AWS Decision Guide

Choosing AWS frontend web and mobile services

Choosing AWS frontend web and mobile services: AWS Decision Guide

Table of Contents

Choosing AWS frontend web and mobile services

Taking the first step

Time to read	14 minutes
Purpose	Help determine which AWS services to use when building your web or mobile application.
Last updated	June 25, 2024
Services covered	• AWS Amplify • Amazon API Gateway • AWS AppSync • Amazon CloudWatch • AWS Device Farm • Amazon Location Service • Amazon Pinpoint

Introduction

AWS offers purpose-built tools and services to support frontend development workflows for native iOS, Android, React Native, and JavaScript developers. You can use these tools to develop, deploy, test, operate, and monitor your app.

In addition to using these services and tools to develop your app, you can:

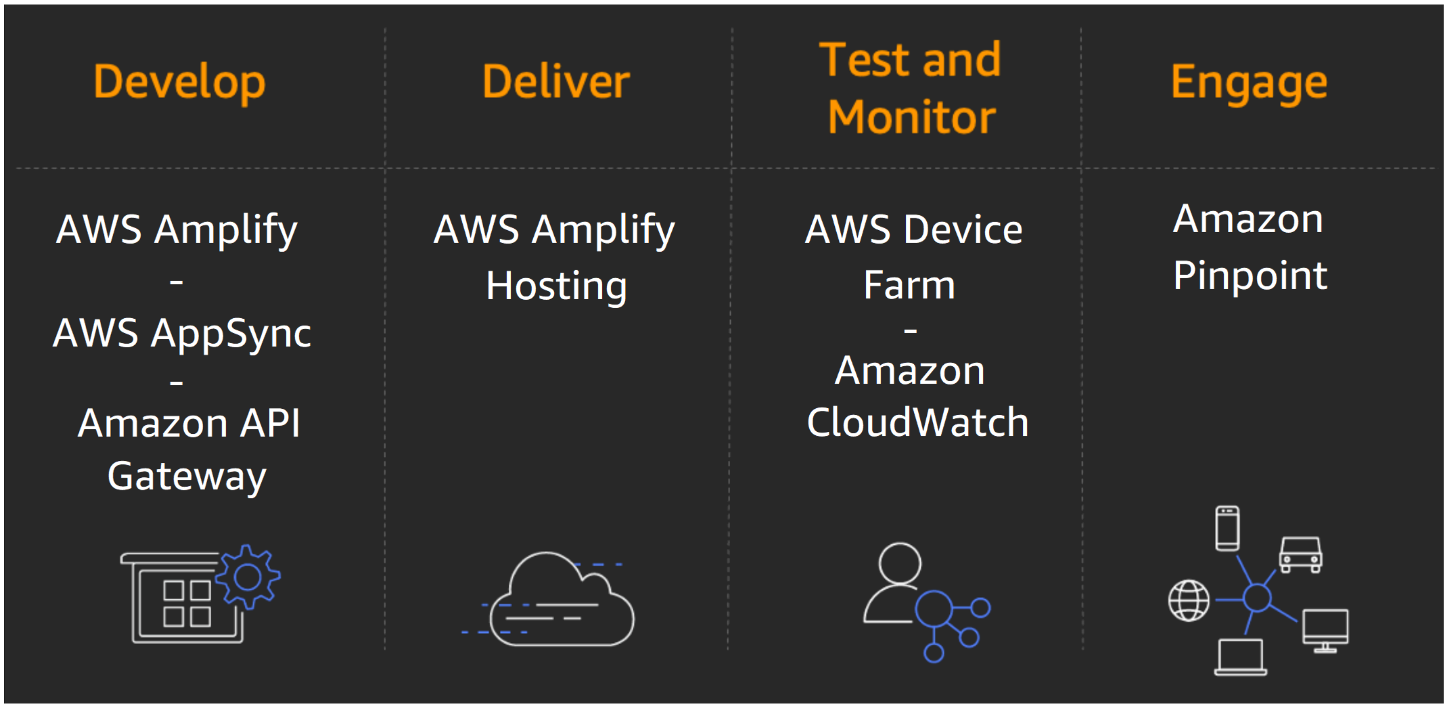
- Take advantage of a built-in software development lifecycle
- Host your app globally
- Combine these tools with other AWS services for storage, caching, authorization, and other components of your app

This decision guide helps you ask the right questions to build your web or mobile app on AWS. It explores how to articulate your business requirements and guides you through the architecture and hosting decisions you'll need to make as you use AWS services to build your app.

