AWS を使ったモバイルアプリの設計と実装

#AWSSummit Tokyo 2016

Akihiro Tsukada
Solutions Architect, Amazon Web Services Japan K.K.
2016/06/03

amazon web services

Gold Sponsor

Silver Sponsor

Bronze Sponsor

DemoPit for Business

DemoPit for Developers

TwitterでAWS Summitに参加しよう!

#AWSSummit

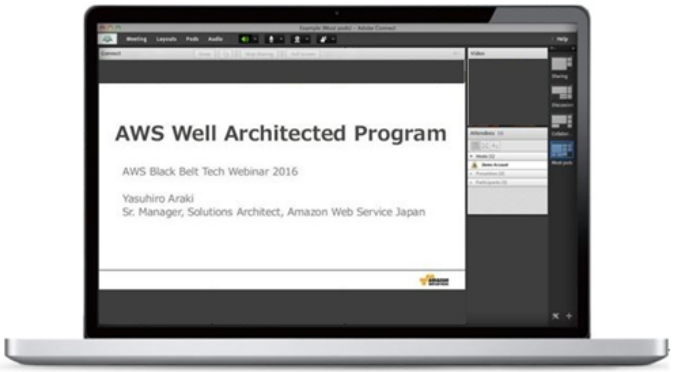
公式アカウント**@awscloud_jp**
をフォローしたお客様に
フリクションボールペンをプレゼント！

【配布場所】ロビーや展示会場のコンパニオンが配布中！お気軽にお声かけください。

AWS Black Belt Online Seminarのご案内

毎週開催

AWSJ の Tech メンバーがAWSに関する様々な事を日本語で紹介・解説する**無料**のオンラインセミナー



AWSについてもっと勉強したい方にオススメ！

AWS　イベント　検索

AWS Black Belt Online Seminar

塚田 朗弘@akitsukada

- AWS Solutions Architect
  - モバイルニンジャ**3**号機
  - スタートアップ
- Ruby, iOS, Android, OOP
- 妻と娘x2 が好きなおじさん

Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

Agenda

1. **このセッションについて**
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

このセッションについて

- **対象者①**
  - モバイルネイティブアプリのエンジニアの方
- **対象者②**
  - モバイルアプリのバックエンドシステムを触っているサーバサイドエンジニアの方
- **ゴール**
  - AWSでモバイルアプリ（含バックエンド）を設計・実装するパターンを知っていただくこと
  - AWSでモバイルアプリの設計・構築・運用コストを減らせることを理解していただくこと

Agenda

1. このセッションについて
2. **AWSのモバイルサービス**
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

AWSのモバイルサービス

https://aws.amazon.com/jp/mobile/

あなたのモバイルアプリ


ゲーム


ユーティリティ


ファイナンス


ソーシャル


エンターテインメント

AWS Mobile SDK


AWS SDK for Android


AWS SDK for iOS


AWS SDK for Unity


AWS SDK for JavaScript

モバイルに最適化されたサービス


Amazon Cognito (Cognito)


Amazon Mobile Analytics (Mobile Analytics)


Amazon SNS Mobile Push (SNS)


AWS Device Farm (Device Farm)


Amazon API Gateway (API GW)

モバイルに最適化されたコネクタ


Amazon DynamoDB (DynamoDB)


Amazon CloudFront (CloudFront)


Amazon Simple Storage Service (S3)

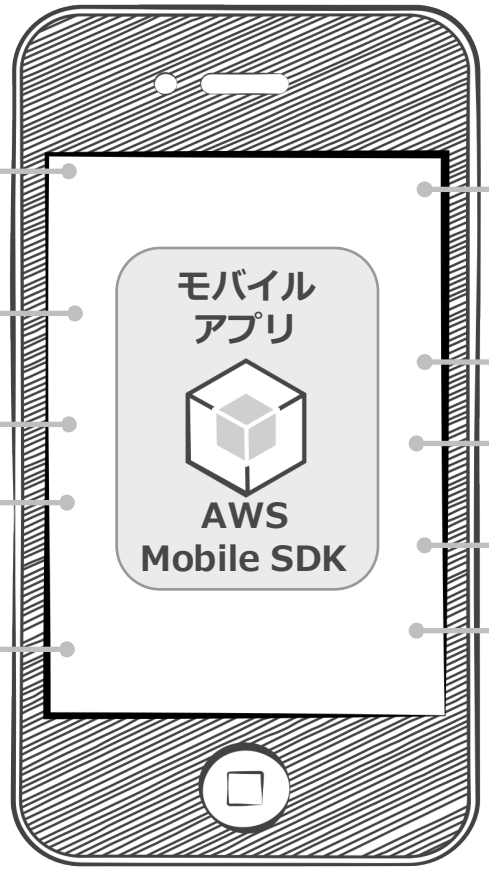

Amazon SQS (SQS)


Amazon SES (SES)


AWS Lambda (Lambda)

AWSのモバイルサービス

https://aws.amazon.com/jp/mobile/



ユーザ認証、アクセス認可
Amazon Cognito
(Identity, Userpools)

データの同期
Amazon Cognito
(Sync)

ユーザ行動分析
Amazon Mobile
Analytics

ビジネスロジックの実行
AWS Lambda

RESTful APIサーバ
Amazon API Gateway

メディアの管理
Amazon S3
Transfer Manager

メディアの配信
Amazon CloudFront
(Device Detection)

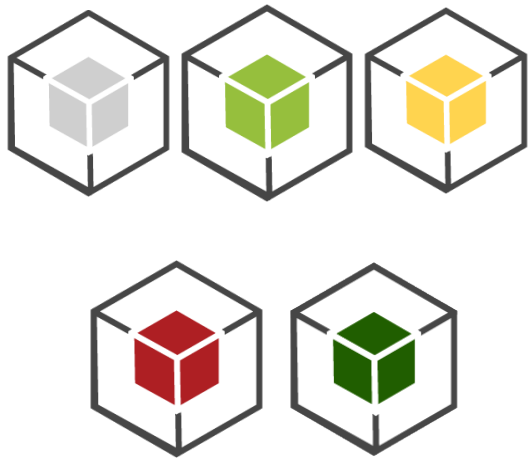
プッシュ通知の送信
Amazon SNS
Mobile Push

共有データの保存
Amazon DynamoDB

実機テストの並列実行
AWS DeviceFarm

AWS Mobile SDK

- 全てのサービスに共通の認証機構
- オンライン・オフラインを自動で
  ハンドリング
- クロスプラットフォームのサポート：
  Android, iOS, Fire OS, Unity, Xamarin
- Mobile OS への最適化
  例： ローカルオフラインキャッシュを
  利用するアーキテクチャ
- メモリフットプリントの削減
- 各プラットフォームのエンハンスに追従



Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. **モバイルアプリアーキテクチャ分類**
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

モバイルアプリアーキテクチャ分類

アーキテクチャ	バックエンドアクセス	利用するSDK	主な登場サービス群
General Web	JSON on HTTP(S)	一般的なネットワークアクセスライブラリ	Elastic Load Balancing(ELB), Amazon Elastic Compute Cloud(EC2), Amazon Relational Database Service(RDS)
Serverless	API GW Call	API GW SDK	API GW, Lambda, DynamoDB, …
2-Tier※	AWS API Call	AWS SDK	Lambda, DynamoDB, …

