金融エンタープライズのミッションクリティカルシステム移行時の勘所

小出淳二
フィンテック事業室 プリンシパル
株式会社QUICK

矢内隼人
デジタル推進本部
株式会社QUICK

aws SUMMIT ONLINE

Agenda

- 会社紹介
- ネットチャンネル向け株価配信サービス移行のポイント
  - AWS環境全体のセキュリティ・ガバナンス
  - オンプレ～AWS間での設計考慮点
  - Multi-AZ構成の設計考慮点
  - インスタンスタイプの選定
  - オンプレからAWSへのサービス移行方式
  - WAF選定のポイント

ミッションクリティカルシステムとは

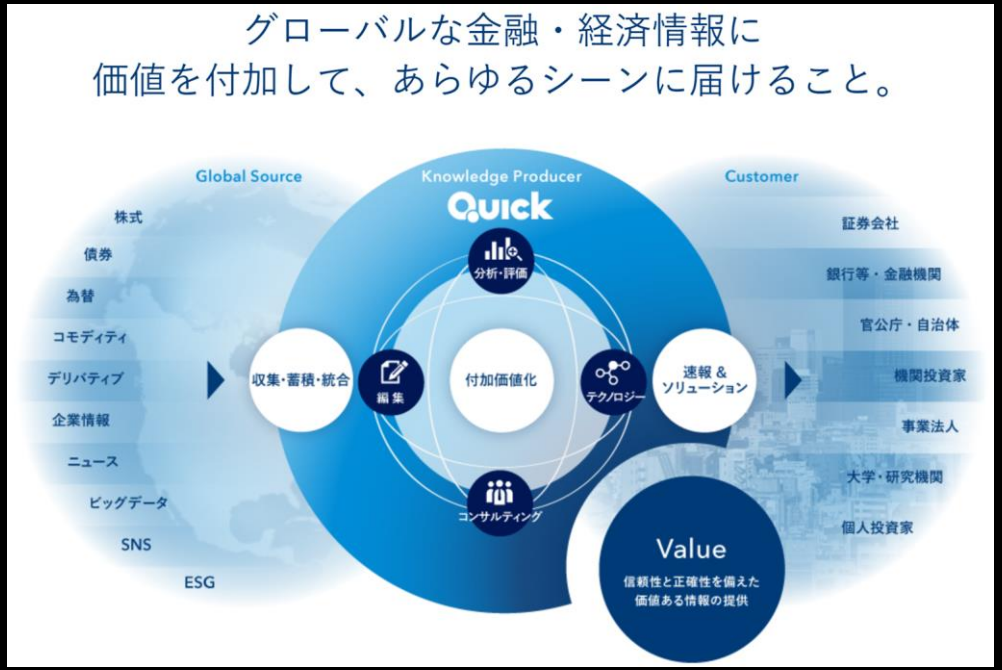
- QUICKにおける「ミッションクリティカルシステム」とは
  - QUICKにとっての本業サービスであること
  - 社会的影響が大きい証券・金融のお客様を対象にしたサービスであること
  - リアルタイムデータの提供をしておりサービスの遅延や停止が許容されないこと

