
AWS Mobile Hub

Developer Guide

Version 1.0



AWS Mobile Hub: Developer Guide

Copyright © 2016 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

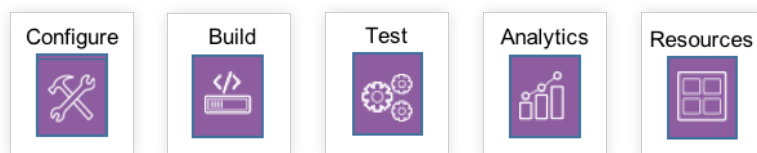
Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is AWS Mobile Hub?	1
How Can I Use AWS Mobile Hub?	2
Setting Up	3
Signing Up for AWS	3
Creating an IAM User	3
Enabling AWS Mobile Hub	4
Signing in to Mobile Hub and Creating Your Project	4
App Analytics	5
App Analytics At a Glance	5
Viewing AWS Resources Provisioned for this Feature	6
Quickstart App Details	6
App Content Delivery	7
App Content Delivery At a Glance	8
Configuring the App Content Delivery Feature	8
Viewing AWS Resources Provisioned for this Feature	8
Quickstart App Details	9
Cloud Logic	10
Cloud Logic At a Glance	11
Viewing AWS Resources Provisioned for this Feature	11
Quickstart App Details	12
NoSQL Database	13
NoSQL Database At a Glance	14
Learn more	14
Example Table Schemas	14
Configuring Your Tables	15
NoSQL Table Terminology	15
Data Permissions	16
Retrieving Data	17
Learn more	17
Quickstart App Details	18
Push Notifications	19
Push Notifications At a Glance	20
Configuring the Push Notifications Feature	20
Setting Up Push Notification Services	20
Setting Up iOS Push Notification	21
Setting Up Android Push Notification	28
Viewing AWS Resources Provisioned for this Feature	31
Quickstart App Details	32
Creating Test Push Notifications for the Quickstart app with Amazon SNS	32
User Data Storage	33
User Data Storage At a Glance	34
Viewing AWS Resources Provisioned for this Feature	34
Quickstart App Details	35
User Sign-in	36
User Sign-in Feature At a Glance	37
Configuring User Sign-in	38
User Sign-in and AWS Identity and Access Management (IAM)	38
Setting Up User Authentication	38
Setting Up Facebook Authentication	39
Setting Up Google Authentication	41
Setting Up Custom Authentication	55
Viewing AWS Resources Provisioned for this Feature	55
Quickstart App Details	55
Document History	57
Reference	59

IAM Managed Policies and Service Roles	59
Learn about AWS IAM Roles and Policies	59
IAM (Identity and Access Management) Managed Policies	59
IAM Service Role	60
Trust Relationship	61
Administrative Privileges	61
Service Policy	61

What Is AWS Mobile Hub?



[AWS Mobile Hub](#) provides an integrated console experience that enables you to quickly create and configure powerful mobile app backend features and integrate them into your mobile app. You create a project by selecting features to add to your app.

The features and AWS services that are supported by Mobile Hub are constantly evolving. Currently they include:

- [App Analytics](#) (p. 5)
- [App Content Delivery](#) (p. 7)
- [Cloud Logic](#) (p. 10)
- [NoSQL Database](#) (p. 13)
- [Push Notifications](#) (p. 19)
- [User Data Storage](#) (p. 33)
- [User Sign-in](#) (p. 36)

When you build your project for iOS Objective-C, iOS Swift, or Android, Mobile Hub automatically provisions and configures all of the AWS service resources that your app's features require. Mobile Hub then guides you through integrating the features into your app code and downloading a fully working quickstart app project that demonstrates those features.

After your mobile app is built, you can use Mobile Hub to test your app, then monitor and visualize how it is being used.

AWS Mobile Hub enables you to select the region in which your project's resources will be created. For more information about AWS regions, see [Regions and Endpoints](#).

When you use AWS Mobile Hub, you pay only for the underlying services that Mobile Hub provisions based on the features you choose in the Mobile Hub console. For more information, see [Pricing](#).

How Can I Use AWS Mobile Hub?

Mobile Hub provides all the information you need to use your sample app project:

- Explore the details of AWS mobile features
- Configure AWS services as mobile back ends
- Build your custom app on top of the solid foundation of your Mobile Hub sample app
- Get the app components and functional sample code you need for an app project you build from scratch.

To get started see [Setting Up AWS Mobile Hub \(p. 3\)](#).

Setting Up AWS Mobile Hub

Before you use AWS Mobile Hub for the first time, you must complete the following tasks:

Topics

- [Signing Up for AWS](#) (p. 3)
- [Creating an IAM User](#) (p. 3)
- [Enabling AWS Mobile Hub](#) (p. 4)

Signing Up for AWS

To use AWS Mobile Hub, you need an AWS account. Your account has access to all available services, but you are charged only for the services you use. If you are a new AWS customer, you can get started with the [AWS Free Tier](#).

Creating an IAM User

To provide better security, we recommend that you do not use your AWS root account to access Mobile Hub. Instead, create an AWS Identity and Access Management (IAM) user, or use an existing IAM user, in your AWS account and then access Mobile Hub with that user. For more information, see [AWS Security Credentials](#) in the AWS General Reference. .

If you signed up for AWS but have not created an IAM user for yourself, you can create one by using the IAM console. First, create an IAM administrator group, then create and assign a new IAM user to that group.

To create an IAM administrators group

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups**, and then choose **Create New Group**.
3. For **Group Name**, type a name for your group, such as **Administrators**, and then choose **Next Step**.
4. In the list of policies, select the check box next to the **AdministratorAccess** policy. You can use the **Filter** menu and the **Search** box to filter the list of policies.
5. Choose **Next Step**, and then choose **Create Group**. Your new group is listed under **Group Name**.

The following procedure describes how to create an IAM user for yourself, add the user to the administrators group, and create a password for the user.

To add an IAM user to your group and assign a password

1. In the navigation pane, choose **Users**, and then choose **Create New Users**.
2. In box **1**, type a user name. Clear the check box next to **Generate an access key for each user**. Then choose **Create**.
3. In the list of users, choose the name (not the check box) of the user you just created. You can use the **Search** box to search for the user name.
4. In the **Groups** section, choose **Add User to Groups**.
5. Select the check box next to the administrators group. Then choose **Add to Groups**.
6. Scroll down to the **Security Credentials** section. Under **Sign-In Credentials**, choose **Manage Password**.
7. Select **Assign a custom password**. Then type a password in the **Password** and **Confirm Password** boxes. When you are finished, choose **Apply**.

Enabling AWS Mobile Hub

AWS Mobile Hub administers AWS resources for mobile app projects on behalf of the customer. This includes automation that creates AWS Identity and Access Management (IAM) roles for mobile app users and updates their permissions based on the features that are enabled in a mobile app project. Because these operations require administrative privileges (the ability to create and modify IAM roles), only a user with administrative privileges may enable Mobile Hub to do this. These are the steps an administrative user must take in order to enable AWS Mobile Hub in an AWS account. This only needs to be done once.

To enable Mobile Hub in an AWS account

1. Navigate to the AWS Mobile Hub console at <https://console.aws.amazon.com/mobilehub/>.
2. Choose **Get Started**.
3. Review the details of the **First things first...** page.
4. Choose **Yes, grant permissions**.

Signing in to Mobile Hub and Creating Your Project

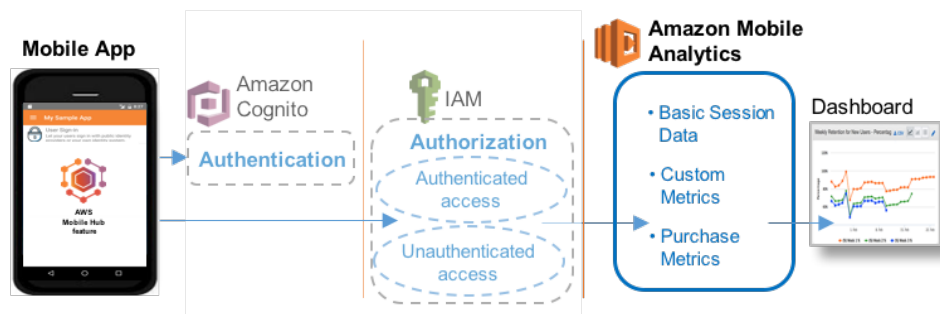
A Mobile Hub project is a logical workspace that contains the features you choose to incorporate into your mobile app. You can create as many projects as you wish.

To create a Mobile Hub project

1. Choose **Get Started** or **Create new project**.
2. For **Project name**, type a name for your project.
3. Choose **Create project**.

App Analytics

Choose AWS Mobile Hub App Analytics mobile backend service feature to capture and visualize metrics that enable you to understand and optimize your app for the usage and purchasing behavior of your users.



The App Analytics feature enables you to collect and visualize usage analytics for your app using [Amazon Mobile Analytics](#). You can collect standard session start/stop events, monetization events, and custom events that you define.

Topics

- [App Analytics At a Glance \(p. 5\)](#)
- [Viewing AWS Resources Provisioned for this Feature \(p. 6\)](#)
- [Quickstart App Details \(p. 6\)](#)

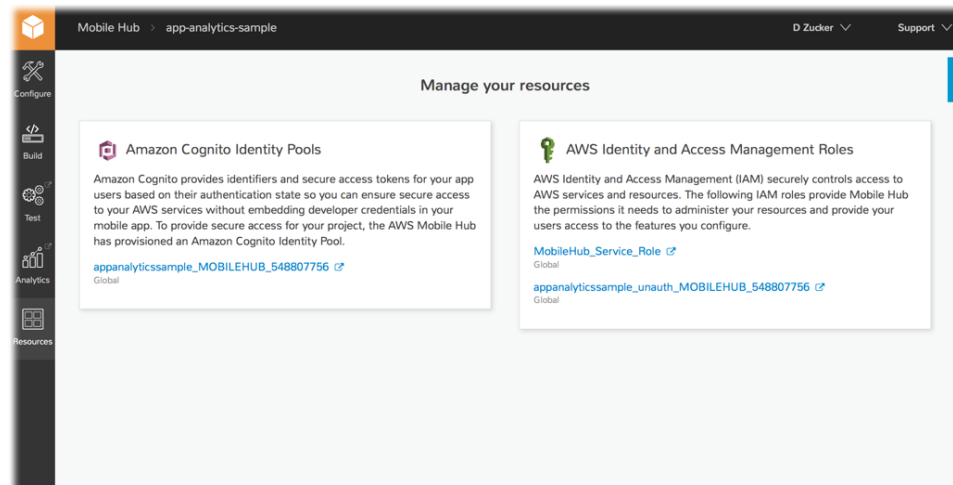
App Analytics At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon Mobile Analytics (see the Amazon Mobile Analytics User Guide) Concepts Console Pricing <p>Mobile Hub features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 6).</p>
--	--

Configuration options	<ul style="list-style-type: none">• Enable the feature to start capturing basic app usage data.• Disable the feature to stop capturing metrics.
Quickstart app demos	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• Causes app usage data to be delivered to AWS on startup.• Causes purchase metrics to flow to and be captured by AWS.

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the **Amazon Mobile Analytics** feature.

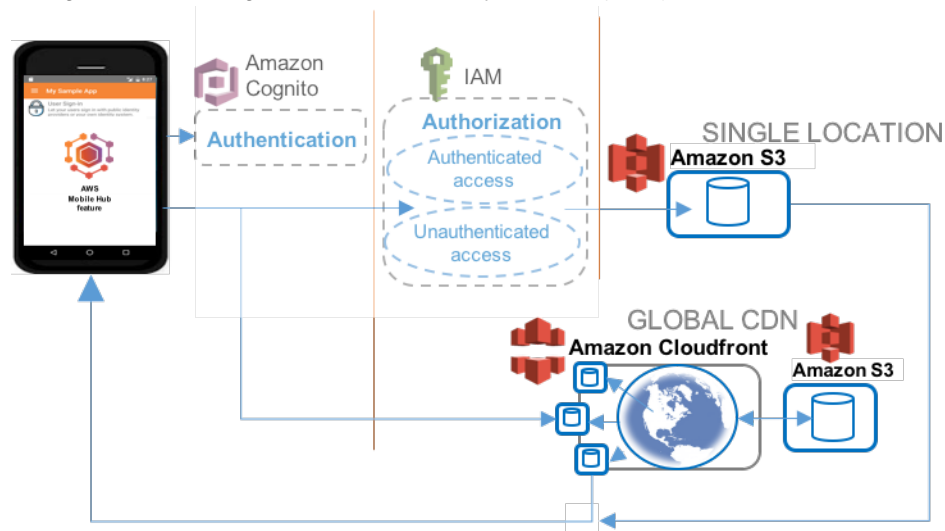


Quickstart App Details

The quickstart app includes code to automatically start collecting app usage on startup. In addition, the app contains two buttons in the Mobile Analytics demo that illustrate triggering custom events and monetization events.

App Content Delivery

Choose AWS Mobile Hub App Content Delivery to add access to cloud content to your mobile app, from a single location or a global Content Delivery Network (CDN).



The App Content Delivery feature enables you to store app assets, like resource or media files, in the cloud so you can download and cache them within your app. Mobile Hub offers two choices for distributing these files: either from a single location using an [Amazon S3](#) bucket or distributed through a global content delivery network by using [Amazon CloudFront](#).

Topics

- [App Content Delivery At a Glance \(p. 8\)](#)
- [Configuring the App Content Delivery Feature \(p. 8\)](#)
- [Viewing AWS Resources Provisioned for this Feature \(p. 8\)](#)
- [Quickstart App Details \(p. 9\)](#)

App Content Delivery At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon CloudFront - Content Delivery Network (see Amazon CloudFront Concepts Console Pricing)• Amazon S3 Bucket (see Amazon Simple Storage Service Getting Started Guide Concepts Console Pricing) <p>Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 8).</p>
Configuration options	<p>This feature enables the following mobile backend capabilities:</p> <ul style="list-style-type: none">• Single location (AWS storage in a single regional location)• Global CDN (AWS storage on a global Content Distribution Network) <p>For more information, see Configuring the App Content Delivery Feature (p. 8).</p>
Quickstart app demos	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• View file list in AWS storage, download and view files, and manage their local cache.• Same behavior from a single storage location and a global content distribution network.

Configuring the App Content Delivery Feature

If you choose the **Single location** option, Mobile Hub creates an Amazon S3 bucket and pre-populates the bucket with a few sample files that are distributed to your quickstart app directly from there.

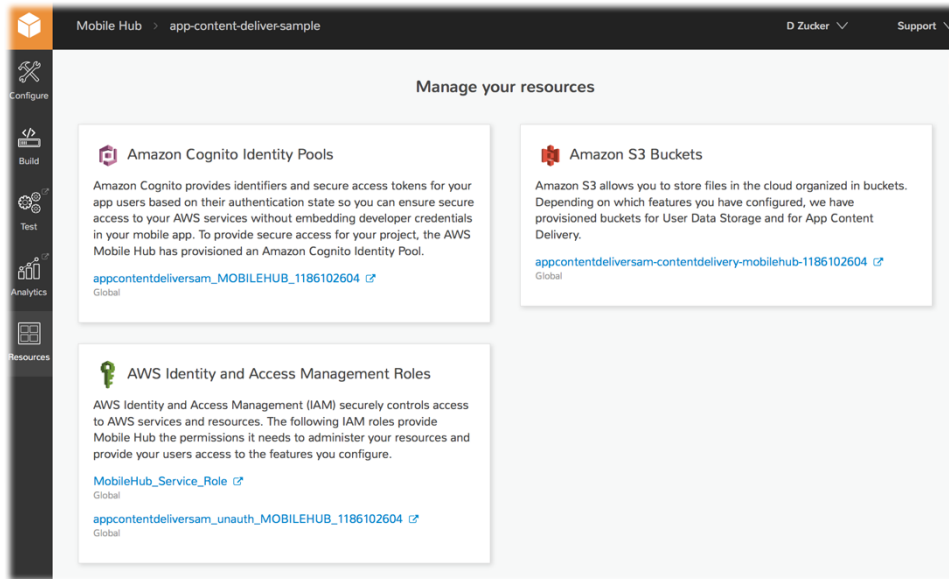
If you choose the **Global CDN** option, Mobile Hub provisions an Amazon CloudFront distribution to deliver files to your app. Amazon CloudFront caches your files using your Amazon S3 bucket as the source (origin) in edge locations around the world to provide faster, lower latency access to your files. Learn more about [Amazon CloudFront](#).

Viewing AWS Resources Provisioned for this Feature

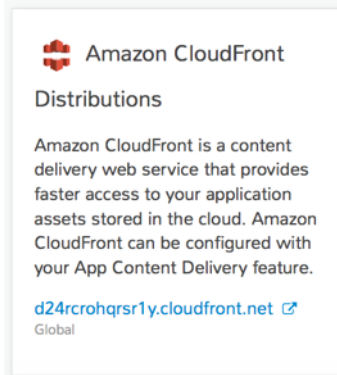
The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the App Content Delivery feature with **Single location** selected.

AWS Mobile Hub Developer Guide

Quickstart App Details



The following image illustrates the resource typically provisioned for the additional CloudFront element of the App Content Delivery feature with **Global CDN** selected.

