クラウド事業者から見たインフラの変化 イマドキのWeb屋さんが考えるネットワーク

Amazon Web Services
技術本部　シニアマネージャ　ソリューションアーキテクト
荒木靖宏

自己紹介

- 名前
  - 荒木 靖宏
- 所属
  - アマゾンデータサービスジャパン株式会社
  - 技術本部レディネスソリューション部
    シニアマネージャ
- 好きなAWSサービス
  - Amazon Virtual Private Cloud
  - AWS Direct Connect

共通認識と業界トレンド

2017年には従業員の半数以上が自分のデバイスを業務に使うようになる（Source：ガートナー）

40%のビジネスと技術の方向付けをする専門家が既に複数のクラウドを使うか、使う予定（Source：Forrester）

2020年までに500億のデバイスがインターネットに繋がる（Source：AT&T）

APTやDDoSに晒される機会は2年前の10倍に（Source： AT＆T）

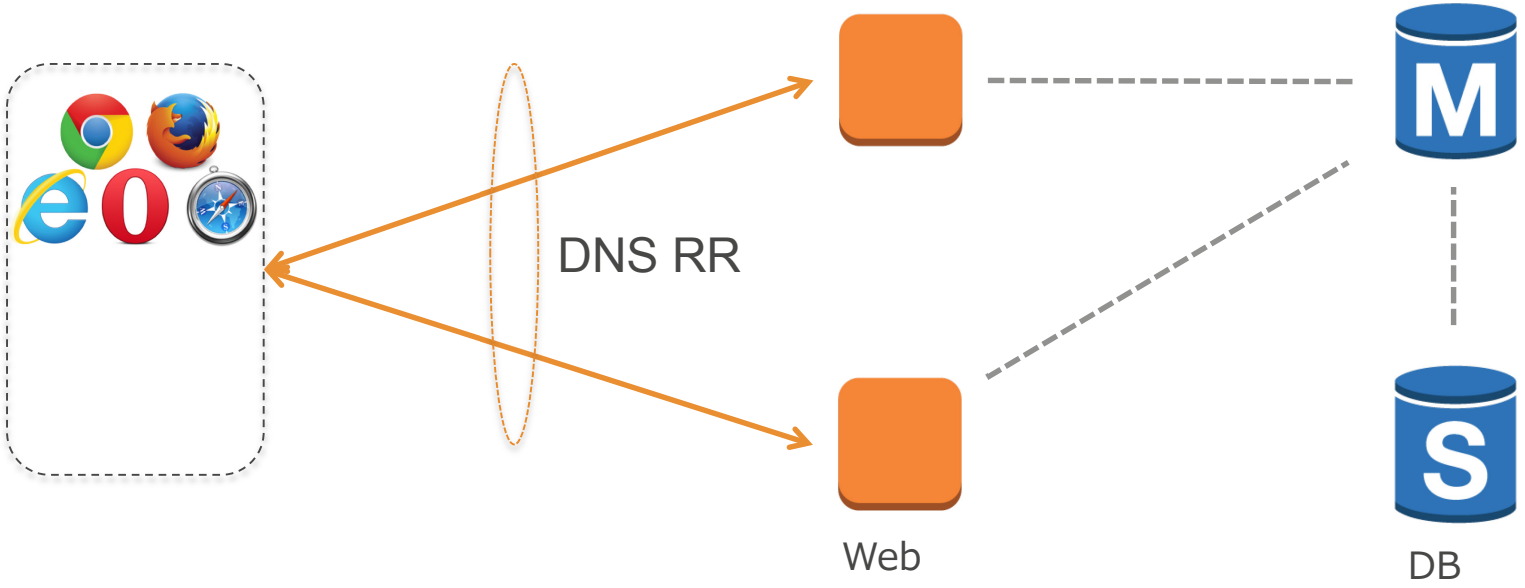
儲けているサービスはどこにいるか

- 2011前半くらいまで、ガラケーの天下
- 2011後半くらいから、スマホが急激に

スマホは今もなお「スマート化」やまず
スマート化された端末むけ
ネイティブアプリケーション化の加速

Webアーキテクチャ変遷

化石時代の一般的なWebアーキテクチャ

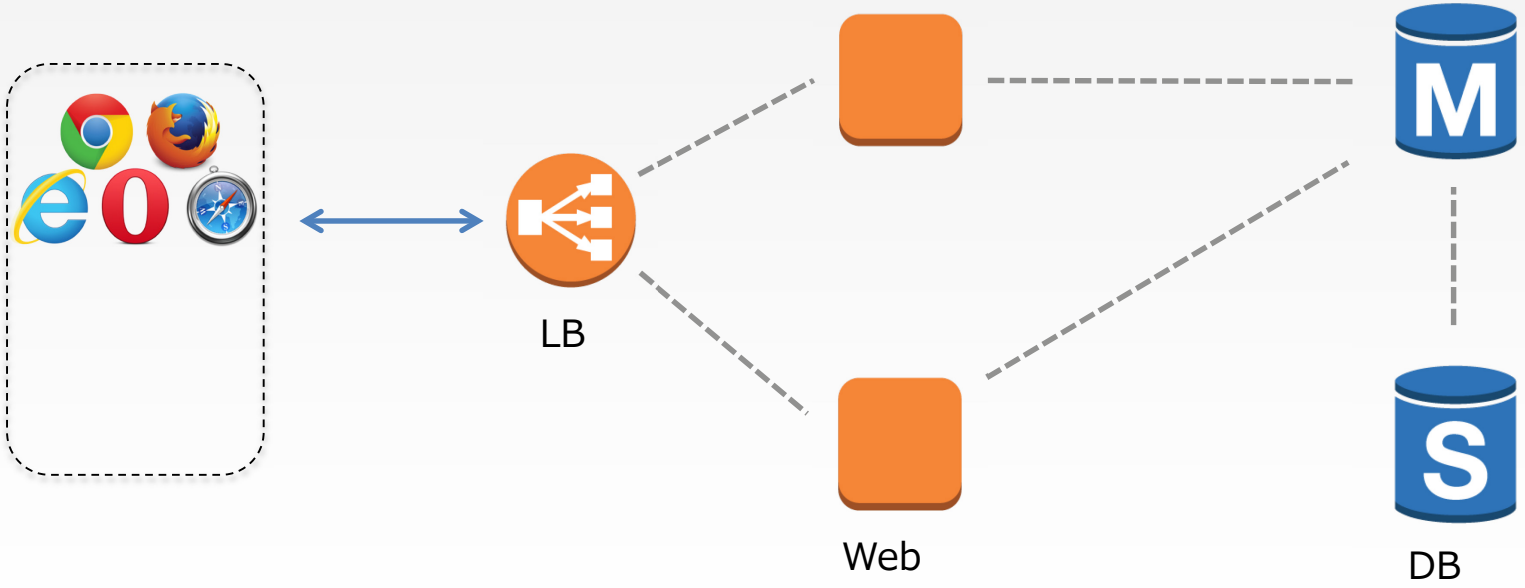


DNS RR

- DNSで名前をクエリ→複数Aレコードの入手
- スケールはしたが、「失敗時」の対応が弱い
- 牧歌的な時代はこれで十分だった

- エンタープライズでは「使えない」

従来の一般的なWebアーキテクチャ



ロードバランサの提供する機能

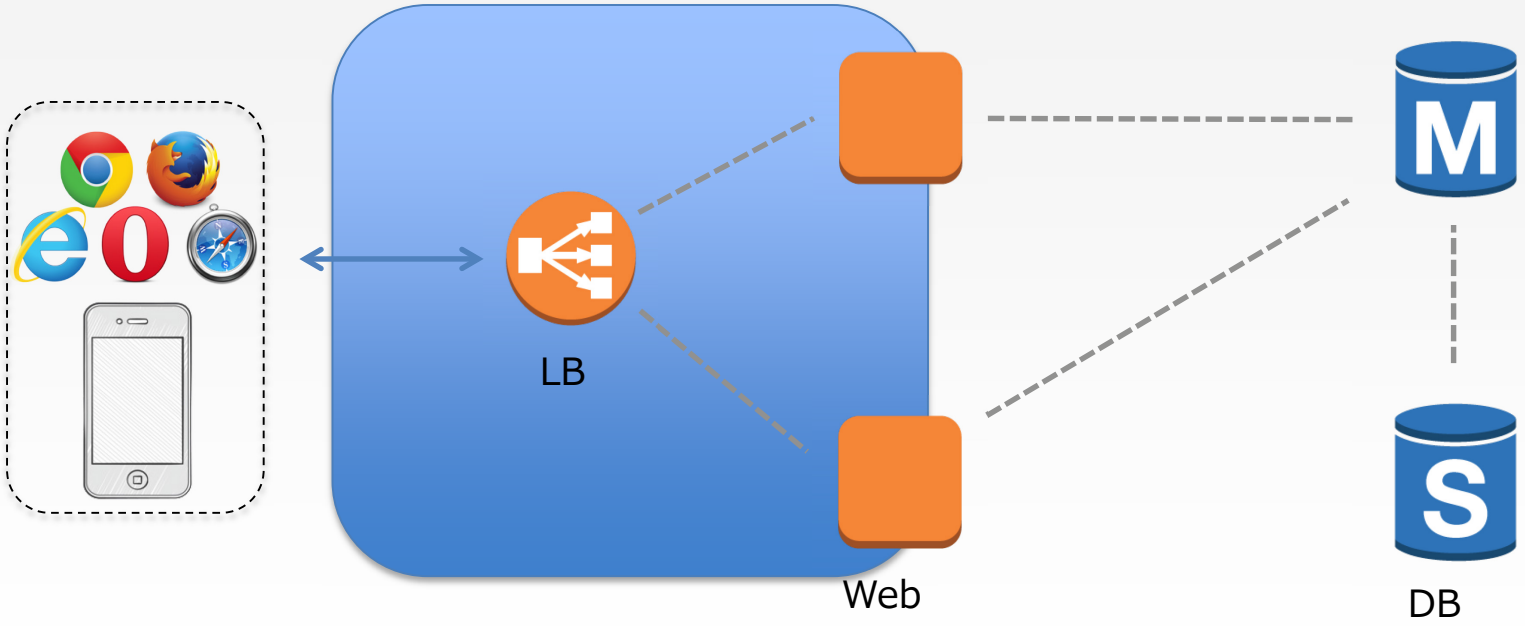
- 障害サーバの切り離し機能
- Session stickness
