

大規模アカウントのスケールなのでやりやすいところを抽出して使いましょう

https://speakerdeck.com/rtechkouhou/da-gui-mo-awsgong-tong-ji-pan-shang-falsefa-jian-de-tong-zhi-hefalsetiao-zhan

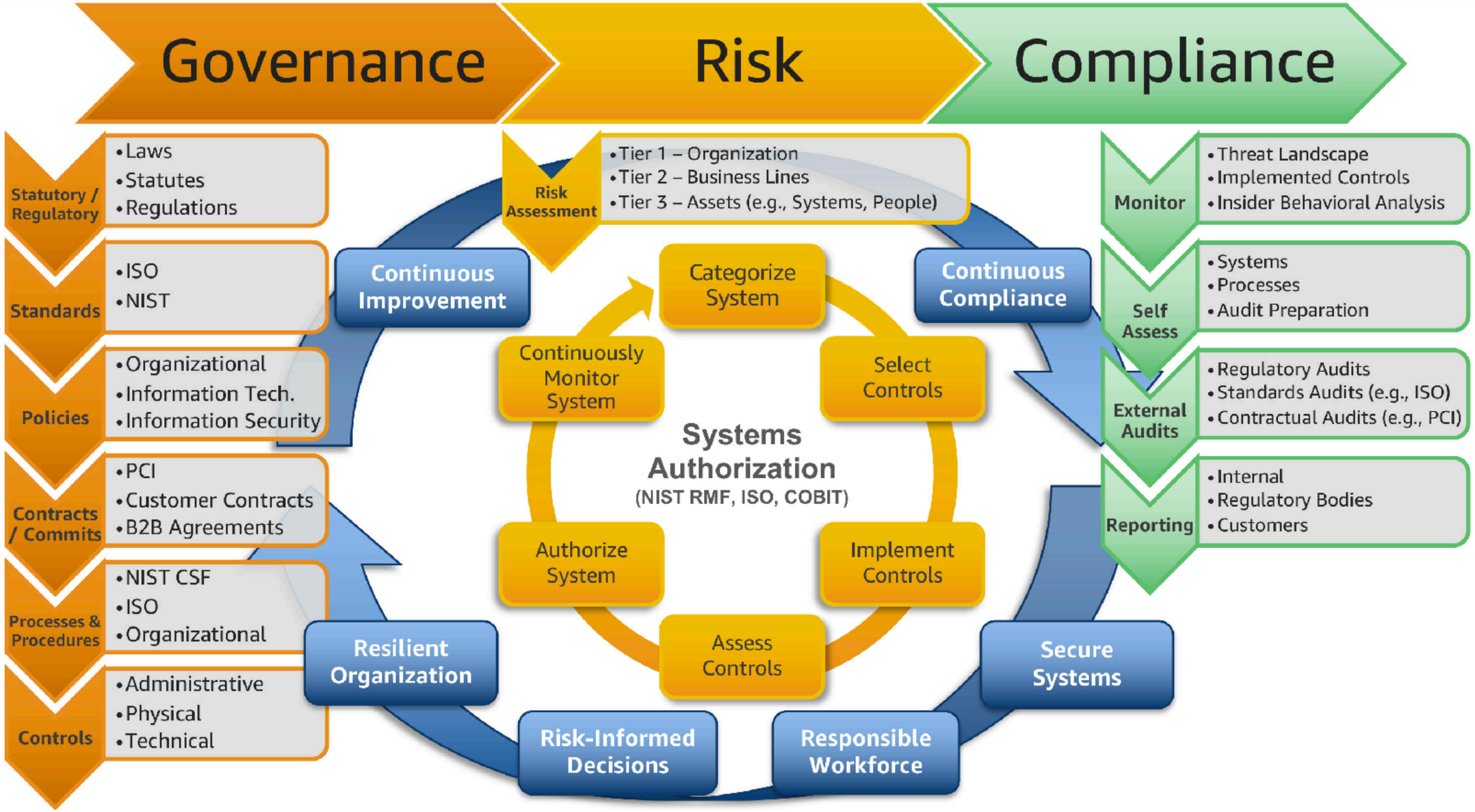
中級

GRC
(ガバナンス / リスク / コンプライアンス)