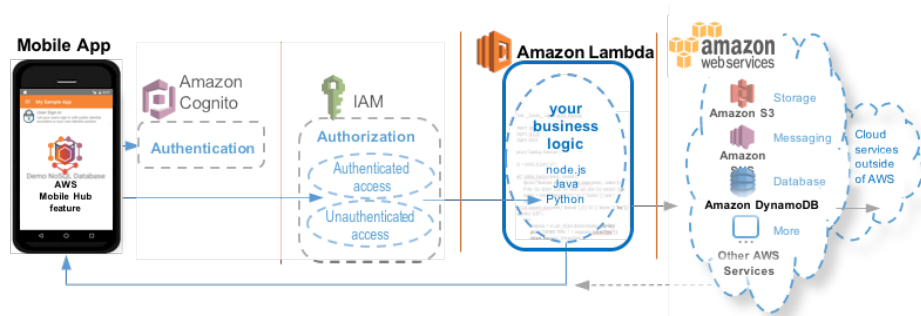


Quickstart App Details

In the Mobile Hub quickstart app, the App Content Delivery demo lists a set of image files that can be downloaded and cached locally and displayed on the device. The user can also delete the local copy of the image files.

Cloud Logic

Choose the AWS Mobile Hub Cloud Logic mobile backend service feature to add business logic functions in the cloud and extend to other AWS services for your app, with no cost for server set up or maintenance.



The Cloud Logic feature lets you build backend services using [AWS Lambda](#) functions that you can call from your mobile app. Using Cloud Logic, you can run code in the cloud to process business logic for your apps and share the same code for both iOS and Android apps. The Cloud logic feature is powered by AWS Lambda functions, which allow you to write code without worrying about managing frameworks and scaling backend infrastructure. You can write your functions in JavaScript, Java, or Python.

Topics

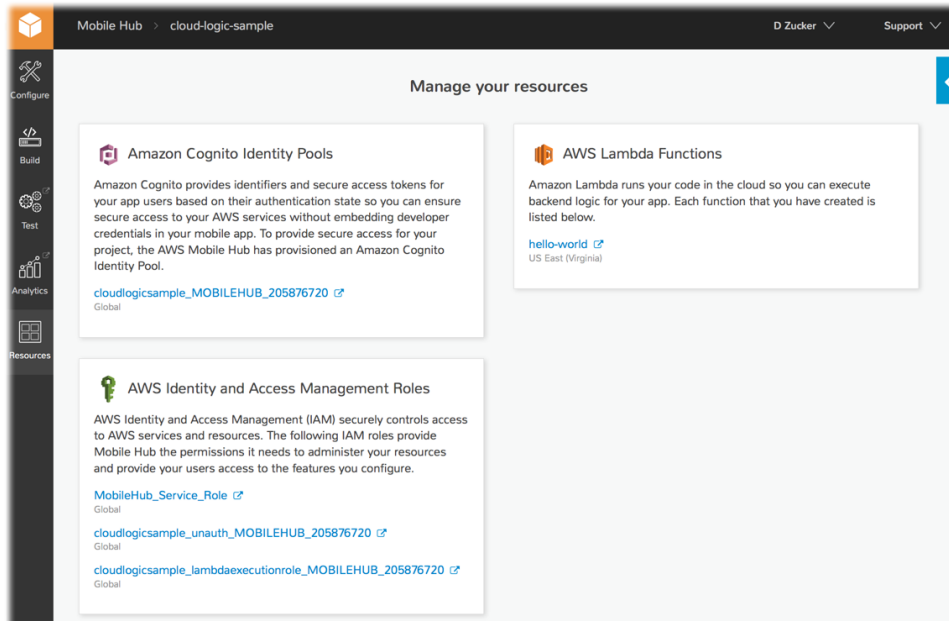
- [Cloud Logic At a Glance](#) (p. 11)
- [Viewing AWS Resources Provisioned for this Feature](#) (p. 11)
- [Quickstart App Details](#) (p. 12)

Cloud Logic At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• AWS Lambda (see AWS Lambda Developer Guide) Concepts Console Pricing <p>Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 8).</p>
Configuration options	<p>This feature enables the following mobile backend capabilities:</p> <ul style="list-style-type: none">• Provides a default Hello World Lambda function that accepts the parameter value entered by the app user and returns it back to an app.• Enables you to choose an existing function from the list provided or use the AWS Lambda console to create new functions.
Quickstart app demos	<p>This feature adds the following functionality to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• User can specify an AWS Lambda function by name, provide parameters and call a function and see the value returned by the function

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the Cloud Logic feature.



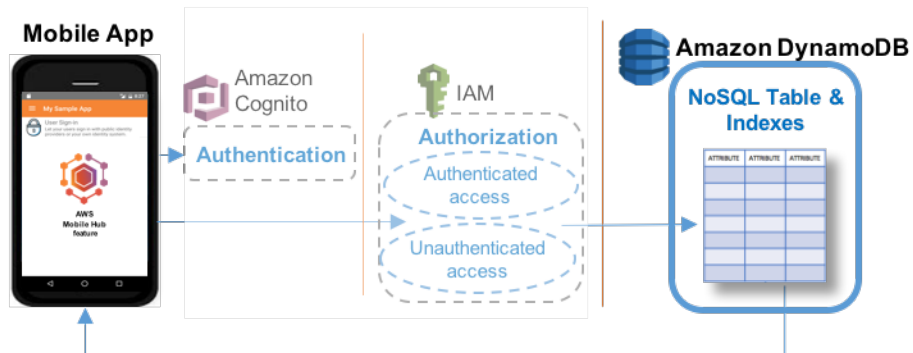
Quickstart App Details

Your quickstart app includes code to use AWS Lambda APIs to invoke any functions you have selected in your project. Adding Cloud Logic to your quickstart app provides a Hello World default Lambda function. You can also choose an existing Lambda function from your AWS account, or you can create a new one. When you choose the edit button, you are taken to the function editor in the AWS Lambda console. From the Lambda console, you can edit the code directly or upload a package of source and libraries as a .zip file.

In the demo screen of the Cloud Logic quickstart app, you can enter the name and input parameters of the Lambda function you wish to invoke. The quickstart app then calls your Lambda function and displays the results it returns.

NoSQL Database

Choose the Mobile Hub NoSQL Database mobile backend feature to your mobile app to add database capabilities that are easy to develop and provide scalable performance and cost.



The NoSQL Database feature uses [Amazon DynamoDB](#) to enable you to create database tables that can store and retrieve data for use by your apps.

NoSQL databases are widely recognized as the method of choice for many mobile backend solutions due to their ease of development, scalable performance, high availability, and resilience. For more information, see [From SQL to NoSQL](#) in the *Amazon DynamoDB Developer Guide*.

Topics

- [NoSQL Database At a Glance](#) (p. 14)
- [Configuring the NoSQL Database Feature](#) (p. 14)
- [Configuring Your Tables](#) (p. 15)
- [Viewing AWS Resources Provisioned for this Feature](#) (p. 17)
- [Quickstart App Details](#) (p. 18)

NoSQL Database At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon DynamoDB tables (see Working with Tables in DynamoDB) Concepts Console Pricing <p>Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 17).</p>
Configuration options	<p>This feature enables the following mobile app backend capabilities:</p> <p>Configuring Your Tables (p. 15) - Using custom schema, based on a sample schema provided, or by using a wizard that guides you through choices while creating a table</p> <p>Data Permissions (p. 16) - Access to your app's data can be:</p> <ul style="list-style-type: none">• Public (enables any mobile app user to read or write any item in the table)• Protected (enables any mobile app user to read any item in the table but only the owner of an item can update or delete it)• Private (enables only the owner of an item to read and write to a table) <p>For more information, see Configuring the NoSQL Database Feature (p. 14).</p>
Quickstart app demos	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• Insert and remove sample data, based on the schema you specify in the console.• Perform and see the results of NoSQL operations on tables including Get, Scan, and all the example queries displayed by the console as you make design selections.

Configuring the NoSQL Database Feature

This section describes steps and options for configuring NoSQL Database features in Mobile Hub.

To add the NoSQL Database feature to your Mobile Hub project

1. Choose **Enable NoSQL**.
2. Choose **Add a new table**.
3. Choose the initial schema for the table. You can use a provided example schema, or generate a schema through the wizard.

Example Table Schemas

AWS Mobile Hub provides a set of example table schemas for typical mobile apps. If you create a table using one of the example schema templates, the table initially has a set of attributes specific to each example. You can choose one of these templates as the starting schema for your table:

- **News**, which stores author, title, article content, keywords, and other attributes of news articles.

- **Locations**, which stores names, latitude, and longitude of geographic locations.
- **Notes**, which stores private notes for each user.
- **Ratings**, which stores user ratings for a catalog of items.
- **Graffiti Wall**, which stores shared drawing items.

To add a table using one of the example schema templates in your Mobile Hub project

1. Choose the example template to use for the initial schema of the table.
2. Type a new name in **Table name** to rename the table if you wish. Each template gives the table a default name matching the name of the template.
3. Choose **Public**, **Protected**, or **Private** permissions to grant to the mobile app users for the table. For more information, see [Data Permissions \(p. 16\)](#).
4. (Optional) Under **What attributes do you want on this table?**, you can add, rename, or delete table attributes.
5. (Optional) Choose **Add index** to add **name**, **partition key**, and (optionally) **sort key** for a secondary index for your table.
6. Choose **Create table**.

Configuring Your Tables

This section describes options for configuring DynamoDB NoSQL tables for your app.

Contents

- [NoSQL Table Terminology \(p. 15\)](#)
- [Data Permissions \(p. 16\)](#)
 - [Enforcing Permissions \(p. 16\)](#)
 - [Restricting Permissions for Multiple Writers \(p. 16\)](#)
 - [Table Permissions Options in Mobile Hub \(p. 16\)](#)
- [Retrieving Data \(p. 17\)](#)

NoSQL Table Terminology

Similar to other database management systems, DynamoDB stores data in tables. A table is a collection of data with the following elements.

Items

Each table contains multiple items. An item is a group of attributes that is uniquely identifiable among all of the other items. Items are similar to rows, records, or tuples in relational database systems.

Attributes

Attributes are the columns in a DynamoDB table. The rows of the table are the individual records you add, update, read, or delete as necessary for your app.

The table schema provides a set of initial attributes based on the needs of each example. You can remove any of these attributes by choosing **Remove**. If you remove the partition key attribute, then you must designate another attribute as the partition key for the primary index of the table.

You can choose **Add attribute** to add a blank attribute to the table. Give the attribute a name, choose the type of data it will store, and choose whether the new attribute is the partition key or the sort key.

Indexes

Each table has a built-in primary index, which has a partition key and may also have a sort key. This index allows specific types of queries. You can see the types of queries the table can perform by expanding the **Queries this table can perform** section. To enable queries using other attributes, create additional secondary indexes. Secondary indexes enable you to access data using a different partition key and optional sort key from those on the primary index.

Data Permissions

When you create a new table, you must set permissions that determine which mobile app users can read and/or write the table's data. You can set these permissions to control access to each table as: public, protected, or private.

Enforcing Permissions

Use the settings in the **What permissions would you like for this table?** section to enable your mobile app to directly access your NoSQL tables in the Amazon DynamoDB service. Because there is no middle layer between the mobile app and the database service, it is important that you use an appropriate access policy to restrict access to your tables. When you choose permissions for each table, Mobile Hub provisions a fine-grained access control policy for your mobile app users. If you select **Protected** or **Private**, then every operation that is attempted on an item in your table will first check if the **userId** field in the table item (or row) matches the user's Amazon Cognito Identity.

As the value of the primary partition key of a restricted NoSQL Database table will contain the Amazon Cognito Identity of the app user whose action created the item, the key must be called **userId** and be of type **string**). The name and data type of secondary indexes for restricted tables must follow the same pattern: **'userId' (string)**.

Restricting Permissions for Multiple Writers

After Mobile Hub provisions access restrictions for your tables with **Protected** or **Private** permissions, IAM ensures that only the mobile app user whose action creates an item in the table will be able to write to the attribute values of that item. To design your schema for the case where multiple users need to write data to an existing item, one strategy is to structure your schema in a way that users write to different tables. In this design, the app queries both tables to join data.

For example, customers may create orders in an **order** table and delivery service drivers may write delivery tracking information to a **deliveries** table, where both tables have secondary indexes that allow fast lookup based on **orderId** or **customerId**.

Table Permissions Options in Mobile Hub

When you create a new table, you must set the table's permissions. These permissions determine who can read data from and who can write data to the table. Mobile Hub offers the following table permissions configurations.

Public

Public permissions allow all mobile app users to read or write all items in the table. There is no restriction on how you configure the partition key.

Protected

Protected permissions allow all mobile app users to read all items in the table but only the owner of an item can update or delete it. These permissions grant full access to retrieve data from the table but limited access to update or remove existing items.

Only app users with an Amazon Cognito Identity ID matching the item's partition key can write to the item. The partition key for the table must follow the pattern of **'userId' (string)** value.

Private

Private permissions allow only the owner of an item to read and write to it. This enforces the most restrictive set of permissions for accessing the table; however, these permissions offer a higher degree of protection by limiting access.

Only app users with an Amazon Cognito Identity ID matching the item's partition key can read or write to the item. The partition key for the table and for any secondary indexes must follow the pattern of **'userId' (string)**value.

Retrieving Data

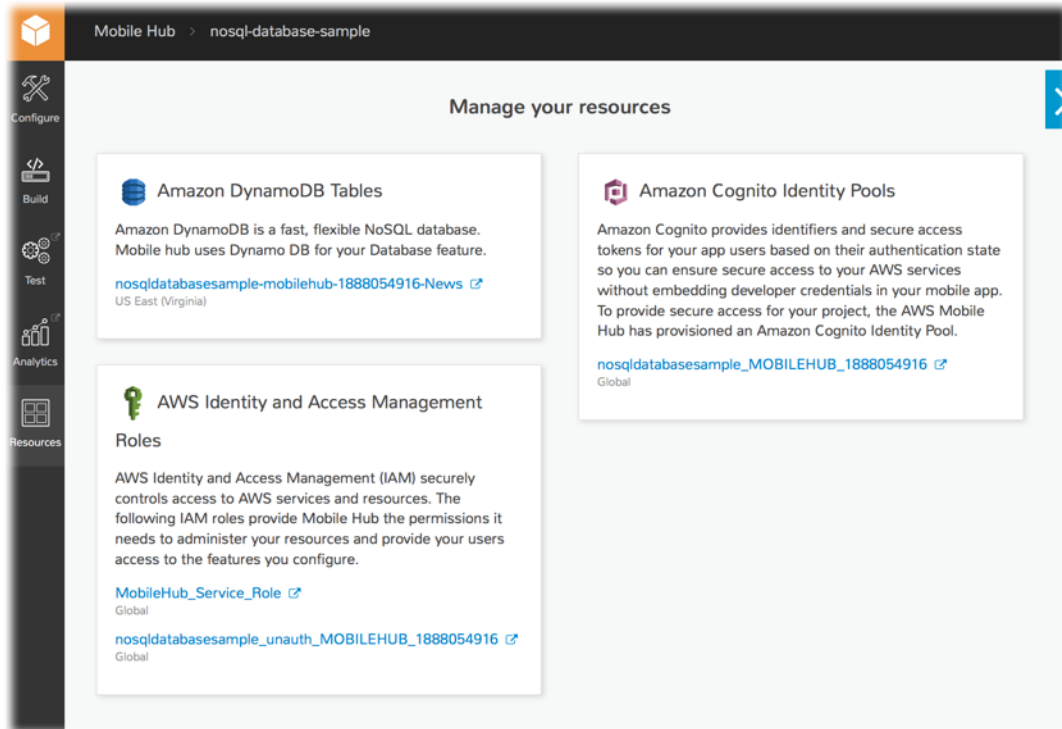
The operations you can use to retrieve data from your NoSQL database include the following:

- `Get`, which retrieves a single item from the table based on matching the primary key.
- `Query`, which finds items in a table or a secondary index using only primary key attribute values.
- `Scan`, which reads every item in a table or secondary index. By default, a `Scan` operation returns all of the data attributes for every item in the table or index. You can use `Scan` to return only some attributes, rather than all of them.
- `Query with Filters`, which performs a `Query` but returns results that are filtered based on a filter expression you create.
- `Scan with Filters`, which performs a `Scan` but returns results that are filtered based on a filter expression you create.

For more information, see [Query and Scan Operations in DynamoDB](#).

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying the AWS elements typically provisioned for the NoSQL Database feature:



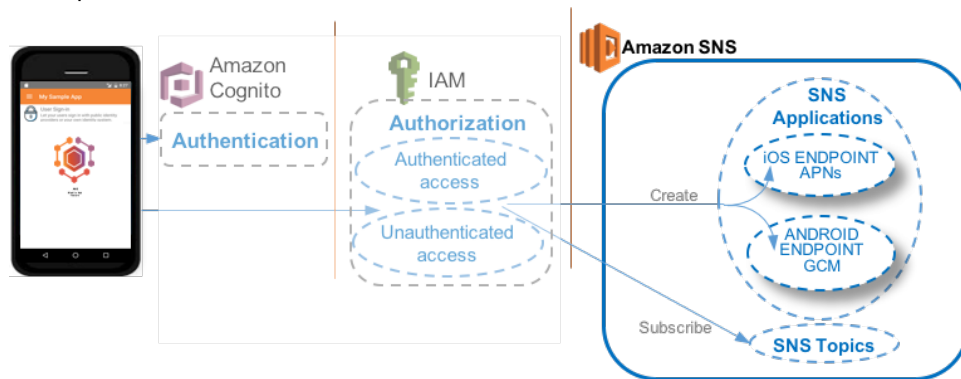
Quickstart App Details

In the Mobile Hub quickstart app, the NoSQL Database demo shows a list of all tables created during app configuration. Selecting a table shows a list of all queries that are available for that table, based on the choices made regarding its primary indexes, secondary indexes, and sort keys. Tables that you make using the example templates enable an app user to insert and remove sample data from within the app.