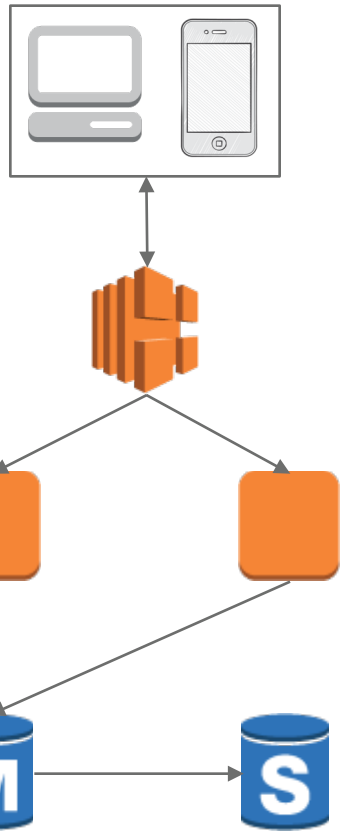
※2-Tier Architecture は Serverless の一形態ですが、ここでは分けて話をします。

（横断的）	AWS API Call	AWS SDK	Cognito, S3, SNS, Device Farm, Mobile Analytics, Amazon Kinesis, …

モバイルアプリアーキテクチャ分類

アーキテクチャ	バックエンドアクセス	利用するSDK	主な登場サービス群
General Web ①	JSON on HTTP(S)	一般的なネットワークアクセスライブラリ	Elastic Load Balancing(ELB), Amazon Elastic Compute Cloud(EC2), Amazon Relational Database Service(RDS)
Serverless ②			API GW, Lambda, DynamoDB, …
	API GW Call	API GW SDK	
2-Tier ③	AWS API Call	AWS SDK	Lambda, DynamoDB, …
（横断的）	AWS API Call	AWS SDK	Cognito, S3, SNS, Device Farm, Mobile Analytics, Amazon Kinesis, …

① General Web Architecture



Elastic Load Balancing
(ELB)

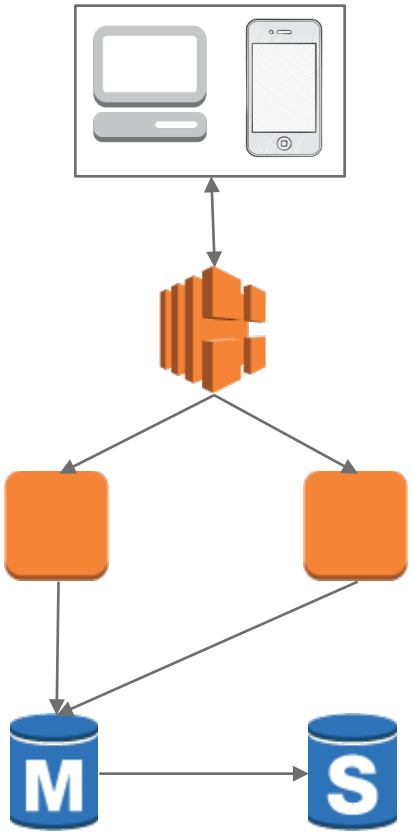
Amazon
Elastic Compute Cloud
(EC2)

Amazon
Relational Database Service
(RDS)

クライアントは
HTTP(S)でWebサーバと
通信

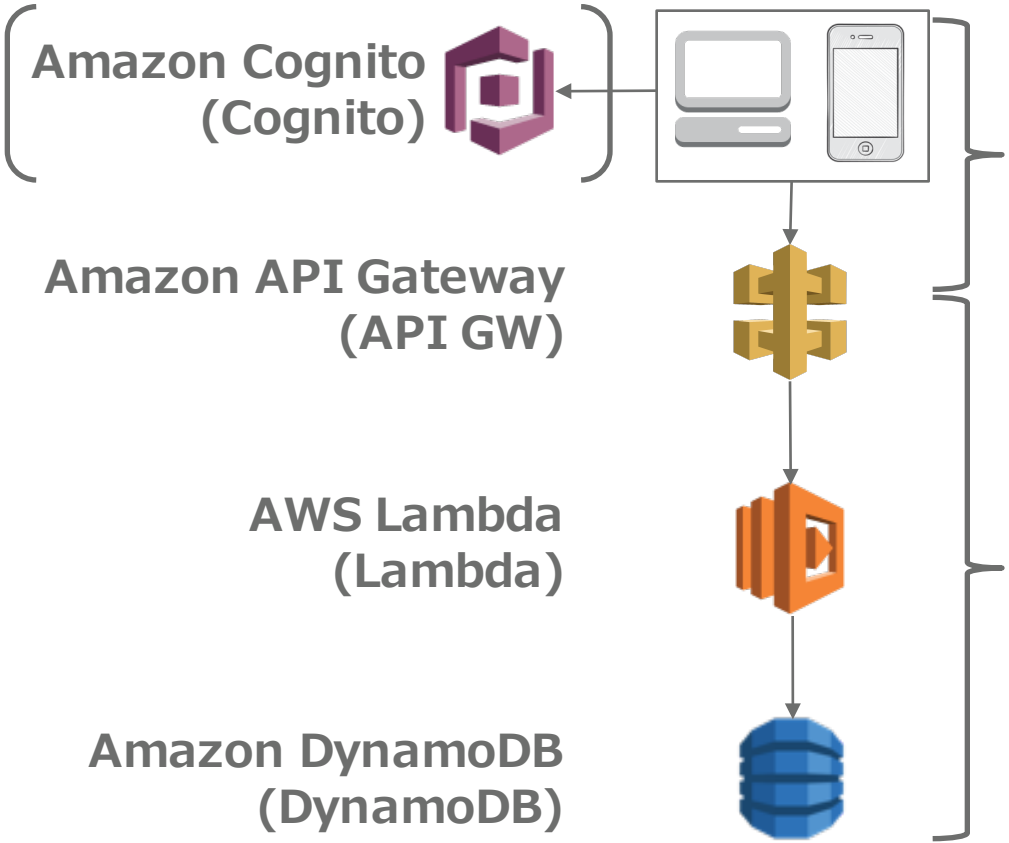
サーバサイドは
ELB + EC2 + RDS
といった**基本構成**

① General Web Architecture



- **メリット**
  - クライアント側は従来のノウハウをフルに活かせる
  - 実績が多く枯れた構成である
  - カスタマイズ性が高い
- **デメリット**
  - サーバのスペック、台数などインフラを意識して設計する必要がある
  - サーバの運用は利用者に任されている

