



ゼロから始めるモバイルアプリ開発

AWS Mobile Hubハンズオン

【本編】

2016年01月16日 JAWS-UG沖縄

西島 幸一郎 @k_nishijima

ゼロから始めるモバイルアプリ開発 /
AWS Mobile Hubハンズオン
にご参加（表明）いただき
ありがとうございます！



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

こちらの資料は「本編」です。
「事前準備編」が完了していることを
前提としています。
終わってない方は、お近くの
サポートスタッフにお声がけください。



勿論分からぬいところや
詰まった所があれば、
お気軽にご質問ください！



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>

Twitterのハッシュタグ



#jawsug #jawsdays

みんな見てるので

反応が早いです！



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

本資料は

http://bit.ly/handson201601-main
よりダウンロード可能です。

GitHubリポジトリはこちら。

**https://github.com/jaws-ug/hands-on/
tree/master/Mobile-Hub**



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

あんた誰？

西島 幸一郎 / にしじま こういちろう

アルスリーインスティテュート ソリューションアーキテクト

<https://www.r3it.com>

JAWS-UG沖縄のコアメンバー

AWSサムライ2013/2014 2年連続拝命

当資料について、ご質問などあればFacebook/Twitterなどで
お気軽にお問い合わせください！



@k_nishijima



nishijima.koichiro



アールスリーインスティテュート

大阪の業務系システム開発会社

AWSとkintoneで「ハイスピードSI」

西島は沖縄から100%リモートワーク

【コミュニティにフルコミットする】と
宣言している珍しい会社



We are hiring!



JAWS-UG

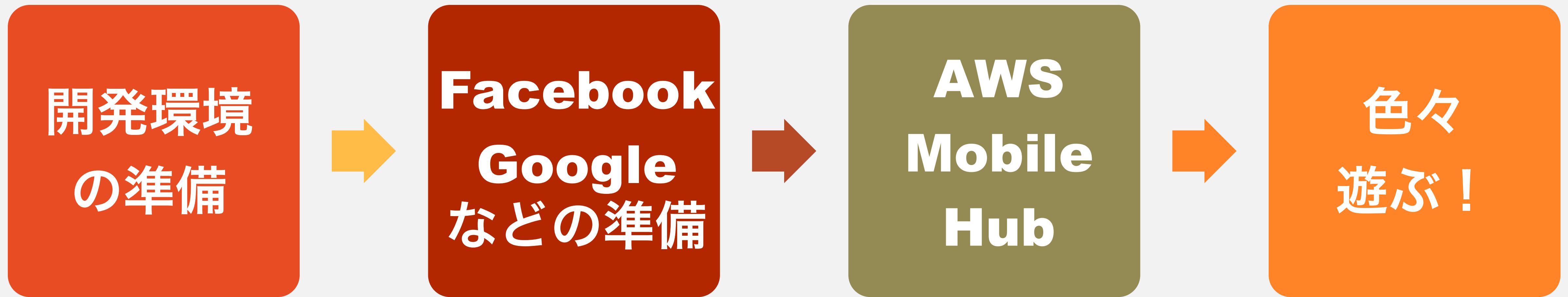
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

ハンズオンの流れ



ハンズオンの流れ



ご注意！

- ◆ 当資料は**Android**向けかつ**Mac**で作成されました
- ◆ ただし**Windows**でもほぼ同じ、**iOS**向けもかなり似たような感じだと思います。
- ◆ **MobileHub**自体は**iOS**および**Windows**での開発もサポートしています。
- ◆ 今回は **iOS**向けではなく**Android**向け資料ですが、きっと将来拡張される予定です（プルリクエストお待ちしております！）
- ◆ **Windows**環境向けのサポートも同じく・・・
(当方は窓環境が全く無いのでサポートできません・・・)



祝 GA! (ベータ取れました!)



キャプチャは古いです(^_^;



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

Mobile Hubハンズオンの流れ

- ♦一旦簡単にアプリを作って動かしてみる
- ♦Cloud Logicを使ってみる
- ♦更にその先へ！



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

AWSコンソールへサインイン

- ◆ **<http://aws.amazon.com/jp/>**
より右上の「コンソールへサインイン」をクリック。
- ◆ MFAを設定してある人はそれぞれの**token**を使ってください。



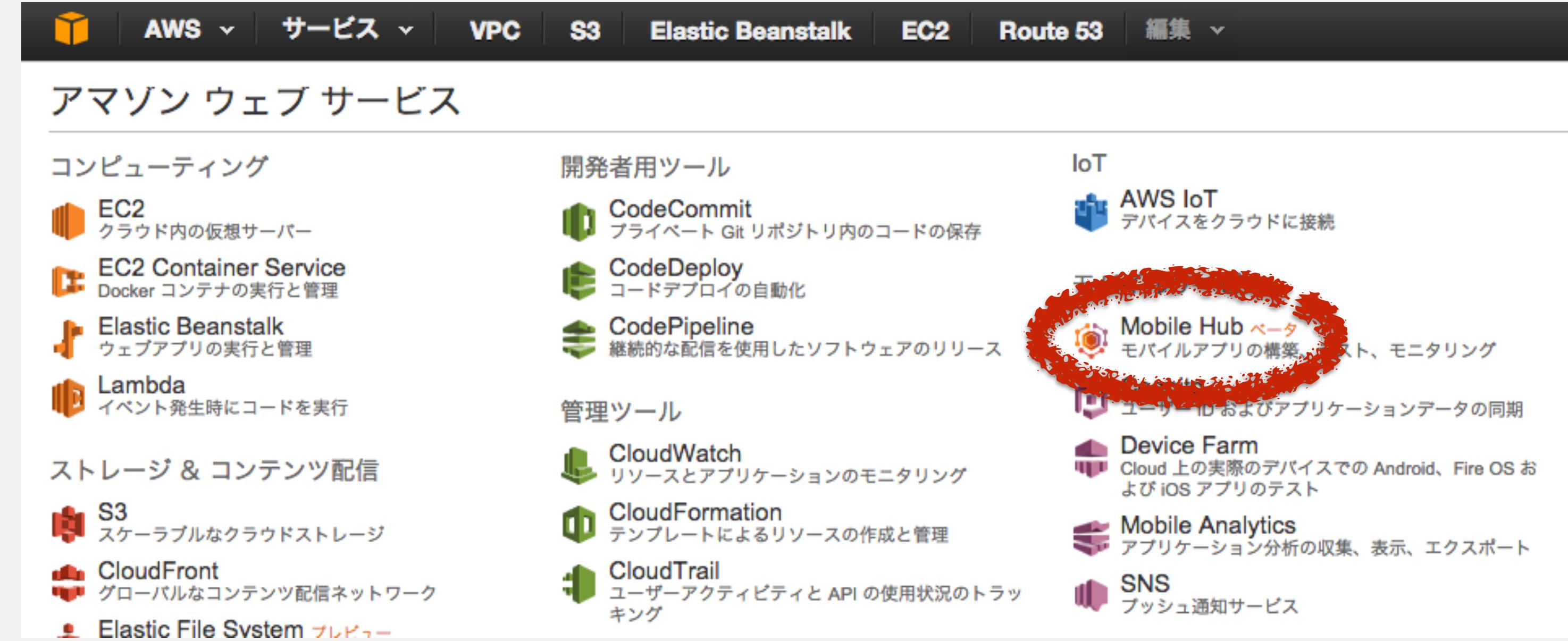
JAWS-UG
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

サービス一覧

♦ モバイルサービスの中の
「Mobile Hub」をクリック



AWS | サービス | VPC | S3 | Elastic Beanstalk | EC2 | Route 53 | 編集

アマゾン ウェブ サービス

コンピューティング

-  EC2 クラウド内の仮想サーバー
-  EC2 Container Service Docker コンテナの実行と管理
-  Elastic Beanstalk ウェブアプリの実行と管理
-  Lambda イベント発生時にコードを実行

ストレージ & コンテンツ配信

-  S3 スケーラブルなクラウドストレージ
-  CloudFront グローバルなコンテンツ配信ネットワーク
-  Elastic File System プレビュー

開発者用ツール

-  CodeCommit プライベート Git リポジトリ内のコードの保存
-  CodeDeploy コードデプロイの自動化
-  CodePipeline 繼続的な配信を使用したソフトウェアのリリース

管理ツール

-  CloudWatch リソースとアプリケーションのモニタリング
-  CloudFormation テンプレートによるリソースの作成と管理
-  CloudTrail ユーザーアクティビティと API の使用状況のトラッキング

IoT

-  AWS IoT デバイスをクラウドに接続

Mobile Hub ページ モバイルアプリの構築、デプロイ、モニタリング

ユーザーIDおよびアプリケーションデータの同期

Device Farm Cloud 上の実際のデバイスでの Android、Fire OS および iOS アプリのテスト

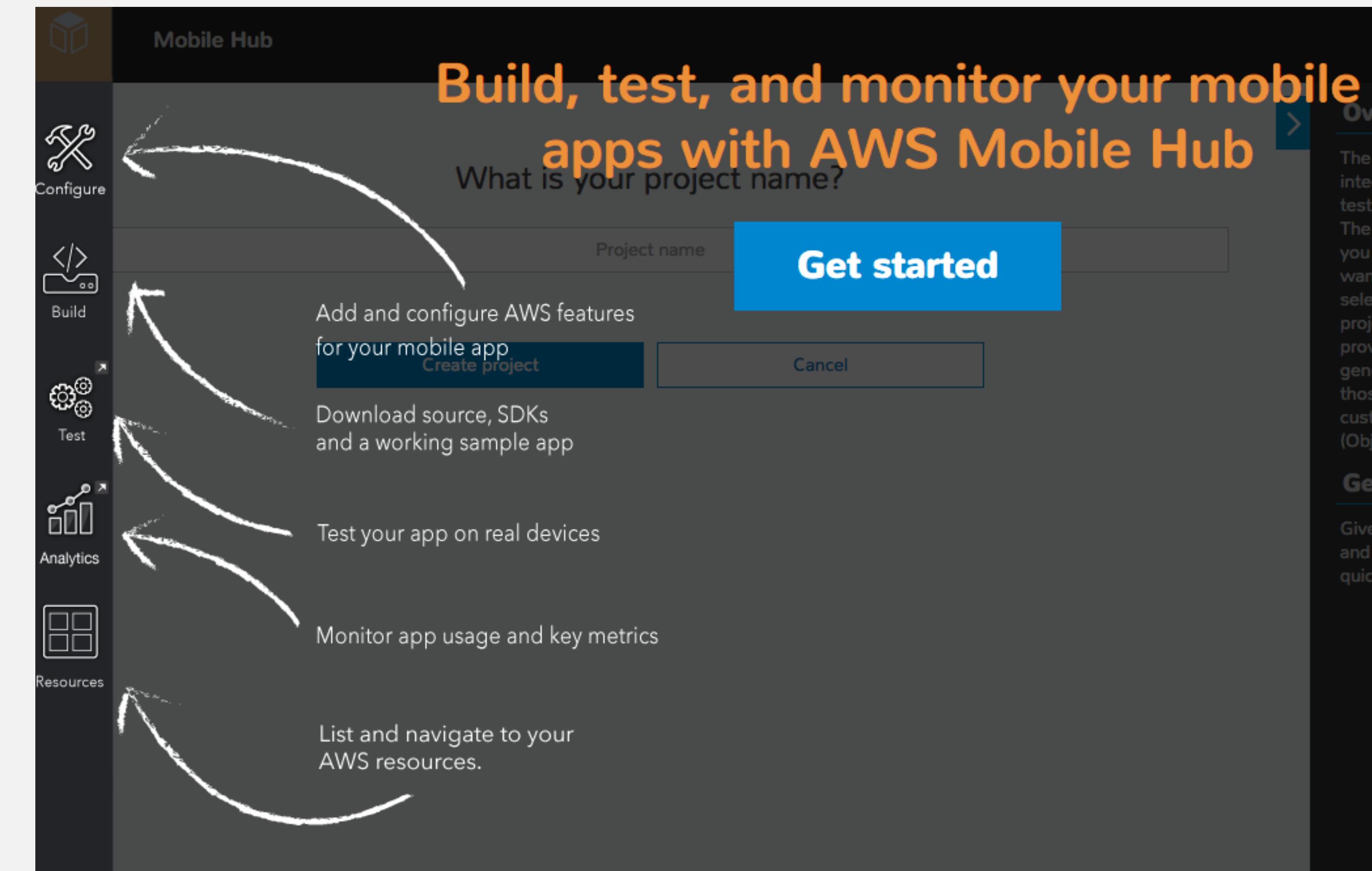
Mobile Analytics アプリケーション分析の収集、表示、エクスポート

SNS ブッシュ通知サービス



AWS Mobile Hub Get started!

- ◆ こんな画面になるので、各項目の説明を見つつ「Get started」をクリック。



JAWS-UG

AWS User Group - Japan

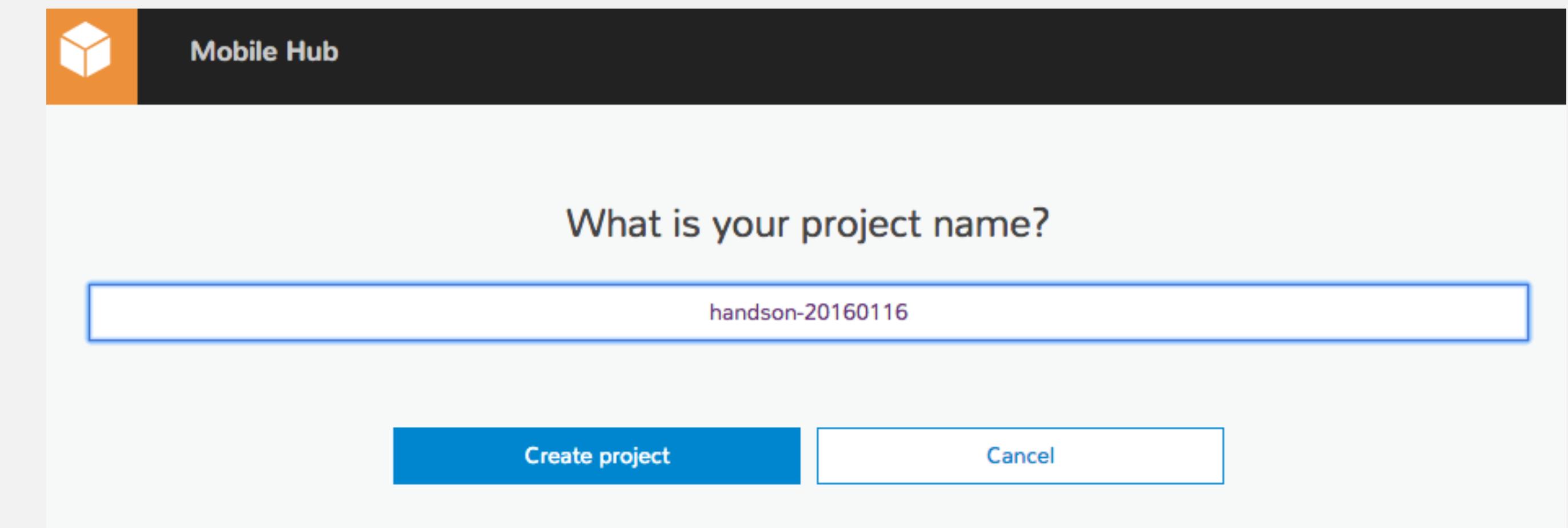
OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

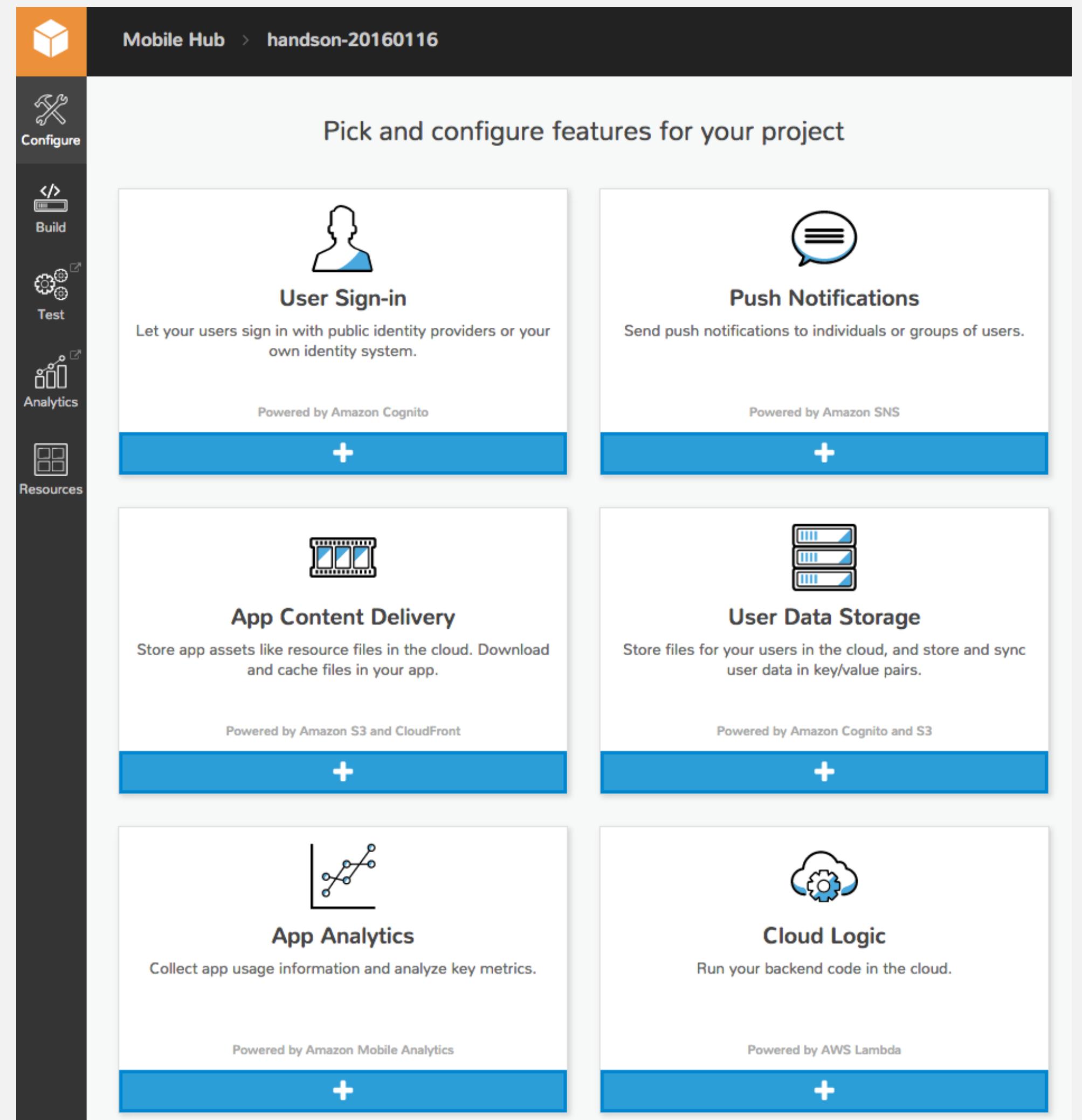
プロジェクトの作成

- ◆ プロジェクト名を
「handson-20160116」と入力し、
「Create project」 をクリック



プロジェクトの設定

- ◆ こんな画面になります。
ここから、プロジェクトの各機能要素を設定します。
- ◆ 左上の「**User Sign-in**」をクリックするところから、はじめましょう。



User Sign-inの設定

- ◆ ユーザー認証を組込むかどうかの選択です。
- ◆ サインインしなくてもアプリが使えるよう
にするなら「**optional**」を、サインイン必
須とするなら「**required**」を選択します。
- ◆ ここでは**optional**を選択します。

 User Sign-in Powered by Amazon Cognito

Let your users sign in with public identity providers or your own identity system.

How will your users sign into your app?

No sign-in


Sign-in is optional

Sign-in is required

How can users sign in?

Coming soon!  Amazon


Facebook

Coming soon!  Google
Coming soon!  Twitter
 Custom

To enable Facebook Login, you will need to configure a Facebook App and enter the App ID here.

Facebook App ID

Save changes
Cancel changes

User Sign-inの設定

- ◆ その下のユーザ認証プロバイダは、現時点（2016年1月）FacebookとCustomしか選択できません。
- ◆ ここではFacebookを選択し、AppIDを入力します。
- ◆ 事前準備編で作成したAppIDを入力して「Save changes」をクリックします。

User Sign-in

Powered by Amazon Cognito

How will your users sign into your app?

No sign-in

Sign-in is optional

Sign-in is required

How can users sign in?

Amazon

Facebook

Google

Twitter

Custom

To enable Facebook Login, you will need to configure a Facebook App and enter the App ID here.

Facebook App ID

Save changes

Cancel changes



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>



User Sign-inの設定完了

- ♦ ひとつ設定をすると、このように色が変わります。
このような感じでどんどん設定していきましょう。
- ♦ 次は右隣の「Push Notifications」をクリックします。

Pick and configure features for your project

User Sign-in

Let your users sign in with public identity providers or your own identity system.

Powered by Amazon Cognito

[Purple button with gear icon]

Push Notifications

Send push notifications to individuals or groups of users.

Powered by Amazon SNS

[Blue button with plus icon]

App Content Delivery

Store app assets like resource files in the cloud. Download and cache files in your app.

Powered by Amazon S3 and CloudFront

[Blue button with plus icon]

User Data Storage

Store files for your users in the cloud, and store and sync user data in key/value pairs.

Powered by Amazon Cognito and S3

[Blue button with plus icon]

App Analytics

Collect app usage information and analyze key metrics.

Powered by Amazon Mobile Analytics

[Blue button with plus icon]

Cloud Logic

Run your backend code in the cloud.

Powered by AWS Lambda

[Blue button with plus icon]



Push Notificationsの設定

- ◆ プッシュ通知を利用する場合 「Enable push」 を選択します。
 - ◆ 事前準備編のGoogle Cloud Messaging(GCM)で作成したAPI Keyを入力します。
 - ◆ 「Sender ID」には、Google Developers Console のダッシュボードに表示されている「プロジェクト番号」を入力します。

Push Notifications

Powered by Amazon SNS

Send push notifications to individuals or groups of users.

Do you want to send push notifications to your app?

Not required

Enable push

What platforms do you want to send messages to?

Android

iOS Prod

iOS Dev

To enable Google Cloud Messaging (GCM), you will need to enter your Google credentials here.

API Key

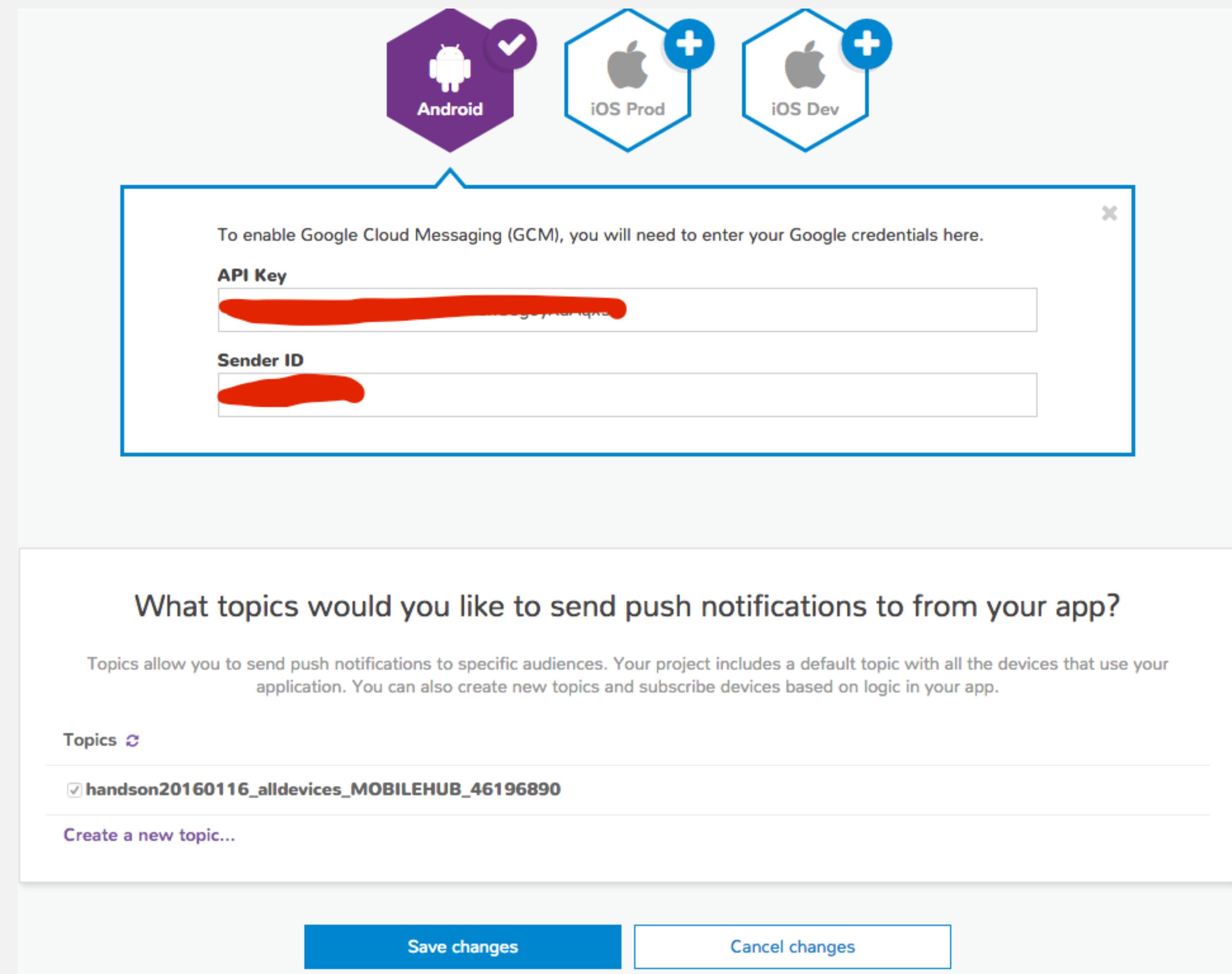
Sender ID

Save changes

Cancel changes

Push Notificationsの設定

- ◆ いま作成されるのは、アプリを入れた全デバイスにクラウド側から一斉に通知するトピックになります。
- ◆ 「**Save changes**」をクリックして保存します。



To enable Google Cloud Messaging (GCM), you will need to enter your Google credentials here.

API Key

Sender ID

What topics would you like to send push notifications to from your app?

Topics

handson20160116_alldevices_MOBILEHUB_46196890

Create a new topic...