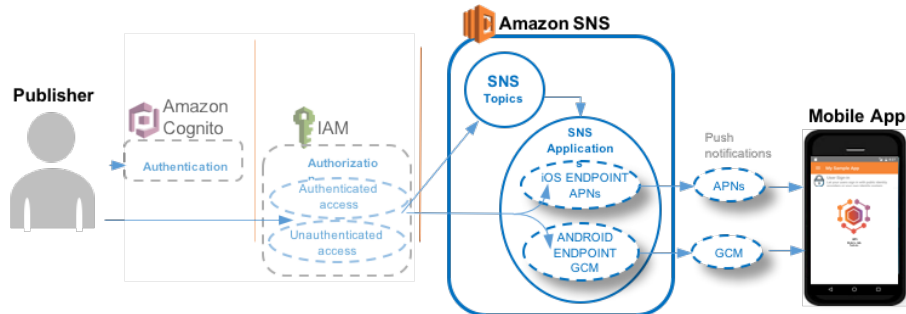
Push Notifications

Choose AWS Mobile Hub Push Notifications mobile backend feature to add mobile messaging to your mobile app.

The following image shows client perspective on subscription of a mobile app to Amazon SNS applications and topics.



The following image shows the server side developer perspective on publishing of push notifications to devices subscribed to Amazon SNS applications and topics.



The Push Notifications feature enables you to send push notification messages to your iOS and Android apps using [Amazon Simple Notification Service \(Amazon SNS\)](#). You can integrate with Apple and Google messaging services by providing credentials that are provided by those services. You then can send messages directly to individual devices, or publish messages to the SNS topics that installed apps are subscribed to.

Topics

- [Push Notifications At a Glance](#) (p. 20)
- [Configuring the Push Notifications Feature](#) (p. 20)
- [Setting Up Push Notification Services](#) (p. 20)
- [Viewing AWS Resources Provisioned for this Feature](#) (p. 31)
- [Quickstart App Details](#) (p. 32)

Push Notifications At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon SNS (see Amazon Simple Notification Service Developer Guide) Concepts Console Pricing <p>Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 31).</p>
Configuration options	<p>This feature enables the following mobile backend capabilities:</p> <p>Messaging Service Integration</p> <ul style="list-style-type: none">• via Google Cloud Messaging (GCM) (see Setting Up Android Push Notification (p. 28))• via Apple Push Notification service (APNs) (see Setting Up iOS Push Notification (p. 21)) <p>For more information, see Configuring the Push Notifications Feature (p. 20).</p>
Quickstart demo features	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• App is registered to receive and display push messages sent to the device individually, and those sent to a topic the device has been subscribed to.

Configuring the Push Notifications Feature

When you include Push Notifications in your project, Mobile Hub creates an Amazon SNS Platform Application based on your choice of push platform. Mobile Hub also creates an Amazon Simple Notification Service topic named `ALL_DEVICES` and modifies the IAM role to allow your app to create a platform endpoint and subscribe that endpoint to the `ALL_DEVICES` topic and any others you configured when creating your project.

For more information on choosing a push provider, see [Setting Up Push Notification Services](#) (p. 20).

Setting Up Push Notification Services

Using AWS Mobile Hub, you can enable a user push notification feature for your app. Push notification works with native platform support for sending push notifications, including Apple Push Notification service (APNs) for iOS apps and Google Cloud Messaging (GCM) for Android apps. Mobile Hub helps you

configure push notifications for APNs or iOS; however you also must set up push notifications with the platforms you plan to use.

The topics in this section detail the setup you must complete with push notification support for iOS and Android apps and obtain data values Mobile Hub needs to configure your push notification feature.

Topics

- [Setting Up iOS Push Notification \(p. 21\)](#)
- [Setting Up Android Push Notification \(p. 28\)](#)

Setting Up iOS Push Notification

Mobile Hub sends push notifications to iOS apps using Apple Push Notification service (APNs). To integrate this service with Mobile Hub, you must obtain and provide a certificate for APNs. To do this, you must prepare a certificate request, and then create an app ID and associated SSL certificate on the Apple Developer website. The certificate allows the Amazon Simple Notification Service server to send push notifications to the app identified by the App ID.

Topics

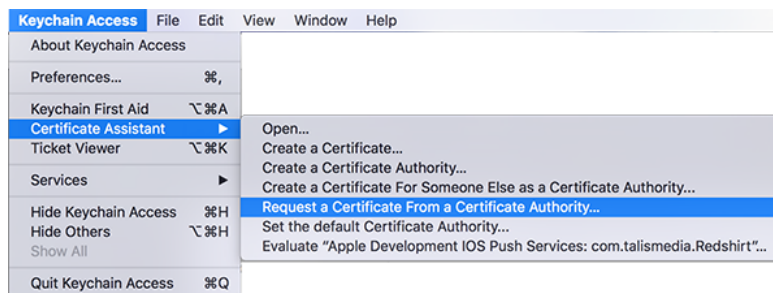
- [Generating a Certificate Request \(p. 21\)](#)
- [Setting up an App ID \(p. 22\)](#)
- [Configuring the App ID for Development Push Notifications \(p. 24\)](#)

Generating a Certificate Request

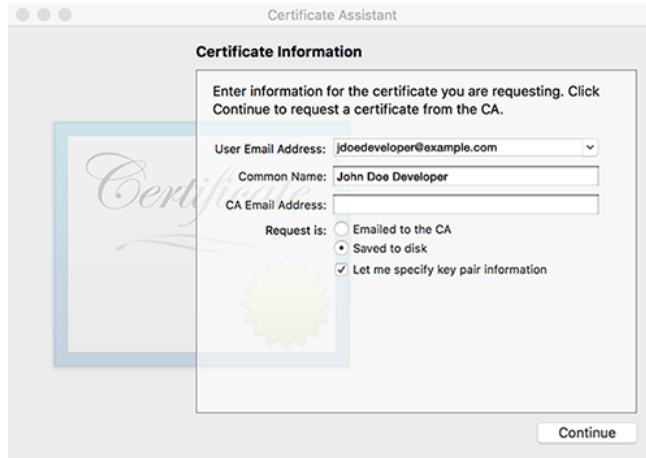
To create the certificate required by iOS to enable push notifications for your app, start by generating a certificate request on your Mac that you will use later to create the certificate.

To generate a certificate request

1. On your Mac, start the Keychain Access application.
2. From the **Keychain Access** menu, choose **Certificate Assistant** and then choose **Request a Certificate From a Certificate Authority...**



3. Enter your e-mail address and name.



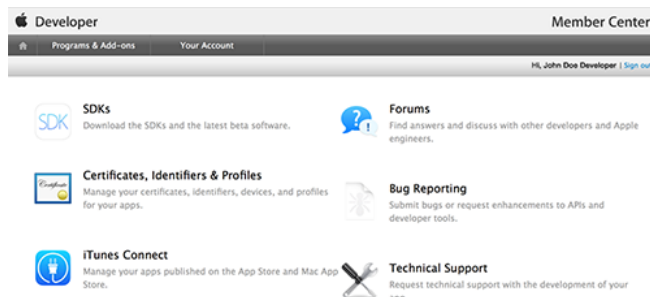
4. Select **Saved to disk** to create a file that contains the certificate request.

Setting up an App ID

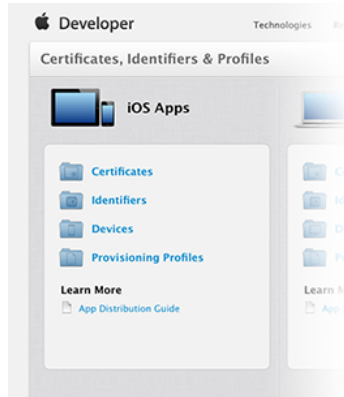
You need to provide an app ID for your app. Every app installed on a developer device needs an app ID. Typically, an app ID consists of a reversed web address, for example com.amazon.mysampleapp. You can use an existing app ID if it doesn't contain a wildcard character ("*").

To assign an app ID to your app

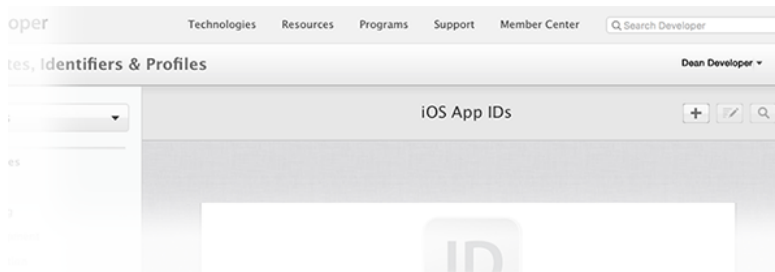
1. Sign into the Apple Developer Member Center website at <https://developer.apple.com/membercenter/index.action>.
2. Choose **Certificates, Identifiers & Profiles**.



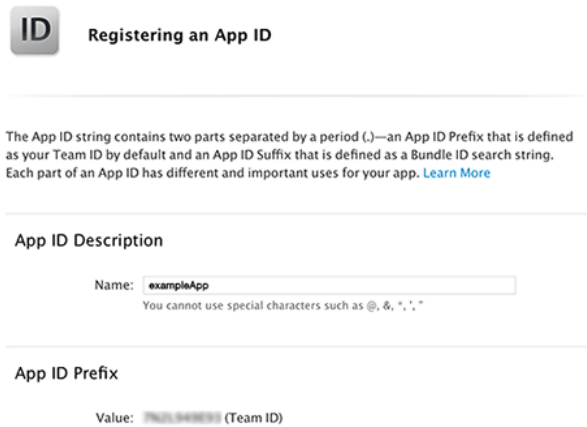
3. In the **iOS Apps** section, choose **Identifiers**.



4. At the top of the list of your iOS apps IDs, select +.



5. Type a name for the new app ID.



6. Choose the default selection for **App ID Prefix**.
7. For **App ID Suffix**, choose **Explicit App ID** and then enter the **Bundle ID** for your app. This ID must match the Bundle Identifier in your app's Info.plist.

App ID Suffix

Explicit App ID

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

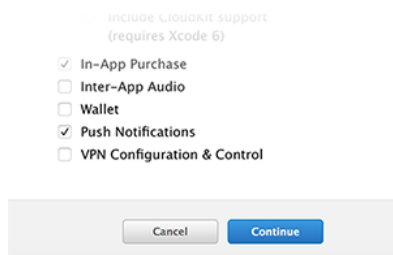
Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

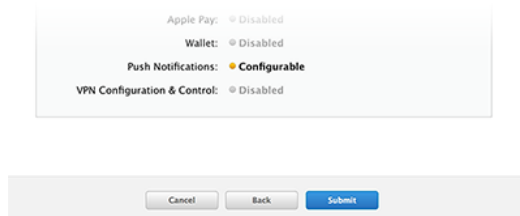
Wildcard App ID

This allows you to use a single App ID to match multiple apps. To create a wildcard App

8. Under **App Services** choose **Push Notification**.



9. Choose **Continue**. Check that all values were entered correctly. The identifier must match your app's bundle identifier and app ID prefix.



10. Choose **Submit** to register the new app ID.

Configuring the App ID for Development Push Notifications

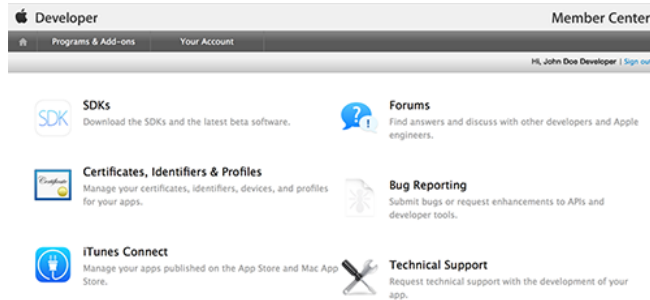
After creating a new app ID or choosing an existing explicit app ID, you must configure the app ID for push notifications.

To configure an app ID for push notifications

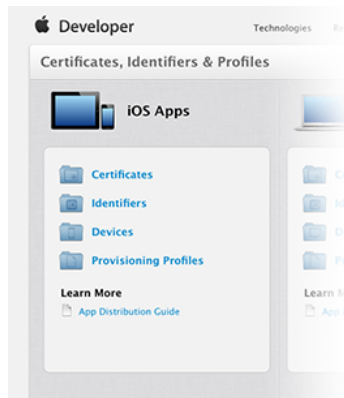
1. Sign into the Apple Developer Member Center website at <https://developer.apple.com/membercenter/index.action>.
2. Choose **Certificates, Identifiers & Profiles**.

AWS Mobile Hub Developer Guide

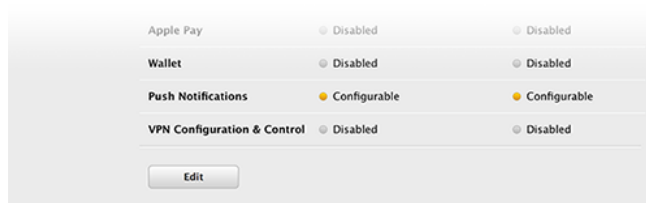
Setting Up iOS Push Notification



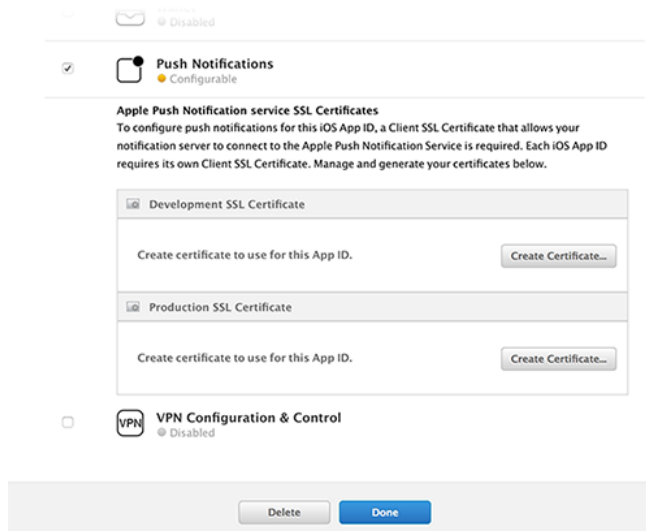
3. From the **iOS Apps** section, choose **Identifiers**.



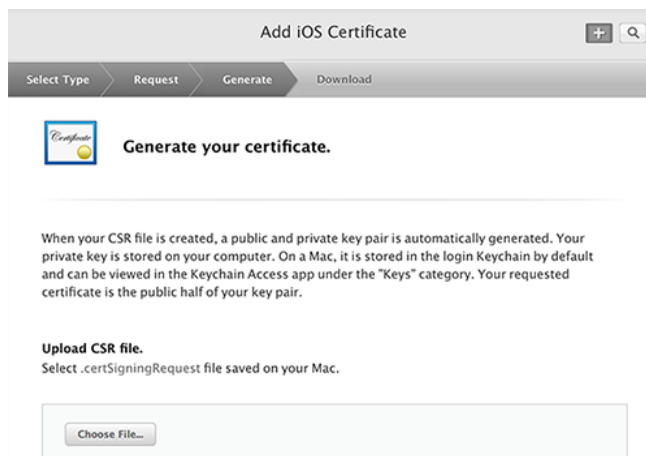
4. Select your newly created app ID from the list of iOS app IDs.
5. Choose **Edit**.



6. Under **Push Notifications** there are options to create a development SSL Certificate as well as a production SSL Certificate. Select **Create Certificate...** in the **Development SSL Certificate** section.



7. Choose **Continue** on the page that provides instructions on generating a Certificate Signing Request (CSR). This is the same certificate request created in [Generating a Certificate Request \(p. 21\)](#) and does not need to be created again.
8. In **Generate your certificate**, select **Choose File...** and then select the CSR file you created.



9. Choose **Generate**.
10. When the certificate is ready, choose **Download** and then save the certificate to your computer.

Download, Install and Backup

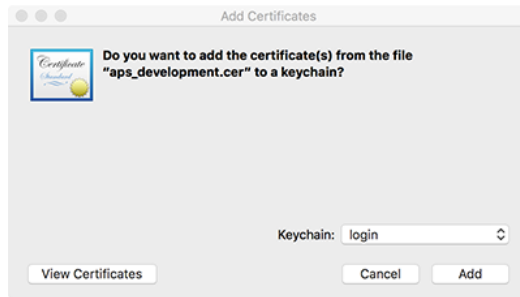
Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.