In this five minute clip from a 2023 re:Invent talk, Mohit Srivastava, senior manager of product management for AWS Amplify, explores some of the challenges to consider in choosing a frontend web and mobile service.

Understand

Modern web and mobile apps require high performance and an engaging user experience. In addition, you probably have your own set of business requirements that support or extend upon these common needs. Along the app lifecycle, you develop, deliver, test, and monitor your app, and engage with end users, as shown in the following image.


Develop
Deliver
Test and Monitor
Engage
AWS Amplify - AWS AppSync - Amazon API Gateway
AWS Amplify Hosting
AWS Device Farm - Amazon CloudWatch
Amazon Pinpoint

AWS offers a variety of services to support each stage of this lifecycle. The following provides a high-level description of these services. Later, this document guides you through choosing which services you want to use.

What AWS services can you use to develop your app frontend?

AWS Amplify	AWS AppSync	Amazon API Gateway
AWS service service for frontend developers to develop and deploy fullstack	Fully managed API service that automates the operation	AWS service for creating, publishing, maintaining, monitoring, and securing

applications. Amplify provides a fully managed CI/CD and hosting service to deploy your frontend, along with libraries that make it easy to add cloud services to your app for features such as authentic ation, real-time data, files, and generative AI.	and maintenance of GraphQL backends.	REST, HTTP, and WebSocket APIs.

What AWS services can you use to deliver your app frontend?

AWS Amplify

Fully managed service for deploying and hosting full stack web applications, with built-in CI/ CD workflows that accelerate your application release cycle. Choose from flexible deployment options, including custom pipelines, monorepos, or multiple repos.

What AWS services can you use to test and monitor your app frontend?

AWS Device Farm	Amazon CloudWatch
App testing service. You can test your web and mobile apps across desktop browsers and real mobile devices hosted in the AWS Cloud.	AWS service that collects and visualizes logs and metrics from your app hosted in the AWS Cloud.

What AWS services can you use to engage with end users?

Amazon Pinpoint

AWS service that helps you send push notifications, in-app notifications, emails, text messages, and voice messages.

Consider

It's important that you choose the frontend web and mobile services that best meet your needs. But how do you articulate those needs and then map them to the services that fit? The following table is designed to help you do just that.

Business requirements

To make a great choice about the services you'll need for your app, you'll need to answer a few questions about your business requirements first.

- **Why are you building it?** Articulate the business need that you're trying to meet by creating your app. This informs your choices about services.
- **What capabilities will your app users need?** The answer to this question might be a roadmap, where you have a minimum set of launch features and then plan for when to add new capabilities.
- **Where will your customers be using this app?** If your users are international, you might need to offer specific features and plan for associated regulatory considerations.
- **When and how will users need to access data from the app?** This will have implications for when and how the data is both stored and accessed.

These are just a few of the questions that you might ask to articulate your business requirements. Each answer will help you explore the other criteria involved in your choice.

Framework

Are you developing an app for web or mobile? Your answer will guide your choices in design and tooling. In addition, your answers to the business requirements questions on the previous tab will guide and inform your decision about which development framework you want to use.

If you are developing:

- A **web app**, consider the frontend JavaScript framework you will be using. Common examples are React, Next.js, Vue, Angular, Astro, and Svelte.
- A **mobile app**, your choice of framework will depend on which mobile operating system you're developing for – including popular choices such as Android, iOS, or Unity.

Your choice of framework is important, but it's worth noting that it might actually not be a choice. In many cases, your business requirements will indicate that you need to develop for both. For example, if your app is designed to host an online store, you'll likely want customers to be able to access it with a mobile app and on the web.

Hosting strategy

To keep your app's asset delivery times quick, you'll want to **host your static assets close to your users**. Consider using AWS Amplify Hosting, a fully managed hosting service, powered by Amazon CloudFrontwith hundreds of points of presence globally, for static and server-side rendered apps.

AWS Amplify Hosting supports:

- Client-side rendered (single-page application), including frameworks such as React, Angular, and Vue
- Server-side rendered (SSR), including frameworks such as Next, Nuxt, and Gatsby
- Static site generators (SSGs), including solutions such as Gatsby, Eleventy, Hugo, VuePress, and Jekyll

Data querying

You'll probably design an application programming interface (API) to allow users to query data for your app on AWS. You can use GraphQL or REST. The key differences between these two approaches are:

- **GraphQL** is an API query language that defines specifications of how a client application should request data from a remote server. For frontend development, AWS AppSync supports GraphQL.
- **REST** enables client applications to exchange data with a server using HTTP verbs, which is the standard communication protocol of the internet. For frontend development, AWS Amplify, Amazon API Gateway, and AWS Lambda together support REST.

