

# AWS re:INVENT

## Mobile State of the Union

**AMIT PATEL**

General Manager, AWS Mobile

November 28, 2017

**AWS re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



This session covers the current state of the union for mobile application development on AWS, providing an overview of the services available to mobile developers from AWS. We discuss the entire lifecycle of the mobile application process from building, testing, deploying, and production, to growing your user base and business with ongoing engagement and campaigns.

## MOBILE INDUSTRY TRENDS

**1.6T  
hours**

Time spent in apps  
and growing

Source: AppAnnie

**50%+**

New enterprise apps  
built with web  
technologies

Source: AWS

**66.7%▲**

JavaScript – most  
commonly used  
programming  
language

Source: Stack Overflow

# Building High Quality, Cloud-Enabled Apps

# THREE SIMPLE STEPS

## 1. Pick a Platform



## 2. Set Up Cloud Services



Mobile Hub  
Mobile CLI

## 3. Connect Your App



Native SDKs  
AWS Amplify

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# MOBILE CLI: SETTING UP CLOUD SERVICES

## Initialize your app

```
> awsmobile init
```

## Enable User Sign-in

```
> awsmobile user-signin enable
```

## Supported services:

- **user-signin** (Amazon Cognito)
- **analytics** (Amazon Pinpoint)
- **database** (Amazon DynamoDB)
- **user-files** (Amazon S3)
- **cloud-api** (API GW & AWS Lambda)

## Web app deployment support:

- **hosting** (Amazon S3 and Amazon CloudFront)

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

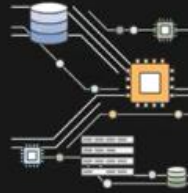


# THE AWS AMPLIFY LIBRARY



## JavaScript Library

- Declarative interfaces
- Convention over configuration



## Categories for application programming with Cloud services

- Auth, Analytics, Storage, API
- Caching, i18n, logging, message bus



## React/React Native extensions

- Higher-order components
- Native bridging for mathematical operations (Amazon Cognito User Pools)



- Open sourced – Apache 2.0
- Implemented with AWS services, open for external contribution

**AWS re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# USING AMPLIFY TO CONNECT YOUR APP

```
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3 import Amplify, { I18n, withAuthenticator } from 'aws-amplify-react-native';
4 import awsExports from './aws-exports';
5
6 Amplify.configure(awsExports);
7 I18n.setLanguage("fr");
8
9 class App extends React.Component {
10   render() {
11     return (
12       <View style={styles.container}>
13         <Text>Open up App.js to start working on your app.
14         <Text>Changes you make here are reflected in the running app.
15         <Text>Shake your phone to turn on the Shake Gesture.
16       </View>
17     );
18   }
19 }
```



Higher-order components enable a functional sign-in UI with just **two** lines of code

**AWS re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

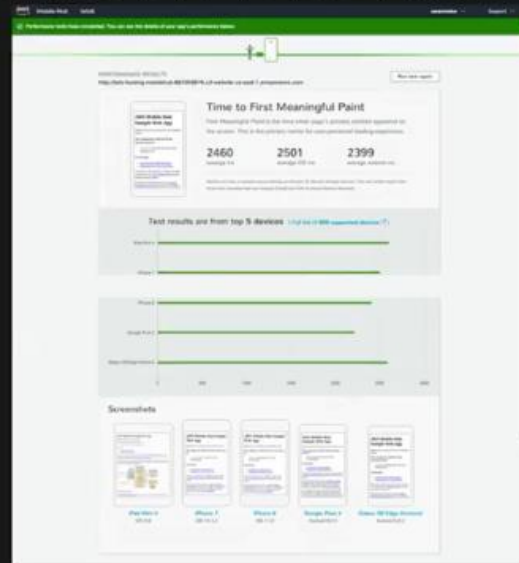


# TESTING AND DEPLOYING YOUR WEB APP

```
> awsmobile publish --test
```

Test your web app on real devices  
for performance insights

Host your web app on S3 (served  
through CloudFront)



AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# THE MOBILE HUB CONSOLE

Use the Mobile Hub console to add more platforms and features

```
> awsmobile console
```

## Apps

A list of apps you can cloud enable with the AWS features you have configured in your backend.

