



SUMMIT
ONLINE

BUI07

Building serverless applications with AWS Amplify

Derek Bingham

Senior Partner Solutions Architect
Amazon Web Services

Quick recap: AWS Amplify

AWS Amplify – a development platform

Components

CLI

Client Libs (iOS, Android, JS)

UI Components

Open Source
Framework

Amplify Console – CI/CD
and Hosting

AWS Device Farm

AWS Managed
Developer
Services



AWS Amplify recap: CLI



```
# create new project
```

```
$ amplify init
```

```
# add feature
```

```
$ amplify add api
```

```
# test locally
```

```
$ amplify mock
```

```
# push changes
```

```
$ amplify push
```

```
# update feature
```

```
$ amplify update api
```

Convention over configuration

Manage single/multi-environment

Local mocking and testing

Code generation



native code



types & statements

AWS Amplify recap: Client



```
// import Amplify components
import { API } from 'aws-amplify'

// call Amazon API Gateway endpoint
const data = await API.get('orderApi', '/orders')
```



```
// import React component
import { withAuthenticator } from 'aws-amplify-react'

// main App component definition
class App extends React.Component {
  // your beautiful code
}

// add authentication
export default withAuthenticator(App)
```

Interact with services via client-side

Amplify Native for iOS and Android

JavaScript (JS) client for web and
React Native

JS framework-specific components



AWS Amplify recap: Categories

DataStore



On-device persistent storage that automatically synchronises data between you apps and the cloud.

Predictions



Add ML capabilities to your app powered by cloud services

Analytics

Track user sessions, custom user attributes and in-app metrics

API

HTTP requests using REST and GraphQL with support for real-time data

Auth

AuthN + AuthZ library with prebuilt UI components for your app

Interactions

Conversational bots powered by deep learning technologies

PubSub

Connect your app to message-oriented middleware on the cloud

Notifications

Push notifications with campaign analytics and targeting

Storage

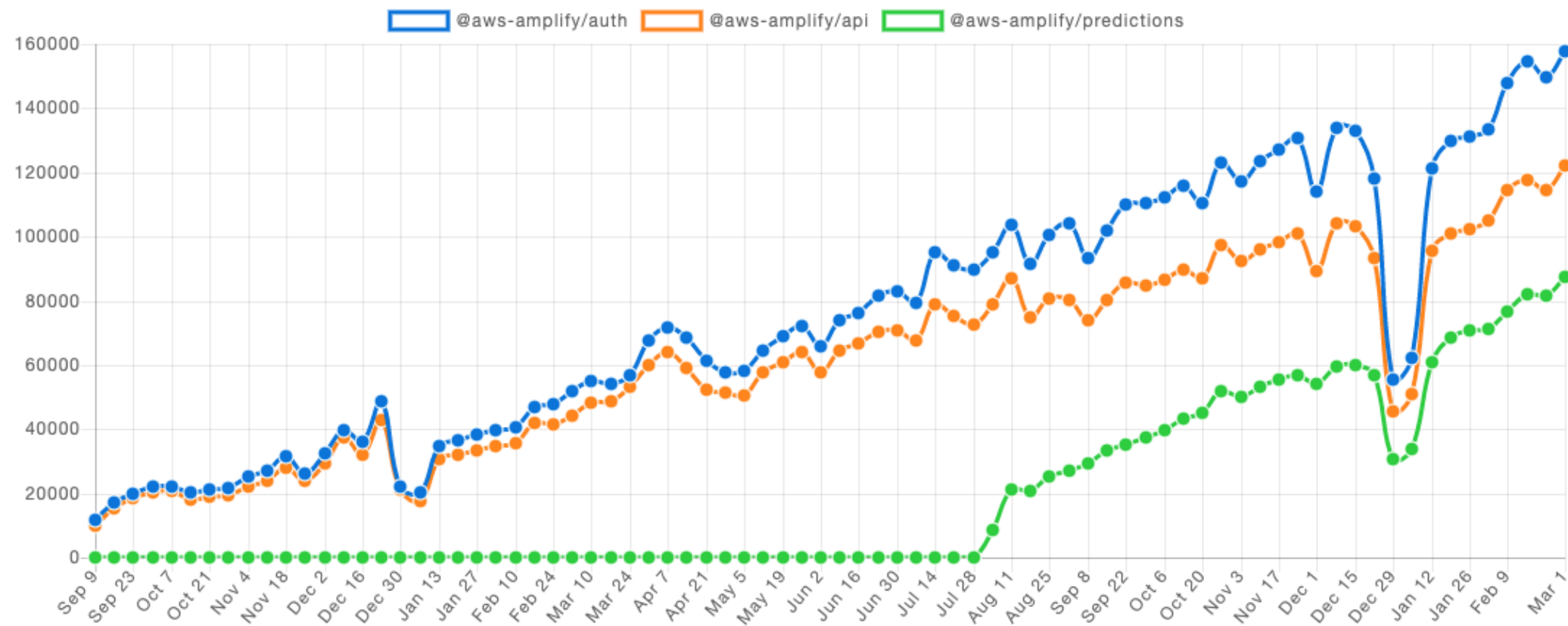
Manage user content securely in public, protected, and private storage.