GraphQL speeds up application development by providing a single endpoint to query and update data from multiple databases, microservices, and APIs. It can improve your performance by making fewer network requests to query data from the backend and reduces the amount of data sent to web and mobile clients.

Client-server communication models

Another key consideration is **how your client will communicate with your server—**and how quickly. Consider how often your client will request or need data, and how fresh that data needs to be. You'll likely communicate using one of two common scenarios:

- **In real time –** If you need real-time or near real-time communication, such as with a chat app or a dashboard for monitoring IoT device health, consider using the following communication models:
  - Constantly active WebSockets
  - Very frequent timed polling

  Keep in mind that these options require more battery life and memory usage, so they'll probably increase your cost.
- **Not in real time –** If it doesn't significantly degrade usability for your users to refresh their view manually, or if you need new data loaded only on application or view entry, consider a simple **unidirectional** communication model.

Your choice will help guide you when you start looking at the capabilities of each AWS frontend web and mobile service.

Database services

Consider what kind of **database** best supports your use case. If you have flexibility in choosing your database, consider what native integrations exist with AWS services. For example, both AWS Amplify and AWS AppSync natively support integration with Amazon DynamoDB and Amazon Aurora Serverless. Read Choosing an AWS database service for more information about the database options that AWS offers.

Your database selection might not be an option. If this is the case, consider how to connect your existing database to the AWS services for building and deploying your app.

User profile

Do you want users to have access to their data when they're offline? If so, you might want to build offline support for the app and asynchronous data loading into the app to help users get what they need in near real time. Amplify DataStore, for example, provides a persistent on-device storage engine to support offline access.

Where will your users be located when they use your app? Consider where your app users will need to access their data. If you plan to support users globally or in a specific locale, that information will help you plan your latency and your required points of presence.

In addition, you might consider adding geospatial data and location functionality to your app. You can use Amazon Location Service, for example, to add capabilities to your app such as maps, points of interest, geocoding, routing, geofences, and tracking.

Development strategy

When building apps on AWS, you can use a serverless approach or containers approach. Overall, building serverless-first helps you harness more efficiency in the cloud, while a containers approach gives you more control. This guide focuses on a **serverless** development strategy.

Your development strategy might not be initially a choice if it's already determined in your business requirements. If you're planning to use a containers approach, see Containers at AWS for resources.

Choose

Now that you know the criteria for evaluating your options, you're ready to choose which AWS frontend web and mobile service or services might be a good fit for your organizational requirements.

The following table highlights which services are optimized for which circumstances. Use the table to help determine which services might best fit your organization and use case.

Lifecycle	Capabilities	AWS tools and services
Develop	Develop your Android, iOS, or web app with features such as authentication, on-device and cloud storage, analytics, AI/ML, IoT, and notifications. Natively integrate with other AWS services.	AWS Amplify

Lifecycle	Capabilities	AWS tools and services
	Use GraphQL APIs, and choose from supported frameworks and protocols.	AWS AppSync
	Use REST APIs, and choose from supported frameworks and protocols.	Amazon API Gateway
Deliver	Deliver your app using a git-based workflow. Connect private and public repositor ies from GitHub, BitBucket, GitLab, and AWS CodeCommi t.	AWS Amplify Hosting
Test and monitor	Test your app by using automated testing or remote access.	AWS Device Farm
	Monitor app health with traces, metrics, logs, alarms, and other resources.	Amazon CloudWatch
Engage	Engage with customers by designing user journeys and analyzing user behavior.	Amazon Pinpoint
	Use maps, points of interest, geocoding, and tracking to target your audience.	Amazon Location Service

Use

You should now have a clear understanding of what each AWS front-end web and mobile service (and the supporting AWS tools and services) does—and which might be right for you.

To explore how to use and learn more about each of these services, we have provided a pathway to explore how they work. The following section provides links to in-depth documentation, hands-on tutorials, and resources to get you started.

AWS Amplify

- **Getting started with AWS Amplify**

  Learn how to continuously build, deploy, and host a modern web app.

  Get started with the tutorial
- **Develop your mobile or web app with Amplify Studio**

  Develop a "to-do app" that syncs app data to the cloud and take off from there.

  Explore the guide
- **Build a GraphQL API in the Amplify Dev Center**

  Define a GraphQL schema that you can use to provision backend resources, store data locally, sync to a cloud database, as well as receive updates over a real-time subscription.

  Get started with the tutorial
- **Build an iOS app using a cloud-based backend**

  Create a data-driven native iOS app, integrated with a cloud-based backend.

