

Overview

Amazon Web Services (AWS) provides many services to help customers architect a secure, agile, and scalable backend for their hybrid mobile apps. This eliminates the need for customers to develop and manage their own backend resources for each mobile app feature and can help reduce costs and increase productivity and innovation. But with so many options to fit almost any developer use case, it can be a challenge to choose the best option.

This webpage discusses best practices and technologies to help simplify the development of a backend for both hybrid and native mobile apps, and also introduces an AWS reference architecture for a RESTful mobile backend. The following sections assume basic knowledge of mobile app development, Amazon Cognito user pools, Amazon DynamoDB, Amazon API Gateway, AWS Lambda, and AWS Identity and Access Management (IAM) roles.

The following questions will help you decide if this solution is applicable to your use case:

- *Does your app need a backend API to access backend resources?*
- *Does your app need to store user preferences or app state in a database?*
- *Does your app need authentication and authorization capabilities and related workflows such as user registration and sign-in?*

If you answered yes to one or more of these questions, you're in the right place.

General Best Practices

When you build mobile apps in the cloud, there are some universal design principles that will help you create highly scalable, performant, and resilient apps. For example, use an ephemeral or event-driven architecture to increase security and decrease steady-state server and operational costs. Use a content delivery network to distribute your data geographically closer to your users, providing an optimal end-user experience while overcoming the limitations of mobile connectivity. Also, create a custom domain name for your mobile backend, associate that domain with the appropriate resources, and embed the custom name in your mobile client. This allows you to make changes to your online resources without the need to update your mobile client. With these general principles in mind, consider the following best practices of leveraging managed services for mobile app development:

- If you require user authentication for updates, offload authentication to an identity management service.
- Build event-driven architectures that do not require persistent compute resources.
- Build REST APIs to facilitate secure communication between mobile apps and your backend.
- Leverage managed, highly available NoSQL database services to securely store and query relevant app content.
- Implement mobile push messaging or emails as needed to notify your users of applicable content updates or perform personalized messaging.
- Implement mobile analytics tools for tracking key usage trends and behaviors. This allows you to make data-driven decisions to increase engagement and monetization for your app.

Application on AWS

The following section briefly describes common features of a robust mobile backend and the AWS-managed services that customers commonly use to build these components. It also provides a high-level overview of an AWS-provided reference architecture that combines these services to create a flexible, scalable, and managed backend for mobile apps using the [Ionic](#) framework.

Identity Management

Amazon Cognito can simplify user authentication and authorization, giving customers the options to authenticate users with Amazon Cognito user pools, social identity providers, or their own identity management system. Amazon Cognito provides user sign-up features and integrates with AWS Identity and Access Management which adds additional security capabilities to your mobile backend while simplifying the management of crucial security features for your app. For a full list of Amazon Cognito's identity management features, please visit the Amazon Cognito [website](#).

Event Driven Architecture

AWS Lambda enables customers to easily implement event-driven architectures that do not require persistent resources. You can use Lambda to host backend logic for mobile apps. Lambda automatically runs your code in response to events, and allocates resources to resolve requests on an as-needed basis. This allows you to put minimal logic in the mobile app itself making it easier to scale and update. AWS Lambda automatically monitors Lambda functions on your behalf, reporting metrics through Amazon CloudWatch. To help you troubleshoot failures, Lambda logs all function requests and automatically stores logs through Amazon CloudWatch Logs.

Backend APIs

Amazon API Gateway is a managed service that makes it easier for mobile developers to create, publish, maintain, monitor, and secure APIs at any scale. Use Amazon API Gateway to build REST APIs that your mobile app can use to securely access your back-end services. When combined with AWS Lambda, Amazon API Gateway allows you to create completely serverless APIs.

App Storage

Amazon Cognito Sync supports reading and writing to a local data store. This means that your app can work in the same way regardless of whether the device is online or offline. You can also save user data, such as user preferences, sign-in, and game state, and then sync this data across a user's devices to create a consistent experience.

Amazon DynamoDB provides a managed, highly available NoSQL database for storing and querying app data while preventing superfluous client downloads and content mining. Amazon DynamoDB includes fine-grained access control to follow the best practice of least privilege for Lambda functions querying specific data.