GRCとは

- ガバナンス・リスクマネジメント・コンプライアンスは共存している
- ガバナンスはビジネス要件に応じた戦略等を確立
- リスクマネジメントでガバナンスと評価されたリスクを紐付けて優先順位付け
- コンプライアンスは要件に応じた統制の遵守と監視

GRCのフィードバックループ



https://aws.amazon.com/jp/blogs/security/scaling-a-governance-risk-and-compliance-program-for-the-cloud/

難しそうだけど簡単に言うと

ビジネスに見合う
リスク管理をして
適度に監査をする
のを状況に合わせてひたすら回す

わかりやすい？

GRCループを回すためにやること抜粋

- FISCやPCIなど準拠するコンプライアンス要件を確認
- NIST CSFなどの適用できるフレームワークを理解
- CCoE作る
- システムとビジネスをリンクしてリスク管理
- 監視・アラートの自動化
- コンプライアンス状況の可視化

参考にするドキュメントいろいろ再掲

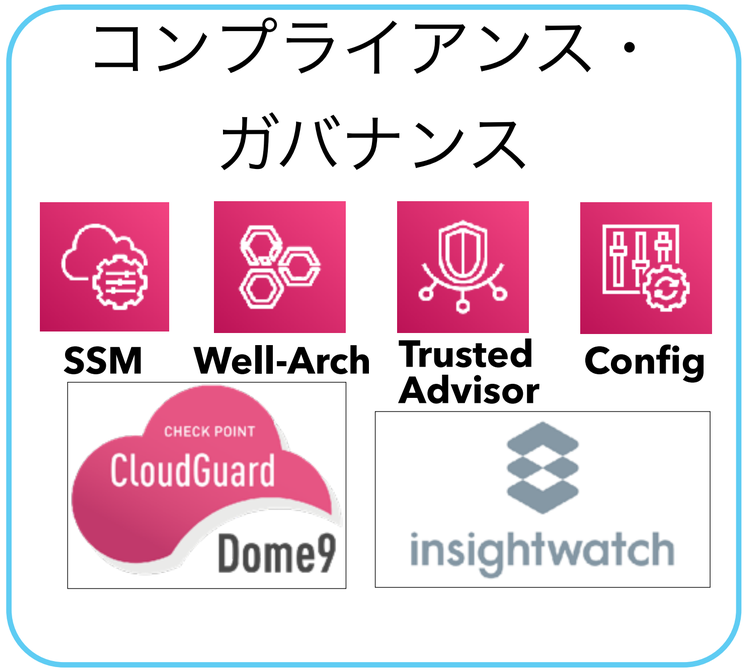
コンプライアンス要件やフレームワーク等

- NIST サーバーセキュリティフレームワーク(CSF)
  - https://aws.amazon.com/jp/blogs/news/updated-whitepaper-now-available-aligning-to-the-nist-cybersecurity-framework-in-the-aws-cloud/
- AWSセキュリティベストプラクティス
  - https://d1.awsstatic.com/whitepapers/ja_JP/Security/AWS_Security_Best_Practices.pdf
- Well-Architectedフレームワーク
  - https://aws.amazon.com/jp/blogs/news/aws-well-architected-whitepaper/
- PCI/FISC/HIPPA等各種コンプライアンスのドキュメント
  - https://aws.amazon.com/jp/compliance/programs/

CCoE作る

- Cloud Center of Excellence （CCoE）
- 多様な専門知識を持ったチーム
- 早い段階からクラウドに取り組み、成熟して全体のクラウド活用の支援に回る役割
- 部署横断でバーチャルな組織にする場合も
- 進め方の例
  - https://aws.amazon.com/jp/blogs/enterprise-strategy/using-a-cloud-center-of-excellence-ccoe-to-transform-the-entire-enterprise/