- ロードバランシングのポリシー管理
- バックエンドサーバの保護
- Cache
- サーキットブレイカー

ロードバランサのクオリティを前提とするアプリとは？

- 1995から2004頃までのブラウザ
- 再送をしないアプリ
- 社内システム
- 「４０４」を見せたくないWebサービス

イマドキのWeb屋さんの考えるネットワーク

スマホ時代のWebアーキテクチャ



DNS RR＋スマホ（今時のブラウザ）でほとんどの機能は置き換え可能

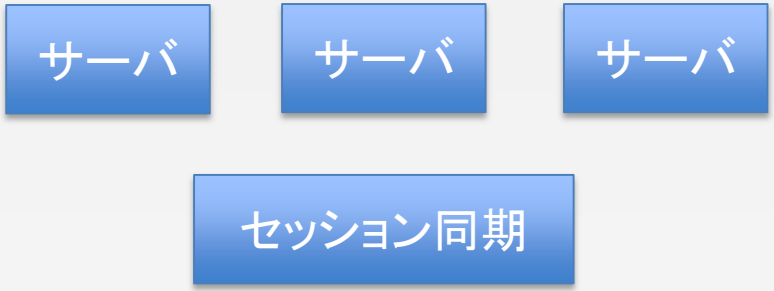
障害サーバの切り離し

- やること
  - ヘルスチェックを行い、その結果に応じてAレコードを更新することができる
- イマドキの方法
  - 最近のブラウザはfailoverの機能を持っている。Aレコードに含まれるIPアドレスを順にトライするため、単にサーバが返事をしなければいいだけのこと
  - IE6、Firefox1.5ではすでに対応