会社紹介

aws SUMMIT ONLINE

会社紹介

- 株式会社QUICK
- 1971年創業
- 日本経済新聞社グループ
- 日本の証券・金融市場を支える情報インフラの役割
- 「日経平均株価」の算出



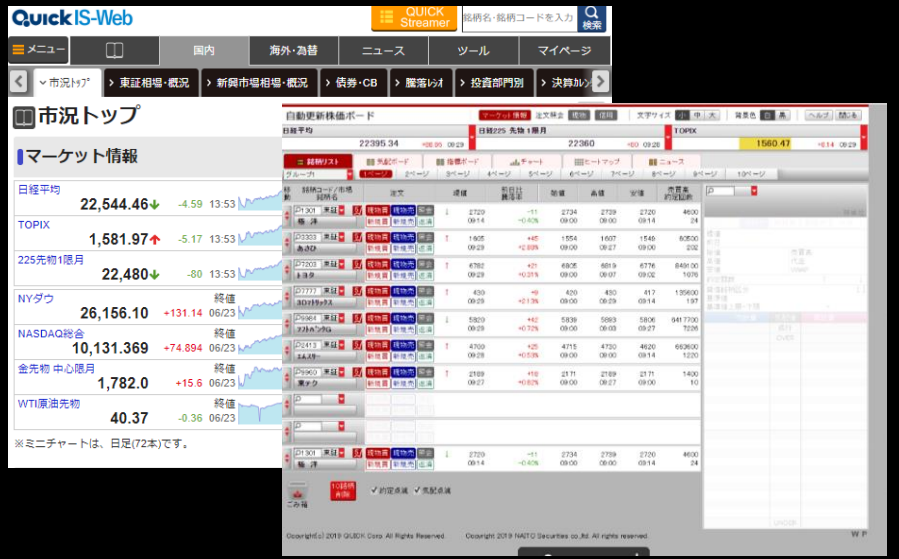
サービス紹介

- Qr1

証券・金融プロフェッショナルの皆様へ提供するスタンダードなMarket Viewer

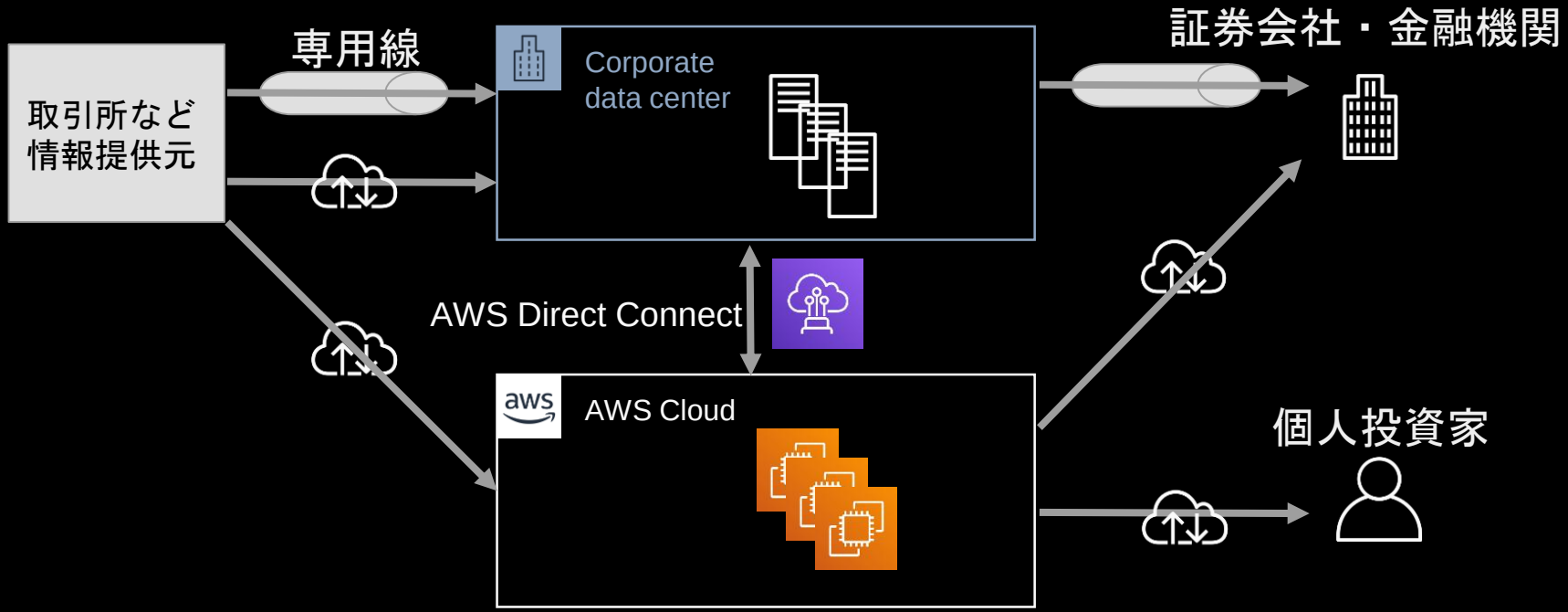
- ネットチャンネル向け株価配信サービス

B2B2C形式で個人投資家の皆様へ提供する証券・金融情報サービス



QUICKの全体のシステム構成

多様な情報源から収集したデータ（株式、債券、為替、コモディティ、デリバティブ、企業情報）を加工・分析し、必要なお客様に様々なチャンネルで届け、証券・金融市場に関わる意思決定をサポートする。

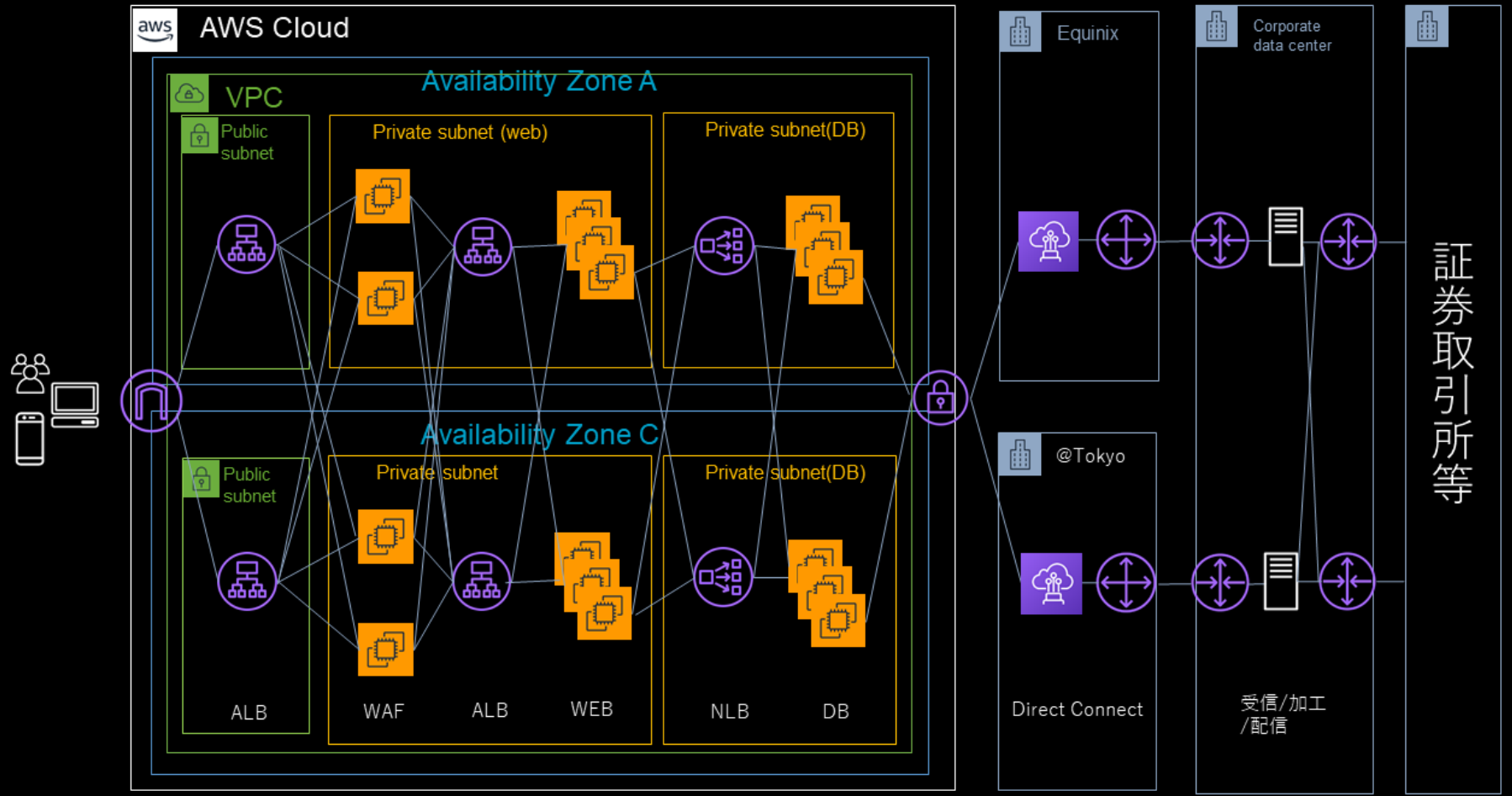


ネットチャンネル向け
株価配信サービス移行のポイント

aws SUMMIT ONLINE

QUICKはミッションクリティカルな証券会社・銀行のネット向け情報サービス（EC2 1000台規模）を一年がかりで2019年11月にAWSに完全移行しました。

全体構成図


AWS Cloud
VPC
Availability Zone A
Public subnet
Private subnet (web)
Private subnet(DB)
Availability Zone C
Public subnet
Private subnet
Private subnet(DB)
ALB
WAF
ALB
WEB
NLB
DB
Equinix
@Tokyo
Direct Connect
Corporate data center
受信/加工/配信
証券取引所等

ネットチャンネル向け株価配信サービス移行のポイント

- 移行に関するポイントをいくつかご紹介します
  - AWS環境全体のセキュリティ・ガバナンス
  - オンプレ～AWS間での設計考慮点
  - Multi-AZ構成の設計考慮点
  - インスタンスタイプの選定
  - オンプレからAWSへのサービス移行方式
  - WAFの移行