XR

Work with augmented reality and virtual reality content in your apps

AWS Amplify recap: Category adoption

Downloads in past 2 Years ▾



AWS Amplify recap: Amplify Console

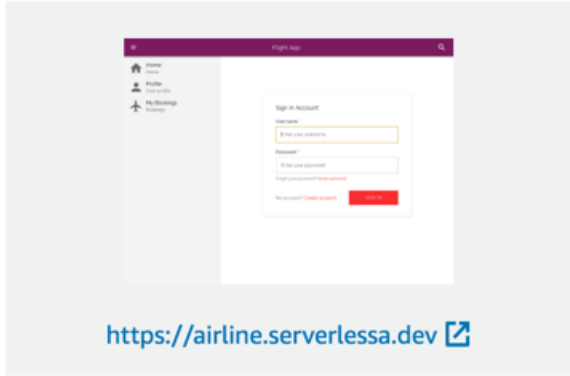
All apps > aws-serverless-airline-booking

aws-serverless-airline-booking

This page lists all connected branches. Select a branch to view build details.

[Connect branch](#) [Actions](#) ▼

twitch



<https://airline.serverlessa.dev>

Last deployment
26/07/2019, 07:50:53

Last commit
Merge branch 'develop' into twitch | 3eae9f | [GitHub - twitch](#)

Git-based CI/CD for full-stack serverless apps

Jumpstart building serverless apps

What's new

What's new: Native SDK's

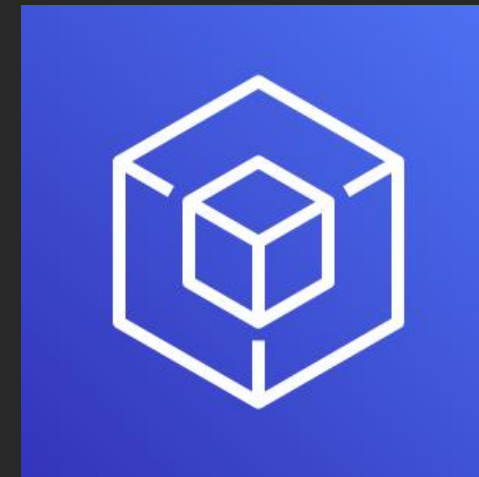


Amplify Clients

Use-case centric

Declarative abstractions

for example : `Storage.put()`



AWS Mobile SDKs

AWS Service centric

Low level generated

For example : `AWSS3TransferUtilityUploadExpression`

What's new: Native categories

DataStore



On-device persistent storage that automatically synchronises data between you apps and the cloud.

Predictions



Add ML capabilities to your app powered by cloud services

Analytics

Track user sessions, custom user attributes and in-app metrics

API

HTTP requests using REST and GraphQL with support for real-time data

Auth

AuthN + AuthZ library with prebuilt UI components for your app

Interactions

Conversational bots powered by deep learning technologies

PubSub

Connect your app to message-oriented middleware on the cloud

Notifications

Push notifications with campaign analytics and targeting

Storage

Manage user content securely in public, protected, and private storage.

XR

Work with augmented reality and virtual reality content in your apps

Example: Predictions in IOS

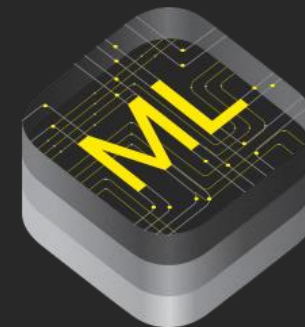


Custom models

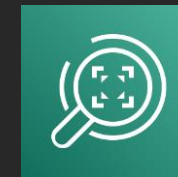


Amazon
SageMaker

CoreML models



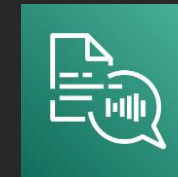
Pre-trained models



Amazon
Rekognition



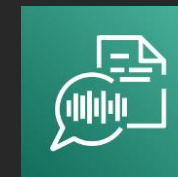
Amazon
Comprehend



Amazon Polly



Amazon
Textract



Amazon Transcribe



Amazon
Translate

Escape hatch

For when convention isn't enough

Currently available categories:

- Analytics
- Authentication
- Predictions
- Storage



```
let rekognitionService = Amplify.Predictions.getEscapeHatch  
(key: .rekognition) as! AWSRekognition  
  
let request = rekognitionService.AWSRekognitionCreateCollectionRequest()  
  
rekognitionService.createCollection(request)
```

Demo – adding predictions

Quick recap

What's new: Amplify DataStore

Multi-platform (**iOS/Android/React Native/Web**) on-device persistent storage engine that automatically synchronises data between mobile/web apps and the cloud using GraphQL.



