1. [Get a free AWS Account.](https://aws.amazon.com/free)
2. [Create a Mobile Hub project](https://console.aws.amazon.com/mobilehub/) to enable backend features for your app, if you don't already have one.

Already have an app using a Mobile Hub custom SDK? [Upgrade an Existing Mobile Hub-based App](http://docs.aws.amazon.com/aws-mobile/latest/developerguide/aws-mobile-sdk-migrate.html).


                           Image of the |AMH| console.
                        

*Analytics are enabled by default for new projects created in Mobile Hub.*

1. Download your Mobile Hub project configuration file.
   1. In the Mobile Hub console, choose your project, and then choose the **Integrate** icon from the left margin.
   2. 
                                       Image of the Mobile Hub console when choosing Download Configuration File.
                                    Choose **Download Configuration File** to get the awsconfiguration.json file that connects your app to your backend.
2. *Remember:*
3. Each time you change the Mobile Hub project for your app, download and use a freshawsconfiguration.json to reflect those changes in your app. If NoSQL Database or Cloud Logic are changed, also download and use fresh files for those features.

**Set Up Your App for AWS Mobile Services**

If you have not created a Mobile Hub project and downloaded its configuration file, see [Basic Backend Setup](http://docs.aws.amazon.com/aws-mobile/latest/developerguide/add-aws-mobile-sdk.html#add-aws-mobile-sdk-basic-setup).

Choose your platform.

**Android - Java**iOS - SwiftJavaScript

1. [Install Android Studio](https://developer.android.com/studio/index.html) version 2.33 or higher .
2. Install Android SDK version 7.11 (Nougat), API level 25

In Android Studio, from the top menu bar choose **Tools > Android > SDK Manager** to install an SDK version.

1. Add the backend service configuration file to your app.`
   1. Open your mobile app project in Android Studio and choose **Project** in the left margin to open project view.
   2. 
                                       Image of the Download Configuration Files button in the |AMH| console.
                                    Right-click your app's res folder, and then choose **New > Android Resource Directory**. Select **raw** in the **Resource type** dropdown menu.

Learn more about [Android Studio](https://developer.android.com/studio/intro/index.html).

* 1. From the location where configuration files were downloaded in a previous step, dragawsconfiguration.json into the res/raw folder.

1. Add dependencies to the your app/build.gradle.

Add the following [Android gradle dependencies](https://docs.gradle.org/current/userguide/artifact_dependencies_tutorial.html) entries and configuration to your app/build.gradle. These libraries enable basic AWS functions, like credentials, and analytics. Adding *:code:`multidex application*<<https://developer.android.com/studio/build/multidex.html>>`\_ configuration ensures that you won't run into method count limitations in your app.

**Copy**

android {

defaultConfig {

...

multiDexEnabled true

}

...

}

dependencies {

compile 'com.android.support:multidex:1.0.1'

compile 'com.amazonaws:aws-android-sdk-core:2.6.0'

compile ('com.amazonaws:aws-android-sdk-auth-core:2.6.0@aar') {transitive = true;}

}

1. Create an Application class and add the following code to its onCreate method.

To create the class, right click on the java folder in your Xcode project explorer, and then choose **New > Java Class**. Name the class Application and choose public for **Visibility** and none for **Modifiers**.

**Copy**

**import** com.amazonaws.mobile.config.AWSConfiguration;

**import** com.amazonaws.mobile.auth.core.IdentityManager;

**import** android.support.multidex.MultiDexApplication;

/\*\*

\* Application class responsible for initializing singletons and other common components.

\*/

**public** **class** **Application** **extends** **MultiDexApplication** {

**private** **static** **final** String LOG\_TAG = Application.class.getSimpleName();

@Override

**public** **void** **onCreate**() {

**super**.onCreate();

initializeApplication();

}

**private** **void** **initializeApplication**() {

AWSConfiguration awsConfiguration = **new** AWSConfiguration(getApplicationContext());

// If IdentityManager is not created, create it

**if** (IdentityManager.getDefaultIdentityManager() == **null**) {

IdentityManager identityManager =

**new** IdentityManager(getApplicationContext(), awsConfiguration);

IdentityManager.setDefaultIdentityManager(identityManager);

}

}

}

1. Create a SplashActivity class or modify your existing splash activity.
   1. To create the activity, right click on the java folder in your Xcode project explorer, and then choose **File > New > Activity > Basic Activity**.


                                 Image of the Download Configuration Files button in the |AMH| console.
                              

* 1. Add the following code to the activity's onCreate method to establish user credentials that enable access to AWS services whenever your app starts.

**Copy**

**import** com.amazonaws.mobile.config.AWSConfiguration;

**import** com.amazonaws.mobile.auth.core.IdentityManager;

**import** com.amazonaws.mobile.auth.core.StartupAuthResultHandler;

**import** com.amazonaws.mobile.auth.core.StartupAuthResult;

**public** **class** **SplashActivity** **extends** **AppCompatActivity** {

@Override

**protected** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.activity\_splash);

Context appContext = getApplicationContext();

AWSConfiguration awsConfig = **new** AWSConfiguration(appContext);

IdentityManager identityManager = **new** IdentityManager(appContext, awsConfig);

IdentityManager.setDefaultIdentityManager(identityManager);

identityManager.doStartupAuth(**this**, **new** StartupAuthResultHandler() {

@Override

**public** **void** **onComplete**(StartupAuthResult startupAuthResult) {

// User identity is ready as unauthenticated user or previously signed-in user.

}

});

// Go to the main activity

**final** Intent intent = **new** Intent(**this**, :samp:`{MainActivity}`.class)

.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

**this**.startActivity(intent);

**this**.finish();

}

}

1. Modify your app manifest to add [Android permissions](https://developer.android.com/guide/topics/permissions/requesting.html)

Delete the intent-filter declarations for the MAIN action and and LAUNCHER category from your original start up activity. If there are no other declarations in the intent-filter, then also delete the empty <intent-filter></intent-filter> tags.

**Copy**

<**uses-permission** android:name="android.permission.INTERNET" />

<**uses-permission** android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<**uses-permission** android:name="android.permission.ACCESS\_WIFI\_STATE" />

. . .

<**application**

android:name="com.:samp:`{yourpackagename}`.Application">

. . .

<**activity** android:name=".SplashActivity" >

<**intent-filter**>

<**action** android:name="android.intent.action.MAIN" />

<**category** android:name="android.intent.category.LAUNCHER" />

</**intent-filter**>

</**activity**>

<**activity** android:name=".MainActivity" >

<**intent-filter**>

<!--

\* REMOVE THESE FROM YOUR START UP ACTIVITY

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

\* IF THERE ARE NO OTHER ITEMS INSIDE THE intent-filter

\* TAGS, DELETE THE TAGS

-->

</**intent-filter**>

</**activity**>

. . .

</**application**>

*Make sure to remove the MAIN action and LAUNCHER category from your previous starting activity's :code:`intent-filter*.`

1. Click the **Run** icon (the one that looks like a Play button) in Android Studio to build your app and run it on your device/emulator. After your app is deployed, search through your logcat for a message similar to "IdentityManager: Got user ID: us-east-1:abcabcabc-0be6-444e-b101-abcabcabc". If you see the log, your app is successfully connected to AWS services.

Your app is now set up to interact with the AWS services you configured in your Mobile Hub project!