AWS環境全体の
セキュリティ・ガバナンス

aws SUMMIT ONLINE

AWS環境の使い分け

- 設計思想の異なるAWS環境を案件に応じて使い分けている
- AWS共通基盤環境
  - 自由度　＜　セキュリティ・ガバナンス
  - ゲートキーパー型環境
- AWS共通基盤”外”環境
  - 自由度　＞　セキュリティ・ガバナンス
  - ガードレール型環境
- 今回話題にする「ネットチャンネル向け株価配信サービス」はQUICKにおける「ミッションクリティカルシステム」なので、AWS共通基盤環境（ゲートキーパー型環境）上で稼働

AWS共通基盤環境

環境のコンセプト

- オンプレのデータセンタの延長としてAmazon EC２を中心に利用する※１
- Amazon EC２以外のAWSサービス※２も利用することで、開発速度・効率性を高める
- オンプレとの接続、セキュリティ・ガバナンス観点の考慮、監視・運用体制、各種ルール、社内申請などが整備された環境
- その枠内でセルフマネージドな（オンプレに比べれば）自由度の高い環境

※1 現在EC2台数は1500台程度の規模感。

※2 現在30個以上のAWSサービスが利用可能

Amazon EC2（ELB含む）, Lambda, Amazon S3, EFS, Amazon RDS, ElastiCache, Redshift, DynamoDB, Amazon VPC, AWS Direct Connect, Route 53, API Gateway, CloudWatch, CloudFormation, Athena, CloudFront, Kinesis Data Firehose, IAM, KMS, ACM, WAF, Amazon SNS, SQSなど

AWS共通基盤環境

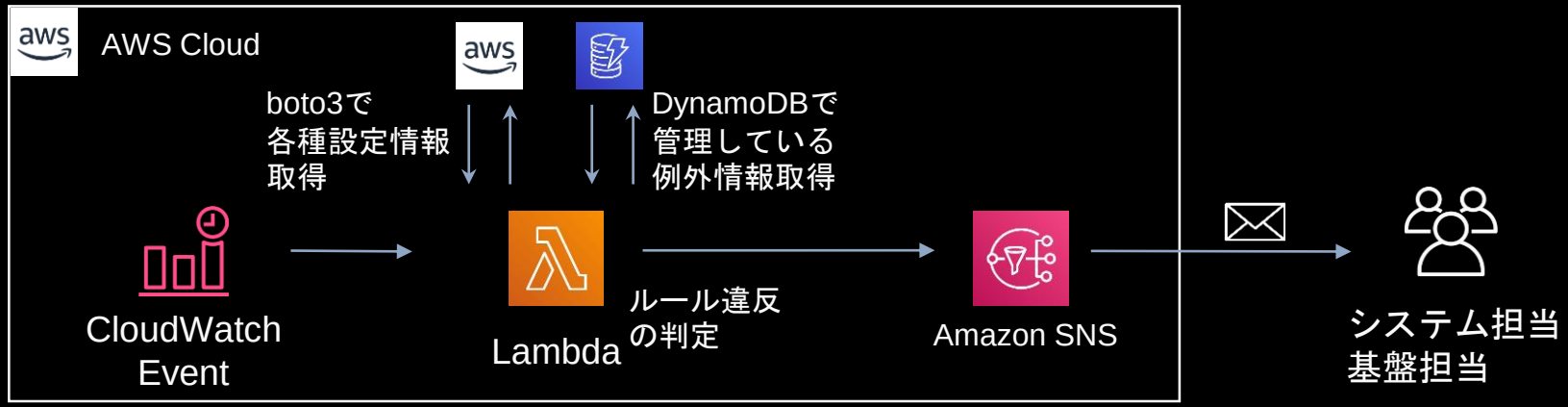
セキュリティ・ガバナンスのポイント→ゲートキーパー型

- 利用可能サービスの限定・ルール化
  - 基盤担当が各AWSサービスを調査し社内基準に照らして利用可否を判定
  - 利用可能と判断したAWSサービスの利用方法をルール化（セキュリティの観点以外も含め）
- 権限を絞ったユーザの提供
  - IAMユーザを基盤担当からシステム担当へ提供することで、権限を制御
- CloudTrail, AWS Config, GuardDutyなどの有効化

- ルールチェック
  - ルールを定めて権限はなるべく移譲し、自動チェックでルールが守られていることを担保

AWS共通基盤環境

- ルールチェックの例
  - Security Group の違反設定有無判定、Amazon S3バケット アクセス権限設定の違反設定有無判定、Lambda アクセス権限設定の違反設定有無判定など様々。
  - AWS一般論的なチェックだけでなく、社内基準に応じたチェックなどサービスの調査の段階で穴になりそうだと判断したポイントをチェックしている。
  - Config Rulesだけではなく、サーバレスサービスを活用し自前でチェックの仕組みを構築。



AWS共通基盤”外”環境

環境のコンセプト

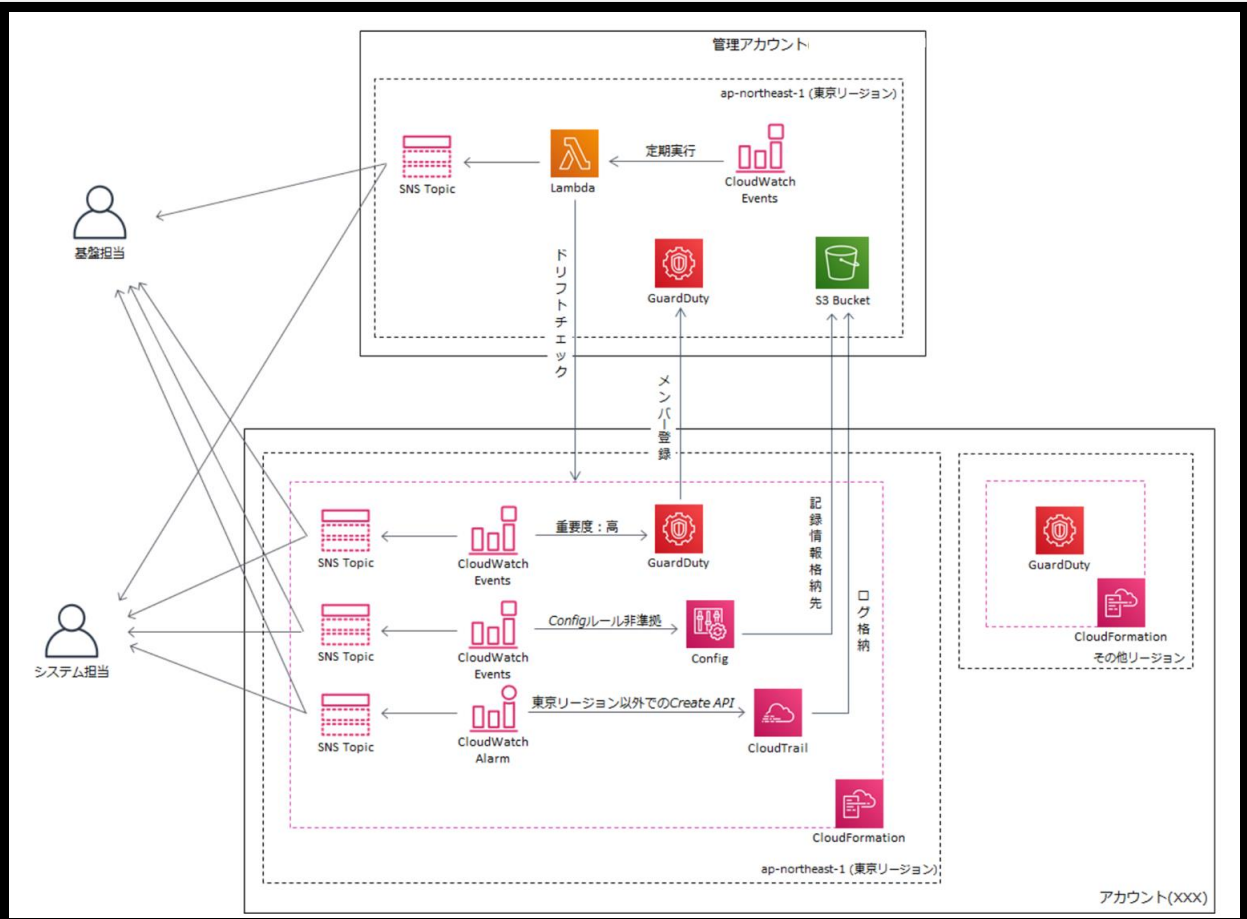
- AWSの積極利用を主目的とし、システム担当者の自由度を最大化
  - 制約をほぼ設けずほぼ全ての権限をシステム担当者に移譲する形でAWSアカウントを提供
  - 基盤担当はガードレールの設置と、AWSアカウント作成、請求代行、ルートアカウントの管理などだけを担当
  - システム担当者にAWS資格保有者（SAA）がいることが環境提供の条件
- 自由度を下げずにセキュリティ・ガバナンスを高める
  - 利用可能なAWSサービスを限定しない
  - 各AWSサービスに関する利用制約を作らない
- システム担当者の手を止めない
- リスクや問題があったときに気づくことが出来る