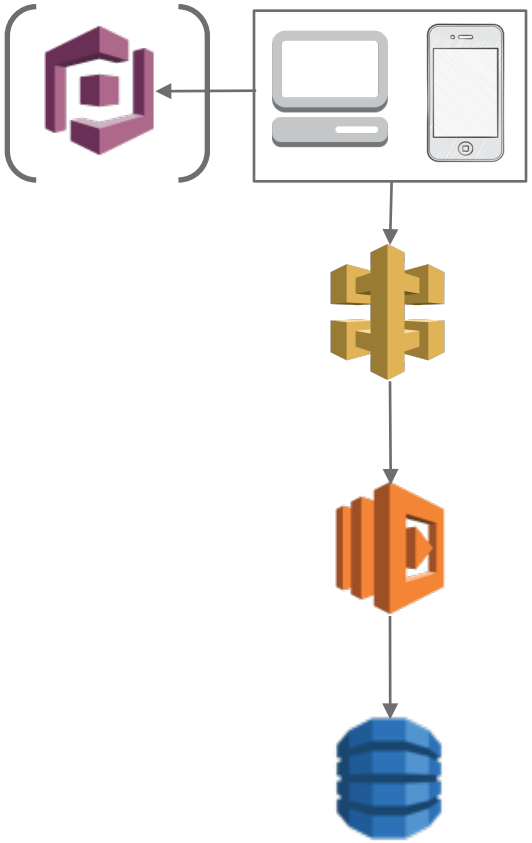
② Serverless Architecture



クライアントアプリは必要に応じて
Cognitoから**一時的なCredentials**を
得た後、**JSONでWeb APIと通信**

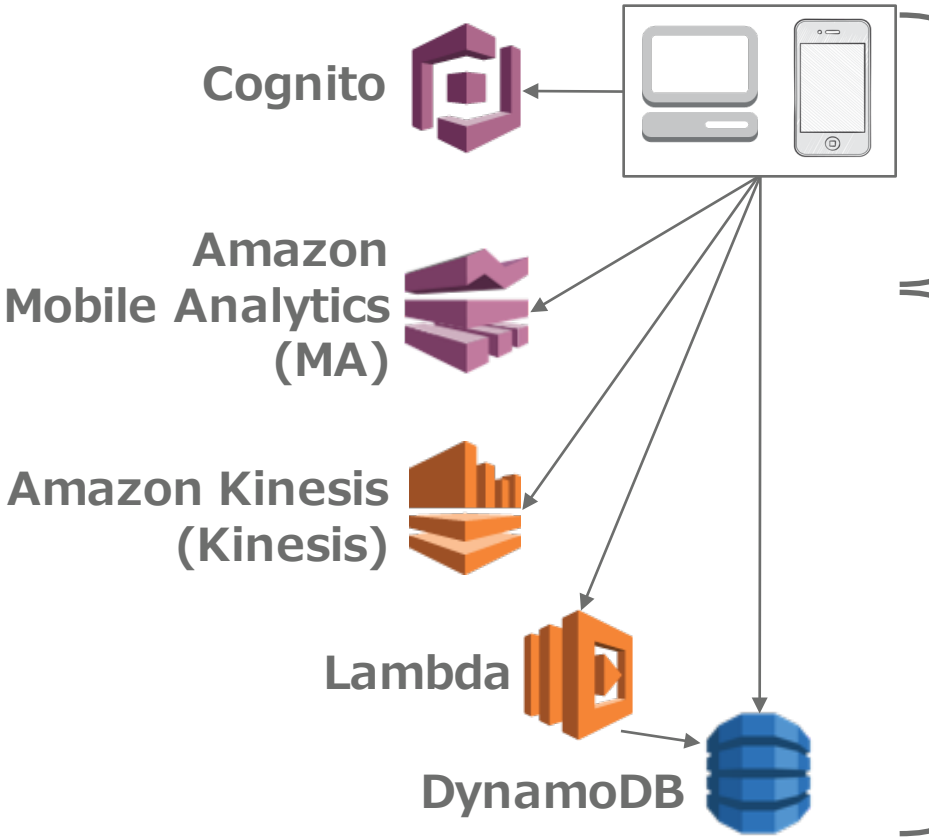
サーバサイドは
API GW/Lambda/DynamoDB
といった**マネージドサービス**を用い、
EC2やELBを利用しない

② Serverless Architecture



- **メリット**
  - クライアント側の実装は従来のノウハウを活かせる
  - サーバの運用、スケールはAWSに一任できる
  - CognitoによるセキュアなAPIアクセス制御が可能
  - 多くの場合コスト効率が高い
- **デメリット**
  - 新規性が高く、まだ枯れていない
  - 個々のサービスのマネージドな部分はカスタマイズしにくい

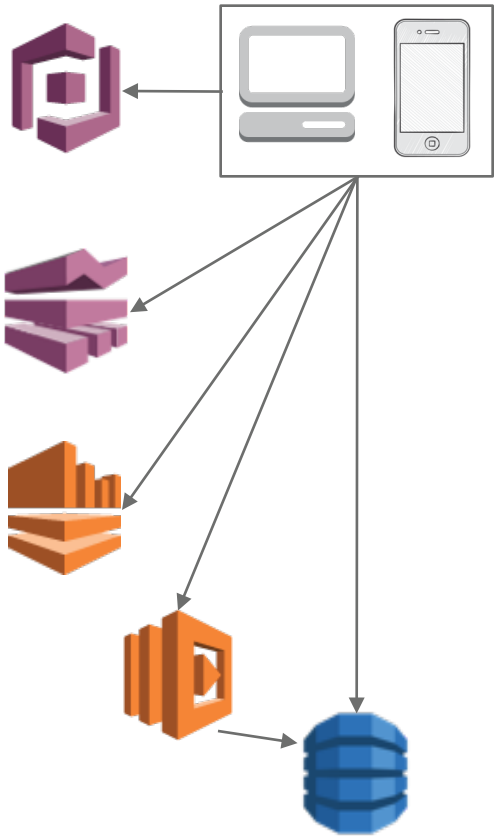
③ 2-Tier Architecture ※Serverlessの一形態



クライアントアプリはCognitoから
Temporary Credentialsを得た後、
AWS SDKを通じて各AWSリソースのAPIを直接叩く

サーバサイドは各AWSリソースを
セッティングしておくのみ
(左図は一部の例)

