AWS Mobile Hub

Developer Guide

AWS Mobile Hub: Developer Guide

Table of Contents

On October 30, 2021, AWS Amplify will replace AWS Mobile Hub. For more information, see Migrating to Amplify (p. 1).

Mobile and Web App Development

Migrating to Amplify

Important
On October 30, 2021, AWS Amplify will replace AWS Mobile Hub. Amplify provides the fastest way to develop, deliver, and manage cloud-connected web and mobile apps. All Mobile Hub features are available with Amplify. For more information about Amplify, see the AWS Amplify Documentation.

Your Mobile Hub project is a container for your app's cloud resources. If you don't migrate your project to Amplify, your app will continue to function, and all your related cloud resources will continue to be available. However, you won't be able to access the Mobile Hub project container after October 30, 2021.

To migrate your Mobile Hub projects to Amplify

1. Sign in to the Mobile Hub console.
2. In the notice banner at the top of the page, choose **Migrate**.

   The console opens to the **Migrate Mobile Hub Project** page, where you'll find a list of your Mobile Hub projects in the current AWS Region.
3. Choose the **Amplify Backend Role** dropdown list, and then choose the AWS Identity and Access Management (IAM) service-linked role that you want to use to run the migration. Service-linked roles perform AWS actions on your behalf. For more information, see Using service-linked roles in the *IAM User Guide*.
   - If you don't already have a service-linked role for migrating Mobile Hub projects to Amplify, choose **Create new role** to open the IAM role creation wizard. On each step of the wizard, review the pre-selected defaults, and then choose **Create role**.
4. To begin migrating all your Mobile Hub projects in the current Region to Amplify, choose **Migrate Projects**. This process can take up to an hour to complete.
5. When the migration finishes, you'll see your migrated projects on the Amplify console dashboard.
6. Repeat these steps for all Regions where you have Mobile Hub projects.

After migrating, you can access your AWS resources in Amplify and add new capabilities to your app. For example, DataStore, Hosting, and GraphQL APIs.

What is Amplify?

Amplify is a set of products and tools that enables mobile and front-end web developers to build and deploy secure, scalable full stack applications, powered by AWS. It includes a comprehensive set of SDKs, libraries, tools, and documentation for client app development. For more information, see the Amplify Framework Documentation.

AWS Mobile Reference

Topics

Android and iOS API References

Android

- AWS Mobile SDK for Android API Reference
- AWS Mobile SDK for Android on GitHub
- AWS Mobile SDK for Android Samples

iOS

- AWS Mobile SDK for iOS API Reference
- AWS Mobile SDK for iOS on GitHub
- AWS Mobile SDK for iOS Samples

AWS Mobile Hub Reference

Topics

AWS Mobile for Android and iOS

The AWS Mobile SDKs for Android and iOS are now part of the Amplify Framework documentation. For more information, see the following links:

- Amplify Android - Recommended for developing new cloud-connected Android apps.
- Amplify iOS - Recommended for developing new cloud-connected iOS apps.
- AWS Mobile SDK for Android - Low-level AWS service-specific APIs for cloud-connected Android apps.
- AWS Mobile SDK for iOS - Low-level AWS service-specific APIs for cloud-connected iOS apps.

AWS Amplify Library for Web

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

AWS Amplify is an open source JavaScript library for frontend and mobile developers building cloud-enabled applications. The library is a declarative interface across different categories of operations in order to make common tasks easier to add into your application. The default implementation works with Amazon Web Services (AWS) resources but is designed to be open and pluggable for usage with other cloud services that wish to provide an implementation or custom backends.

The AWS Mobile CLI, built on AWS Mobile Hub, provides a command line interface for frontend JavaScript developers to seamlessly enable and configure AWS services into their apps. With minimal configuration, you can start using all of the functionality provided by the AWS Mobile Hub from your favorite terminal application.

Topics

Get Started

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Overview

The AWS Mobile CLI provides a command line experience that allows front end JavaScript developers to quickly create and integrate AWS backend resources into their mobile apps.

Prerequisites

1. Sign up for the AWS Free Tier.
2. Install Node.js with NPM.
3. Install AWS Mobile CLI

```
npm install -g awsmobile-cli
```

4. Configure the CLI with your AWS credentials

   To setup permissions for the toolchain used by the CLI, run:

```
awsmobile configure
```

   If prompted for credentials, follow the steps provided by the CLI. For more information, see provide IAM credentials to AWS Mobile CLI (p. 33).

Set Up Your Backend

Need to create a quick sample React app? See Create a React App.

To configure backend features for your app

1. In the root folder of your app, run:

   ```
   awsmobile init
   ```

   The `init` command creates a backend project for your app. By default, analytics and web hosting are enabled in your backend and this configuration is automatically pulled into your app when you initialize.
2. When prompted, provide the source directory for your project. The CLI will generate `aws-exports.js` in this location. This file contains the configuration and endpoint metadata used to link your front end to your backend services.

   ```
   ? Where is your project's source directory:  src
   ```
3. Respond to further prompts with the following values.

   ```
   ? Where is your project's distribution directory to store build artifacts:  build
   ? What is your project's build command:  npm run-script build
   ? What is your project's start command for local test run:  npm run-script start
   ? What awsmobile project name would you like to use:  YOUR-APP-NAME-2017-11-10-15-17-48
   ```

After the project is created you will get a success message which also includes details on the path where the aws-exports.js is copied.

```
awsmobile project's details logged at: awsmobilejs/#current-backend-info/backend-
details.json
awsmobile project's access information logged at: awsmobilejs/#current-backend-info/aws-
exports.js
awsmobile project's access information copied to: src/aws-exports.js
awsmobile project's specifications logged at: awsmobilejs/#current-backend-info/mobile-hub-
project.yml
contents in #current-backend-info/ is synchronized with the latest information in the aws
 cloud
```

Your project is now initialized.

Note
You can add the AWS backend resources you create for this project to another exisiting app using `awsmobile init YOUR_MOBILE_HUB_PROJECT_ID`. To find the project ID, open your Mobile Hub project in the Mobile Hub console by running `awsmobile console`. The project ID is the GUID portion of the console address, in the form of `XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`.

Connect to Your Backend

AWS Mobile uses the open source AWS Amplify library to link your code to the AWS features configured for your app.

This section of the guide shows examples using a React application of the kind output by `create-react-app` or a similar tool.

To connect the app to your configured AWS features

In `index.js` (or in other code that runs at launch-time), add the following imports.

```
import Amplify from 'aws-amplify';
import awsmobile from './YOUR-PATH-TO/aws-exports';
```

Then add the following code.

```
Amplify.configure(awsmobile);
```

Run Your App Locally

Your app is now ready to launch and use the default features configured by AWS Mobile.

To launch your app locally in a browser

In the root folder of your app, run:

```
awsmobile run
```

Behind the scenes, this command runs `npm install` to install the Amplify library and also pushes any backend configuration changes to AWS Mobile. To run your app locally without pushing backend changes you can choose to run `npm install` and then run `npm start`.

Anytime you launch your app, app analytics are gathered and can be visualized (p. 8) in an AWS console.

AWS Free Tier	Initializing your app or adding features through the CLI will cause AWS services to be configured on your behalf. The pricing for AWS Mobile services enables you to learn and prototype at little or no cost using the AWS Free Tier.

Next Steps

Topics

Deploy your app to the cloud

Using a simple command, you can publish your app's front end to hosting on a robust content distribution network (CDN) and view it in a browser.

To deploy your app to the cloud and launch it in a browser

In the root folder of your app, run:

```
awsmobile publish
```

To push any backend configuration changes to AWS and view content locally, run `awsmobile run`. In both cases, any pending changes you made to your backend configuration are made to your backend resources.

By default, the CLI configures AWS Mobile Hosting and Streaming (p. 61) feature, that hosts your app on Amazon CloudFront CDN endpoints. These locations make your app highly available to the public on the Internet and support media file streaming

You can also use a custom domain (p. 22) for your hosting location.

Test Your App on Our Mobile Devices

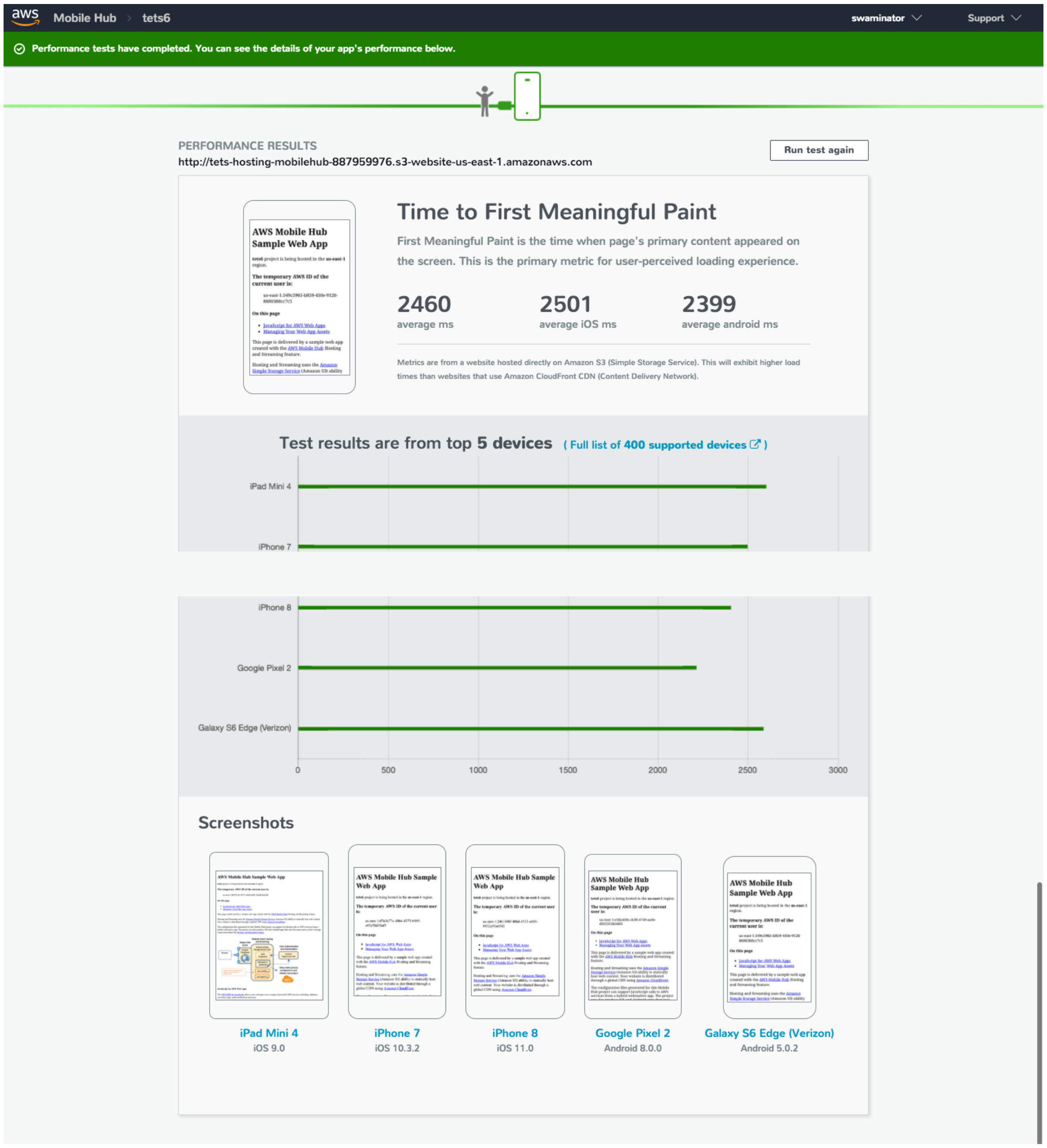
Invoke a free remote test of your app on a variety of real devices and see results, including screen shots.

To invoke a remote test of your app

In the root folder of your app, run:

```
awsmobile publish --test
```

The CLI will open the reporting page for your app in the Mobile Hub console to show the metrics gathered from the test devices. The device that runs the remote test you invoke resides in AWS Device Farm which provides flexible configuration of tests and reporting.



Performance tests have completed. You can see the details of your app's performance below.

PERFORMANCE RESULTS
http://tets-hosting-mobilehub-887959976.s3-website-us-east-1.amazonaws.com

Time to First Meaningful Paint

First Meaningful Paint is the time when page's primary content appeared on the screen. This is the primary metric for user-perceived loading experience.

2460	2501	2399
average ms	average iOS ms	average android ms

Metrics are from a website hosted directly on Amazon S3 (Simple Storage Service). This will exhibit higher load times than websites that use Amazon CloudFront CDN (Content Delivery Network).

Test results are from top 5 devices (Full list of 400 supported devices)

Screenshots

iPad Mini 4	iPhone 7	iPhone 8	Google Pixel 2	Galaxy S6 Edge (Verizon)
iOS 9.0	iOS 10.3.2	iOS 11.0	Android 8.0.0	Android 5.0.2

Add Features

Add the following AWS Mobile features to your mobile app using the CLI.

- Analytics (p. 8)
- User Sign-in (p. 9)
- NoSQL Database (p. 10)
- User File Storage (p. 15)
- Cloud Logic (p. 18)

Learn more

To learn more about the commands and usage of the AWS Mobile CLI, see the AWS Mobile CLI reference (p. 24).

Learn about AWS Mobile Amplify.

Add Analytics

> **Important**
> The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Basic Analytics Backend is Enabled for Your App

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 3).

When you complete the AWS Mobile CLI setup and launch your app, anonymized session and device demographics data flows to the AWS analytics backend.

To send basic app usage analytics to AWS

Launch your app locally by running:

```
npm start
```

When you use your app the Amazon Pinpoint service gathers and visualizes analytics data.

To view the analytics using the Amazon Pinpoint console

1. Run `npm start`, `awsmobile run`, or `awsmobile publish --test` at least once.
2. Open your project in the AWS Mobile Hub console.

   ```
   awsmobile console
   ```

3. Choose the **Analytics** icon on the left, to navigate to your project in the Amazon Pinpoint console.
4. Choose **Analytics** on the left.

You should see an up-tick in several graphs.

Add Custom Analytics to Your App

You can configure your app so that Amazon Pinpoint gathers data for custom events that you register within the flow of your code.

To instrument custom analytics in your app

In the file containing the event you want to track, add the following import:

```
import { Analytics } from 'aws-amplify';
```

Add the a call like the following to the spot in your JavaScript where the tracked event should be fired:

```
componentDidMount() {
   Analytics.record('FIRST-EVENT-NAME');
}
```

Or to relevant page elements:

```
handleClick = () => {
     Analytics.record('SECOND-EVENT-NAME');
}

<button onClick={this.handleClick}>Call request</button>
```

To test:

1. Save the changes and run `npm start`, `awsmobile run`, or `awsmobile publish --test` to launch your app. Use your app so that tracked events are triggered.
2. In the Amazon Pinpoint console, choose **Events** near the top.
3. Select an event in the **Event** dropdown menu on the left.

Custom event data may take a few minutes to become visible in the console.

Next Steps

Learn more about the analytics in AWS Mobile which are part of the Messaging and Analytics (p. 59) feature. This feature uses Amazon Pinpoint.

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

Add Auth / User Sign-in

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 3).

The AWS Mobile CLI components for user authentication include a rich, configurable UI for sign-up and sign-in.

To enable the Auth features

In the root folder of your app, run:

```
awsmobile user-signin enable
awsmobile push
```

Connect to Your Backend

The AWS Mobile CLI enables you to integrate ready-made sign-up/sign-in/sign-out UI from the command line.

To add user auth UI to your app

1. Install AWS Amplify for React library.

   ```
   npm install --save aws-amplify-react
   ```
2. Add the following import in `App.js` (or other file that runs upon app startup):

   ```
   import { withAuthenticator } from 'aws-amplify-react';
   ```
3. Then change `export default App;` to the following.

   ```
   export default withAuthenticator(App);
   ```

To test, run `npm start`, `awsmobile run`, or `awsmobile publish --test`.

Next Steps

Learn more about the AWS Mobile User Sign-in (p. 67) feature, which uses Amazon Cognito.

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

Access Your Database

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

The AWS Mobile CLI and Amplify library make it easy to perform create, read, update, and delete ("CRUD") actions against data stored in the cloud through simple API calls in your JavaScript app.

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 3).

To create a database

1. Enable the NoSQL database feature and configure your table.

   In the root folder of your app, run:

```
awsmobile database enable --prompt
```

2. Choose `Open` to make the data in this table viewable by all users of your application.

```
? Should the data of this table be open or restricted by user?
# Open
  Restricted
```

3. For this example type in `todos` as your `Table name`.

```
? Table name: todos
```

Add columns and queries

You are creating a table in a NoSQL database and adding an initial set of columns, each of which has a name and a data type. NoSQL lets you add a column any time you store data that contains a new column. NoSQL tables must have one column defined as the Primary Key, which is a unique identifier for each row.

1. For this example, follow the prompts to add three columns: `team` (string), `todoId` (number), and `text` (string).

```
? What would you like to name this column: team
? Choose the data type: string
```

2. When prompted to `? Add another column`, type `Y` and then choose enter. Repeat the steps to create `todoId` and `text` columns.
3. Select `team` as the primary key.

```
? Select primary key
# team
  todoId
  text
```

4. Choose `(todoId)` as the sort key and then `no` to adding any more indexes, to keep the example simple.

Sort Keys and Indexes	To optimize performance, you can define a column as a Sort Key. Choose a column to be a Sort Key if it will be frequently used in combination with the Primary key to query your table. You can also create Secondary Indexes to make additional columns sort keys.

```
? Select sort key
# todoId
  text
  (No Sort Key)

? Add index (Y/n): n
Table todos saved.
```

The `todos` table is now created.

Use a cloud API to do CRUD operations

To access your NoSQL database, you will create an API that can be called from your app to perform CRUD operations.

Why an API?	Using an API to access your database provides a simple coding interface on the front end and robust flexibility on the backend. Behind the scenes, a call to an Amazon API Gateway API end point in the cloud is handled by a serverless Lambda function.

To create a CRUD API

1. Enable and configure the Cloud Logic feature**

```
awsmobile cloud-api enable --prompt
```

2. Choose `Create CRUD API for an existing Amazon DynamoDB table` API for an existing Amazon DynamoDB table" and then choose enter.

```
? Select from one of the choices below. (Use arrow keys)
  Create a new API
# Create CRUD API for an existing Amazon DynamoDB table
```

3. Select the `todos` table created in the previous steps, and choose enter.

```
? Select Amazon DynamoDB table to connect to a CRUD API
# todos
```

4. Push your configuration to the cloud. Without this step, the configuration for your database and API is now in place only on your local machine.

```
awsmobile push
```

The required DynamoDB tables, API Gateway endpoints, and Lambda functions will now be created.

Create your first Todo

The AWS Mobile CLI enables you to test your API from the command line.

Run the following command to create your first todo.

```
awsmobile cloud-api invoke todosCRUD POST /todos '{"body": {"team": "React", "todoId": 1,
 "text": "Learn more Amplify"}}'
```

Connect to Your Backend

The examples in this section show how you would integrate AWS Amplify library calls using React (see the AWS Amplify documentation to use other flavors of Javascript).

The following component is a simple Todo list that you might add to a `create-react-app` project. The Todos component currently adds and displays `todos` to and from an in memory array.

```
// To Do app example

import React from 'react';

class Todos extends React.Component {
  state = { team: "React", todos: [] };

  render() {
    let todoItems = this.state.todos.map(({todoId, text}) => {
      return <li key={todoId}>{text}</li>;
    });

    return (
      <div style={styles}>
        <h1>{this.state.team} Todos</h1>
        <ul>
          {todoItems}
        </ul>

        <form>
          <input ref="newTodo" type="text" placeholder="What do you want to do?" />
          <input type="submit" value="Save" />
        </form>
      </div>
    );
  }
}

let styles = {
  margin: "0 auto",
  width: "25%"
};

export default Todos;
```

Displaying todos from the cloud

The `API` module from AWS Amplify allows you connect to DynamoDB through API Gateway endpoints.

To retrieve and display items in a database

1. Import the `API` module from `aws-amplify` at the top of the Todos component file.

```
import { API } from 'aws-amplify';
```

2. Add the following `componentDidMount` to the `Todos` component to fetch all of the `todos`.

```
async componentDidMount() {
  let todos = await API.get('todosCRUD', `/todos/${this.state.team}`);
  this.setState({ todos });
}
```

When the `Todos` component mounts it will fetch all of the `todos` stored in your database and display them.

Saving todos to the cloud

The following fragment shows the `saveTodo` function for the Todo app.

```
async saveTodo(event) {
  event.preventDefault();

  const { team, todos } = this.state;
  const todoId = todos.length + 1;
  const text = this.refs.newTodo.value;

  const newTodo = {team, todoId, text};
  await API.post('todosCRUD', '/todos', { body: newTodo });
  todos.push(newTodo);
  this.refs.newTodo.value = '';
  this.setState({ todos, team });
}
```

Update the `form` element in the component's render function to invoke the `saveTodo` function when the form is submitted.

```
<form onSubmit={this.saveTodo.bind(this)}>
```

Your entire component should look like the following:

```
// To Do app example

import React from 'react';
import { API } from 'aws-amplify';

class Todos extends React.Component {
  state = { team: "React", todos: [] };

  async componentDidMount() {
    const todos = await API.get('todosCRUD', `/todos/${this.state.team}`)
    this.setState({ todos });
  }

  async saveTodo(event) {
    event.preventDefault();

    const { team, todos } = this.state;
    const todoId = todos.length + 1;
    const text = this.refs.newTodo.value;

    const newTodo = {team, todoId, text};
    await API.post('todosCRUD', '/todos', { body: newTodo });
    todos.push(newTodo);
    this.refs.newTodo.value = '';
    this.setState({ todos, team });
  }

  render() {
    let todoItems = this.state.todos.map(({todoId, text}) => {
      return <li key={todoId}>{text}</li>;
    });

    return (
      <div style={styles}>
        <h1>{this.state.team} Todos</h1>
        <ul>
          {todoItems}
        </ul>

        <form onSubmit={this.saveTodo.bind(this)}>
```

```
          <input ref="newTodo" type="text" placeholder="What do you want to do?" />
          <input type="submit" value="Save" />
        </form>
      </div>
    );
  }
}

let styles = {
  margin: "0 auto",
  width: "25%"
}

export default Todos;
```

Next Steps

- Learn how to retrieve specific items and more with the API module in AWS Amplify.
- Learn how to enable more features for your app with the AWS Mobile CLI.

Add Storage

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up the Backend

The AWS Mobile CLI and AWS Amplify library make it easy to store and manage files in the cloud from your JavaScript app.

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 3).

Enable the User File Storage feature by running the following commands in the root folder of your app.

```
awsmobile user-files enable

awsmobile push
```

Connect to the Backend

The examples in this section show how you would integrate AWS Amplify library calls using React (see the AWS Amplify documentation to use other flavors of Javascript).

The following simple component could be added to a `create-react-app` project to present an interface that uploads images and download them for display.

```
// Image upload and download for display example component
// src/ImageViewer.js

import React, { Component } from 'react';

class ImageViewer extends Component {
  render() {
    return (
      <div>
        <p>Pick a file</p>
        <input type="file" />
      </div>
    );
  }
}

export default ImageViewer;
```

Upload a file

The `Storage` module enables you to upload files to the cloud. All uploaded files are publicly viewable by default.

Import the `Storage` module in your component file.

```
// ./src/ImageViewer.js

import { Storage } from 'aws-amplify';
```

Add the following function to use the `put` function on the `Storage` module to upload the file to the cloud, and set your component's state to the name of the file.

```
uploadFile(event) {
  const file = event.target.files[0];
  const name = file.name;

  Storage.put(key, file).then(() => {
    this.setState({ file: name });
  });
}
```

Place a call to the `uploadFile` function in the `input` element of the component's render function, to start upload when a user selects a file.

```
render() {
  return (
    <div>
      <p>Pick a file</p>
      <input type="file" onChange={this.uploadFile.bind(this)} />
    </div>
  );
}
```

Display an image

To display an image, this example shows the use of the `S3Image` component of the AWS Amplify for React library.