セキュリティ・ガバナンスのポイント→ガードレール型

- ガードレールのベースはLanding Zone
  - Landing Zone用CloudFormationテンプレートを自社に合わせてカスタマイズ。
- CloudTrail, AWS Config, GuardDutyなどを有効化

- ルールチェックを実施
  - ゲートキーパー型環境のように各AWSサービスの検討を経て穴になりそうな点が洗い出せている状態ではないので、独自ルールでチェックすることは不可能。
  - したがって基本的にはチェックはConfig Rulesの標準ルールを用いる。
  - 一部IAMや、利用していないはずのリージョンのチェックなど独自チェックも実装。
- ガードレールの構築を自動化
  - AWSアカウントが増加していくことを見据えガードレール構築を（半）自動化
  - CloudFormation, Step Functionsなどを活用

AWS共通基盤”外”環境


管理アカウント
ap-northeast-1 (東京リージョン)
SNS Topic
Lambda
定期実行
CloudWatch Events
基盤担当
ドリフトチェック
GuardDuty
S3 Bucket
メンバー登録
SNS Topic
CloudWatch Events
重要度：高
GuardDuty
記録情報格納先
ログ格納
GuardDuty
CloudFormation
その他リージョン
システム担当
SNS Topic
CloudWatch Events
Configルール非準拠
Config
SNS Topic
CloudWatch Alarm
東京リージョン以外でのCreate API
CloudTrail
CloudFormation
ap-northeast-1 (東京リージョン)
アカウント(xxx)