③ 2-Tier Architecture ※Serverlessの一形態

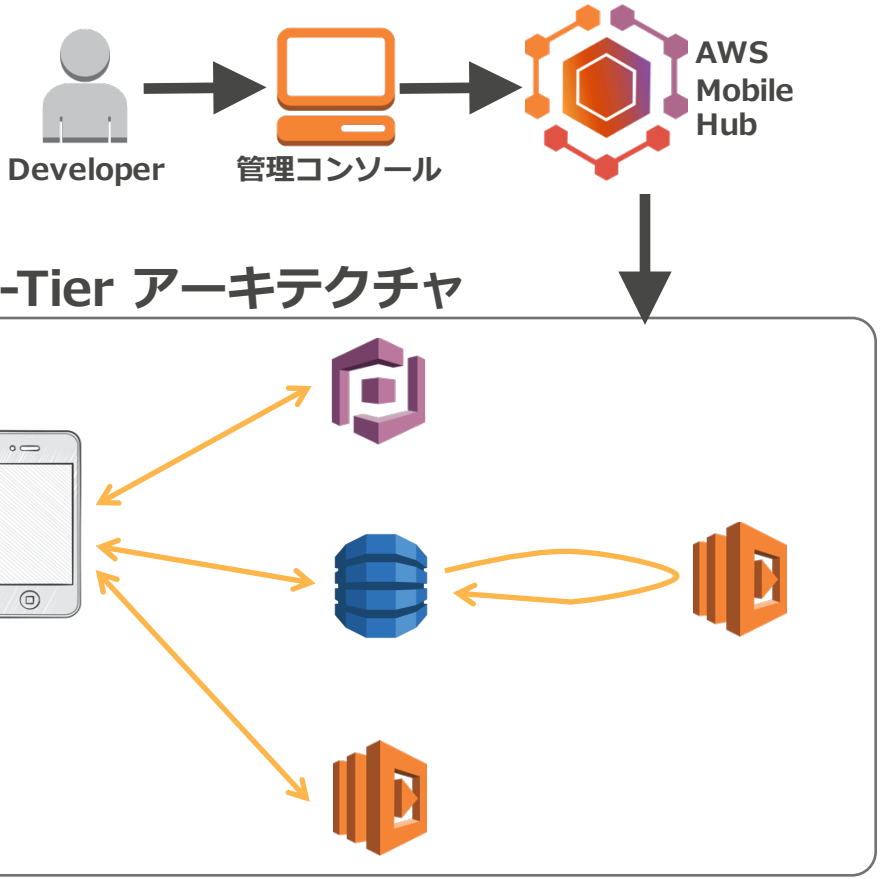


- **メリット**
  - インフラの運用、スケールはAWSに一任できる
    - 上限値の緩和申請は除く
  - Cognitoによるセキュアなアクセス制御が可能
  - Web APIの設計が不要で手軽に使える
  - 最小限のパーツを組み合わせて使うことができる
  - 多くの場合コスト効率が高い
- **デメリット**
  - 新規性が高く、まだ枯れていない
  - クライアントサイドが各AWSリソースに依存する
  - 個々のサービスのマネージドな部分はカスタマイズしにくい

③ 2-Tier Architecture ※Serverlessの一形態

AWS Mobile Hub

所要時間数分で
2-Tierアーキテクチャを
プロビジョニングし、
スターターアプリの
自動生成までするツール



どのアーキテクチャを採用すべきか?

どのアーキテクチャを採用すべきか?

三者択一でなく

メリット/デメリットを

見て**組み合わせを**

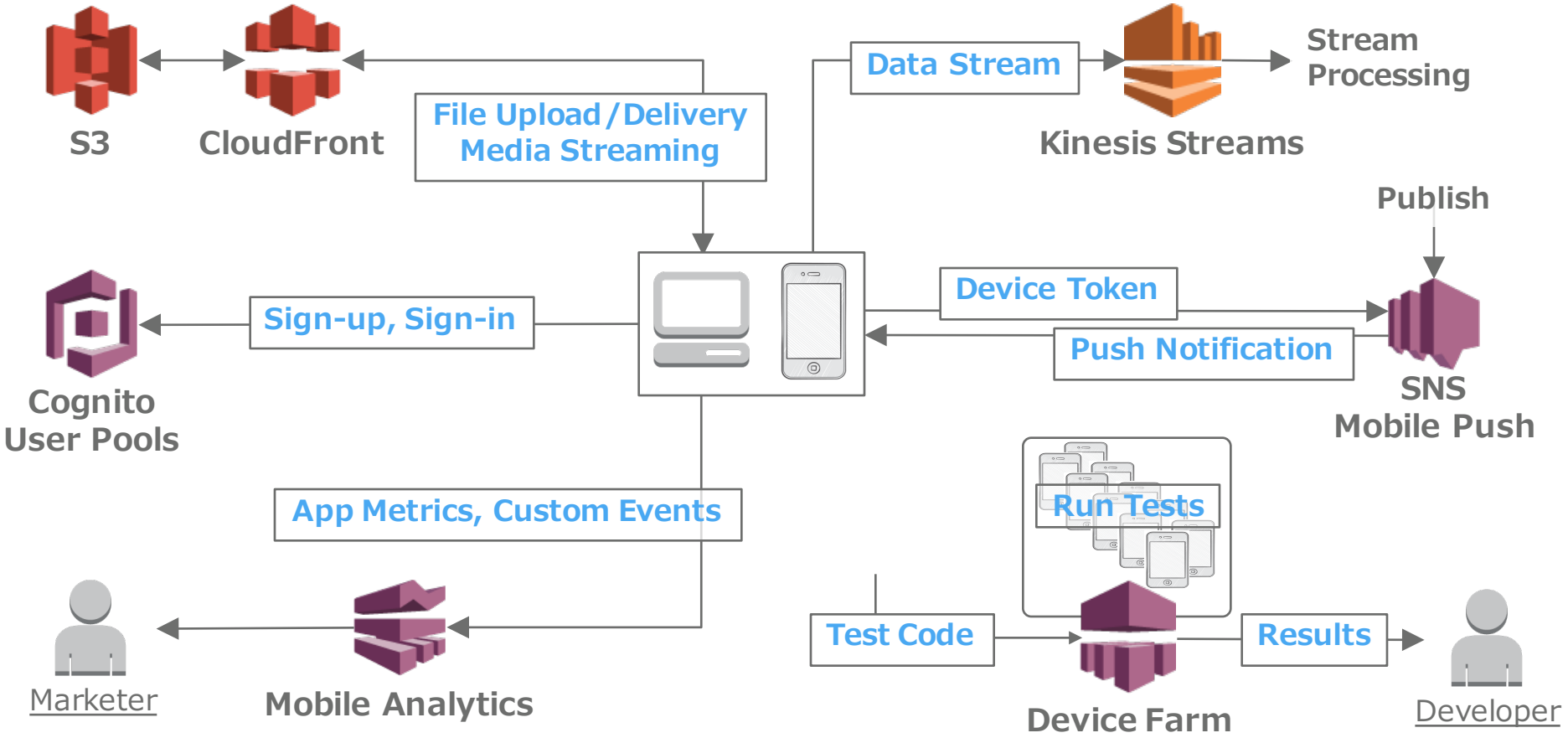
Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. **アーキテクチャ分類を横断するモバイルサービス**
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

モバイルアプリアーキテクチャ分類

アーキテクチャ	バックエンドアクセス	利用するSDK	主な登場サービス群
General Web	JSON on HTTP(S)	一般的なネットワークアクセスライブラリ	Elastic Load Balancing(ELB), Amazon Elastic Compute Cloud(EC2), Amazon Relational Database Service(RDS)
Serverless			API GW, Lambda, DynamoDB, …
	API GW Call	API GW SDK	
2-Tier	AWS API Call	AWS SDK	Lambda, DynamoDB, …
（横断的）	**AWS API Call**	**AWS SDK**	**Cognito, S3, SNS, Device Farm, Mobile Analytics, Amazon Kinesis, …**

アーキテクチャ分類を横断するモバイルサービス



アーキテクチャ分類を横断するモバイルサービス

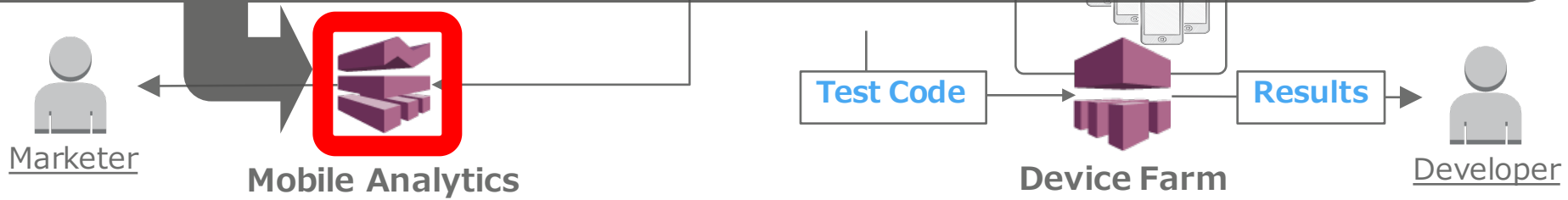
Amazon Cognito User Pools (Public Beta)