Save changes

Cancel changes

各設定画面のボタン

- ◆ ちなみにこのボタン、設定外終わった項目では右のようになります。
- ◆ 「**Configure more features**」で各設定一覧に戻り、「**Build your app**」でビルド画面に進みます。



JAWS-UG

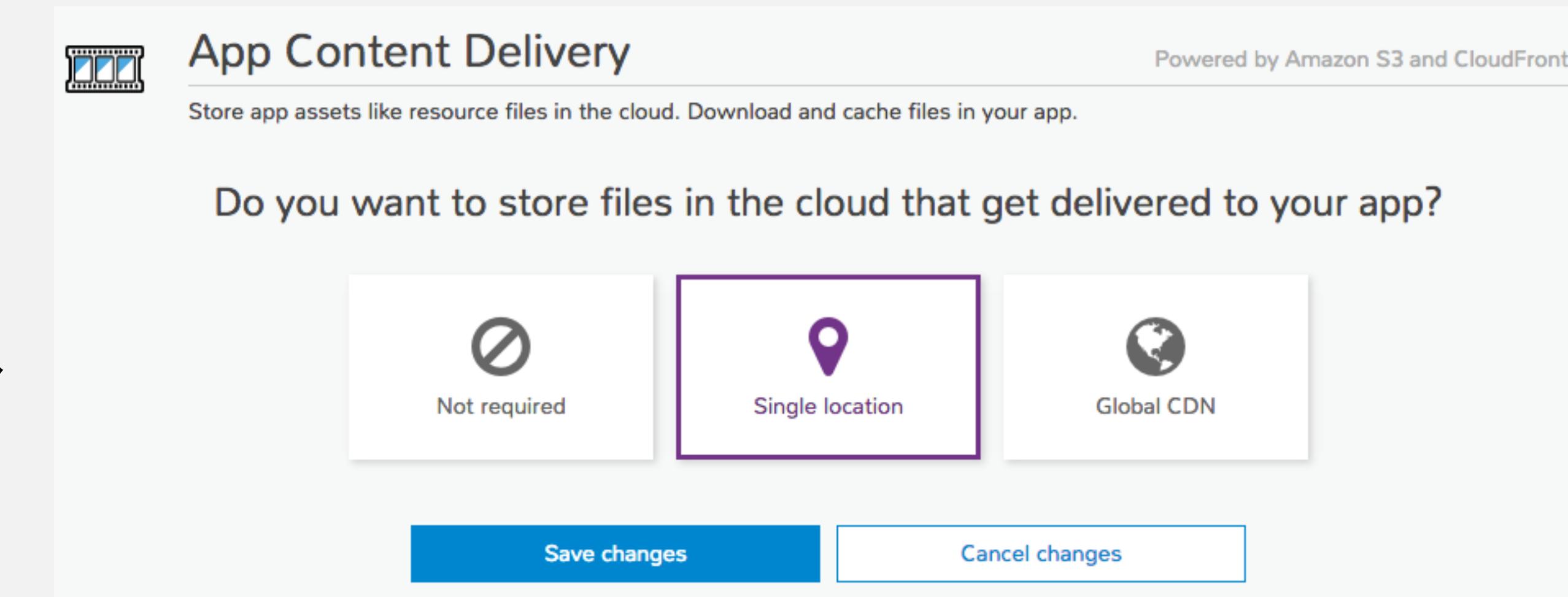
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

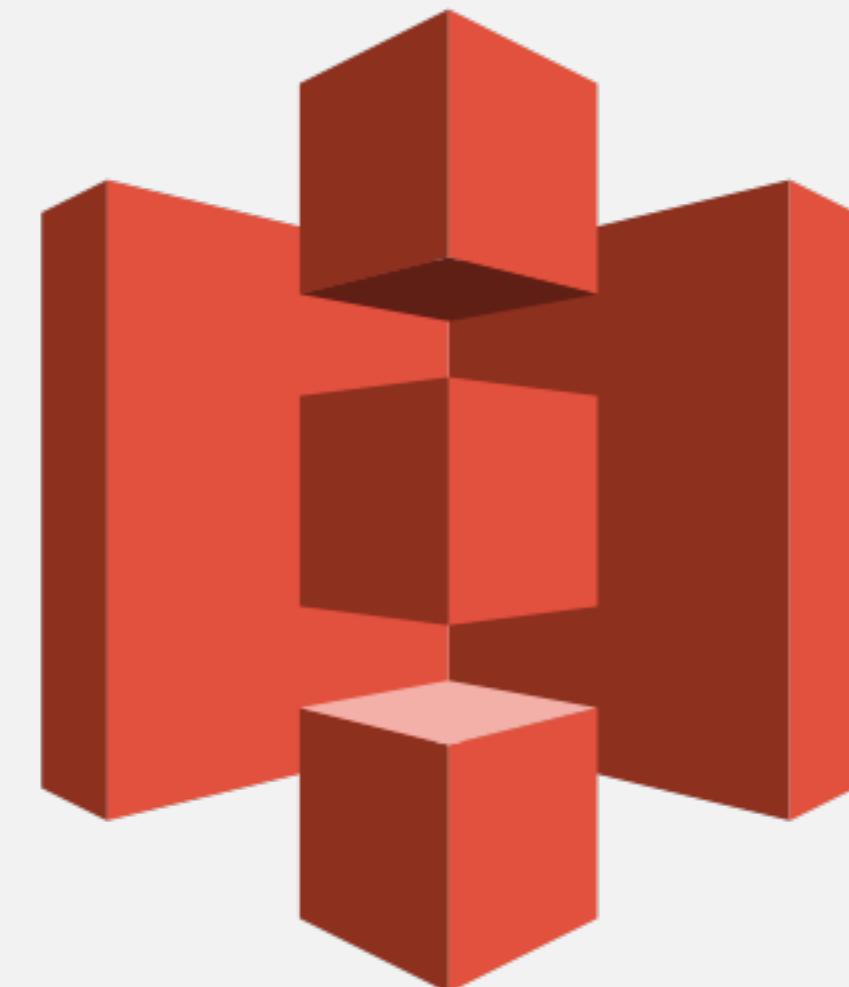
App Content Deliveryの設定

- ◆ アプリが利用する画像ファイルなどを外部クラウドストレージにおいて、アプリに対して配信する機能の設定です。
- ◆ 「**Single location**」を選択するとS3から直接ダウンロード、「**Global CDN**」を選択するとCloudFrontからファイルを配信します。
- ◆ ここでは**Single location**を選択します。



ちょっと待った！Amazon Simple Storage Service (Amazon S3) (えすすりー) って何？

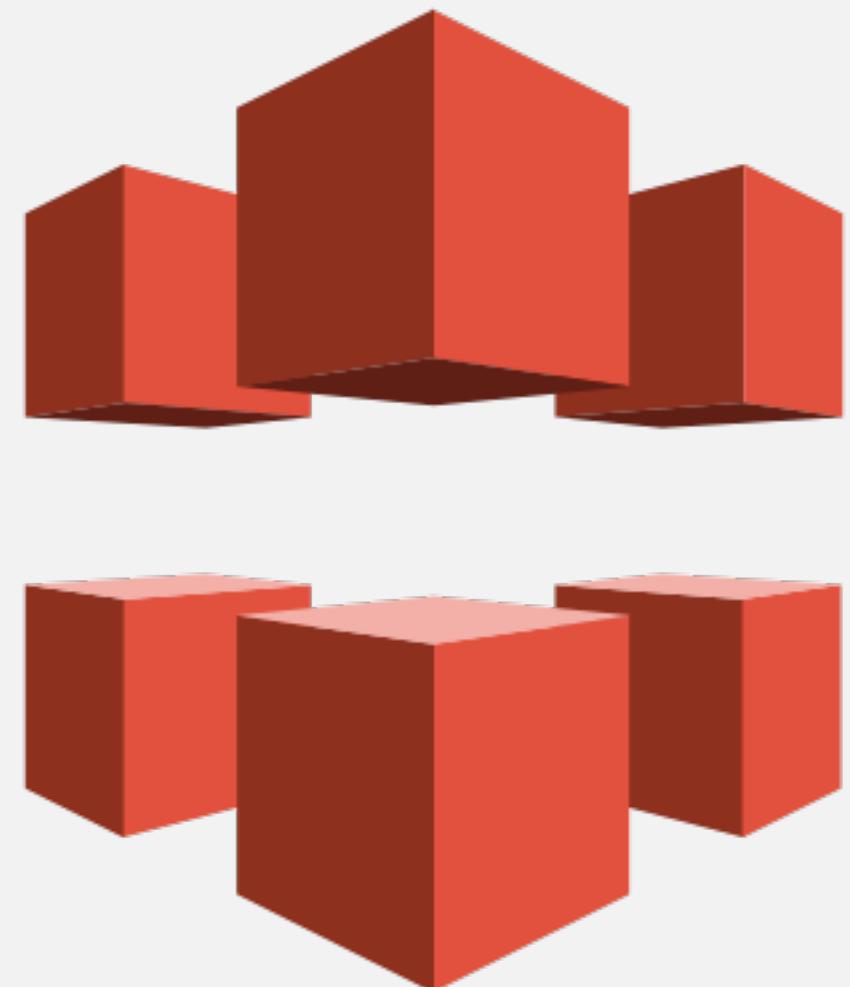
- ♦ シンプル、スケーラブルなクラウドストレージであり、
AWSの根幹をなすサービスのひとつ
- ♦ イレブンナイン(**99.99999999%**)の堅牢性と、**99.99%**の可用性
- ♦ **https**でデータのやり取りが出来るので、**HTML**等を置いて静的な**Web**サーバの代わりに使える
- ♦ 勿論アプリのデータ保管場所としても使える
- ♦ デフォルト非公開状態になるが、間違って機密データを
公開状態に設定しないように！
- ♦ 詳細：**<https://aws.amazon.com/jp/cloudfront/details/>**



ちょっと待った！

Amazon CloudFront (くらうどふろんと) って何？

- ◆ 全世界にエッジロケーションを持つコンテンツ配信ネットワーク
(CDN)
- ◆ 日本にも2箇所エッジロケーションがあり、ネットワーク的に近い方からコンテンツ配信される
- ◆ 入れておくと大元のサーバまでアクセスが来ないので、
Yahoo砲にも余裕で耐えられる様になる
- ◆ 大元のコンテンツ配信サーバは、オンプレミスでも普通に使える
- ◆ 最近は**WAF**とも統合され、ますます進化
- ◆ 詳細：<https://aws.amazon.com/jp/cloudfront/details/>



App Content Deliveryの設定

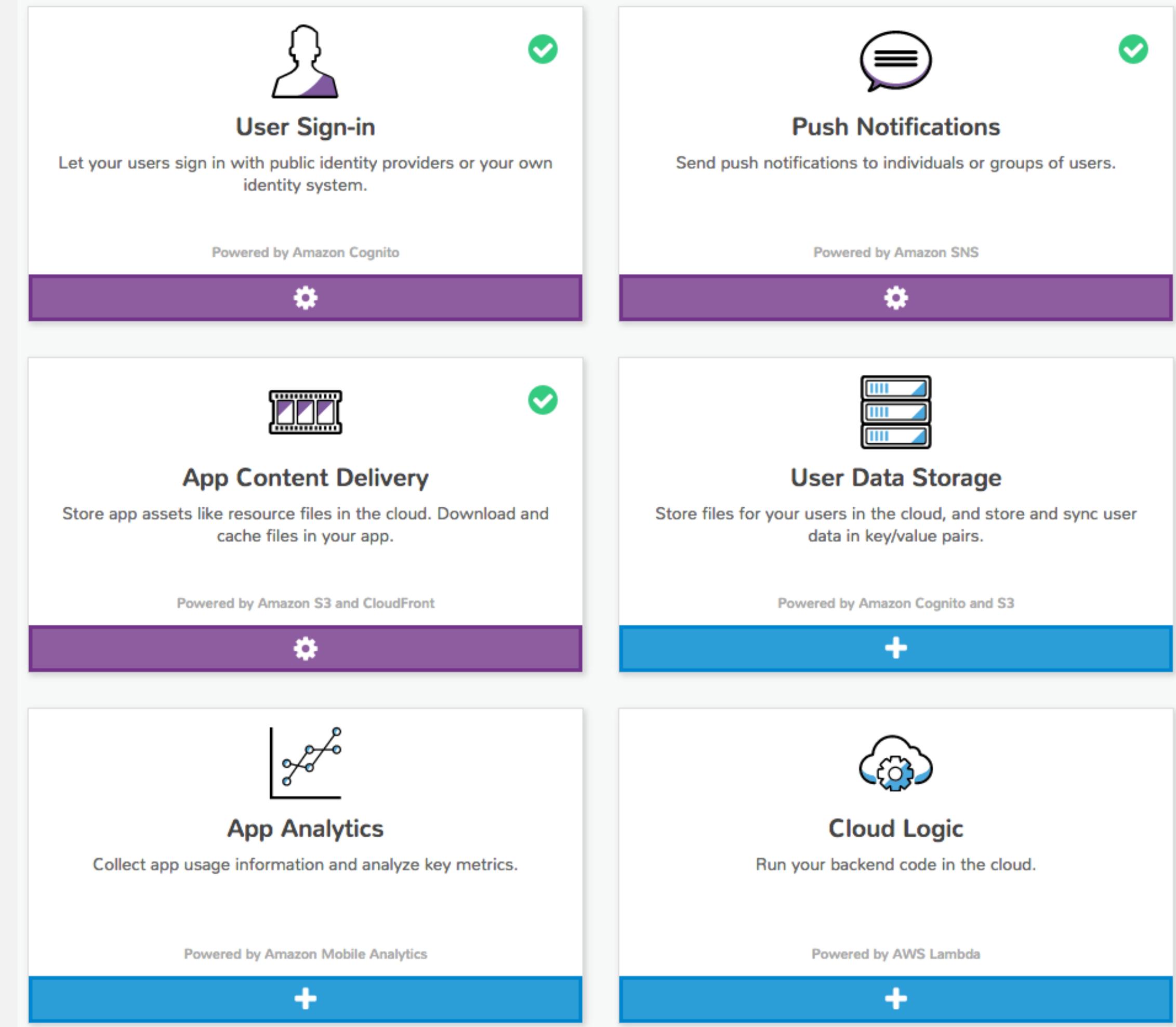
ご注意ください！

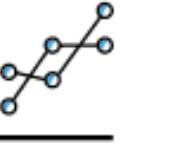
- ♦ これらのファイルの実体はS3の「**handson-contentdelivery-mobilehub-数字**」というようなバケット名でパブリック状態で置かれています（URLが分かれば誰でもダウンロードできる状態）。
- ♦ アクセス制御が必要なコンテンツの配信には、相応の作りこみが必要になります。



ここまで設定

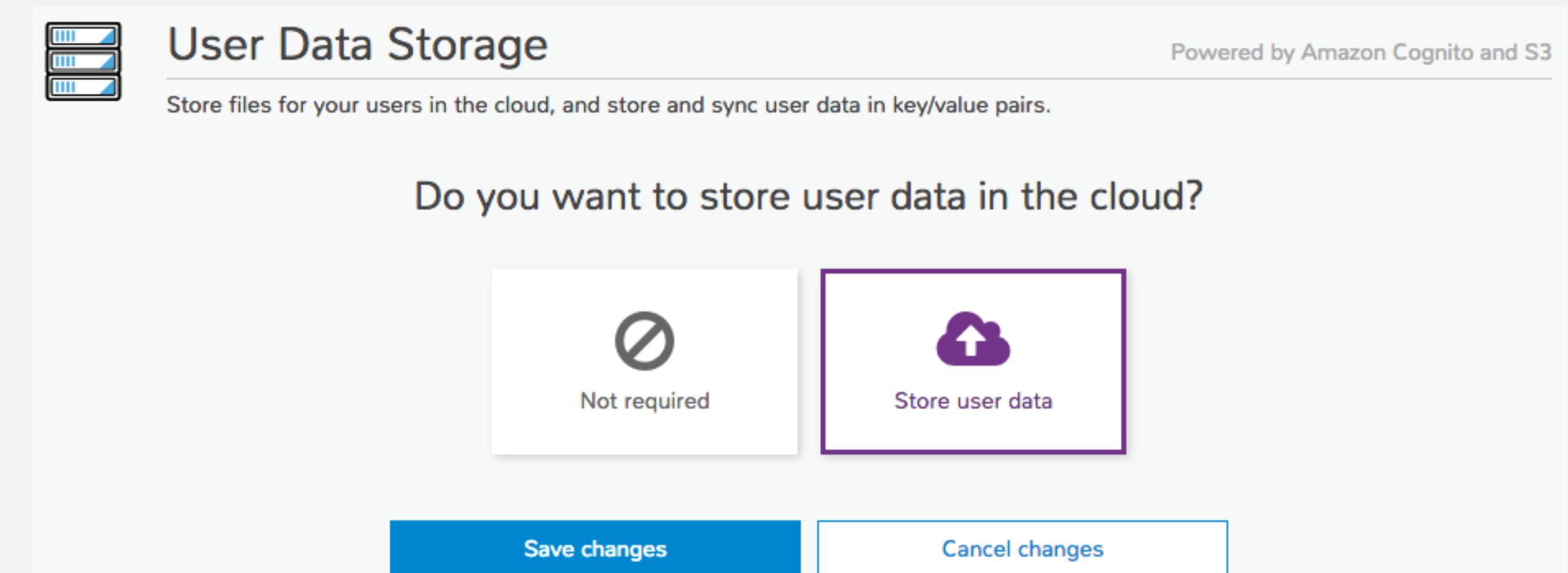
- ◆ どんどん色が変わってきました！
- ◆ あと2つ、**User Data Storage** と
App Analyticsを設定してみましょう。



 User Sign-in Let your users sign in with public identity providers or your own identity system. Powered by Amazon Cognito	 Push Notifications Send push notifications to individuals or groups of users. Powered by Amazon SNS
 App Content Delivery Store app assets like resource files in the cloud. Download and cache files in your app. Powered by Amazon S3 and CloudFront	 User Data Storage Store files for your users in the cloud, and store and sync user data in key/value pairs. Powered by Amazon Cognito and S3
 App Analytics Collect app usage information and analyze key metrics. Powered by Amazon Mobile Analytics	 Cloud Logic Run your backend code in the cloud. Powered by AWS Lambda

User Data Storageの設定

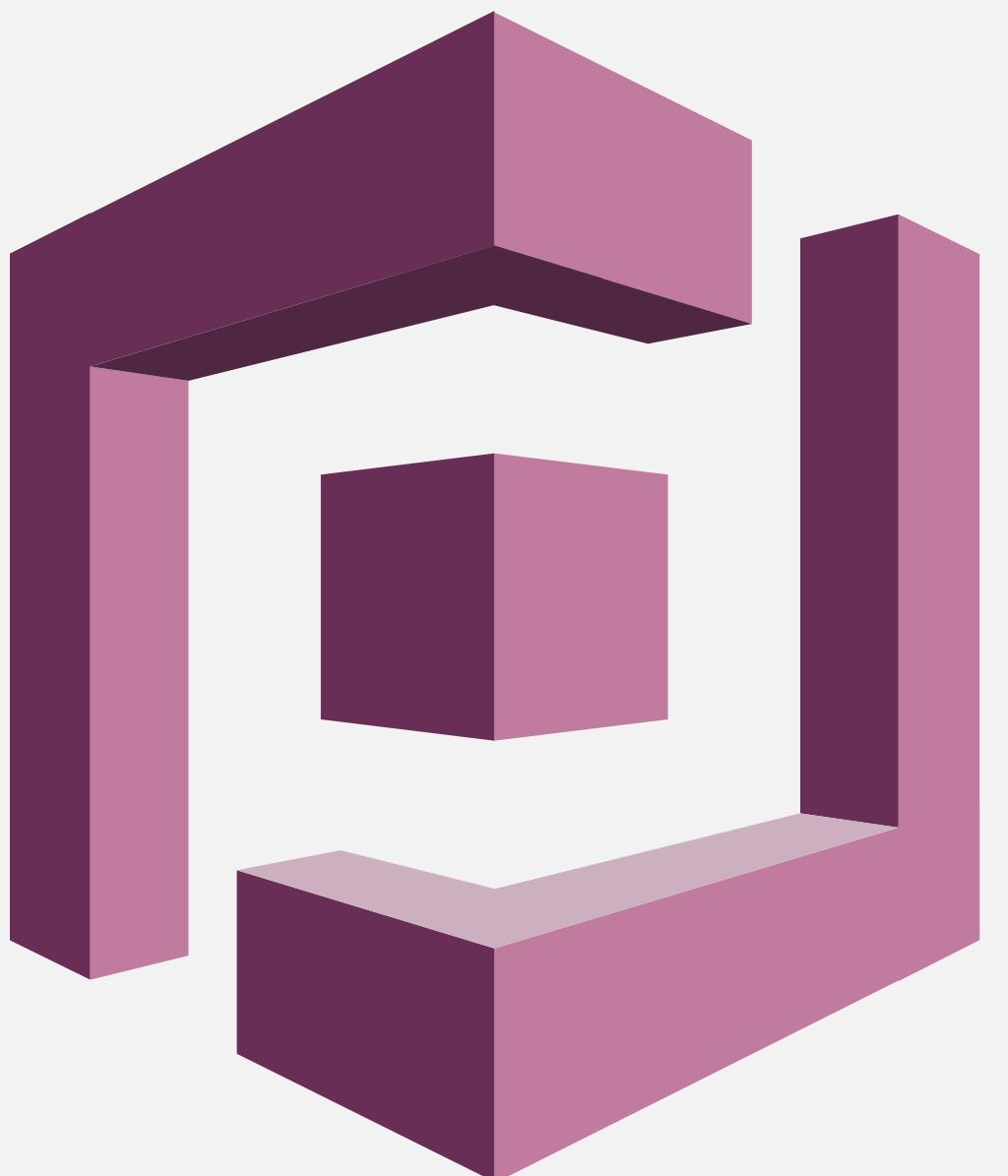
- ♦ ユーザに紐づく情報をハッシュマップのような形か、**S3**上にファイルとしてクラウド側に保存できるようになります。
- ♦ ユーザ認証を利用しない場合でも、匿名ユーザ扱いで**Cognito**にハッシュマップを保存可能です。
- ♦ **S3**上にファイルをプライベートな状態で保存するには、ユーザ認証が必要になります。
- ♦ ここでは「**Store user data**」を選択し「**Save changes**」をクリックして保存します。



ちょっと待った！

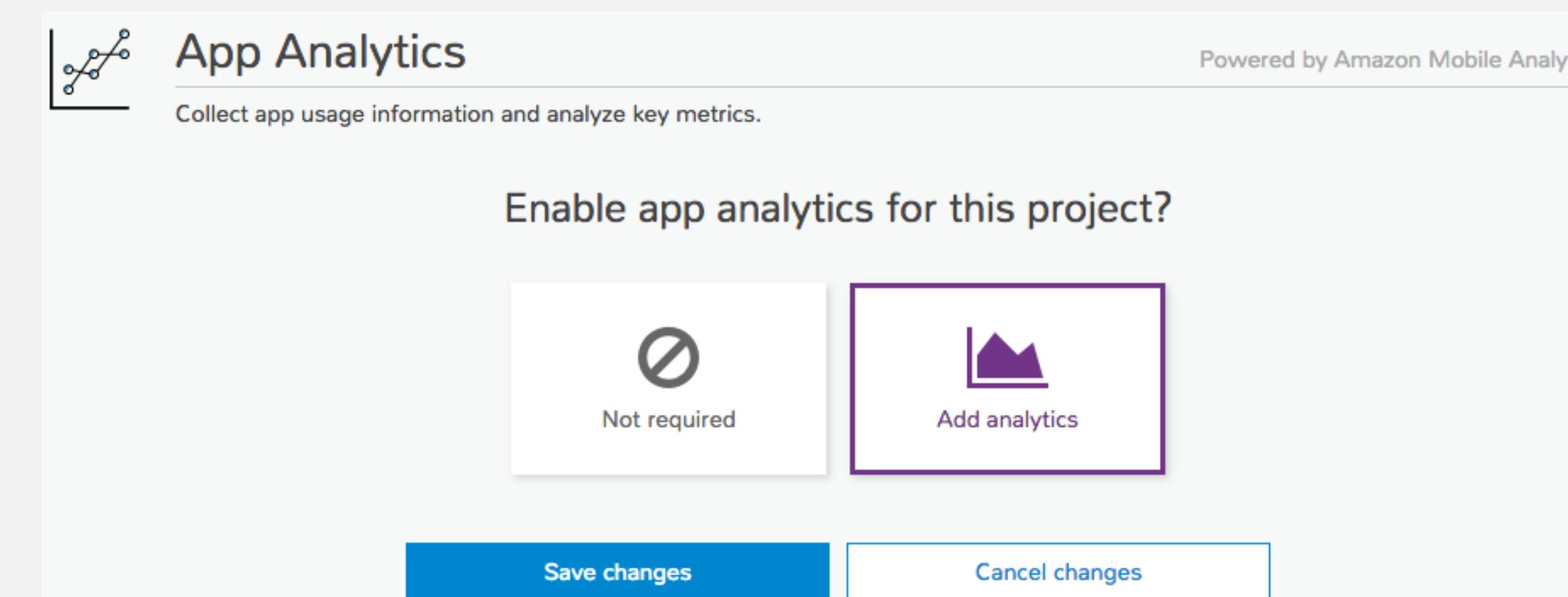
Amazon Cognito (こぐにーと) って何？

- ◆ 一時的なAWS認証情報の配布及び管理、モバイル向けにはユーザID管理とデータ同期サービスを提供
- ◆ モバイル向けにオフライン機能
- ◆ デバイス間での情報の同期をサポート
- ◆ 実質2つのサービス (**Cognito Identity**と**Cognito Sync**) が含まれているので若干混乱を招くので注意
- ◆ 詳細：<http://docs.aws.amazon.com/cognito/devguide/>



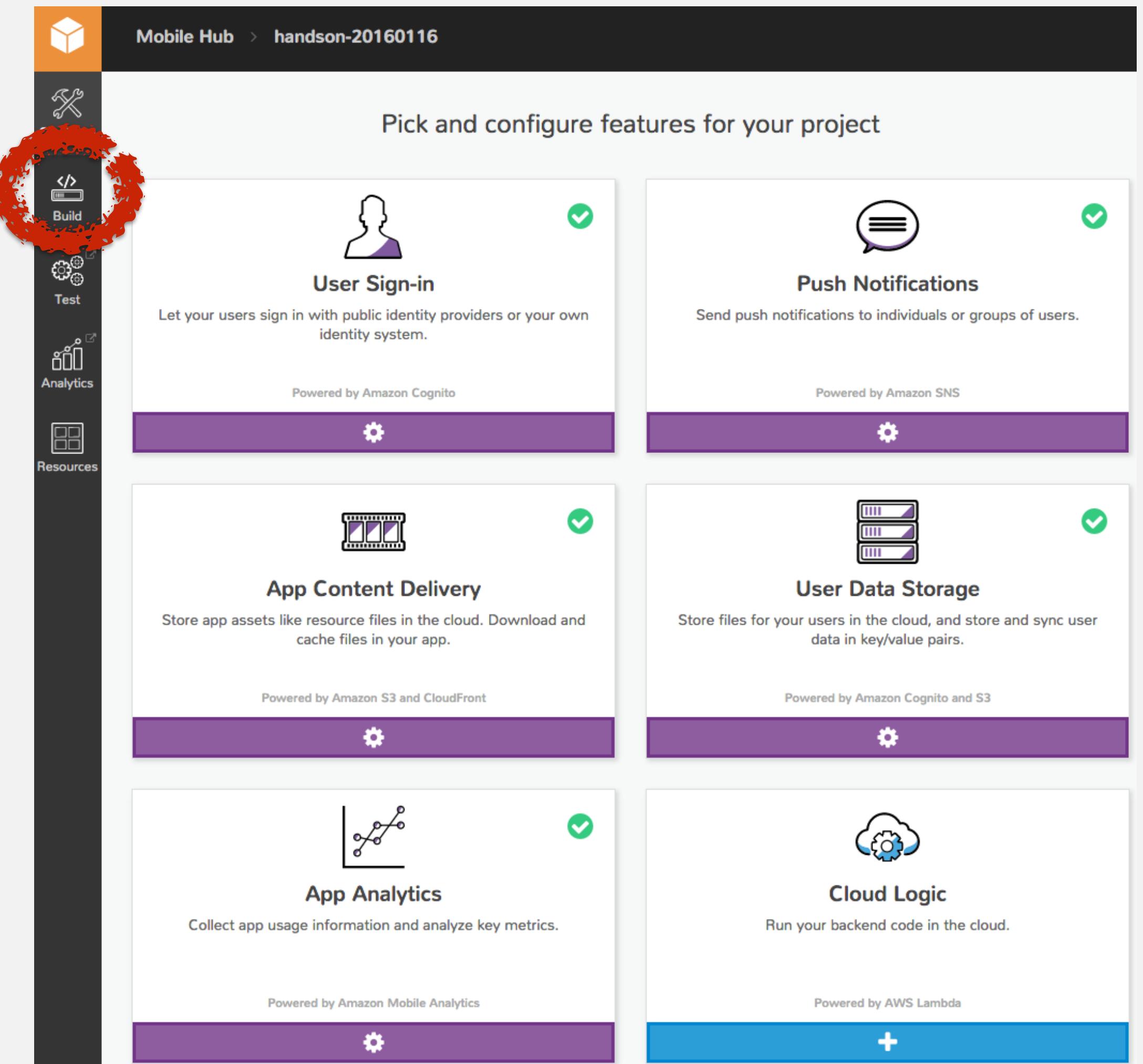
App Analyticsの設定

- ◆ アプリの利用動向をレポートするツール「**Mobile Analytics**」を利用するかどうかの設定です。
- ◆ 日毎のアクティブユーザ数（**DAU**）をはじめ、様々な指標を確認することが出来ます。ここでは利用する方を選択しています。
- ◆ 「**Save changes**」をクリックして保存します。



ここまで設定

- ❖ 残り1つが気になりますが、
ここで一旦左メニューの上から2番目
「Build」 アイコンをクリックして、
ソースをビルドしてみます。



The screenshot shows the AWS Mobile Hub console with the title "Mobile Hub > handson-20160116". The left sidebar has icons for Build, Test, Analytics, and Resources, with "Build" being the second item from the top and circled in red. The main area is titled "Pick and configure features for your project" and contains six feature cards:

- User Sign-in**: Powered by Amazon Cognito.
- Push Notifications**: Powered by Amazon SNS.
- App Content Delivery**: Powered by Amazon S3 and CloudFront.
- User Data Storage**: Powered by Amazon Cognito and S3.
- App Analytics**: Powered by Amazon Mobile Analytics.
- Cloud Logic**: Powered by AWS Lambda.