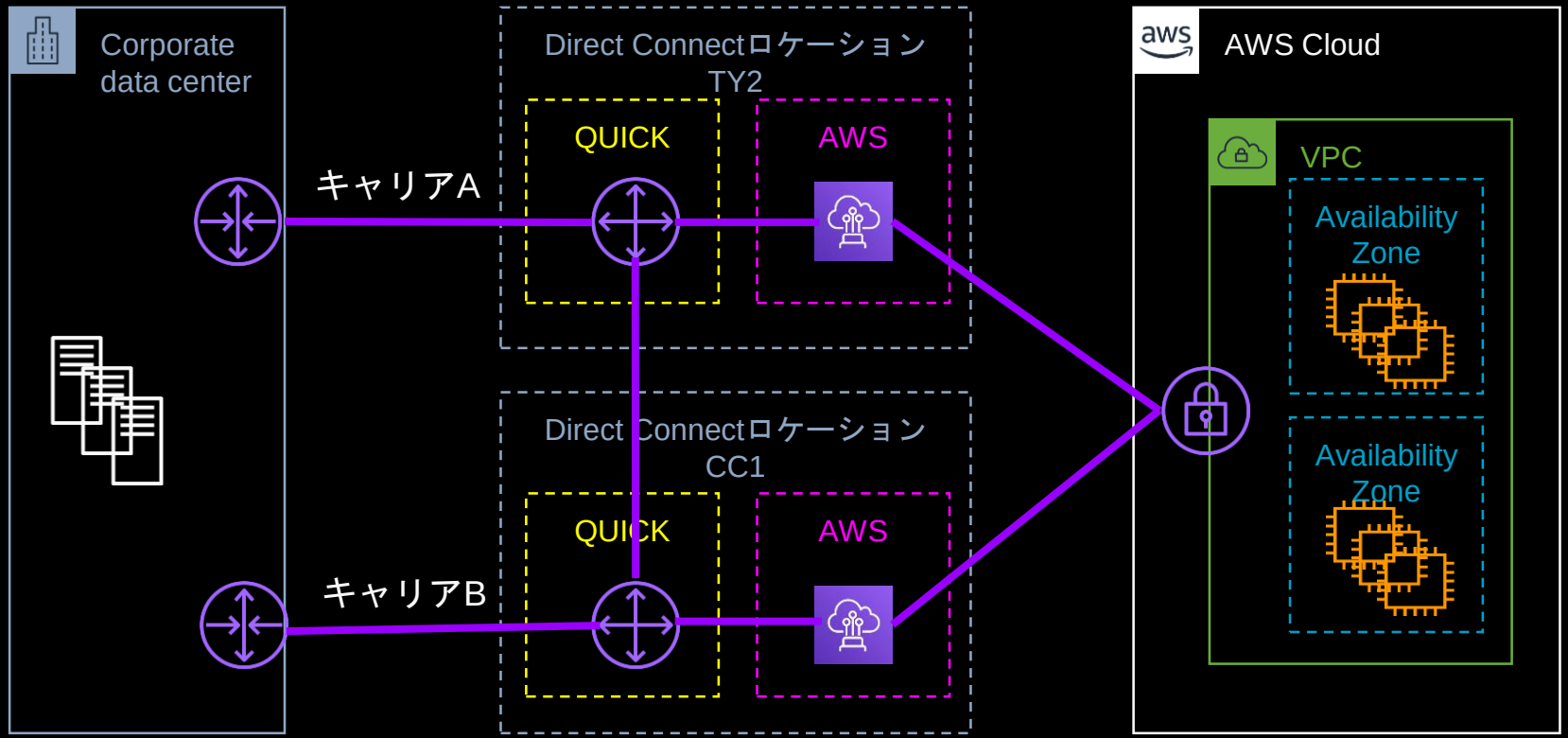
AWS環境全体のセキュリティ・ガバナンス のまとめと今後の課題

- AWSの積極利用を進める際はガードレール型の考え方が現実的。
- ただしゲートキーパー型に比べるとガードレール型は考慮が漏れる範囲が広いので、全てのシステムがガードレール型で十分だとは考えていない。
- ミッションクリティカルシステムに関しては、今のところゲートキーパー型が望ましいと考えている。ただし今後それらの領域にも多様なAWSサービスを活用していく予定なので、その際は何が最適解なのかまだ見えていない。
- 最初から最終形を見据えて綺麗に進んできたわけではなく、継続的に地道な改善活動を続けている。

オンプレ～AWS間での設計考慮点

aws SUMMIT ONLINE

オンプレ～AWS間での設計考慮点



ラックを借り自社所有のルータなどを設置

AWS管轄領域

AWSアカウント、VPCが複数ありDXのVIFが各VGWと接続

オンプレ～AWS間での設計考慮点

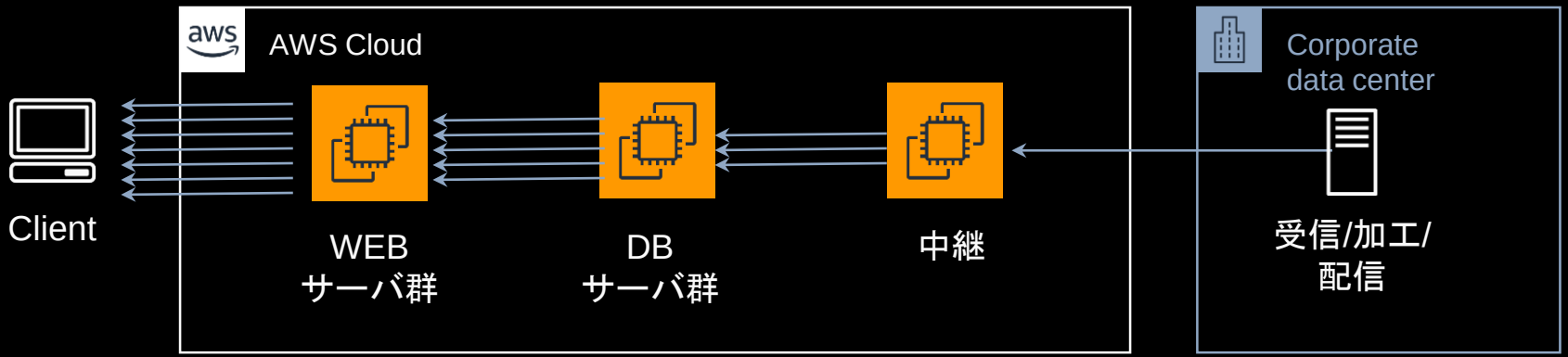
- 冗長化
  - パートナーサービス利用では独自設計余地が足りなかったので、拠点間のWAN回線やAWS Direct Connectロケーション内のラックは自社で調達し設計・構築
  - 異なるWAN回線キャリア、異なるAWS Direct Connectロケーションを採用
  - Act-Stb設計（経路制御はBGPのLocal PreferenceとAS Path Prependで実現）を基本とする
  - WAN回線などの障害でAWS Direct Connectの切り替わりを発生させないようAWS Direct Connectロケーション間をWAN回線で接続

オンプレ～AWS間での設計考慮点

- 障害時の通信影響
  - リアルタイムデータの提供をしているので、秒単位の通信断を気にする要件あり。
  - AWS Direct Connect部分はBGPに加えBFDを採用し、障害検知と切り替わりに要する時間を短縮
  - 実際はAWS Direct ConnectのConnectionは切り替わらず、影響リソースが限定された通信断も発生
  - サービス上許容できない通信断が発生した際には、通信先を切り替える、あるいは該当構成をサービスから離脱させるなどアプリケーションレイヤーで考慮が必要。特に、「秒単位の通信断の影響が大きいシステム」「オンプレではネットワークレイヤーで通信断を秒単位で吸収するよう設計していたシステム」など。

オンプレ～AWS間での設計考慮点

- オンプレ～AWS間のAWS Direct Connectは10Gbps回線を複数のシステムで共用している。各々のシステムでオンプレ～AWS間のトラフィックを少なく抑えるようアプリケーションの設計・配置が必要
- オンプレ～AWS間は4～5ms程度の遅延が発生するので、何回も往復するような配置設計は避ける　※数値はAWSの公開仕様ではなくQUICKでの実測値
- 今回は下記構成として極力オンプレ～AWS間のトラフィックを抑えた