[Add new app](#)

Platform	Backend features	Integrate	Performance test	Run test
iOS		<button>Integrate</button>		
android		<button>Integrate</button>		
JS web		<button>Integrate</button>		<button>Run test</button>
react native		<button>Integrate</button>		

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.





# APP DATA: HARD PROBLEMS REMAIN



Data requirements vary across devices and become harder when multiple users share data



Users want instant access to data



Users want to continue using their apps even with low or no connectivity



Building scalable data-driven apps without learning distributed systems concepts is hard

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



## AWS AppSync

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



**AWS AppSync uses GraphQL** to store data and subscriptions

# CREATE POWERFUL DATA-DRIVEN APPS



**Real-time,  
Collaborative  
Apps**



**Offline  
Programming  
Model with  
Sync**



**Get Only the  
Data You Need  
with GraphQL**



**Access Data  
from Multiple  
Sources**



**Fine-grained  
Access Control**

**AWS  
re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

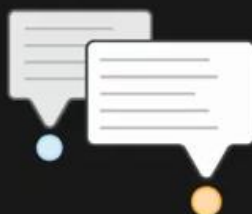


# APPS YOU CAN BUILD WITH APPSYNC

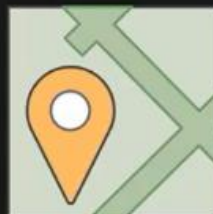


**Collaboration Apps**

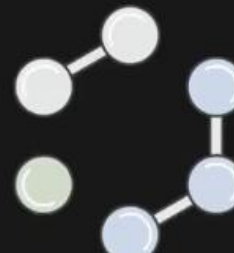
- Dashboards
- Leaderboards
- Whiteboards
- ...



**Social Media, Chat,  
and Dating Apps**



**Geo Apps**



**Apps with Complex  
Data Structures  
and Types**

**AWS  
re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# START BUILDING FAST



AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# EASY ACCESS TO RICH DATA

```
type Query {  
  getTodos: [Todo]  
}
```

```
type Todo {  
  id: ID!  
  name: String  
  description: String  
  priority: Int  
  dueDate: String  
}
```

**Model data with  
application schema**

```
query {  
  getTodos {  
    id  
    name  
    priority  
  }  
}
```

**Client requests what it  
needs**

```
{  
  "id": "1",  
  "name": "Get Milk",  
  "priority": "1"  
},  
  "id": "2",  
  "name": "Go to gym",  
  "priority": "5"  
},...  
}
```

**Only that data is  
returned**

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

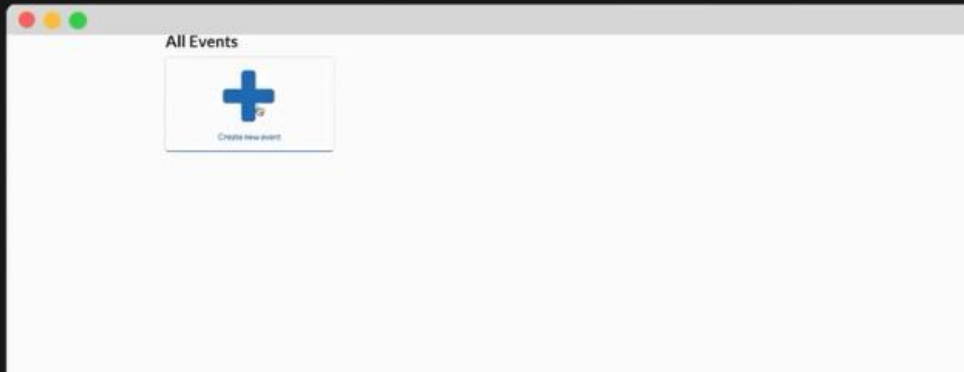


## Getting Started with AWS AppSync

MICHAEL PARIS

Software Development Engineer, AWS Mobile

# EVENT APPLICATION



# EVENT APPLICATION

A screenshot of a web application window titled "Create an event". The window has a light gray header bar with three colored window control buttons (red, yellow, green) on the left. Below the header, there is a white form area. The form contains the following fields:

- Name\***: A text input field with the placeholder text "Build data driven apps with GraphQL and AWS AppSync".
- When\***: A date and time picker showing "November 20, 2017 5:15 PM".
- Where\***: A text input field with the placeholder text "S Palazzo North".
- Description\***: A text area with the placeholder text "Come learn more about AWS AI".