JAWS-UG

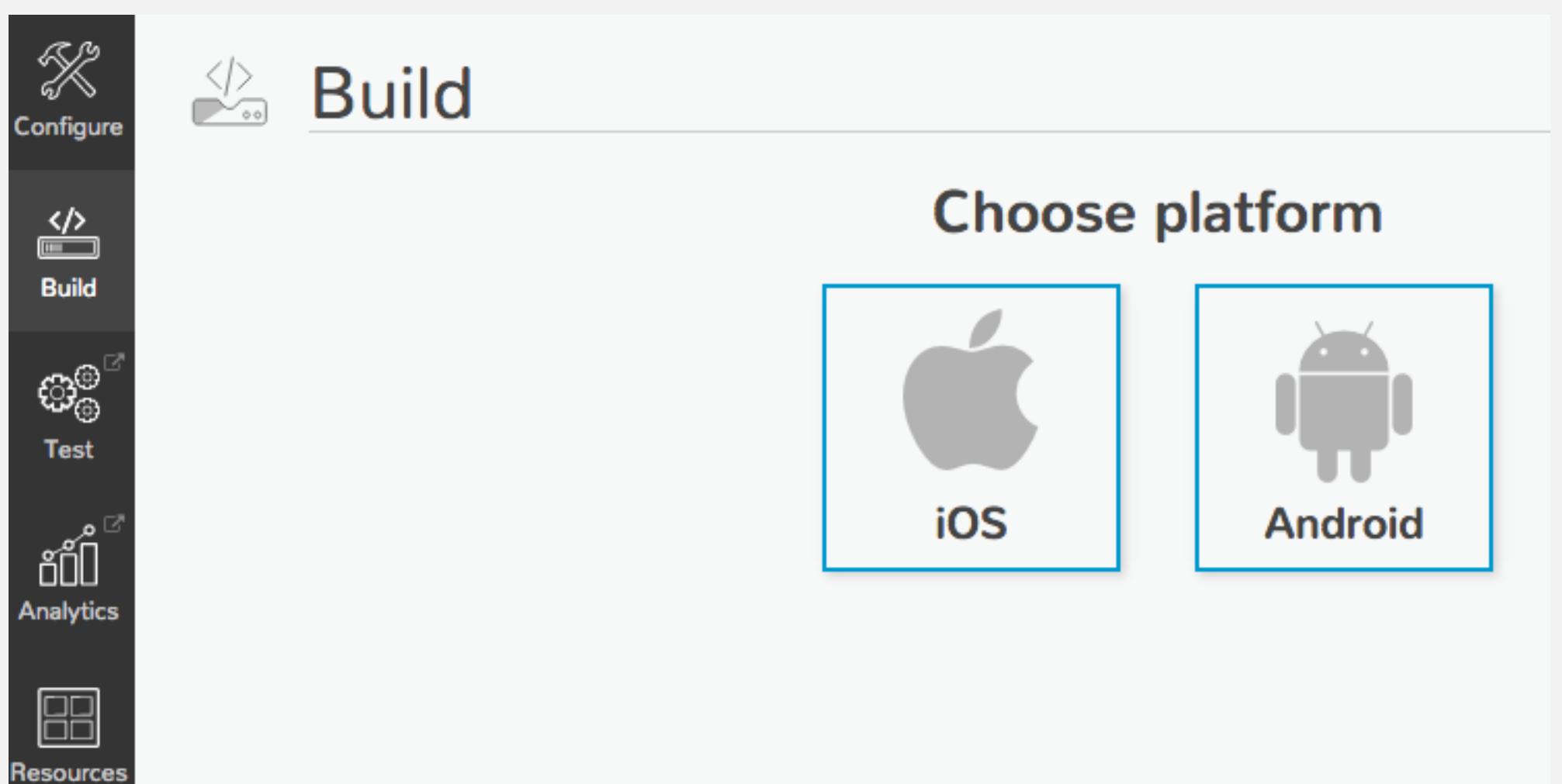
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

Buildプラットフォームの選択

- ◆ iOSまたはAndroidが選択できますが、ここではAndroidをクリックします。
- ◆ クリックした瞬間ソースがビルドされますので、しばらく待ちます。



JAWS-UG

AWS User Group - Japan

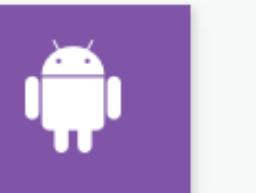
OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

ソースダウンロード

- ◆ 「Download Android source package」のリンクが現れたら、クリックしてソースのZIPファイルをダウンロードします。

 Build

 Download Android source package

- 1. Setup
- 2. Compile and Run
- 3. Discover
- 4. Develop
- 5. Troubleshoot

System Requirements

Your system must include the following:

- Android Studio 1.4 or newer
- Android SDK 4.4 (KitKat) API Level 19 or newer
- Android SDK Build-tools 23.0.1
- Android Device with Android OS 4.0.3 (IceCreamSandwich) API Level 15 or newer

Install Android Studio

If you have not already installed the Android Studio IDE (Integrated Development Environment) on your host, you can download it from the following location:
<https://developer.android.com/sdk>

Install Android SDK (Software Development Kit)

If you have not already installed the latest Android SDK, you can do so within Android Studio.

Go to...
 Tools -> Android -> SDK Manager -> Appearance & Behavior -> System Settings -> Android SDK

Then, select the Android SDK Platform version you would like to use. The SDK Manager will download the appropriate Android SDK Platform version and apply any available updates.

Install AVD (Android Virtual Device)

If you have not already installed an Android Virtual Device, you can do so within Android Studio.

Go to...
 Tools -> Android -> AVD Manager

Then, click "+ Create Virtual Device" and select the device you would like to use in the Android emulator (e.g., Nexus 5). Select the appropriate image from the list (e.g., Lollipop API Level 22 x86 Android 5.1.1) and click the "Next" button. Name the AVD and click the "Finish" button.

Skip
I'm done with this step



JAWS-UG

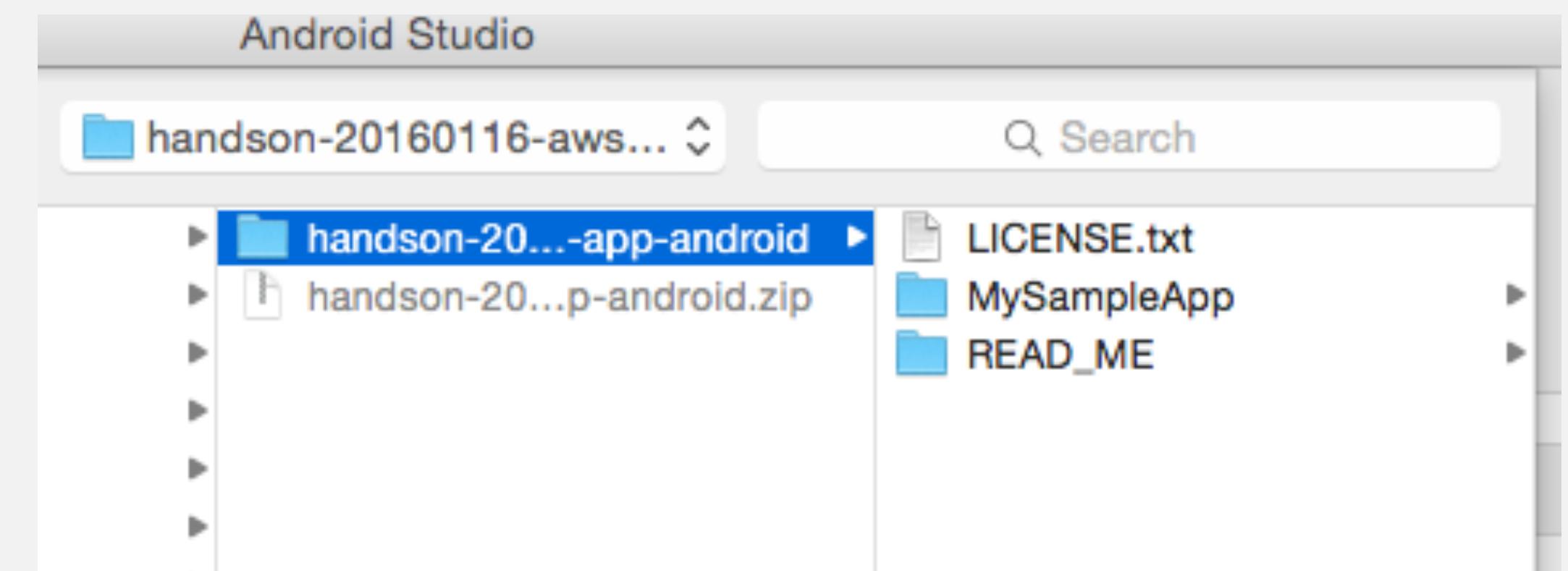
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
 GitHub repositories <https://github.com/jaws-ug/>

Android Studioでアプリをビルド

- ◆ まずダウンロードしたZIPファイルを解凍します。
- ◆ ここから先はAndroid Studioで作業を行います。

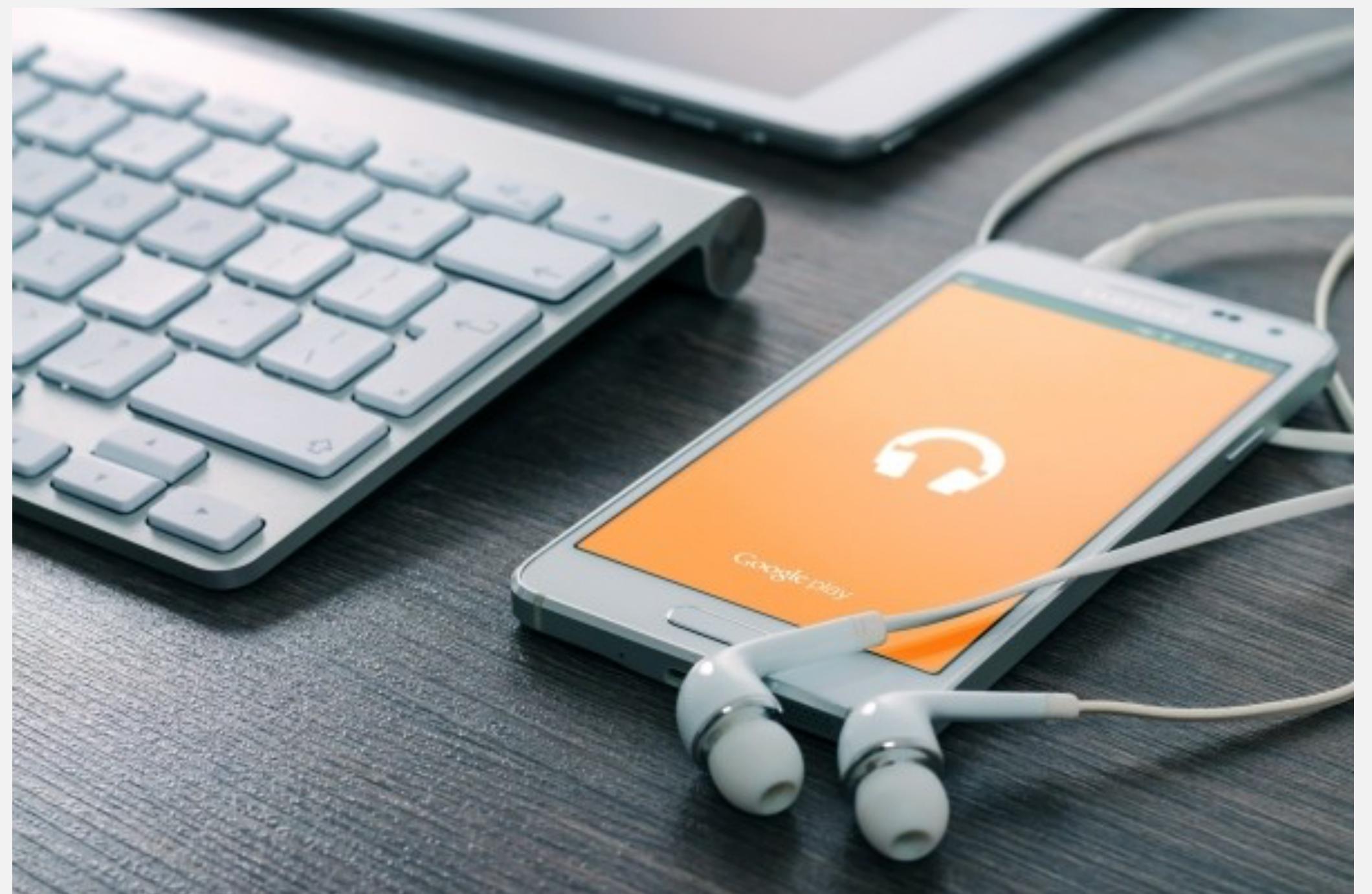


この辺で
ひとやすみ



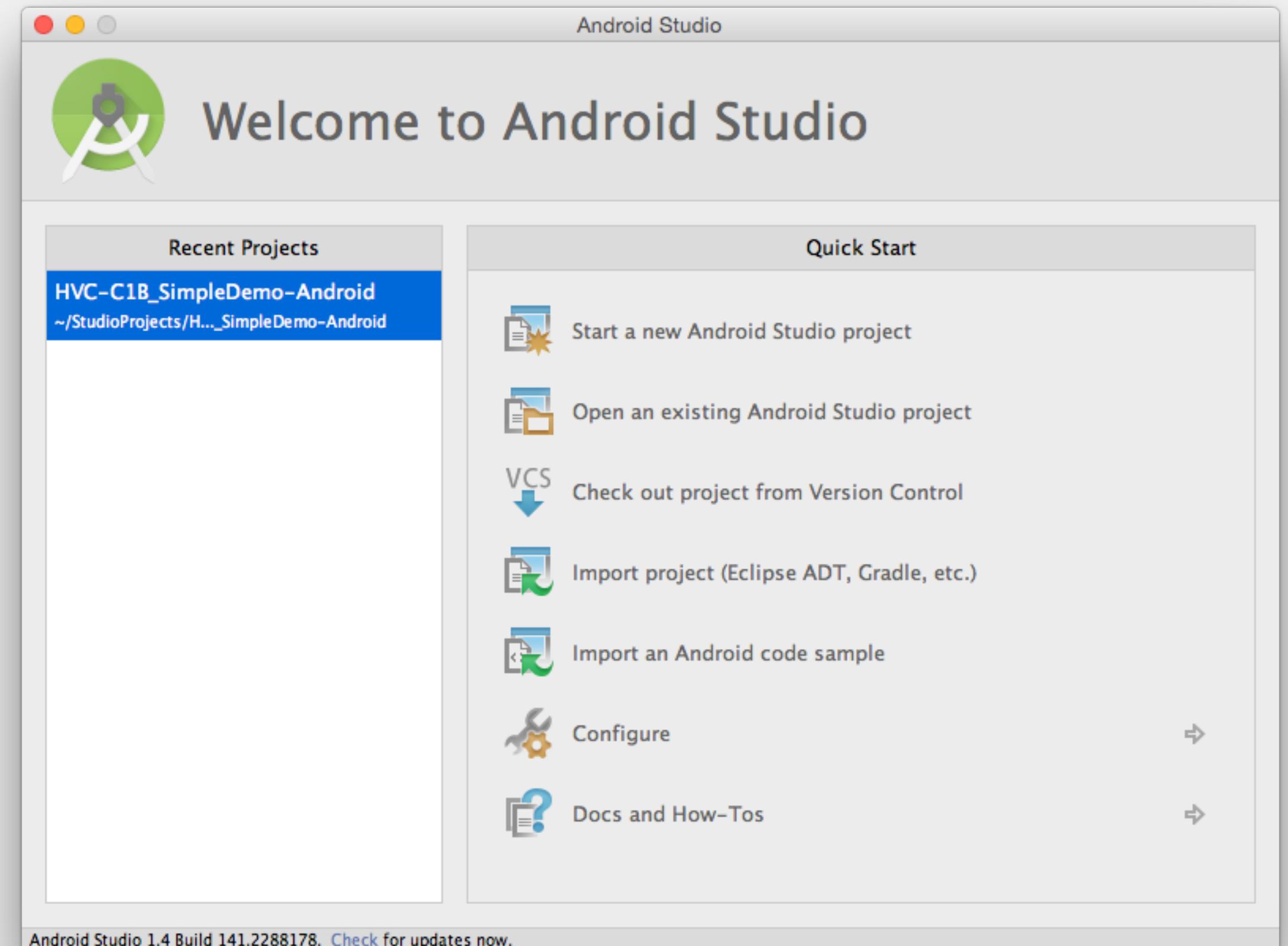
実機の端末で開発する場合

- ♦ **USBケーブルで端末とPCを接続します。**
- ♦ 端末側に「**USBデバッグが接続されました**」と表示されれば**OK**です。
- ♦ 表示されない人は、周りの方やサポートスタッフにお声がけください。



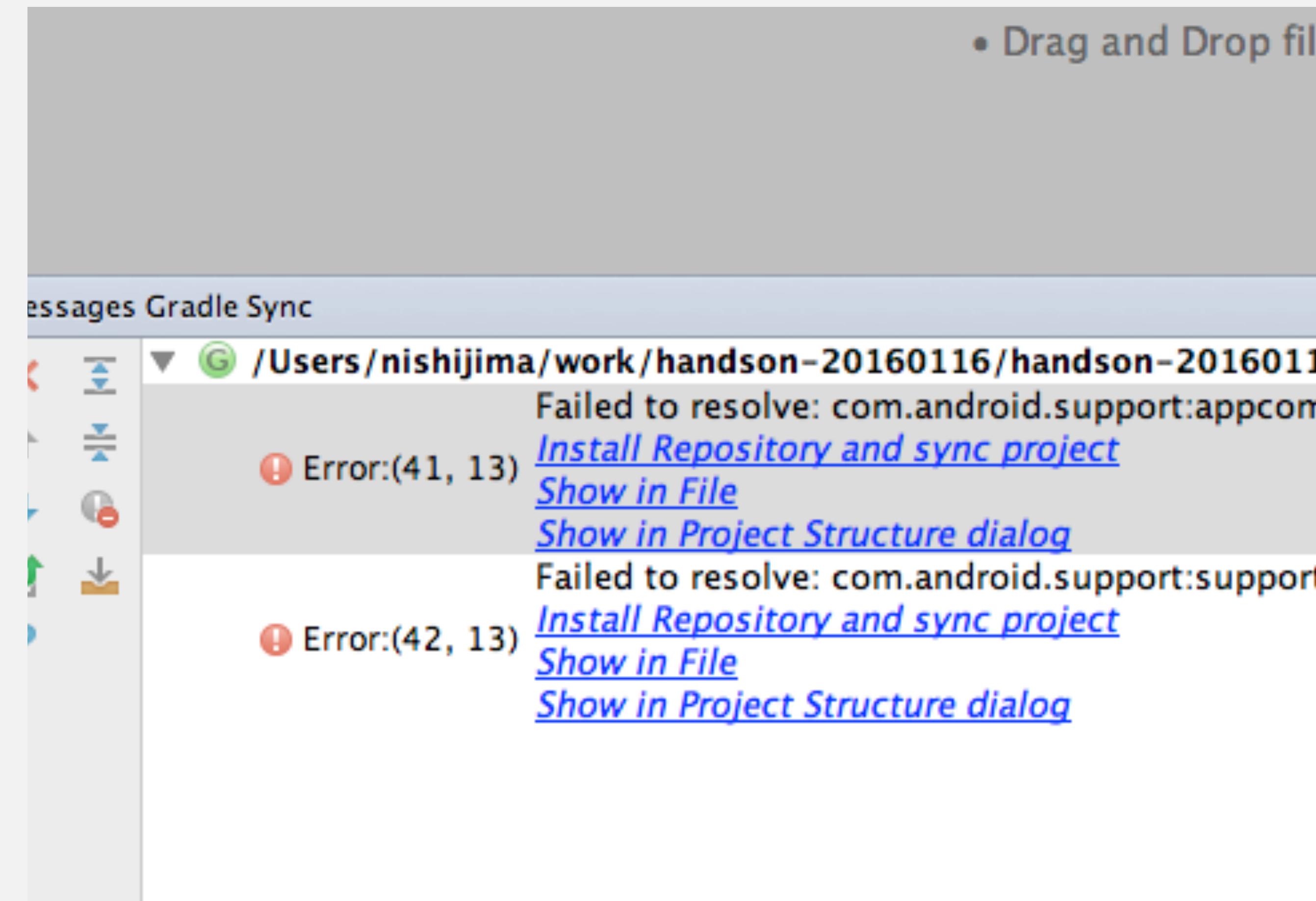
Android Studioでアプリをビルド

- ◆ **Android Studioを起動して、「Import Project (Eclipse...)** を選択。
- ◆ フォルダは先程解凍したフォルダの「MySampleApp」を選択。



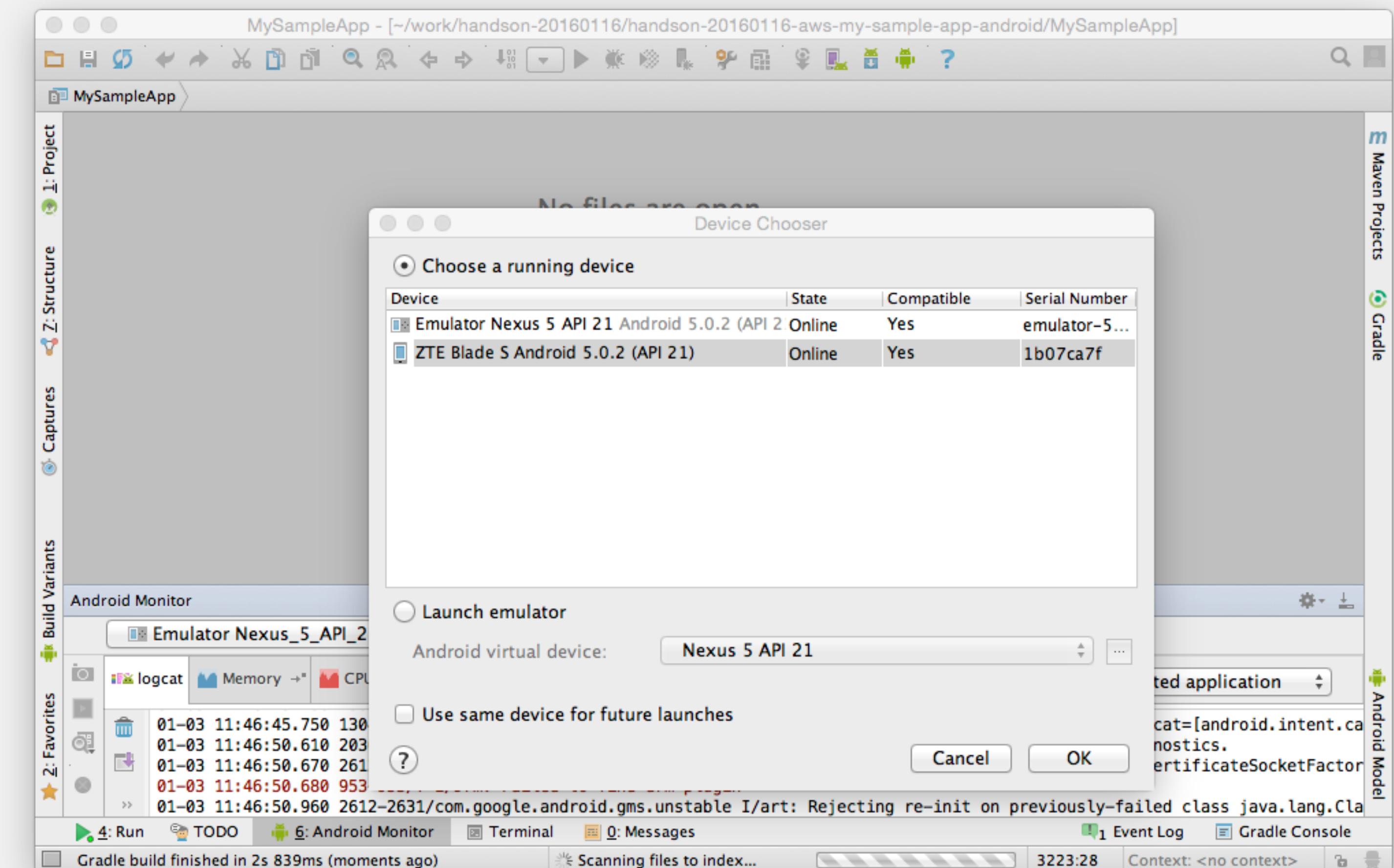
この時点で何かエラーが出たら

- ◆ 例えばこのようなエラー。
- ◆ 「読んで適切に対処しましょう」としか言いようがありません(^^;)
- ◆ 右のエラーは「**Install Repository and sync project**」をクリックすることで解決します。
- ◆ プロジェクト読み込み直後のフルビルドは、ネットワークが安定していないと厳しいことが多いと思います。



アプリの実行

- ◆ メニューの**Run -> Run ‘app’** を選択して実行してみましょう。
- ◆ 「**Choose a running device**」側には、起動していればエミュレーターと、接続している実機が表示されているはずです。
- ◆ 「**Launch emulator**」側を選択すると、事前に登録したAVDを指定してエミュレーターを起動することができます。
- ◆ どちらかを選択して「**OK**」をクリックします。