Automatic versioning,
conflict detection
and resolution in
the cloud



Familiar and
local first
programming
model

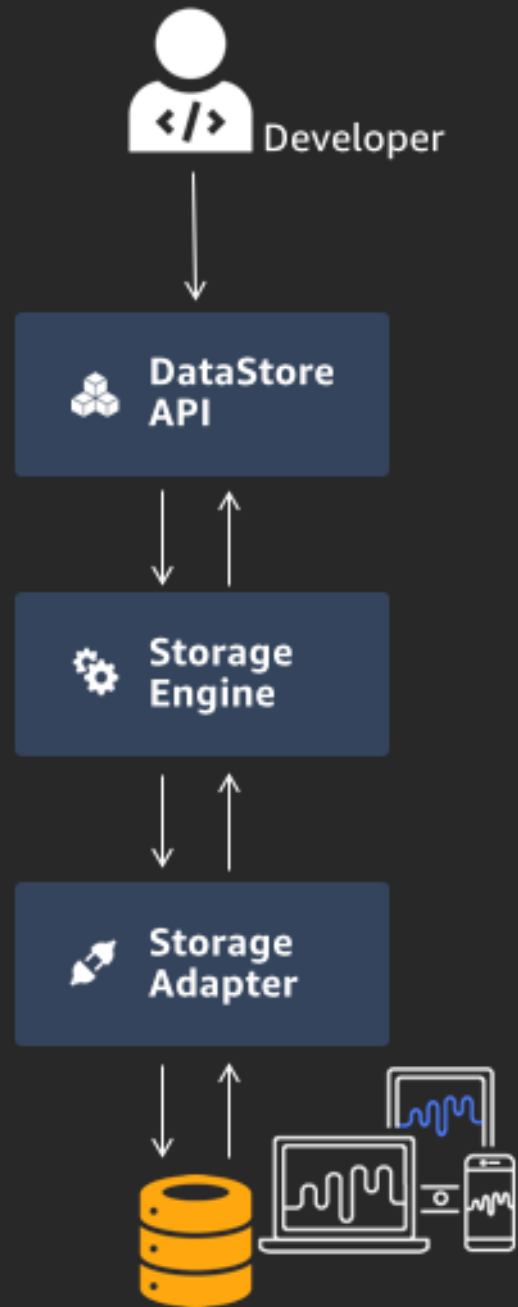


Delta Sync
and Auto-merge



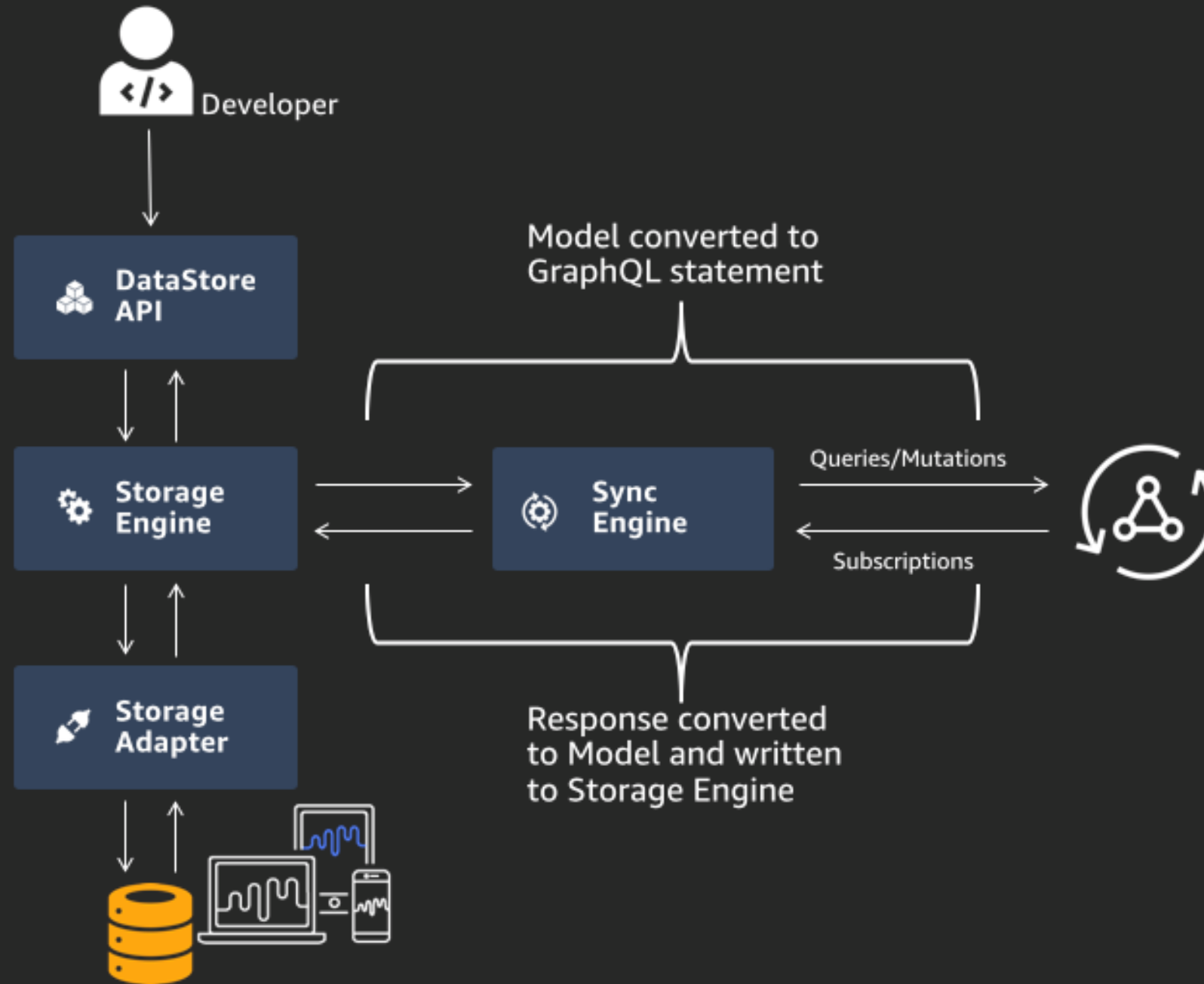
Powered by
AppSync and
GraphQL

What's new: Amplify DataStore



No AWS
Account
needed yet!

What's new: Amplify DataStore



before

```
function Posts({ params }) {
  const { subscribeToMore, ...result } = useQuery(POST_QUERY)

  return (
    <PostsPage
      {...result}
      subscribeToNewPosts={() =>
        subscribeToMore({
          document: POSTS_SUBSCRIPTION,
          updateQuery: (prev, { subscriptionData }) => {
            if (!subscriptionData.data) return prev;
            const newPost = subscriptionData.data.newPost;
            return {
              ...prev,
              {
                posts: [newPost, ...prev.posts]
              }
            }
          }
        })
      }
    />
  )
}
```

after

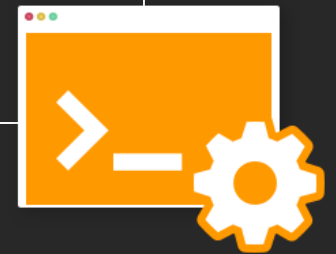
```
const subscription = DataStore.observe(Post)
  .subscribe(msg => fetchPosts())

async function fetchPosts() {
  const posts = await DataStore.query(Post)
  setState(current => ({ ...current, posts }))
}
```

What's new: Transformers

```
$ amplify add api
```

```
# schema.graphql  
type Post @model {  
  id: ID!  
  title: String!  
}
```



What's new: Transformers

@model