At the bottom of the form, there are two buttons: "Cancel" and "Save". The "Save" button is green and has a white checkmark icon.

# EVENT APPLICATION

A screenshot of a web application window titled "Create an event". The window has a light gray header bar with three colored window control buttons (red, yellow, green) on the left. Below the header, there is a white form area. The form contains the following fields:

- Name\***: A text input field with the placeholder text "Build data driven apps with GraphQL and AWS AppSync".
- When\***: A date and time picker showing "November 27, 2017 3:30 PM".
- Where\***: A text input field with the placeholder text "S Palazzo North".
- Description\***: A text area with the placeholder text "Come learn more about AWS AI".

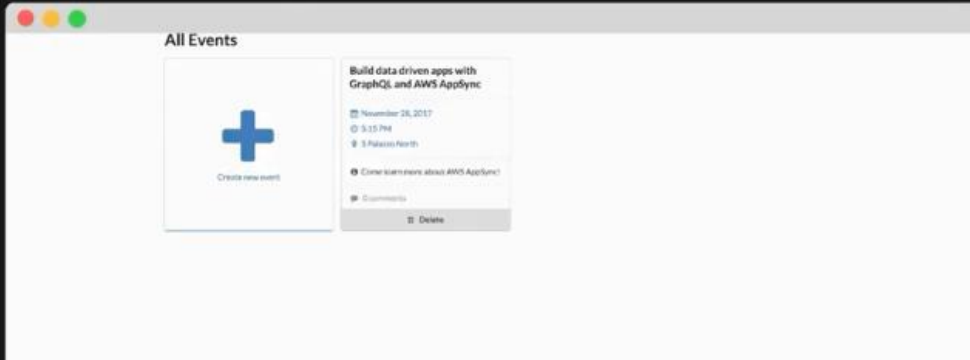
At the bottom of the form, there are two buttons: "Cancel" and "Save". The "Save" button is green and has a white checkmark icon.

The date and time picker is open, showing a calendar for November 2017 and a list of times. The selected date is November 27, 2017, and the selected time is 3:30 PM.

November 2017							Time
Su	Mo	Tu	We	Th	Fr	Sa	
							10:00 am
							04:15 pm
29	30	31	1	2	3	4	04:30 pm
5	6	7	8	9	10	11	04:45 pm
12	13	14	15	16	17	18	05:00 pm
19	20	21	22	23	24	25	05:15 pm
26	27	28	29	30	1	2	05:30 pm
3	4	5	6	7	8	9	05:45 pm

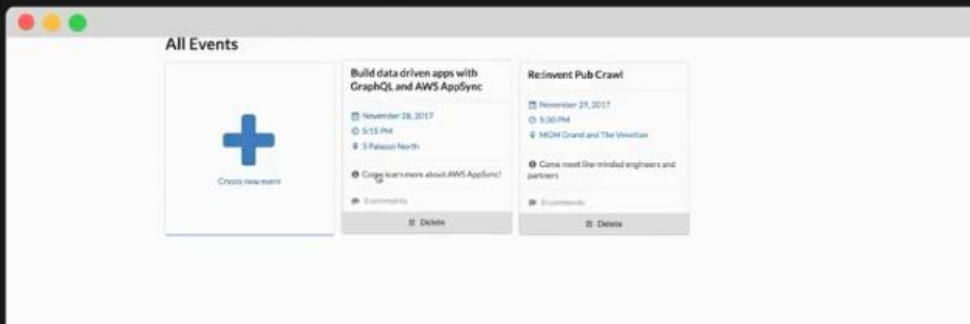


# EVENT APPLICATION



This app uses 2 types of models, an Event model and a Comment model.