JAWS-UG

AWS User Group - Japan

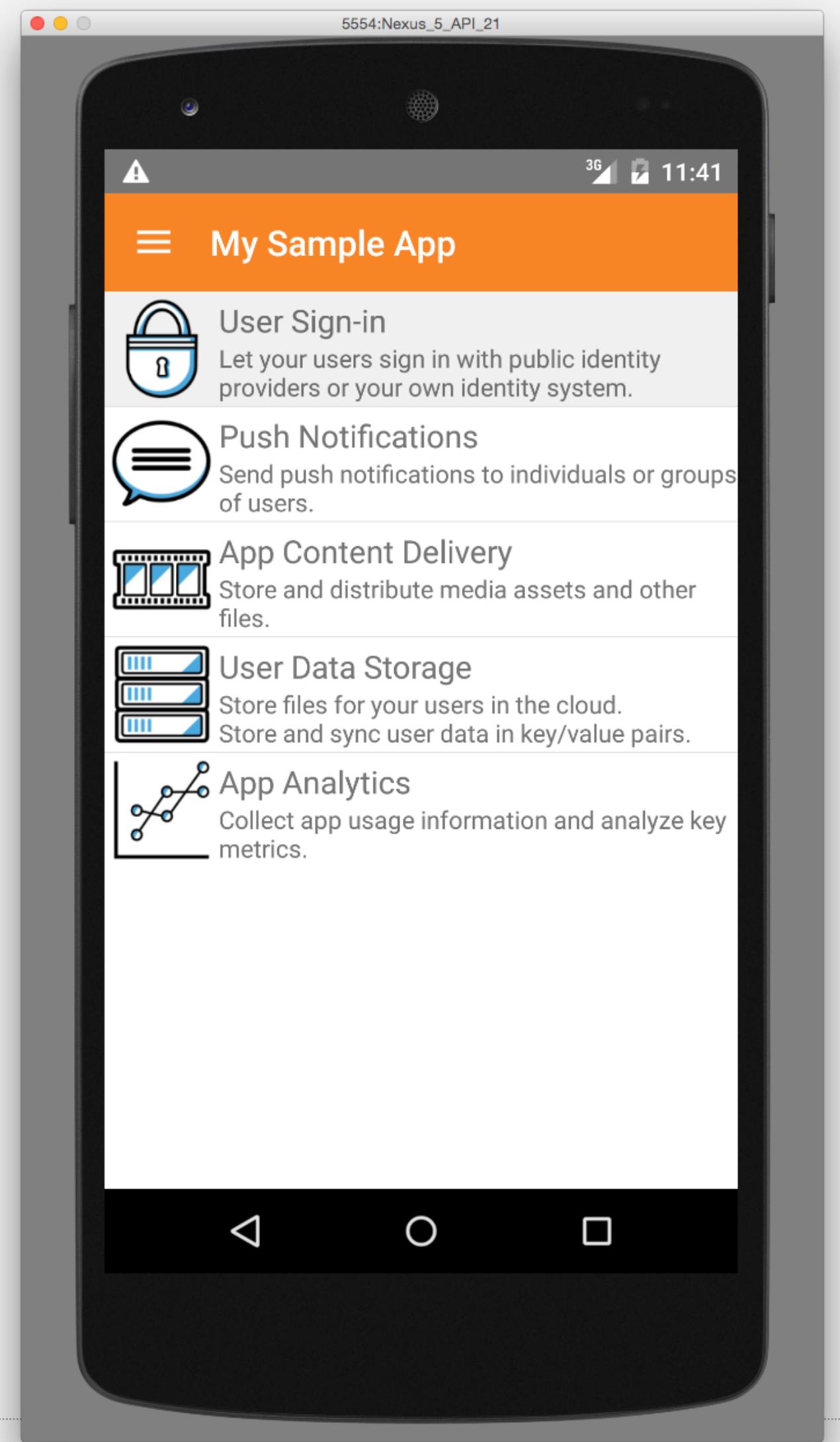
OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>

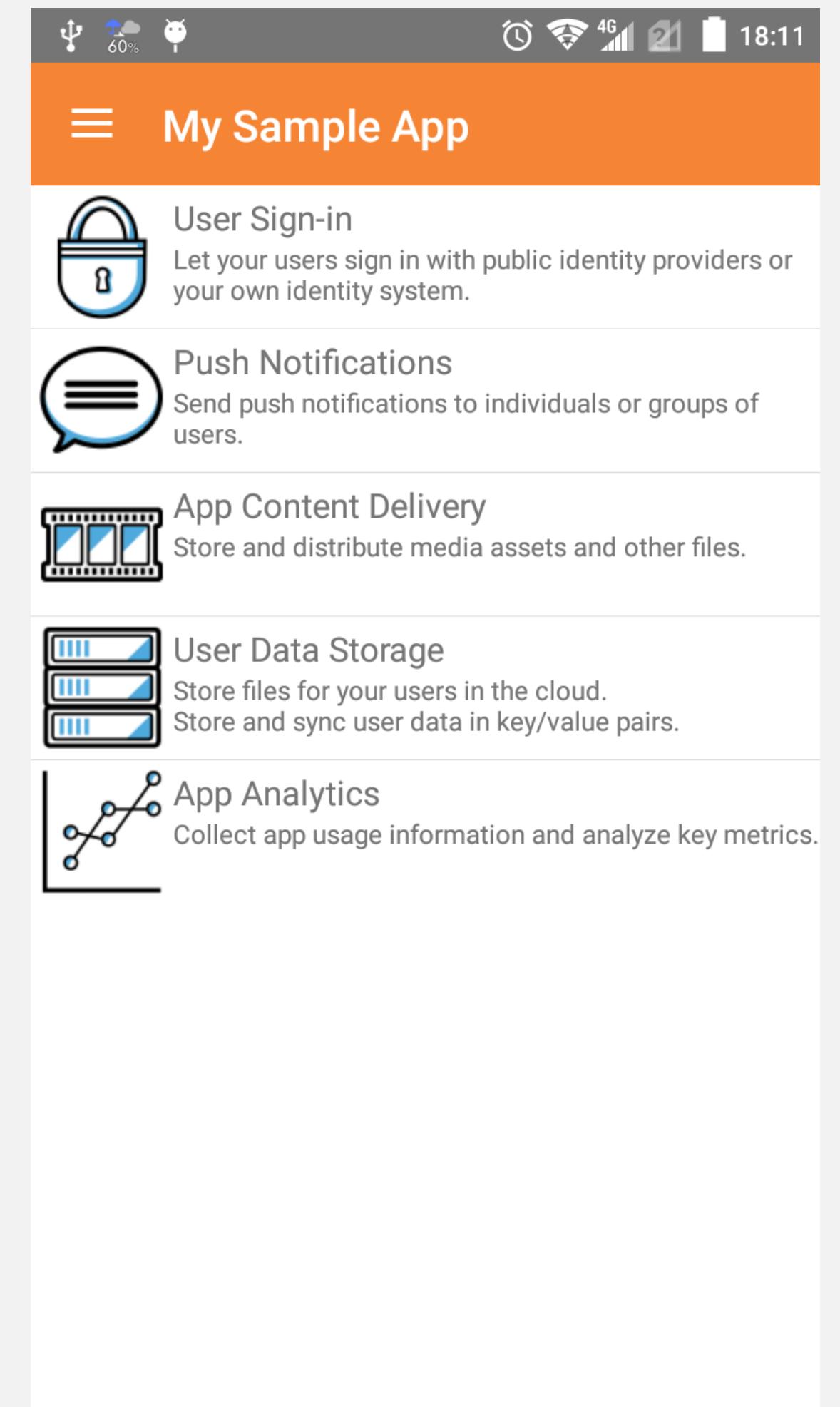
エミュレーターでのアプリの実行

- ◆ エミュレーターの場合には、**3-5分程度!**待ちます。
エミュレーター上でOSが起動した後、アプリが自動的に起動するまでしばらく待ってみてください。
- ◆ 左のような画面が出てきたら、アプリを操作することができます。
- ◆ 開発中はアプリを終了しても、エミュレーターを終了する必要はありません。



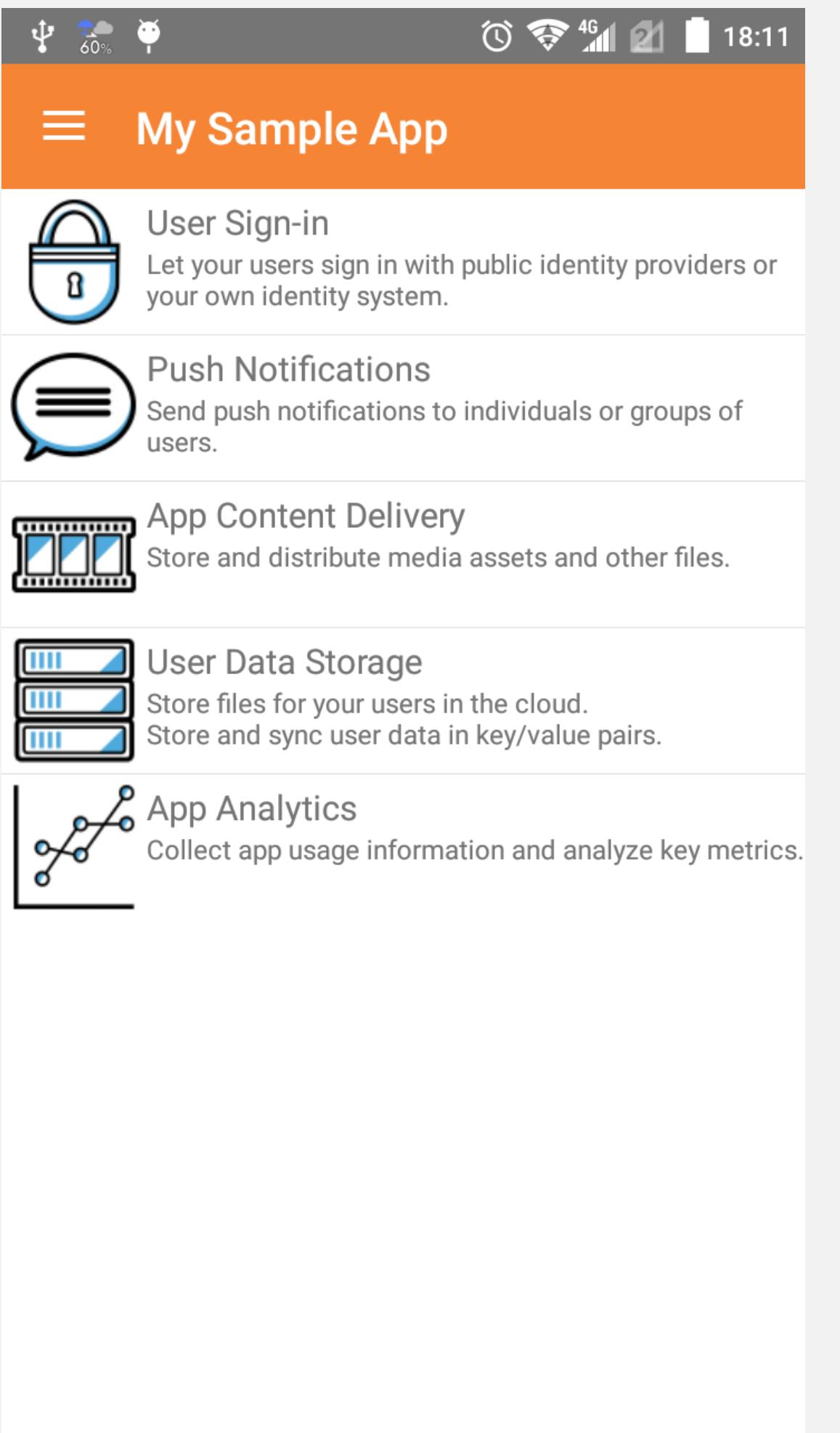
実機でのアプリの実行

- ◆ 実機での動作も、当然エミュレーターとほぼ同じ動きになります。
- ◆ 各メニューを色々いじってみてください。例えば「**App Content Delivery**」メニューでは、S3バケットの中身を直接見ることが出来ます。



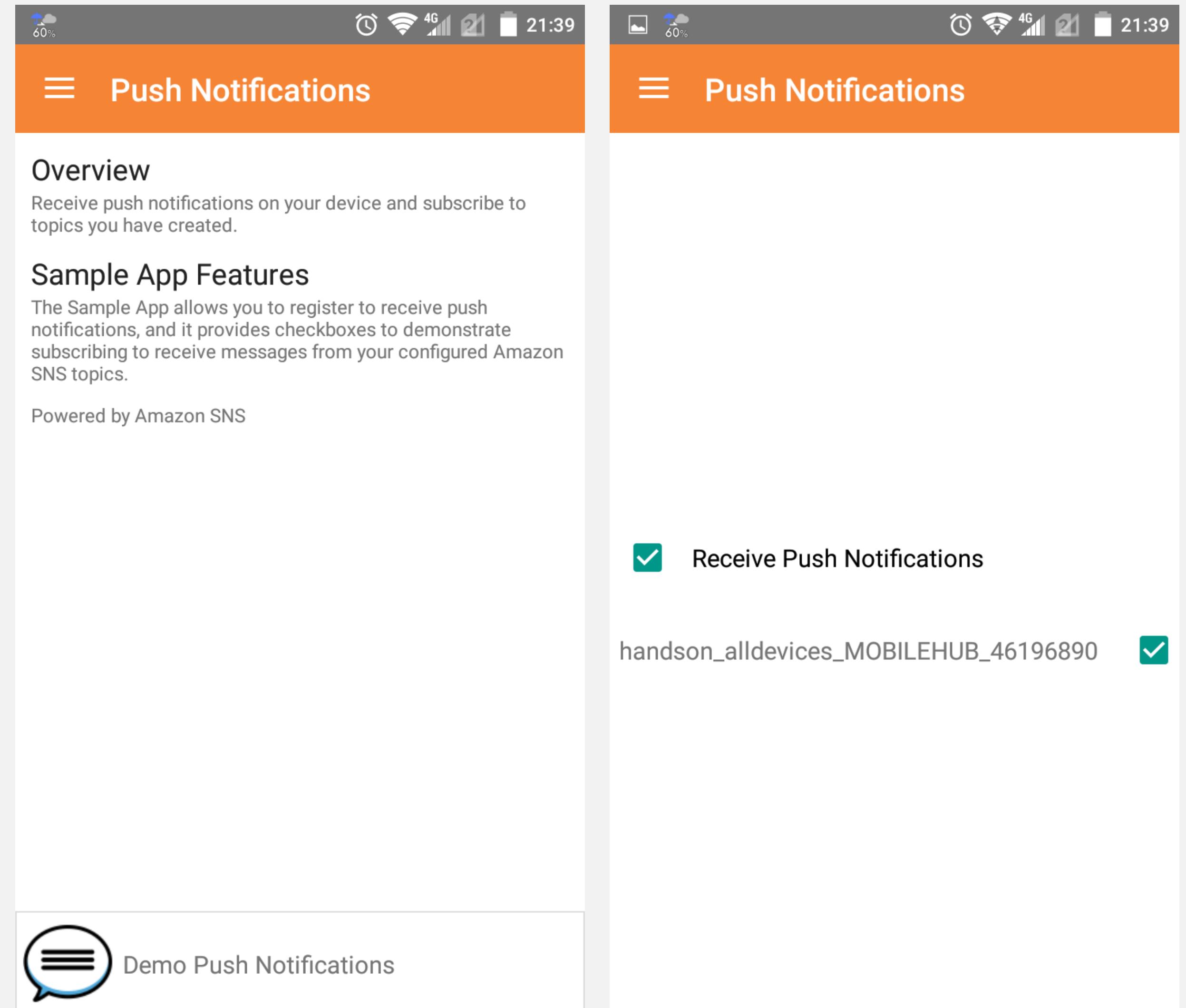
push通知の実行

- ◆ エミュレーターでは動かない（と思われる）動作を1つ試してみましょう。
- ◆ アプリのメニューから「**Push Notifications**」をタップしてください。



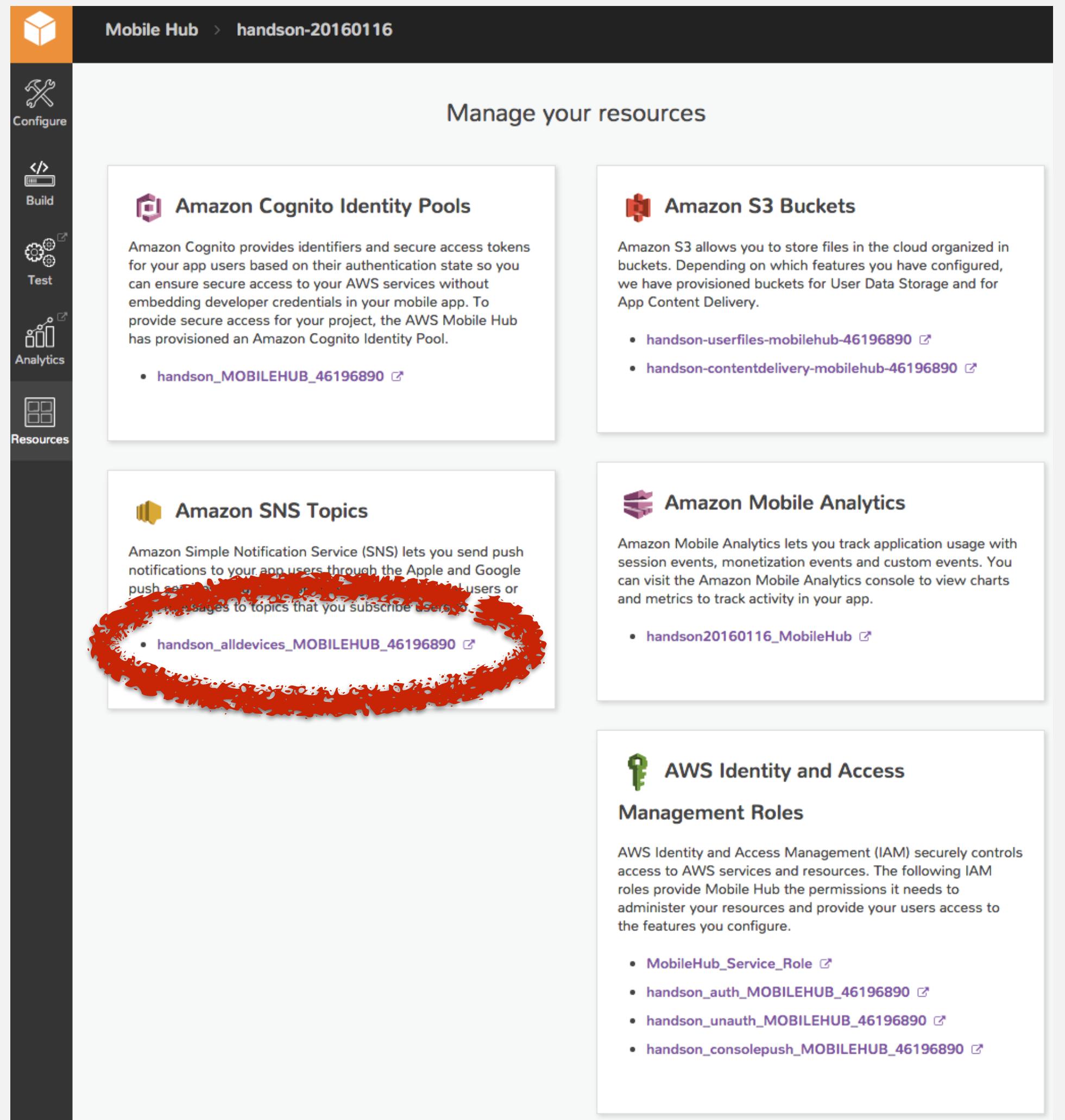
push通知の実行

- ◆ push通知の概要の画面下部の「Demo Push Notifications」をタップし、「Receive Push Notifications」にチェックが入っていることを確認します。



Mobile Hubが作成したAWS上のリソース

- ◆ ブラウザに戻り、**Mobile Hub**の左メニューの一番下のアイコン「**Resources**」をクリックします。
- ◆ ここに、**Mobile Hub**が自動的に作成した AWS上のリソースが一覧されています。
- ◆ **Amazon SNS Topics**の中の「**handson_alldevices_....**」をクリックしてください。



The screenshot shows the AWS Mobile Hub 'Resources' page for a project named 'handson-20160116'. The left sidebar includes icons for Configure, Build, Test, Analytics, and Resources. The main content area is titled 'Manage your resources' and lists several services:

- Amazon Cognito Identity Pools**: Provides identifiers and secure access tokens for app users.
- Amazon S3 Buckets**: Allows storing files in the cloud organized in buckets.
- Amazon SNS Topics**: Lets you send push notifications to app users through Apple and Google push services. A red oval highlights the link for 'handson_alldevices_MOBILEHUB_46196890'.
- Amazon Mobile Analytics**: Lets you track application usage with session events, monetization events, and custom events.
- AWS Identity and Access Management Roles**: Securely controls access to AWS services and resources.



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>



SNSトピック詳細

- ◆ 自動的に作成されたSNSトピックの詳細が開きます。
- ◆ Regionの項目を見ると、us-east-1リージョンに作成されていることが分かります。
- ◆ 左上の青いボタン「Publish to topic」をクリックしてください。

Topic Details: handson_alldevices_MOBILEHUB_46196890

Publish to topic Other topic actions ▾

Topic ARN	arn:aws:sns:us-east-1: XXXXXXXXXX :handson_alldevices_MOBILEHUB_46196890
Topic Owner	XXXXXXXXXX
Region	us-east-1
Display name	

Subscriptions

Create Subscription Request confirmations Confirm Subscription Other Subscription Actions ▾

Subscription ID	Protocol	Endpoint
arn:aws:sns:us-east-1: XXXXXXXXXX :handson_alldevices_MOBILEHUB_46196890:8a724...	application	arn:aws:sns:us-east-...
arn:aws:sns:us-east-1: XXXXXXXXXX :handson_alldevices_MOBILEHUB_46196890:fc79f0...	application	arn:aws:sns:us-east-...



SNSトピックからメッセージ送信

- ◆ **Message**部分に適当に文章を入力して、「Publish message」をクリックすると・・・
(Subjectは出ません)

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN: arn:aws:sns:us-east-1:XXXXXXXXXXXX:hanson_alldevices_MOBILEHUB_46196890

Subject:

Message format: Raw JSON

Message: こんにちは世界

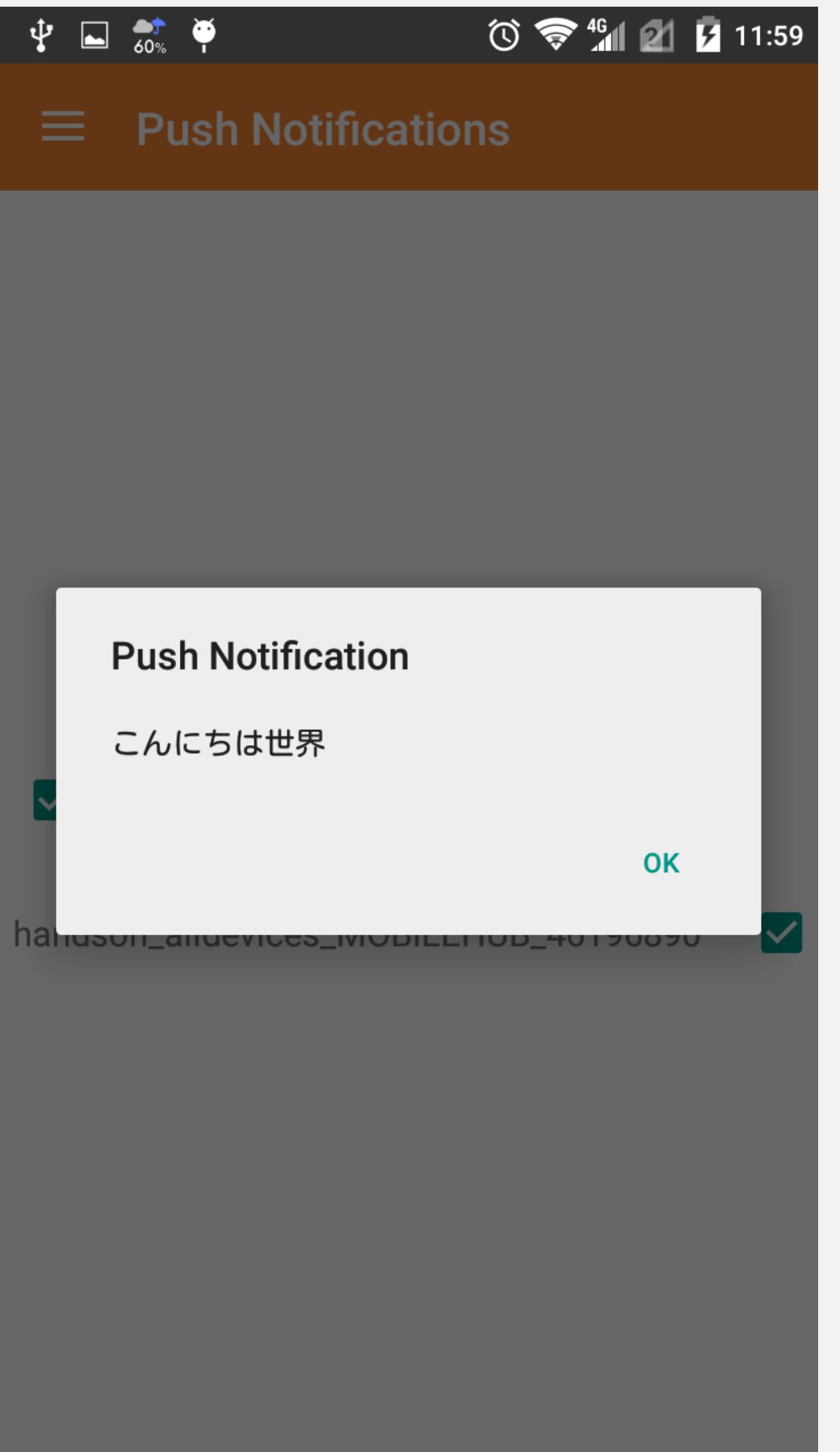
JSON message generator

Time to live (TTL):