- マネージドなユーザ管理基盤
- サインアップ、サインイン、パスワード紛失、MFAログインなど柔軟な設定が可能
- Lambdaによるフック処理で拡張可能
  - Pre Sign-up, Custom Message, Pre Sign-in, Post Sign-in, Post Confirmation
- 50,000MAUまで無料

アーキテクチャ分類を横断するモバイルサービス

Amazon Mobile Analytics

- 新規ユーザ、リピートユーザ、ユーザー定着率、アクティブ率、リテンションなどの主要なメトリクスを自動取得
- カスタムイベント、マネタイズイベントを取得可能
- S3、Amazon Redshiftへオートエクスポート可能
- サンプリングを一切しない正確なデータ
- 毎月1億イベントまで無料、それ以上は$1/100万イベント

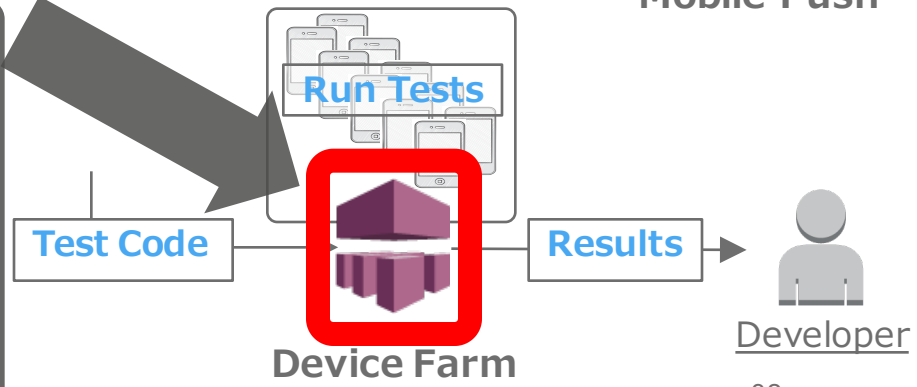


アーキテクチャ分類を横断するモバイルサービス

AWS Device Farm

- 自前でモバイル実機を揃えなくてもOK
- 任意のモデルを選択して並列テスト実行可能
- カスタムのテストの他、Fuzzテストを今すぐ実行可能
- JenkinsとGradleのプラグインを提供

- Appium、Calabash、Espressoなどのテスティングフレームワークをサポート
- 250デバイス分(device minutes)の無料枠



Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. **シンプルなアプリ側の実装**
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. まとめ

以下の操作をする際の実装例（iOS）

- Cognito IdentityからCredentialsを取得
- DeviceToken取得 → SNSに登録
- Cognito User Poolsの Sign-up
- MobileAnalyticsの カスタムイベント送信
- Lambda Invoke

Cognito IdentityからCredentialsを取得

```
func application(application: UIApplication,
  didFinishLaunchingWithOptions launchOptions:
  [NSObject: AnyObject]?) -> Bool {

    let credentialsProvider = AWSCognitoCredentialsProvider(
        regionType: AWSRegionType.USEast1, identityPoolId: "IDENTITY_POOL_ID“
    )
    let configuration = AWSServiceConfiguration(
        region: AWSRegionType.USEast1, credentialsProvider: credentialsProvider
    )

    AWSServiceManager.defaultServiceManager().
       defaultServiceConfiguration = configuration

    credentialsProvider.credentials().continueWithBlock {(task: AWSTask) -> AnyObject? in
        let credentials = task.result as! AWSCredentials
        print(credentials.accessKey)
        return nil
    }

    return true
}
```

DeviceToken取得 → SNSに登録

```
func application(application: UIApplication,
  didRegisterForRemoteNotificationsWithDeviceToken deviceToken: NSData) {
    let deviceTokenString = ”¥(deviceToken)"
        .stringByTrimmingCharactersInSet(NSCharacterSet(charactersInString:"<>"))
        .stringByReplacingOccurrencesOfString(" ", withString: "")

    print("deviceTokenString: ¥(deviceTokenString)")

    let sns = AWSSNS.defaultSNS()
    let request = AWSSNSCreatePlatformEndpointInput()
    request.token = deviceTokenString
    request.platformApplicationArn = "SNSPlatformApplicationArn"
    sns.createPlatformEndpoint(request).
       continueWithExecutor(AWSExecutor.mainThreadExecutor(),
                            withBlock: { (task: AWSTask!) -> AnyObject! in
            let createEndpointResponse = task.result as! AWSSNSCreateEndpointResponse
            print("endpointArn: ¥(createEndpointResponse.endpointArn)")
            return nil

        }
    )
}
```

Cognito User PoolsのSign-up

```
AWSCognitoIdentityUserAttributeType * phone = [AWSCognitoIdentityUserAttributeType new];
phone.name = @"phone_number"; phone.value = @”0123456789";
AWSCognitoIdentityUserAttributeType * email = [AWSCognitoIdentityUserAttributeType new];
email.name = @"email"; email.value = @”cognito@example.com";

[[self.pool signUp:self.username.text password:self.password.text
    userAttributes:@[phone, email] validationData:nil] continueWithBlock:^id
    _Nullable(AWSTask<AWSCognitoIdentityUserPoolSignUpResponse *> * _Nonnull task) {
        NSLog(@"Successful signUp user: %@",task.result.user.username);
        dispatch_async(dispatch_get_main_queue(), ^{
            if(task.error){
                // error
            }else if(task.result.user.confirmedStatus !=
                        AWSCognitoIdentityUserStatusConfirmed){
                NSLog(@“Confirmation Sent to: %@",
                      task.result.codeDeliveryDetails.destination);
            }else{
                // SignUp Completed
            }});
        return nil;
    }];
}
```

MobileAnalyticsのカスタムイベント送信

```
func application(application: UIApplication, handleActionWithIdentifier identifier: String?,
forRemoteNotification userInfo: [NSObject : AnyObject], completionHandler: () -> Void) {

    let mobileAnalytics = AWSMobileAnalytics.defaultMobileAnalytics()
    let eventClient = mobileAnalytics.eventClient
    let pushNotificationEvent = eventClient.createEventWithEventType("PushNotificationEvent")

    var action = "Undefined"
    if identifier == "READ_IDENTIFIER" {
        action = "Read"
    } else if identifier == "DELETE_IDENTIFIER" {
        action = "Deleted"
    } else {
        action = "Undefined"
    }

    pushNotificationEvent.addAttribute(action, forKey: "Action")
    eventClient.recordEvent(pushNotificationEvent)
    mainViewController()?.displayUserAction(action)

    completionHandler()
}
```