1. From a terminal, run the following command in the root folder of your app.

```
npm install --save aws-amplify-react
```

2. Import the `S3Image` module in your component.

```
import { S3Image } from 'aws-amplify-react';
```

Use the S3Image component in the render function. Update your render function to look like the following:

```
render() {
  return (
     <div>
       <p>Pick a file</p>
       <input type="file" onChange={this.handleUpload.bind(this)} />
       { this.state && <S3Image path={this.state.path} /> }
     </div>
  );
}
```

Put together, the entire component should look like this:

```
// Image upload and download for display example component

import React, { Component } from 'react';
import { Storage } from 'aws-amplify';
import { S3Image } from 'aws-amplify-react';

class ImageViewer extends Component {

  handleUpload(event) {
    const file = event.target.files[0];
    const path = file.name;
    Storage.put(path, file).then(() => this.setState({ path }) );
  }

  render() {
    return (
      <div>
        <p>Pick a file</p>
        <input type="file" onChange={this.handleUpload.bind(this)} />
        { this.state && <S3Image path={this.state.path} /> }
      </div>
    );
  }
}

export default ImageViewer;
```

Next Steps

- Learn how to do private file storage and more with the Storage module in AWS Amplify.
- Learn how to enable more features for your app with the AWS Mobile CLI.
- Learn how to use those features in your app with the AWS Amplify library.
- Learn more about the analytics for the User File Storage feature.

- Learn more about how your files are stored on Amazon Simple Storage Service.

Access Your APIs

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

The AWS Mobile CLI and Amplify library make it easy to create and call cloud APIs and their handler logic from your JavaScript.

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 3).

Create Your API

In the following examples you will create an API that is part of a cloud-enabled number guessing app. The CLI will create a serverless handler for the API behind the scenes.

To enable and configure an API

1. In the root folder of your app, run:

```
awsmobile cloud-api enable --prompt
```

2. When prompted, name the API `Guesses`.

```
? API name: Guesses
```

3. Name a HTTP path `/number`. This maps to a method call in the API handler.

```
? HTTP path name (/items): /number
```

4. Name your Lambda API handler function `guesses`.

```
? Lambda function name (This will be created if it does not already exists): guesses
```

5. When prompted to add another HTTP path, type `N`.

```
? Add another HTTP path (y/N): N
```

6. The configuration for your Guesses API is now saved locally. Push your configuration to the cloud.

```
awsmobile push
```

To test your API and handler

From the command line, run:

```
awsmobile cloud-api invoke Guesses GET /number
```

The Cloud Logic API endpoint for the `Guesses` API is now created.

Customize Your API Handler Logic

The AWS Mobile CLI has generated a Lambda function to handle calls to the `Guesses` API. It is saved locally in `YOUR-APP-ROOT-FOLDER/awsmobilejs/backend/cloud-api/guesses`. The `app.js` file in that directory contains the definitions and functional code for all of the paths that are handled for your API.

To customize your API handler

1. Find the handler for POST requests on the `/number` path. That line starts with `app.post('number',`. Replace the callback function's body with the following:

   ```
   # awsmobilejs/backend/cloud-api/guesses/app.js
   app.post('/number', function(req, res) {
     const correct = 12;
     let guess = req.body.guess
     let result = ""

     if (guess === correct) {
       result = "correct";
     } else if (guess > correct) {
       result = "high";
     } else if (guess < correct) {
       result = "low";
     }

     res.json({ result })
   });
   ```

2. Push your changes to the cloud.

   ```
   awsmobile push
   ```

The `Guesses` API handler logic that implements your new number guessing functionality is now deployed to the cloud.

Connect to Your Backend

The examples in this section show how you would integrate AWS Amplify library calls using React (see the AWS Amplify documentation to use other flavors of Javascript).

The following simple component could be added to a `create-react-app` project to present the number guessing game.

```
// Number guessing game app example

# src/GuessNumber.js

class GuessNumber extends React.Component {
  state = { answer: null };

  render() {
    let prompt = ""
```

```
    const answer = this.state.answer

    switch (answer) {
      case "lower":
        prompt = "Incorrect. Guess a lower number."
      case "higher":
        prompt = "Incorrect. Guess a higher number."
      case "correct":
        prompt = `Correct! The number is ${this.refs.guess.value}!`
      default:
        prompt = "Guess a number between 1 and 100."
    }

    return (
      <div style={styles}>
        <h1>Guess The Number</h1>
        <p>{ prompt }</p>

        <input ref="guess" type="text" />
        <button type="submit">Guess</button>
      </div>
    )

  }
}

let styles = {
  margin: "0 auto",
  width: "30%"
};

export default GuessNumber;
```

Make a Guess

The `API` module from AWS Amplify allows you to send requests to your Cloud Logic APIs right from your JavaScript application.

To make a RESTful API call

1. Import the `API` module from `aws-amplify` in the `GuessNumber` component file.

```
import { API } from 'aws-amplify';
```

2. Add the `makeGuess` function. This function uses the `API` module's `post` function to submit a guess to the Cloud Logic API.

```
async makeGuess() {
  const guess = parseInt(this.refs.guess.value);
  const body = { guess }
  const { result } = await API.post('Guesses', '/number', { body });
  this.setState({
    guess: result
  });
}
```

3. Change the Guess button in the component's `render` function to invoke the `makeGuess` function when it is chosen.

```
<button type="submit" onClick={this.makeGuess.bind(this)}>Guess</button>
```

Open your app locally and test out guessing the number by running `awsmobile run`.

Your entire component should look like the following:

```
// Number guessing game app example

import React from 'react';
import { API } from 'aws-amplify';

class GuessNumber extends React.Component {
  state = { guess: null };

  async makeGuess() {
    const guess = parseInt(this.refs.guess.value, 10);
    const body = { guess }
    const { result } = await API.post('Guesses', '/number', { body });
    this.setState({
      guess: result
    });
  }

  render() {
    let prompt = ""

    switch (this.state.guess) {
      case "high":
        prompt = "Incorrect. Guess a lower number.";
        break;
      case "low":
        prompt = "Incorrect. Guess a higher number.";
        break;
      case "correct":
        prompt = `Correct! The number is ${this.refs.guess.value}!`;
        break;
      default:
        prompt = "Guess a number between 1 and 100.";
    }

    return (
      <div style={styles}>
        <h1>Guess The Number</h1>
        <p>{ prompt }</p>

        <input ref="guess" type="text" />
        <button type="submit" onClick={this.makeGuess.bind(this)}>Guess</button>
      </div>
    )

  }
}

let styles = {
  margin: "0 auto",
  width: "30%"
};

export default GuessNumber;
```

Next Steps

- Learn how to retrieve specific items and more with the API module in AWS Amplify.

- Learn how to enable more features for your app with the AWS Mobile CLI.
- Learn more about what happens behind the scenes, see Set up Lambda and API Gateway.

Host Your Web App

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Topics

About Hosting and Streaming

The first time that you push your web app to the cloud, the Hosting and Streaming feature is enabled to statically host your app on the web. Using the AWS Mobile CLI, this happens when you first run:

```
$ awsmobile publish
```

A container for your content is created using an Amazon S3 bucket. The content is available publicly on the Internet and you can preview the content directly using a testing URL.

Content placed in your bucket is automatically distributed to a global content delivery network (CDN). Amazon CloudFront implements the CDN which can host your app on an endpoint close to every user, globally. These endpoints can also stream media content. To learn more, see CloudFront streaming tutorials.

By default, Hosting and Streaming deploys a simple sample web app that accesses AWS services.

Managing Your App Assets

You can use the AWS Mobile CLI or the Amazon S3 console to manage the content of your bucket.

Use the AWS CLI to Manage Your Bucket Contents

AWS CLI allows you to review, upload, move or delete your files stored in your bucket using the command line. To install and configure the AWS CLI client, see Getting Set Up with the AWS Command Line Interface.

As an example, the sync command enables transfer of files to and from your local folder (`source`) and your bucket (`destination`).

```
$ aws s3 sync {source destination} [--options]
```

The following command syncs all files from your current local folder to the folder in your web app's bucket defined by `path`.

```
```

$ aws s3 sync . s3://AWSDOC-EXAMPLE-BUCKET/path

To learn more about using AWS CLI to manage Amazon S3, see Using Amazon S3 with the AWS Command Line Interface

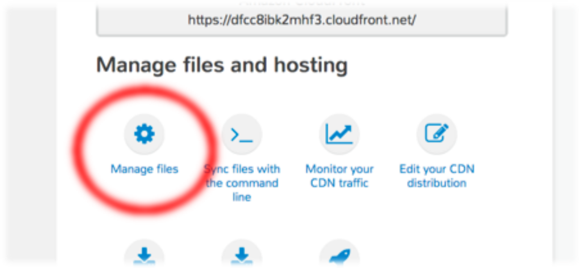
Use the Amazon S3 Console to Manage Your Bucket

To use the Amazon S3 console to review, upload, move or delete your files stored in your bucket, use the following steps.

1. From the root of your project, run:

```
awsmobile console
```

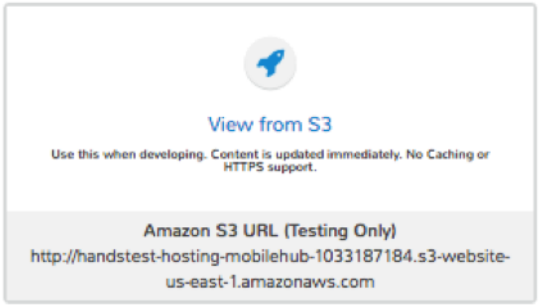
2. Choose the tile with the name of your project, then choose the Hosting and Streaming tile.
3. Choose the link labelled **Manage files** to display the contents of your bucket in the Amazon S3 console.


https://dfcc8ibk2mhf3.cloudfront.net/
Manage files and hosting
Manage files
Sync files with the command line
Monitor your CDN traffic
Edit your CDN distribution

Other Useful Functions in the AWS Mobile Hub Console

The Mobile Hub console also provides convenient ways to browse to your web content, return to the AWS CLI content on this page, and other relevant tasks. These include:

- The **View from S3** link browses to the web contents of your bucket. When Hosting and Streaming is enabled, the bucket is populated with the files for a default web app files that is viewable immediately.

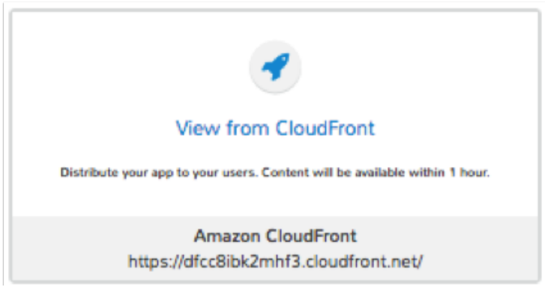


View from S3

Use this when developing. Content is updated immediately. No Caching or HTTPS support.

Amazon S3 URL (Testing Only)
http://handstest-hosting-mobilehub-1033187184.s3-website-us-east-1.amazonaws.com

- The **View from CloudFront** browses to the web contents that have propagated from your bucket to CDN. The endpoint propagation is dependent on network conditions. You can expect your content to be distributed and viewable within one hour.

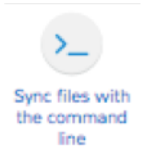


View from CloudFront

Distribute your app to your users. Content will be available within 1 hour.

Amazon CloudFront
https://dfcc8ibk2mhf3.cloudfront.net/

- The **Sync files with the command line** link takes you to content on this page that describes how to use the command line to manage the web app and streaming media files in your bucket.



Configure a Custom Domain for Your Web App

To use your custom domain for linking to your Web app, use the Route 53 service to configure DNS routing.

For a web app hosted in a single location, see Routing Traffic to a Website that Is Hosted in an Amazon S3 Bucket.

For a web app distributed through a global CDN, see Routing Traffic to an Amazon CloudFront Web Distribution by Using Your Domain Name

AWS Mobile Hub Features

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

The following pages contain reference material for the AWS Mobile CLI for Web (JavaScript).

Topics

AWS Mobile CLI Reference

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

The AWS Mobile CLI provides a command line interface for front end JavaScript developers to seamlessly enable AWS services and configure into their apps. With minimal configuration, you can start using all of the functionality provided by the AWS Mobile Hub from your favorite terminal program.

Installation and Usage

This section details the usage and the core commands of the `awsmobile CLI` for JavaScript.

Install AWS Mobile CLI

1. Sign up for the AWS Free Tier.
2. Install Node.js with NPM.
3. Install AWS Mobile CLI

```
npm install -g awsmobile-cli
```

4. Configure the CLI with your AWS credentials

   To setup permissions for the toolchain used by the CLI, run:

   ```
   awsmobile configure
   ```

   If prompted for credentials, follow the steps provided by the CLI. For more information, see provide IAM credentials to AWS Mobile CLI (p. 33).

Usage

The AWS Mobile CLI usage is designed to resemble other industry standard command line interfaces.

```
awsmobile <command> [options]
```

The `help` and `version` options are universal to all the commands. Additional special options for some commands are detailed in the relevant sections.

```
-V, --version  output the version number
-h, --help     output usage information
```

For example:

```
awsmobile -help
or
awsmobile init --help
```

Summary of CLI Commands

The current set of commands supported by the `awsmobile CLI` are listed below.

awsmobile init (p. 26)	Initializes a new Mobile Hub project, checks for IAM keys, and pulls the aws-exports.js file
awsmobile configure (p. 27)	Shows existing keys and allows them to be changed if already set. If keys aren't set, deep links the user to the IAM console to create keys and then prompts for the access key and secret key. This command helps edit configuration settings for the AWS account or the project.
awsmobile pull (p. 28)	Downloads the latest aws-exports.js, YAML or any other relevant project details from the Mobile Hub project
awsmobile push (p. 28)	Uploads local metadata, Lambda code, DynamoDB definitions or any other relevant project details to Mobile Hub
awsmobile publish (p. 28)	Executes `awsmobile push`, then builds and publishes client-side application to S3 and Cloud Front

awsmobile run (p. 29)	Executes `awsmobile push`, then executes the project's start command to test run the client-side application
awsmobile console (p. 29)	Open the web console of the awsmobile Mobile Hub project in the default browser
awsmobile features (p. 29)	Shows available and enabled features. Toggle to select or de-select features.
awsmobile <feature-name> enable [–prompt] (p. 29)	Enables the feature with the defaults (and prompt for changes)
awsmobile <feature-name> disable (p. 31)	Disables the feature
awsmobile <feature-name> configure (p. 31)	Contains feature-specific sub commands like add-table, add-api, etc.
awsmobile cloud-api invoke <apiname> <method> <path> [init] (p. 32)	Invokes the API for testing locally. This helps quickly test unsigned APIs in your local environment.
awsmobile delete (p. 33)	Deletes the Mobile hub project.
awsmobile help [cmd] (p. 33)	Displays help for [cmd].

init

The `awsmobile init` command initializes a new Mobile Hub project, checks for IAM keys, and pulls the aws-exports.js file.

There are two usages of the `awsmobile init` command

1. Initialize the current project with awsmobilejs features

```
awsmobile init
```

When prompted, set these project configs:

```
Please tell us about your project:
? Where is your project's source directory:  src
? Where is your project's distribution directory that stores build artifacts:  build
? What is your project's build command:  npm run-script build
? What is your project's start command for local test run:  npm run-script start

? What awsmobile project name would you like to use:  my-mobile-project
```

The source directory is where the AWS Mobile CLI copies the latest `aws-exports.js` to be easily available for your front-end code. This file is automatically updated every time features are added or removed. Specifying a wrong / unavailable folder will not copy the file over.

The Distribution directly is essentially the build directory for your project. This is used during the `awsmobile publish` process.

The project's build and start values are used during the `awsmobile publish` and `awsmobile run` commands respectively.

The awsmobile project name is the name of the backend project created in the Mobile hub.

You can alter the settings about your project by using the awsmobile configure project (p. 27) command.

2. Initialize and link to an existing awsmobile project as backend

```
awsmobile init <awsmobile-project-id>
```

The awsmobile-project-id is the id of the existing backend project in the Mobile Hub. This command helps attach an existing backend project to your app.

3. Remove the attached awsmobile project from the backend.

```
awsmobile init --remove
```

This command removes the attached backend project associated with your app and cleans the associated files. This will not alter your app in any way, other than removing the backend project itself.

configure

The `awsmobile configure` shows existing keys and allows them to be changed if already set. If keys aren't set, deep links the user to the IAM console to create keys and then prompts for the access key and secret key. There are two possible usages of this command. Based on the argument selected, this command can be used to set or change the AWS account settings OR the project settings.

```
awsmobile configure [aws|project]
```

1. Configuring the AWS account settings using the `aws` argument. This is the default argument for this command

```
awsmobile configure
or
awsmobile configure aws
```

You will be prompted with questions to set the AWS account credentials as below

```
configure aws
? accessKeyId:  <ACCESS-KEY-ID>
? secretAccessKey:  <SECRET-ACCESS-KEY>
? region:  <SELECT-REGION-FROM-THE-LIST>
```

2. Configuring the project settings using the `project` argument

```
awsmobile configure project
```

You will be prompted with questions to configure project as detailed below

```
? Where is your project's source directory:  src
? Where is your project's distribution directory to store build artifacts:  dist
? What is your project's build command:  npm run-script build
? What is your project's start command for local test run:  npm run-script start
```

3. Retrieve and display the AWS credentials using the `--list` option

```
awsmobile configure --list
```

pull

The `awsmobile pull` command downloads the latest aws-exports.js, YAML and any relevant cloud / backend artifacts from the Mobile Hub project to the local dev environment. Use this command if you modified the project on the Mobile Hub and want to get the latest on your local environment.

```
awsmobile pull
```

push

The `awsmobile push` uploads local metadata, Lambda code, Dynamo definitions and any relevant artifacts to Mobile Hub. Use this command when you enable, disable or configure features on your local environment and want to update the backend project on the Mobile Hub with the relevant updates.

```
awsmobile push
```

Use `awsmobile push` after using `awsmobile features`, `awsmobile <feature> enable`, `awsmobile <feature> disable` or `awsmobile <feature> configure` to update the backend project appropriately. This can be used either after each of these or once after all of the changes are made locally.

publish

The `awsmobile publish` command first executes the awsmobile `push` command, then builds and publishes client-side code to Amazon S3 hosting bucket. This command publishes the client application to s3 bucket for hosting and then opens the browser to show the index page. It checks the timestamps to automatically build the app if necessary before deployment. It checks if the client has selected hosting in their backend project features, and if not, it'll prompt the client to update the backend with hosting feature.

```
awsmobile publish
```

The publish command has a number of options to be used.

1. Refresh the Cloud Front distributions

```
awsmobile publish -c
 or
awsmobile publish --cloud-front
```

2. Test the application on AWS Device Farm

```
awsmobile publish -t
or
awsmobile publish --test
```

3. Suppress the tests on AWS Device Farm

```
awsmobile publish -n
```

4. Publish the front end only without updating the backend

```
awsmobile publish -f
or
awsmobile publish --frontend-only
```

run

The `awsmobile run` command first executes the `awsmobile push` command, then executes the start command you set in the project configuration, such as `npm run start` or `npm run ios`. This can be used to conveniently test run your application locally with the latest backend development pushed to the cloud.

```
awsmobile run
```

console

The `awsmobile console` command opens the web console of the awsmobile Mobile Hub project in the default browser

```
awsmobile console
```

features

The `awsmobile features` command displays all the available awsmobile features, and allows you to individually enable/disable them locally. Use the arrow key to scroll up and down, and use the space key to enable/disable each feature. Please note that the changes are only made locally, execute awsmobile push to update the awsmobile project in the cloud.

```
awsmobile features
```

The features supported by the AWS Mobile CLI are:

- user-signin (Amazon Cognito)
- user-files (Amazon S3)
- cloud-api (Lambda / API Gateway)
- database (DynamoDB)
- analytics (Amazon Pinpoint)
- hosting (Amazon S3 and CloudFront)

```
? select features:  (Press <space> to select, <a> to toggle all, <i> to inverse selection)
## user-signin
 # user-files
 # cloud-api
 # database
 # analytics
 # hosting
```

Use caution when disabling a feature. Disabling the feature will delete all the related objects (APIs, Lambda functions, tables etc). These artifacts can not be recovered locally, even if you re-enable the feature.

Use `awsmobile push` after using `awsmobile <feature> disable` to update the backend project on the AWS Mobile Hub project with the selected features.

enable

The `awsmobile <feature> enable` enables the specified feature with the default settings. Please note that the changes are only made locally, execute `awsmobile` push to update the AWS Mobile project in the cloud.

```
awsmobile <feature> enable
```

The features supported by the AWS Mobile CLI are:

- user-signin (Amazon Cognito)
- user-files (Amazon S3)
- cloud-api (Lambda / API Gateway)
- database (DynamoDB)
- analytics (Amazon Pinpoint)
- hosting (Amazon S3 and CloudFront)

The `awsmobile <feature> enable --prompt` subcommand allows user to specify the details of the mobile hub feature to be enabled, instead of using the default settings. It prompts the user to answer a list of questions to specify the feature in detail.

```
awsmobile <feature> enable -- prompt
```

Enabling the `user-signin` feature will prompt you to change the way it is enabled, configure advanced settings or disable sign-in feature to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile user-signin enable --prompt

? Sign-in is currently disabled, what do you want to do next (Use arrow keys)
# Enable sign-in with default settings
  Go to advance settings
```

Enabling the `user-files` feature with the `--prompt` option will prompt you to confirm usage of S3 for user files.

```
awsmobile user-files enable --prompt

? This feature is for storing user files in the cloud, would you like to enable it? Yes
```

Enabling the `cloud-api` feature with the `--prompt` will prompt you to create, remove or edit an API related to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile cloud-api enable --prompt

 ? Select from one of the choices below. (Use arrow keys)
 # Create a new API
```

Enabling the `database` feature with the `--prompt` will prompt you to with initial questions to specify your database table details related to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile database enable --prompt

? Should the data of this table be open or restricted by user? (Use arrow keys)
# Open
  Restricted
```

Enabling the `analytics` feature with the `--prompt` will prompt you to confirm usage of Pinpoint Analytics.

```
 awsmobile analytics enable --prompt

? Do you want to enable Amazon Pinpoint analytics? (y/N)
```

Enabling the `hosting` feature with the `--prompt` will prompt you to confirm hosting and streaming on CloudFront distribution.

```
awsmobile hosting enable --prompt

? Do you want to host your web app including a global CDN? (y/N)
```

Execute `awsmobile push` after using `awsmobile <feature> enable` to update the awsmobile project in the cloud.

disable

The `awsmobile <feature> disable` disables the feature in their backend project. Use caution when disabling a feature. Disabling the feature will delete all the related objects (APIs, Lambda functions, tables etc). These artifacts can not be recovered locally, even if you re-enable the feature.

```
awsmobile <feature> disable
```

The features supported by the AWS Mobile CLI are:

- user-signin (Amazon Cognito)
- user-files (Amazon S3)
- cloud-api (Lambda / API Gateway)
- database (DynamoDB)
- analytics (Amazon Pinpoint)
- hosting `

Use `awsmobile push` after using `awsmobile <feature> disable` to update the backend project on the AWS Mobile Hub project with the disabled features.

configure

The `awsmobile <feature> configure` configures the objects in the selected feature. The configuration could mean adding, deleting or updating a particular artifact. This command can be used only if the specific feature is already enabled.

```
awsmobile <feature> configure
```

The features supported by the AWS Mobile CLI are:

- user-signin (Amazon Cognito)
- user-files (Amazon S3)
- cloud-api (Lambda / API Gateway)
- database (DynamoDB)
- analytics (Amazon Pinpoint)
- hosting (Amazon S3 and CloudFront)

Configuring the `user-signin` feature will prompt you to change the way it is enabled, configure advanced settings or disable sign-in feature to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile user-signin configure

? Sign-in is currently enabled, what do you want to do next (Use arrow keys)
# Configure Sign-in to be required (Currently set to optional)
  Go to advance settings
  Disable sign-in
```

Configuring the `user-files` feature will prompt you to confirm usage of S3 for user files.

```
 awsmobile user-files configure

? This feature is for storing user files in the cloud, would you like to enable it? (Y/n)
```

Configuring the `cloud-api` feature will prompt you to create, remove or edit an API related to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile cloud-api configure

? Select from one of the choices below. (Use arrow keys)
# Create a new API
  Remove an API from the project
  Edit an API from the project
```

Configuring the `database` feature will prompt you to create, remove or edit a table related to the project. Selecting the desired option may prompt you with further questions.