Cancel **Publish message**

実機側で確認

- ◆ 簡単に実機側でクラウド側からの通知を受け取れることが確認できると思います。

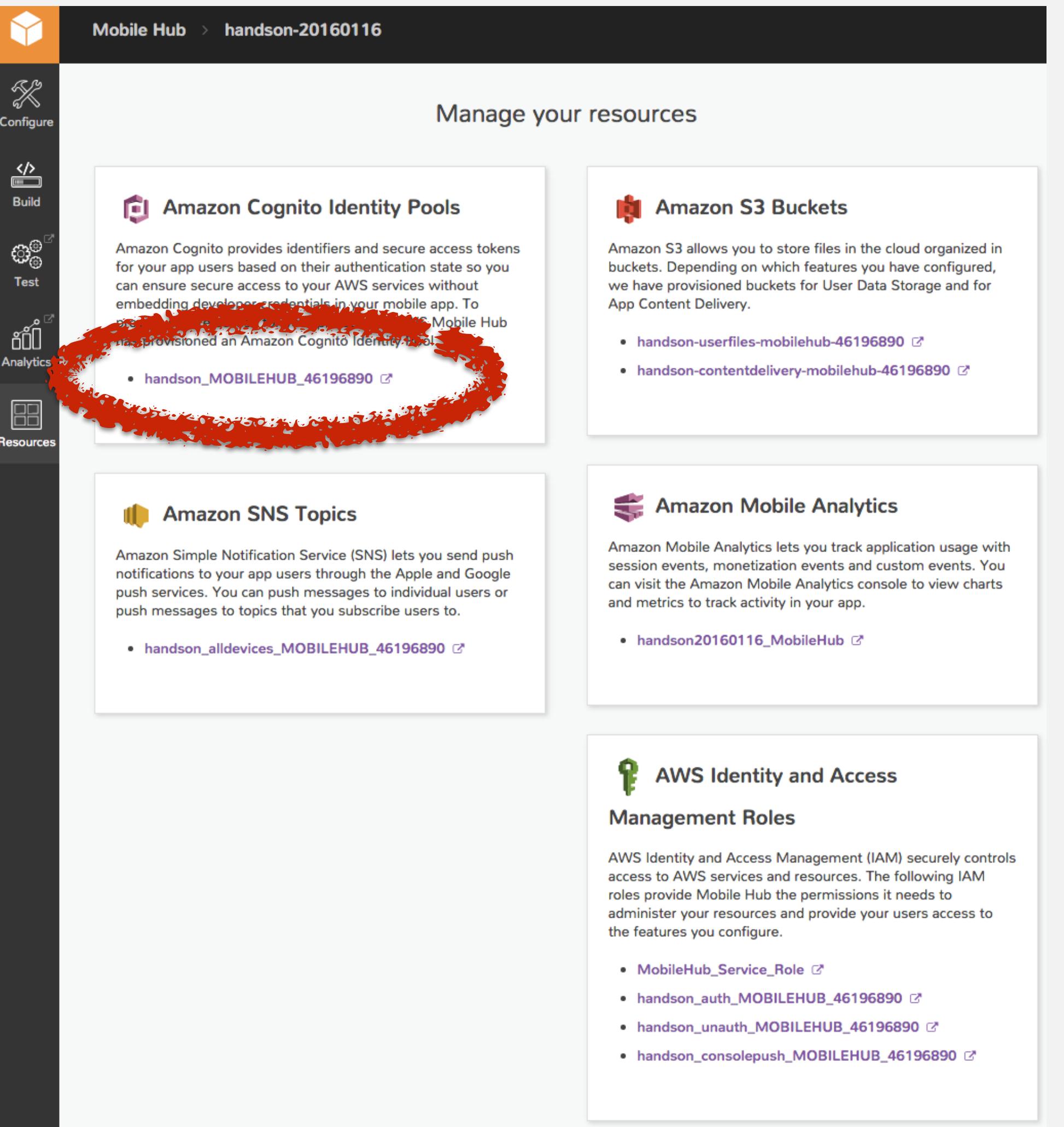


クラウド側に保存されているユーザ情報

◆ ブラウザに戻り、別の**Resource**を見てみましょう。

◆ 左上にある**Amazon Cognito Identity Pools**

Poolsの中の「**handson_MOBILEHUB_...**」をクリックしてください。



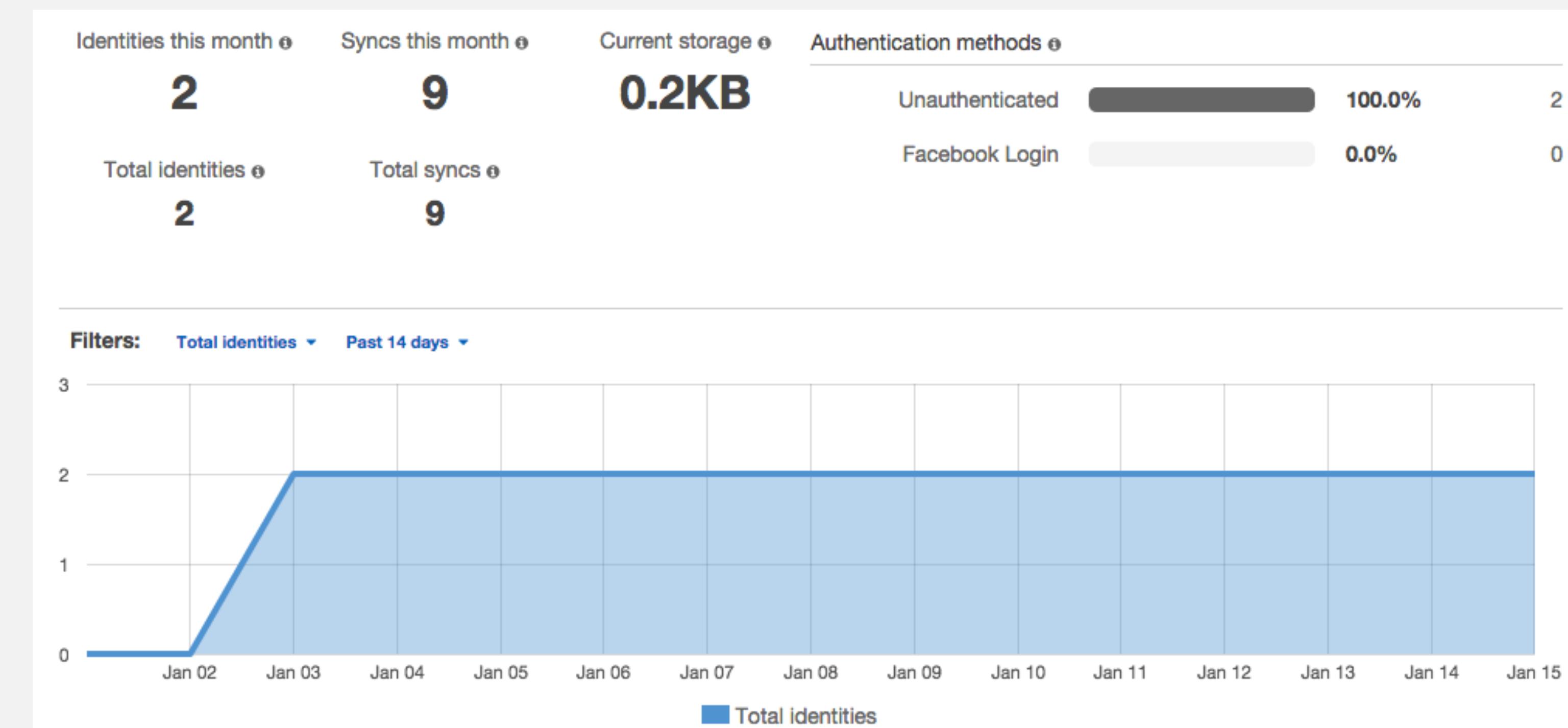
The screenshot shows the AWS Mobile Hub interface with the navigation bar "Mobile Hub > handson-20160116". On the left, a sidebar lists "Configure", "Build", "Test", "Analytics", and "Resources". The main area is titled "Manage your resources" and contains four cards:

- Amazon Cognito Identity Pools**: A card describing Amazon Cognito Identity Pools and listing one pool: "handson_MOBILEHUB_46196890". This item is circled in red.
- Amazon S3 Buckets**: A card describing Amazon S3 buckets and listing two buckets: "handson-userfiles-mobilehub-46196890" and "handson-contentdelivery-mobilehub-46196890".
- Amazon SNS Topics**: A card describing Amazon Simple Notification Service (SNS) topics and listing one topic: "handson_alldevices_MOBILEHUB_46196890".
- Amazon Mobile Analytics**: A card describing Amazon Mobile Analytics and listing one resource: "handson20160116_MobileHub".
- AWS Identity and Access Management Roles**: A card describing IAM roles and listing four roles: "MobileHub_Service_Role", "handson_auth_MOBILEHUB_46196890", "handson_unauth_MOBILEHUB_46196890", and "handson_consolepush_MOBILEHUB_46196890".



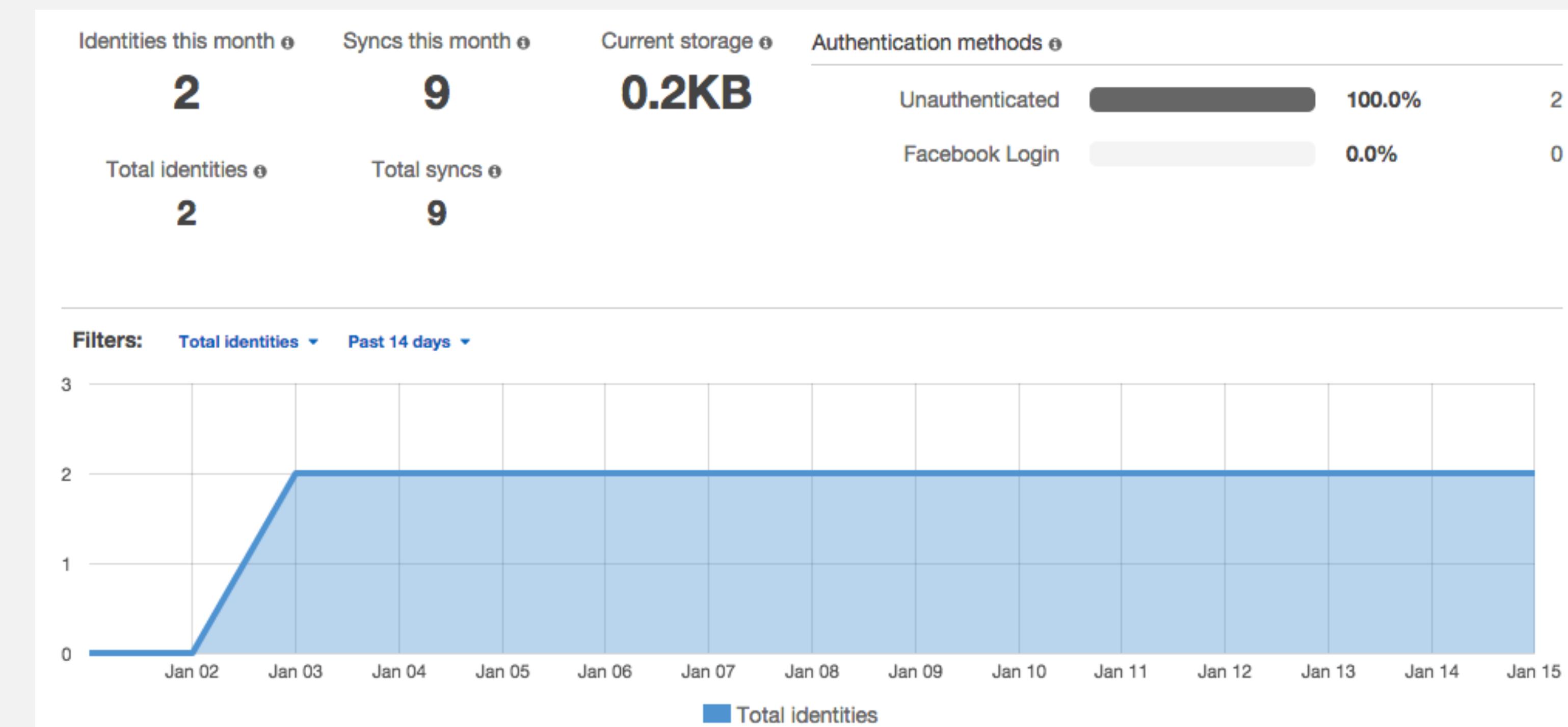
Cognitoのダッシュボード

- ◆ ここでは**Unauthenticated**ユーザが2人いることが分かります。
- ◆ 他にも、今月9回syncしたことやストレージのサイズが表示されています。



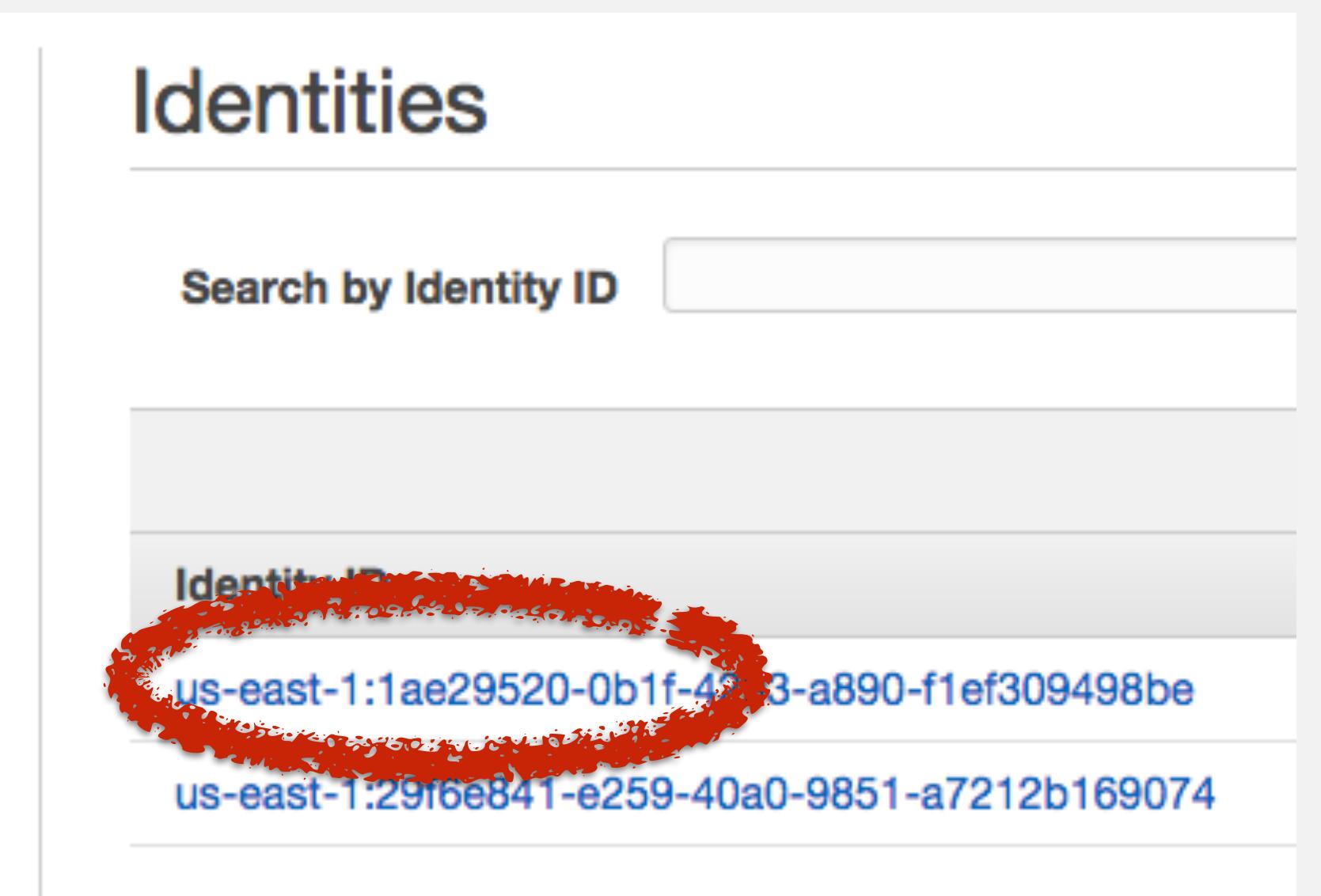
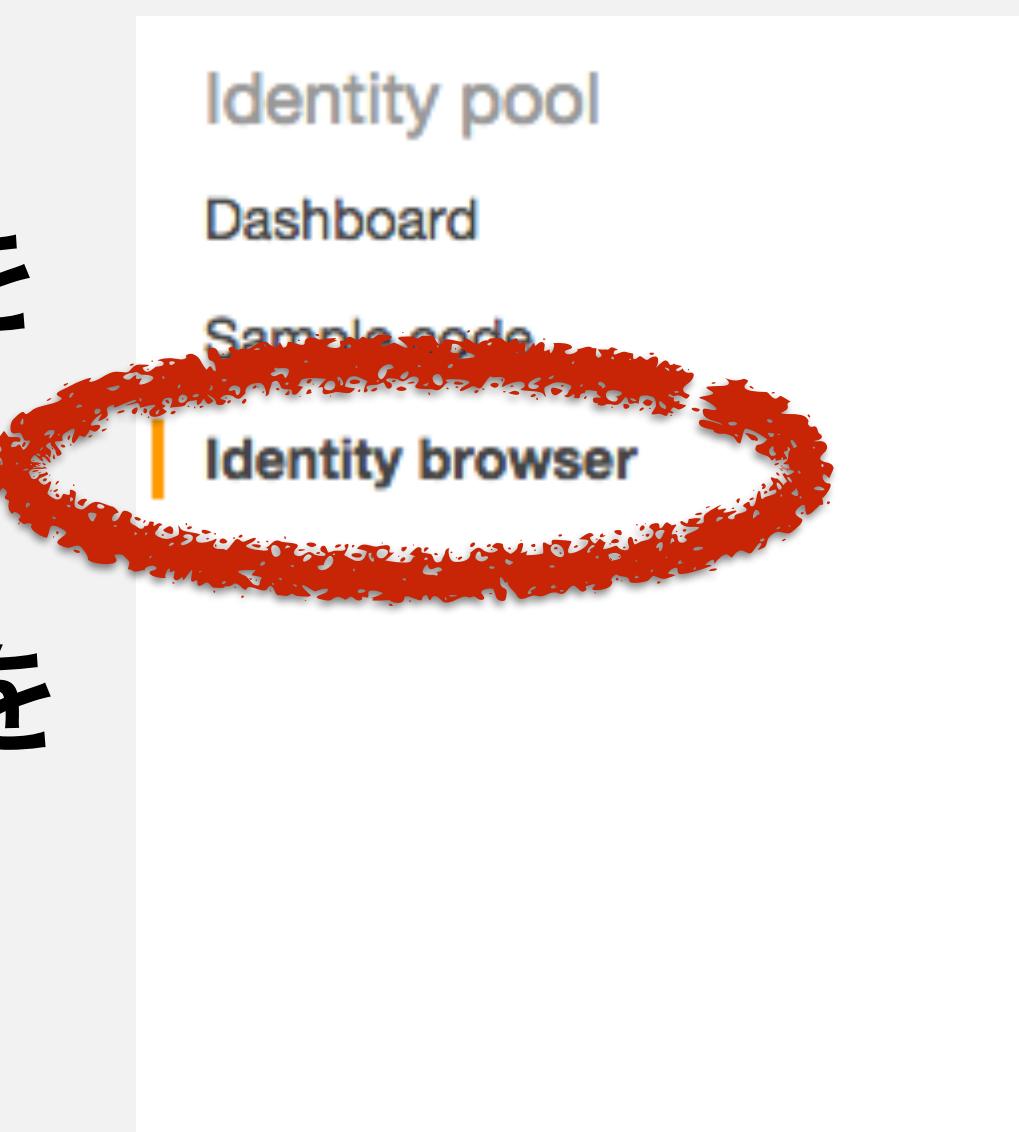
Cognitoのダッシュボード

- ◆ ここでは**Unauthenticated**
(いわゆる匿名ユーザのこと)
ユーザが2人いることが分かり
ます。
- ◆ 他にも、今月9回**sync**して
ることやストレージのサイズが
表示されています。



CognitoのIdentity browser

- ◆ 左のメニューで**Identity browser**をクリックします。
- ◆ 匿名ユーザのIDが出てるので、それをクリックしてみます。



Identity pool
Dashboard
Sample code
Identity browser

Identities

Search by Identity ID

Identity ID
us-east-1:1ae29520-0b1f-4333-a890-f1ef309498be
us-east-1:2916e841-e259-40a0-9851-a7212b169074

Cognitoが保存しているdataset

- ◆ このユーザが持っている**Dataset**の一覧が出てくるので、”**user_settings**”をクリックしてみます。
- ◆ **key=value**の形式でデータが保存されているのが見えると思います。
- ◆ このような形で、クラウド側にデータが保存され、閲覧したり書き換えたりすることが可能となっています。

Datasets		
	Create dataset	Delete selected

Dataset name Data Last modified

<input type="checkbox"/>	user_settings	2018-07-10 14:44:20
--------------------------	---------------	---------------------

Current dataset - user_settings		
	Create record	Delete selected
	Key	Value
<input type="checkbox"/>	background_color	-1
<input type="checkbox"/>	title_bar_color	-686795
<input type="checkbox"/>	title_text_color	-1

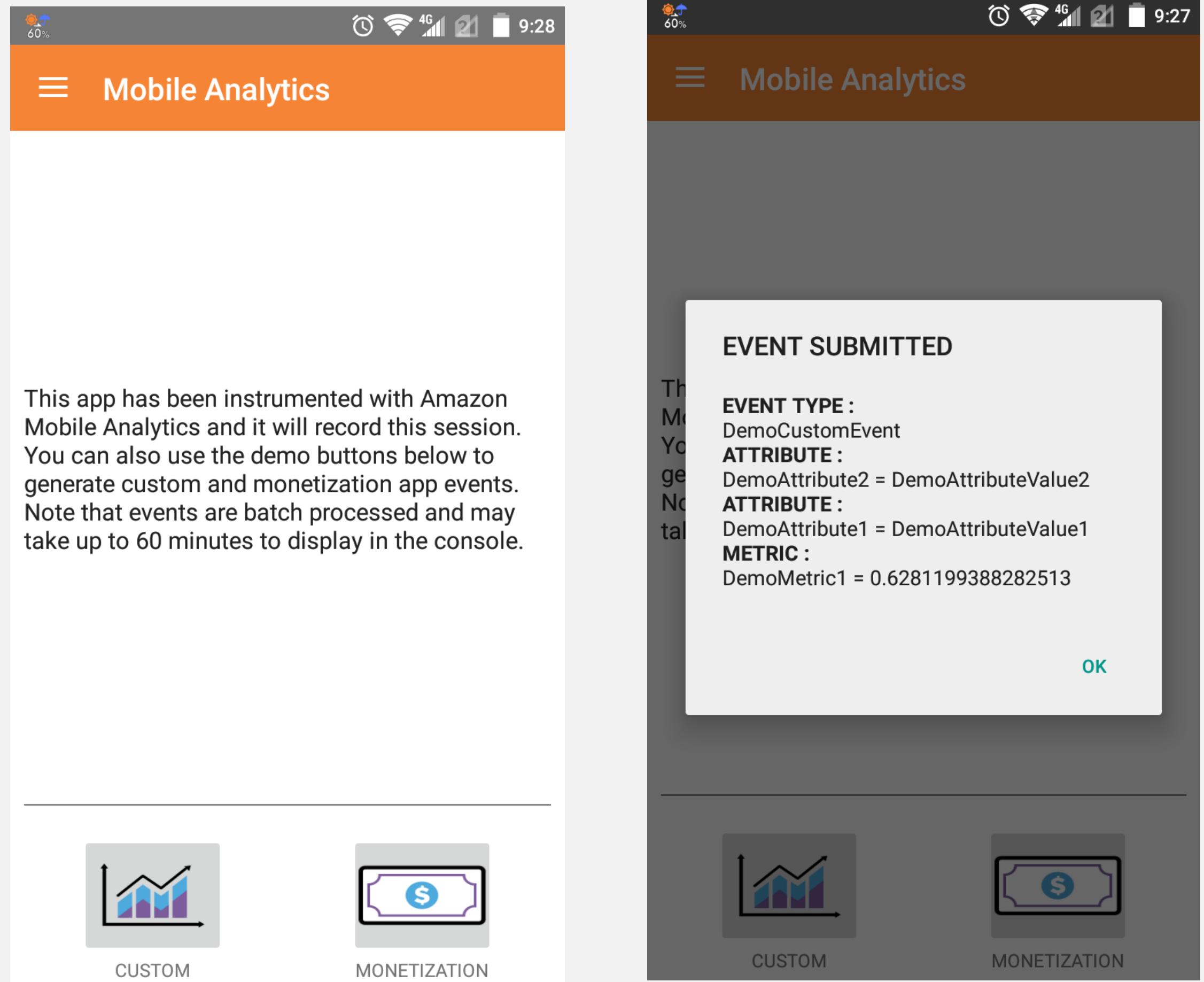
App Analyticsの実行

- ◆ 今度はアプリ側に戻ってみてください。
- ◆ App Analyticsのメニューも試してみましょう。

「**Demo App Analytics**」をタップし、左下の

「**CUSTOM**」ボタンをクリックしてみます。

- ◆ これは、アプリ側からクラウド側にカスタムイベントを発行した様子です。
- ◆ 何人のユーザがアプリをダウンロードしたか、どのボタンをクリックしたか、などの統計レポートがイベントごとに自動的に作成されます。



JAWS-UG

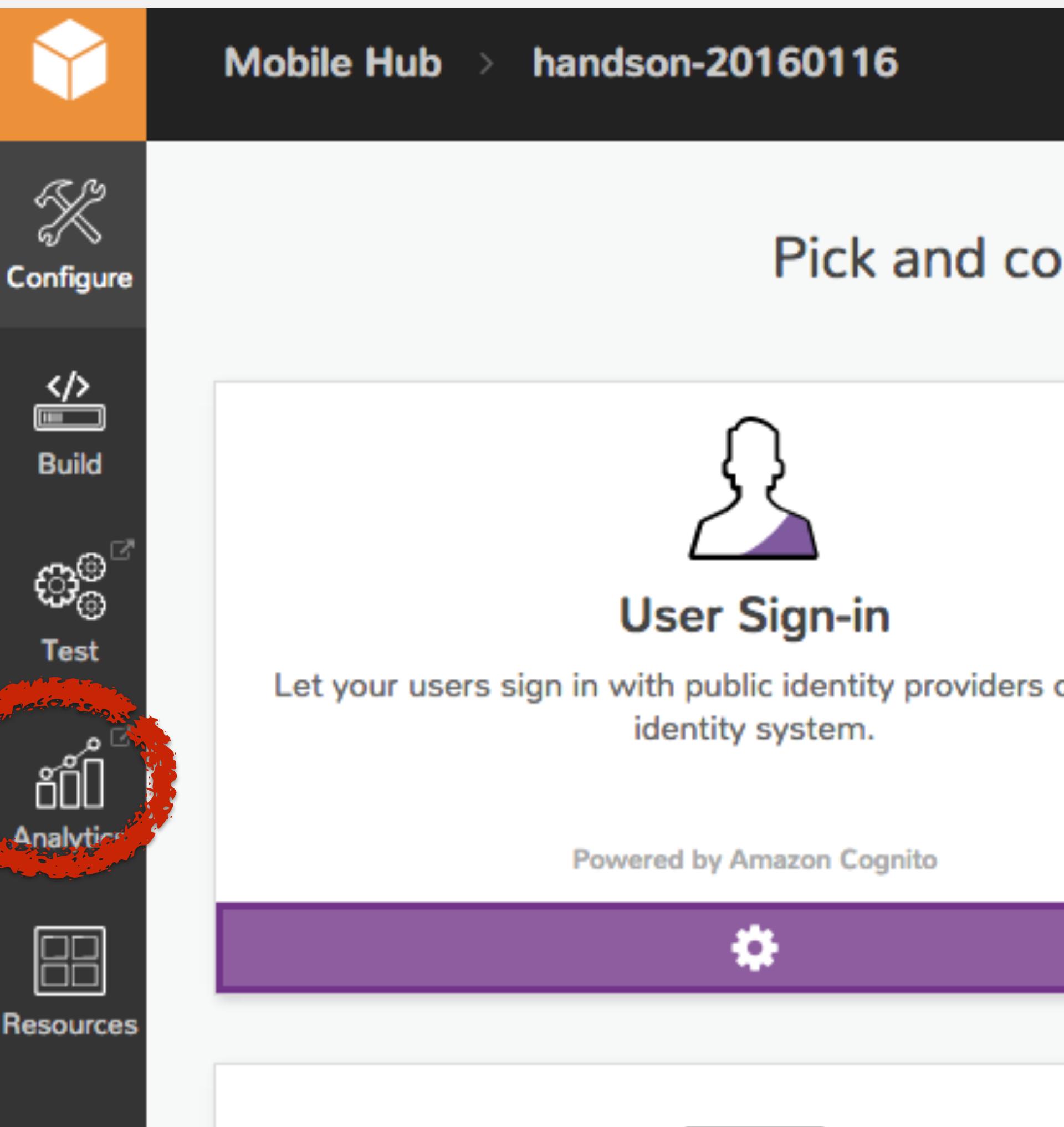
AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
GitHub repositories <https://github.com/jaws-ug/>