コンプライアンス・ガバナンス系サービス



- 最低限のセキュリティチェックが可能なinsightwatchは無料のためすべてのアカウントで有効活用してほしい
- AWS Trusted Advisorも必ずチェック
- アプリ規模が大きい・ステークホルダーが多い・コンプライアンス用件が高い場合には追加の活用を検討
- Dome9はPCI DSSなどのコンプライアンスチェックができる他、Security Groupからネットワークを可視化したりIAMの管理を行えるので大規模アカウント・マルチアカウントの管理コストを大きく削減できる

初めやすいコンプライアンス: CIS Benchmark

- CISはLinuxやApache等様々なセキュリティ基準を作成している団体
- CIS AWS Foundations BenchmarkとしてAWSのセキュリティチェックの具体的な項目を定義している
- 弊社提供のセキュリティチェックツール「insightwatch」で無料で診断可能

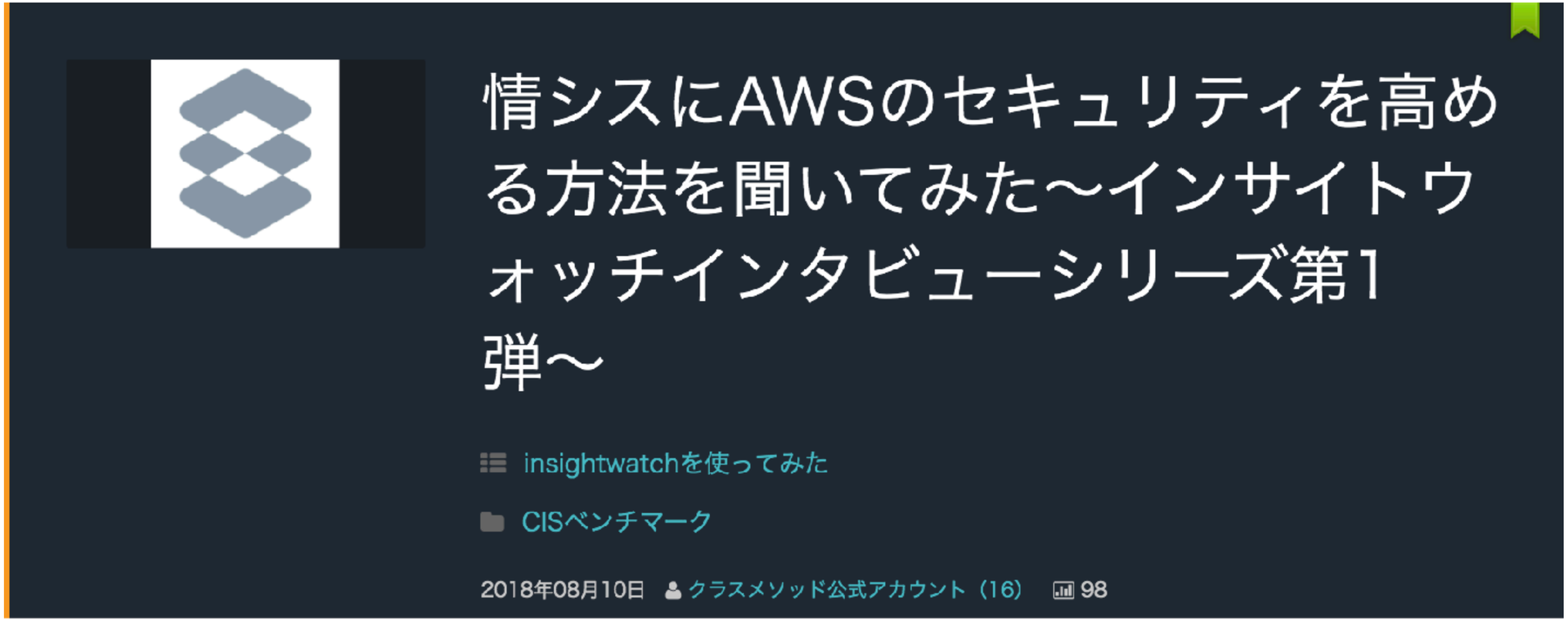
「小さな発見を 大きな安心に」をコンセプトにAWS環境を診断しレポートを出力するツール

insightwatch

https://insightwatch.io/

insightwatchをどう使うといいか

情シスにAWSのセキュリティを高める方法を聞いてみた〜インサイトウォッチインタビューシリーズ第1弾〜



https://dev.classmethod.jp/cloud/aws/insightwatch-interview_1/

Dome9でよりたくさんのコンプライアンスに対応

- Dome9はコンプライアンス対応やガバナンスを利かせるためのSaaS
- 対応コンプライアンス
  - HIPAA
  - PCI DSS
  - GDPR
  - NIST 800-53
  - FedRAMP
  - ISO 27001
  - CIS Foundations Benchmark