```
awsmobile database configure

  ? Select from one of the choices below. (Use arrow keys)
  # Create a new table
    Remove table from the project
    Edit table from the project
```

Configuring the `analytics` feature will prompt you to confirm usage of Pinpoint Analytics.

```
awsmobile analytics configure

? Do you want to enable Amazon Pinpoint analytics? Yes
```

Configuring the `hosting` feature will prompt you to confirm hosting and streaming on CloudFront distribution.

```
awsmobile hosting configure

? Do you want to host your web app including a global CDN? Yes
```

Use `awsmobile push` after using `awsmobile <feature> configure` to update the backend project on the AWS Mobile Hub project with the configured features.

invoke

The `awsmobile cloud-api invoke` invokes the API for testing locally. This helps quickly test the unsigned API locally by passing the appropriate arguments. This is intended to be used for the development environment or debugging of your API / Lambda function.

```
awsmobile cloud-api invoke <apiname> <method> <path> [init]
```

For example you could invoke the sampleCloudApi post method as shown below

```
awsmobile cloud-api invoke sampleCloudApi post /items '{"body":{"test-key":"test-value"}}'
```

The above test will return a value that looks like

```
{ success: 'post call succeed!',
  url: '/items',
  body: { 'test-key': 'test-value' } }
```

Similarly, you could invoke the sampleCloudApi get method as shown below

```
awsmobile cloud-api invoke sampleCloudApi get /items
```

The above test will return a value that looks like

```
{ success: 'get call succeed!', url: '/items' }
```

delete

The `awsmobile delete` command deletes the Mobile hub project in the cloud. Use extra caution when you decide to execute this command, as it can irrevocably affect your team's work, the mobile hub project will be delete and cannot be recovered once this command is executed.

```
awsmobile delete
```

help

The `awsmobile help` command can be used as a standalone command or the command name that you need help in can be passed as an argument. This gives the usage information for that command including any options that can be used with it.

For Example:

```
awsmobile help
or
awsmobile help init
```

The `--help` option detailing at the beginning of this page and the `awsmobile help` command provide the same level of detail. The difference is in the usage.

AWS Mobile CLI User Credentials

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Overview

The first time you set up the CLI you will be prompted to provide AWS user credentials. The credentials establish permissions for the CLI to manage AWS services on your behalf. They must belong to an AWS IAM user with administrator permissions in the account where the CLI is being used.

Permissions

Administrator permissions are granted by an AWS account administrator. If don't have administrator permissions you will need to ask an administrator for the AWS account to grant them.

If you are the account owner and signed in under the root credentials for the account, then you have, or can grant yourself, administrator permissions using the `AdministratorAccess` managed policy. Best practice is to create a new IAM user under your account to access AWS services instead of using root credentials.

For more information, see Control Access to Mobile Hub Projects (p. 77).

Get Account User Credentials

If you have administrator permissions, the values you need to provide the CLI are your IAM user's Access Key ID and a Secret Access Key. If not, you will need to get these from an administrator.

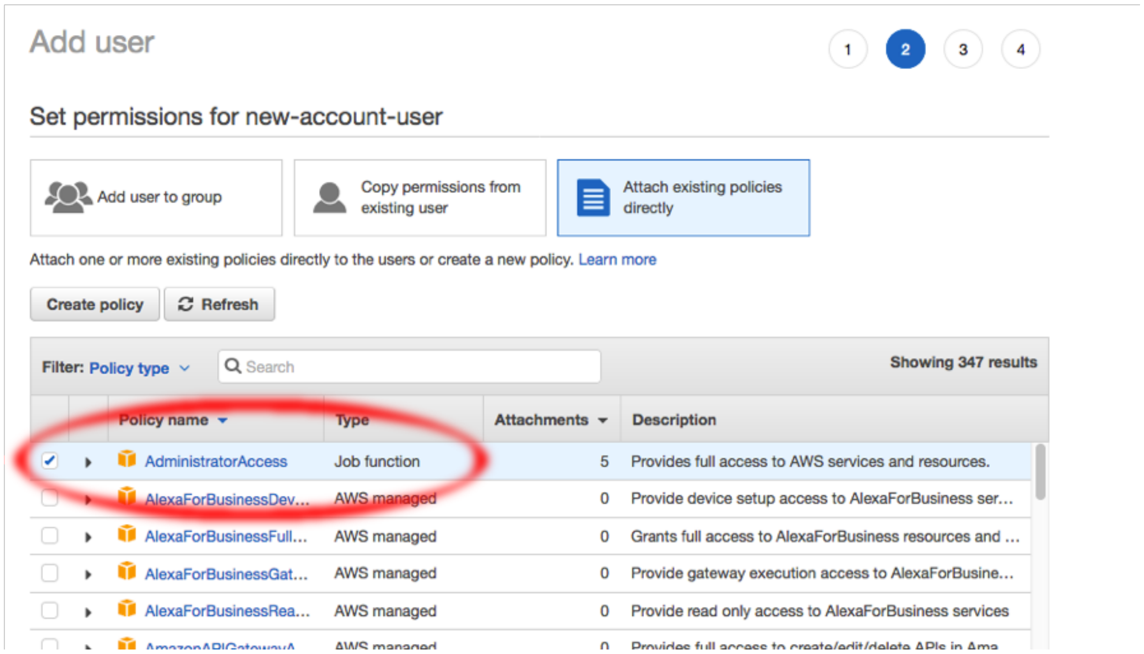
To provide the ID and the Key to AWS CLI, follow the CLI prompts to sign-in to AWS, and provide a user name and AWS region. The CLI will open the AWS IAM console **Add user** dialog, with the `AdministratorAccess` policy attached, and the **Programmatic access** option selected by default.

Topics

Get credentials for a new user

1. Choose **Next: Permissions** and then choose **Create user**.

   Alternatively, you could add the user to a group with `AdministratorAccess` attached.



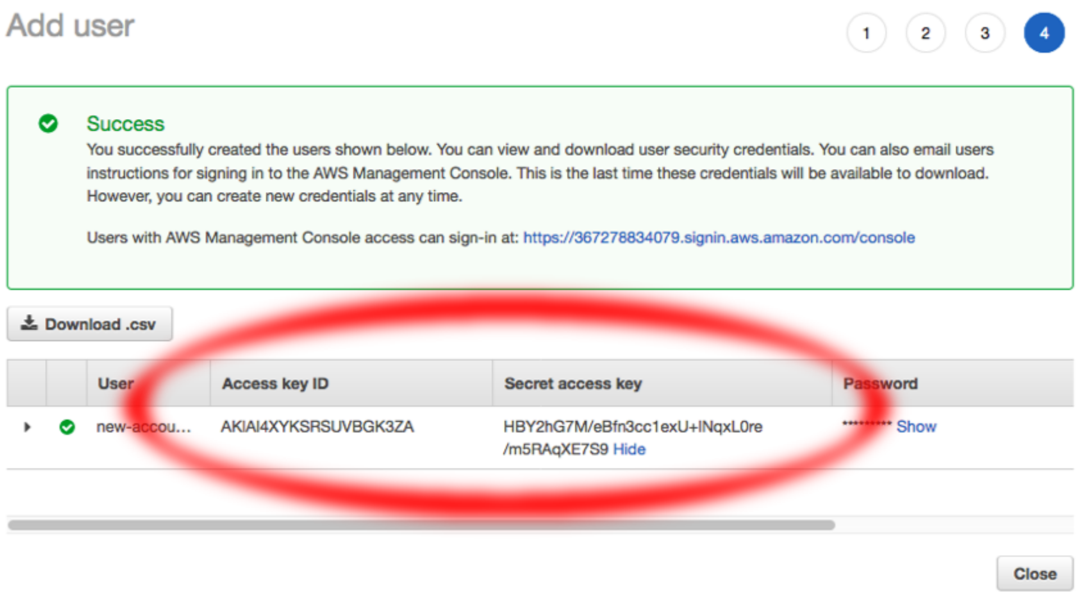
Set permissions for new-account-user

Attach one or more existing policies directly to the users or create a new policy. Learn more

Showing 347 results

		Policy name	Type	Attachments	Description
✓	▸	AdministratorAccess	Job function	5	Provides full access to AWS services and resources.
	▸	AlexaForBusinessDev...	AWS managed	0	Provide device setup access to AlexaForBusiness ser...
	▸	AlexaForBusinessFull...	AWS managed	0	Grants full access to AlexaForBusiness resources and ...
	▸	AlexaForBusinessGat...	AWS managed	0	Provide gateway execution access to AlexaForBusine...
	▸	AlexaForBusinessRea...	AWS managed	0	Provide read only access to AlexaForBusiness services
	▸	AmazonAPIGatewayA	AWS managed	0	Provides full access to create/edit/delete APIs in Ama

2. Choose **Create user**.
3. Copy the values from the table displayed, or choose **Download .csv** to save the values locally, and then type them into the prompts.



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://367278834079.signin.aws.amazon.com/console

		User	Access key ID	Secret access key	Password
▸	✓	new-accou...	AKIAI4XYKSRSUVBGK3ZA	HBY2hG7M/eBfn3cc1exU+INqxL0re /m5RAqXE7S9 Hide	********* Show

For more detailed steps, see add a new account user with administrator permissions (p. 78).

Get credentials for an existing user

1. Choose **cancel**.
2. On the left, choose **Users**, then select the user from the list. Choose **Security credentials**, then choose **Create access key**.

Logging AWS Mobile CLI API Calls with AWS CloudTrail

> **Important**
> The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

The AWS Mobile CLI is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in the CLI. CloudTrail captures all API calls for the CLI as events, including calls from code calls to the CLI APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for the CLI. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to the CLI, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS Mobile CLI Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Mobile CLI, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS Mobile CLI, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS Mobile CLI actions are logged by CloudTrail and are documented in the AWS Mobile CLI API Reference (p. 24). For example, calls to the `awsmobile init`, `awsmobile pull` and `awsmobile push` generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding AWS Mobile ClI Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `ListProjects` action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEFGHIJK0123456789",
        "arn": "arn:aws:iam::012345678901:user/Administrator",
        "accountId": "012345678901",
        "accessKeyId": "ABCDEFGHIJK0123456789",
        "userName": "YOUR_ADMIN_USER_NAME"
    },
    "eventTime": "2017-12-18T23:10:13Z",
    "eventSource": "mobilehub.amazonaws.com",
    "eventName": "ListProjects",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "111.111.111.111",
    "userAgent": "aws-cli/1.11.140 Python/2.7.13 Darwin/15.6.0 botocore/1.6.7 ",
    "requestParameters": {
        "maxResults": 0
    },
    "responseElements": {
        "projects": [{
            "name": "YOUR_PROJECT_NAME-0123456789012",
            "projectId": "abcd0123-0123-0123-0123-abcdef012345"
        }]
    },
    "requestID": "abcd0123-0123-0123-0123-abcdef012345",
    "eventID": "abcd0123-0123-0123-0123-abcdef012345",
    "eventType": "AwsApiCall",
    "recipientAccountId": "012345678901"
}
```

AWS Amplify Library for React Native

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

AWS Amplify is an open source JavaScript library for frontend and mobile developers building cloud-enabled applications. The library is a declarative interface across different categories of operations in order to make common tasks easier to add into your application. The default implementation works with Amazon Web Services (AWS) resources but is designed to be open and pluggable for usage with other cloud services that wish to provide an implementation or custom backends.

The AWS Mobile CLI, built on AWS Mobile Hub, provides a command line interface for frontend JavaScript developers to seamlessly enable and configure AWS services into their apps. With minimal configuration, you can start using all of the functionality provided by the AWS Mobile Hub from your favorite terminal application.

Topics

Get Started

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Overview

The AWS Mobile CLI provides a command line experience that allows frontend JavaScript developers to quickly create and integrate AWS backend resources into their mobile apps.

Prerequisites

1. Sign up for the AWS Free Tier to learn and prototype at little or no cost.
2. Install Node.js with NPM.
3. Install the AWS Mobile CLI

```
npm install --global awsmobile-cli
```

4. Configure the CLI with your AWS credentials

To setup permissions for the toolchain used by the CLI, run:

```
awsmobile configure
```

If prompted for credentials, follow the steps provided by the CLI. For more information, see Provide IAM credentials to AWS Mobile CLI (p. 33).

Set Up Your Backend

Need to create a quick sample React Native app? See Create a React Native App.

To configure backend features for your app

1. In the root folder of your app, run:

```
awsmobile init
```

The `init` command creates a backend project for your app. By default, analytics and web hosting are enabled in your backend and this configuration is automatically pulled into your app when you initialize.

2. When prompted, provide the source directory for your project. The CLI will generate `aws-exports.js` in this location. This file contains the configuration and endpoint metadata used to link your frontend to your backend services.

```
? Where is your project's source directory:  /
```

Then respond to further prompts with the following values.

```
Please tell us about your project:
? Where is your project's source directory:  /
? Where is your project's distribution directory that stores build artifacts:  build
? What is your project's build command:  npm run-script build
? What is your project's start command for local test run:  npm run-script start
```

Connect to Your Backend

AWS Mobile uses the open source AWS Amplify library to link your code to the AWS features configured for your app.

To connect the app to your configured AWS services

1. Install AWS Amplify for React Native library.

```
npm install --save aws-amplify
```

2. In `App.js` (or in other code that runs at launch-time), add the following imports.

```
import Amplify from 'aws-amplify';

import aws_exports from './YOUR-PATH-TO/aws-exports';
```

3. Then add the following code.

```
Amplify.configure(aws_exports);
```

Run Your App Locally

Your app is now ready to launch and use the default services configured by AWS Mobile.

To launch your app locally

Use the command native to the React Native tooling you are using. For example, if you made your app using `create-react-native-app` then run:

```
npm run android

# OR

npm run ios
```

Anytime you launch your app, app usage analytics are gathered and can be visualized (p. 39) in an AWS console.

AWS Free Tier	Initializing your app or adding features through the CLI will cause AWS services to be configured on your behalf. The pricing for AWS Mobile services enables you to learn and prototype at little or no cost using the AWS Free Tier.

Next Steps

Add Features

Add the following AWS Mobile features to your mobile app using the CLI.

- Analytics (p. 39)
- User Sign-in (p. 41)
- NoSQL Database (p. 42)
- User File Storage (p. 47)
- Cloud Logic (p. 48)

Learn more

To learn more about the commands and usage of the AWS Mobile CLI, see the AWS Mobile CLI reference (p. 24).

Learn about AWS Mobile Amplify.

Add Analytics

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to

your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Basic Analytics Backend is Enabled for Your App

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 37).

When you complete the AWS Mobile CLI setup and launch your app, anonymized session and device demographics data flows to the AWS analytics backend.

To send basic app usage analytics to AWS

Launch your app locally, for instance, if you created your app using `create-react-native-app`, by running:

```
npm run android

# Or

npm run ios
```

When you use your app the Amazon Pinpoint service gathers and visualizes analytics data.

To view the analytics using the Amazon Pinpoint console

1. Launch your app at least once.
2. Open your project in the AWS Mobile Hub console.

   ```
   awsmobile console
   ```
3. Choose the **Analytics** icon on the left, to navigate to your project in the Amazon Pinpoint console.
4. Choose **Analytics** on the left.

You should see an up-tick in several graphs.

Add Custom Analytics to Your App

You can configure your app so that Amazon Pinpoint gathers data for custom events that you register within the flow of your code.

To instrument custom analytics in your app

In the file containing the event you want to track, add the following import:

```
import { Analytics } from 'aws-amplify';
```

Add the a call like the following to the spot in your JavaScript where the tracked event should be fired:

```
componentDidMount() {
   Analytics.record('FIRST-EVENT-NAME');
}
```

Or to relevant page elements:

```
handleClick = () => {
     Analytics.record('SECOND-EVENT-NAME');
}

<Button title="Record event" onPress={this.handleClick}/>
```

To test:

1. Save the changes and launch your app. Use your app so that tracked events are triggered.
2. In the Amazon Pinpoint console, choose **Events** near the top.
3. Select an event in the **Event** dropdown menu on the left.

Custom event data may take a few minutes to become visible in the console.

Next Steps

Learn more about the analytics in AWS Mobile which are part of the Messaging and Analytics (p. 59) feature. This feature uses Amazon Pinpoint.

Learn about AWS Mobile CLI (p. 24).

Learn about the AWS Amplify for React Native library.

Add Auth / User Sign-in

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 37).

The AWS Mobile CLI components for user authentication include a rich, configurable UI for sign-up and sign-in.

To enable the Auth features

In the root folder of your app, run:

```
awsmobile user-signin enable

awsmobile push
```

Connect to Your Backend

The AWS Mobile CLI enables you to integrate ready-made sign-up/sign-in/sign-out UI from the command line.

To add user auth UI to your app

1. Install AWS Amplify for React Nativelibrary.

```
npm install --save aws-amplify
npm install --save aws-amplify-react-native
```

Note	If your react-native app was not created using `create-react-native-app` or using a version of Expo lower than v25.0.0 (the engine behind `create-react-native-app`), you need to link libraries in your project for the Auth module on React Native, `amazon-cognito-identity-js`.
	
	To link to the module, you must first eject the project:
	
	`npm run eject`
	`react-native link amazon-cognito-identity-js`

1. Add the following import in `App.js` (or other file that runs upon app startup):

```
import { withAuthenticator } from 'aws-amplify-react-native';
```

2. Then change `export default App;` to the following.

```
export default withAuthenticator(App);
```

To test, run `npm start` or `awsmobile run`.

Next Steps

Learn more about the AWS Mobile User Sign-in (p. 67) feature, which uses Amazon Cognito.

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

Access Your Database

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 37).

AWS Mobile `database` feature enables you to create tables customized to your needs. The CLI then guides you to create a custom API to access your database.

Create a table

To specify and create a table

1. In your app root folder, run:

```
awsmobile database enable --prompt
```

2. Design your table when prompted by the CLI.

   The CLI will prompt you for the table and other table configurations such as columns.

```
Welcome to NoSQL database wizard
You will be asked a series of questions to help determine how to best construct your
 NoSQL database table.

? Should the data of this table be open or restricted by user? Open
? Table name Notes


You can now add columns to the table.

? What would you like to name this column NoteId
? Choose the data type string
? Would you like to add another column Yes
? What would you like to name this column NoteTitle
? Choose the data type string
? Would you like to add another column Yes
? What would you like to name this column NoteContent
? Choose the data type string
? Would you like to add another column No
```

   Choose a Primary Key that will uniquely identify each item. Optionally, choose a column to be a Sort Key when you will commonly use those values in combination with the Primary Key for sorting or searching your data. You can additional sort keys by adding a Secondary Index for each column you will want to sort by.

```
Before you create the database, you must specify how items in your table are uniquely
 organized. This is done by specifying a Primary key. The primary key uniquely identifies
 each item in the table, so that no two items can have the same key.
This could be and individual column or a combination that has "primary key" and a "sort
 key".
To learn more about primary key:
http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.CoreComponents.html#HowItWorks.CoreComponents.PrimaryKey


? Select primary key NoteId
? Select sort key (No Sort Key)

You can optionally add global secondary indexes for this table. These are useful when
 running queries defined by a different column than the primary key.

To learn more about indexes:

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.CoreComponents.html#HowItWorks.CoreComponents.SecondaryIndexes

? Add index No
```

```
Table Notes added
```

Create a CRUD API

AWS Mobile will create a custom API for your app to perform create, read, update, and delete (CRUD) actions on your database.

To create a CRUD API for your table

1. In the root folder of your app, run:

```
awsmobile cloud-api enable --prompt
```

2. When prompted, choose `Create CRUD API for existing Dynamo table`, select the table name from the previous steps, choose the access permissions for the table. Using the example table from the previous section:

```
? Select from one of the choices below.
  Create a new API
# Create CRUD API for an existing Amazon DynamoDB table
```

   The prompt response will be:

```
Path to be used on API for get and remove an object should be like:
/Notes/object/:NoteId

Path to be used on API for list objects on get method should be like:
/Notes/:NoteId

JSON to be used as data on put request should be like:
{
  "NoteTitle": "INSERT VALUE HERE",
  "NoteContent": "INSERT VALUE HERE",
  "NoteId": "INSERT VALUE HERE"
}
To test the api from the command line (after awsmobile push) use this commands
awsmobile cloud-api invoke NotesCRUD <method> <path> [init]
Api NotesCRUD saved
```

   Copy and keep the path of your API and the JSON for use in your app code.

   This feature will create an API using Amazon API Gateway and AWS Lambda. You can optionally have the lambda function perform CRUD operations against your Amazon DynamoDB table.

3. Update your backend.

   To create the API you have configured, run:

```
awsmobile push
```

   Until deployment of API to the cloud the has completed, the CLI displays the message: `cloud-api update status: CREATE_IN_PROGRESS`. Once deployed a successful creation message `cloud-api update status: CREATE_COMPLETE` is displayed.

   You can view the API that the CLI created by running `awmobile console` and then choosing **Cloud Logic** in the Mobile Hub console.

Connect to Your Backend

Topics

To access to database tables from your app

1. In `App.js` import the following.

```
import Amplify, { API } from 'aws-amplify';
import aws_exports from 'path_to_your_aws-exports';
Amplify.configure(aws_exports);
```

2. Add the following `state` to your component.

```
state = {
  apiResponse: null,
  noteId: ''
     };

  handleChangeNoteId = (event) => {
        this.setState({noteId: event});
}
```

Save an item (create or update)

To save an item

In the part of your app where you access the database, such as an event handler in your React component, call the `put` method. Use the JSON and the root path (`/Notes`) of your API that you copied from the CLI prompt response earlier.

```
// Create a new Note according to the columns we defined earlier
  async saveNote() {
    let newNote = {
      body: {
        "NoteTitle": "My first note!",
        "NoteContent": "This is so cool!",
        "NoteId": this.state.noteId
      }
    }
    const path = "/Notes";

    // Use the API module to save the note to the database
    try {
      const apiResponse = await API.put("NotesCRUD", path, newNote)
      console.log("response from saving note: " + apiResponse);
      this.setState({apiResponse});
    } catch (e) {
      console.log(e);
    }
  }
```

To use the command line to see your saved items in the database run:

```
awsmobile cloud-api invoke NotesCRUD GET /Notes/object/${noteId}
```

Get a specific item

To query for a specific item

Call the `get` method using the API path (copied earlier) to the item you are querying for.

```
// noteId is the primary key of the particular record you want to fetch
    async getNote() {
      const path = "/Notes/object/" + this.state.noteId;
      try {
        const apiResponse = await API.get("NotesCRUD", path);
        console.log("response from getting note: " + apiResponse);
        this.setState({apiResponse});
      } catch (e) {
        console.log(e);
      }
    }
```

Delete an item

To delete an item

Add this method to your component. Use your API path (copied earlier).

```
// noteId is the NoteId of the particular record you want to delete
    async deleteNote() {
      const path = "/Notes/object/" + this.state.noteId;
      try {
        const apiResponse = await API.del("NotesCRUD", path);
        console.log("response from deleting note: " + apiResponse);
        this.setState({apiResponse});
      } catch (e) {
        console.log(e);
      }
    }
```

UI to exercise CRUD calls

The following is and example of how you might construct UI to exercise these operations.

```
<View style={styles.container}>
        <Text>Response: {this.state.apiResponse &&
 JSON.stringify(this.state.apiResponse)}</Text>
        <Button title="Save Note" onPress={this.saveNote.bind(this)} />
        <Button title="Get Note" onPress={this.getNote.bind(this)} />
        <Button title="Delete Note" onPress={this.deleteNote.bind(this)} />
        <TextInput style={styles.textInput} autoCapitalize='none'
 onChangeText={this.handleChangeNoteId}/>
</View>

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  textInput: {
      margin: 15,
      height: 30,
      width: 200,
      borderWidth: 1,
      color: 'green',
      fontSize: 20,
      backgroundColor: 'black'
   }
});
```

Next Steps

Learn more about the AWS Mobile NoSQL Database (p. 54) feature, which uses Amazon DynamoDB.

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

Add Storage

> **Important**
> The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

The AWS Mobile CLI User File Storage (p. 72) feature enables apps to store user files in the cloud.

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 37).

To configure your app's cloud storage location

In your app root folder, run:

```
awsmobile user-files enable

awsmobile push
```

Connect to Your Backend

To add User File Storage to your app

In your component where you want to transfer files:

Import the `Storage` module from aws-amplify and configure it to communicate with your backend.

```
import { Storage } from 'aws-amplify';
```

Now that the Storage module is imported and ready to communicate with your backend, implement common file transfer actions using the code below.