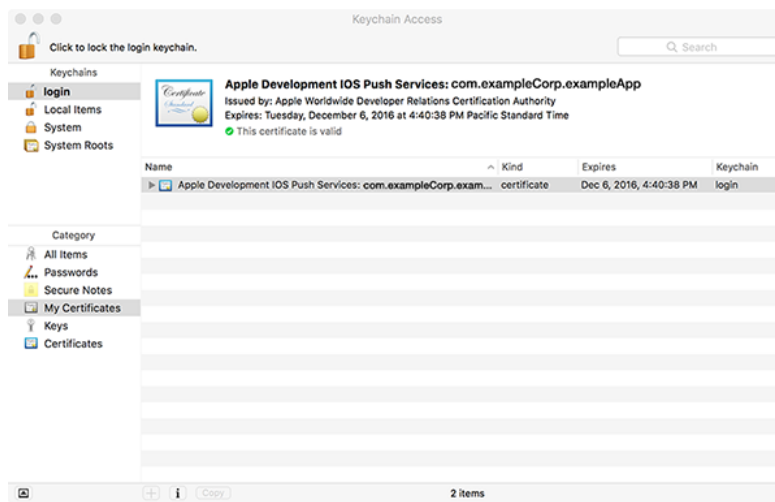
Documentation

For more information on using and managing your certificates read:

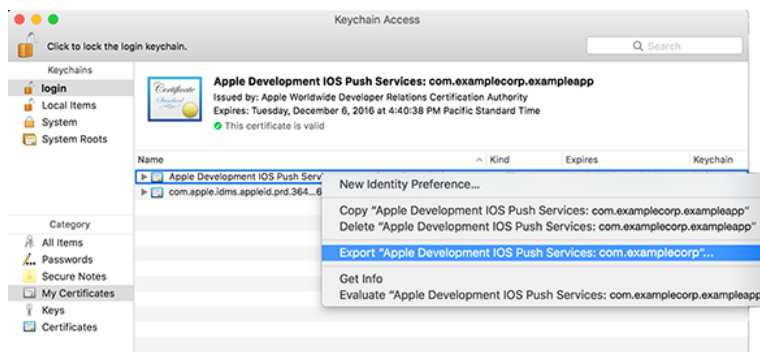
11. Double-click the downloaded certificate to install it to the Keychain on your Mac. In the **Add Certificates** dialog box, choose **Add**.



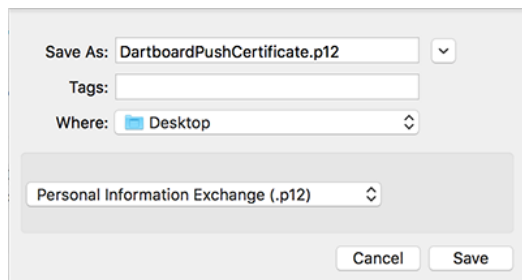
12. On your Mac, start the Keychain Access application.
13. In **My Certificates**, navigate to the certificate you just added. It should be called "Apple Development iOS Push Services:".



14. Context-select this certificate and then choose **Export...** from the context menu to export a file that contains the certificate.



15. Name the exported certificate "MobileHubPushCertificate.p12" and then save it to your computer. Do not provide an export password when prompted. You need to upload this certificate when creating your app in the Mobile Hub console.



Push notification is now enabled for your app in development mode. Prior to releasing your app on the App Store, you must repeat these steps but choose **Production Push SSL Certificate**.

Setting Up Android Push Notification

Mobile Hub sends push notifications to Android apps using Google Cloud Messaging (GCM). To use GCM, you must set it up for your app.

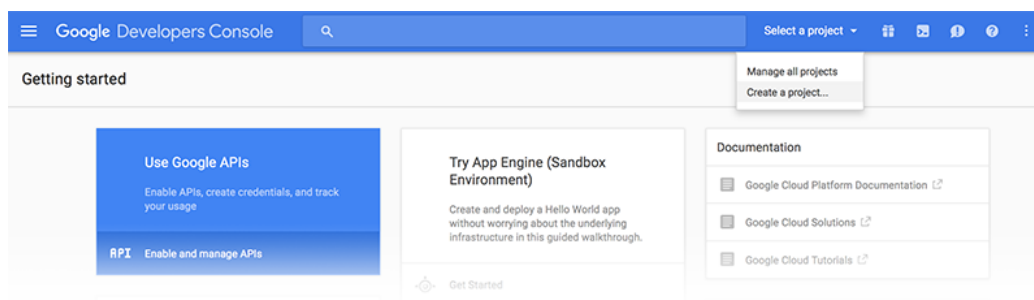
You will need the following:

- The app server's sender ID. This is a unique numerical value that is created when you configure your API project. The sender ID is used in the registration process to identify an app server that is permitted to send messages to the client app.
- A server API key. The key is saved on the app server that is authorized to access Google services.

For more information about GCM, see [Google Cloud Messaging: Overview](#).

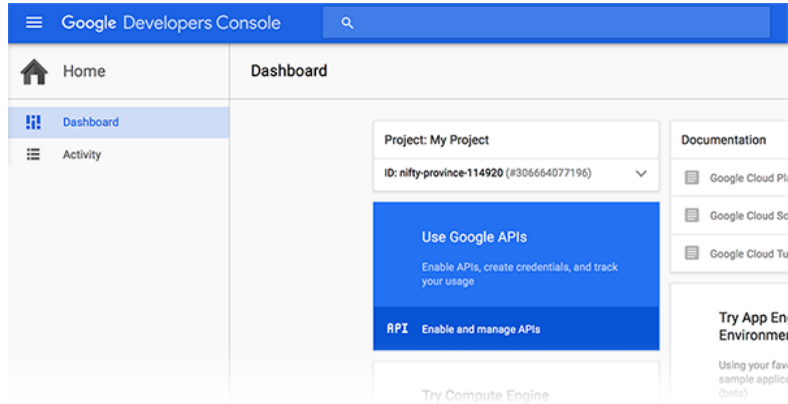
To set up Google Cloud Messaging

1. Go to the Google Developer Console at <https://console.developers.google.com>.
2. If you have not created a project yet, choose **Select a project** from the menu bar, and then choose **Create a project...**

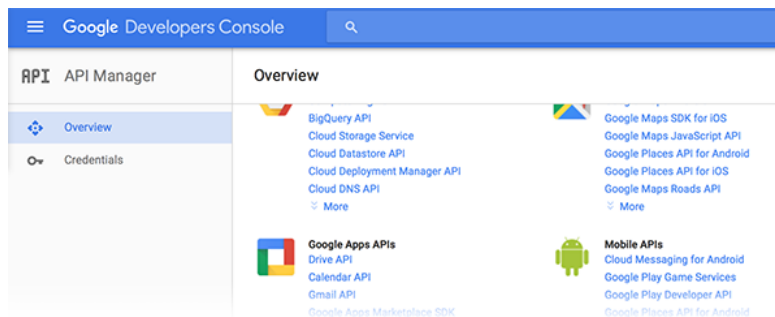


3. Complete the form displayed to create your new project.
4. In the **Dashboard** for your project, go to the **Use Google APIs** section and then choose **Enable and manage APIs**.

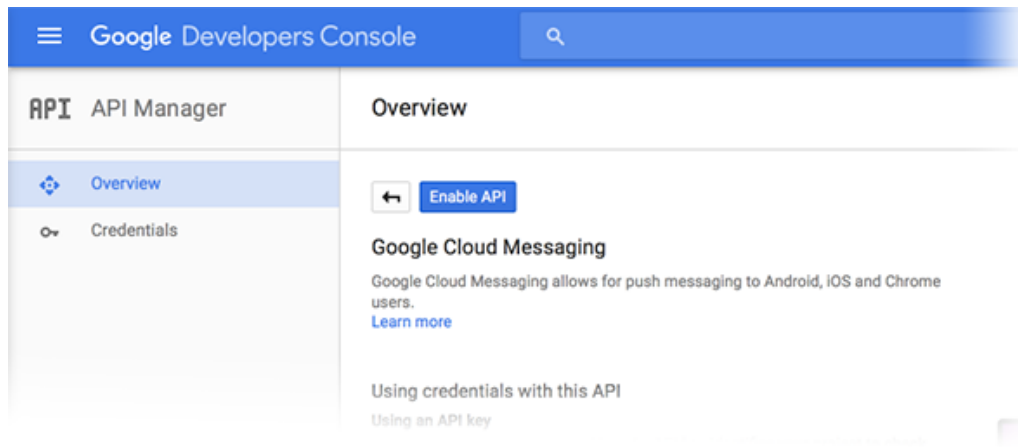
AWS Mobile Hub Developer Guide Setting Up Android Push Notification



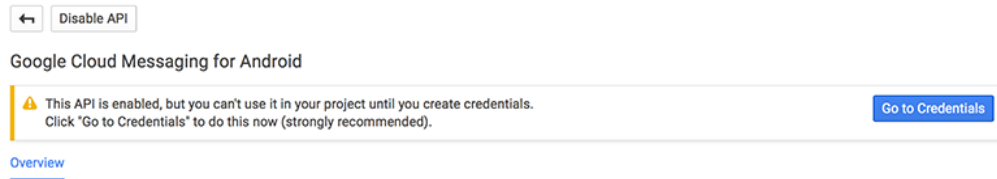
5. In the API Manager, find the **Mobile APIs** section and then choose **Cloud Messaging for Android**.



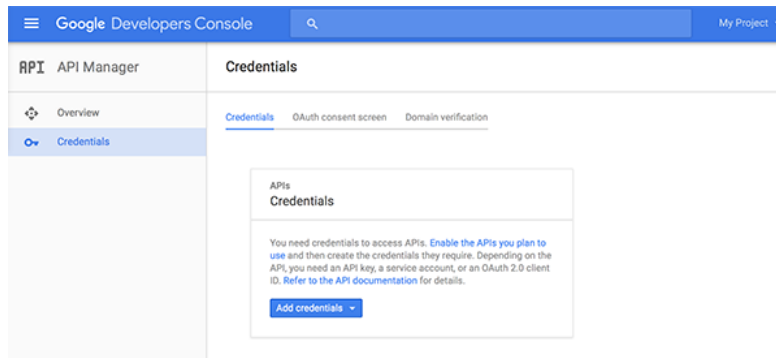
6. In the **Overview** for Cloud Messaging for Android, choose **Enable API**.



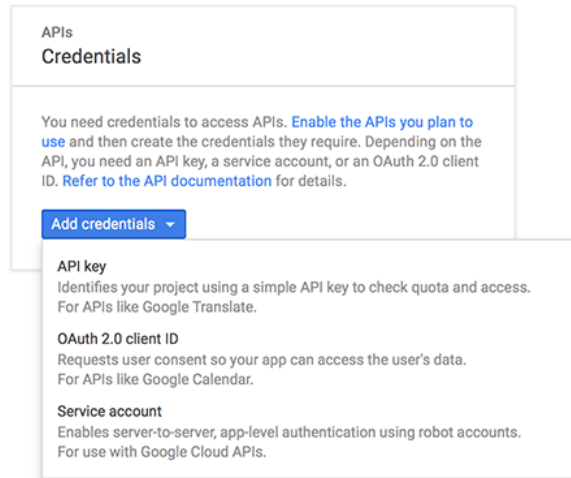
7. A message appears to inform you that the API is enabled but that it requires credentials before you can use it. Choose **Go to Credentials**.



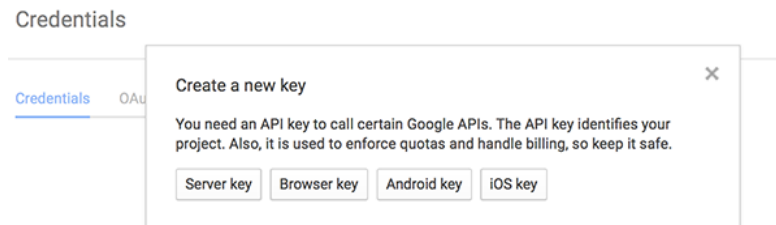
8. In **Credentials**, choose **Add credentials**.



9. From **Add credentials**, choose **API key**.



10. In **Create a new key**, choose **Server key**.



11. In **Create server API key**, type a name for the key and then choose **Create**.

Create server API key

This key should be kept secret on your server

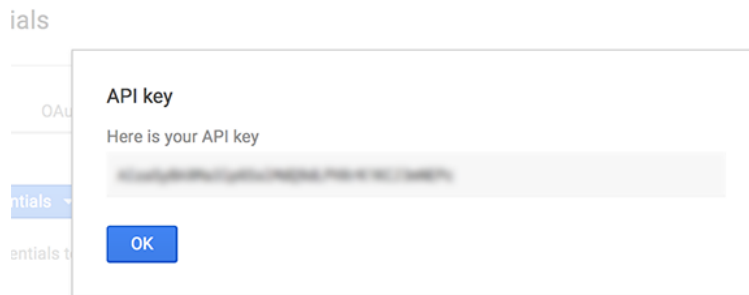
Every API request is generated by software running on a machine that you control. Per-user limits will be enforced using the address found in each request's userIp parameter, if specified. If the userIp parameter is missing, your machine's IP address will be used instead. [Learn more](#)

Name

Accept requests from these server IP addresses (Optional)

Examples: 192.168.0.1, 172.16.0.0/12, 2001:db8::1 or 2001:db8::/64

12. The API key is displayed. Save this key. You need it to register your app for push notifications. You supply this key to Mobile Hub when creating your app in the console.

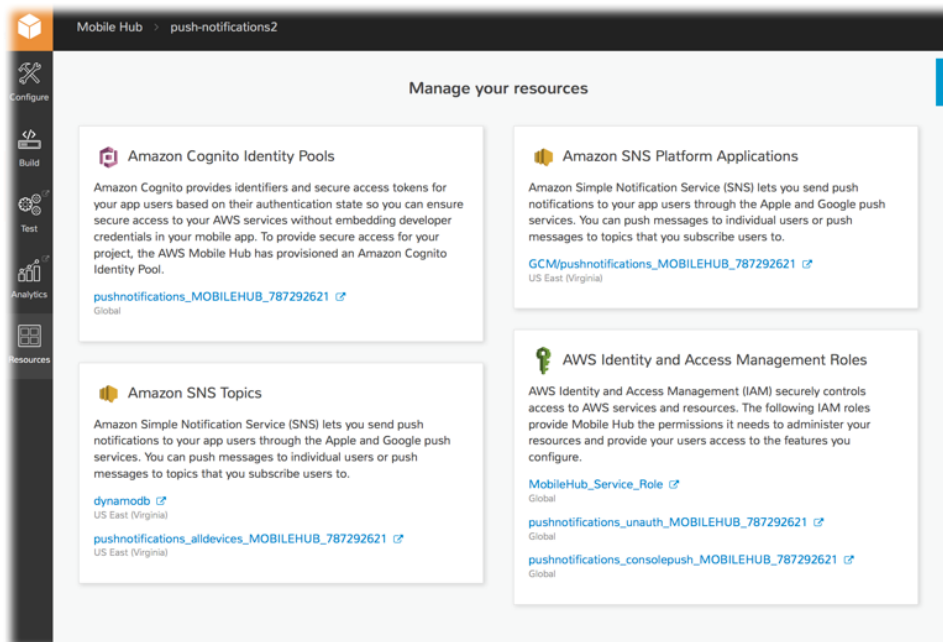


13. The sender ID is the project ID you used in the Google Developers Console to sign up. It is a numeric string. For example, in the following URI, the sender ID is 0123456789. You need to provide the sender ID for your app in the Mobile Hub console when configuring push notifications for Android apps.

```
https://code.google.com/apis/console/#project:0123456789
```

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the Push Notifications feature:



Quickstart App Details

Upon launch, the Push Notification demo in a quickstart app automatically registers the installed app for push notifications with the configured provider. The app obtains a push token from the provider and passes it to create a new platform endpoint in the Amazon SNS application created for your project.

In addition to creating the Amazon SNS platform application, Mobile Hub provisions the **ALL_DEVICES** topic that will be automatically subscribed to by the quickstart app. The IAM role is also modified to allow the quickstart app to create a platform endpoint and subscribe to the Amazon SNS topic. The demo also provides a list of all Amazon Amazon SNS topics that the device can subscribe to which includes the **ALL_DEVICES** topic and any others you configured while creating your project. Selecting/unselecting a topic from the list of topics subscribes or unsubscribes the device's Amazon SNS platform endpoint from that topic.

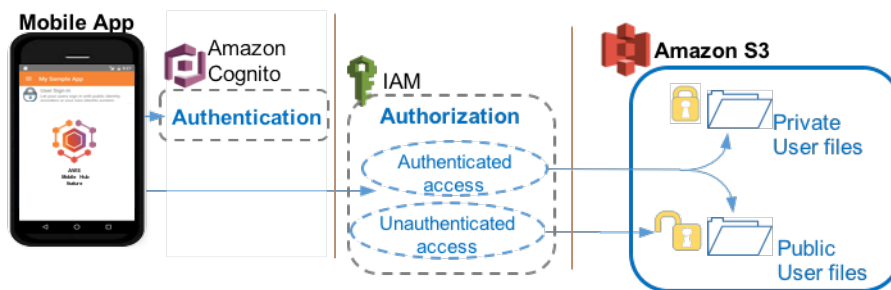
Creating Test Push Notifications for the Quickstart app with Amazon SNS

To demonstrate the Push Notification feature and send messages to your quickstart app, you can open the [Amazon SNS console](#), choose the application, select the endpoint you wish to notify, and choose **Publish to Endpoint**. If you want to test publishing notifications via SNS topic, choose **Topics**, select the topic, and choose **Publish to topic**.

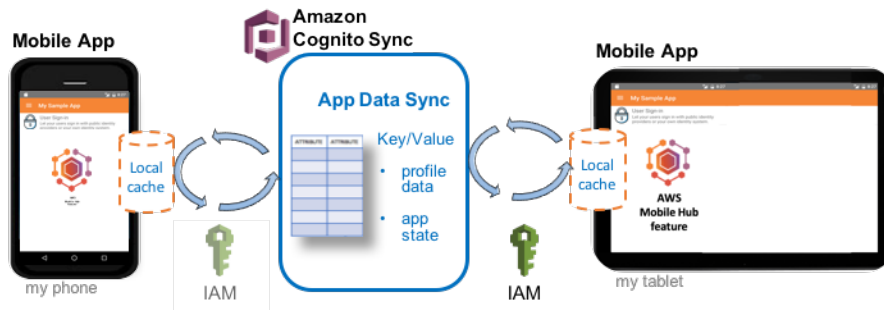
User Data Storage

Choose AWS Mobile Hub User Data Storage to add cloud storage of user files, profile data, and app state to your mobile app. This feature enables sync and caching of data between devices using a simple programming model.