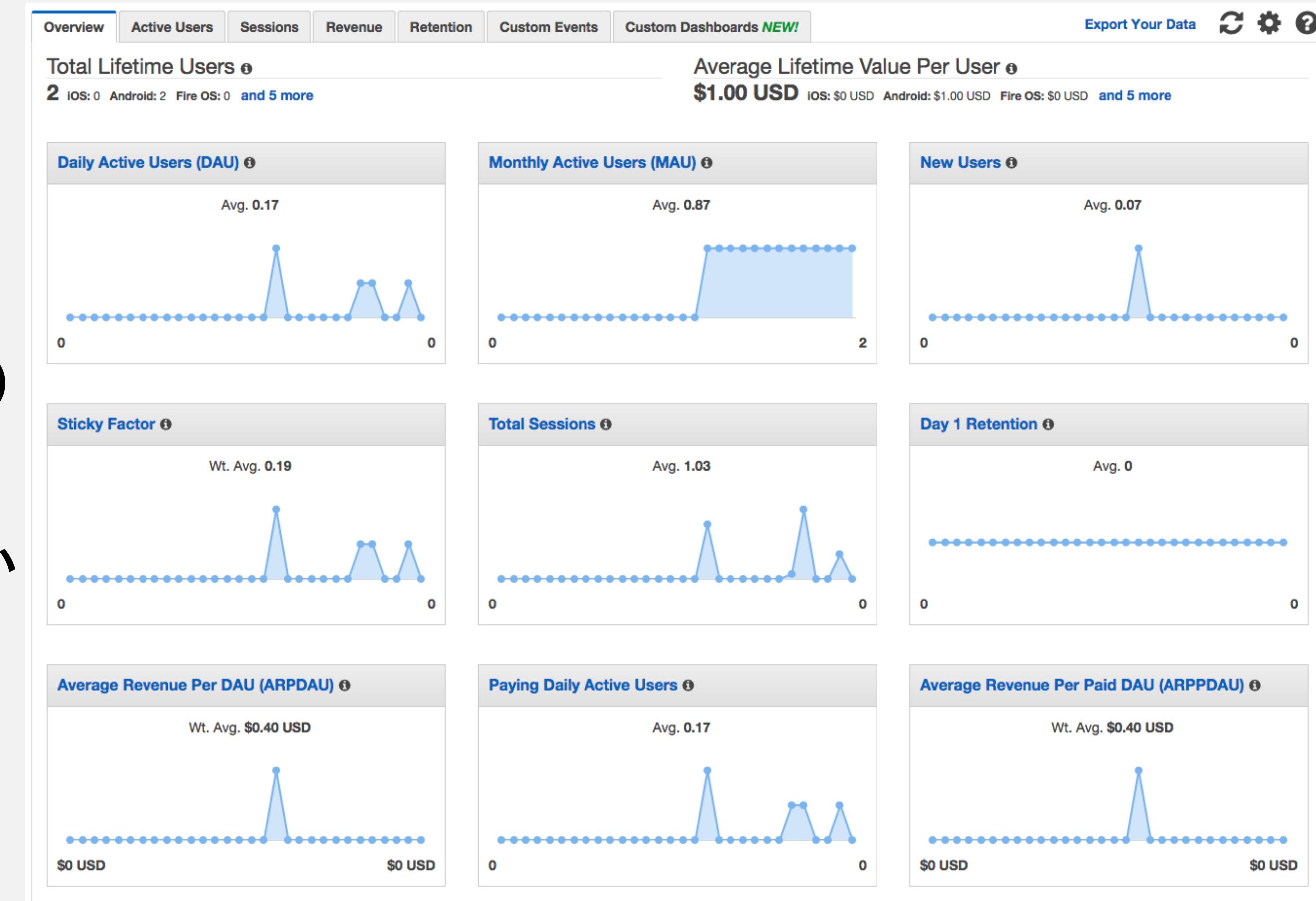
Mobile Analyticsのレポート

- ◆ 色々いじったら、アプリを一旦終了しておきます。
- ◆ ブラウザに戻って、**Mobile Hub**ページの左下の**Analytics**をクリックします。



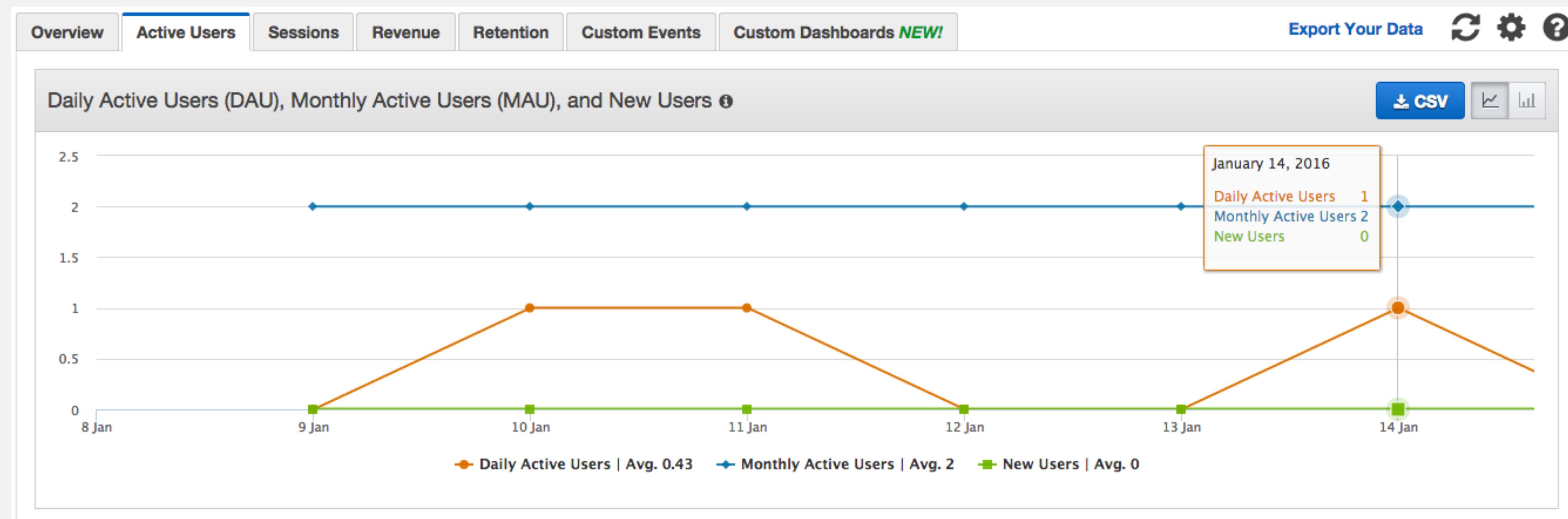
Mobile Analyticsのレポート

- ◆ 今は色々出て「いない」！と思います。
(レポートはリアルタイムではないので(^_^;)
- ◆ 30分もしないうちに、出てくると思います。
- ◆ 以下参考までに、どんなレポートが見えるのか
貼っておきます。





Mobile Analyticsのレポート

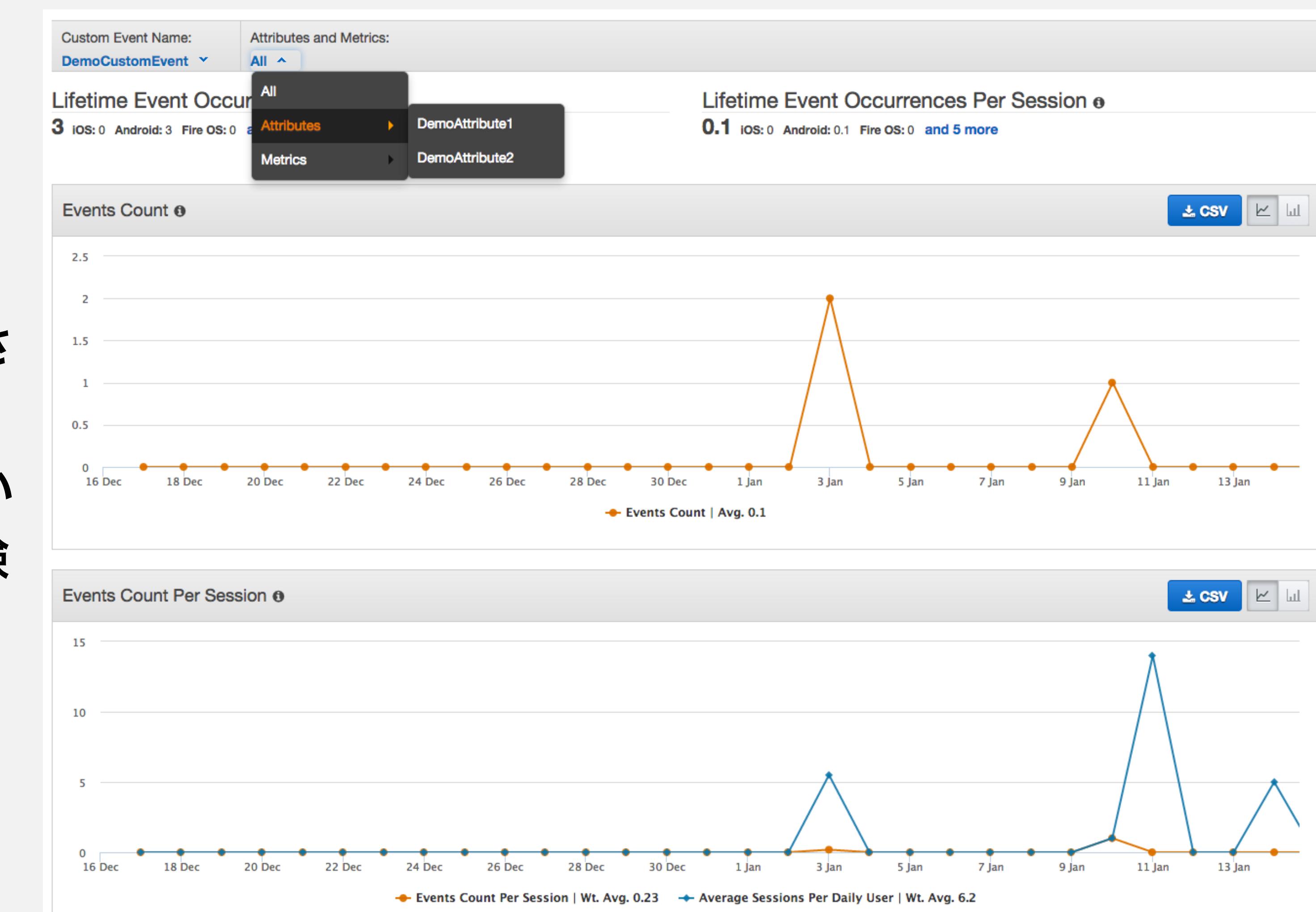


- ♦ アクティブユーザーはこんな感じ
- ♦ マンスリーとデイリーが出てます



Mobile Analyticsのレポート

- ◆ 先ほど送ったカスタムイベントの例
- ◆ 項目ごと、もしくはまとめてグラフ化されます。
- ◆ 上司に「実際アプリはどこがどのくらい使われてるんだ？」って詰められた経験のある方に、朗報ですw

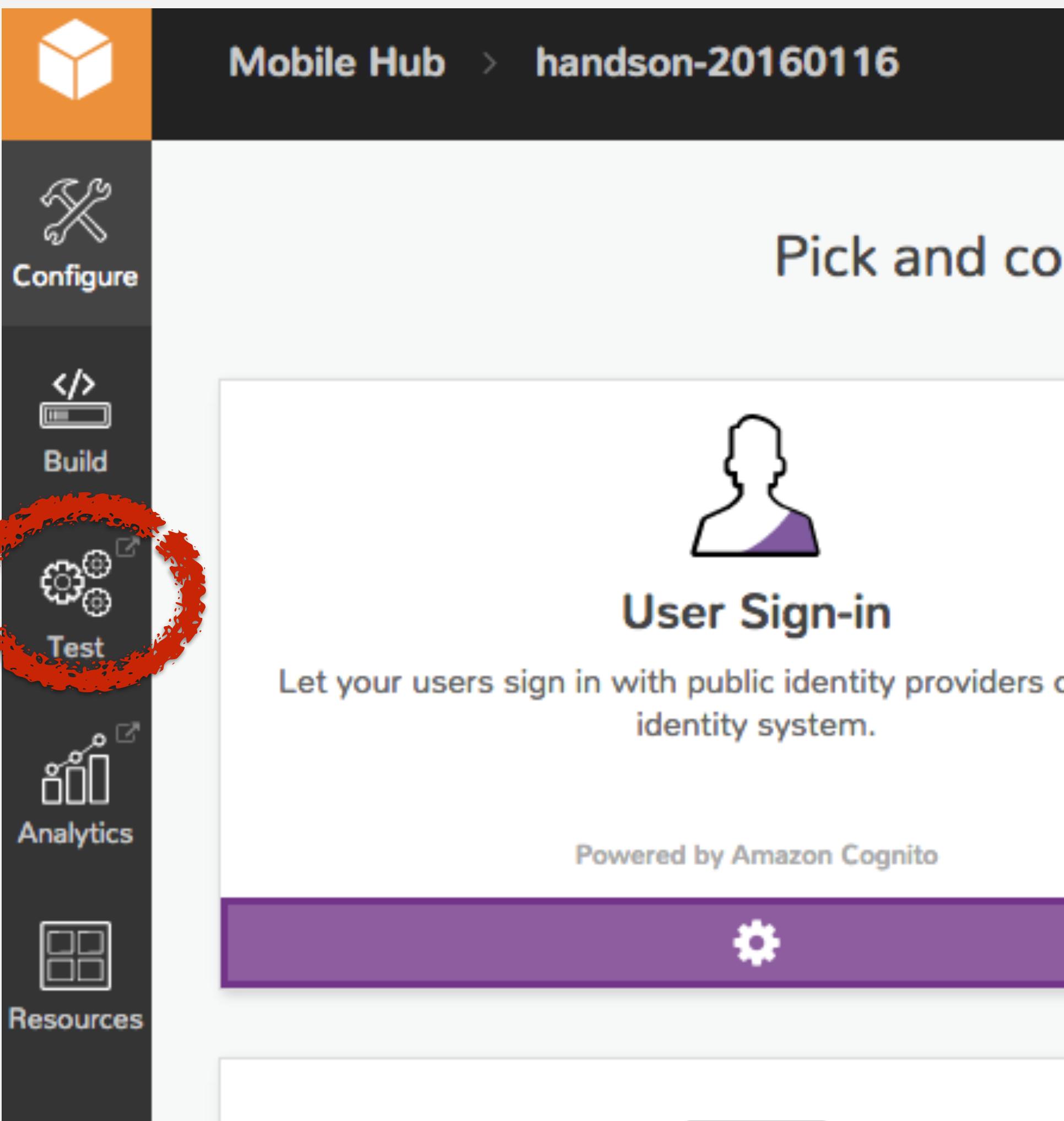

JAWS-UG

AWS User Group - Japan

OKINAWA
<http://jaws-ug.jp/>
 GitHub repositories <https://github.com/jaws-ug/>

クラウドでテストも出来る

- ◆ 今回は触れませんが、「Test」というメニューをクリックすると「AWS Device Farm」というサービスにリンクします。
- ◆ これを使うと、クラウド上で数あるAndroid / iOS端末を選んで、テストを走らせることができます。
- ◆ 詳細：<http://aws.amazon.com/jp/device-farm/>





この辺で
ひとやすみ

お楽しみはこれからだ！

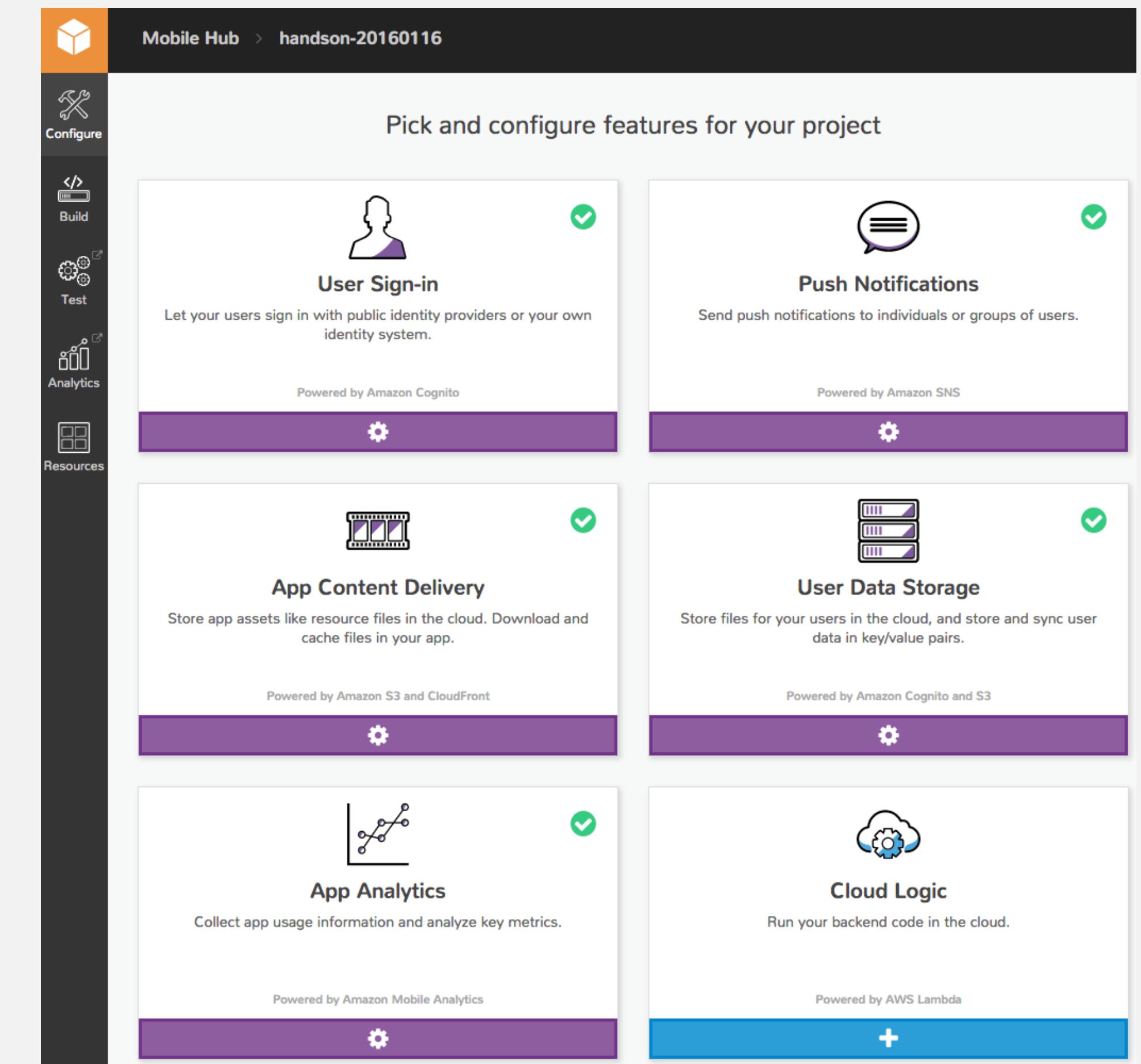
◆ Cloud Logicとは？

= AWS Lambda をアプリから

直接呼び出せる機能

◆ Mobile Hubのメニューから右下

のCloud Logicをクリック



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>

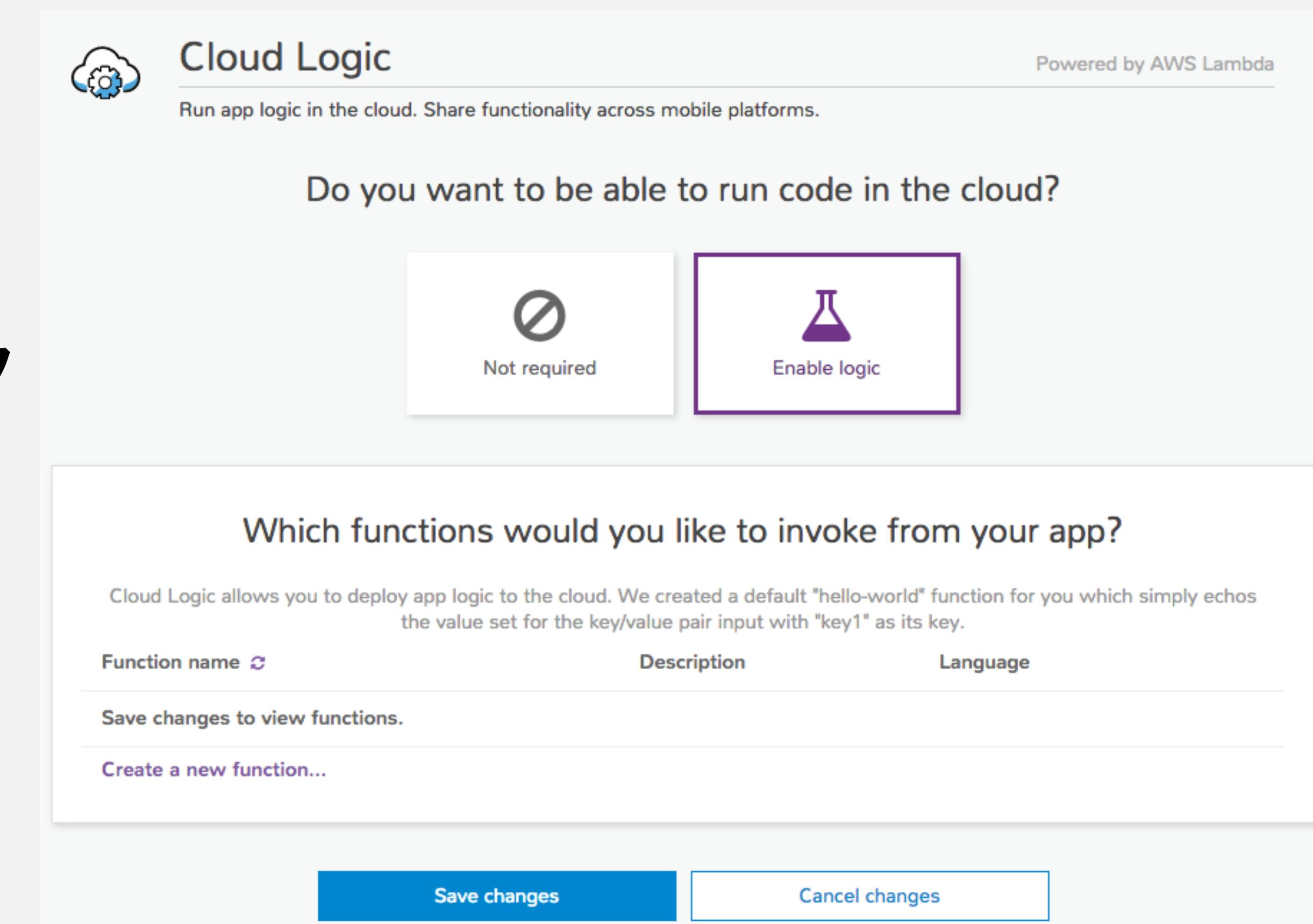
ちょっと待った！ AWS Lambda (らむだ) って何？

- ◆ クラウド上 (**AWS**が運用管理するサーバ上) で、自分の書いたコードが実行されるサービス
- ◆ 言語は現在のところ**Node.js (JavaScript)**、**Python**、および**Java8**をサポート
- ◆ 「サーバレスアーキテクチャ」はマーケティング用語なので、実際にはマネージドサービスの上で動く仕組みなんだ、というのを理解しておくと良いと思います
- ◆ 詳細：<https://aws.amazon.com/jp/lambda/details/>



Cloud Logic の設定

- ❖ 「Enable Logic」を選択。
- ❖ 「Create a new function」をクリックすると、ラムダ関数作成ウィザードが始まってしまうので、グッとこらえて次頁に進みましょう。



Cloud Logic

Run app logic in the cloud. Share functionality across mobile platforms.

Do you want to be able to run code in the cloud?

Not required Enable logic

Which functions would you like to invoke from your app?

Cloud Logic allows you to deploy app logic to the cloud. We created a default "hello-world" function for you which simply echos the value set for the key/value pair input with "key1" as its key.

Function name	Description	Language
hello-world		JavaScript

Save changes to view functions.

Create a new function...