Upload a file

To upload a file to storage

Add the following methods to the component where you handle file uploads.

```
async uploadFile() {
  let file = 'My upload text';
  let name = 'myFile.txt';
  const access = { level: "public" }; // note the access path
  Storage.put(name, file, access);
}
```

Get a specific file

To download a file from cloud storage

Add the following code to a component where you display files.

```
async getFile() {
  let name = 'myFile.txt';
  const access = { level: "public" };
  let fileUrl = await Storage.get(name, access);
  // use fileUrl to get the file
}
```

List all files

To list the files stored in the cloud for your app

Add the following code to a component where you list a collection of files.

```
async componentDidMount() {
  const path = this.props.path;
  const access = { level: "public" };
  let files = await Storage.list(path, access);
   // use file list to get single files
}
```

Use the following code to fetch file attributes such as the size or time of last file change.

```
file.Size; // file size
file.LastModified.toLocaleDateString(); // last modified date
file.LastModified.toLocaleTimeString(); // last modified time
```

Delete a file

Add the following state to the element where you handle file transfers.

```
async deleteFile(key) {
  const access = { level: "public" };
  Storage.remove(key, access);
}
```

Next Steps

Learn more about the analytics in AWS Mobile which are part of the User File Storage (p. 72) feature. This feature uses Amazon Simple Storage Service (S3).

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

Access Your APIs

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to

your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

Set Up Your Backend

BEFORE YOU BEGIN	The steps on this page assume you have already completed the steps on Get Started (p. 37).

The AWS Mobile Cloud Logic (p. 51) feature lets you call APIs in the cloud. API calls are handled by your serverless Lambda functions.

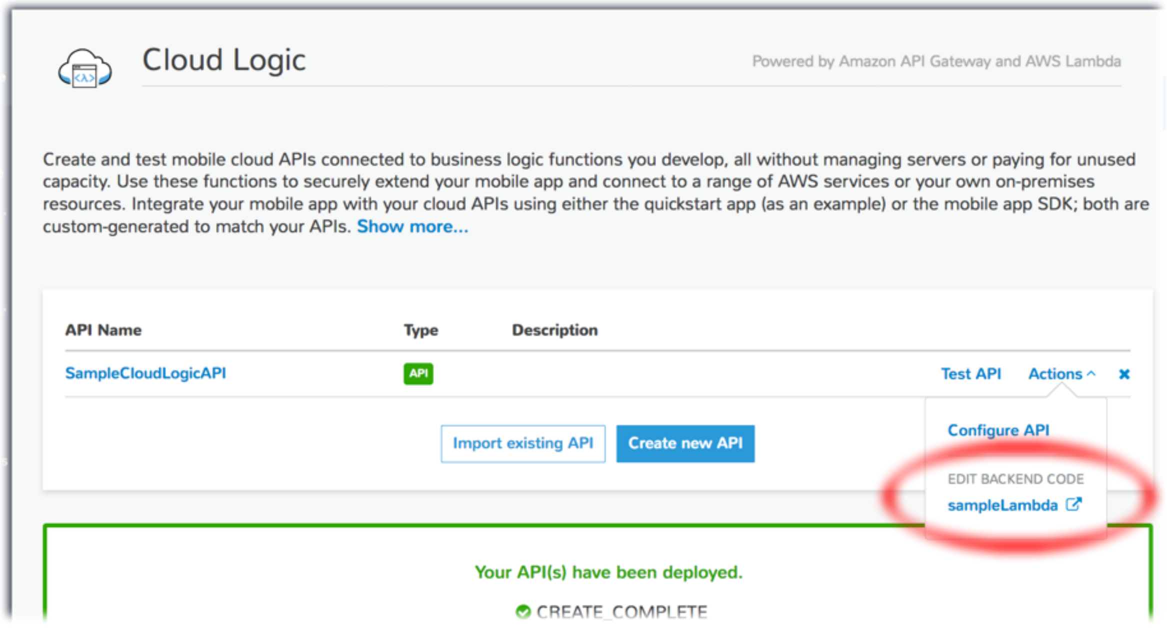
To enable cloud APIs in your app

```
awsmobile cloud-api enable

awsmobile push
```

Enabling Cloud Logic in your app adds a sample API, `sampleCloudApi` to your project that can be used for testing.

You can find the sample handler function for the API by running `awsmobile console` in your app root folder, and then choosing the **Cloud Logic** feature in your Mobile Hub project.



Create and test mobile cloud APIs connected to business logic functions you develop, all without managing servers or paying for unused capacity. Use these functions to securely extend your mobile app and connect to a range of AWS services or your own on-premises resources. Integrate your mobile app with your cloud APIs using either the quickstart app (as an example) or the mobile app SDK; both are custom-generated to match your APIs. Show more...

API Name	Type	Description
SampleCloudLogicAPI	API	

Your API(s) have been deployed.

CREATE_COMPLETE

Quickly Test Your API From the CLI

The `sampleCloudApi` and its handler function allow you to make end to end API calls.

To test invocation of your unsigned APIs in the development environment

```
awsmobile cloud-api invoke <apiname> <method> <path> [init]
```

For the `sampleCloudApi` you may use the following examples to test the `post` method

```
awsmobile cloud-api invoke sampleCloudApi post /items '{"body": {"testKey":"testValue"}}'
```

This call will return a response similar to the following.

```
{ success: 'post call succeed!',
url: '/items',
body: { testKey: 'testValue' } }
```

To test the :get method

```
awsmobile cloud-api invoke sampleCloudApi get /items
```

This will return a response as follows.

```
{ success: 'get call succeed!', url: '/items' }
```

Connect to Your Backend

Once you have created your own Cloud Logic (p. 51) APIs and Lambda functions, you can call them from your app.

To call APIs from your app

In `App.js` (or other code that runs at launch-time), add the following import.

```
import Amplify, { API } from 'aws-amplify';
import aws_exports from './aws-exports';
Amplify.configure(aws_exports);
```

Then add this to the component that calls your API.

```
state = { apiResponse: null };

async getSample() {
 const path = "/items"; // you can specify the path
  const apiResponse = await API.get("sampleCloudApi" , path); //replace the API name
  console.log('response:' + apiResponse);
  this.setState({ apiResponse });
}
```

To invoke your API from a UI element, add an API call from within your component's `render()` method.

```
<View>
   <Button title="Send Request" onPress={this.getSample.bind(this)} />
   <Text>Response: {this.state.apiResponse && JSON.stringify(this.state.apiResponse)}</
Text>
</View>
```

To test, save the changes, run `npm run android` or `npm run ios`` to launch your app. Then try the UI element that calls your API.

Next Steps

Learn more about the AWS Mobile Cloud Logic (p. 51) feature which uses Amazon API Gateway and AWS Lambda.

To be guided through creation of an API and it's handler, run `awsmobile console` to open your app in the Mobile Hub console, and choose **Cloud Logic**.

Learn about AWS Mobile CLI (p. 24).

Learn about AWS Mobile Amplify.

AWS Mobile Hub Features

Important
The following content applies if you are already using the AWS Mobile CLI to configure your backend. If you are building a new mobile or web app, or you're adding cloud capabilities to your existing app, use the new AWS Amplify CLI instead. With the new Amplify CLI, you can use all of the features described in Announcing the AWS Amplify CLI toolchain, including AWS CloudFormation functionality that provides additional workflows.

The following pages contain reference material for the AWS Mobile CLI for Web (JavaScript).

- AWS Mobile CLI Reference (p. 24)
- AWS Mobile CLI Credentials (p. 33)

AWS Mobile Hub Features

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

AWS Mobile Hub is a service that enables even a novice to easily deploy and configure mobile app backend features using a range of powerful AWS services.

You create a free project, then choose and configure mobile app features using a point and click console. Mobile Hub takes care of the complexities in the background and then supplies you with step by step integration instructions.

Topics

Cloud Logic

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

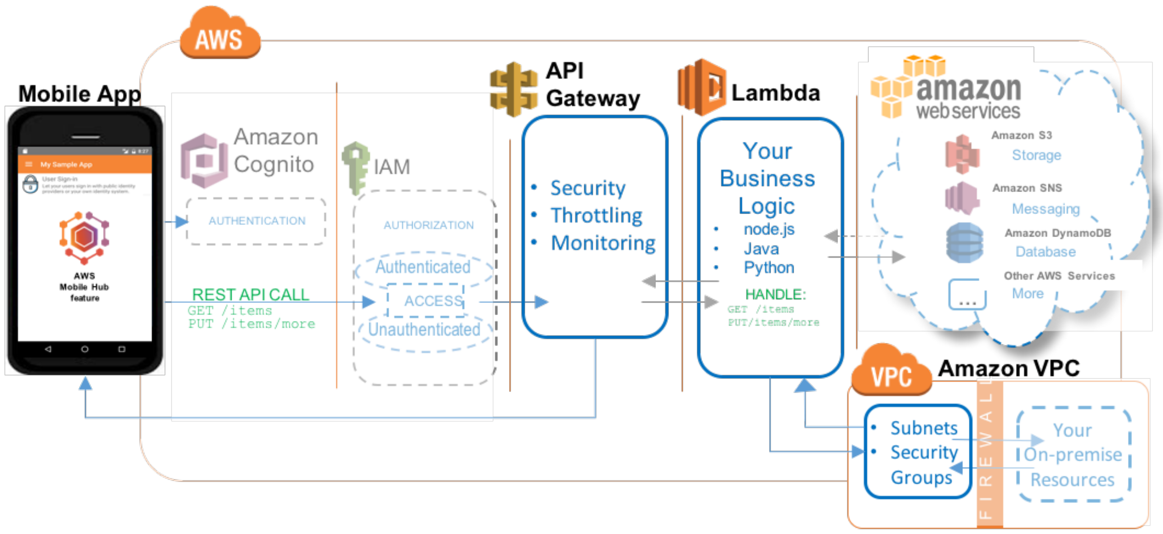
Choose the AWS Mobile Hub Cloud Logic mobile backend service feature to:

- Add business logic functions in the cloud with no cost for server set up or maintenance
- Extend your app to other services within AWS and beyond

Create a free Mobile Hub project and add the Cloud Logic feature.

Feature Details

The following image show Cloud Logic using the combination of Amazon API Gateway and AWS Lambda to implement serverless business logic and extension to other services.



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.

The Lambda functions you create are exposed to your app as a REST API by Amazon API Gateway which also provides a single secure endpoint with flexible traffic monitoring and throttling capabilities.

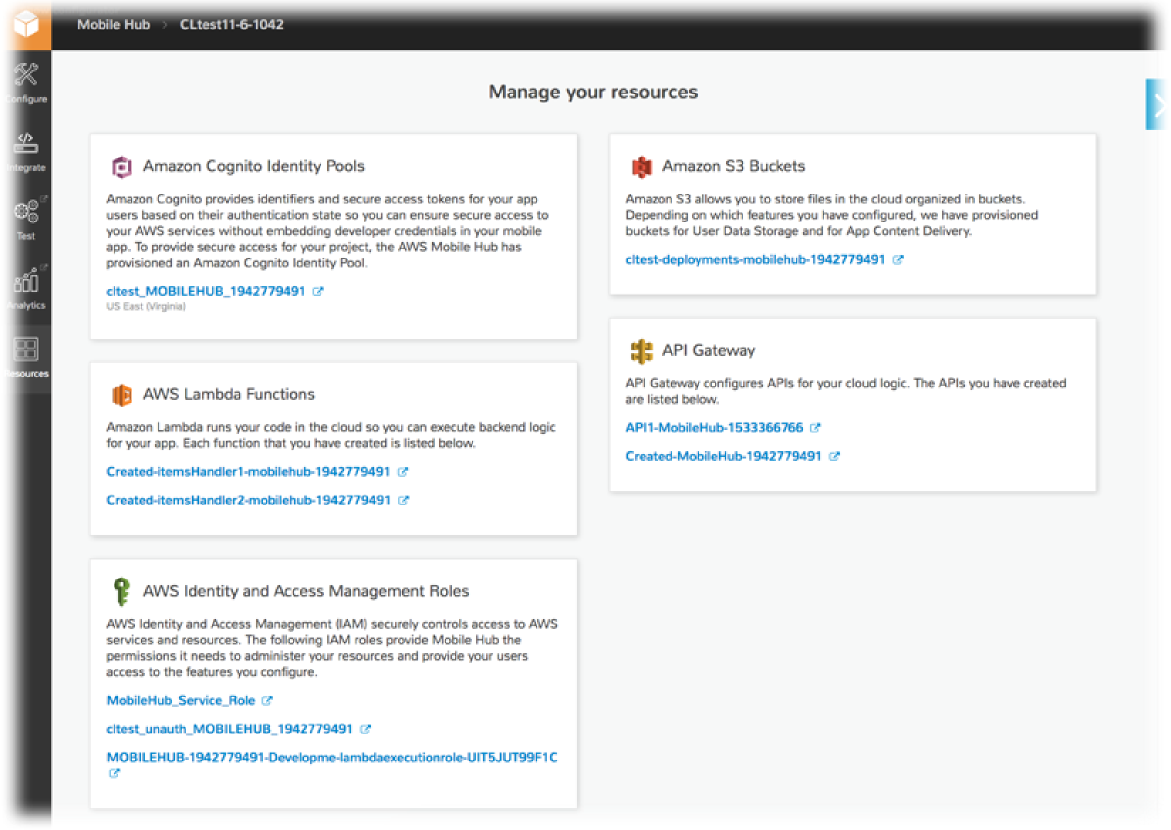
Cloud Logic At a Glance

AWS services and resources configured	• **Amazon API Gateway** (see Amazon API Gateway Developer Guide) Concepts | Console | Pricing • **AWS Lambda** (see AWS Lambda Developer Guide) Concepts | Console | Pricing • **Amazon Virtual Private Cloud** (see Amazon VPC User Guide) Concepts | Console | Pricing • **AWS CloudFormation** (see AWS CloudFormation User Guide) Concepts | Console | Pricing

	Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67). For more information, see Viewing AWS Resources Provisioned for this Feature (p. 53).
Configuration options	This feature enables the following mobile backend capabilities: • 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	This feature adds the following functionality to a quickstart app generated by Mobile Hub: • 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.



Manage your resources

Amazon Cognito Identity Pools

Amazon Cognito provides identifiers and secure access tokens for your app users based on their authentication state so you can ensure secure access to your AWS services without embedding developer credentials in your mobile app. To provide secure access for your project, the AWS Mobile Hub has provisioned an Amazon Cognito Identity Pool.

cltest_MOBILEHUB_1942779491
US East (Virginia)

AWS Lambda Functions

Amazon Lambda runs your code in the cloud so you can execute backend logic for your app. Each function that you have created is listed below.

Created-itemsHandler1-mobilehub-1942779491

Created-itemsHandler2-mobilehub-1942779491

AWS Identity and Access Management Roles

AWS Identity and Access Management (IAM) securely controls access to AWS services and resources. The following IAM roles provide Mobile Hub the permissions it needs to administer your resources and provide your users access to the features you configure.

MobileHub_Service_Role

cltest_unauth_MOBILEHUB_1942779491

MOBILEHUB-1942779491-Developme-lambdaexecutionrole-UIT5JUT99F1C

Amazon S3 Buckets

Amazon S3 allows you to store files in the cloud organized in buckets. Depending on which features you have configured, we have provisioned buckets for User Data Storage and for App Content Delivery.

cltest-deployments-mobilehub-1942779491

API Gateway

API Gateway configures APIs for your cloud logic. The APIs you have created are listed below.

API1-MobileHub-1533366766

Created-MobileHub-1942779491

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

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

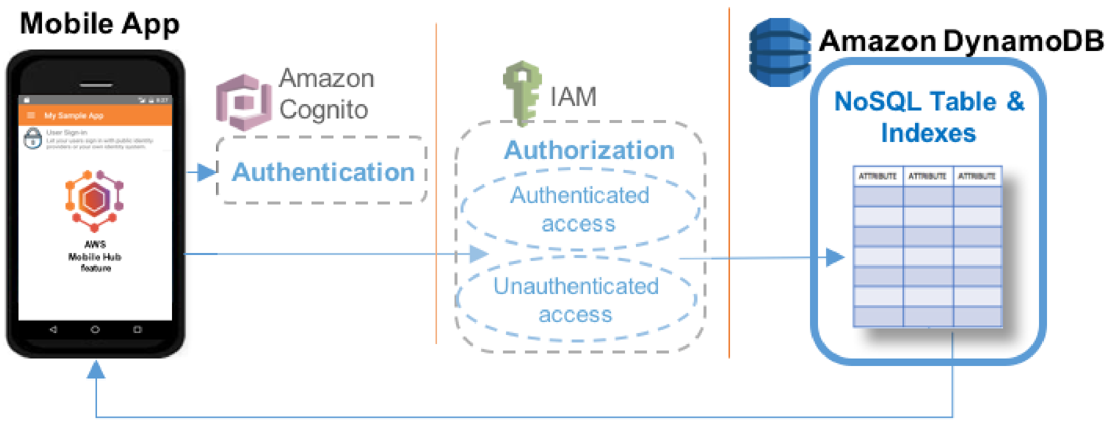
Choose the Mobile Hub NoSQL Database mobile backend feature to:

- Add easy to develop database capabilities with scalable performance and cost

Create a free Mobile Hub project and add the NoSQL DB feature in minutes.

Feature Details

The following image shows the typical connection between a mobile app and Amazon DynamoDB using the NoSQL pattern.



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*.

NoSQL Database At a Glance

AWS services and resources configured	• **Amazon DynamoDB Tables** (see Working with Tables in DynamoDB) Concepts | Console | Pricing Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67). For more information, see Viewing AWS Resources Provisioned for this Feature (p. 58).
Configuration options	This feature enables the following mobile app backend capabilities: Configuring Your Tables (p. 56) - Using custom schema, based on a sample schema provided, or by using a wizard that guides you through choices while creating a table. Data Permissions (p. 57) - Access to your app's data can be: • **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) For more information, see Configuring the NoSQL Database Feature (p. 55). For more information, see Configuring the NoSQL Database Feature (p. 55).
Quickstart app demos	This feature adds the following to a quickstart app generated by Mobile Hub: • 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 |AMH| 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 |AMH| 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. 57).
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.

Topics

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

Best practice for data security is to allow the minimum access to your tables that will support your app design. Mobile Hub provides two methods to protect your data: user authentication using the User Sign-in (p. 67) feature; and NoSQL Database data table user permissions.

Note: When NoSQL Database is enabled your app communicates directly with the DynamoDB service. If you do not make the User Sign-in (p. 67) feature **Required** then, where not blocked by table user permissions, unauthenticated users will have access to read and/or write data.

Grant Permissions Only to Authenticated Users

Unless users who have not signed-in need to read or write data in a table in your app, scope down access by requiring users to sign in (authenticate) before they are allowed to use app features that perform database operations. The AWS Mobile Hub User Sign-in (p. 67) feature offers a range of methods for authenticating users that includes: federating with a sign-in provider like Facebook, Google, Active Directory, or your existing custom service. In a few clicks, you can also create your own sign-in provider backed by AWS services.

To add User Sign-in to your app, use the **Configure more features button** on a feature configuration page, or the **Configure** icon on the left. Then choose and enable **User Sign-in**.

Grant Permissions to Table Data Items Per User

When you create a new table in NoSQL Database, you choose between **Public**, **Private**, or **Protected** options, to determine which app users can read or write the table's data. Mobile Hub attaches a fine-grained access control policy to the table, that can restrict the operations available to a user based on whether or not they are the creator of data being accessed.

Public

- Public permissions allow all users to read or update all items (data rows) in the table.

Protected

- Protected permissions allow all users to read all items in the table, but only the owner of an item can update or delete that item.

Private

- Private permissions allow only the owner of an item to read or write to it.

Note
Users own a data item if their Amazon Cognito identity ID matches the value of the item's primary key.
If you choose **Protected** or **Private** permissions for a table, then the partition key of the table must be `userId`, and be of type `string`. Secondary indexes for protected or private tables follow the same pattern as primary indexes.
When a user creates an item in a protected or private table, AWS populates the value of the item's primary key with that user's Amazon Cognito identity ID.
Enforcement happens when a data operation is attempted on a protected or private item. IAM will check if the item's `userId` matches the current user's Amazon Cognito identity ID, and allow or prevent the operation based on the policy attached to the table.
When you choose **Public**, permissions for a table there is no ownership enforcement. There are no restrictions on name or data type of the primary key and secondary index primary keys of a public table.

Managing Permissions to Restricted Items 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 `orders` 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`.

Retrieving Data

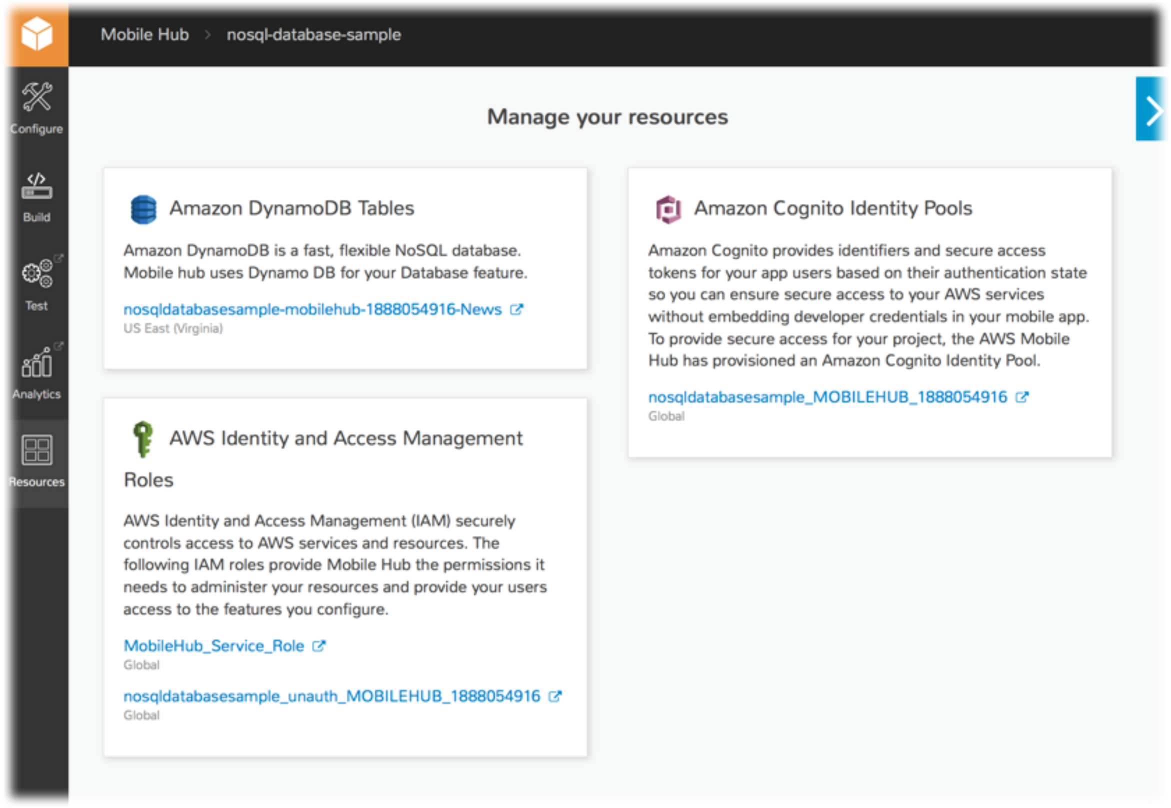
The operations you can use to retrieve data from your NoSQL database include the following:

- `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 Filter`s, which performs a :code:`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:



Manage your resources

Amazon DynamoDB Tables

Amazon DynamoDB is a fast, flexible NoSQL database. Mobile hub uses Dynamo DB for your Database feature.

nosqldatabasesample-mobilehub-1888054916-News
US East (Virginia)

AWS Identity and Access Management Roles

AWS Identity and Access Management (IAM) securely controls access to AWS services and resources. The following IAM roles provide Mobile Hub the permissions it needs to administer your resources and provide your users access to the features you configure.