The following image shows access policy enforcement for public and private user files.



The following image shows user profile data sync for persisting user data and synchronizing it across devices.



The User Data Storage feature enables you to store user files such as photos or documents in the cloud, and it also allows you to save user profile data in key/value pairs, such as app settings or game state. When you select this feature, an [Amazon S3](#) bucket is created as the place your app will store user files.

Mobile Hub will also configure [Amazon Cognito Sync](#) so you can save user profile data in key/value pairs and synchronize that data across a user's authenticated devices.

Topics

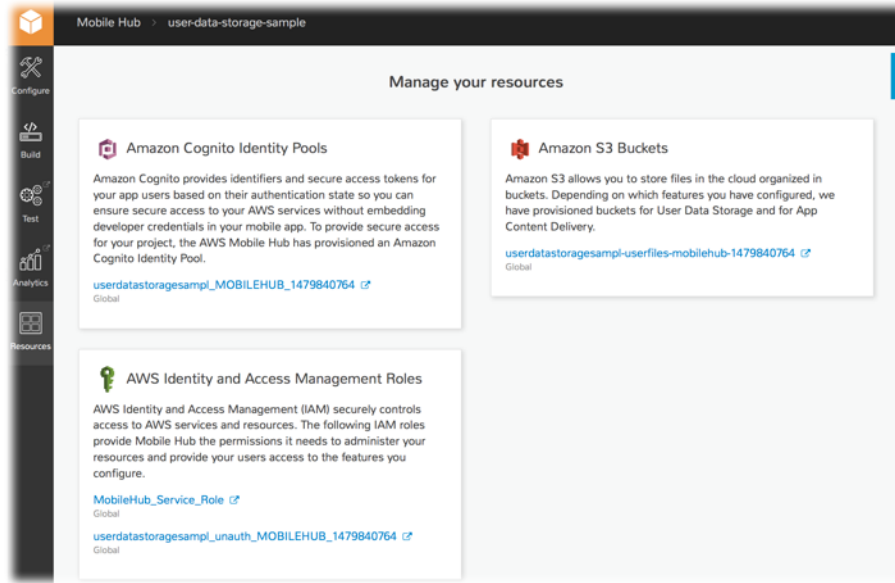
- [User Data Storage At a Glance \(p. 34\)](#)
- [Viewing AWS Resources Provisioned for this Feature \(p. 34\)](#)
- [Quickstart App Details \(p. 35\)](#)

User Data Storage At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon S3 bucket (see Amazon Simple Storage Service Getting Started Guide) Concepts Console Pricing• Amazon Cognito Sync (see Amazon Cognito Sync) Concepts Console Pricing <p>Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 36).</p> <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 34).</p>
Configuration options	<p>This feature enables the following configuration options mobile backend capabilities:</p> <ul style="list-style-type: none">• Store user files and app data using Amazon S3.• Synchronize data to the cloud and between a user's devices using Amazon Cognito Sync.
Quickstart demo features	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• File explorer for accessing sample files in the public and private folders in your S3 bucket.• User settings sync persists user's choice of color theme to the cloud.

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the User Data Storage feature.



Quickstart App Details

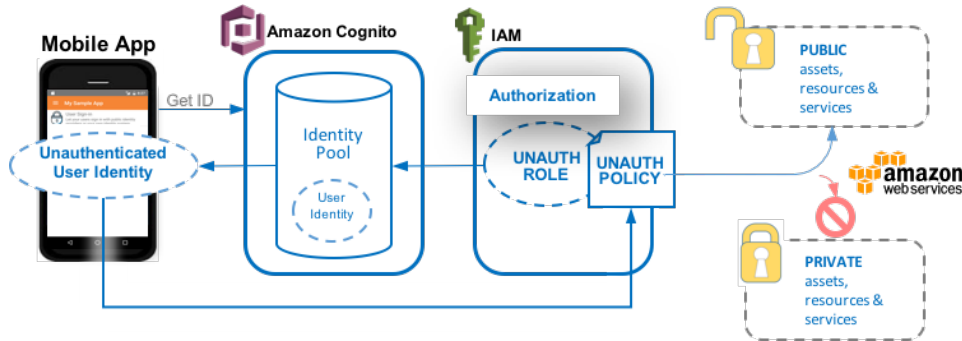
In the Mobile Hub quickstart app, the User Data Storage demo enables all users to see the contents of a public folder. When this feature is used in combination with User Sign-in, users who are signed in are able to access a private folder; unauthenticated users are not.

The demo also includes an option for the user to change the color scheme of the app. That choice is stored in Amazon Cognito Sync Profile. Any time the user returns to the app, their chosen theme is loaded from the stored user profile. If the user is authenticated, the same theme can sync across all devices they own.

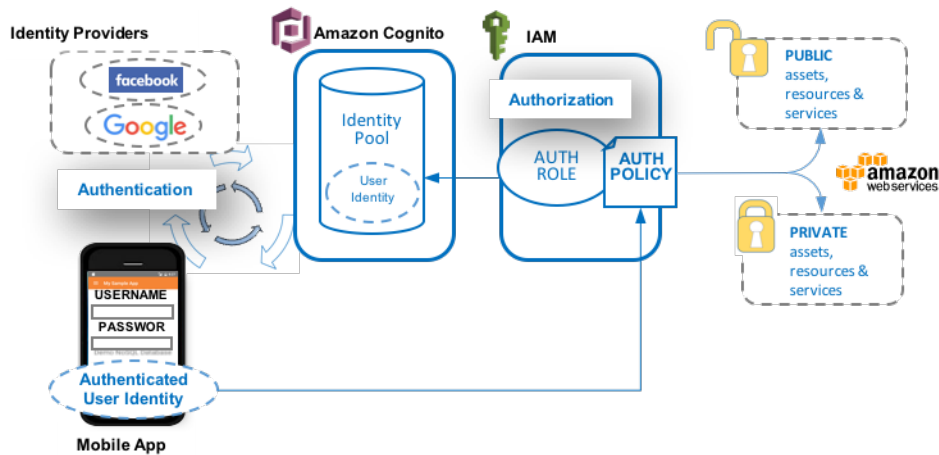
User Sign-in

Choose the AWS Mobile Hub User Sign-in mobile backend feature to add AWS user authentication and identity access management to your mobile app, including unauthenticated access and integration of identity providers like Facebook or Google.

The following image shows a resource access policy being enforced for an unauthenticated user.



The following image shows a resource access policy being enforced for an authenticated user.



This feature enables you to configure how your users gain access to AWS resources and services used by your app, either with no sign in process or through authentication provided by one or more identity

providers. In both cases, AWS identity creation and credentials are provided by [Amazon Cognito Identity](#), and access authorization comes through [AWS Identity and Access Management \(IAM\)](#).

When you create a project, Mobile Hub provisions the AWS identity, user role, and access policy configuration required to allow all users access to unrestricted resources. When you add the User Sign-in feature to your app, you are able to restrict access to allow only those who sign in with credentials validated by an identity provider to use protected resources. Through Amazon Cognito Identity, your app user obtains AWS credentials to directly access the AWS services that you have enabled and configured for your Mobile Hub project. Both authenticated and unauthenticated users are granted temporary, limited-privilege credentials with the same level of security enforcement.

Amazon Cognito can federate validated user identities from multiple identity providers to a single AWS identity. AWS Mobile Hub helps you integrate identity providers in your mobile app so that users can sign in using their existing credentials from Facebook, Google, user pools in Amazon Cognito Identity, and your own identity system.

Topics

- [User Sign-in Feature At a Glance \(p. 37\)](#)
- [Configuring User Sign-in \(p. 38\)](#)
- [Setting Up User Authentication \(p. 38\)](#)
- [Viewing AWS Resources Provisioned for this Feature \(p. 55\)](#)
- [Quickstart App Details \(p. 55\)](#)

User Sign-in Feature At a Glance

AWS services and resources configured	<ul style="list-style-type: none">• Amazon Cognito Identity Pool (see Using Federated Identities) Concepts Console Pricing• IAM role and security policies (see IAM Managed Policies and Service Roles (p. 59)) Concepts Console Pricing <p>For more information, see Viewing AWS Resources Provisioned for this Feature (p. 55).</p>
Configuration options	<p>This feature enables the following mobile backend capabilities:</p> <p>No Sign-in (unauthenticated access)</p> <p>Required Sign-in (authenticated access)</p> <ul style="list-style-type: none">• via Google authentication (see Setting Up Google Authentication (p. 41))• via Facebook authentication (see Setting Up Facebook Authentication (p. 39))• via Custom authentication (see Developer Authenticated Identities) <p>Optional Sign-in (users gain greater access when they sign in)</p> <p>For more information, see Configuring User Sign-in (p. 38)</p>

Quickstart demo features	<p>This feature adds the following to a quickstart app generated by Mobile Hub:</p> <ul style="list-style-type: none">• Unauthenticated access (if allowed by your app's configuration), displaying the ID that AWS assigns to the app instance's device.• Sign-in screen that authenticates users using the selected method: Facebook, Google, or Custom.• With Optional Sign-in and Require Sign-in, the app demonstrates an access barrier to protected folders for unauthenticated users.
---------------------------------	---

Configuring User Sign-in

The following options are available for configuring your users' sign-in experience.

No sign-in

Users do not sign in. Instead, your app receives temporary, limited privilege access credentials from Amazon Cognito Identity as unauthenticated guest users so that your app can access your AWS services securely. The device retains an Amazon Cognito token in memory that makes it a recognizable entity, but that identity cannot be associated with other owned devices for synchronization or other purposes.

Sign-in is optional

Users have the option to sign in (authenticate) with your chosen sign-in identity provider(s) or users can skip sign-in (unauthenticated). Your app receives temporary, limited privilege access credentials from Amazon Cognito Identity as either authenticated users or unauthenticated guest users so that your app can access your AWS services securely.

Sign-in is required

Users are required to sign in with one of your chosen sign-in providers. Your app receives temporary, limited privilege access credentials from Amazon Cognito Identity as authenticated users so that your app can access your AWS services securely.

User Sign-in and AWS Identity and Access Management (IAM)

When your mobile app is saved, Mobile Hub creates an Amazon Cognito identity pool and a new IAM role that is used to generate temporary AWS credentials for the quickstart app users to access your AWS resources. The AWS IAM role security policies are updated based on the sign-in features enabled.

At this point, your mobile project is set up for users to sign in. Each chosen identity provider has been added to the login screen of the quickstart app.

For more information, see [IAM \(Identity and Access Management\) Managed Policies](#) (p. 59).

Setting Up User Authentication

With AWS Mobile Hub, you can enable a user sign-in feature for your app. User sign-in works with various user authentication services, including Facebook, Google, and custom authentication. Mobile Hub helps you to configure sign-in with Facebook, Google, or your own identity system; however, you may also need to set up user authentication with the different authentication services you plan to use.

Learn more about how Amazon Cognito performs authentication using external identity providers, see [Understanding Amazon Cognito Authentication](#).

The topics in this section detail the setup you must complete with various user authentication services and how to obtain the data values Mobile Hub needs to configure your sign-in feature.

Topics

- [Setting Up Facebook Authentication](#) (p. 39)
- [Setting Up Google Authentication](#) (p. 41)
- [Setting Up Custom Authentication](#) (p. 55)

Setting Up Facebook Authentication

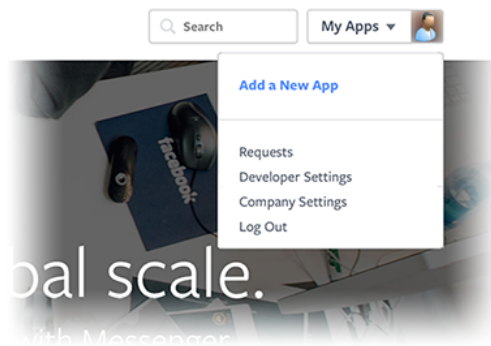
You must first register your application with Facebook by using the [Facebook Developers portal](#).

Mobile Hub generates code that enables you to use Facebook to provide federated authentication for your mobile app users. This topic explains how to set up Facebook as an identity provider for your app.

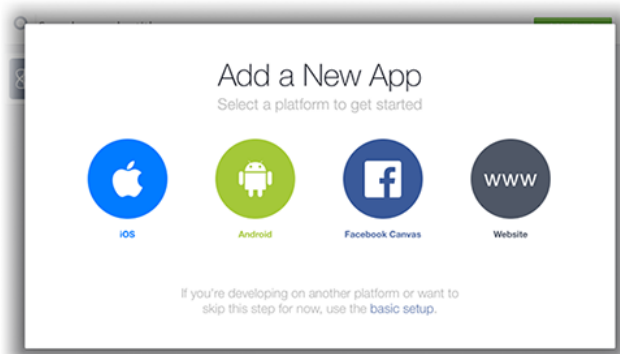
If you already have a Facebook app ID, copy and paste it into the **Facebook App ID** field in the Mobile Hub console, and choose **Save changes**.

To get a Facebook app ID

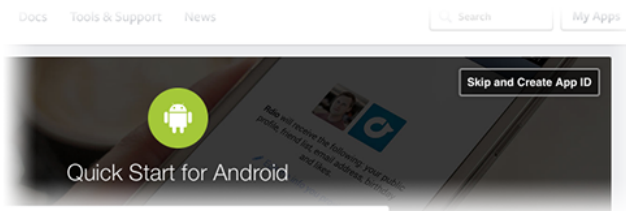
1. In the [Facebook Developers portal](#), sign in with your Facebook credentials.
2. From **My Apps**, choose **Add a New App**.



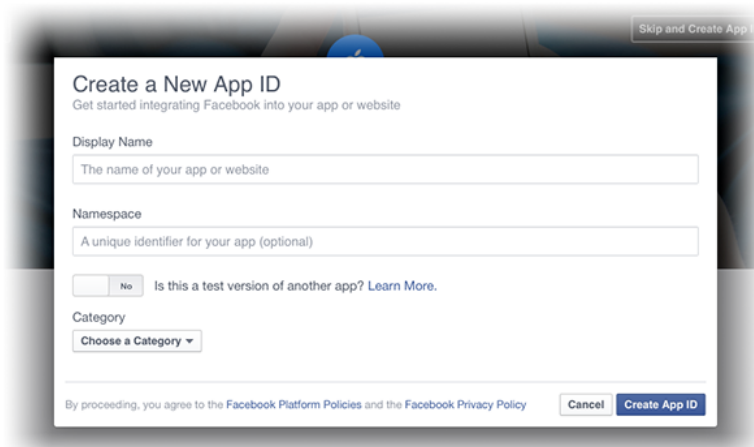
3. In **Add a New App**, choose **iOS** to create an app ID for an iOS app, or choose **Android** to create an app ID for an Android app. If your app will support both iOS and Android platforms, select either iOS or Android since you can use the same app ID across platforms.



4. In the Quick Start page for the platform you selected, choose **Skip and Create App ID**.



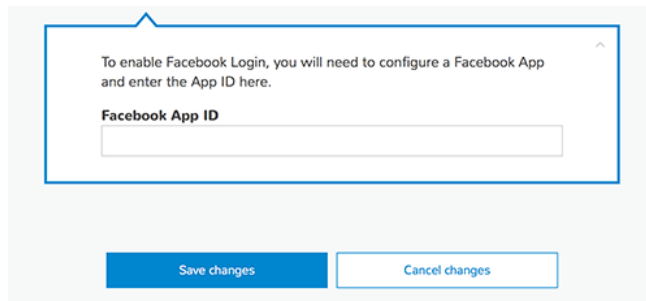
5. Type a display name for your app, select a category for your app from the **Category** drop-down list, and then choose **Create App ID**.



6. Complete the **Security Check** that appears. Your new app then appears in the **Dashboard**.



7. Copy the App ID and paste it into the **Facebook App ID** field in the Mobile Hub console.



8. Choose **Save changes**.

For more information about integrating with Facebook Login, see the [Facebook Getting Started Guide](#).

Setting Up Google Authentication

With AWS Mobile Hub, you can configure a working Google Sign-In feature for both Android and iOS apps. To fully integrate Google Sign-In with your sample app, Mobile Hub needs information that you must first obtain through Google's setup process. The process has several parts, one of which is required regardless of which platforms you're supporting with your app. There are other parts to complete only for specific platforms:

- Create a Google Developers project and an OAuth Web Application Client ID (required for **all apps** regardless of platform)
- Create an OAuth Android client ID (required for all **Android apps**)
- Create an OAuth iOS client ID (required for all **iOS apps**)

This section details the Google Sign-In requirements as well as how to integrate Google Sign-In for both iOS and Android apps.

Topics

- [Creating a Google Developers Project and OAuth Web Client ID \(p. 41\)](#)
- [Creating an OAuth Android Client ID \(p. 47\)](#)
- [Creating an OAuth iOS Client ID \(p. 51\)](#)
- [Verifying All Platform Client IDs \(p. 54\)](#)

Creating a Google Developers Project and OAuth Web Client ID

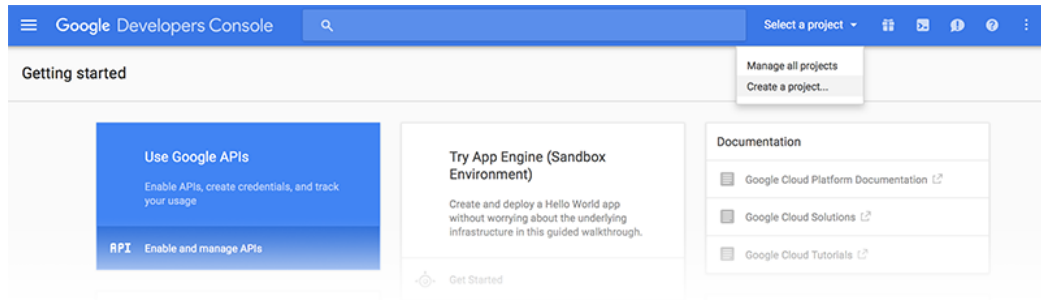
Before you enable Google Sign-In in an app, you must create a project in the Google Developers Console. Google recommends using a single project to create and manage all of the platform instances of your app, such as iOS, Android, and web.