リトライ時の動きについて

1. クライアントのOSがAレコードを問い合わせると、DNSはAを複数返す
2. クライアントOSは１を複数保持する
3. クライアントからAレコードの一つに接続→ホストがport unreach等を返す
4. クライアントは次のAレコードに接続する

Session Stickness



- やりたいこと
  - ステートフルなアプリケーション動作
- イマドキの方法
  - バックエンドサーバは、セッション情報を共有する形のスケールアウトが当たり前に
  - そのためのミドルウェアも多数
  - パフォーマンスのために可能な限り処理は同じバックエンドで行うが、そのFailoverも可能になっている

ロードバランスのポリシー管理

- やりたいこと
  - 例えば100台あるバックエンドサーバの接続クライアント数を精緻にコントロールする
- イマドキの方法
  - バックエンドサーバがクライアントに応答しなければ、クライアントは次のサーバをトライするだけ

バックエンドサーバの保護

- やりたいこと
  - クライアントからのアクセスを制限して、サーバのダウンを防ぎ、Sorry Pageへ誘導する
- イマドキの方法
  - バックエンドサーバそれ自体が処理できる数だけを受け取る
  - その数を超えたらスケールアウトする設計へ

今時のプロトコル向けにはロードバランサ不要

- WebSocket
  - ポートを専有するためLBがはいらない
- HTTP/2
  - 完全ステートフルなのでLBがはいらない
- TCPレベルでのロードバランスに意味はない

HTTP/2の歴史振り返り

- 2012/01 次世代HTTPの話はじまる
- 2012/11 SPDYが開始点となる
- 2013/01 草案
- 2013/08 実装向け草案

HTTP/2で変わるもの（2015.5 RFC7540）

- ヘッダの無駄が激減（バイナリ化、差分利用）
- TCPコネクションが激減（1つで済む）
- 事実上TLS1.2が必須に
- HTTP Alternativeによる別ホスト別ポート誘導
- サーバプッシュ
  - サーバ側でクライアントのリクエスト無しにデータ送付が可能。例：HTML内のsrcにあるJSや画像のpush

HTTP/2のネゴシエーション

主要ブラウザベンダサポートはこちらが中心

- https の場合は TLS の拡張
  - NPN（Next Protocol Negotiation） もしくは
  - ALPN（Application-Layer Protocol Negotiation）
- http の場合は HTTP Upgrade
  - クライアントが いったん HTTP/1.1 で Upgrade ヘッダ付きリクエストをサーバーに送出
  - Upgrade のトークンによって HTTP/2 にスイッチする

「クラウドネイティブ」というキーワード

- クラウドで提供されるサービス利用を前提に構築するシステム及びアプリケーション
- サーバレスアーキテクチャへの移行
  - サーバ台数の増加は運用コストの増加要因であるためサーバレスへ
  - アプリケーションの書き換えコストを払ってでも移行を検討