MobileHub_Service_Role
Global

nosqldatabasesample_unauth_MOBILEHUB_1888054916
Global

Amazon Cognito Identity Pools

Amazon Cognito provides identifiers and secure access tokens for your app users based on their authentication state so you can ensure secure access to your AWS services without embedding developer credentials in your mobile app. To provide secure access for your project, the AWS Mobile Hub has provisioned an Amazon Cognito Identity Pool.

nosqldatabasesample_MOBILEHUB_1888054916
Global

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.

Messaging and Analytics

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Choose the AWS Mobile Hub Messaging and Analytics feature to:

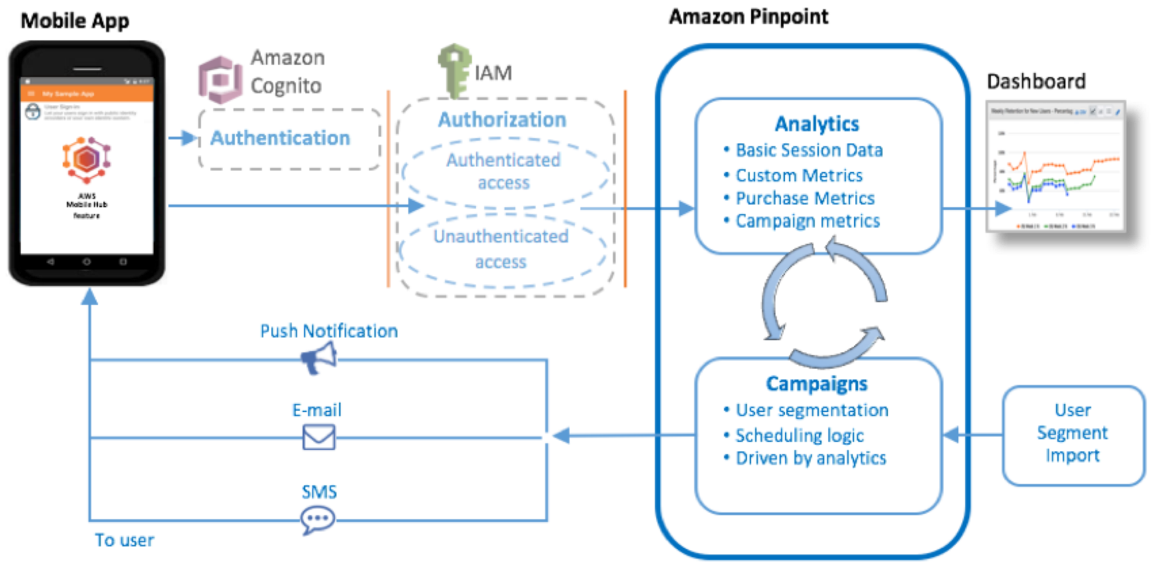
- Gather data to understand your app users' behavior
- Use that information to add campaigns to engage with your users through push notification, e-mail, and SMS

Create a free Mobile Hub project and add the Messaging and Analytics feature.

Feature Details

AWS Mobile Hub Messaging and Analytics (formerly User Engagement) helps you understand how your users use your app. It enables you to engage them through push notification, e-mail, or SMS. You can tie your analytics to your messaging so that what you communicate flows from users' behavior.

The following image shows Messaging and Analytics using Amazon Pinpoint to collect usage data from a mobile app. Amazon Pinpoint then sends messaging to selected app users based on the campaign logic designed for the app.



You can configure messaging and analytics functions separately, or use the two together to carry out campaigns to interact with your users based on the their app usage. You can configure which users receive a campaign's messaging, as well as the conditions and scheduling logic for sending messages. You can configure notifications to communicate text or cause a programatic action, such as opening an application or passing custom JSON to your client.

When you choose **Analytics**, Amazon Pinpoint performs capture, visualization, and analysis of app usage and campaign data:

- By default, Amazon Pinpoint gathers app usage session data.
- If you configure a campaign, metrics about your campaign are included.
- If you add custom analytics to your app, you can configure Amazon Pinpoint to visualize those metrics and use the data as a factor in your campaign behavior. To learn more about integrating custom analytics, see Integrating Amazon Pinpoint With Your App in the *Amazon Pinpoint User Guide*.
- Amazon Pinpoint enables you to construct funnel analytics, which visualize how many users complete each of a series of step you intend them to take in your app.
- To perform more complex analytics tasks, such as merging data from more than one app or making flexible queries, you can configure Amazon Pinpoint to stream your data to Kinesis. To learn more about using Amazon Pinpoint and Kinesis together, see Streaming Amazon Pinpoint Events to Amazon Kinesis.

When you choose **Messaging** you can configure your project to enable Amazon Pinpoint to send:

- Send Push Notifications to your Android users, through Firebase/Google Cloud Messaging, or iOS, through APNs
- E-mails to your app users using the sender ID and domain of your choice

- SMS messages

Once you have enabled Messaging and Analytics options in your Mobile Hub project, use the Amazon Pinpoint console to view visualizations of your analytics or configure your user segments and campaigns. You can also import user segment data into Amazon Pinpoint to use campaigns for any group of users.

Messaging and Analytics At a Glance

AWS services and resources configured	• **Amazon Pinpoint** (see Amazon Pinpoint Developer Guide) Concepts | Console Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67).
Configuration options	This feature enables the following mobile backend capabilities: • Gather and visualize analytics of your app users' behavior. • Integrate Amazon Pinpoint user engagement campaigns into your mobile app. • Communicate to app users using push notifications through APNs, GCM, and FCM. • via **Firebase or Google Cloud Messaging (FCM/GCM)** (see Setting Up Android Push Notifications) • via **Apple Push Notification service (APNs)** (see Setting Up iOS Push Notifications) For more information, see Configuring Push Notification. • Communicate to app users through e-mail. • Communicate to app users through SMS.
Quickstart app demos	This feature adds **User Engagement** functionality to a quickstart app generated by Mobile Hub: • Demonstrate enabling the app user to receive campaign notifications. The app user can cause events that generate session, custom, campaign and purchase data. Analytics for these events is available in the Amazon Pinpoint console in close to real time. • Demonstrate providing the app user with a view of an Amazon Pinpoint data visualization, on their mobile phone.

Hosting and Streaming

Choose AWS Mobile Hub Hosting and Streaming to:

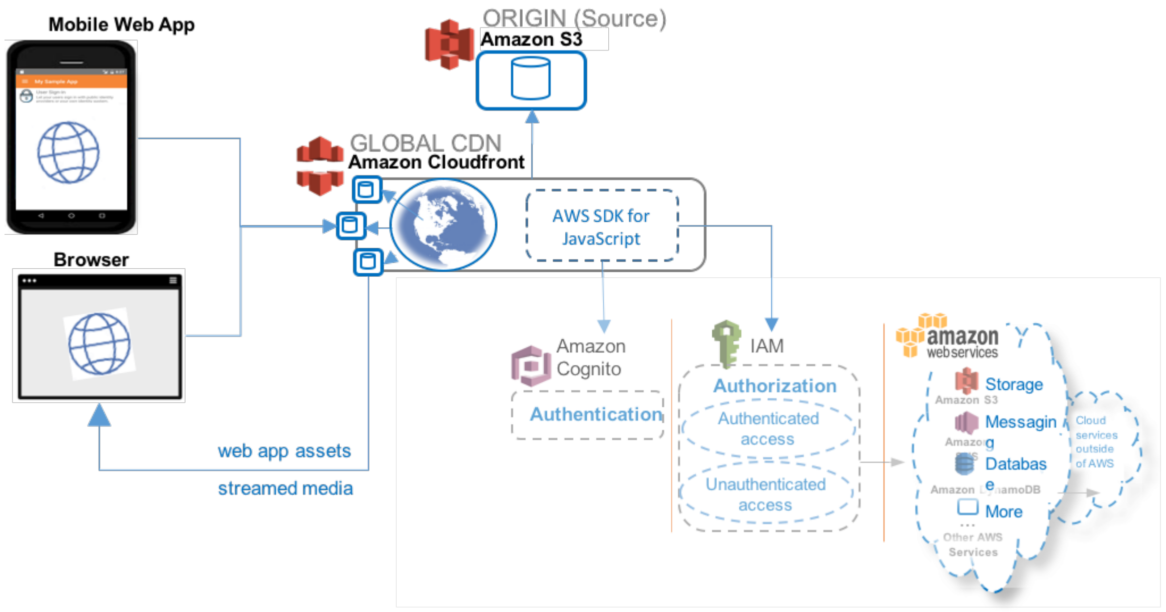
- Host content for you mobile web, native mobile or hybrid app
- Distribute your content through a global Content Delivery Network (CDN)
- Stream your media

Create a free Mobile Hub project with Hosting and Streaming. Get a custom sample app and SDK.

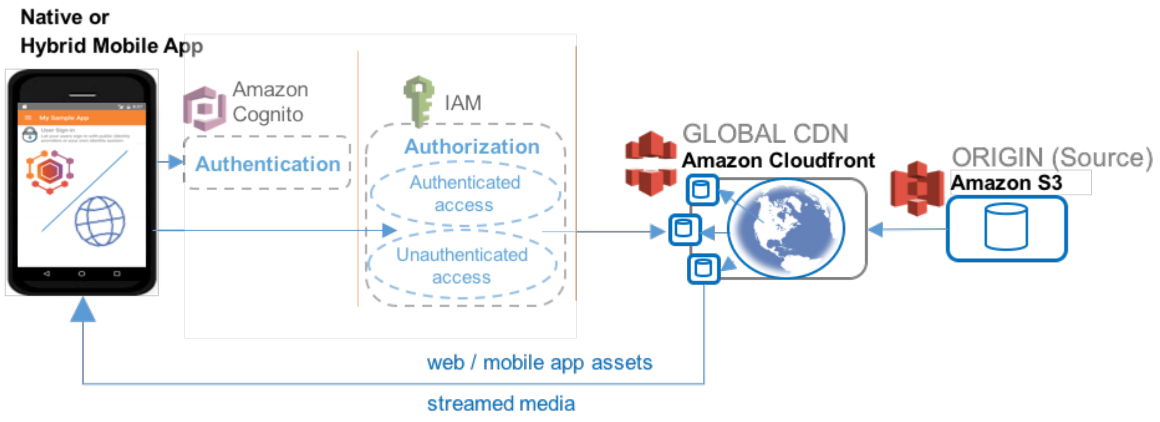
Feature Details

The Hosting and Streaming feature delivers content through a global network of endpoints using Amazon Simple Storage Service (Amazon S3) and Amazon CloudFront.

The following image shows how website assets and streaming media are distributed to a mobile web app or browser. The web app is shown requesting AWS credentials and accessing AWS services through the AWS SDK for JavaScript.



The following image shows a native or hybrid mobile app requesting AWS credentials to access content from a CDN edge location.



The Hosting and Streaming feature enables you to host website and app assets in the cloud, such as HTML, JavaScript, image, or media files. Mobile Hub creates a content source storage location (origin) using an Amazon S3 bucket. The bucket is made accessible to the internet through the Amazon S3 ability to statically host web content with no web server.

Low latency access to your content is provided to users in all regions by caching your source content on a global network of edge locations. This Content Distribution network (CDN) is provided through an Amazon CloudFront distribution which also supports media file streaming (see Amazon CloudFront streaming).

Hosting and Streaming At a Glance

AWS services and resources configured	• **Amazon CloudFront - Content Delivery Network** (see Amazon CloudFront) Concepts | Console | Pricing • **Amazon S3 Bucket** (see *Amazon S3 Getting Started Guide* <https://aws.amazon.com/cloudfront/pricing/>`__) Concepts | Console | Pricing Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67). For more information, see Viewing AWS Resources Provisioned for this Feature (p. 65).
Configuration options	This feature enables the following mobile backend capabilities: • **Web app content hosting** (Internet access for your content, no web servers required) • **AWS SDK for JavaScript** (Call AWS services via standard scripting) • **Global CDN** (Global content distribution and media streaming) CloudFront offers several options for regional scope and cost of your distribution. For more information, see Configuring the Hosting and Streaming Feature (p. 64).
Web app demo	Sample • The AWS SDK for Javascript and a custom-generated configuration file are provisioned to your bucket. For more information, see Web App Support (p. 64).
Quickstart native app demos	This feature adds the following to a quickstart app generated by Mobile Hub: • View file list in AWS storage, download and view files, and manage their local cache.

Web App Support

When you enable Hosting and Streaming, Mobile Hub provisions a local copy of the AWS SDK for JavaScript in the root of your bucket.

Mobile Hub also generates the project configuration files `aws-config.js` and `aws-exports.js`, which contain endpoint constants for each AWS services Mobile Hub configured for your project. `aws-exports.js` is provided for integration with ES6 compatible scripting languages like Node.js. Use these values to make SDK calls to your services from your hosted web app.

> **Note**
> Best security practice is to reduce access to an app's resources as much as possible. These configuration files are publically accessible and contain identifiers for all of your app's AWS resources. If it suits your design, we recommend you protect your resources by allowing only authenticated users to access them. You can do this in this project by enabling the Mobile Hub User Sign-in (p. 67) with the **Require sign-in** option.

You can also copy the appropriate configuration file into your hybrid native/web mobile app to enable calling your AWS services from your app using JavaScript.

Configuring the Hosting and Streaming Feature

Topics

Browsing Your Content

With Hosting and Streaming enabled, you have several options:

- **Launch from Amazon S3**: This option browses to the un-cached index.html in the root of your source bucket.
- **Launch from Amazon CloudFront**: This option browses to the index.html that is cached on the CDN edge servers.

  > **Note**
  > Provisioning the edge locations for the distribution can take up to an hour. This link will not resolve until the distribution finishes propagating in the network.
- **Manage files**: This option opens the Amazon S3 console to review and manage the contents of your source bucket. You can also find your bucket in the Amazon S3 console by opening your project in Mobile Hub and then choosing the **Resources** icon on the left. The name of the bucket configured for Hosting and Streaming contains the string `hosting`.

Managing Your App Assets

You can choose from a variety of ways to manage your web app assets through use of the Amazon S3 console, the AWS Command Line Interface (CLI) or one of the many third party applications available.

Using the Amazon S3 Console

To use the Amazon S3 console to review, upload, move or delete your files stored in your bucket, navigate to the Amazon S3 console and choose the bucket whose name contains your project name. Your web app content will reside in the root folder.

Using AWS CLI

AWS CLI allows you to review, upload, move or delete your files stored in your bucket using the command line.

To install and configure the AWS CLI client, see Getting Set Up with the AWS Command Line Interface.

As an example, the sync command enables transfer of files to and from your local folder (`source`) and your bucket (`destination`).

```
$ aws s3 sync {source destination} [--options]
```

The following command syncs all files from your current local folder to the folder in your web app's bucket defined by `path`.

```
$ aws s3 sync . s3://AWSDOC-EXAMPLE-BUCKET/path
```

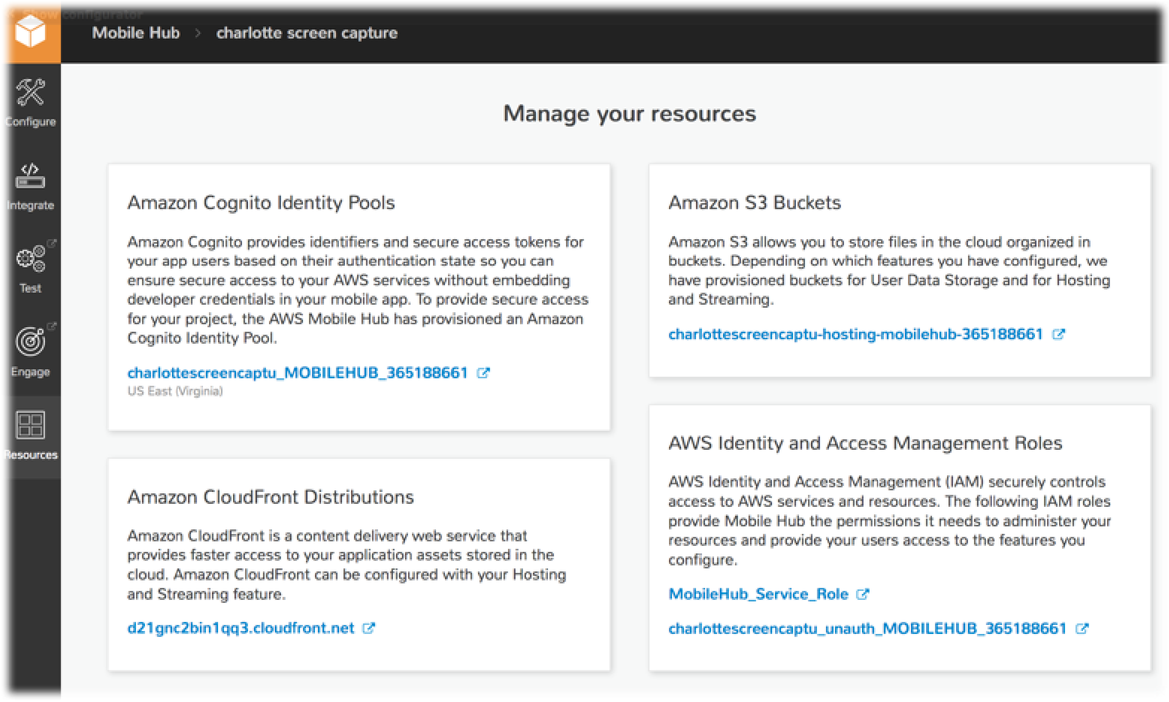
To learn more about using AWS CLI to manage Amazon S3, see Using Amazon S3 with the AWS Command Line Interface

Using a Custom Domain for Your Web App

To configure your Hosting and Streaming CDN as the destination of your custom domain, see Routing Traffic to an Amazon CloudFront Web Distribution by Using Your Domain Name.

Viewing AWS Resources Provisioned for this Feature

The following image shows the Mobile Hub**Resources** pane displaying elements typically provisioned for the Hosting and Streaming feature.



Manage your resources

Amazon Cognito Identity Pools

Amazon Cognito provides identifiers and secure access tokens for your app users based on their authentication state so you can ensure secure access to your AWS services without embedding developer credentials in your mobile app. To provide secure access for your project, the AWS Mobile Hub has provisioned an Amazon Cognito Identity Pool.

charlottescreencaptu_MOBILEHUB_365188661
US East (Virginia)

Amazon S3 Buckets

Amazon S3 allows you to store files in the cloud organized in buckets. Depending on which features you have configured, we have provisioned buckets for User Data Storage and for Hosting and Streaming.

charlottescreencaptu-hosting-mobilehub-365188661

Amazon CloudFront Distributions

Amazon CloudFront is a content delivery web service that provides faster access to your application assets stored in the cloud. Amazon CloudFront can be configured with your Hosting and Streaming feature.

d21gnc2bin1qq3.cloudfront.net

AWS Identity and Access Management Roles

AWS Identity and Access Management (IAM) securely controls access to AWS services and resources. The following IAM roles provide Mobile Hub the permissions it needs to administer your resources and provide your users access to the features you configure.

MobileHub_Service_Role

charlottescreencaptu_unauth_MOBILEHUB_365188661

Quickstart App Details

In the Mobile Hub quickstart app, the Hosting and Streaming 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.

Conversational Bots

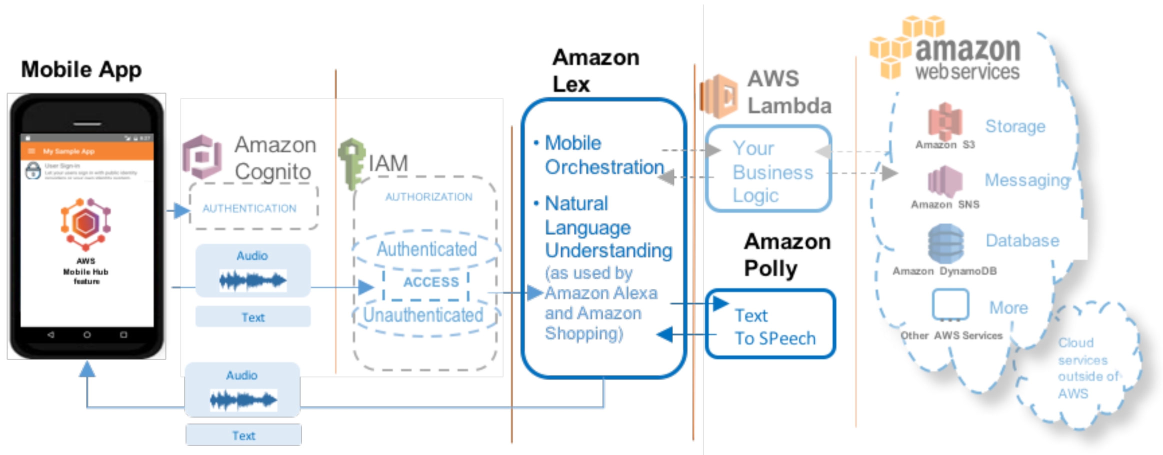
Choose the AWS Mobile Hub conversational bots mobile backend service feature to:

- Add voice and text natural language understanding interface to your app
- Use natural language voice and text to interact with your business logic in the cloud

Create a free Mobile Hub project and add the Conversational Bots feature.

Feature Details

The following image shows Conversational Bots using Amazon Lex to add natural language to a mobile app interface and as an integration point for other services.



AWS Mobile Hub conversational bots bring your mobile app the same natural language understanding and business logic integration that power the Amazon Alexa and Amazon Shopping voice and text conversation experiences.

Mobile Hub conversational bots use Amazon Lex, an AWS service for building voice and text conversational interfaces into applications. Amazon Lex has built-in integration with Lambda.

With conversational bots and Amazon Lex, no deep learning expertise is necessary. Specify the basic conversation flow in the Amazon Lex console to create a bot. The service manages the dialogue and dynamically adjusts the responses in the conversation. Using Mobile Hub conversation bots, you can provision and test bots based on demonstration templates or bots you have created in the Amazon Lex console. Mobile Hub provides integration instructions and customized components for reusing the sample app code we generate in your own app.

Conversational Bots At a Glance

AWS services and resources configured	• Amazon Lex (see Amazon Lex Developer Guide) Concepts | Console Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67).
Configuration options	This feature enables the following mobile backend capabilities:

	• Create and configure conversational bots in the Amazon Lex service based on provided demonstration templates or by using the Amazon Lex console to add your customized text and/or speech interactions to your app. • Integrate your app by downloading and reusing the code of the quickstart app, a package of native iOS and Android SDKs, plus helper code and on line guidance, all of which are dynamically generated to match your Mobile Hub project.
Quickstart app demos	This feature adds the following functionality to a quickstart app generated by Mobile Hub: • Enables user to interact with a conversational bot that interacts with Amazon Lex.

User Sign-in

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Choose the AWS Mobile Hub User Sign-in mobile backend feature to:

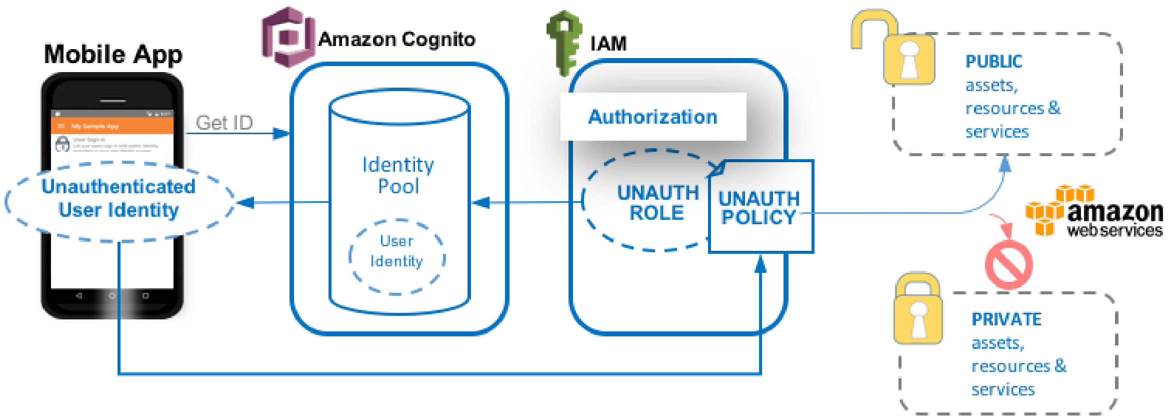
- Add AWS user authentication and secure identity access management to your mobile app.