Multi-AZ構成の設計考慮点

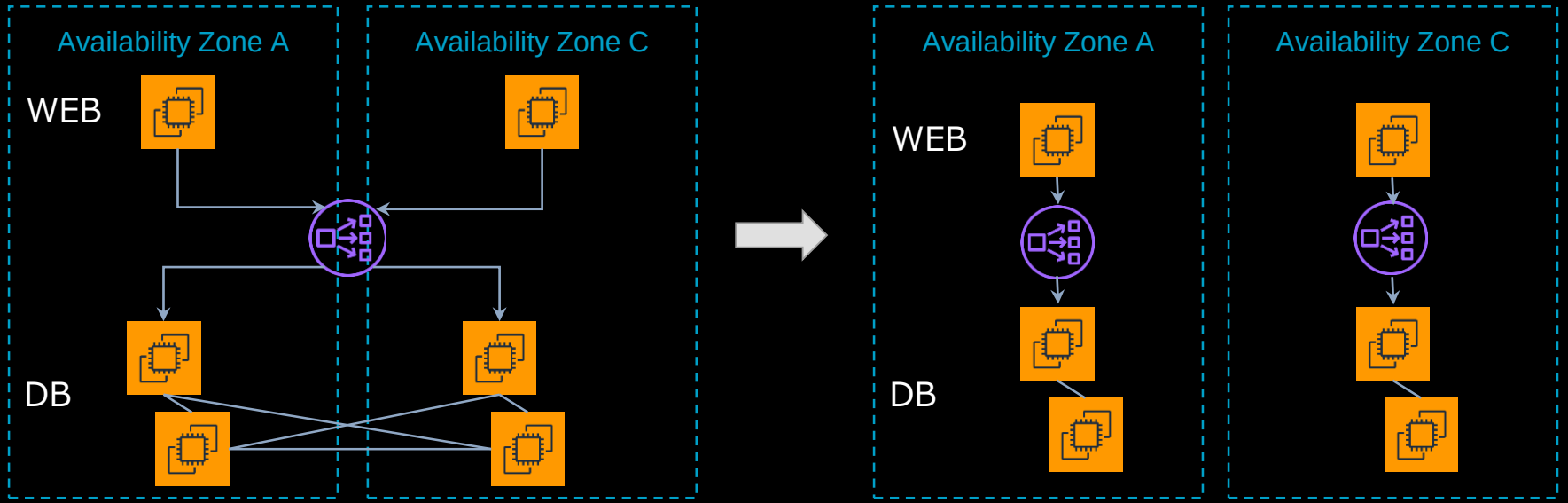
aws SUMMIT ONLINE

Multi-AZ構成の設計考慮点

- AZ間で2.5ms程度の遅延が発生するので数百回積み重なるとサービスに影響のある遅延が発生する。レガシーな密結合アプリは要注意。

  → 各AZ内で通信が完結するようにアプリケーションを配置 (AZ内は0.5ms以下)

  ※数値はAWSの公開仕様ではなくQUICKでの実測値



インスタンスタイプの選定

aws SUMMIT ONLINE

インスタンスタイプの選定

- 新しいインスタンスタイプは価格は下がって性能は向上といいことばかり
- ただしNitro系をかなり初期に採用したため、不具合（現在は解消済み）に直面して移行計画の見直しが発生
- ミッションクリティカルシステムでの採用は枯れてからの方が無難
  - 東京リージョンで利用可能になってから約１年後
  - Amazon RDSでそのインスタンスタイプが利用可能になったタイミング
  - AWS一般論ではないQUICKのノウハウ

オンプレからAWSへの
サービス移行方式

aws SUMMIT ONLINE

オンプレからAWSへのサービス移行方式

- AWSにオンプレ同等のフルセットを構築し、十分検証/テスト実施の上リリースした
- サービス面の移行方式は、各サービス単位でのDNS切替え
- 300近くのサービスを約70ステップに分けて1年かけて移行した
- IPアドレス固定が必須のサービスに関してはALBではなくNLBで構築
- バックエンドにHTTP1.0が必須のレガシーサービスもNLBで構築
- 株価等のDBはオンプレとAWSの両アクティブ構成で、両サイトで最新データを保持
- 登録情報等のDBは週末深夜にオンプレからAWSに移行