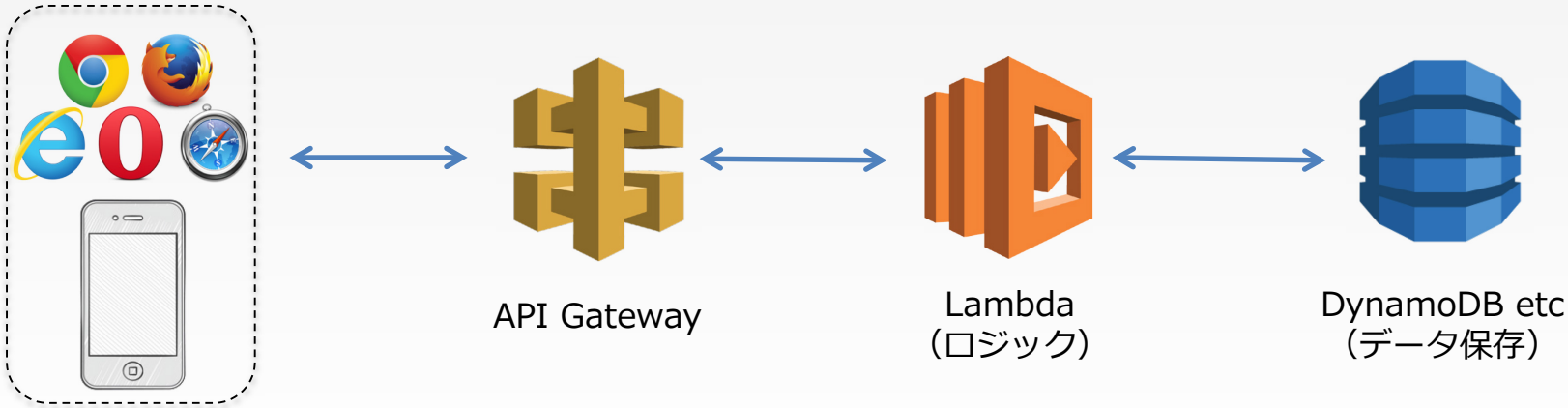
アプリケーションのクラウドネイティブ化

- 開発コストを最小化→バックエンド側のコード減
- 運用コストを最小化→バックエンド側のサーバ減
- スケーラビリティやキャパシティ、セキュリティの心配→マネージドサービスに期待する点
- **ビジネスにフォーカスできる**