あわせて読みたい

Developers.IO 2019 in OSAKAで「Dome9ではじめるAWSセキュリティリスク管理」を話しました #cmdevio

Dome9のよく分かる解説

コンプライアンス以外のIAM / Security Group管理機能も強力！

https://dev.classmethod.jp/etc/developers-io-2019-in-osaka-dome9-security-risk-control/

より大きな環境でガバナンスを利かせる具体策

- キーワードやサービス
  - マルチアカウント戦略
  - AWS Organizations
  - Landing Zone
  - Control Tower

再掲: あわせて読みたい

AWSアカウントとVPC、分ける？ 分けない？: 分割パターンのメリット・デメリット

https://dev.classmethod.jp/cloud/aws/account-and-vpc-dividing-pattern/

アカウント分割(戦略)の殿堂入り記事

マルチアカウント戦略

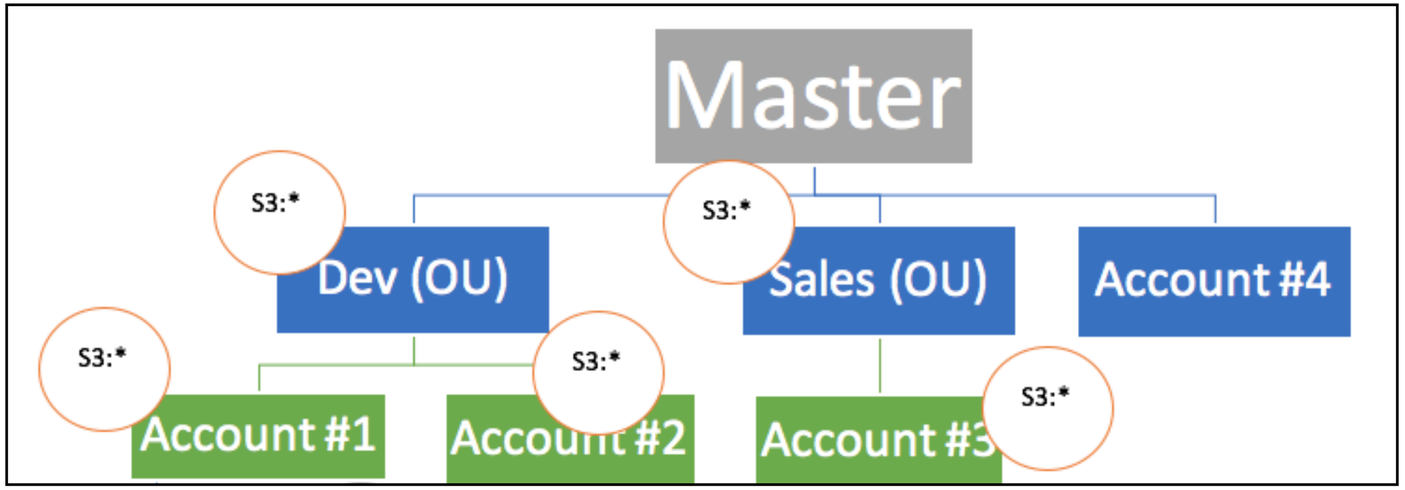
- 扱うAWS環境が多くなる場合には、環境毎にAWSアカウント自体を分けるほうがいい
- IAMやネットワークを完全に切り離すことができる
  - = 責任分界点やセキュリティの境界が明確になる
- システム毎分けるか、システム + ステージ(開発 / 検証 / 本番等)で分ける
- オススメはシステム + ステージ
  - 開発環境のセキュリティがゆるくてやられるパターンがよくあるので、そこから影響が本番環境まで波及しないように