  Note: Secure unauthenticated access to AWS resources is available to all Mobile Hub projects with or without the User Sign-in feature.
- Enable your users to sign-in to access AWS resources with existing credentials from identity providers like Facebook, Google, Microsoft Active Directory Federation Services or your own custom user directory.

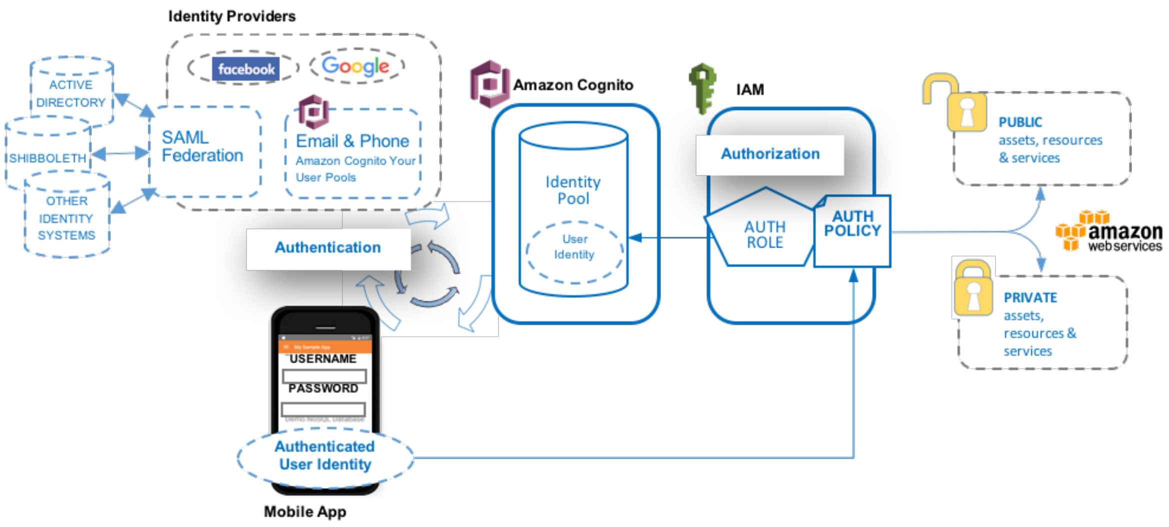
Create a free Mobile Hub project and add the User Sign-in feature.

Feature Details

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 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. Mobile Hub helps you integrate identity providers into your mobile app so that users can sign in using their existing credentials from Facebook, Google, and your own identity system. You can also create and configure your own email- and password-based user directory using Amazon Cognito Your User Pools.

User Sign-in Feature At a Glance

AWS services and resources configured	• **Amazon Cognito**

	Concepts | Console | Pricing • **Amazon Cognito Identity Pool** (see Using Federated Identities) • **Amazon Cognito Your User Pools** (see Creating and Managing User Pools) • **Amazon Cognito SAML Federation** (see Overview of Configuring SAML 2.0-Based Federation) • **IAM role and security policies** (see Control Access to Mobile Hub Projects (p. 77)) Concepts | Console | Pricing For more information, see Viewing AWS Resources Provisioned for this Feature (p. 71).
Configuration options	This feature enables the following mobile backend capabilities: **Sign-in Providers** (users gain greater access when they sign in)

- via **Google** authentication (see Using Amazon Cognito Hosted UI)
  - via **Facebook** authentication (see mobile-auth-setup)
  - via **Email and Password** authentication (see User Sign-in Providers (p. 69))
  - via **SAML Federation** authentication (see User Sign-in Providers (p. 69))

  Required Sign-in (authenticated access)

  Optional Sign-in (users gain greater access when they sign in) For more information, see Configuring User Sign-in (p. 69)

  - • **Quickstart demo features**
    - This feature adds the following to a quickstart app generated by Mobile Hub:
      - 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 Email and Password (your own user pool).
      - 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.

User Sign-in Providers

Facebook

- To enable Facebook user authentication, register your application with Facebook.

If you already have a registered Facebook app, copy the App ID from the Facebook Developers App Dashboard. Paste the ID into the Facebook App ID field and choose Save Changes.

If you do not have a Facebook App ID yet, you'll need to create one before you can integrate Facebook in your mobile app. The Facebook Developers portal takes you through the process of setting up your Facebook application.

For full instructions on integrating your application with Facebook, see Setting Up Facebook Authentication.

Google

- To authenticate your users through Google, fully integrate your sample app with Google+ Sign-in.

  If you already have a registered Google Console project with the Google+ API, a web application OAuthClient and a client ID for the platform of your choice set up, then copy and paste the Google Web App Client ID and client ID(s) from the Google Developers Console into those fields and choose **Save Changes**.

  Regardless of the platform you choose (Android or iOS), you'll need to at least create the following.
  - A Google Console project with the Google+ API enabled (used for Google Sign-in)
  - A web application OAuth client ID
  - An iOS and/or Android client ID, depending on which platform you are supporting

  For full instructions on integrating your application with Google+, see :Setting Up Google Authentication.

Email and Password

- Choose Email and Password sign-in when you want to create your own AWS-managed user directory and sign-in process for your app's users. Configure the characteristics of their sign-in experience by:
  - Selecting user login options (*email, username, and/or phone number*)
  - Enabling multi-factor authentication (*none, required, optional*) which adds delivery of an entry code via text message to a user's phone, and a prompt to enter that code along with the other factor to sign-in
  - Selecting password character requirements (*minimum length, upper/lower cases, numbers or special characters allowed*).

SAML Federation

- SAML Federation enables users with credentials in your existing identity store to sign in to your mobile app using their familiar username and password. A user signs into to your identity provider (IdP) which is configured to return a validating SAML assertion. Your app then uses Amazon Cognito Federated Identities to exchange the SAML assertion for typical temporary, limited privilege credentials to access your AWS backend services.

  SAML 2.0 (Security Assertion Markup Language 2.0) is an open standard used by many IdPs, including Microsoft Active Directory Federation Service and Shibboleth. Your IdP must be SAML 2.0 compatible to use this Mobile Hub option. To establish federation between AWS and your IdP the two systems must exchange SAML federation metadata. AWS federation metadata can be found at https://signin.aws.amazon.com/static/saml-metadata.xml. This xml file demonstrates the form that your IdP's metadata should take. For more information on SAML federation metadata for your IdP, see Integrating Third-Party SAML Solution Providers with AWS.

To implement this exchange:

1. View your IdP's documentation to understand how to use the AWS federation metadata file to register AWS as a service provider.
2. Ensure that your Mobile Hub project is configured to use Email & Password sign-in to create an Amazon Cognito User Pool.
3. Configure your IdP as an Identity Provider for your user pool using the steps on Creating SAML Identity Providers for Your User Pool.

To learn more about how AWS supports SAML federation, see Overview of Configuring SAML 2.0-Based Federation.

User Sign-in Requirement

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 an authenticated user or an unauthenticated guest user 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 an authenticated user so that your app can access your AWS services securely.

Note
If user sign-in is not required, unauthenticated users can access to data in your database tables and files in your storage buckets, unless those resources are explicitly restricted through another mechanism.

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

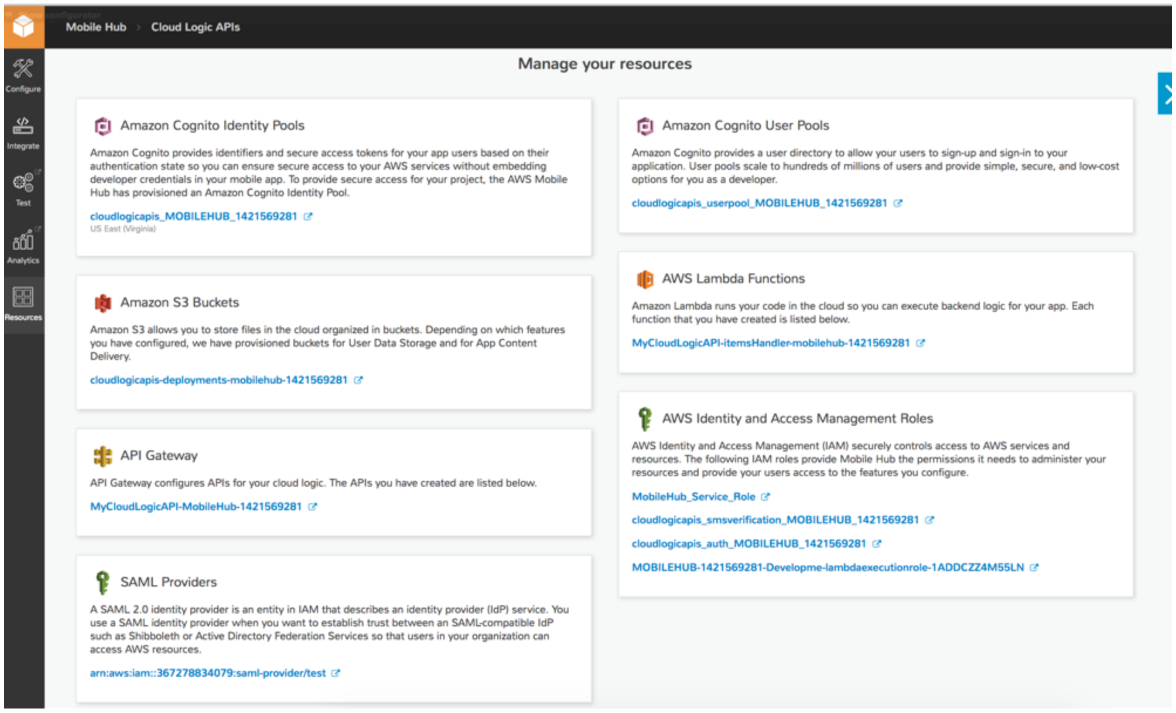
When your mobile app is saved, Mobile Hub creates an Amazon Cognito identity pool and a new IAM role. These are used to generate temporary AWS credentials for the quickstart app users to access your AWS resources. The AWS IAM role security policies are updated based on the sign-in features enabled.

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 Control Access to Mobile Hub Projects (p. 77).

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.



Manage your resources

Amazon Cognito Identity Pools

Amazon Cognito provides identifiers and secure access tokens for your app users based on their authentication state so you can ensure secure access to your AWS services without embedding developer credentials in your mobile app. To provide secure access for your project, the AWS Mobile Hub has provisioned an Amazon Cognito Identity Pool.

cloudlogicapis_MOBILEHUB_1421569281
US East (Virginia)

Amazon S3 Buckets

Amazon S3 allows you to store files in the cloud organized in buckets. Depending on which features you have configured, we have provisioned buckets for User Data Storage and for App Content Delivery.

cloudlogicapis-deployments-mobilehub-1421569281

API Gateway

API Gateway configures APIs for your cloud logic. The APIs you have created are listed below.

MyCloudLogicAPI-MobileHub-1421569281

SAML Providers

A SAML 2.0 identity provider is an entity in IAM that describes an identity provider (IdP) service. You use a SAML identity provider when you want to establish trust between an SAML-compatible IdP such as Shibboleth or Active Directory Federation Services so that users in your organization can access AWS resources.

arn:aws:iam::367278834079:saml-provider/test

Amazon Cognito User Pools

Amazon Cognito provides a user directory to allow your users to sign-up and sign-in to your application. User pools scale to hundreds of millions of users and provide simple, secure, and low-cost options for you as a developer.

cloudlogicapis_userpool_MOBILEHUB_1421569281

AWS Lambda Functions

Amazon Lambda runs your code in the cloud so you can execute backend logic for your app. Each function that you have created is listed below.

MyCloudLogicAPI-itemsHandler-mobilehub-1421569281

AWS Identity and Access Management Roles

AWS Identity and Access Management (IAM) securely controls access to AWS services and resources. The following IAM roles provide Mobile Hub the permissions it needs to administer your resources and provide your users access to the features you configure.

MobileHub_Service_Role

cloudlogicapis_smsverification_MOBILEHUB_1421569281

cloudlogicapis_auth_MOBILEHUB_1421569281

MOBILEHUB-1421569281-Developme-lambdaexecutionrole-1ADDCZZ4M55LN

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, SAML Federation or Email and Password.

When you add User Sign-in to your project with the **Optional Sign-in** option, choosing the app's quickstart sign-in demo returns and displays the user's Amazon Cognito Identity Pool ID. 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(s) configured in the project. Signing in to the demo authenticates the user in the selected identity provider service and returns and displays the Amazon Cognito Identity Pool ID identifier of the user.

User File Storage

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Choose AWS Mobile Hub User File Storage to:

- Add cloud storage of user files, profile data, and app state to your mobile app
- Use fine-grained control of access to files and data, implementing four common patterns of permissions policy

Looking for Amazon Cognito Sync?	Amazon Cognito Sync has been deprecated. For real time data sync between devices, with built-in offline capabilities, see AWS AppSync.

Create a free Mobile Hub project and add the User File Storage feature.

Feature Details

The Mobile Hub User File Storage feature, creates and configures four folders for each user, inside an Amazon Simple Storage Service (Amazon S3) bucket belonging to the app.

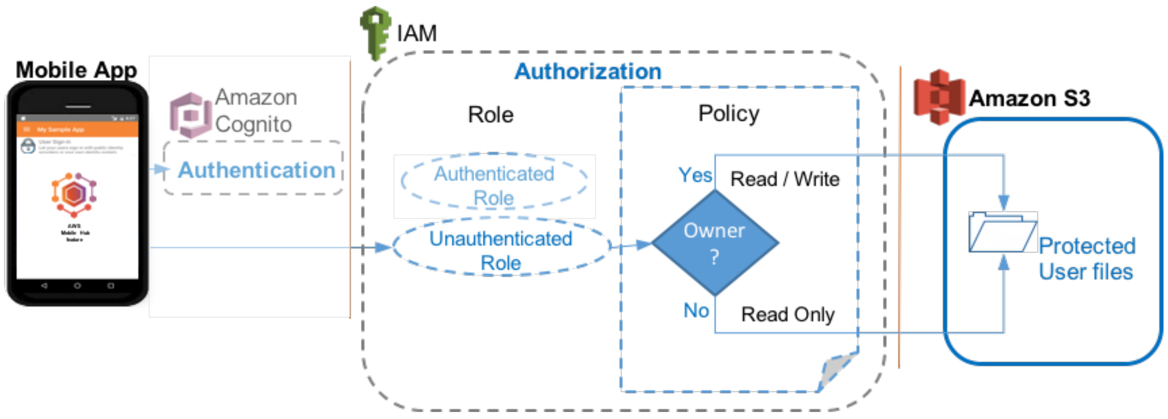
Best practice for app security is to allow the minimum access to your buckets that will support your app design. Each of the four folders provisioned has a policy illustrating different permissions choices attached. In addition, Mobile Hub provides the option to restrict access to your app to only authenticated users using the User Sign-in (p. 67) feature.

Note: If you do not make the User Sign-in (p. 67) feature **Required** then, where not blocked by a folder or bucket access policy, unauthenticated users will have access to read and/or write data.

The following table shows the details of permissions policies that are provisioned for each folder type.

Folder name	Owner permissions	Everyone else permissions
Public	Read/Write	Read/Write
Private	Read/Write	None
Protected	Read/Write	Read Only
Uploads	Write Only	Write Only

The following image shows IAM policy being applied to control file access in a Protected folder. The policy grants read/write permissions for the user who created the folder, and read only permissions for everyone else.



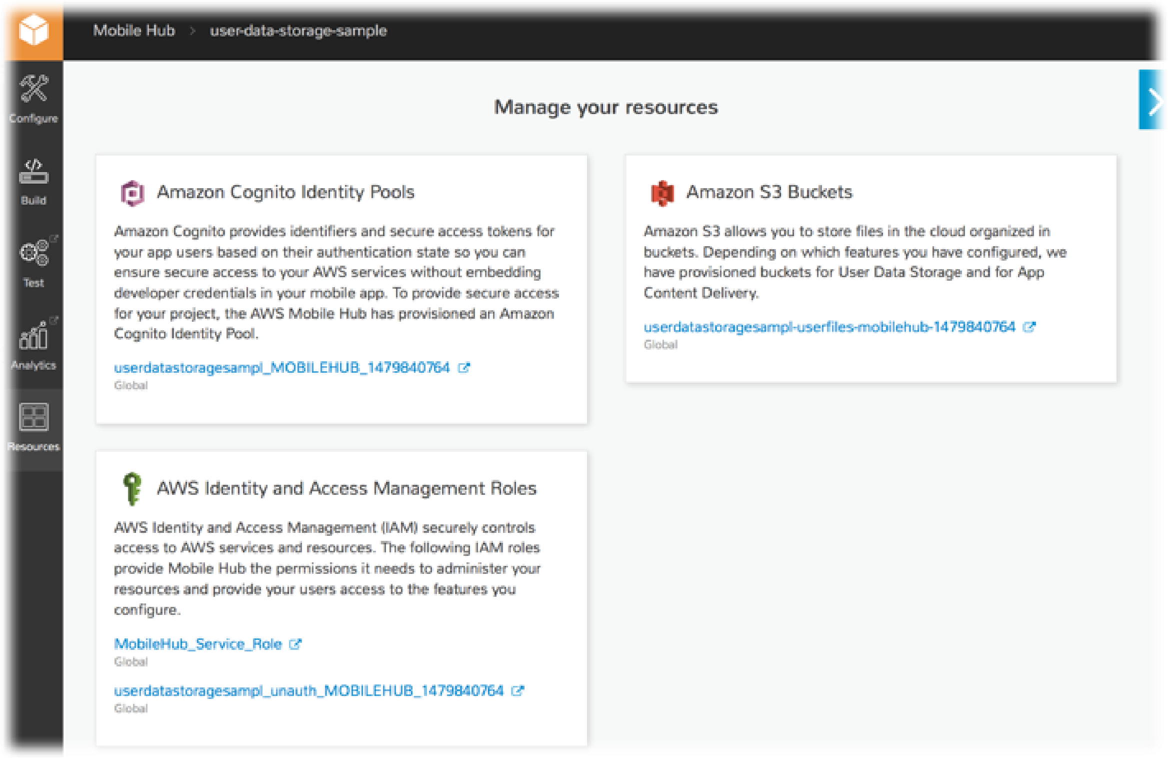
The User File Storage feature enables you to store user files such as photos or documents in the cloud, and it also allows you to save user profile data in key/value pairs, such as app settings or game state. When you select this feature, an Amazon S3 bucket is created as the place your app will store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

User File Storage At a Glance

AWS services and resources configured	• **Amazon S3 bucket** (see Amazon S3 Getting Started Guide) Concepts | Console | Pricing Mobile Hub-enabled features use Amazon Cognito for authentication and IAM for authorization. For more information, see User Sign-in (p. 67). For more information, see Viewing AWS Resources Provisioned for this Feature (p. 74).
Configuration options	This feature enables the following configuration options mobile backend capabilities: • Store user files and app data using Amazon S3. When you enable User File Storage four folders are provisioned, each with a distinct access policy configuration: • `private` - Each mobile app user can create, read, update, and delete their own files in this folder. No other app users can access this folder. • `protected` - Each mobile app user can create, read, update, and delete their own files in this folder. In addition, any app user can read any other app user's files in this folder. • `public` ? Any app user can create, read, update, and delete files in this folder.
Quickstart demo features	This feature adds the following to a quickstart app generated by Mobile Hub: • File explorer for the app's S3 bucket allows the user to: • Upload and view files in any **Public** folder. • View and download files in a **Private** folder that the user created. • View and download files in a **Protected** folder anyone created and upload files to that folder if the user created it. • Upload files to any **Uploads** folder. User setting of choice of color theme can be persisted to and retrieves from 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 File Storage feature.



Manage your resources

Amazon Cognito Identity Pools

Amazon Cognito provides identifiers and secure access tokens for your app users based on their authentication state so you can ensure secure access to your AWS services without embedding developer credentials in your mobile app. To provide secure access for your project, the AWS Mobile Hub has provisioned an Amazon Cognito Identity Pool.

userdatastoragesampl_MOBILEHUB_1479840764
Global

Amazon S3 Buckets

Amazon S3 allows you to store files in the cloud organized in buckets. Depending on which features you have configured, we have provisioned buckets for User Data Storage and for App Content Delivery.

userdatastoragesampl-userfiles-mobilehub-1479840764
Global

AWS Identity and Access Management Roles

AWS Identity and Access Management (IAM) securely controls access to AWS services and resources. The following IAM roles provide Mobile Hub the permissions it needs to administer your resources and provide your users access to the features you configure.

MobileHub_Service_Role
Global

userdatastoragesampl_unauth_MOBILEHUB_1479840764
Global

AWS Identity and Access Management Usage in AWS Mobile Hub

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Note
In depth understanding of AWS IAM, authentication, and access controls are not required to configure a backend for your mobile app using Mobile Hub.

- Control Access to Mobile Hub Projects (p. 77) - learn how to grant permissions for configuration of your Mobile Hub project.
- Mobile Hub Project Permissions Model (p. 76) - learn more about permissions you give Mobile Hub to configure AWS resources and services, see .
- IAM Authentication and Access Control for Mobile Hub (p. 80) - learn details of IAM and AWS authentication and access controls.

Mobile Hub Project Permissions Model

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Important

To modify Mobile Hub projects in an account, a user must be granted administrative permissions (p. 78) by an account Administrator. Read this section for more information.

If you are a user who needs additional permissions for a project, contact an administrator for the AWS account. For help with any issues related to the new permissions model, contact aws-mobilehub-customer@amazon.com.

Topics

Mobile Hub Permissions Model

Currently, Mobile Hub's permissions model uses the user's permissions directly when they perform operations in the Mobile Hub console or command line interface. This model provides account administrators fine-grained access control over what operations their users can perform in the account, regardless of whether they are using Mobile Hub or they're using the console or command line interface to interact with services directly.

In order to modify projects, users are required to have permissions to use Mobile Hub (granted by AWSMobileHubFullAccess IAM policy), and they must have permission to perform whatever actions Mobile Hub takes on their behalf. In almost every case, this means an account administrator must grant the user the AdministratorAccess policy (p. 78) in order to provide access to the AWS resources Mobile Hub modifies. This is because, as project settings are modified, Mobile Hub will modify the IAM roles and policies used to enable the features affected by those settings. Changing IAM roles and policies allows the user to control access to resources in the account, and so they must have administrative permissions.

When an administrator does not want to grant administrative permissions for the full account, they can choose instead to provide each user or team their own sub-account using AWS Organizations (p. 79). Within their sub-account, a user will have full administrative permissions. Sub-account owners are only limited in what they can do by the policy put in place by their administrator, and billing rolls up to the parent account.

What if I Currently Use MobileHub_Service_Role to Grant Mobile Hub Permissions?

Previously, Mobile Hub assumed a service role called `MobileHub_Service_Role` in order to modify service configurations on your behalf using the following managed policy:

https://console.aws.amazon.com/iam/home?#/policies/arn:aws:iam::aws:policy/service-role/AWSMobileHub_ServiceUseOnly

In that older model, all that was required to modify Mobile Hub projects was permissions to call Mobile Hub APIs, through the console or command line. An administrator could delegate those permissions by attaching the `AWSMobileHub_FullAccess` policy to an AWS IAM user, group, or role.

If the account of your Mobile Hub projects relies on the old model, the impact on those who are not granted AdministratorAccess permissions will be as follows.

- IAM users, groups and roles that have the `AWSMobileHub_FullAccess` policy will no longer have sufficient permissions to perform any mutating operations in Mobile Hub, either via the console or `awsmobile` command line interface (CLI).
- In order for IAM users, groups, or roles to be able to perform mutating operations using Mobile Hub, they must have the appropriate permissions. The two choices for an administrator to grant users permission (p. 77) to invoke all available operations in Mobile Hub are: attach the `AdministratorAccess` policy to the user, or a role they are attached to, or a group they are a member of; or alternatively, to use AWS Organizations to manage permissions.

Why Did the Permissions Model Change?

AWS Mobile Hub creates IAM roles and assigns them permissions in order to enable use of AWS resources in mobile apps. Such operations are considered administrative because they include enabling permission to perform operations on resources in the account. Previously, Mobile Hub's service role provided users who have been granted `AWSMobileHub_FullAccess` permissions with a path to escalate their own privileges to act on resources, potentially in ways their administrator did not intend to permit. Removing the service role, removes the path to escalate privileges and puts control of user permissions directly in the hands of the administrator for a Mobile Hub project.