今風のアーキテクチャ

API Gateway + Lambdaが実現する サーバレスアーキテクチャ

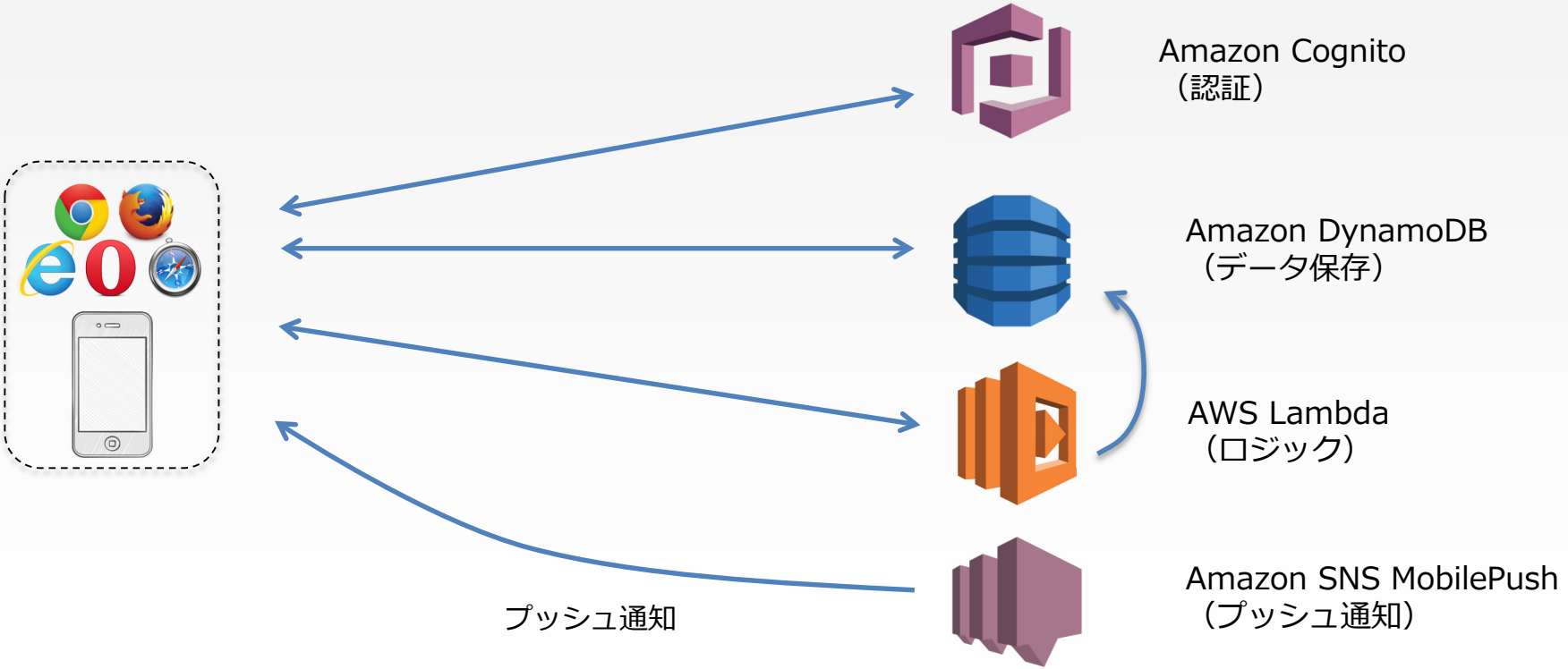
クラウドサービスを活用することでサーバを利用せずに構成するアーキテクチャ



API GW: REST APIの受け口
Lambda: イベントドリブンのプログラム実行環境

サーバレスアーキテクチャ（モバイル）

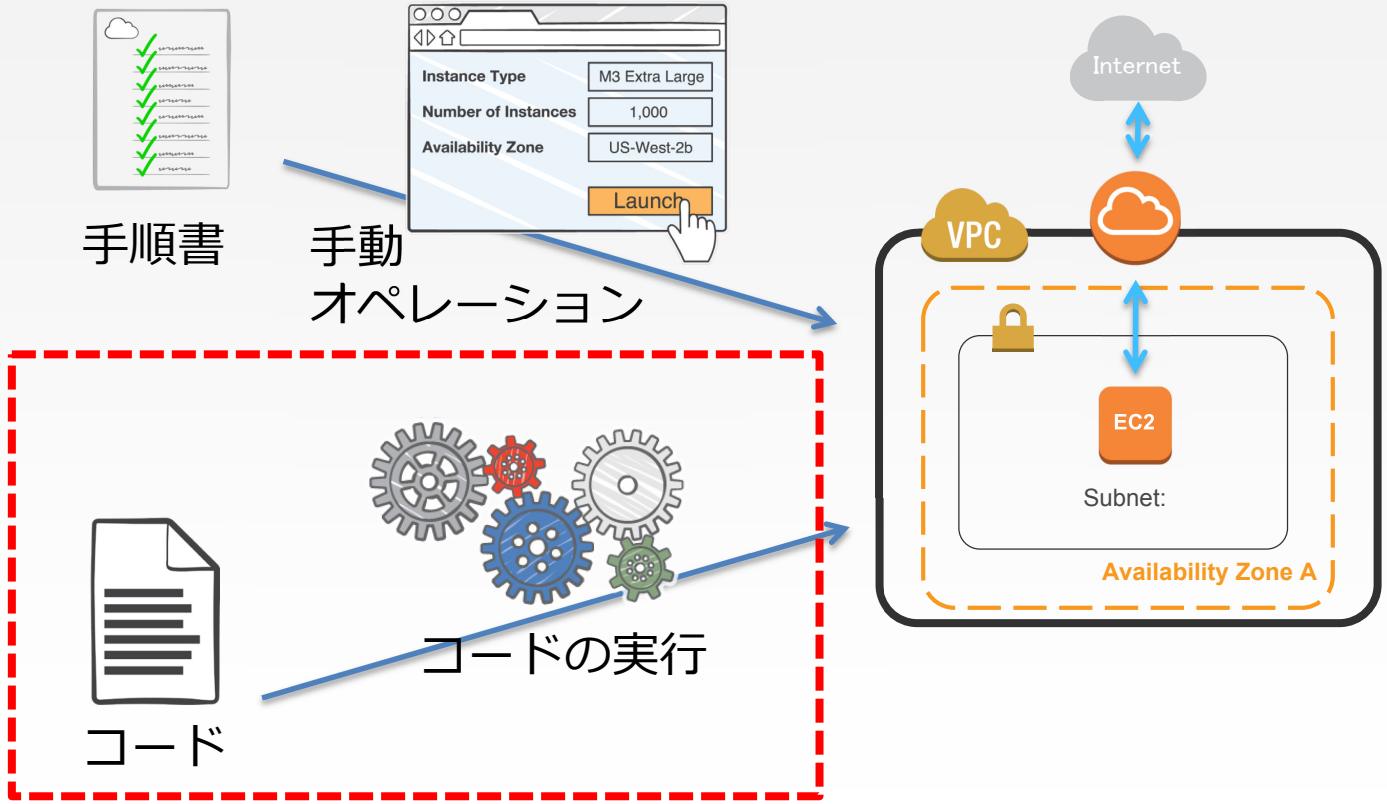
Mobile SDKを活用して、バックエンドとしてAWSサービスを直接利用することも可能