Mobile Push Notifications

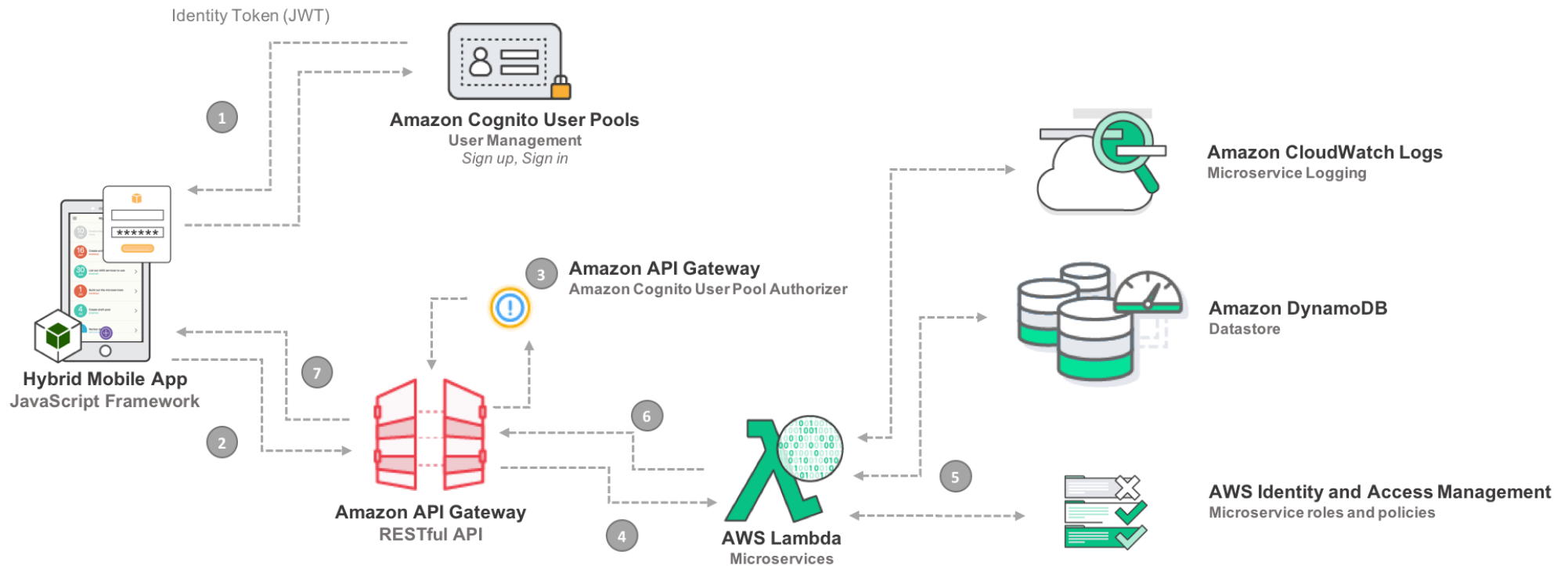
Amazon Simple Notification Service (Amazon SNS) is a fast, flexible, managed push notification service that makes it simple and cost effective to send push notifications to mobile device users, email recipients or even send messages to other distributed services. Mobile push notifications send messages directly to apps on mobile devices, which can appear in the mobile app as message alerts, badge updates, or even sound alerts.

App Analytics

Amazon Mobile Analytics provides capabilities for tracking key trends such as new vs. returning users, app revenue, user retention, and custom in-app behavior such as custom metrics or attributes which you, the app developer, can add to meet any analytical tracking need. Additionally, Amazon Pinpoint helps you reach out to user segments through push notification campaigns to improve user engagement. Use these targeted campaigns to invite inactive users back to your app, offer special promotions, and measure how effective your messages are at keeping your users engaged.

AWS Reference Architecture

The following reference architecture represents a RESTful mobile backend infrastructure that uses AWS managed services to address common requirements for backend resources. The architecture provides capabilities to identify and authenticate users and perform complex queries to return user-relevant data. The following diagram outlines the architectural flow of a RESTful mobile backend for a mobile app, and includes functional components to address these common requirements.



1. When a user signs in to the mobile app, the user's credentials are sent to the Amazon Cognito user pool for authentication. After successful authentication, Amazon Cognito returns an ID token to the app.
2. The mobile app sends HTTPS requests to the Amazon API Gateway RESTful interface with the Amazon Cognito user pool ID token in the authorization header.
3. An Amazon Cognito user pool authorizer associated with the Amazon API Gateway RESTful API validates that the token in the authorization header is an authenticated user.
4. The Amazon API Gateway invokes the AWS Lambda microservice function associated with the requested API resource. AWS Lambda assumes appropriate IAM role to execute a defined task, such as accessing user-specific data in Amazon DynamoDB. All requests that Lambda handles are recorded and stored through Amazon CloudWatch Logs.
5. AWS Lambda returns the results in an HTTP-formatted response to the RESTful API in Amazon API Gateway.
6. Amazon API Gateway returns the results to the mobile app.

AWS offers a [tutorial](#) for customers whose mobile app needs a backend API to access backend services, authorization and authorization capabilities and related workflows such as user registration and sign-in, and/or to store user preferences or app state in a database. The tutorial provides step-by-step instructions for creating secure RESTful APIs using the AWS managed services mentioned above coupled with a sample hybrid app using the Ionic framework.

Resources

[Mobile App Backend for Hybrid Apps Tutorial](#)

<https://s3.amazonaws.com/solutions-reference/mobile-backend/latest/mobile-app-backend.pdf>

A tutorial for building a flexible, scalable, and managed mobile backend to power mobile apps. The guide provides integration details with a sample hybrid mobile app built using the Ionic framework.

[AWS Mobile](#)

<https://aws.amazon.com/mobile>

AWS webpage with AWS Mobile product and pricing details, FAQs, and links to developer resources and other technical documentation

[Amazon Cognito](#)

<https://aws.amazon.com/cognito/>

AWS webpage with Amazon Cognito product and pricing details, FAQs, and links to developer resources and other technical documentation

[AWS Lambda](#)

<https://aws.amazon.com/cognito/>

AWS webpage with AWS Lambda product and pricing details, FAQs, and links to related blogs and forums

[Amazon DynamoDB](#)

https://aws.amazon.com/dynamodb/?nc2=h_m1

AWS webpage with AWS DynamoDB product and pricing details, FAQs, and links to related blogs and forums

[Amazon API Gateway](#)

https://aws.amazon.com/api-gateway/?nc2=h_m1

AWS webpage with Amazon API Gateway product and pricing details, FAQs, and links to related blogs and forums

[Amazon Pinpoint](#)

<https://aws.amazon.com/pinpoint/>

AWS webpage with Amazon Pinpoint product and pricing details, FAQs, and links to related blogs and forums