Control Access to Mobile Hub Projects

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Overview

This section describes two different ways to control access to your Mobile Hub projects:

- Grant a user administrative account permissions (p. 78)

  For individual developers, or groups whose requirements for segmenting access to their Mobile Hub projects are simple, permission can be granted by attaching the managed AdministratorAccess (p. 80) or AWSMobileHub_ReadOnly (p. 80) AWS managed policies to a user, a role they are attached to, or a group they belong to.

Or:

- Use AWS Organizations to manage permissions (p. 79)

  For organizations that require fine-grained access control and cost tracking for their Mobile Hub projects, AWS account administrators can provide sub-accounts and determine the policies that apply to their users.

To understand how Mobile Hub uses IAM policies attached to a user to create and modify services on a users behalf, see Mobile Hub Project Permissions Model (p. 76).

To understand AWS Identity and Access Management (IAM) in more detail, see IAM Authentication and Access Control for Mobile Hub (p. 80) and IAM Authentication and Access Control for Mobile Hub (p. 80).

Best Practice: Create IAM Users to Access AWS

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 in your AWS account, or use an existing IAM user, and then access Mobile Hub with that user. For more information, see AWS Security Credentials in the AWS General Reference.

You can create an IAM user for yourself or a delegate user using the IAM console. First, create an IAM administrator group, then create and assign a new IAM user to that group.

Note
Before any IAM user within an account can create a mobile Hub project, a user with administrative privileges for the account must navigate to the Mobile Hub console and create an initial project. This step provides confirmation that Mobile Hub can manage AWS services on your behalf.
To learn more about assigning access rights to IAM users or groups, see IAM Authentication and Access Control for Mobile Hub (p. 80).

Grant Users Permissions to Mobile Hub Projects

Topics

Use the following steps to create a group and/or users, and grant users access to your Mobile Hub projects.

To grant permissions to a role, see Adding Permissions in the *AWS IAM User Guide*.

Create a New IAM User in Your Account and Grant Mobile Hub Permissions

1. Open the IAM console. On the left, choose **Users**, and then choose **Add User**.
2. Type a user name, select the check boxes for **Programmatic access** and **AWS Management Console access**.
3. Choose the password policy you prefer. Then choose **Next: Permissions**.
4. In the **Add user to group** tab, select the **Administrators** or **Read_Only** group for the user, and choose **Next, Review**.

   In the process, you will see options to customize the user's password, alert them about their new account via email, and to download their access key ID, key value and password.
5. Choose **Create user**.
6. To apply policy:
   - If you have created a group to manage project permissions, choose **Add user to group**, select the group, choose **Next: Review**, then choose **Create User**.

Or:

- If you are managing project permissions per user, choose **Attach existing policies directly**, select the policy you want to attach, **AdministratorAccess** or **AWSMobileHub_ReadOnly**, and then choose **Create user**.

Create an IAM Group

1. Sign in to the AWS Management Console and open the IAM console at http://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` or `Read_Only`, and then choose **Next Step**.
4. In the list of policies, select the check box next to the **AdministratorAccess** policy to grant full permissions to the group, or **AWSMobileHub_ReadOnly** to grant only read access. 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**.

Grant Mobile Hub Permissions to an Existing Account User

1. On the left, choose **Policies**.
2. Choose the link for the managed policy, **AdministratorAccess** or **AWSMobileHub_ReadOnly** you want to attach.
3. Choose **Attached Entities**.
4. Choose **Attach**.
5. Choose the users, roles, or groups you want to grant permissions.
6. Choose **Attach Policy**.

Use AWS Organizations to Manage Permissions

AWS Organizations can be used to manage permissions for groups that need to segment access to their Mobile Hub projects. For example, an administrator could provide an account for each developer on a team. Within their own account, each user would have the permissions granted by the administrator. The steps to achieve this would be:

1. If you do not have an AWS account, sign up for the AWS Free Tier.
2. Create an organization in the AWS Organizations console.
3. Create or add existing accounts for each user in the organization.
4. Invite the users.
5. Create a organizational unit for the developers.
6. Enable and attach a policy for members of the unit.

   The policy you attach will apply within the scope of the AWS account of a user. You may want to limit access to services and capabilities not required for Mobile Hub use. For instance, the following policy, grants all permissions defined in the `FullAWSAccess` managed policy, but excludes access to the Amazon EC2 service.

```
"Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Effect": “Deny”,
            "Action": “ec2:*”,
            "Resource": "*"
        }
]
```

For step by step instructions, see the tutorial at Creating and Managing an AWS Organization.

AWS Managed (Predefined) Policies for Mobile Hub Project Access

The AWS Identity and Access Management service controls user permissions for 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.

- **AdministratorAccess**

  This policy provides unlimited access to AWS services in the account. That includes 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 of the permissions that are allowed under the `AWSMobileHub_ReadOnly` managed policy. After you sign in to the Mobile Hub console and create a project, you can use the following link to view this policy and the IAM identities that are attached to it.

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

- **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). This policy does not allow 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. After you sign in to the Mobile Hub console and create a project, you can use the following link to view this policy and the IAM identities that are attached to it.

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

  If your IAM user, role, or group has read-only permissions for use in an AWS Mobile Hub project, then the project information you see in the console will not reflect any changes made outside of Mobile Hub. For example, if you remove a Cloud Logic API in API Gateway, it may still be present in the Cloud Logic Functions list of your Mobile Hub project, until a user with **mobilehub:SynchronizeProject** permissions visits the console. Users who are granted console access through the **AdminstratorAccess** policy have those permissions. If you need additional permissions in Mobile Hub, please contact your administrator and request the **AdminstratorAccess** policy.

IAM Authentication and Access Control for Mobile Hub

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Note
In depth understanding of AWS IAM, authentication, and access controls are not required to configure a backend for your mobile app using Mobile Hub.

Mobile Hub uses AWS credentials and permissions policies to allow a user to view and/or create and configure the back-end features the user selects for their mobile app.

The following sections provide details on how IAM works, how you can use IAM to securely control access to your projects, and what IAM roles and policies Mobile Hub configures on your behalf.

Topics

Authentication

AWS resources and services can only be viewed, created or modified with the correct authentication using AWS credentials (which must also be granted access permissions (p. 82) to those resources and services). You can access AWS as any of the following types of identities:

- **AWS account root user**

  When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your root credentials and they provide complete access to all of your AWS resources.

  Important
  For security reasons, we recommend that you use the root credentials only to create an administrator user, which is an IAM user with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the *IAM User Guide*.
- **IAM user**

  An IAM user is simply an identity within your AWS account that has specific custom permissions (for example, read-only permissions to access your Mobile Hub project). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself.
- **IAM role**

  An IAM role is another IAM identity you can create in your account that has specific permissions. It is similar to an IAM user, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access**

Instead of creating an IAM user, you can use preexisting user identities from your enterprise user directory or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.

- **Cross-account access**

  You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the *IAM User Guide*.
- **AWS service access**

  You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2**

  Instead of storing access keys within the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot access or modify a Mobile Hub project. The same is true for Mobile Hub when it creates and configures services and resources you have configured for your project.

The following sections describe how to manage permissions and understand those that are being managed on your behalf by Mobile Hub.

- Control Access to Mobile Hub Projects (p. 77)

Overview of Access Permissions Management for Mobile Hub Projects

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Note
In depth understanding of AWS IAM, authentication, and access controls are not required to configure a backend for your mobile app using Mobile Hub.

Every AWS resource is owned by an AWS account. Permissions to view, create, and/or access the resources (p. 77) are governed by policies.

An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

> **Note**
> An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.
> When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

Understanding Resource Ownership for AWS Mobile Hub

The primary resource of a Mobile Hub project is the project itself. In first use of the Mobile Hub console, you allow Mobile Hub to manage permissions and access the project resource for you. A resource owner is the AWS account that created a resource. That is, the resource owner is the AWS account of the principal entity (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an AWS Mobile Hub project, your AWS account is the owner of the resources associated with that project.
- If you create an IAM user in your AWS account and grant permissions to create Mobile Hub projects to that user, the user can also create projects. However, your AWS account, to which the user belongs, owns the resources associated with the project.
- If you create an IAM role in your AWS account with permissions to create AWS Mobile Hub projects, anyone who can assume the role can create, edit, or delete projects. Your AWS account, to which the role belongs, owns the resources associated with that project.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
> This section discusses using IAM in the context of AWS Mobile Hub. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS Identity and Access Management Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM polices) and policies attached to a resource are referred to as resource-based policies.

Topics

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account**? An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to view or modify an AWS Mobile Hub project.

- **Attach a permissions policy to a role (grant cross-account permissions)** ? You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, when you first enter Mobile Hub and agree, as account principal, to grant it permissions to provision and configure your project, you are granting the AWS managed `MobileHub_Service_Role` role cross-account permissions. An AWS managed policy, `AWSMobileHub_ServiceUseOnly`, is attached to that role in the context of your Mobile Hub project. The role has a trust policy that allows Mobile Hub to act as account principal with the ability to grant permissions for services and resources used by your project.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

As an example of using an identity-based policy, the following policy grants permissions to a user to create an Amazon S3 bucket. A user with these permissions can create a storage location using the Amazon S3 service.

```
{
       "Version":"2012-10-17",
       "Statement":[
          {
             "Effect":"Allow",
             "Action":"s3:CreateBucket*",
             "Resource":"*"
          }
       ]
    }
```

For more information about using identity-based policies with Mobile Hub , see :ref: reference-mobile-hub-project-permissions-model`.

For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an Amazon S3 bucket to manage access permissions to that bucket.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

Each service that is configured by Mobile Hub defines a set of API operations. To grant Mobile Hub permissions for these API operations, a set of actions is specified in an AWS managed policy. Performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** - In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies.
- **Action** - You use action keywords to identify resource operations that you want to allow or deny. For example, the `s3:Createbucket` permission allows Mobile Hub to perform the Amazon `S3CreateBucket` operation.
- **Effect** - You specify the effect when the user requests the specific action?this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** - In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only).

Mobile Hub Project Service Region Hosting

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

The configuration settings of your Mobile Hub project are stored in the AWS US East (Virginia) region.

The AWS services you configure are hosted in the region you select for your project, if they are available in that region. If services are not available in that region, then Mobile hub will host the services in another region.

For more details about regional endpoints, see AWS Regions and Endpoints.

To understand where services for your project will be hosted, find the region for your project in the following tables.

Select your project's region:

US East (Virginia)

If you selected US East (Virginia) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	US East (Virginia)
Amazon Cognito (User Sign-in / User File Storage)	US East (Virginia)
Amazon DynamoDB (NoSQL Database)	US East (Virginia)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)

Hosting for these services:	Is located in:
Amazon S3 (User File Storage / Messaging and Hosting)	US East (Virginia)
AWS Lambda (Cloud Logic)	US East (Virginia)

US East (Ohio)

If you selected US East (Ohio) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	US East (Ohio)
Amazon Cognito (User Sign-in / User File Storage)	US East (Ohio)
Amazon DynamoDB (NoSQL Database)	US East (Ohio)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	US East (Ohio)
AWS Lambda (Cloud Logic)	US East (Ohio)

US West (California)

If you selected US West (California) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	US West (California)
Amazon Cognito (User Sign-in / User File Storage)	US West (Oregon)
Amazon DynamoDB (NoSQL Database)	US West (California)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	US West (California)
AWS Lambda (Cloud Logic)	US West (California)

US West (Oregon)

If you selected US West (Oregon) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	US West (Oregon)
Amazon Cognito (User Sign-in / User File Storage)	US West (Oregon)
Amazon DynamoDB (NoSQL Database)	US West (Oregon)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	US West (Oregon)
AWS Lambda (Cloud Logic)	US West (Oregon)

EU West (Ireland)

If you selected EU West (Ireland) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	EU West (Ireland)
Amazon Cognito (User Sign-in / User File Storage)	EU West (Ireland)
Amazon DynamoDB (NoSQL Database)	EU West (Ireland)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	EU West (Ireland)
AWS Lambda (Cloud Logic)	EU West (Ireland)

EU West (London)

If you selected EU West (London) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	EU West (London)
Amazon Cognito (User Sign-in / User File Storage)	EU West (London)
Amazon DynamoDB (NoSQL Database)	EU West (London)
Amazon Lex (Conversational Bots)	US East (Virginia)

Hosting for these services:	Is located in:
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	EU West (London)
AWS Lambda (Cloud Logic)	EU West (London)

EU (Frankfurt)

If you selected West EU (Frankfurt) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	EU (Frankfurt)
Amazon Cognito (User Sign-in / User File Storage)	EU (Frankfurt)
Amazon DynamoDB (NoSQL Database)	EU (Frankfurt)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	EU (Frankfurt)
AWS Lambda (Cloud Logic)	EU (Frankfurt)

Asia Pacific (Tokyo)

If you selected Asia Pacific (Tokyo) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	Asia Pacific (Tokyo)
Amazon Cognito (User Sign-in / User File Storage)	Asia Pacific (Tokyo)
Amazon DynamoDB (NoSQL Database)	Asia Pacific (Tokyo)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	Asia Pacific (Tokyo)
AWS Lambda (Cloud Logic)	Asia Pacific (Tokyo)

Asia Pacific (Seoul)

If you selected Asia Pacific (Seoul) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	Asia Pacific (Seoul)
Amazon Cognito (User Sign-in / User File Storage)	Asia Pacific (Seoul)
Amazon DynamoDB (NoSQL Database)	Asia Pacific (Seoul)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	Asia Pacific (Seoul)
AWS Lambda (Cloud Logic)	Asia Pacific (Seoul)

Asia Pacific (Mumbai)

If you selected Asia Pacific (Mumbai) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	Asia Pacific (Mumbai)
Amazon Cognito (User Sign-in / User File Storage)	Asia Pacific (Mumbai)
Amazon DynamoDB (NoSQL Database)	Asia Pacific (Mumbai)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	Asia Pacific (Mumbai)
AWS Lambda (Cloud Logic)	Asia Pacific (Mumbai)

Asia Pacific (Singapore)

If you selected Asia Pacific (Singapore) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	Asia Pacific (Singapore)
Amazon Cognito (User Sign-in / User File Storage)	Asia Pacific (Singapore)
Amazon DynamoDB (NoSQL Database)	Asia Pacific (Singapore)
Amazon Lex (Conversational Bots)	US East (Virginia)

Hosting for these services:	Is located in:
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	Asia Pacific (Singapore)
AWS Lambda (Cloud Logic)	Asia Pacific (Singapore)

Asia Pacific (Sydney)

If you selected Asia Pacific (Sydney) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	Asia Pacific (Sydney)
Amazon Cognito (User Sign-in / User File Storage)	Asia Pacific (Sydney)
Amazon DynamoDB (NoSQL Database)	Asia Pacific (Sydney)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	Asia Pacific (Sydney)
AWS Lambda (Cloud Logic)	Asia Pacific (Sydney)

South America (São Paulo)

If you selected South America (São Paulo) as the preferred region for your project:

Hosting for these services:	Is located in:
Amazon API Gateway (Cloud Logic)	South America (São Paulo)
Amazon Cognito (User Sign-in / User File Storage)	US East (Virginia)
Amazon DynamoDB (NoSQL Database)	South America (São Paulo)
Amazon Lex (Conversational Bots)	US East (Virginia)
Amazon Pinpoint (Messaging and Analytics)	US East (Virginia)
Amazon S3 (User File Storage / Messaging and Hosting)	US East (Virginia)
AWS Lambda (Cloud Logic)	South America (São Paulo)

Mobile Hub Project Troubleshooting

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

The following sections describe issues you might encounter when setting up, importing or exporting Mobile Hub projects, and their remedies.

Topics

Cannot Import an API

Error Message

- `Project owner does not own existing API : arn:aws:execute-api:us-east-1:012345678901:abcdefghij.`

  (where the API identifier arn:aws:execute-api:us-east-1:012345678901:abcdefghij is specific to the project being imported)

Description

- This message means that the API with the ID shown cannot be imported because it does not exist in the current AWS account. This occurs when the APIs in the original project were created outside of the Mobile Hub Cloud Logic feature and then imported.

Remedy

- **To remedy this condition, take the following steps.**
  1. Modify the YAML of the project definition you are importing by removing the sections under the `features:components` node that begin with the name of an API that was imported into the original project's Cloud Logic feature.
  2. Save and import the project definition.
  3. Enable the Mobile Hub Cloud Logic feature in your imported project and recreate the API and its handler.

Cannot Import a NoSQL Table

Error Message

- There is already an existing DynamoDB table called ‘someprojectname-mobilehub-012345678-TableName’ in your account. Please choose a different name or remove the existing table and retry your request.

  (where the table name someprojectname-mobilehub-012345678-TableName is specific to the project being imported)

Description

- This message occurs when you import a project containing the NoSQL Database Feature. It indicates that the Amazon DynamoDB table in the project configuration already exists. This can occur when a YAML tablename value was edited in the project definition file and there is more than one attempt to import it into the same account.

Remedy

- **To remedy this condition, take the following steps**
  1. Modify any tablename values to remove the conflict.
  2. Save and import the project definition.
  3. Adjust the code of the imported app where it references the old tablename value.

Cannot Import Multiple NoSQL Tables

Error Message

- Project file(s) cannot be decoded. They may contain data that was encrypted by a different account. Failed to decode push feature. Failed to decode credential attribute.

Description

- This message occurs when you import Push Notifications messaging service credentials or Amazon SNS topic identifiers for features that are not associated with your AWS account.

Remedy

- **To remedy this condition, take the following steps**
  1. Modify the YAML of the project definition you are importing by removing table definition sections.
  2. Save and import the project definition.
  3. Use the table definitions you removed to manually create those tables using the Mobile Hub NoSQL Database feature.

Cannot Import Push Credentials

Error Message

- Project file(s) cannot be decoded. They may contain data that was encrypted by a different account. Failed to decode push feature. Failed to decode credential attribute.

Description

- This message occurs when you import Push Notifications messaging service credentials or Amazon SNS topic identifiers for features that are not associated with your AWS account.

Remedy

- **To remedy this condition, take the following steps**
  1. Modify the YAML of the project definition you are importing by removing the push: node.
  2. Save and import the project definition.
  3. Enable the Mobile Hub Push Notifications or User Engagement feature using your own messaging service credentials and topics.

Build Artifacts Can't be Found

Error Message

- Unable to find build artifact uploads/exported-project-definition.zip in Amazon S3 bucket archive-deployments-mobilehub-0123456789 for project-name.

  where exported-project-definition, the numerical portion of the Amazon S3 bucket identifier, and the project-name are specific to the project being imported)

Description

- This message occurs when a project import fails because Mobile Hub can't find the file of a Cloud Logic API handler function (Lambda) that is specified in the .yml project definition file.

Remedy

- **To remedy this condition, take the following steps**

  The remedy for this condition is to make the location of the Lambda file(s) match the path specified in the project definition YAML.

  The error occurs if, for any reason, the path described in the codeFilename: key in the YAML does not match the actual location of the Lambda function file relative to the root of the `...-deployments-...` Amazon S3 bucket that Mobile Hub deploys when Cloud Logic is enabled.

Unable to Configure S3 Bucket During

Error Message

- It looks like there was a problem creating or configuring your S3 bucket.

Description

- Mobile Hub was unable to create a S3 bucket for your project's deployment artifacts during Mobile Hub project import.

Remedy

- **To remedy this condition, try the following steps**

Check that you are not at maximum bucket capacity using the Amazon S3 console.

Administrator Required Error During Setup

Error Message

- It looks like you do not have permission for this operation.

Description

- The user does not have permission to create the required Mobile Hub Service Role during configuration of a Mobile Hub project.

Remedy

- **To remedy this condition, try the following steps**

  Contact an administrator for your AWS account and ask them to create the service role at the following location: https://console.aws.amazon.com/mobilehub/home#/activaterole/.

Account Setup Incomplete

Error Message

- It looks like your AWS account is not fully set up.

Description

- This error can occur for a range of reasons during Mobile Hub project configuration.

Remedy

- **To remedy this condition, try the following steps**
  - Sign out of the AWS console and close down all browser windows. Then try to log in to the AWS Management Console and attempt the operation that initially caused the error.

File Too Large to Import

Error Message

- The project file is too large. The max file size is 10 MB.

Description

- This message occurs when you attempt to import a project definition file that is larger than 10MB.

Remedy

- Reduce the size of the project export file. Project exporters may want to deliver large file payloads outside of their project definition files, along with providing instructions for importers about how to use AWS consoles to incorporate those accompanying files.

Exporting and Importing AWS Mobile Hub Projects

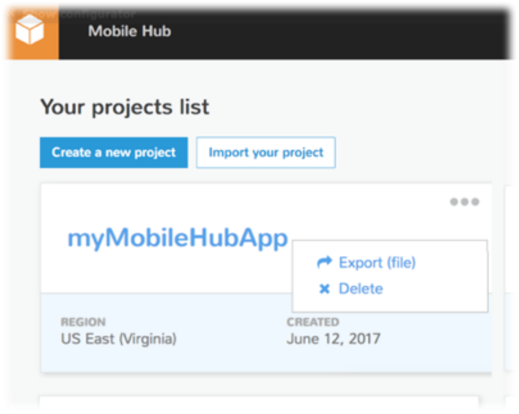
Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

Overview

Mobile Hub provides the ability to export and import YAML files that describe the configuration of your Mobile Hub project. Anyone with an AWS account can import an exported project configuration file to deploy a new project, with new AWS resources that match the configuration being imported.

This feature enables you to replicate the AWS service configuration of an exported project. While the data in a project's tables is not exported, files in storage or hosting buckets and API handler function code can be manually added to your exported project definition. To learn more, see import-export-manual.



To export a project configuration file

1. Navigate to your project list in the Mobile Hub console.
2. Hover over the ellipses (three dots) in the upper right of the project card.
3. Choose **Export (file)** in the upper right of the card for the project you want to export.
4. Save your project export file.

To learn more about the content of an exported project configuration file, see Structure of a Project Export .yml File (p. 99).

To import a project

1. Navigate to your project list in the Mobile Hub console.
2. Choose **Import your project** in the upper left of the page.
3. Browse or drag a project definition file into the **Import project configuration** dialog.
4. Choose **Import project**.

Sharing Your Project Configuration with a Deploy to AWS Mobile Hub Link

In any public GitHub repo, you can provide a link that instantly kicks off creation of a new Mobile Hub project by importing the exported project configuration file define in the link's querystring. The form of the link should be:

```
https://console.aws.amazon.com/mobilehub/home?#/?config=YOUR-MOBILE-HUB-
PROJECT-CONFIGURATION-LOCATION
```

For example, the following HTML creates a link that provides instant configuration of an app's AWS backend services, based on Mobile Hub features defined in `react-sample.zip`. To see this code in action, see `README.md` for the AWS Mobile React Sample.

```
<p align="center">
   <a target="_blank" href="https://console.aws.amazon.com/mobilehub/home?#/?
config=https://github.com/awslabs/aws-mobile-react-sample/blob/master/backend/
import_mobilehub/react-sample.zip">
   <span>
       <img height="100%" src="https://s3.amazonaws.com/deploytomh/button-deploy-aws-
mh.png"/>
   </span>
   </a>
</p>
```

The querystring portion of the link can point to the location of a Mobile Hub project configuration `mobile-hub-project.yml` file or a project export `.zip` file containing a `mobile-hub-project.yml` file.

> **Important**
> If you are using a `.zip` file it must conform to the structure and content required by a Mobile Hub project configuration import. For details, see Structure of a Project Export .zip File (p. 98).

Limitations of Importing Projects

Topics

Maximum Project Definition File Size is 10MB

Import of Mobile Hub project `.zip` or `.yml` files larger than 10MB is not supported.

Project Components that Require Manual Export

To enable import of the following project configuration items, you must manually modify your project's exported `.zip` file:

- Data User Storage Contents

  To import files stored in a User File Storage Amazon S3 bucket in your original project, see Importing User File Storage Contents (p. 101).
- Hosting and Streaming Contents

To import files hosted in a Hosting and Streaming bucket in your original project, see Importing Hosting and Streaming Contents (p. 102).

- SAML Federation

  To import User Sign-in SAML federation configuration from your original project, see Importing SAML Federated User Sign-in (p. 102).
- Cloud Logic API Handlers

  To import Cloud Logic API handler code and configuration from your original project, see Importing API Handlers for Cloud Logic APIs (p. 103).