Save changes Cancel changes



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>
 GitHub repositories <https://github.com/jaws-ug/>



今回のハンズオンで 用意したLambda関数について

- ◆ コードは**Node.js**です。ソースコードは**GitHub**にあります。

<https://github.com/jaws-ug/hands-on/tree/master/>

Mobile-Hub

- ◆ サンプルとして以下を用意しました。

以下より**ZIP**ファイルをダウンロードしてください。

Safariユーザーは**ZIP**自動解凍機能に注意！ゴミ箱に元**ZIP**が入っていますので、それを戻して使ってください。

- ◆ **Twitter**に投稿する
- ◆ **天気予報**を取得する





(全体共通)

Lambda関数のデプロイ手順

- ◆ 関数を作るリージョンを選択
- ◆ 関数の名前を決めて
- ◆ ランタイム（実行環境）を選択
- ◆ コードをフォームで書くもしくはファイルでアップロードする
- ◆ ロール（実行権限）を設定して
- ◆ 利用するメモリ量を設定





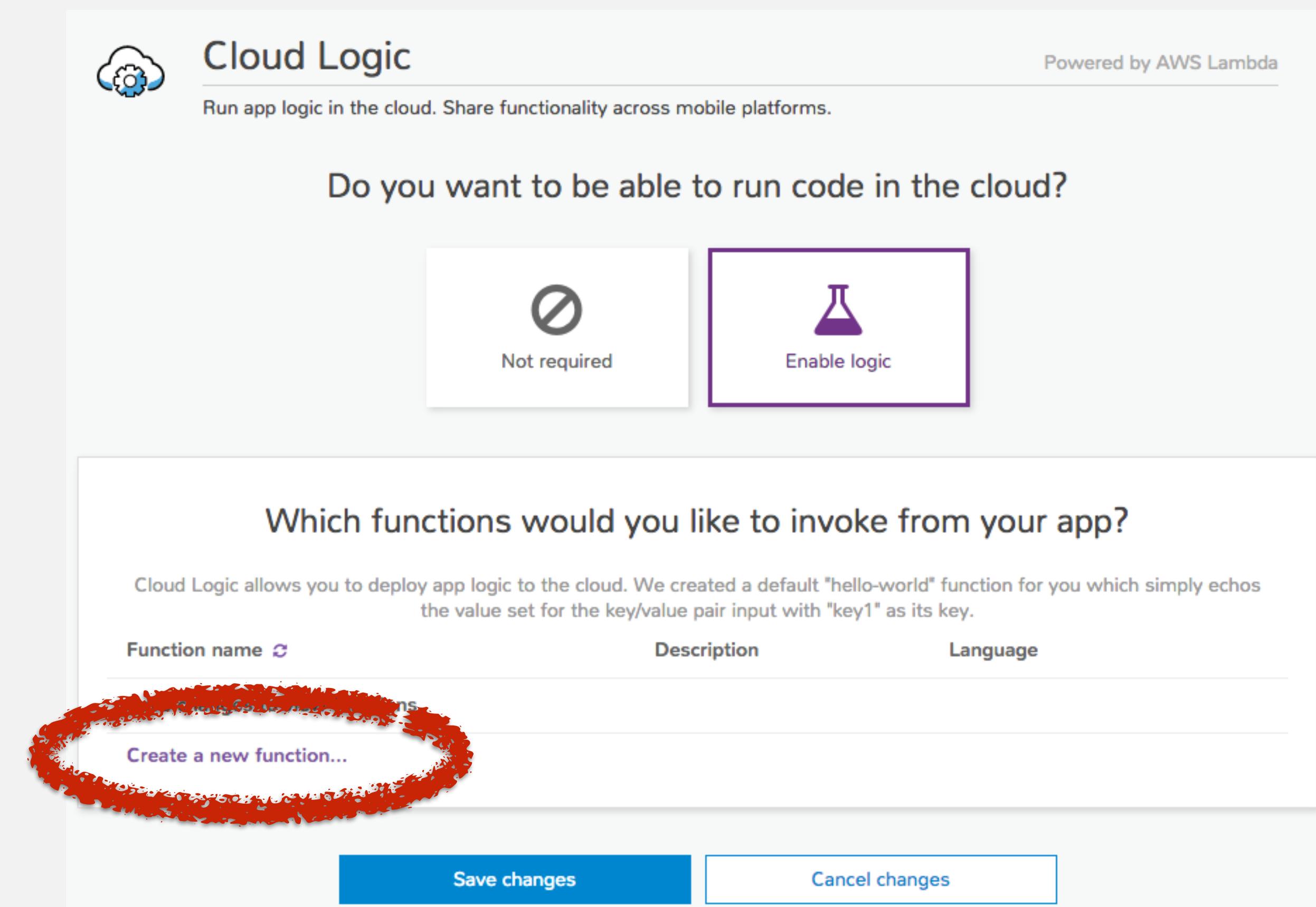
Lambdaサンプル

Twitter投稿：事前準備

- ♦ 本来はご自分のTwitterアカウントで <https://apps.twitter.com/> にアクセスして「Create New App」して、**consumer_key**・**consumer_secret**などを取得しておく必要があります。
- ♦ 本**lambdaTweet.zip**には @handson201601 アカウントの **consumer_key**などが予め含まれています
「悪用厳禁！」

Cloud Logic の設定画面から

- ◆ では先程の画面で
「Create a new function」 を
クリックします。
- ◆ 別ウィンドウで開くので、
ポップアップ禁止設定をしてある
ブラウザは注意。



Lambdaを作るリージョンの確認

♦ MobileHubの仕様で、

現在のところバージニアにある

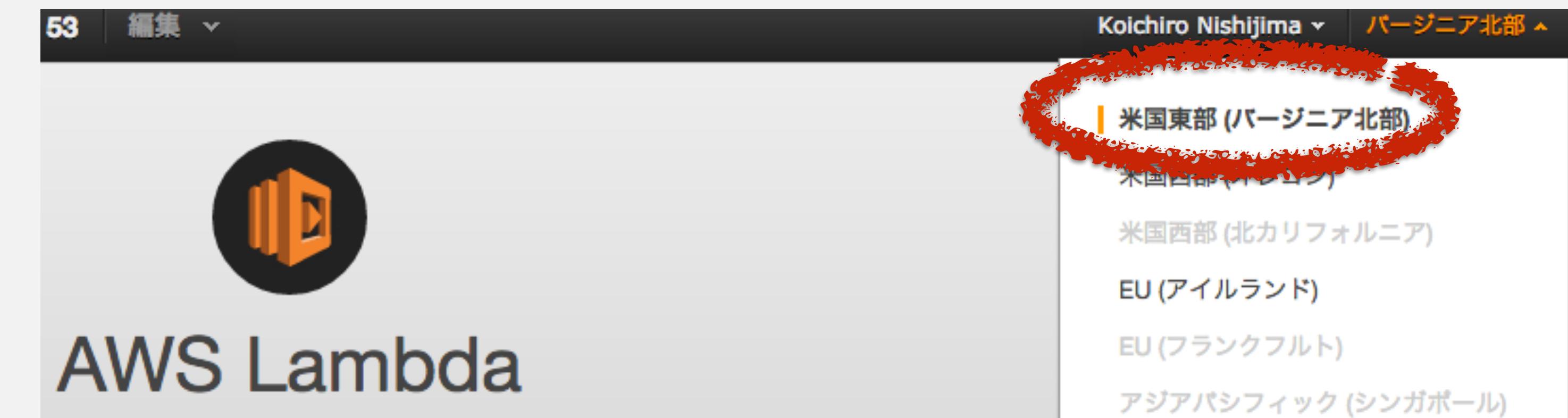
関数しか選択できないので、

右上のリージョン選択が

「バージニア北部」

となっているかを確認します

(なって無ければ「バージニア北部」を選択して変更)



Lambda関数の作成

◆ ブループリントには色々と興味深いものがあるが、
今回は「Skip」をクリック

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or skip this step if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under CC0.

<input type="button" value="Filter"/> <input type="button" value="All languages"/> << < Viewing 1-9 of 35 > >		
s3-get-object-python An Amazon S3 trigger that retrieves metadata for the object that has been updated. <small>python2.7 · s3</small>	config-rule-change-triggered An AWS Config rule that is triggered by configuration changes to EC2 instances. Checks instance types. <small>nodejs · config</small>	dynamodb-process-stream An Amazon DynamoDB trigger that logs the updates made to a table. <small>nodejs · dynamodb</small>
microservice-http-endpoint A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway. <small>nodejs · api-gateway</small>	node-exec Demonstrates running an external process using the Node.js child_process module. <small>nodejs</small>	slack-echo-command-python A function that handles a Slack slash command and echoes the details back to the user. <small>python2.7 · api-gateway · slack</small>
simple-mobile-backend A simple mobile backend (read/write to DynamoDB). <small>nodejs · mobile</small>	kinesis-process-record-python An Amazon Kinesis stream processor that logs the data being published. <small>python2.7 · kinesis</small>	dynamodb-process-stream-p... An Amazon DynamoDB trigger that logs the updates made to a table. <small>python2.7 · dynamodb</small>

Lambda関数の作成

- ◆ **Nameは「lambdaTweet」**
- ◆ **RuntimeはNode.js**
- ◆ **Code entry typeは「Upload a .ZIP file」を選択し、
「Upload」をクリック**

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name*	lambdaTweet
Description	
Runtime*	Node.js

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk) libraries, you can upload your code and libraries as a .ZIP file. [Learn more](#) about deploying Lambda functions.

Code entry type	<input type="radio"/> Edit code inline <input checked="" type="radio"/> Upload a .ZIP file <input type="radio"/> Upload a .ZIP from Amazon S3
-----------------	---

For .ZIP files larger than 10 MB, consider uploading via S3.

Upload

You have chosen to upload a .zip file but have not selected a file yet.



JAWS-UG

AWS User Group - Japan

OKINAWA

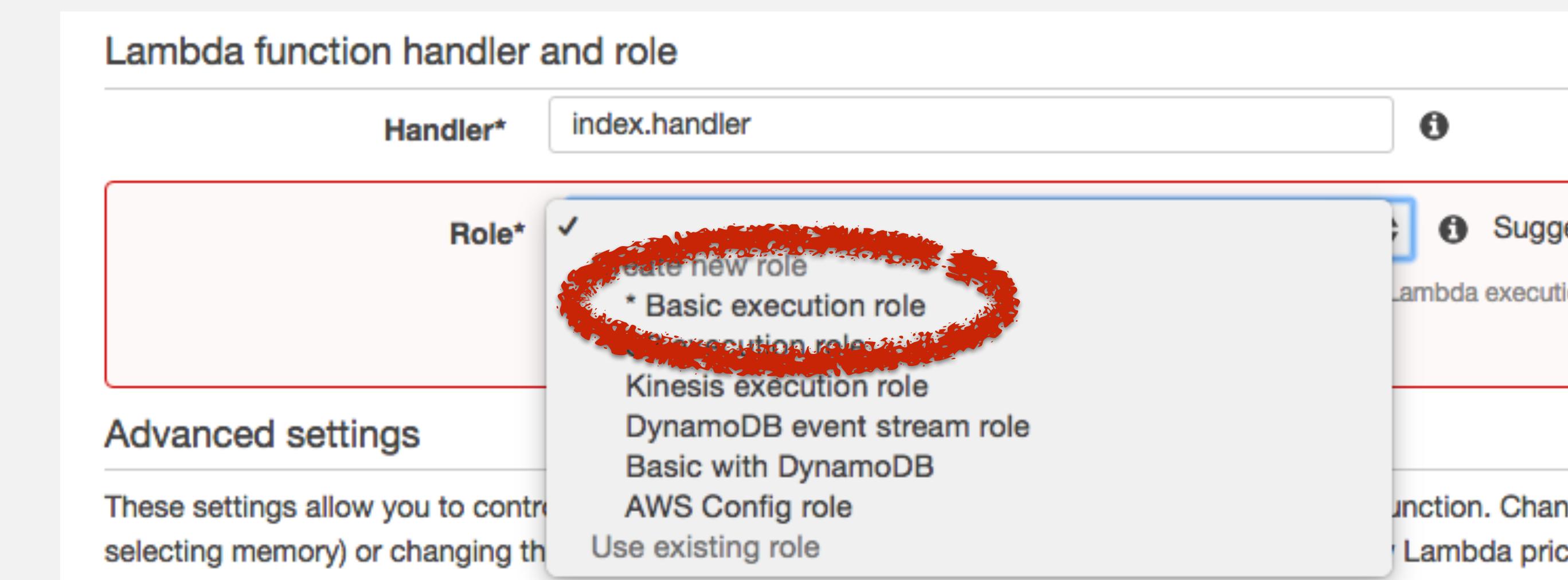
<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>



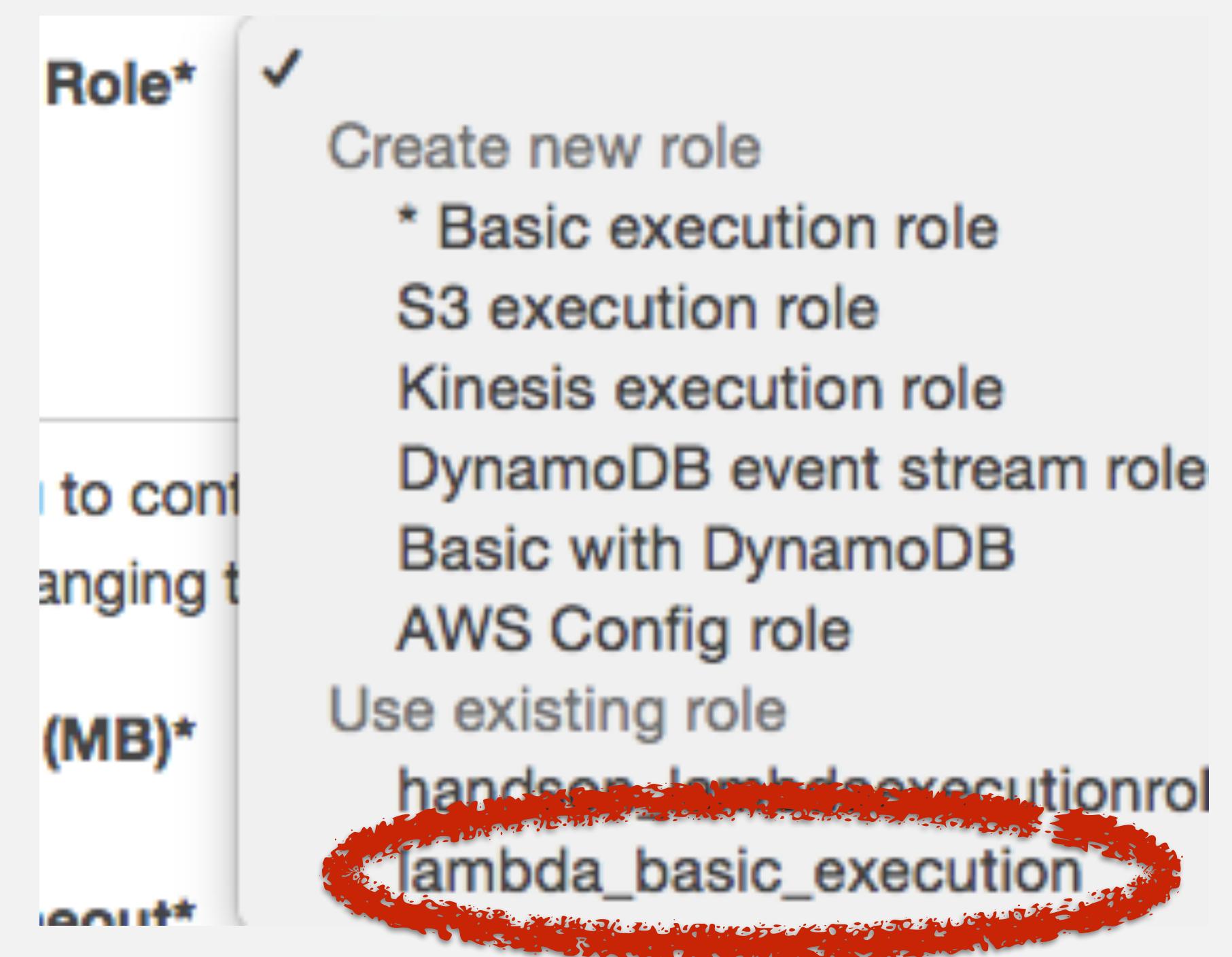
Lambda関数の作成

- ◆ Handlerはそのまま「**index.handler**」
- ◆ RoleはCreate new roleの中の「**Basic execution role**」を選ぶと、別ウィンドウが開く（初めての設定の場合）



Lambda関数の作成

- ◆ 2回目以降は
Use existing roleの中から
「**lambda_basic_execution**」を
選択するだけ



IAM roleの作成

- ◆ 適当にロール名をつけてくれるので、右下の「許可」をクリックします。
- ◆ これはこのLambda関数が実行されるときに利用する権限の設定になるので、例えばS3に書き込むとか、EC2インスタンスを起動するとか、必要な場合には権限を付与する必要があります。
- ◆ その場合は適当なロール名ではなくて、権限を表すロール名を付けておかないと混乱するので注意。