# EVENT APPLICATION



# EVENT APPLICATION



**AWS AppSync helps you build applications faster**

# DEPLOY AN API

The screenshot shows the AWS AppSync console landing page. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and a user profile. The main heading is 'MOBILE SERVICES' followed by 'AWS AppSync' and the tagline 'Build data driven apps with real-time and off-line capabilities'. A paragraph describes how AppSync makes it easy to build data-driven mobile and browser-based apps. Below this is a 'How it works' diagram showing the flow from a data source to an API and then to a client. On the right, there's a 'Create AWS AppSync API' button, a 'Pricing (US)' section with details on query and update costs, and a 'Getting started' section with links to documentation.

**MOBILE SERVICES**

## AWS AppSync

Build data driven apps with real-time and off-line capabilities

AWS AppSync makes it easy to build data driven mobile and browser-based apps that deliver responsive, collaborative experiences by keeping the data updated when devices are connected, enabling the app to use local data when offline, and synchronizing the data when the devices reconnect. AWS AppSync uses the open standard GraphQL query language so you can request, change, and subscribe to the exact data you need with just a few lines of code.

### How it works

The diagram illustrates the workflow: 'Create and update schema' (a document icon) leads to 'Connect Data Source' (a database icon), which then connects to 'AppSync enables API to real-time updates' (a server icon). Below the diagram, three steps are listed: 1. Create and update schema, 2. Connect Data Source, and 3. AppSync enables API to real-time updates.

### Create AWS AppSync API

[Create API](#) Estimated 5-5 mins

### Pricing (US)

- \$4 per million Query and Data Modification Operations\*
- \$2 per million Real-time Updates\*
- \$0.08 per million minutes connected to Real-time Updates

### Getting started

- [What is AWS AppSync?](#) 2 min read
- [Getting started with AWS AppSync](#) 6 min watch
- [Designing GraphQL schemas](#) 14 min read
- [GraphQL Overview](#) 10 min read

We want to create an API for our app

# DEPLOY AN API

The screenshot shows the 'Create new API' form in the AWS AppSync console. The form has two main sections. The first section, 'API name', has a text input field with 'EventApp' entered. The second section, 'Select a template or custom schema', has two radio buttons: 'Custom schema' (selected) and 'Sample schema'. At the bottom right, there are 'Cancel' and 'Create' buttons. The footer of the console shows 'Feedback', 'English (US)', and copyright information.

## Create new API

### API name

API name  
Type a name for your API.

EventApp

A name starts with letter and contains only numbers, letters and ".".

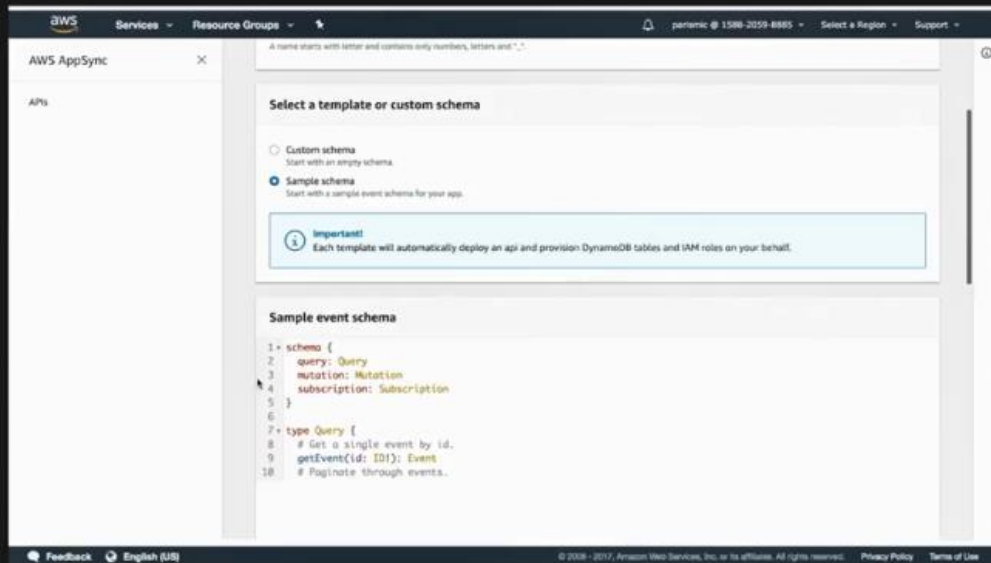
### Select a template or custom schema

- ☒ Custom schema  
Start with an empty schema.
- ☐ Sample schema  
Start with a sample event schema for your app.