Each platform requires its own OAuth client ID, which you obtain through the project you create for your app in the Google Developers Console. The first thing you must do is create a project for your app in the Google Developers Console that has the Google+ API enabled, and then enable an OAuth web client ID that Amazon Cognito uses to enable user authentication for your app.

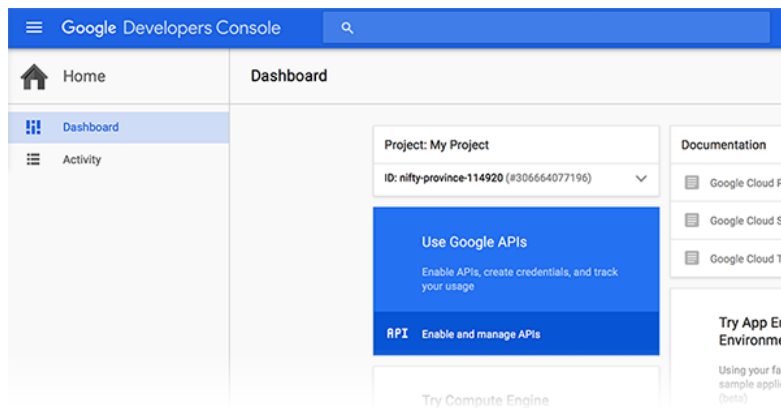
To create a Google Developers project and OAuth web client ID

1. Go to the Google Developers Console at <https://console.developers.google.com>.

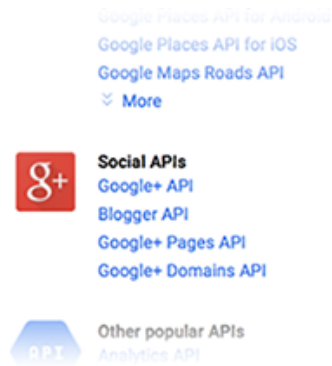
2. If you have not created a project yet, choose **Select a project** from the menu bar, and then choose **Create a project...**



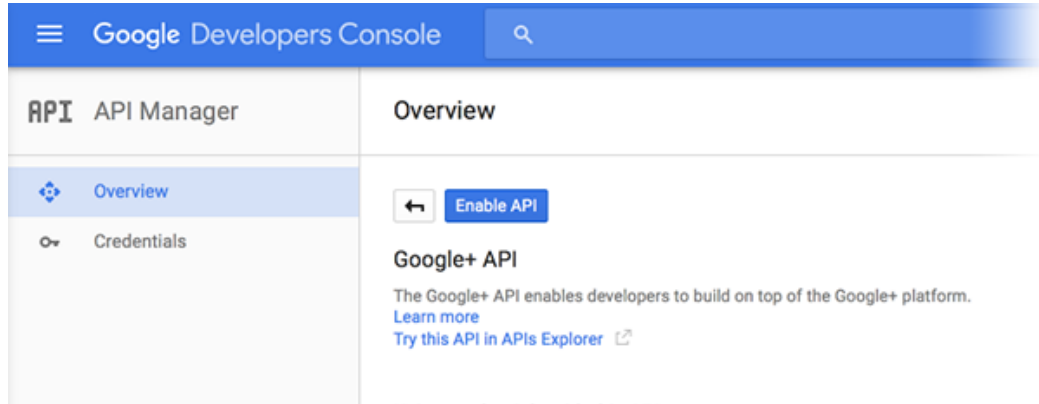
3. Complete the form that is displayed to create your new project.
4. In the **Dashboard** for your project, go to the **Use Google APIs** section and then choose **Enable and manage APIs**.



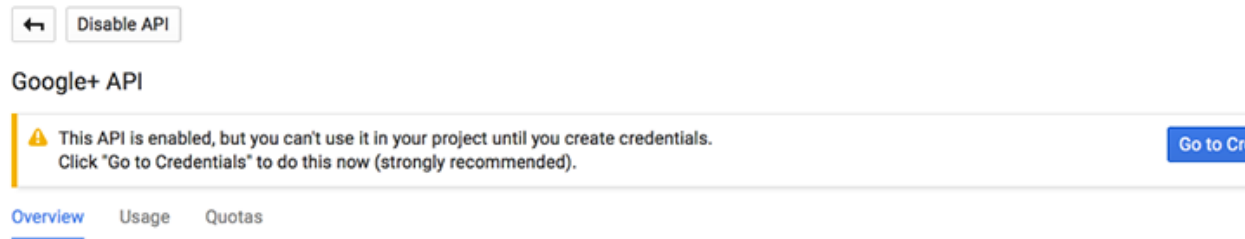
5. In the API Manager, in the **Social APIs** section, choose **Google+ API**.



6. In the **Overview** for Google+ API, choose **Enable API**.



7. A message appears to inform you that the API is enabled but that it requires credentials before you can use it. Choose **Go to Credentials**.



8. Your Mobile Hub sample app authenticates users through Amazon Cognito Identity, so you need an OAuth web application client ID for Amazon Cognito. In **Credentials**, choose **client ID** from the links in the first step.

Credentials

Add credentials to your project

1 Find out what kind of credentials you need

We'll help you set up the correct credentials

If you wish you can skip this step and create an [API key](#), [client ID](#), or [service account](#)

Which API are you using?

Determines what kind of credentials you need.

Google+ API

Where will you be calling the API from?


Determines which settings you'll need to configure.

9. A message appears to inform you that you must set a product name. Choose **Configure consent screen**.

Credentials



Create client ID

 To create an OAuth client ID, you must first set a product name on the consent screen

[Configure consent screen](#)

Application type

- Web application
- Android [Learn more](#)
- Chrome App [Learn more](#)
- iOS [Learn more](#)
- PlayStation 4
- Other

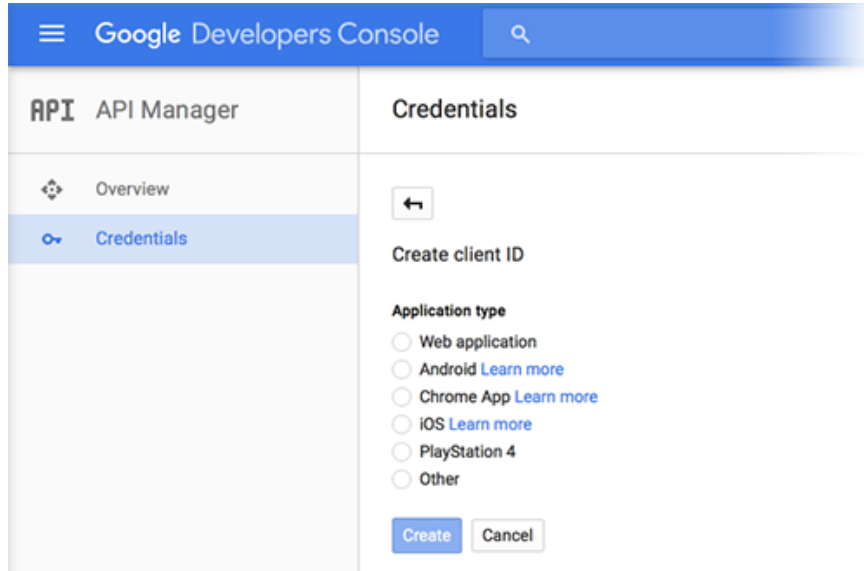
10. In **OAuth consent screen**, enter the name of your app in **Product name shown to users**. Leave the remaining fields blank. Then choose **Save**.

The screenshot shows the AWS IAM console interface for configuring an OAuth consent screen. The page title is 'Credentials' and the active tab is 'OAuth consent screen'. The form contains the following fields and elements:

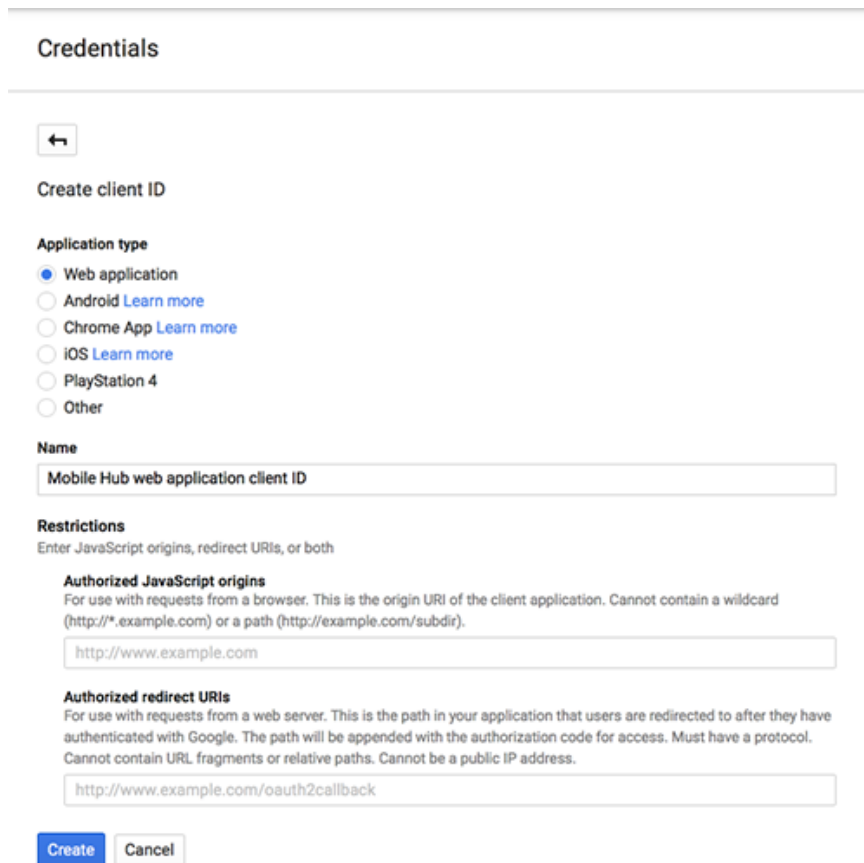
- Email address:** A dropdown menu showing an email address ending in '@gmail.com'.
- Product name shown to users:** A text input field containing 'Mobile Hub Sample App'.
- Homepage URL (Optional):** An empty text input field.
- Product logo URL (Optional):** A text input field containing 'http://www.example.com/logo.png'.
- Product logo:** A placeholder box with the text 'This is how your logo will look to end users' and 'Max size: 120x120 px'.
- Privacy policy URL (Optional):** An empty text input field.
- Terms of service URL (Optional):** An empty text input field.
- Buttons:** 'Save' and 'Cancel' buttons at the bottom left.

On the right side of the form, there is a laptop icon and a note: 'The consent screen is shown to users when they log in to their private account. It will be used for all applications in the project. You must provide a privacy policy and product logo to work.' Below this note, there is a partially visible 'Save' button.

11. In **Create client ID**, choose **Web application**.



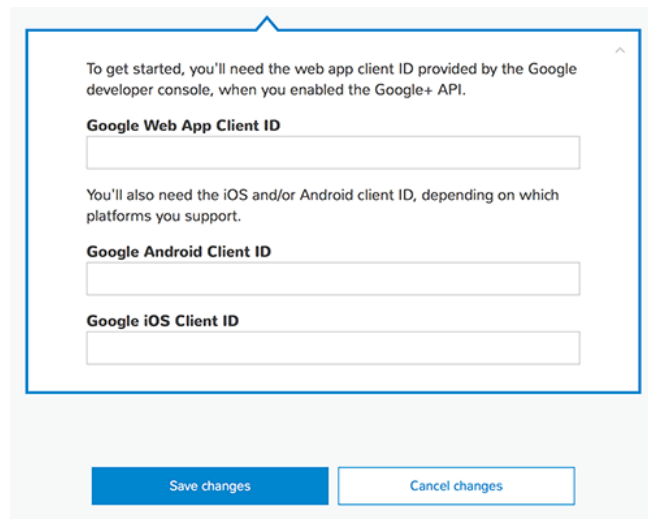
12. In **Name**, enter a name for the web client credentials for your app. Leave the **Authorized JavaScript origins** and **Authorized Redirect URIs** fields blank. Mobile Hub configures this information indirectly through Amazon Cognito Identity integration. Choose **Create**.



13. In the **OAuth client** pop-up, copy and save the value that was generated for your client ID. You will need the client ID to implement Google Sign-In in your Mobile Hub app. After you copy the client ID, choose **OK**.



14. Paste the web application client ID value into the Mobile Hub **Google Web App Client ID** field for your project.



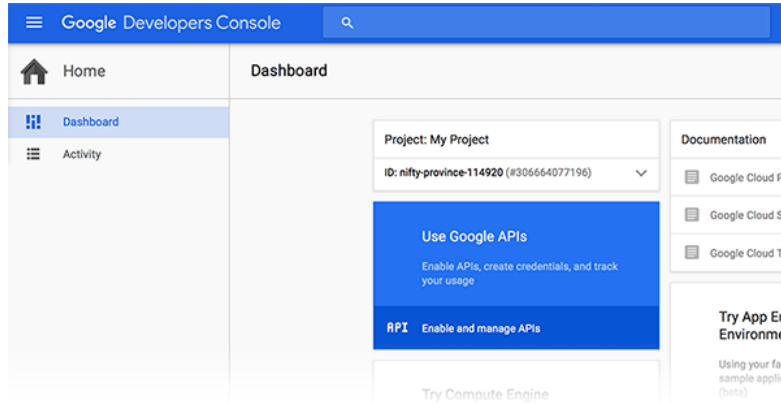
Creating an OAuth Android Client ID

To enable Google Sign-In for your Android Mobile Hub sample app, you must create an Android OAuth client ID so that the Android sample app that is generated by Mobile Hub can access Google APIs directly and manage token lifecycle through Amazon Cognito Identity. This Android OAuth client ID is in addition to the Web application OAuth client ID you created while [Creating a Google Developers Project and OAuth Web Client ID](#) (p. 41).

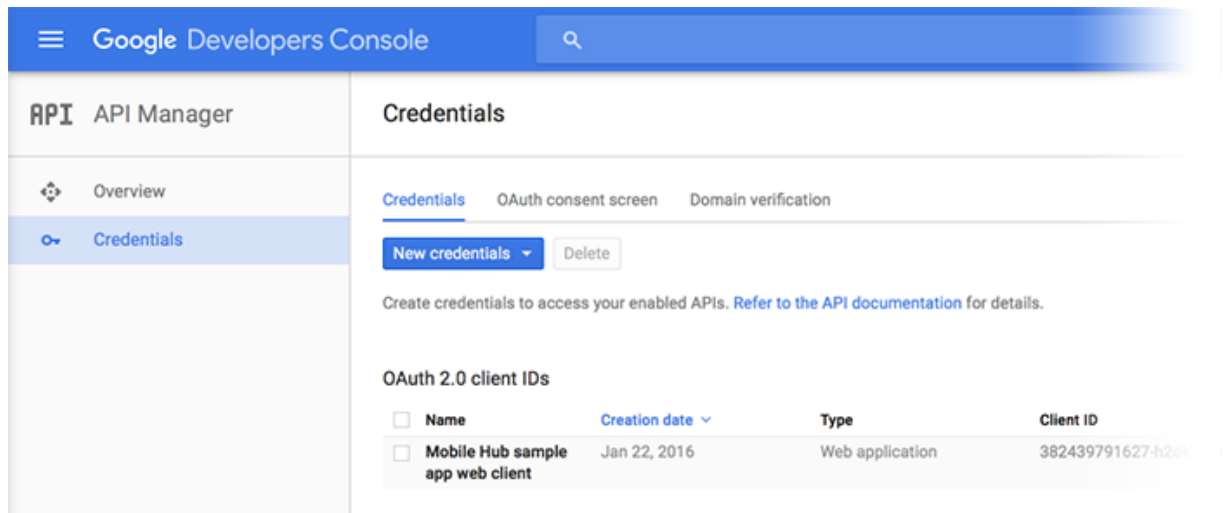
To integrate Google Sign-In for your Android sample app, you must generate an Android OAuth client ID in the Google Developers Console. You will provide this client ID to Mobile Hub during the Google Sign-In configuration.

To create an OAuth Android client ID

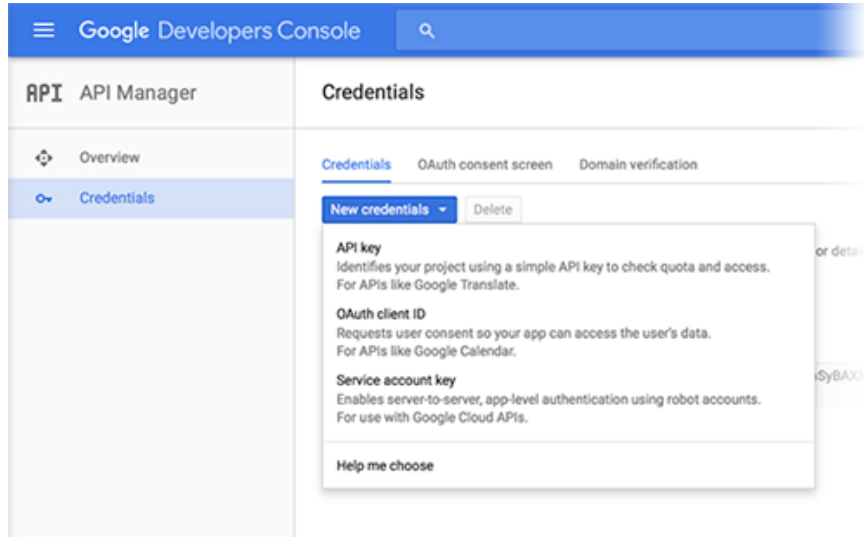
1. Go to the Google Developers Console at <https://console.developers.google.com>.
2. In the **Dashboard** for your project, go to the **Use Google APIs** section and then choose **Enable and manage APIs**.