  Note
  Calling Cloud Logic APIs from a browser requires that Cross-Origin Resource Sharing (CORS) is configured for each API path. To enable CORS configuration when your project is imported, see Importing Cross-Origin Resource Sharing (CORS) Configuration (p. 105).

Cross Account Credentials

Some features require credentials and assets that are associated with the AWS account where they are configured. Mobile Hub projects that contain such features can only be imported into the account that exported them. Features with this restriction include Cloud Logic APIs that were created outside of the Mobile Hub project being exported, messaging provider credentials for Push Notification, and Amazon SNS topics.

Mobile Hub Feature	Can be exported from one AWS account and imported into another?
User Sign-in	Yes
NoSQL Database	Yes
Cloud Logic	Using APIs created within your Mobile Hub project: Yes Using APIs imported into your project: No (for remedy, see Cannot Import an API (p. 91))
User File Storage	Yes
App Content Delivery	Yes
Connectors	Yes
Push Notifications	No (for remedy, see Cannot Import Push Credentials (p. 92))
Messaging and Analytics (Push Notification)	No (for remedy, see Cannot Import Push Credentials (p. 92))

Project Components that Are Not Exported

The following items are not supported by the Mobile Hub import/export feature:

- Custom policy

When you enable a Mobile Hub feature, a set of AWS services is deployed. Mobile Hub attaches default access roles and policies to these objects. When a project is imported, the default roles and policies are applied.

In your original project, you can modify or add to these defaults; for example, to set access to a data table to read only. When you export your project configuration, any such customizations are not included in the project export. To enable your custom policy in an imported project, the importer must manually configure those policies in the imported project. In addition to your project export file, we recommend you provide both your policy JSON and step by step instructions for importers. These instructions should describe how to use AWS consoles or the AWS CLI to implement your customizations.

- Legacy Cloud Logic

  Import and export are not supported for projects using the legacy Cloud Logic feature. A project of this kind calls Lambda functions directly. The current version of Cloud Logic makes RESTful calls to Amazon API Gateway APIs linked to Lambda function handlers.

Mobile Hub Project Export Format

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

AWS Mobile Hub provides the ability to export a YAML file containing the configuration of your project. The YAML file itself can be imported or it can be included in a `.zip` file with other project components that get deployed during project import. This section describes the anatomy of the YAML and a typical Mobile Hub project export `.zip` file. For more information about the Mobile Hub Import/Export feature, see Exporting and Importing AWS Mobile Hub Projects (p. 95).

Topics

Structure of a Project Export .zip File

When you choose **Export (file)**, Mobile Hub generates a `.zip` file named for your project.

Default file structure

Mobile Hub also generates a `mobile-hub-project.yml` project configuration file in the `.zip` root. A valid `mobile-hub-project.yml` file in this location is required for Mobile Hub project import to succeed.

Example file structure

File structure of the `.zip` file an exported project, configured to include deployment of both SAML federation and Cloud Logic API handlers, is as follows:

- *`/your-project-name`*`.zip`
  - `mobile-hub-project.yml`

- `saml.xml`
- `lambda API handler functions`
- `user data stored files`
- `hosted files`

Files in a project export `.zip` file can be arranged in folders. The relative paths within the archive must be reflected in the project definition YAML key values that refer to their paths.

> **Note**
> The presence of any files or folders in the project configuration `.zip` file, other than those described in the preceding section, may be ignored or cause issues upon import.

Structure of a Project Export .yml File

In the abstract, the basic structure of a Mobile Hub project export `.yml` file is as follows:

```
features:
    FEATURE-TYPE: !com.amazonaws.mobilehub.v0.:FEATURE-TYPE
          components:
            FEATURE-NAME: !com.amazonaws.mobilehub.v0.FEATURE-TYPE
                attributes:
                    ATTRIBUTE-NAME: !com.amazonaws.mobilehub.v0.ATTRIBUTE-VALUE
                OTHER-FEATURE-PROPERTY-TYPES: OTHER-FEATURE-PROPERTY-VALUES
            . . .
```

The following YAML is a sample of the `mobile-hub-project.yml` exported from a project with many Mobile Hub features enabled. The project definition has also been manually updated to enable the import and upload of components of the original project. These components include files stored in the original project's User File Storage bucket, files hosted in its Hosting and Streaming bucket, and API handler code in its Lambda functions.

```
--- !com.amazonaws.mobilehub.v0.Project
features:
  cloudlogic: !com.amazonaws.mobilehub.v0.CloudLogic
    components:
      api-name: !com.amazonaws.mobilehub.v0.API
        attributes:
          name: api-name
          requires-signin: true
          sdk-generation-stage-name: Development
        paths:
          /items: !com.amazonaws.mobilehub.v0.Function
            codeFilename: uploads/lambda-archive.zip
            description: "Handler for calls to resource path : /items"
            enableCORS: true
            handler: lambda.handler
            memorySize: "128"
            name: handler-name
            runtime: nodejs6.10
            timeout: "3"
          "/items/{proxy+}": !com.amazonaws.mobilehub.v0.Function
            codeFilename: uploads/lambda-archive.zip
            description: "Handler for calls to resource path : /items/{proxy+}"
            enableCORS: true
            handler: lambda.handler
            memorySize: "128"
            name: handler-name
            runtime: nodejs6.10
            timeout: "3"
  content-delivery: !com.amazonaws.mobilehub.v0.ContentDelivery
    attributes:
      enabled: true
      visibility: public-global
    components:
      release: !com.amazonaws.mobilehub.v0.Bucket {}
  database: !com.amazonaws.mobilehub.v0.Database
    components:
      database-nosql: !com.amazonaws.mobilehub.v0.NoSQLDatabase
        tables:
          - !com.amazonaws.mobilehub.v0.NoSQLTable
            attributes:
            id: S
            hashKeyName: id
            hashKeyType: S
            rangeKeyName: ""
            rangeKeyType: ""
            tableName: ___DYNAMIC_PREFIX___-bbq-order
            tablePrivacy: public
          - !com.amazonaws.mobilehub.v0.NoSQLTable
            attributes:
            id: S
            hashKeyName: id
            hashKeyType: S
            rangeKeyName: ""
            rangeKeyType: ""
            tableName: ___DYNAMIC_PREFIX___-bbq_restaurants
            tablePrivacy: public
          - !com.amazonaws.mobilehub.v0.NoSQLTable
            attributes:
            id: S
            restaurant_id: S
            hashKeyName: restaurant_id
            hashKeyType: S
            rangeKeyName: id
            rangeKeyType: S
            tableName: ___DYNAMIC_PREFIX___-bbq_menu_item
            tablePrivacy: public
  sign-in: !com.amazonaws.mobilehub.v0.SignIn
    attributes:
      enabled: true
      optional-sign-in: false
    components:
      sign-in-user-pools: !com.amazonaws.mobilehub.v0.UserPoolsIdentityProvider
        attributes:
          alias-attributes:
            - email
            - phone_number
          mfa-configuration: ON
          name: userpool
          password-policy: !com.amazonaws.mobilehub.ConvertibleMap
            min-length: "8"
            require-lower-case: true
            require-numbers: true
            require-symbols: true
            require-upper-case: true
  user-files: !com.amazonaws.mobilehub.v0.UserFiles
    attributes:
      enabled: true
  user-profiles: !com.amazonaws.mobilehub.v0.UserSettings
    attributes:
      enabled: truename: myProject
region: us-east-1
uploads:
    - !com.amazonaws.mobilehub.v0.Upload
      fileName: stored-file
      targetS3Bucket: user-file.png
    - !com.amazonaws.mobilehub.v0.Upload
```

```
      fileName: hosted-file
      targetS3Bucket: hosting.html
    - !com.amazonaws.mobilehub.v0.Upload
      fileName: api-handler-file.zip
      targetS3Bucket: deployments
```

Manually Exported Project Components

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

This section describes how to manually add project components to an exported project definition.

Topics

Importing User File Storage Contents

When a project that enables User File Storage is exported, files stored in its Amazon S3 bucket are not included in its exported project definition. You can manually configure the project definition to upload those files to the new bucket of the imported project.

To configure import and upload of project files stored in a User File Storage bucket

1. Uncompress your exported project `.zip` file.
2. Copy and paste each file that you want uploaded during import into the unzipped file folder.
3. Add file paths to your exported project definition:
   a. Open the `mobile-hub-project.yml` file of the export in an editor.
   b. If not already present, create an `uploads:` node at the root level.
   c. For each file to be uploaded, add the following three items under `uploads:`.
      i. The namespace – `!com.amazonaws.mobilehub.v0.Upload`
      ii. The key `fileName:` with the value of the path to the file within the project definition `.zip` file.
      iii. The key `targetS3Bucket:` with the value of `user-files`.

```
--- !com.amazonaws.mobilehub.v0.Project
features:
  sign-in: !com.amazonaws.mobilehub.v0.SignIn {}
  user-files: !com.amazonaws.mobilehub.v0.UserFiles
    attributes:
      enabled: true
  user-profiles: !com.amazonaws.mobilehub.v0.UserSettings
    attributes:
      enabled: true
name: userfiles
region: us-east-1
```

```
uploads:
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {example1.png}
    targetS3Bucket: user-files
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {example2.xml}
    targetS3Bucket: user-files
. . .
```

4. Rezip the files within the uncompressed project definition file (not the folder containing those files, because that causes a path error).

Importing Hosting and Streaming Contents

When a project that enables Hosting and Streaming is exported, files stored in its Amazon S3 bucket are not included in the exported project definition. You can manually configure the project definition to upload those files to the new bucket of the imported project.

To configure import and upload of project files stored in a Hosting and Streaming bucket

1. Uncompress your exported project `.zip` file.
2. Copy and paste each file that you want uploaded during import into the unzipped file folder.
3. Add file paths to your exported project definition:
   a. Open the `mobile-hub-project.yml` file of the export in an editor.
   b. If not already present, create an `uploads:` node at the root level.
   c. For each file to be uploaded, add the following three items under `uploads:`.
      i. The namespace – `!com.amazonaws.mobilehub.v0.Upload`
      ii. The key `fileName:` with the value of the path to the file within the project definition `.zip` file.
      iii. The key `targetS3Bucket:` with the value of `hosting`.

```
--- !com.amazonaws.mobilehub.v0.Project
features:
  content-delivery: !com.amazonaws.mobilehub.v0.ContentDelivery
    attributes:
      enabled: true
      visibility: public-global
    components:
      release: !com.amazonaws.mobilehub.v0.Bucket {}

. . .

uploads:
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {example1.html}
    targetS3Bucket: hosting
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {example2.js}
    targetS3Bucket: hosting
. . .
```

4. Rezip the files within the uncompressed project definition file (not the folder containing those files, because that causes a path error).

Importing SAML Federated User Sign-in

Configuring SAML federation for the Mobile Hub User Sign-in feature requires you to supply the SAML XML configuration (`saml.xml`) of the identity provider you federate. The SAML XML configuration is not included in the `.zip` file exported by Mobile Hub.

To configure an exported project to deploy the original project's SAML federation when it is imported

1. Uncompress your exported project `.zip` file.
2. Copy your identity provider's `saml.xml` file into the root folder of the uncompressed `.zip` file.
3. Rezip the files within the uncompressed project definition file (not the folder containing those files, because that causes a path error).

Importing API Handlers for Cloud Logic APIs

The Mobile Hub Cloud Logic feature pairs a RESTful API surface (API Gateway) with serverless API handler functions (Lambda). While Mobile Hub supports exporting and importing the definitions of API and handler objects that Cloud Logic configures, the API handler function code is not exported.

Mobile Hub enables you to manually configure your project export `.zip` file to deploy your API handler function code as part of the project import when the following conditions are met:

- Your API handler accesses only DynamoDB tables. Import of API handlers that access other AWS services, such as Amazon S3, is not currently supported.
- Your handler code is factored to use Lambda environmental variables to refer to those DynamoDB tables.

  When Mobile Hub imports API handler code, it uses environmental variables to map data operations to the new tables created by the import. You can define the key name of environmental variables in the project's definition YAML to match constant names you define in the project's Lambda API handler function code. The following example shows a Lambda function constant being equated to an environmental variable.

  ```
  const YOUR-FUNCTION-CONSTANT-NAME = process.env.KEY-NAME-DEFINED-IN-YAML;";

  // example
  const MENU_TABLE_NAME = process.env.MENU_TABLE_NAME;
  ```

  The steps that follow these notes describe how to define your environmental variables in project definition YAML.

  Note
  An alternative is to use the `MOBILE_HUB_DYNAMIC_PREFIX` project identifier prefix that Mobile Hub generates. Mobile Hub configures its value to be the unique identifier for the imported project. When you append a valid table name to that prefix in your function code, it composes a valid identifier for the table in the imported project. The following example shows a Lambda function constant being equated to an environmental variable.

  ```
  const YOUR-FUNCTION-CONSTANT-NAME = process.env.MOBILE_HUB_DYNAMIC_PREFIX + "-
  YOUR-TABLE-NAME";

  // example
  const MENU_TABLE_NAME = process.env.MOBILE_HUB_DYNAMIC_PREFIX + "-bbq-menu";
  ```

  This method does not require additional manual configuration of the project definition YAML.

The AWS Mobile React sample app provides an end to end example of using environmental variables to access data tables through an API and its handler. Take the following steps for each API handler whose code you want to import. Examples from the sample app are given in line.

To enable import of |LAM| handler functions for your exported Cloud Logic API

1. Uncompress your exported project `.zip` file.
2. Copy your Lambda function(s) into the uncompressed file.
   a. Go to the Amazon S3 console and search for your Mobile Hub project name.
   b. Choose the bucket with the name containing `-deployments-`, then choose the `uploads` folder.
   c. Copy and save the name(s) of the Lambda function file(s) in the folder for use in following steps.
   d. Copy the Lambda function file(s) in the folder into your unzipped exported project file.
3. Add file paths to your exported project definition.
   a. Open the `mobile-hub-project.yml` file of the export in an editor.
   b. If not already present, create an `uploads:` node at the root level.
   c. For each file to be uploaded, add the following three items under `uploads:`.
      i. The namespace - `!com.amazonaws.mobilehub.v0.Upload`
      ii. The key `fileName:` with the value of the path to the file within the project definition `.zip` file.
      iii. The key `targetS3Bucket:` with the value of `deployments`.
   d. If not already present in each Cloud Logic `. . . paths: items` node, create a `codeFilename:` key with the value of the path of the Lambda function code file for that handler.

      Note
      The path in this case is relative to the root of the `-deployments-`Amazon S3 bucket Mobile Hub provisioned for Cloud Logic. Typically, Mobile Hub places these files in an `/uploads` folder.
      If no `codeFilename` is specified, then Mobile Hub deploys a default handler that echos requests it receives.
   e. Add environmental variables to your exported project definition.

      For each Cloud Logic `. . . paths: items` node that describes a handler that interacts with a DynamoDB table, add an `environment:` node with child members that are composed by concatenating an environmental variable name, with the string `__DYNAMIC_PREFIX__`, and the associated table name. The variable name should map to the associated variable in your Lambda API handler function code.

```
--- !com.amazonaws.mobilehub.v0.Project
features:
  cloudlogic: !com.amazonaws.mobilehub.v0.CloudLogic
    components:
      api-name: !com.amazonaws.mobilehub.v0.API
        attributes:
          name: api-name
          requires-signin: true
          sdk-generation-stage-name: Development
        paths:
          /items: !com.amazonaws.mobilehub.v0.Function
            codeFilename: {uploads/lambda-archive.zip}
            description: "Handler for calls to resource path : /items"
            enableCORS: true
            handler: lambda.handler
            memorySize: "128"
            name: handler-name
            runtime: nodejs6.10
            timeout: "3"
            environment:
              {MENU_TABLE_NAME}: ___DYNAMIC_PREFIX___{-bbq_menu_item}
              {ORDERS_TABLE_NAME}: ___DYNAMIC_PREFIX___{-bbq_orders}
              {RESTAURANTS_TABLE_NAME}: ___DYNAMIC_PREFIX___-{bbq_restaurants}
          "/items/{proxy+}": !com.amazonaws.mobilehub.v0.Function
            codeFilename: {uploads/lambda-archive.zip}
            description: "Handler for calls to resource path : /items/{proxy+}"
            enableCORS: true
            handler: lambda.handler
            memorySize: "128"
            name: handler-name
            runtime: nodejs6.10
            timeout: "3"
            environment:
              {MENU_TABLE_NAME}: ___DYNAMIC_PREFIX___{-bbq_menu_item}
              {ORDERS_TABLE_NAME}: ___DYNAMIC_PREFIX___{-bbq_orders}
              {RESTAURANTS_TABLE_NAME}: ___DYNAMIC_PREFIX___-{bbq_restaurants}
. . .

uploads:
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {lambda-archive.zip}
    targetS3Bucket: deployments
  - !com.amazonaws.mobilehub.v0.Upload
    fileName: {lambda.jar}
    targetS3Bucket: deployments
. . .
```

4. Save the `.yml` file and rezip the files within the uncompressed project definition file (not the folder containing those files, because that causes a path error).
5. Test your revised project export definition by importing it through the Mobile Hub console. You can verify your environmental variables through the Lambda console.

Note
By default, the Mobile Hub NoSQL Database feature configures a table's permissions to grant read and write access for Lambda functions. The kind of custom IAM policy configuration required to change the table's permissions is not included in the export of a project. An importer of a project dependent on custom policy needs enough information to recreate the policy once they have imported the project. For such a case, we recommend you provide both your policy JSON and step by step instructions (console or AWS CLI) on how and where to attach it. For more information on those steps, see Authentication and Access Control for Amazon DynamoDB.

Importing Cross-Origin Resource Sharing (CORS) Configuration

By default, AWS security infrastructure prevents calls to an API Gateway API from a browser. Configuring CORS for each path of your API securely enables your API calls over the web. CORS configuration is not included in Mobile Hub project export. The following steps describe how to manually include import of CORS configuration in your project export file.

To include CORS configuration for your |ABP| API paths

1. Unzip your exported project definition `.zip` file.
2. Open the export's `mobile-hub-project.yml` file in an editor.
3. For each API path, add a key named `enableCORS` with the value `true` under `... paths: "/items/. . .": !com.amazonaws.mobilehub.v0.Function`, as shown in the following fragment.

```
--- !com.amazonaws.mobilehub.v0.Project
    features:
      cloudlogic: !com.amazonaws.mobilehub.v0.CloudLogic
        components:
          ReactSample: !com.amazonaws.mobilehub.v0.API
            attributes:
              name: ReactSample
              requires-signin: false
            paths:
              "/items/{proxy+}": !com.amazonaws.mobilehub.v0.Function
                name: FirstHandler
                handler: lambda.handler
                enableCORS: true
                runtime: nodejs6.10
                . . .
```

4. Rezip the files within the uncompressed project definition file (not the folder containing those files, because that causes a path error).

Amazon CloudFront Security Considerations for Mobile Hub Users

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

When you enable the AWS Mobile Hub Hosting and Streaming (p. 61) feature, an Amazon CloudFront distribution is created in your account. The distribution caches the web assets you store within an associated Amazon S3 bucket throughout a global network of Amazon edge servers. This provides your customers with fast local access to the web assets.

This topic describes the key CloudFront security-related features that you might want to use for your distribution. For the same type of information regarding the source bucket, see s3-security.

Access management

Hosting and Streaming makes assets in a distribution publically available. While this is the normal security policy for Internet based resources, you should consider restricting access to the assets if this is not the case. The best practice for security is to follow a ?minimal permissions? model and restrict access to resources as much as possible. You may want to modify resource-based policies, such as the distribution policy or access control lists (ACLs), to grant access only to some users or groups of users.

To protect access to any AWS resources associated with a Hosting and Streaming web app, such as buckets and database tables, we recommend restricting access to only authenticated users. You can add this restriction to your Mobile Hub project by enabling the User Sign-in (p. 67) feature, with the sign-in required option.

For more information, see Authentication and Access Control for CloudFront in the *Amazon CloudFront Developer Guide*.

Requiring the HTTPS Protocol

CloudFront supports use of the HTTPS protocol to encrypt communications to and from a distribution. This highly recommended practice protects both the user and the service. CloudFront enables you to require HTTPS both between customers and your distribution endpoints, and CloudFront between your distribution's caches and the source bucket where your assets originate. Global redirection of HTTP traffic to HTTPS, use of HTTPS for custom domains and other options are also supported.

For more information, see Using HTTPS with CloudFront in the *Amazon CloudFront Developer Guide*.

Securing Private Content

CloudFront supports a range of methods for protecting private content in a distribution cache. These include the use of signed cookies and signed URLs to restrict access to authenticated, authorized users.

A best practice is to use techniques like these on both the connection between the user and the distribution endpoint and between the distribution and the content Amazon S3 source bucket.

For more information, see the Serving Private Content through CloudFront section in the *Amazon CloudFront Developer Guide*.

Distribution Access Logging

Distribution logging helps you learn more about your app users, helps you meet your organization's audit requirements, and helps you understand your CloudFront costs. Each access log record provides details about a single access request, such as the requester, distribution name, request time, request action, response status, and error code, if any. You can store logs in an Amazon S3 bucket. To help manage your costs, you can delete logs that you no longer need, or you can suspend logging.

For more information, see Access Logs for CloudFront in the *Amazon CloudFront Developer Guide*.

Amazon S3 Security Considerations for Mobile Hub Users

Looking for the AWS SDKs for iOS and Android? These SDKs and their docs are now part of AWS Amplify.

The content on this page applies only to apps that were configured using AWS Mobile Hub or awsmobile CLI. For existing apps that use AWS Mobile SDK prior to v2.8.0, we highly recommend you migrate your app to use AWS Amplify and the latest SDK.

When you enable the Mobile Hub User File Storage or Hosting and Streaming features, it creates an Amazon S3 bucket in your account. This topic describes the key Amazon S3 security-related features that you might want to use for this bucket. Hosting and Streaming also configures a CloudFront distribution that caches the assets stored in the bucket it creates. For the same type of information regarding the distribution, see cloudfront-security.

Access management

By default, access to Amazon S3 buckets and related objects are private: only the resource owner can access a bucket or assets contained in it. The administrator of a bucket can grant access that suits their design by attaching resource-based policies, such as bucket policy or access control lists (ACLs) to grant access to users or groups of users.

The Amazon S3 configuration provisioned by the AWS Mobile Hub Hosting and Streaming (p. 61) feature is example of setting bucket policy to a allow access to all users. This access policy makes sense in the context of publicly hosting a web app through this feature. We recommend, if it meets app design criteria, that developers also add the User Sign-in (p. 67) feature so that only authenticated users have access to an app's AWS resources like buckets and database.

For more information, see Managing Access Permissions to Your Amazon S3 Resources in the *Amazon S3 Developer Guide*.

Object Lifecycle Management

You can use object lifecycle management to have Amazon S3 take actions on files (also referred to in Amazon S3 as *objects*) in a bucket based on specific criteria. For example, after a specific amount of time since a mobile app user uploaded a file to the bucket, you might want to permanently delete that file or move it to Amazon S3 Glacier. You might want to do this to reduce the amount of data in files that other mobile app users can potentially access. You might also want to manage your costs by deleting or archiving files that you know you or mobile app users no longer need.

For more information, see Object Lifecycle Management in the *Amazon S3 Developer Guide*.

Object Encryption

Object encryption helps increase the protection of the data in files while they are traveling to and from a bucket as well as while they are in a bucket. You can use Amazon S3 to encrypt the files, or you can encrypt the files yourself. Files can be encrypted with an Amazon S3-managed encryption key, a key managed by AWS Key Management Service (AWS KMS), or your own key.

For more information, see the Protecting Data Using Encryption section in the *Amazon S3 Developer Guide*.

Object Versioning

Object versioning helps you recover data in files more easily after unintended mobile app user actions and mobile app failures. Versioning enables you to store multiple states of the same file in a bucket. You can uniquely access each version by its related file name and version ID. To help manage your costs, you can delete or archive older versions that you no longer need, or you can suspend versioning.

For more information, see the Using Versioning section in the *Amazon S3 Developer Guide*.

Bucket Logging

Bucket logging helps you learn more about your app users, helps you meet your organization's audit requirements, and helps you understand your Amazon S3 costs. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. You can store logs in the same bucket or in a different one. To help manage your costs, you can delete logs that you no longer need, or you can suspend logging.

For more information, see Managing Bucket Logging in the *Amazon S3 User Guide*.