マルチアカウント戦略

- マルチアカウントでのユーザ管理
  - マルチアカウント時のIAMは1箇所にまとめてSwitch Roleを利用して各アカウントにアクセスする
  - あるいはOneLoginなどのIdPに寄せる

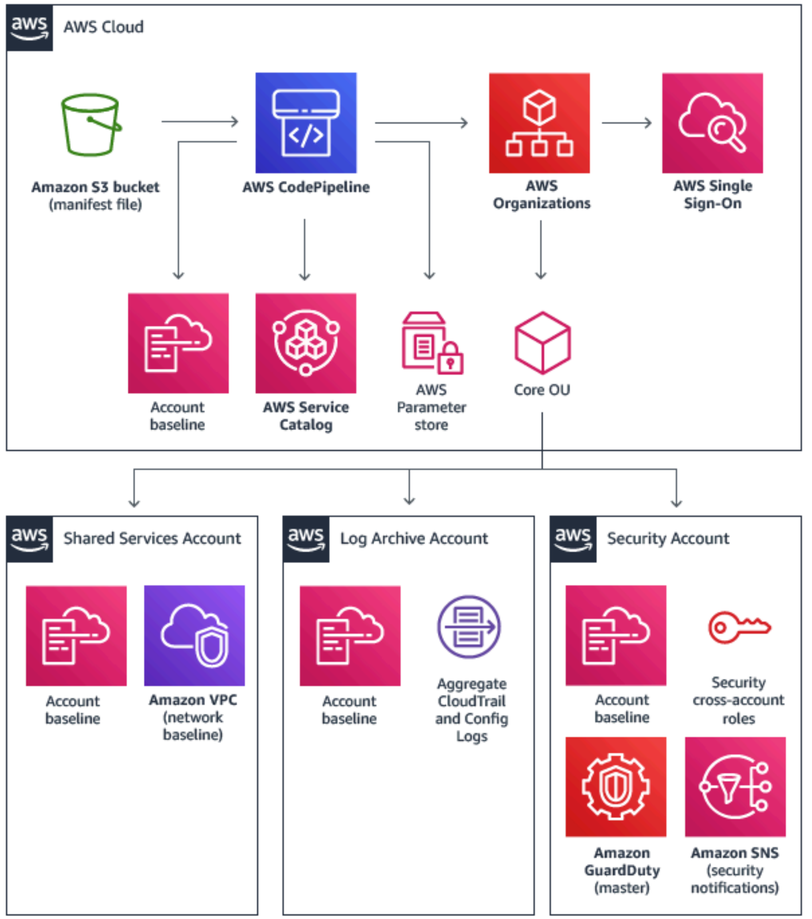
AWS Organizations

- マルチアカウントの管理を行うサービス
- OU(組織単位)を階層的に作成してAWSアカウントを所属させる事ができる
- サービスコントロールポリシー(SCP)を利用して利用可能な権限を制御できる
- 代理店経由のAWS利用の場合制約があることもあるので注意