インフラ・ネットワークのコード化
(Infrastrucute/Network as Code)

ユーザが欲しいのはサービスであり、
テクノロジーそのものではない（どうでもいい）

AWSでのコード化について



インフラの構成をコードで管理

AWSにおけるインフラをコード化するツール

CloudFormation

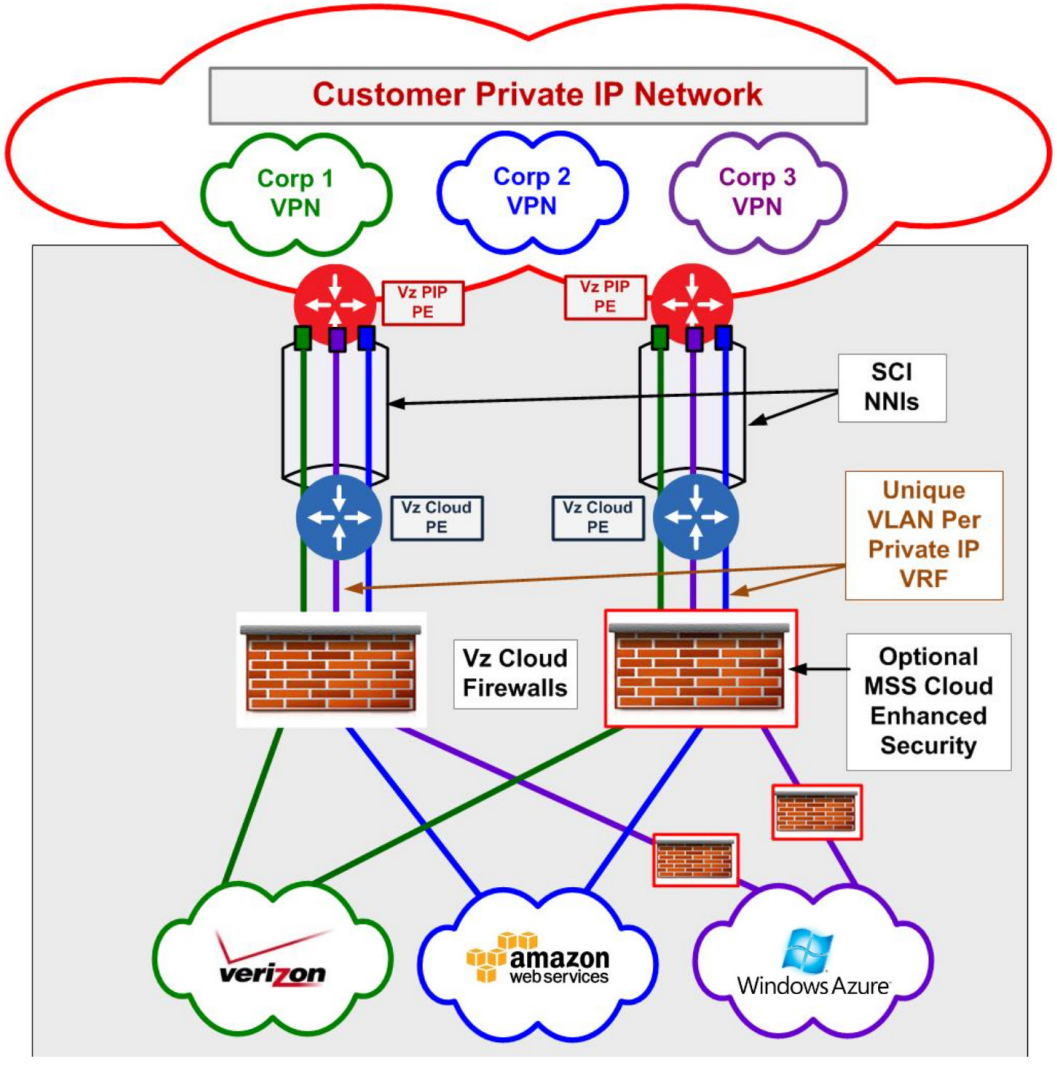
JSONテンプレートを元にAWSリソースの構築を自動化。
レイヤーを分けて定義も可（ex. ネットワークレイヤー）
複製やアップデート、バージョン管理が容易