Top-level entity; creates DynamoDB table, resolvers, and additional schema (queries, mutations, and subscriptions) for base type

@connection

Enables relationships between @model types

@auth

Enables set of authorisation rules

@searchable

Handles streaming the data of an @model object type to Amazon Elasticsearch Service and configures search resolvers

@versioned

Enables versioning

@function

Enables adding a Lambda function as a data source

@key

Enables configuring custom indexes for @model types

@predictions

Access an orchestration of AI/ML services such as Amazon Rekognition, Amazon Translate, and/or Amazon Polly

Demo – creating a new data model

Quick recap

6 tips and best practices

AWS Amplify tips and best practices (1)



Enable authorisation with
`@auth` to protect models

AWS Amplify tips and best practices (2)



Take advantage of field-level authorisation

AWS Amplify tips and best practices (3)



Use local mocking and testing
to quickly test things out

AWS Amplify tips and best practices (4)



Use multiple environments to
test in production

AWS Amplify tips and best practices (5)



Amplify Console
branch previews

AWS Amplify tips and best practices (6)



Get to know [@key](#) directives for additional access patterns

Go build serverless apps
(with AWS Amplify)

Links and Resources

<https://amplify.aws/community/>

<https://awsappsync.dev/>

<https://aws-amplify.github.io/>

<https://github.com/aws-amplify>

Thank you!

Derek Bingham

 @derekwbingham

 @deekob