Landing Zone

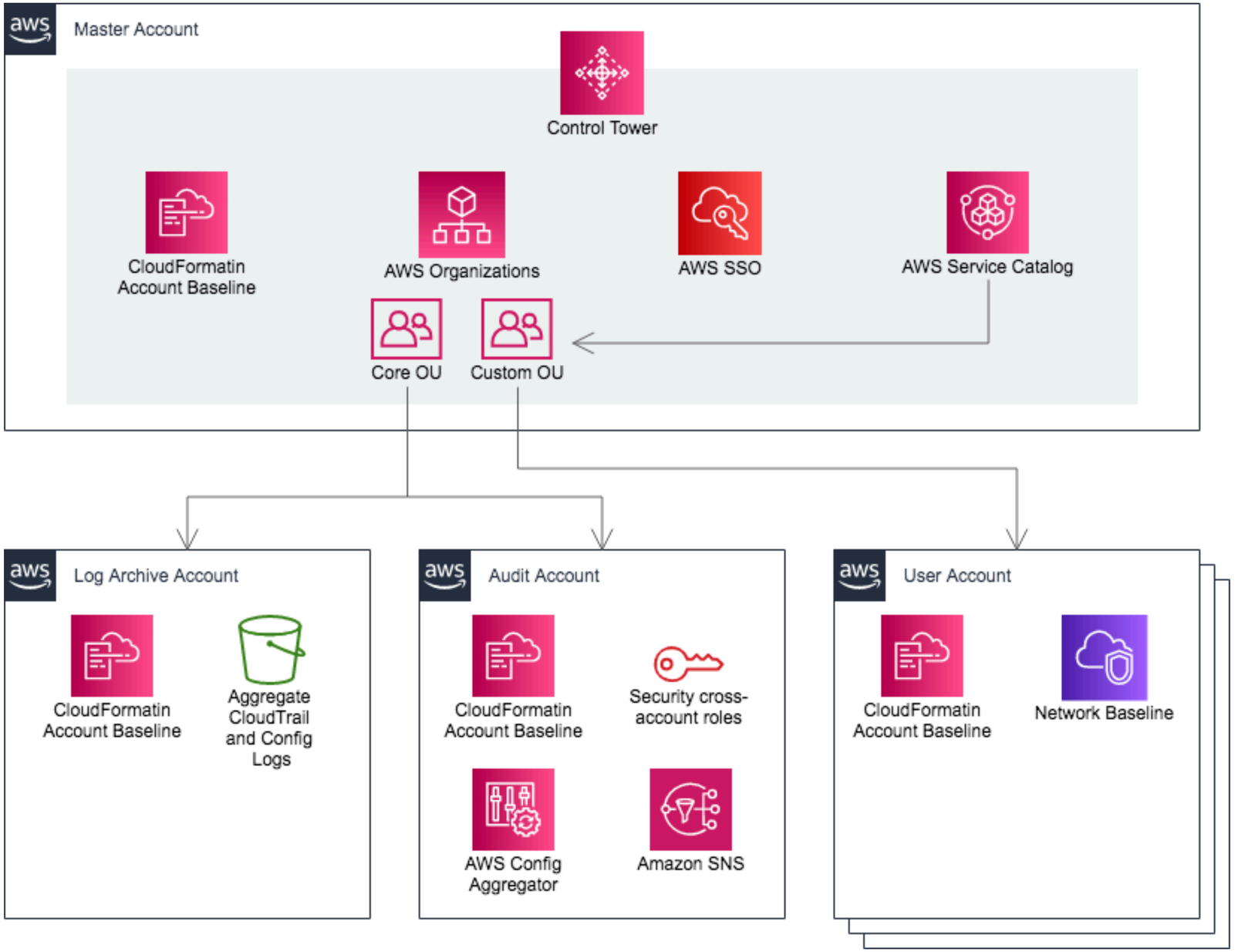
- Organizationsをベースに様々なサービスを組み合わせてマルチアカウントでガバナンスやセキュリティを確保する考え方
- AWS Landing ZoneソリューションとしてCloudFormationテンプレートの提供もあり
- Organizationsを利用していると制約がある場合があるが、Organizations無しでも同じようなマルチアカウント連携は可能なので参考になる



Control Tower

- Landing Zoneをマネージドで提供するサービス
- AWS Landing Zoneソリューションとは少しアーキテクチャが違う
- ダッシュボードで各アカウントのコンプライアンス状況が把握できる
- Organizationsを利用するので同じような制約あり
- かつ新規Organizationsを作成する必要がある
- 現状東京リージョンでは提供されていない

Control Towerのアーキテクチャ



Control Towerのアーキテクチャ

- マスターアカウントでControl Tower作成
- Organizationsが作成され各ユーザはAWS SSOで管理される
- 管理用のコアアカウント2つ作成
  - ログアーカイブ: TrailやConfigログ集約
  - 監査: GuardDutyやConfig Rules等各種セキュリティアラート集約 + 各アカウントへのAdmin権限
- Service Catalogから決められたパターンのAWSアカウントを払い出す(VPCやコンプライアンス設定入り)