LambdaのInvoke

```
let invocationRequest = AWSLambdaInvokerInvocationRequest()
invocationRequest.functionName = "awesomeFunction"
invocationRequest.invocationType =
                        AWSLambdaInvocationType.RequestResponse
invocationRequest.payload = [
    "param1": "value1",
    "param2": "value2"
]

let lambdaInvoker = AWSLambdaInvoker.defaultLambdaInvoker()
let task =  lambdaInvoker.invoke(invocationRequest).
            continueWithSuccessBlock() { (task) -> AWSTask! in
    print("response: ", task.result)
    return task
}
```

実装時の参考に

- AWS Mobile SDK https://aws.amazon.com/jp/mobile/sdk/?nc1=h_ls
- 開発者ガイド
  - iOS http://docs.aws.amazon.com/mobile/sdkforios/developerguide/
  - Android http://docs.aws.amazon.com/mobile/sdkforandroid/developerguide/
  - Unity http://docs.aws.amazon.com/mobile/sdkforunity/developerguide/
- サンプル
  - iOS https://github.com/awslabs/aws-sdk-ios-samples
  - Android https://github.com/awslabs/aws-sdk-android-samples
  - Unity https://github.com/awslabs/aws-sdk-unity-samples
- API リファレンス
  - iOS http://docs.aws.amazon.com/AWSiOSSDK/latest/
  - Android http://docs.aws.amazon.com/AWSAndroidSDK/latest/javadoc
  - Unity http://docs.aws.amazon.com/sdkfornet/v3/apidocs/Index.html

Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. **ケーススタディ**
   - **GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様**
7. まとめ

GunosyにおけるAWS Mobileの活用

少リソースでスケールするサービスに向けて

Gunosy Inc. 松本

2016/06/02

amazon web services

自己紹介

- 名前
  - 松本勇気
- 所属
  - 株式会社Gunosy 開発本部執行役員
- 担当範囲
  - iOS/Android/サーバサイド等幅広く
- TwitterID
  - @y_matsuwitter