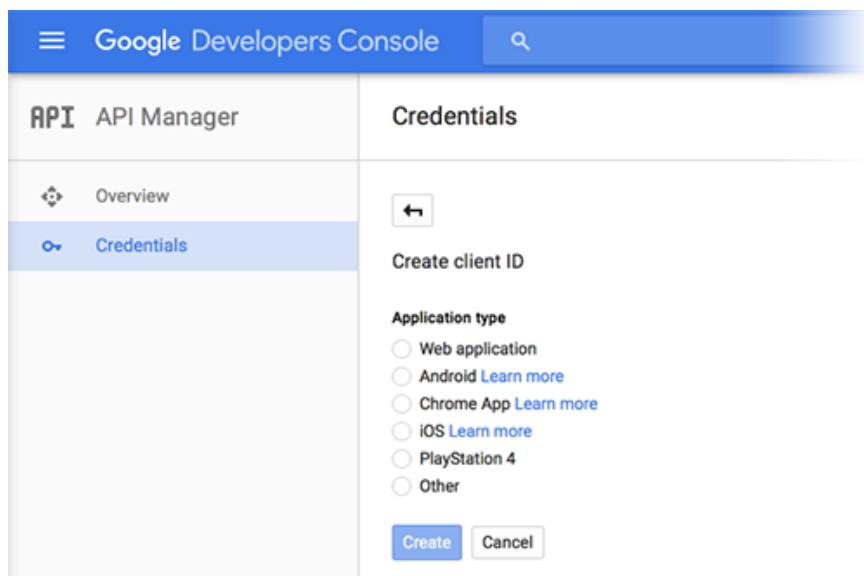
3. In the API Manager, choose **Credentials** in the left side menu.



4. Choose **New credentials** and then choose **OAuth client ID**.



5. In **Create client ID**, choose **Android**.



6. In **Name**, enter a name in the format *com.amazon.mysampleapp Android client ID*.
7. In **Signing-certificate fingerprint**, enter the SHA-1 fingerprint. For more information about Google's process for obtaining your SHA-1 fingerprint, see [this Google support article](#).

Credentials

←

Create client ID

Application type

Web application

Android [Learn more](#)

Chrome App [Learn more](#)

iOS [Learn more](#)

PlayStation 4

Other

Name

com.amazon.mysampleapp Android client ID

Signing-certificate fingerprint

Android devices send API requests directly to Google. Google verifies that each request comes from an Android app that matches a package name and SHA-1 signing-certificate fingerprint that you provide. Use the following command to get the fingerprint. [Learn more](#)

```
keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -list -v
```

00:00

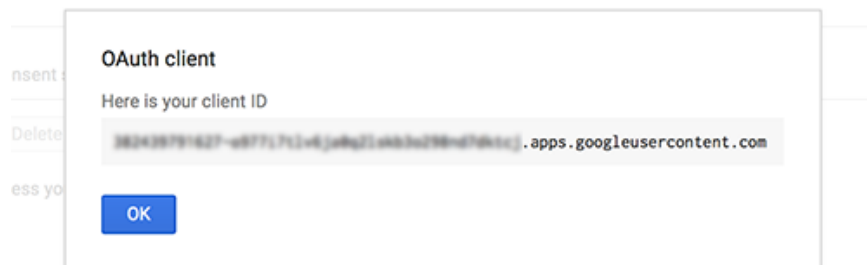
Package name

From your AndroidManifest.xml file

com.amazon.mysampleapp

Create Cancel

- In **Package name**, enter the package name in the format `com.amazon.mysampleapp`.
- Choose **Create**.
- In the **OAuth client** pop-up, copy and save the value generated for your Android client ID. You will need this client ID to implement Google Sign-In in your Mobile Hub app. After you copy the client ID, choose **OK**.



- Paste the Android client ID value into the Mobile Hub **Google Android Client ID** field for your project.

To get started, you'll need the web app client ID provided by the Google developer console, when you enabled the Google+ API.

Google Web App Client ID

You'll also need the iOS and/or Android client ID, depending on which platforms you support.

Google Android Client ID

Google iOS Client ID

Save changes **Cancel changes**

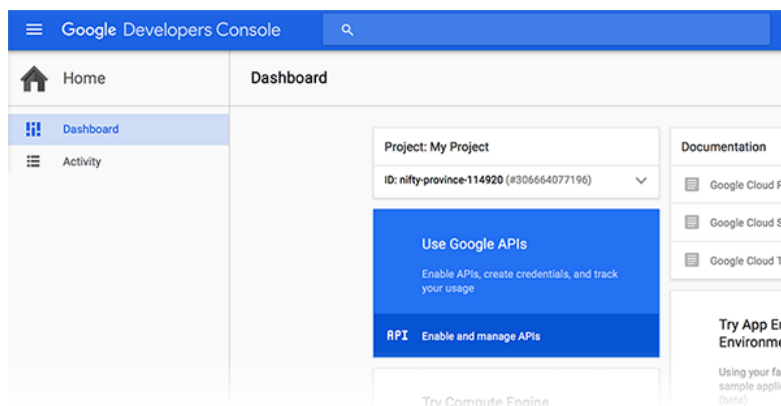
Creating an OAuth iOS Client ID

To enable Google Sign-In for your iOS Mobile Hub sample app, you must create an iOS OAuth client ID for your app. This enables your Mobile Hub sample app to access Google APIs directly and to manage token lifecycle through Amazon Cognito Identity. This iOS OAuth client ID is in addition to the web application OAuth client ID that you created while [Creating a Google Developers Project and OAuth Web Client ID](#) (p. 41).

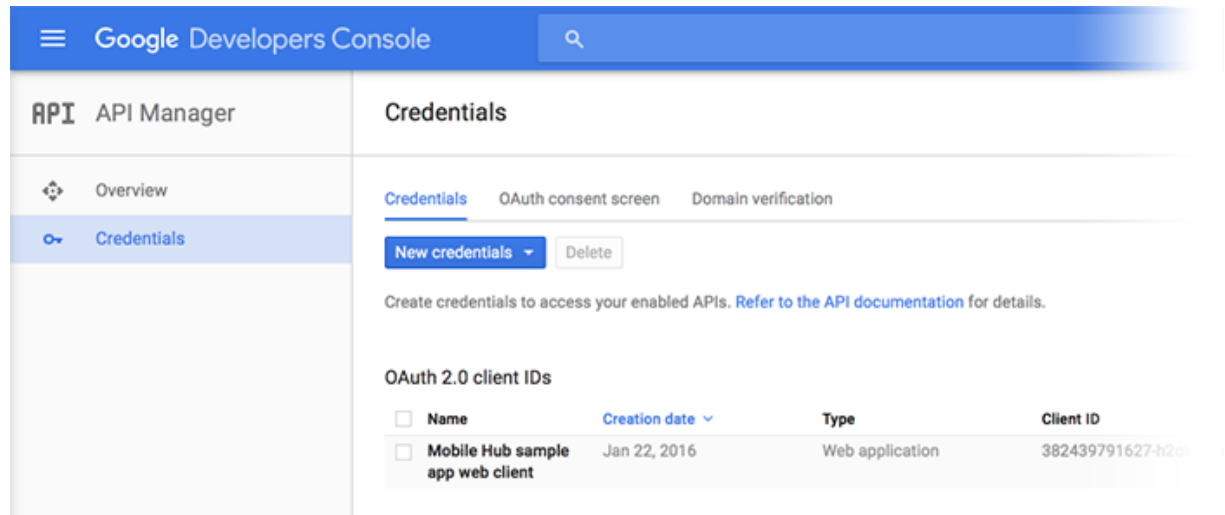
To integrate Google Sign-In for your iOS sample app, you must generate an iOS OAuth client ID in the Google Developers Console. You will provide this client ID to Mobile Hub during the Google Sign-In configuration.

To create an OAuth iOS client ID

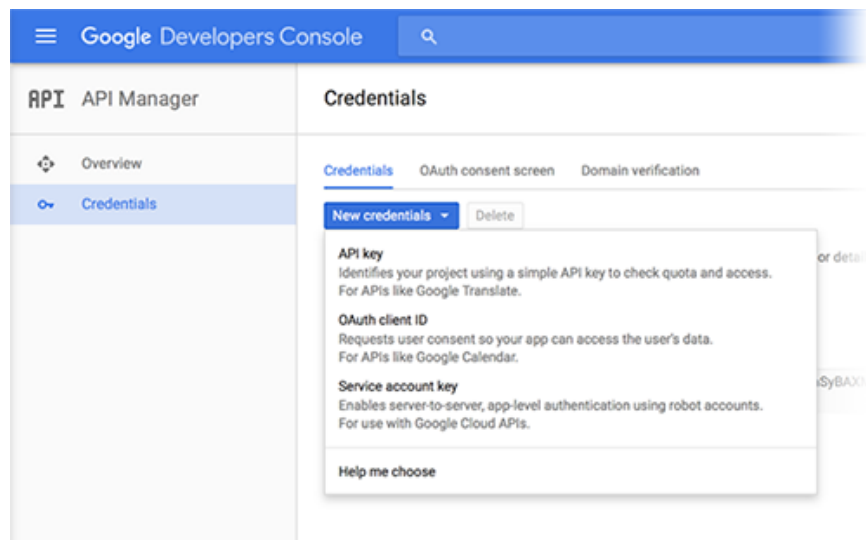
1. Go to the Google Developers Console at <https://console.developers.google.com>.
2. In the **Dashboard** for your project, go to the **Use Google APIs** section and then choose **Enable and manage APIs**.



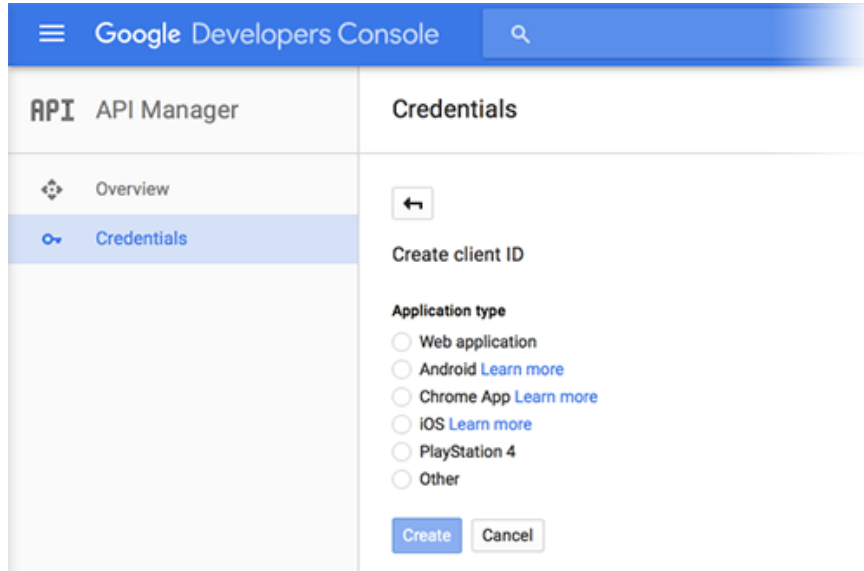
3. In the API Manager, choose **Credentials** in the left side menu.



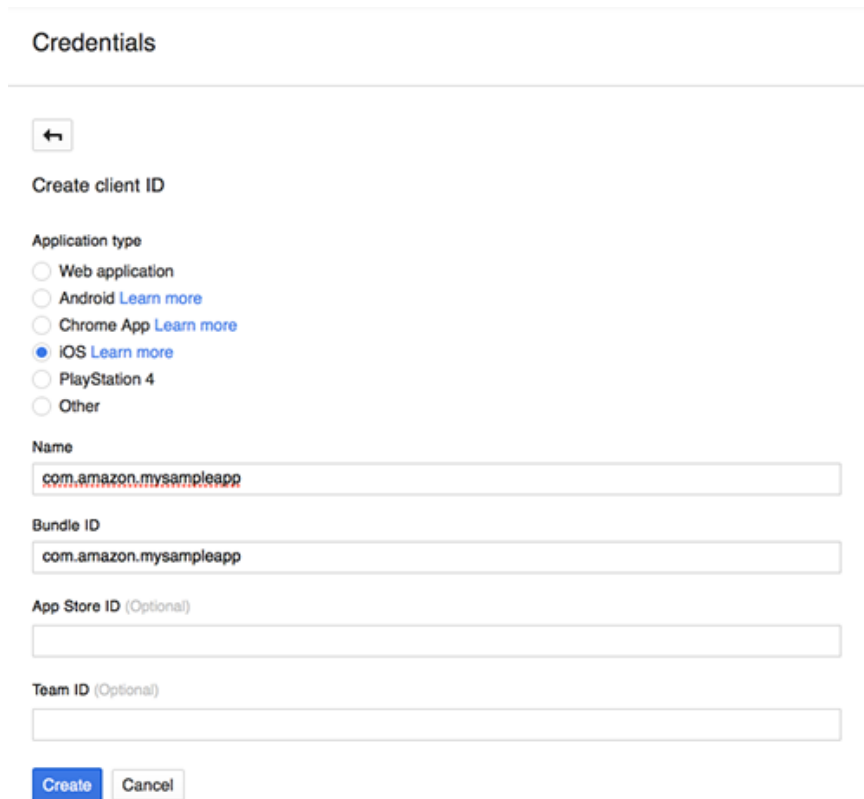
4. Choose **New Credentials** and then choose **OAuth client ID**.



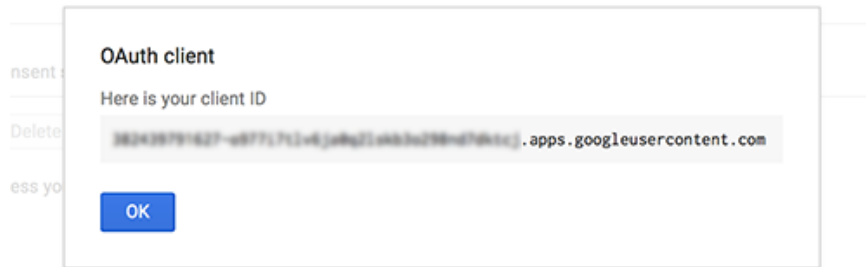
5. In **Create client ID**, choose **iOS**.



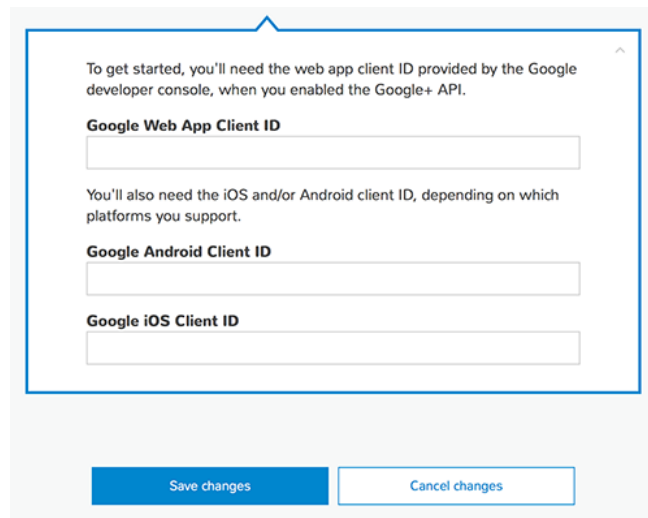
6. In **Name**, enter a name in the format *com.amazon.mysampleapp iOS client ID*.
7. In **Bundle ID**, enter the bundle name in the format *com.amazon.mysampleapp*.



8. Choose **Create**.
9. In the **OAuth client** pop-up, copy and save the value that was generated for your iOS client ID. You will need these values to implement Google Sign-In in your Mobile Hub app. After you copy the client ID, choose **OK**.

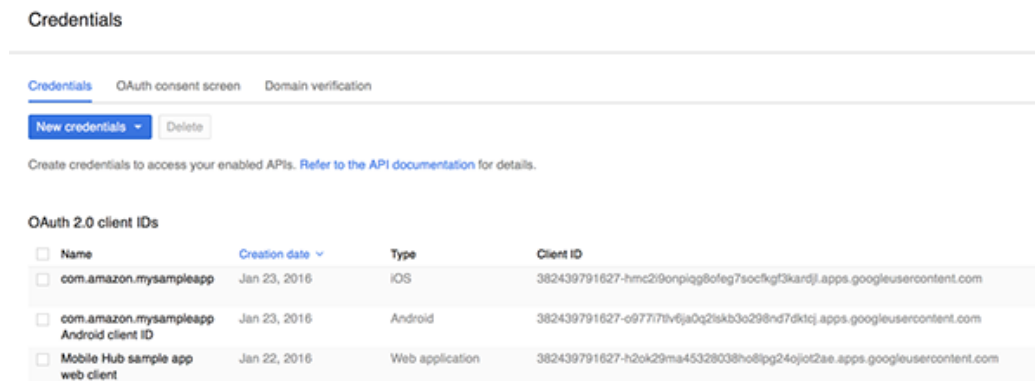


10. Paste the iOS client ID value into the Mobile Hub **Google iOS Client ID** field for your project.



Verifying All Platform Client IDs

If your app supports both Android and iOS platforms, then your app project in the Google Developers Console will now have three client IDs: one for web application, one for Android, and one for iOS. You can verify that you have all of the credentials for all of the platforms by looking at the **Credentials** panel in the API Manager for your app, as shown in the following.