AWS Control Towerのラボをやりながら学んでみた-B1セットアップ編

チュートリアルをやりながら学ぶシリーズ

https://dev.classmethod.jp/cloud/aws/aws-control-tower-labs-b1/

Control Towerへの期待

- まだまだ制約が多い
  - 代理店経由での提供がしづらい
  - 新規Organizationsを作成しないといけない
  - 東京に来てない
- この辺が解決されたらいい感じにマルチアカウントの管理が楽になるかも
- しかしながら、現状でも同じようなことはOrganizationsを利用しなくてもできるので参考にしていく

AWS環境の監査

- AWSはオンプレミスと比べて新しい要素が沢山
- 社内の監査を行う部署(大企業にしか無いかもだけど)は古いチェックシートのまま、古い考え方(ポリシー)のままでは正常に監査できない
  - AWSを利用している部署で説明すればOKは監査していないのと同じ
- 監査部門もクラウドジャーニーを辿っていく必要がある

AWS環境なら監査も進化できる

- 気が重いかもしれないが、実はすごいメリットがある
- AWS環境はAPIを駆使して監査ができる
- コンプライアンス要件を冗長で解釈によってブレが大きい文章で定義するのではなく、コードで定義できる
- より明確なコンプライアンス要件となる
- Compliance as Code(CaC)と呼ぶ

Your first compliance-as-code - GRC305-R - AWS re:Inforce 2019

re:Inforce2019で行われたCaCセッション

https://www.slideshare.net/AmazonWebServices/your-first-complianceascode-grc305r-aws-reinforce-2019

AWS re:Inforce 2019: Cloud Control Fitness (GRC202)

これからAWS環境のGRCをやっていくエントリー向けセッション

https://www.youtube.com/watch?v=61pK22dadd0

あわせて読みたい

- re:Inforce2019の注目セッション動画一覧
- ジャンル
  - Security Deep Dive leadership session
  - Foundational Security leadership session
  - Governance, Risk & Compliance leadership session
  - Aspirational Security leadership session
- https://aws.amazon.com/jp/blogs/security/reinforce-2019-wrap-up-and-session-links/

中級まとめ

- 発見的統制 / インシデントレスポンスの体制を作ってなにか起きても大丈夫なようにする
  - 日頃から訓練する
- GRCの観点はすごく大事
  - セキュリティは一人でやるものではないのでみんな巻き込む
  - 監査も自動化できる！
  - 攻めのセキュリティを実践していく

全体まとめ

話した内容

- 全般
  - ゲートからガードレールへ
  - すべての人がBuilders
  - 参考になるドキュメントを使って自分たちに応用
- 初級
  - AWSはきちんと覚えつつOS/アプリは従来どおり + α
  - 運用設計！
- 中級
  - 発見的統制 / IRの体制作る
  - GRCを意識する

行動に移せるように

- 網羅的に色んな要素に触れましたが気になるものはありましたか？
- 技術的な話から組織論まで持ち出してみました
- 一つでも多く持って帰って実践してください
- 資料は公開されているので活用してください
- 自分の領域だけじゃなくて周りを巻き込んで攻めていきましょう

攻めのセキュリティを
実践していこう

補足

各種情報収集

- BlackBelt
  - https://aws.amazon.com/jp/aws-jp-introduction/
- AWS Security Blog
  - https://aws.amazon.com/jp/blogs/security/
- AWS Developer Forums Security Category RSS
  - https://dev.classmethod.jp/cloud/aws/aws-rss-feeds/
- ALAS (Amazon Linux Security Center)
  - https://alas.aws.amazon.com/

各種情報収集

- re:Invent / re:Inforce
  - AWS最大のイベント / 最大のセキュリティイベント
  - 現地参加は特に刺激的！
- JAWS-UG (日本のAWSユーザーグループ)
  - https://jaws-ug.jp/
  - Security-JAWS (セキュリティ特化の支部)
    - https://s-jaws.doorkeeper.jp/
  - JAWS DAYS (年に一度のJAWS最大イベント)
    - https://jawsdays2019.jaws-ug.jp/

classmethod