WAF選定のポイント

aws SUMMIT ONLINE

WAF選定のポイント

・金融ミッションクリティカルシステムにおけるAWS WAFとWAFアプライアンスの使い分け

・WAFアプライアンス設計のポイント

・WAFアプライアンス よい所、イマイチな所

使い分けのポイント

① 機能要件

　WAFでどういう防御や保護を実現したくて、どこまでの機能が必要なのかを明確に

機能要件例（１）

　誤検知時にセキュリティポリシー（シグネチャ）の修正がピンポイントで出来ること

WAF アプライアンス

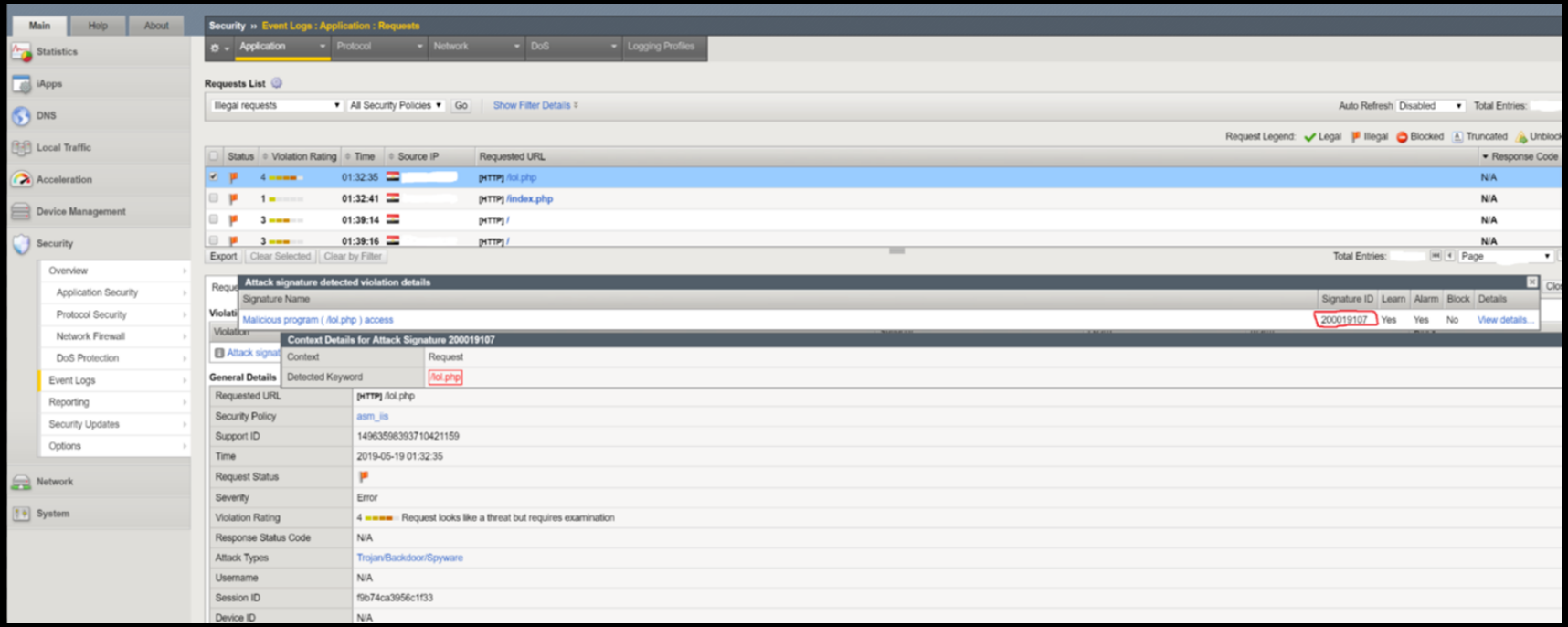
きめ細かい設定が可能

AWS WAF

チューニングをする際、URI/パラメータと言った単位で各シグネチャをON/OFF出来ない

金融ミッションクリティカルシステムにおけるAWS WAFとWAFアプライアンスの使い分け

WAFアプライアンスは管理画面から誤検知した問合せのSignatureIDを特定し、そのSignatureIDだけを無効化可能



機能要件例（２）

　シグネチャーのステージング機能で、新しいシグネチャーで誤検知がないかを事前に確認したい

WAF アプライアンス

　シグネチャのステージング機能あり

AWS WAF

　マネージドルールの場合、ユーザ側でシグネチャ更新のタイミングをコントロール不可

機能要件例（３）

クレジットカード番号の漏洩を防ぎたい（マスキングしたい）

WAF アプライアンス

保護機能あり。

AWS WAF

inboundトラフィックのみのチェックなので、outboundトラフィックは制御不可。

機能要件例（４）

CSRF(Cross Site Request Forgery)攻撃、パラメータ・タンパリング攻撃、セッションハイジャック攻撃への対策をおこないたい

WAF アプライアンス

各種機能で実装可能。

AWS WAF

対応不可。

機能要件例（５）

グラフィカルなレポーティング機能が欲しい

WAF アプライアンス

提供可能。

AWS WAF

グラフィカルなレポーティング画面はない。

機能面ではWAFアプライアンスに軍配が上がるが、どこまでの機能が必要かを十分精査の上、選定する必要がある。

使い分けのポイント

② 非機能要件

- 可用性
- 性能/拡張性
- 運用/保守性
- コスト

- 可用性
  - **WAF アプライアンス**
    - Multi-AZ設計がそれなりに大変
    - インフラエンジニアがいないと設計および運用メンテナンスは難しい
  - **AWS WAF**
    - マネージドでAZ分散

- 性能/拡張性

WAF アプライアンス

柔軟にスケール出来ないので設計が難しい

（インスタンスタイプおよび台数）

AWS WAF

圧倒的なスケーラビリティ

・運用/保守性

金融ミッションクリティカルシステムでの一番のポイント

24h/365d運用要件がある場合、内製化が難しいのでセキュリティベンダーに運用（SOC）を委託するケースが多い。

WAF アプライアンス

オンプレ同様、複数のベンダーを選択可能

AWS WAF

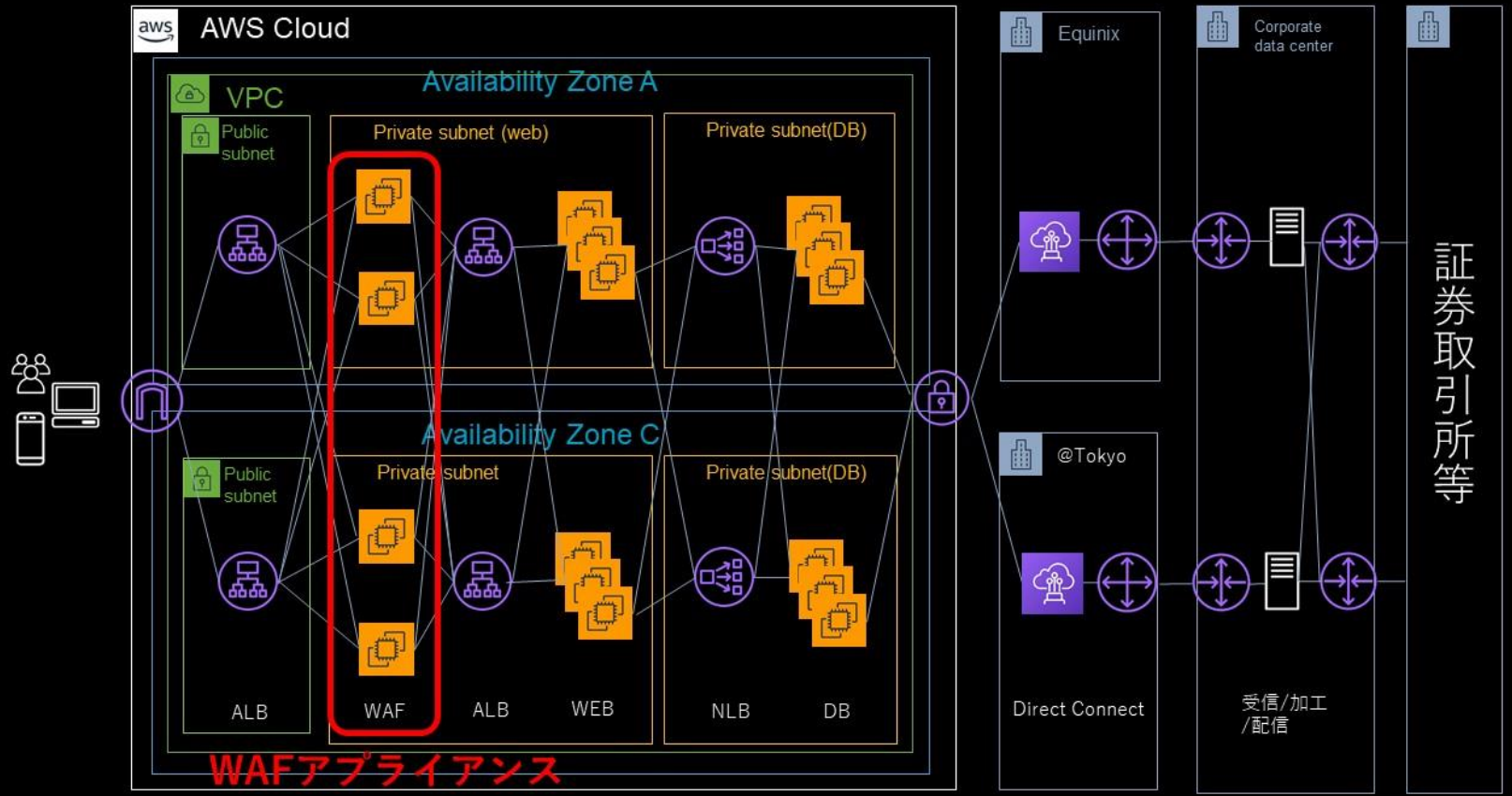
24h/365dで緊急対応が可能なベンダーがなかった

※AWS WAFはShield Advancedと絡めれば 24h/365dのプロアクティブなサポートが得られるものの、未だ日本語のサポートがされていない

金融ミッションクリティカルシステムにおけるAWS WAFとWAFアプライアンスの使い分け

- コスト
  - **WAF アプライアンス**
    - スモールスタートが出来ない
    - 初期の設計・構築費用が必ず必要
  - **AWS WAF**
    - スモールスタート可能
    - 従量課金でコスパがいい