現行プロジェクトの背景

KDDI様と共同で新規メディア開発へ

KDDI様のもつ「顧客接点」「多様なコンテンツ」

×

Gunosyにおける「配信ロジック」「運営ノウハウ」

今回の課題

非常に小さなPJチーム

- 意思決定を迅速に行う目的
- 開発リソースは少数精鋭で進める
- PJスタートからリリースまで規模に比して短期間

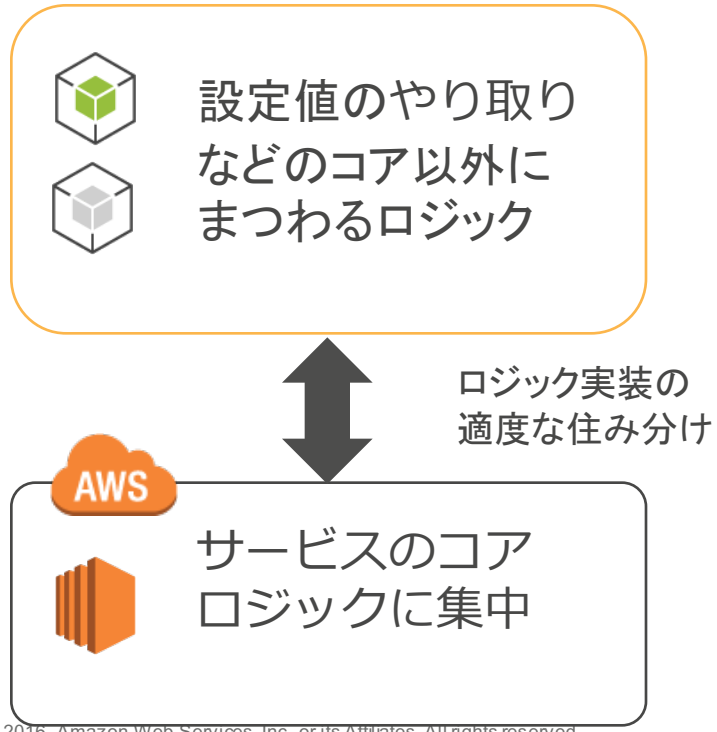
高い安定性を求められる

- 想定するユーザー規模の大きさ
- 依存システムは非常に多い
- それに対しての運用者の少なさ

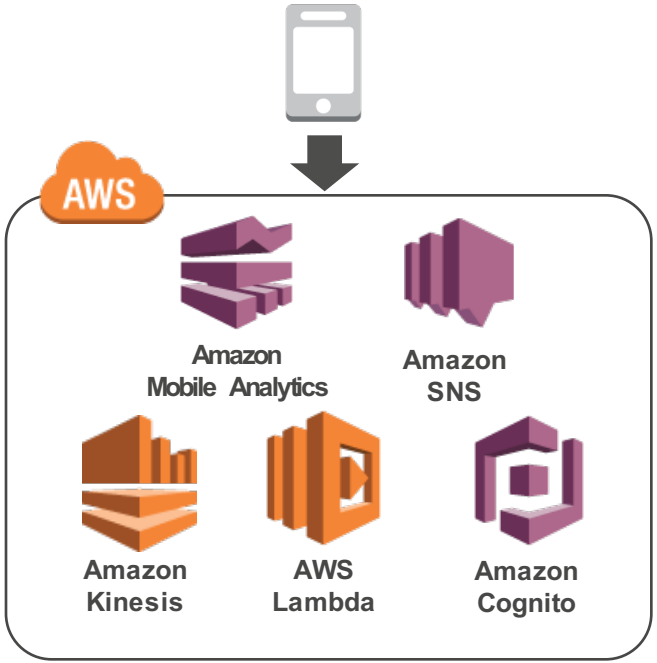
AWSを如何に活用したか

AWSをフル活用してのアーキテクチャ設計

クライアントへの処理委譲

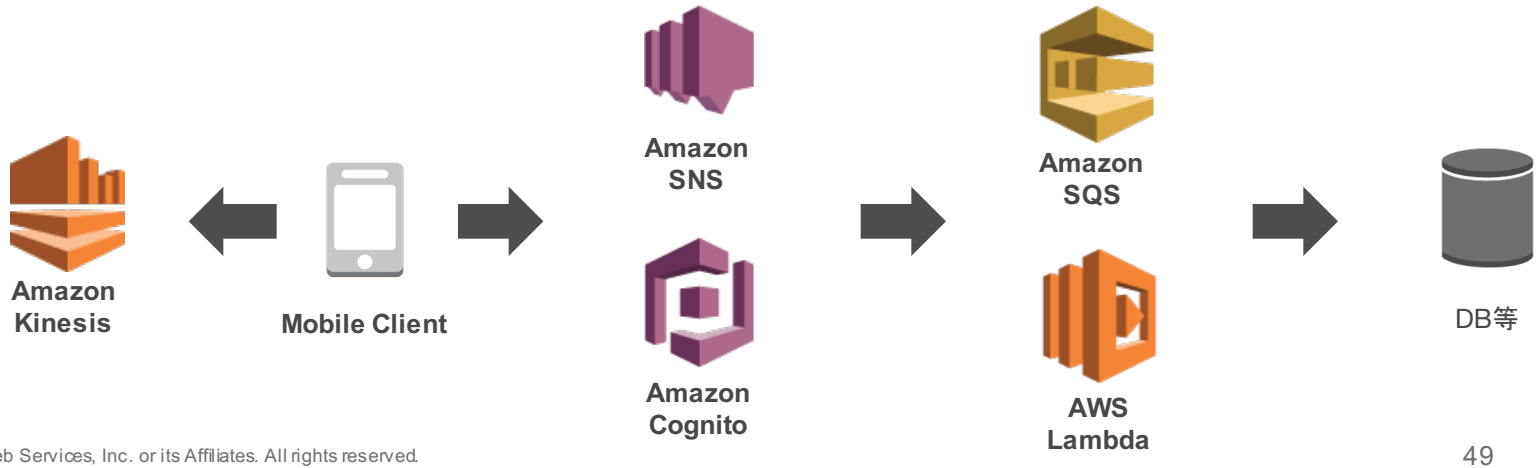


サーバレスアーキテクチャの部分的適用



クライアントに対する処理委譲

- AWS Mobile SDKを利用
- ユーザー認証・認可をCognito経由で
- 通知のトークン管理をSNSで
- ログの配送は直接Kinesisへ



サーバレスアーキテクチャの部分的適用

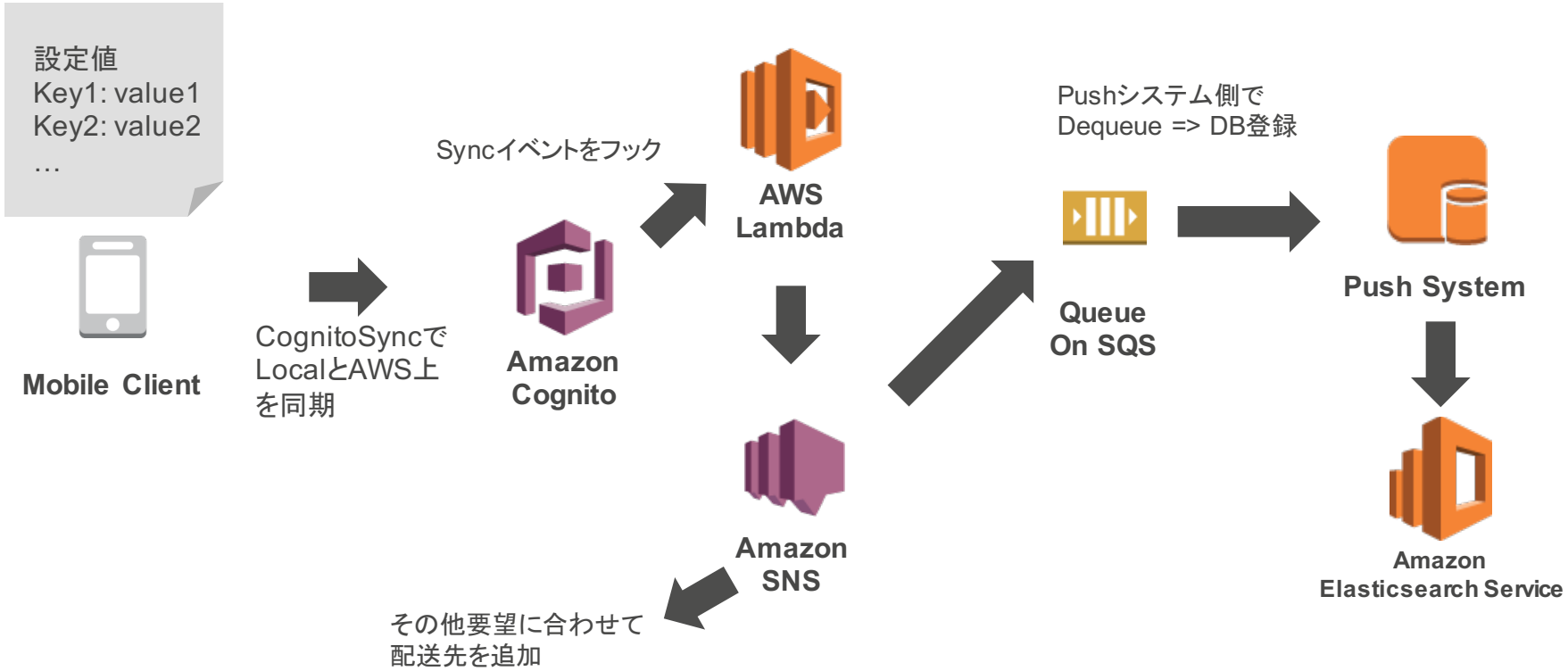
- すべてをサーバレスにはしない
    - 求めるレスポンス速度など非機能要件に応じて選定
- 非同期で運用可能なものほど向いている
    - 後述するユーザー設定管理
    - ログ配送・加工・集計

開発にあたってのtips

Cognito Syncによるサーバレスなユーザー管理

- Mobile側よりCognito　Syncにユーザー設定値を保持
- Cognito Syncのevent => Lambdaをフック
- LambdaからElasticsearch等へ設定値を保存
    - SNSのTopicにendpointをぶら下げるなども簡単
- 通知時刻などの管理をCognitoSync経由で完結
    - ストレージを除きサーバを持つ必要なし