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ 詳細を非表示

ロールの概要

ロールの説明 Lambda execution role permissions

IAM ロール 新しい IAM ロールの作成

ロール名 lambda_basic_execution

▼ ポリシードキュメントを非表示

[編集](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs>CreateLogGroup",
        "logs>CreateLogStream",
        "logs>PutLogEvents"
      ],
      ...
    }
  ]
}
```

許可




JAWS-UG

AWS User Group - Japan

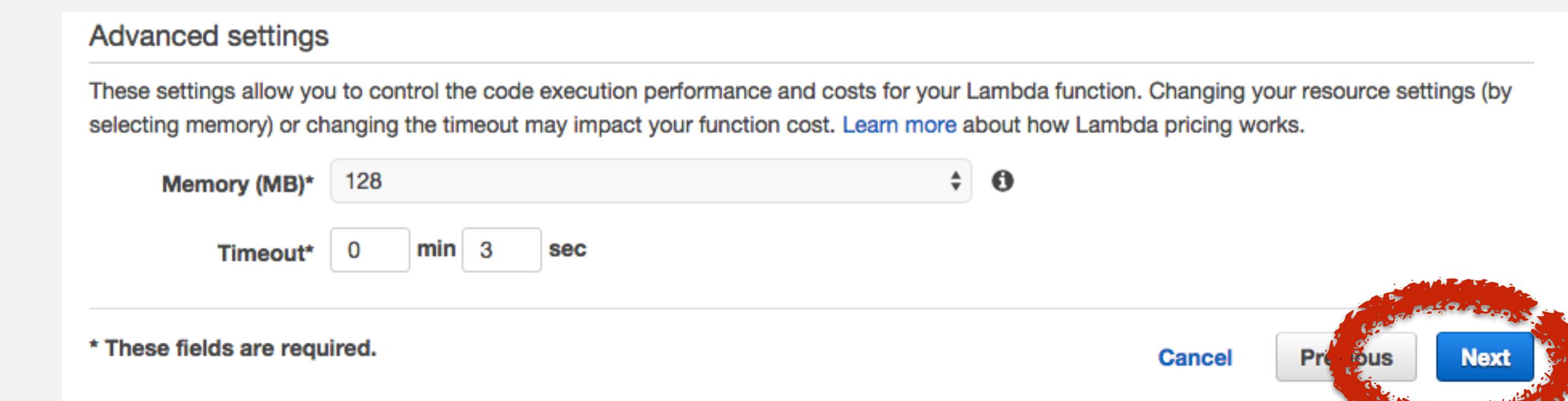
OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>

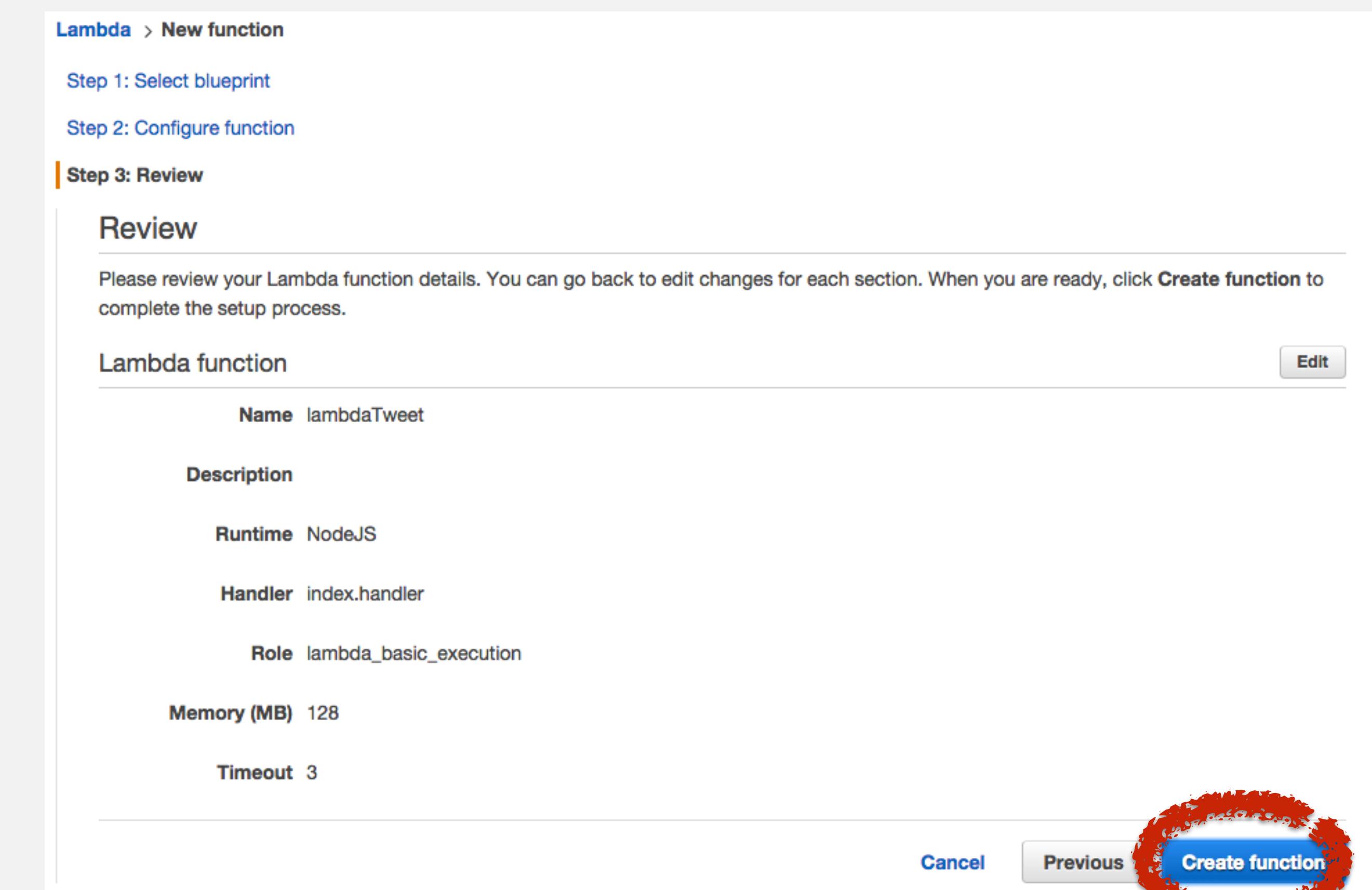
Lambda関数の作成

- ◆ 最後にメモリ使用量とタイムアウト時間を設定します。
- ◆ 今回は恐らくデフォルトで良いが、規模の大きいアプリケーションの場合はこの辺りを調整する必要があります。



Lambda関数の作成

- ◆ 確認画面で「Create function」をクリックすると・・・



Lambda関数の作成

- ◆ Lambda関数の完成！
- ◆ 以後の設定変更やソースの更新はこのページから行います。

Lambda > Functions > lambdaTweet ARN - arn:aws:lambda:us-east-1:123456789012:function:lambdaTweet

Test Actions

Congratulations! Your Lambda function "lambdaTweet" has been successfully created.

Code Configuration Event sources API endpoints Monitoring

It looks like your Lambda function "lambdaTweet" is unable to be edited inline, so you need to re-upload or your zip file contains more than one file to edit. However, you can still invoke your function.

Code entry type Upload a .ZIP file Upload a .ZIP from Amazon S3

For .ZIP files larger than 10 MB, consider uploading via S3.

 Upload



JAWS-UG

AWS User Group - Japan

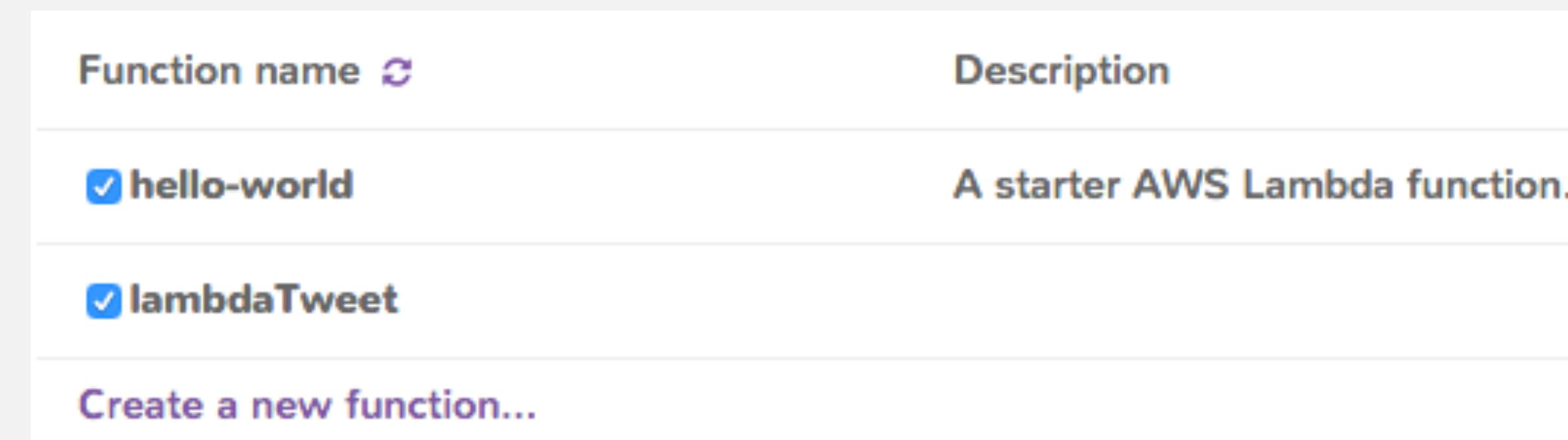
OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>

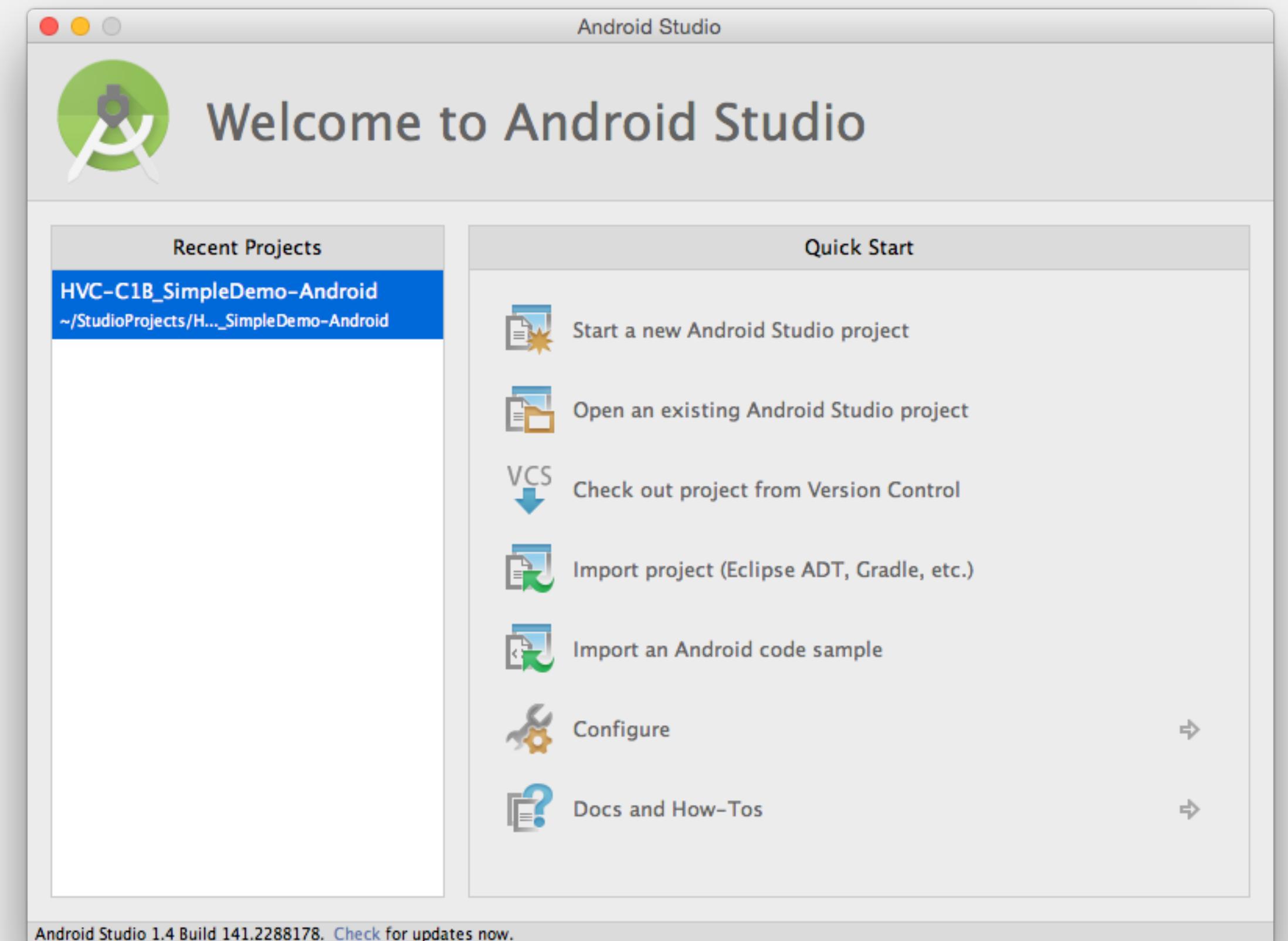
Lambda関数の作成

- ♦ **MobileHub**のタブに戻って、リロードアイコン(**Function name**の右隣)をクリックすると、いま作成した関数が出てくるのでチェックを入れます。
- ♦ 出てこなければ、8ページほど戻って「**Lambda**を作るリージョンの確認」を再度確認。
- ♦ 「**Save changes**」->左メニューの「**Build**」で、再度ソースをビルドして、ダウンロードします。



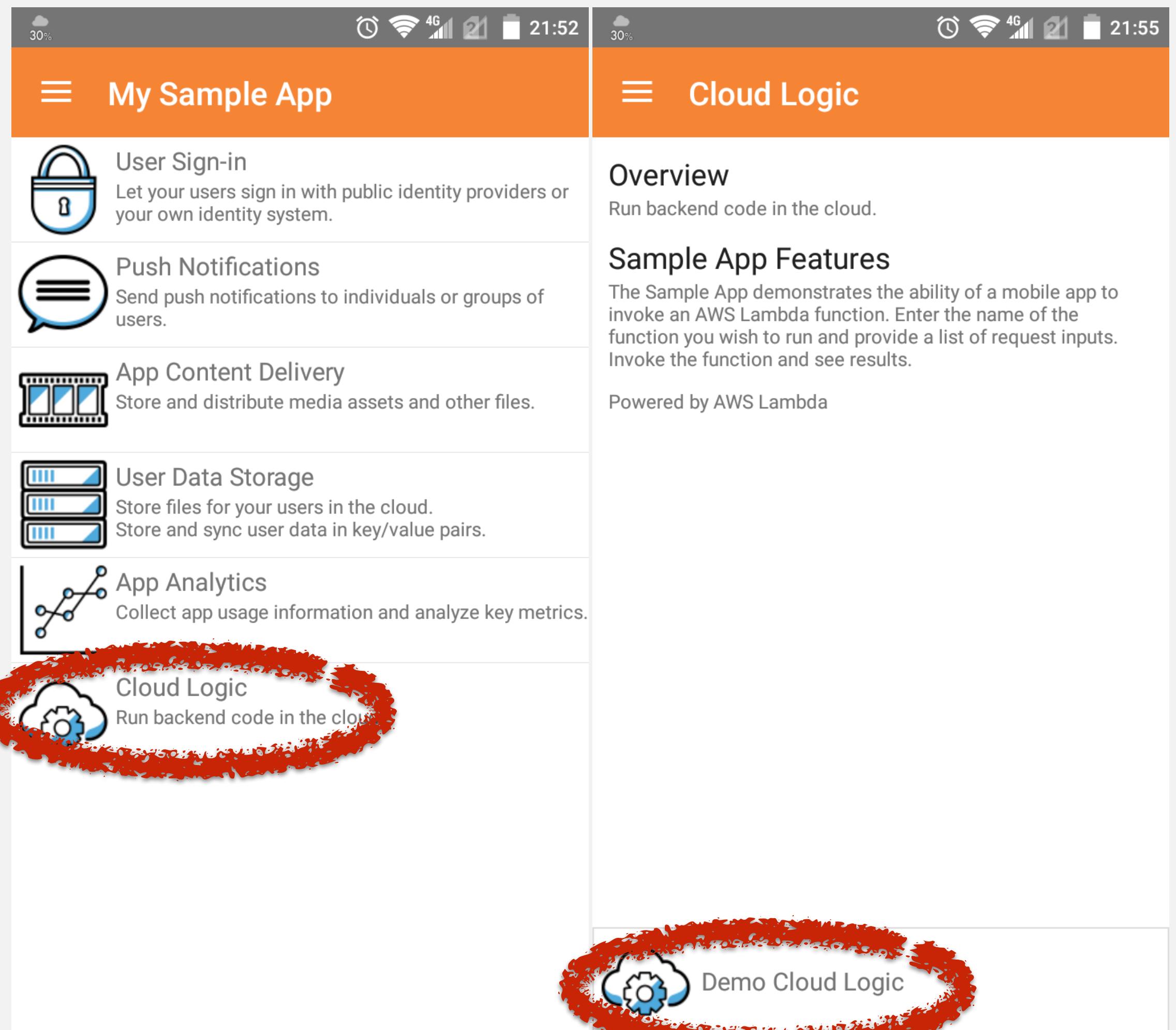
Android Studioで再度アプリをビルド

- ◆ ZIPファイルを解凍して、ソースを上書き
- ◆ Android Studioを起動して、「Import Project (Eclipse...)」を選択。
- ◆ フォルダは先程上書きしたフォルダの「MySampleApp」を選択します。
- ◆ メニューのRun -> Run 'app' を選択して実行してください。



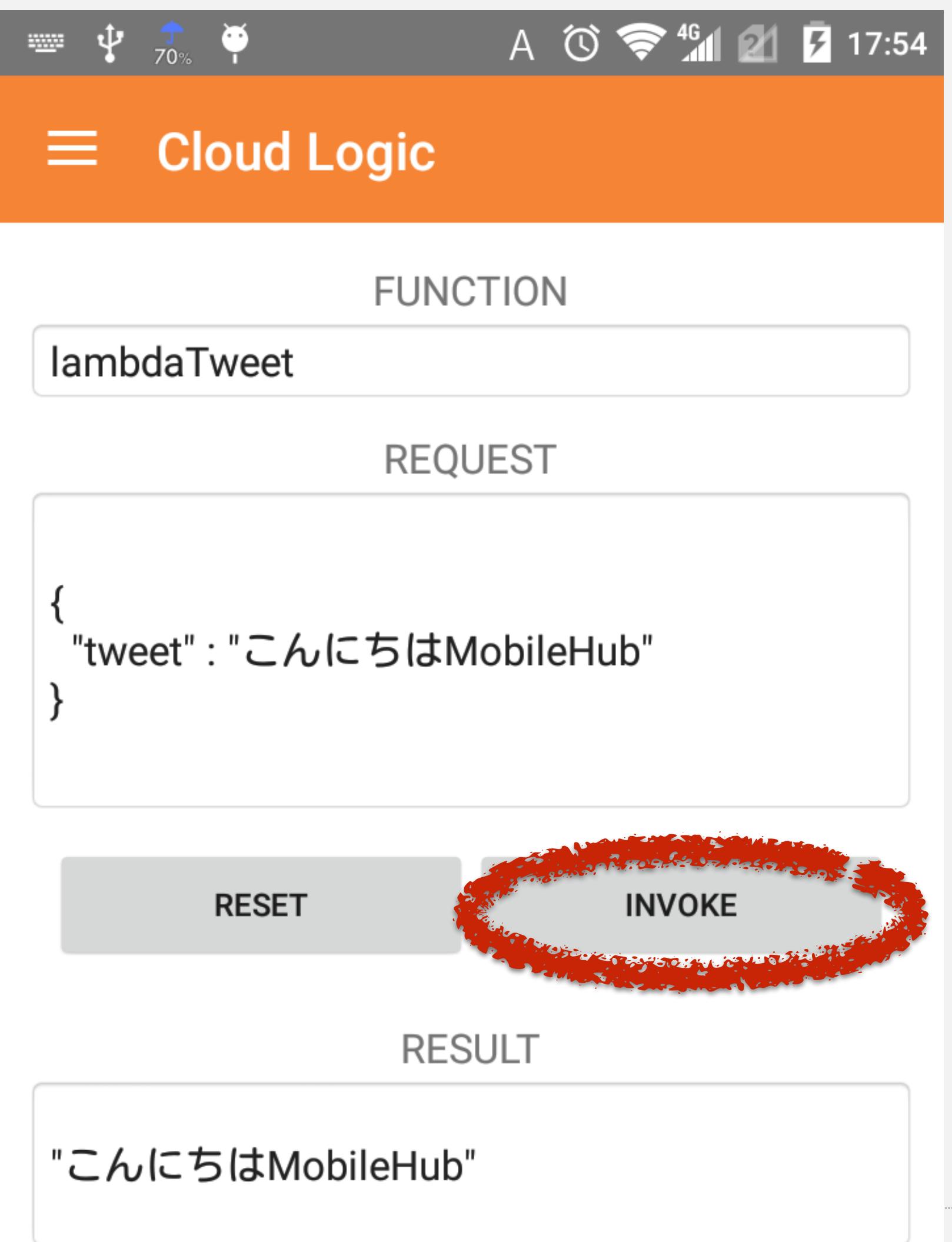
実機でのアプリの実行

- ♦ 一番下にメニューが増えてる！
- ♦ クリックして先に進み、**Cloud Logic**を実行してみましょう。



実機でのアプリの実行

- ◆ **FUNCTION**に先ほどの**Lambda**関数名
「LambdaTweet」
- ◆ **REQUEST**にJSONで
「{“tweet”: “投稿したいメッセージ”}」
- ◆ 最後に**「INVOKE」**をクリックすると・・・
- ◆ see <https://twitter.com/handson201601>





どこかで上手く行かなかった場合 その他もろもろのFAQ

- ◆ Lambdaの実行時ログはどこに出るの？
◆ CloudWatch -> ログ -> 関数名のロググループ
- ◆ Lambda関数の更新は？
◆ ZIPを作りなおして再度アップロード後、「Saveボタン」
- ◆ MobileHubが生成したソースの更新はAndroidStudioでどうやる？
◆ ディレクトリごと上書きして、「Import Project」が手っ取り早い





Lambdaサンプル

天気予報を取得：手順はほぼ一緒

- ◆ 関数名を「**lambdaWeather**」とする
- ◆ ダウンロードしたZIPファイルを**Upload**
- ◆ **Role**は先ほど作った「**lambda_basic_execution**」を選択
- ◆ **Lambda**関数が出来たら、**MobileHub**のタブに戻って、リロードアイコン
- ◆ 「**lambdaWeather**」にチェックを入れて、「**Save changes**」->左メニューの「**Build**」で、再度ソースをビルドして、ダウンロード、上書きしておく
- ◆ **Android Studio**でプロジェクトを**Import**、**Run**で実行



実機で実行

- ◆ **FUNCTION**に「**lambdaWeather**」
- ◆ **REQUEST**に**JSON**で
「**{"id": "数字"}**」と入力します。
- ◆ 数字は <http://weather.yahoo.co.jp/weather/rss/>
の警報・注意報「以外」を参照してください。
- ◆ 最後に「**INVOKE**」をクリック！

The screenshot shows the AWS Mobile Hub Cloud Logic interface. At the top, there's a header bar with various icons and the time '23:31'. Below it, a navigation bar says 'Cloud Logic'. The main area has tabs for 'FUNCTION' (selected) and 'REQUEST'. Under 'FUNCTION', it says 'lambdaWeather'. Under 'REQUEST', there's a text input field containing the JSON: { "id" : "9110" }. At the bottom, there are 'RESET' and 'INVOKE' buttons. In the 'RESULT' section, the output is displayed as a JSON array of weather forecasts: ["【14日（木）本島中南部（那霸）】曇り - 17°C/14°C - Yahoo!天気・災害", "【15日（金）本島中南部（那霸）】曇後雨 - 19°C/14°C - Yahoo!天気・災害", "【16日（土）本島中南部（那霸）】曇時々雨 - 20°C/14°C - Yahoo!天気・災害", "【17日（日）本島中南部（那"].



更にその先へ

更にその先へ

- ◆進む前に、再度現状の確認
- ◆実案件で使うにはどうすればいい？
- ◆自動生成されたソースコードの解析



現状の確認

- ◆ 現在の**Mobile Hub**は
「**AWSのリソース生成ウィザード**」 +
「**生成されたリソースにアクセスする**
サンプルソースコード自動生成ツール」
と見ることが出来ます。

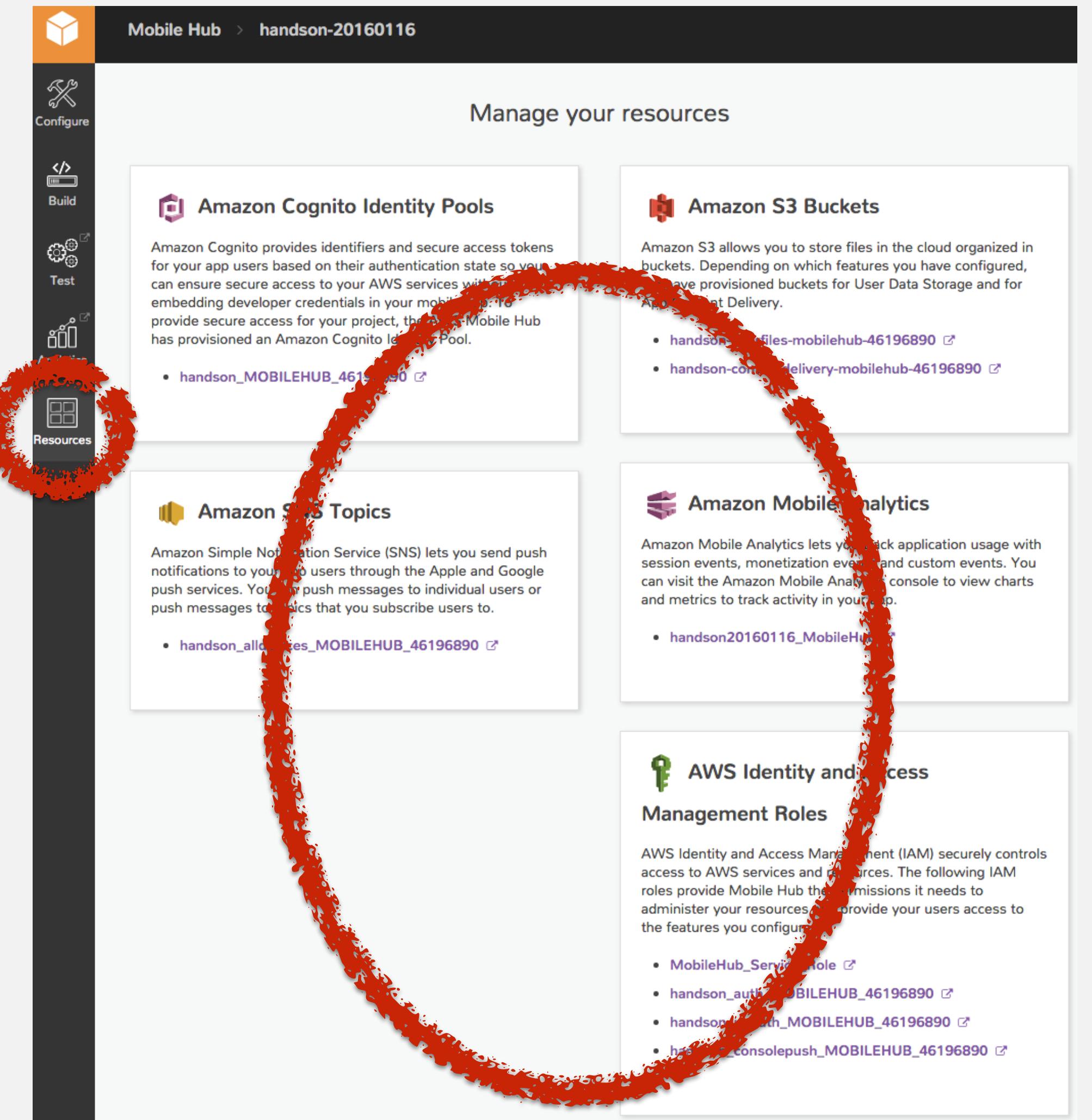
Mobile Hubが作成したAWS上のリソース

- ◆ **Mobile Hub**の左メニューの一番下のアイコン

「Resources」 をクリックします。

- ◆ ここに、**Mobile Hub**が自動的に作成した

AWS上のリソースが一覧されています。



完全無料で利用したい場合

♦ ここまで の作業は、全て無料枠の範囲内で収まる
はずですが（頑張って行っても数百円程度）、
心配の方はこのリソースを全てクリックして行っ
て、削除しておくと良いでしょう。

実案件で使うには？

- ♦ アプリ側で**GUI**を作りこむ
- ♦ デバイスから情報を取得する部分 (**GPS**とか)
も作る必要あり
- ♦ サーバ側で必要なロジック
(主にデータの読み書き) を作りこむ



実案件で使うには？

- ◆ AWS部分はインフラ担当に
「こんな感じで作ったからよろしくね！」
- ◆ アプリ部分はアプリ担当に
「これ参考によろしくね！」



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

Github repositories <https://github.com/jaws-ug/>



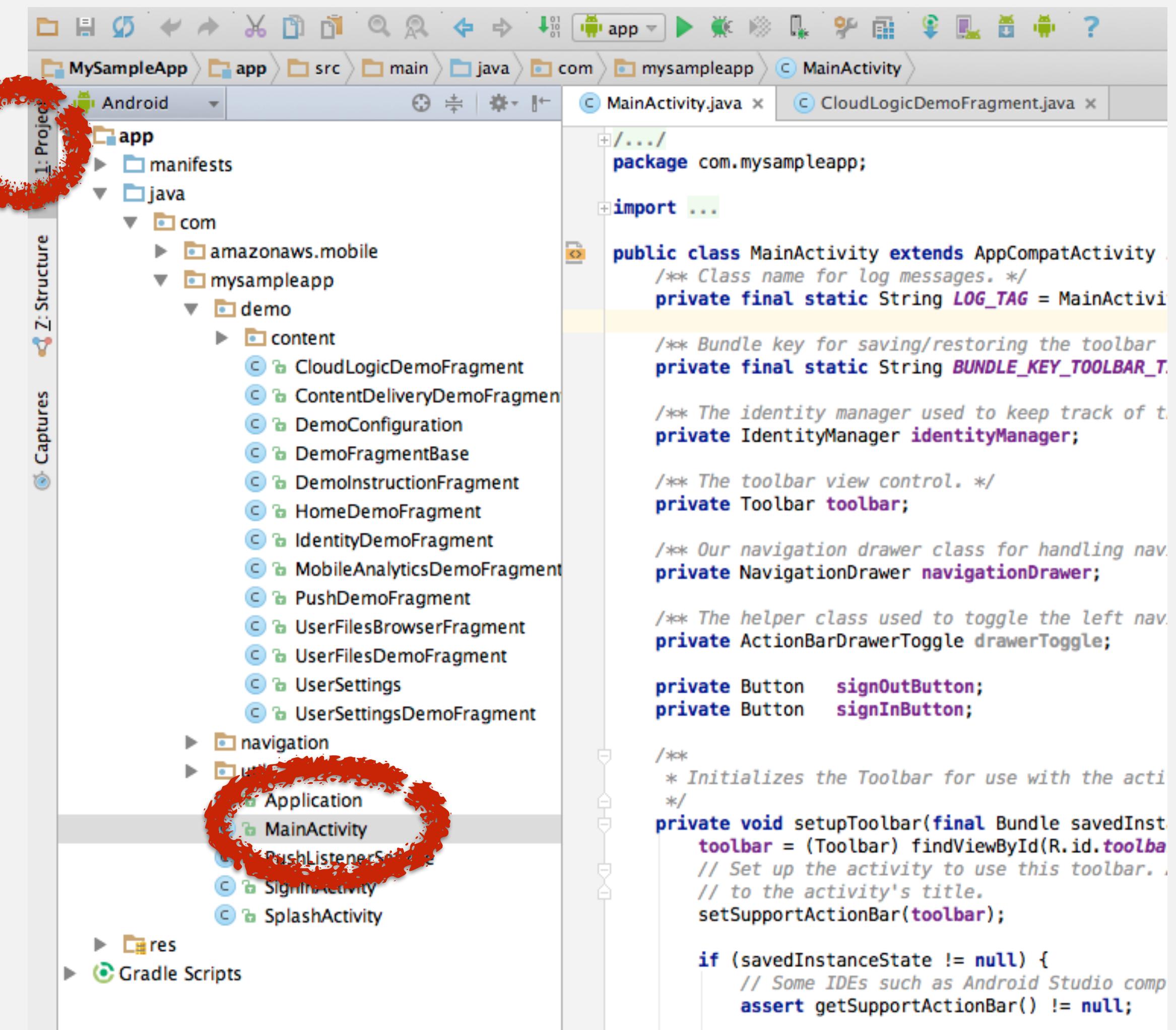
そうそう上手く
逃げられるはずがない

ここから少しだけ 自動生成されたソースコードの解析

- ♦ この先はほんの少しだけJavaの知識が必要になります
- ♦ 「オレ無理そう」というかたは、
アプリをいじって遊んでましょう(^^)
- ♦ ただし、エンジニアと共通言語を持つという意味では、
ちらっと見ておくと良いかと思います。

Android Studioでアプリを見てみる

- ◆ **Android Studio**を起動すると今まで開いていたプロジェクトが開くはず。
- ◆ 左上の「1: Project」をクリックすると、プロジェクトの一覧が見える。
- ◆ 「app」フォルダをクリックして中身をどんどん開いていって、「**MainActivity**」を探す。



ここで分かること

- ◆ パッケージ名は「**com.mysampleapp**」
- ◆ 起動時の**Activity**は「**com.mysampleapp.MainActivity**」

↑この2つは、事前準備編でFacebookApp側に設定したやつ！

これを変えたら、Facebook側の設定を変えるのをお忘れなく



UserSettingsクラス

- ◆ **com.mysampleapp.demo.UserSettingsを開く**
- ◆ **これはアプリの情報をクラウド側に保存するときに使われるもの**
- ◆ **アプリ起動時にクラウド側から情報を取ってくるコードになっている**
- ◆ **MainActivityの中で"syncUserSettings"を探してみると・・・**

The screenshot shows the file structure of a sample application named 'mysampleapp'. The 'demo' package contains several Java files, including 'UserSettings'. The code for 'UserSettings' is partially visible on the right, showing comments and variable declarations.

```
/*
 * Simple class for user set
 */
public class UserSettings {
    private static final String DATASET_NAME = "UserSettings";
    private static final String KEY_TITLE_TEXT_COLOR = "titleTextColor";
    private static final String KEY_TITLE_BAR_COLOR = "titleBarColor";
    private static final String KEY_BACKGROUND_COLOR = "backgroundColor";

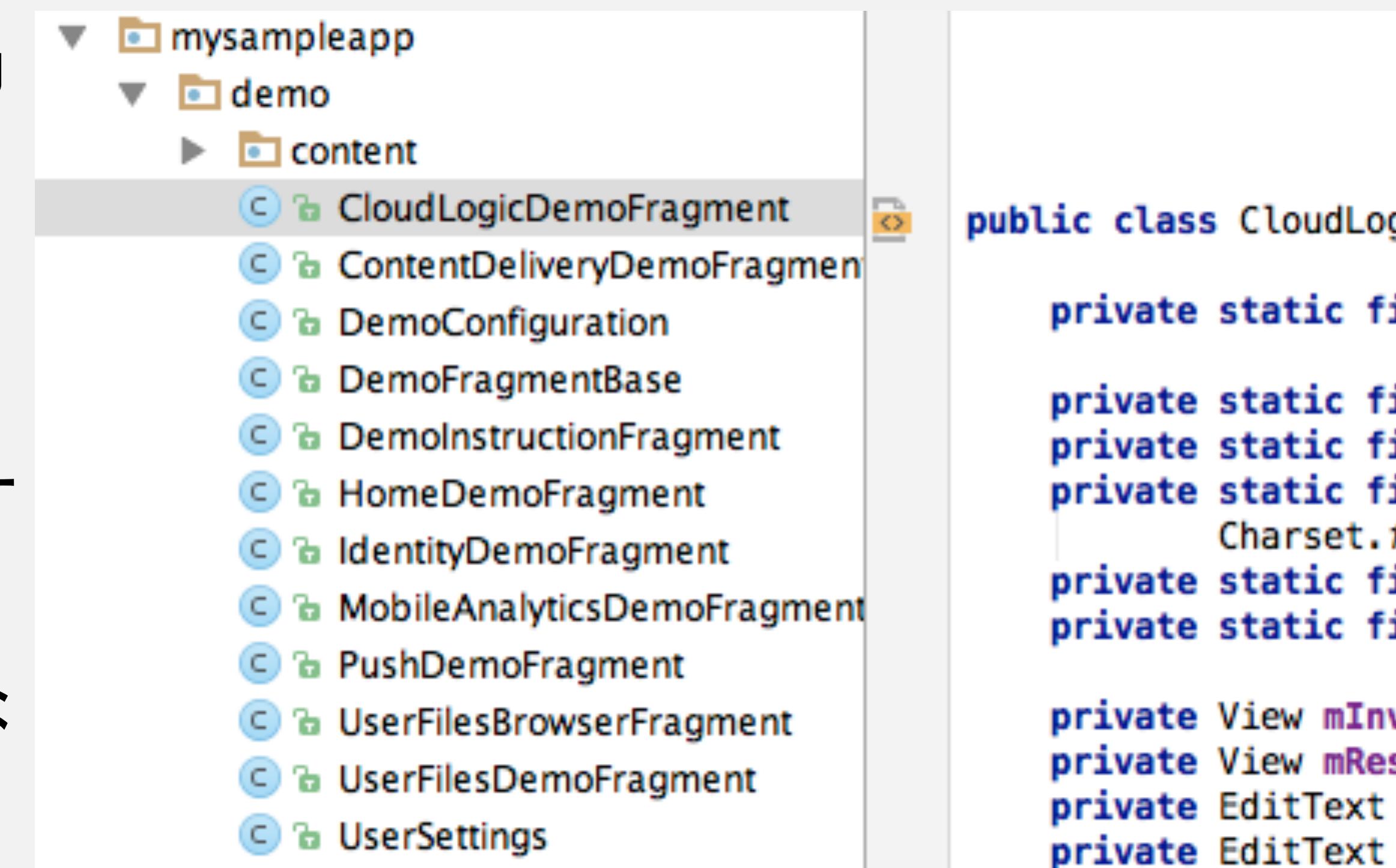
    // default color
    private int titleTextColor;
    private int titleBarColor;
    private int backgroundColor;

    private static UserSettings instance;
}
```



CloudLogicDemoFragment クラス

- ◆ **com.mysampleapp.demo.CloudLogicDemoFragment**を開く
 - ◆ これはCloud Logicの画面を開いたときに使われるもの
 - ◆ **DEFAULT_FUNCTION_NAME** や
DEFAULT_REQUEST_CONTENTS をいじって保存すると、画面を開いた時のデフォルト値が変わる
 - ◆ **invokeFunction**メソッドがLambda呼出のコードになるので参考にすると良い



その他もろもろ

- ◆ **ContentDeliveryDemoFragment**はS3利用のサンプル
- ◆ **MobileAnalyticsDemoFragment**は
Mobile Analytics利用のサンプル
- ◆ **PushDemoFragment**クラスはSNS購読のサンプル
などなど

アプリ構築のテンプレートとして利用可能！



JAWS-UG

AWS User Group - Japan

OKINAWA

<http://jaws-ug.jp/>

GitHub repositories <https://github.com/jaws-ug/>

お疲れ様でした
今回のハンズオンはここまで！

Any questions?

再掲：資料やソースコードはこちらから

<https://github.com/jaws-ug/hands-on/tree/master/Mobile-Hub>