WAFアプライアンス設計のポイント



WAFアプライアンス設計のポイント

- ALBにてWAFアプライアンスのサンドイッチ構成
- Multi-AZのActive-Active
- スケールアウト構成
- ブロックモードで運用

WAFアプライアンス設計のポイント

設計ポイント

① 2台のMulti-AZ HA構成ではなく、スケールアウト可能なMulti-AZ Active-Active構成

・クラスタアドレスを0.0.0.0とし、同期するconfigは全台同じ

・クラスタのポート番号毎にサービスを割り当てる

オンプレ）　サービスA　クラスタ1　172.16.0.1:443

サービスB　クラスタ2　172.16.0.2:443

→Multi-AZだと172.16.0.1,2を共有できない

AWS）　サービスA　クラスタ1　0.0.0.0:50001

サービスB　クラスタ2　0.0.0.0:50002

※各前段ALBでサービスとクラスタのポート番号を紐付け

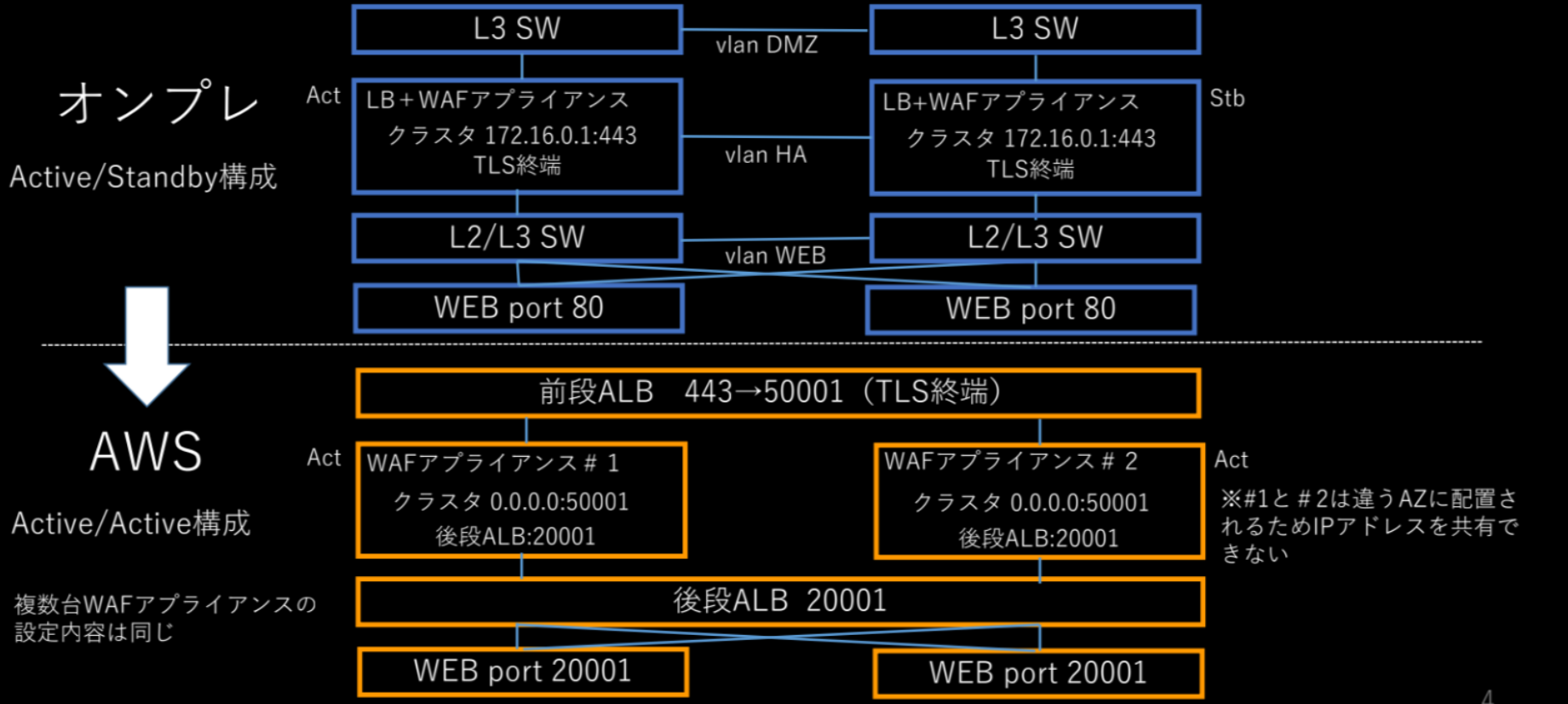
WAFアプライアンス設計のポイント

・SNATのアドレスは別途作成したスクリプトで生成し、同期するconfigは全台同じ

・SNATポート枯渇問題にも対応

　クラスタのIPアドレス、Port番号情報から紐づけるSNAT-IP(クラスタ毎にユニーク)を判定するスクリプトにより、WAFアプライアンス(EC2)に付与できる最大IP数の制限を突破。

WAFアプライアンス設計のポイント



WAFアプライアンス設計のポイント

WAFアプライアンス設計のポイント

② WAFアプライアンス構成変更等の影響を極力受けない設計

・前段ALBでのWAFアプライアンスのターゲット設定はインスタンスIDではなくIPアドレス指定

→WAFアプライアンス再構築等実施した場合も前段ALB設定変更不要

・WAFアプライアンスはWEB側ENIを追加して2NIC構成（WEB側 eth1、運用管理側 eth0）とし、再構築等実施時もeth1のENIを付け替える

・後段ALBのルートテーブルもENI経由で設定変更不要

WAFアプライアンスよい所、イマイチな所

WAFアプライアンスのよいところ

・安定している

・機能が充実している

・充実した管理画面

（リクエスト、レスポンスの内容、インシデントの詳細が可視化）

・スケールアウト構成が可能

WAFアプライアンスよい所、イマイチな所

WAFアプライアンスのイマイチなところ

- 高い（SI費用も必要）
- 設計（サイジング、可用性等）が必要
  お手軽にボタン一つで導入できるものではない（要SI）

WAFアプライアンスよい所、イマイチな所

WAFアプライアンスのイマイチなところ

・構成が複雑　障害時等切り分けが大変

　ALBでサンドイッチしているのでそれぞれのログを分析する必要がある

　WAFアプライアンスの管理コンソールが統合されていないため4台のどこでインシデントが発生しているかの切り分けが必要

　前段ALB→WAFアプライアンス(4台)→後段ALB→EC2

WAFアプライアンスよい所、イマイチな所

WAFアプライアンスのイマイチなところ

・ソフトウェアEOLとの戦いが続く

リリースされてからのサポート期間が３年だったのでオンプレ時代はかなり苦労した。延々下記の繰り返し。

新バージョンリリース
→リリース後HotFixが出るまで様子見
→HotFix出たから検証して本番環境でリリース
（ここまでで１年）
あと２年でサポート切れなので、来年のバージョンアップ計画立案
→翌年バージョンアップの検証＆更新作業

WAF選定のポイント　まとめ

まとめ

・WAFは運用含めて考える必要があり、運用体制の構築が肝要
・金融ミッションクリティカルシステムの要件では、運用も考慮すると現状WAFアプライアンスの一択
・QUICKはWAFアプライアンスを選定
・WAFアプライアンスは運用含め**オンプレと変わらない感**あるので、AWS WAFの今後の機能拡張等に期待したい

Thank you!

小出淳二
junji.koide@quick.jp

矢内隼人
hayato.yanai@quick.jp

aws SUMMIT ONLINE