  Explore the workshop
- **Dive deep with the AWS Amplify Core workshop**

  Build a retail store web application in React using Amplify.

  Explore the workshop

AWS AppSync

- **Getting started with AWS AppSync**

  Use the AWS AppSync console to launch a sample event app schema, automatically provision a DynamoDB data source and connect resolvers, write GraphQL queries and mutations, and use the API in a sample app.

Explore the guide

- **GraphQL Real-time Race workshop**

  Build a fully functioning application for a fast-paced sports event called the GraphQL Real-time Race. Connect the frontend of your application to a robust, performant cloud-enabled backend that delivers real-time updates.

  Explore the workshop

- **AWS AppSync immersion day workshop**

  Manage the service in the AWS Management Console and deploy advanced functionality using the AWS Cloud Development Kit (AWS CDK).

  Explore the workshop

- **GraphQL decision guide**

  Determine if GraphQL is right for your organization.

  Explore the guide

Amazon API Gateway

- **Getting started with the REST API console**

  Create a serverless REST API using the API Gateway REST API console. This exercise takes less than 20 minutes to complete, and is possible within the AWS Free Tier.

  Explore the guide

- **Build an API Gateway REST API with HTTP integration**

  Build an API with HTTP proxy integration or the HTTP custom integration.

  Get started with the tutorial

- **Build an API Gateway REST API with Lambda integration**

  Build an API with Lambda proxy integration or Lambda non-proxy integration.

  Get started with the tutorial

- **Use API Gateway for creating endpoints to support your microservices**

Explore the guide

Amazon CloudWatch

- **Getting started with Amazon CloudWatch**

  Use the CloudWatch console to see key metrics from all AWS services, and focus on specific metrics and alarms.

  Explore the guide
- **Analyzing log data with CloudWatch Logs Insights**

  Use CloudWatch Logs Insights to search and analyze your log data. Query for operational issues, identify potential causes, and validate deployed fixes.

  Explore the guide
- **Using CloudWatch anomaly detection**

  Enable anomaly detection that continuously analyzes you app's metrics, determines baselines, and surfaces anomalies with minimal user intervention.

  Explore the guide
- **Using Amazon CloudWatch dashboards**

  Use CloudWatch dashboards to monitor your resources in a single view, and create customized views of the metrics and alarms for your AWS resources.

  Explore the guide

AWS Device Farm

- **Getting started with AWS Device Farm**

  Use AWS Device Farm to test a native Android or iOS app. Create a project, upload an `.apk` or `.ipa` file, run a suite of standard tests, and then view the results.

  Explore the guide
- **AWS Device Farm test devices list**

View a list of devices available for testing on AWS Device Farm.

Explore the list

- **Access your private network from real mobile devices using AWS Device Farm**

  Use the VPC-ENI connectivity feature of AWS Device Farm to connect to a web application that is hosted in Amazon VPC.

  Read the blog

Amazon Location Service

- **Quick start: Creating a web app**

  Create a static webpage with a map and the ability to search at a location.

  Get started with the tutorial
- **Creating an Android app**

  Create an Android application with a map and the ability to search at a location.

  Get started with the tutorial
- **Create a Custom Map Style with Amazon Location Service**

  Style an existing map using Maputnik to help visualize the changes in real-time.

  Read the blog

Amazon Pinpoint

- **Getting started with Amazon Pinpoint**

  Start sending targeted messages in Amazon Pinpoint, including creating segments that target certain customers and using analytics dashboards.

  Explore the guide
- **Using Postman with the Amazon Pinpoint API**

  Set up and use Postman with Amazon Pinpoint. Postman is a popular tool for testing APIs in an easy-to-use graphical environment.

  Get started with the tutorial

- **Setting up an SMS registration system**

  Set up a web form to capture customers' contact information. Amazon Pinpoint then sends the customer a message asking them to confirm their subscription and opts them in to receiving your messages.

  Get started with the tutorial

Explore

- **Architecture diagrams**

  Explore reference architecture diagrams to help you develop, scale, test, and deploy your front-end and mobile applications.

  Explore architecture diagrams
- **Whitepapers**

  Explore whitepapers to help you get started, learn best practices, and understand your front-end and mobile applications options on AWS.

  Explore whitepapers
- **AWS Solutions**

  Explore vetted solutions and architectural guidance for common use cases for front-end and mobile applications.

  Explore solutions

Document history

The following table describes the important changes to this decision guide. For notifications about updates to this guide, you can subscribe to an RSS feed.

Change	Description	Date
Minor update	Added new links.	July 23, 2025
Feature update	Editorial changes throughout.	June 25, 2024
Initial publication	Guide first published.	January 12, 2024