AWS CLI
AWS SDK
Powershell

AWS CLI/PowershellはAWSの操作をコマンドラインで行なうツール

AWS SDKはいろいろなプログラミング言語に調整されたAPIを利用してAWSの操作を行なう

ネットワークのサービス化によるデリバリの高速化（Verizon Secure Cloud InterConnectの例）



1. 事前のNNI接続
2. ネットワーク機器の仮想化、マルチテナント化
3. セキュリティ等の付加オプション

まとめ

共通認識と業界トレンドへの対応

2017年には従業員の半数以上が自分のデバイスを業務に使うようになる（Source：ガートナー）

モバイルへの対応急務

40%のビジネスと技術の方向付けをする専門家が既に複数のクラウドを使うか、使う予定（Source：Forrester

アプリのクラウド選択

2020年までに500億のデバイスがインターネットに繋がる（Source：AT&T）

クラウドネイティブによるスケール対応

APTやDDoSに晒される機会は2年前の10倍に（Source： AT＆T）

攻撃への対応もスケール要

イマドキのWeb屋さんが考えるネットワーク

- リッチクライアントが前提に
  - ロードバランシング技術やネットワークの品質担保は端末にかんりオフロードできる
- ネットワークは土管（の安さ）
- スケールを支える安全なプラットフォームであってほしい

ネットワークエンジニアはどうする？

- モバイルとクラウドの技術はスタンダードに

- クラウドサービス利用技術を磨く？
  - 上位レイヤの要求に迅速かつ的確に応える？
- サービス提供側の技術にこだわる？

次の道

- 同レイヤで新しいもの→SDN, NFV
- 同レイヤでマネージドサービスをうまくつかえるようになる→クラウドに精通
- 上位レイヤもできるようになる
- 分散コンピューティングの知見を持つ

インシデント対応、DDoS対応

- ネットワークエンジニア
- →インシデント対応できるエンジニアになる。
- →CSIRTの人になる？

- アプリケーションエンジニア
- →攻撃時に縮退させる設計ができるようになる