Setting Up Custom Authentication

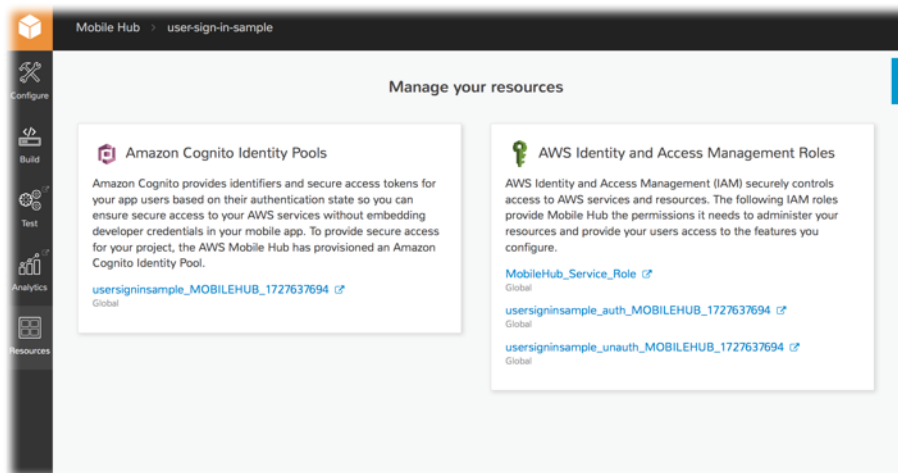
You can use your own authentication system, rather than identity federation provided by Facebook or Google, to register and authenticate your customers. The use of developer-authenticated identities involves interaction between the end-user device, your authentication back end, and Amazon Cognito. For more information, see the following blog entries:

- [Understanding Amazon Cognito Authentication](#)
- [Understanding Amazon Cognito Authentication Part 2: Developer-Authenticated Identities](#)

To use your own authentication system, you must implement an identity provider by extending the `AWSAbstractCognitoIdentityProvider` class and associating your provider with an Amazon Cognito identity pool. For more information, see [Developer Authenticated Identities](#) in the Amazon Cognito Developer Guide.

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub **Resources** pane displaying elements typically provisioned for the User Sign-in feature.



Quickstart App Details

In the Mobile Hub quickstart app, the User Sign-in demo enables users to use features that access AWS resources without authentication or by signing in to the app via identity providers including Facebook, Google, or your custom solution.

When you add User Sign-in to your project with the **No Sign-in** or the **Optional Sign-in** options, choosing the app's quickstart sign-in demo returns and displays the Amazon Cognito identifier. This identifier is associated with the app instance's device currently accessing AWS resources.

When you add User Sign-in to your project with **Required Sign-in**, choosing the app's quickstart sign-in demo displays a sign-in experience branded to match the identity provider. Signing in to the demo

authenticates the user in the selected identity provider service and returns and displays the Amazon Cognito identifier that is associated with authentication of the user in that service.

Document History for AWS Mobile Hub

The following table describes important changes to the documentation since the release of AWS Mobile Hub.

- **Latest documentation update:** August 19, 2016

Change	Description	Date Changed
AWS Mobile Hub Developer Guide Redesign	Site restructured around using the key mobile app backend features Mobile Hub facilitates. Most pages in the site are updated with additional information.	August 17, 2016
iOS and Android Push Notification Setup Documentation	The documentation for setting up iOS push notifications in the Apple Developer Member Center website and Android push notifications in the Google Developers Console website has been updated to provide more detail about the process for setting up these features outside the Mobile Hub console.	February 9, 2016

Change	Description	Date Changed
Facebook and Google Authentication Process Documentation	The documentation now has a section describing how to create a Google Developers Console project and create all client IDs Mobile Hub needs to enable Google Sign-In in both iOS and Android apps. For more information, see Setting Up Google Authentication (p. 41) . The documentation on creating a Facebook app ID has been updated to reflect changes in the Facebook Developer portal. For more information, see Setting Up Facebook Authentication (p. 39) .	January 26, 2016
IAM Managed Policies Added	Details about the managed policies required to view and modify configuration for any project with AWS Mobile Hub. For more information, see IAM (Identity and Access Management) Managed Policies (p. 59) .	January 4, 2016
IAM Service Role for Mobile Hub Added	Details about the service policy and permissions for the <code>MobileHub_Service_Role</code> IAM role created by Mobile Hub to configure the features of each mobile app is documented. For more information, see The Mobile Hub IAM Service Role (p. 60) .	November 9, 2015
New Guide	This is the first release of the AWS Mobile Hub service. This is a beta release and is subject to change.	October 8, 2015

AWS Mobile Hub Reference

The reference topics in this section provide more detailed information about how Mobile Hub works.

Topics

- [IAM \(Identity and Access Management\) Managed Policies \(p. 59\)](#)

IAM (Identity and Access Management) Managed Policies

Topics

- [Learn about AWS IAM Roles and Policies \(p. 59\)](#)
- [IAM \(Identity and Access Management\) Managed Policies \(p. 59\)](#)
- [The Mobile Hub IAM Service Role \(p. 60\)](#)
- [Trust Relationship \(p. 61\)](#)
- [Administrative Privileges \(p. 61\)](#)
- [Service Policy \(p. 61\)](#)

Learn about AWS IAM Roles and Policies

Options for learning more about IAM include:

- Visit the “Resources” page for your Mobile Hub Project to discover and access resources you have configured in the AWS Console
- Visit the Amazon Cognito Developer Guide [IAM Roles](#) page
- Visit the [AWS IAM User Guide](#)

IAM (Identity and Access Management) Managed Policies

The Identity and Access Management service controls user permissions with respect to AWS services and resources. Specific permissions are required in order to view and modify configuration for any project

with AWS Mobile Hub. These permissions have been grouped into the following managed policies, which you can attach to an IAM User, Role, or Group.

AWSMobileHub_FullAccess

This policy provides read and write access to AWS Mobile Hub projects. Users with this policy attached to their IAM User, Role, or Group, are allowed to create new projects, modify configuration for existing projects, and delete projects and resources. This policy also includes all the permissions which are allowed under the `AWSMobileHub_ReadOnly` managed policy.

https://console.aws.amazon.com/iam/home?region=us-east-1#policies/arn:aws:iam::aws:policy/AWSMobileHub_FullAccess

AWSMobileHub_ReadOnly

This policy provides read-only access to AWS Mobile Hub projects. Users with this policy attached to their IAM User, Role, or Group, are allowed to view project configuration and generate sample quick start app projects that can be downloaded and built on a developer's desktop (e.g., in Android Studio or Xcode). It does not allow any modification to Mobile Hub project configuration, and it does not allow the user to enable the use of AWS Mobile Hub in an account, where it has not already been enabled.

https://console.aws.amazon.com/iam/home?region=us-east-1#policies/arn:aws:iam::aws:policy/AWSMobileHub_ReadOnly

Attaching a Managed Policy to a User, Role, or Group

To use these managed policies, a user with administrative privileges must attach one of them to a User, Role or Group in the Identity and Access Management console.

To attach a managed policy

1. Choose the link for the managed policy you want to attach.
 - https://console.aws.amazon.com/iam/home?region=us-east-1#policies/arn:aws:iam::aws:policy/AWSMobileHub_FullAccess
 - https://console.aws.amazon.com/iam/home?region=us-east-1#policies/arn:aws:iam::aws:policy/AWSMobileHub_ReadOnly
2. Choose **Attached Entities**.
3. Choose **Attach**.
4. Choose the Users, Roles, or Groups you want to grant permissions.
5. Choose **Attach Policy**.

The Mobile Hub IAM Service Role

AWS Mobile Hub provides an integrated console experience in which you select features to use in a mobile app. When you select and enable a feature, Mobile Hub configures multiple AWS services and resources on your behalf. Configuring any AWS service or resource requires your permission. You give Mobile Hub these permissions by allowing it to create an IAM (AWS Identity and Access Management) administrative service role, called `MobileHub_Service_Role`.

After this service role is created, you can examine it at https://console.aws.amazon.com/iam/home?region=us-east-1#roles/MobileHub_Service_Role.

Topics

Trust Relationship

In the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/home?region=us-east-1#roles/MobileHub_Service_Role, there is a section for the trust relationship. The trust relationship dictates which entities can assume this role and make use of its permissions. The trust relationship for this service role has an access control policy that looks like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "mobilehub.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

This access control policy dictates that only AWS Mobile Hub (`mobilehub.amazonaws.com`) can assume this role. This policy should not be modified. No other user or system can assume this role and use its permissions.

Administrative Privileges

By allowing Mobile Hub to create and assume the `MobileHub_Service_Role` role, you give Mobile Hub permissions to create additional roles as necessary to support the features enabled in your project. The `MobileHub_Service_Role` gives Mobile Hub permission to enable any service policies necessary on these additional created roles for proper operation of the mobile app.

There are no limits on the number or scope of service policies or roles Mobile Hub may create. Actions taken by Mobile Hub in this regard are always in response to your actions in the Mobile Hub console. Roles or policies are never created without direct action from you, such as creating a Mobile Hub project or configuring an app feature.

Revoking Privileges

To disallow Mobile Hub access to any users of your account, delete the `MobileHub_Service_Role` role. Make sure your users don't have permission to re-create the role, for example by having the `IAM:CreateRole` permission.

Service Policy

The service policy states which operations an entity that assumes the `MobileHub_Service_Role` role can perform. If the role has been created, go to https://console.aws.amazon.com/iam/home?region=us-east-1#roles/MobileHub_Service_Role to see the service policy used by AWS Mobile Hub. It looks like the following example:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudfront:CreateDistribution",
        "cloudfront>DeleteDistribution",
        "cloudfront:GetDistribution",
        "cloudfront:GetDistributionConfig",
        "cloudfront:UpdateDistribution",
        "cognito-identity:CreateIdentityPool",
        "cognito-identity:UpdateIdentityPool",
        "cognito-identity>DeleteIdentityPool",
        "cognito-identity:SetIdentityPoolRoles",
        "iam:AddClientIDToOpenIDConnectProvider",
        "iam:CreateOpenIDConnectProvider",
        "iam:GetOpenIDConnectProvider",
        "iam:ListOpenIDConnectProviders",
        "lambda:CreateFunction",
        "lambda>DeleteFunction",
        "lambda:GetFunction",
        "mobileanalytics:CreateApp",
        "mobileanalytics>DeleteApp",
        "sns:CreateTopic",
        "sns>DeleteTopic"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:CreatePlatformApplication",
        "sns>DeletePlatformApplication",
        "sns:GetPlatformApplicationAttributes",
        "sns:SetPlatformApplicationAttributes"
      ],
      "Resource": [
        "arn:aws:sns:*:*:app/*_MOBILEHUB_*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3>DeleteBucket",
        "s3>DeleteBucketPolicy",
        "s3:ListBucket",
        "s3:ListBucketVersions"
      ],
      "Resource": [
        "arn:aws:s3::*-userfiles-mobilehub-*",
        "arn:aws:s3::*-contentdelivery-mobilehub-*"
      ]
    }
  ]
}

```

```
    "Effect": "Allow",
    "Action": [
      "s3:DeleteObject",
      "s3:DeleteVersion",
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": [
      "arn:aws:s3:::*-userfiles-mobilehub-*/*",
      "arn:aws:s3:::*-contentdelivery-mobilehub-*/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:CreateRole",
      "iam>DeleteRole",
      "iam>DeleteRolePolicy",
      "iam:GetRole",
      "iam:GetRolePolicy",
      "iam>ListRolePolicies",
      "iam:PassRole",
      "iam:PutRolePolicy",
      "iam:UpdateAssumeRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::*:role/*_unauth_MOBILEHUB_*",
      "arn:aws:iam::*:role/*_auth_MOBILEHUB_*",
      "arn:aws:iam::*:role/*_consolepush_MOBILEHUB_*",
      "arn:aws:iam::*:role/*_lambdaexecutionrole_MOBILEHUB_*",
      "arn:aws:iam::*:role/MobileHub_Service_Role"
    ]
  }
]
}
```

All of these permissions pertain to resources Mobile Hub creates on your behalf. You can see these resources by choosing **Resources** in the left navigation panel of the Mobile Hub console.

AWS Identity and Access Management

This is the portion of the service policy for the Mobile Hub service role defining IAM permissions.

```
"iam:CreateRole",
"iam>DeleteRole",
"iam>DeleteRolePolicy",
"iam:GetRole",
"iam>ListRolePolicies",
"iam:PassRole",
"iam:PutRolePolicy",
"iam:UpdateAssumeRolePolicy",
```

Mobile Hub creates one or more IAM roles to use with your mobile app project, depending on the configuration options you choose for each feature. By default, IAM creates an unauthenticated app user role to allow users of your app to get temporary permissions to perform various operations with other

services you've enabled. For example, you need this role when your app calls an AWS Lambda function in the Cloud Logic feature.

If you enable the Cloud Logic feature, Mobile Hub also creates an AWS Lambda execution role. This role provides your AWS Lambda functions the permissions they need to carry out their tasks; for example, writing debug logs to Amazon CloudWatch.

If you enable the User Sign-in feature, Mobile Hub creates an authenticated app user role. This authenticated app user role is used by the users of your app when they sign in using a sign-in provider such as Facebook or Google+. If you select the **Sign-in is required** option in User Sign-in, the unauthenticated app user role is removed. All access to your resources from the app then require use of the authenticated role.

In addition, if you select Google as a sign-in provider, Mobile Hub needs access to the following Open ID Connect Provider APIs from IAM:

```
"iam:AddClientIDToOpenIDConnectProvider",  
"iam:CreateOpenIDConnectProvider",  
"iam:GetOpenIDConnectProvider",  
"iam:ListOpenIDConnectProviders",
```

These permissions let the service create an Open ID Connect Provider for Google if it does not already exist, and add ClientIDs to that provider.

Amazon Cognito

This is the portion of the service policy for the Mobile Hub service role defining Amazon Cognito permissions.

```
"cognito-identity:CreateIdentityPool",  
"cognito-identity:UpdateIdentityPool",  
"cognito-identity>DeleteIdentityPool",  
"cognito-identity:SetIdentityPoolRoles",
```

Amazon Cognito provides temporary credentials that give app users access to your AWS resources. By default Mobile Hub creates an Amazon Cognito identity pool to provide a scope or namespace for user identities. If you enable the User Sign-in feature and configure a sign-in provider, such as Facebook or Google+, Mobile Hub updates the identity pool to support that feature in your app.

Amazon Mobile Analytics

This is the portion of the service policy for the Mobile Hub service role defining Mobile Analytics permissions.

```
"mobileanalytics:CreateApp",  
"mobileanalytics>DeleteApp",
```

When you enable the App Analytics feature in Mobile Hub, it creates an App ID for your app in Amazon Mobile Analytics. This App ID can be removed if you delete the project.

Amazon Simple Notification Service

This is the portion of the service policy for the Mobile Hub service role defining Amazon SNS permissions.

```
"sns:CreateTopic",  
"sns>DeleteTopic",  
"sns:CreatePlatformApplication",  
"sns>DeletePlatformApplication",  
"sns:GetPlatformApplicationAttributes",  
"sns:SetPlatformApplicationAttributes",
```

When you enable the Push Notifications feature, Mobile Hub creates an Amazon SNS platform application for each push platform you configure. It also creates a default Amazon SNS topic you can use to push messages to all users of your app. The topic and platform application are deleted if you delete the associated Mobile Hub project.

Amazon Simple Storage Service

This is the portion of the service policy for the Mobile Hub service role defining Amazon S3 permissions.

```
"s3:CreateBucket",  
"s3>DeleteBucket",  
"s3>DeleteBucketPolicy",  
"s3:ListBucket",  
"s3:ListBucketVersions",  
"s3>DeleteObject",  
"s3>DeleteVersion",  
"s3:PutObject",  
"s3:PutObjectAcl",
```

App Content Delivery and User Data Storage features both use Amazon Simple Storage Service. When you enable one of these features, Mobile Hub creates an S3 bucket on your behalf. Mobile Hub also puts example files and folders in the bucket so you can demonstrate your app downloading and navigating between folders. Some of these permissions are required to set up your S3 bucket for use with Amazon CloudFront if you enable the App Content Delivery feature and select Multi-Region CDN.

Amazon CloudFront

This is the portion of the service policy for the Mobile Hub service role defining CloudFront permissions.

```
"cloudfront:CreateDistribution",  
"cloudfront>DeleteDistribution",  
"cloudfront:GetDistribution",  
"cloudfront:GetDistributionConfig",  
"cloudfront:UpdateDistribution",
```

If you enable the App Content Delivery feature and configure it for Multi-Region CDN, Mobile Hub creates a CloudFront distribution with your Amazon S3 bucket set as the origin.

AWS Lambda

This is the portion of the service policy for the Mobile Hub service role defining Lambda permissions.

```
"lambda:CreateFunction",  
"lambda>DeleteFunction",  
"lambda:GetFunction",
```

If you enable the Cloud Logic feature, Mobile Hub creates an example Lambda function. You can use this function to demonstrate invocation of a Lambda function from your app.