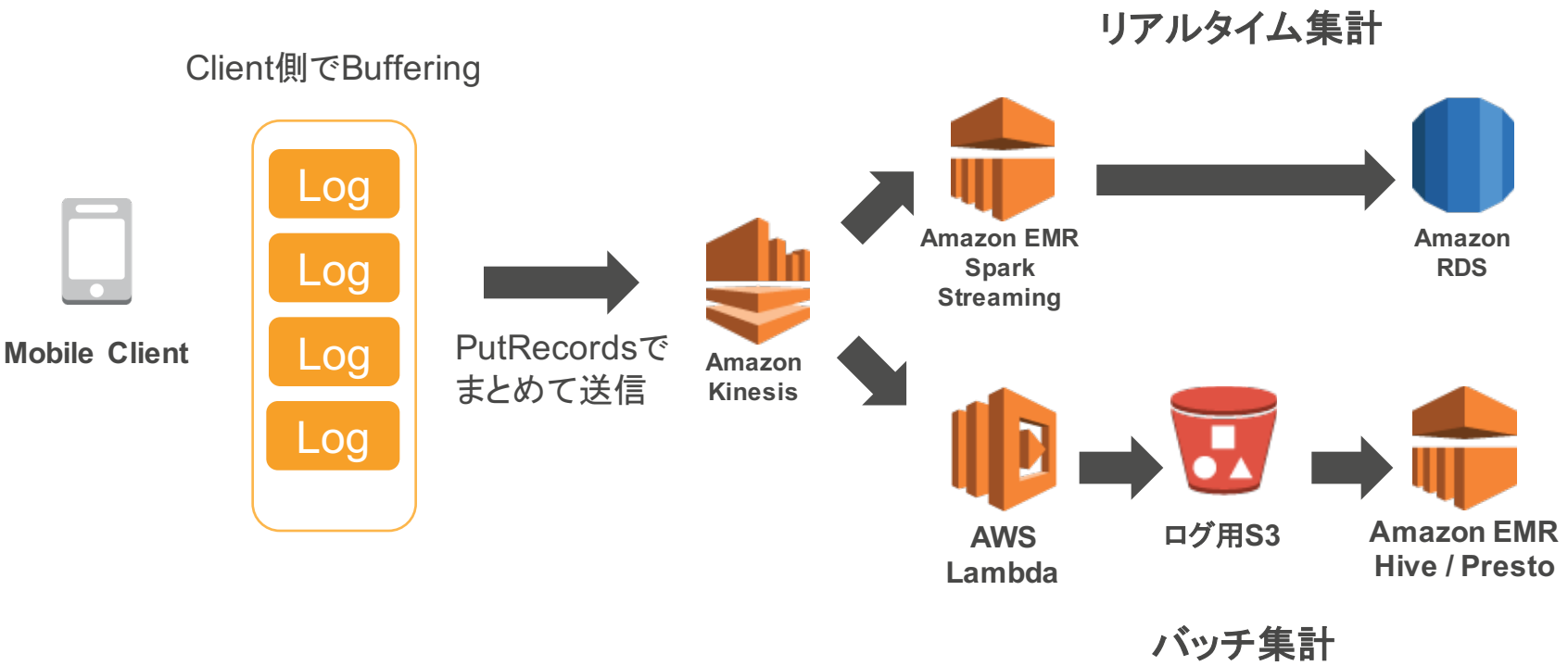
Cognito Syncによるサーバレスなユーザー管理



Kinesisによるログ管理フロー

- ログ中継などのリソースを持たない
  - 大規模なログ送信の管理コストは思った以上に高い
- クライアントからKinesisへ直接ログの配送
  - MobileSDKからの送信
  - 再送ポリシーなどはクライアント側で担保する必要あり

Kinesisによるログ管理フロー



最後に

- 新規プロジェクトに関しても幅広くメンバーを募集中
- AWSで一緒に最新事例を作っていきましょう！

Agenda

1. このセッションについて
2. AWSのモバイルサービス
3. モバイルアプリアーキテクチャ分類
4. アーキテクチャ分類を横断するモバイルサービス
5. シンプルなアプリ側の実装
6. ケーススタディ
   - GunosyにおけるAWS Mobileの活用 by Gunosy Inc. 松本様
7. **まとめ**

まとめ

1. モバイルアプリ開発時はAWSを柔軟に活用可能
2. 必要なものを選び組み合わせることで
   モバイルアプリをトータルにサポート
3. AWS SDKでクライアント側の実装もバッチリ
4. Serverlessアーキテクチャ、2-Tierアーキテクチャで
   モバイルアプリエンジニアにも使いやすく
   スケーラブルでコスト効率が高いバックエンドを！

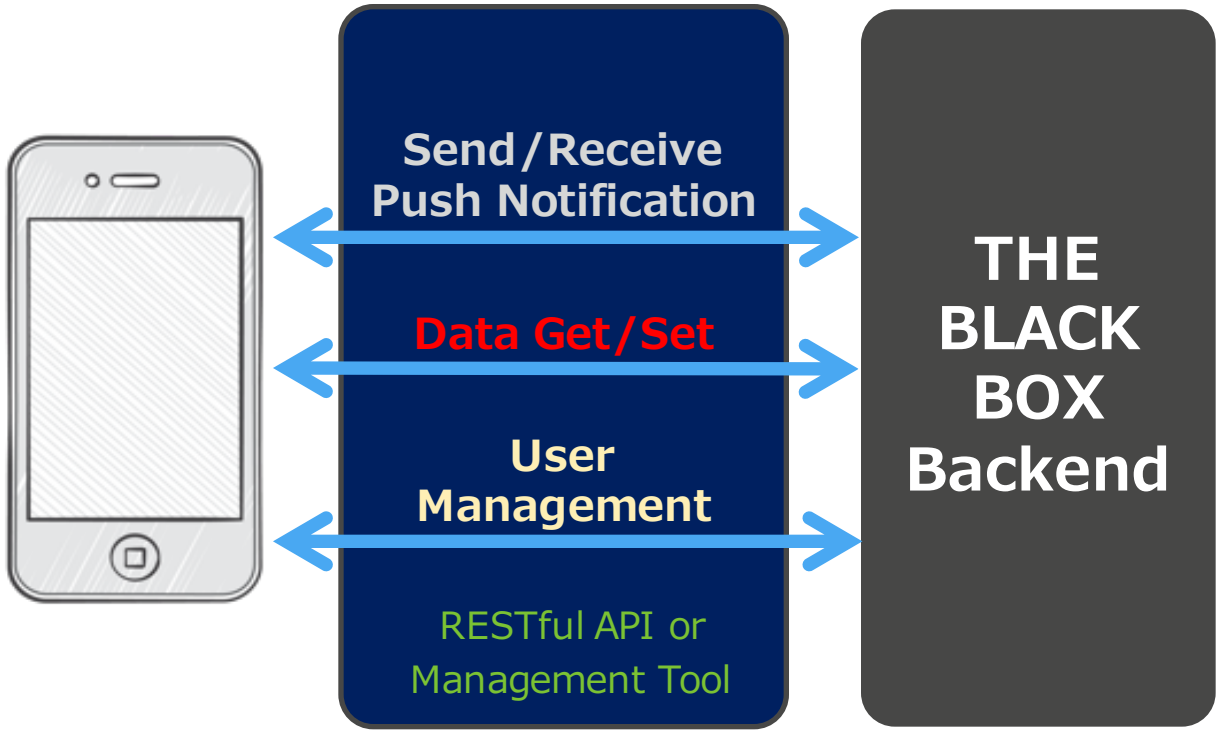
Thank you

Appendix

2-Tierアーキテクチャとの違い

mBaaS

- 基本的にバックエンドはブラックボックス
- 抽象度が高く扱いやすい ≒ カスタマイズに限界あり
- スタートダッシュには向くがサービス成長後はスケール不足になりがち



Parse.com から AWS への移行サポート

- Migrate From Parse Push to Amazon SNS
  https://mobile.awsblog.com/post/Tx3NE69QDHI7LJK/Migrating-from-Parse-Push-to-Amazon-SNS
  - Parse からのデータエクスポート/インポート
  - APNS/GCM の認証情報の取得
  - AWS リソースおよびAWS Mobile Hubプロジェクトの作成
  - Amazon SNS Mobile Push へのコード変更方法
- Set up Parse Server and MongoDB on AWS
  https://mobile.awsblog.com/post/Tx3LNEWJS45OYA9/Setting-up-Parse-Server-and-MongoDB-on-AWS-using-CloudFormation
  - Parse Server/MongoDBのインスタンスをAWS CloudFormationでデプロイ
- Parse Analytics to Mobile Analytics
  http://mobile.awsblog.com/post/Tx34JL4KCS9RBNH/Transition-Guide-Parse-Analytics-to-Amazon-Mobile-Analytics
  - Parse Analytics の代替としてのAmazon Mobile Analytics
- Parse IoT to AWS
  https://mobile.awsblog.com/post/Tx9EQ3MIJ85M4V/Welcome-Parse-IoT-Customers
  - Parse IoT を AWS に移行する方法
- Webinar – Migrate your apps from Parse to AWS
  https://www.youtube.com/watch?v=0Q0RBYoFIWU
  - ParseからAWSへの全般的な移行方法をWebinarで解説
- またはお近くのAWS Japan スタッフまで！

Thank you !!