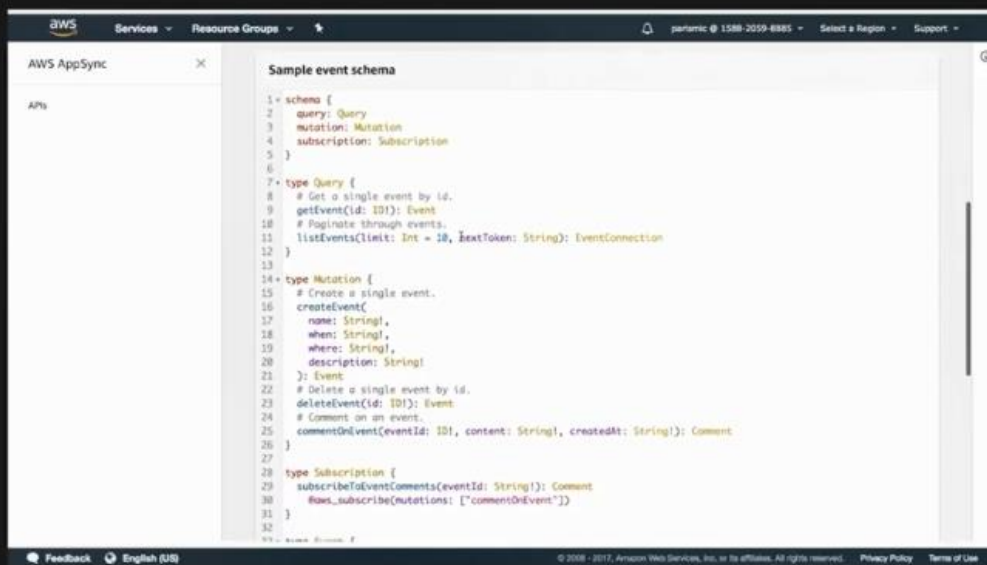
[Cancel](#) [Create](#)

Feedback English (US) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

# DEPLOY AN API

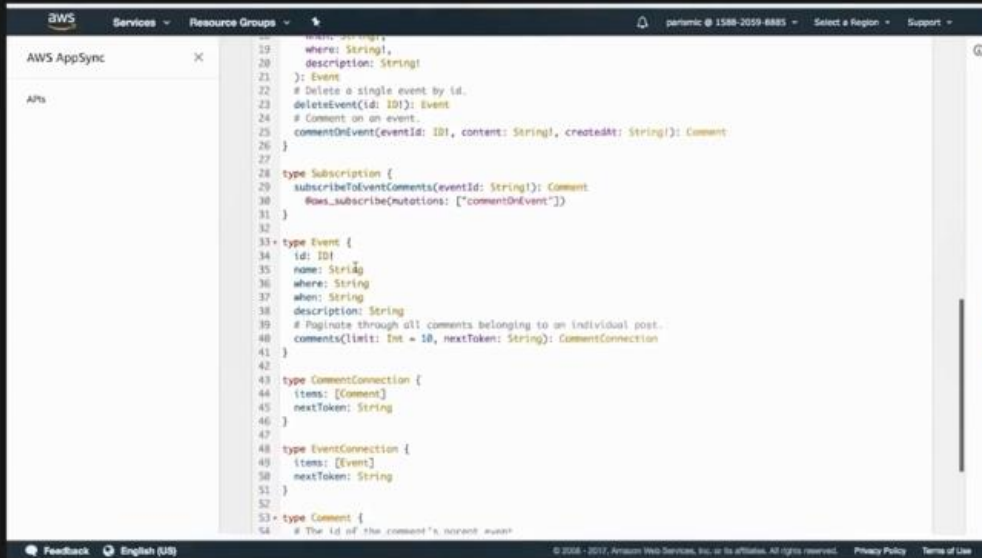


# DEPLOY AN API



A schema in GraphQL defines the data model and Types, as well as the operations that you have access to within your API

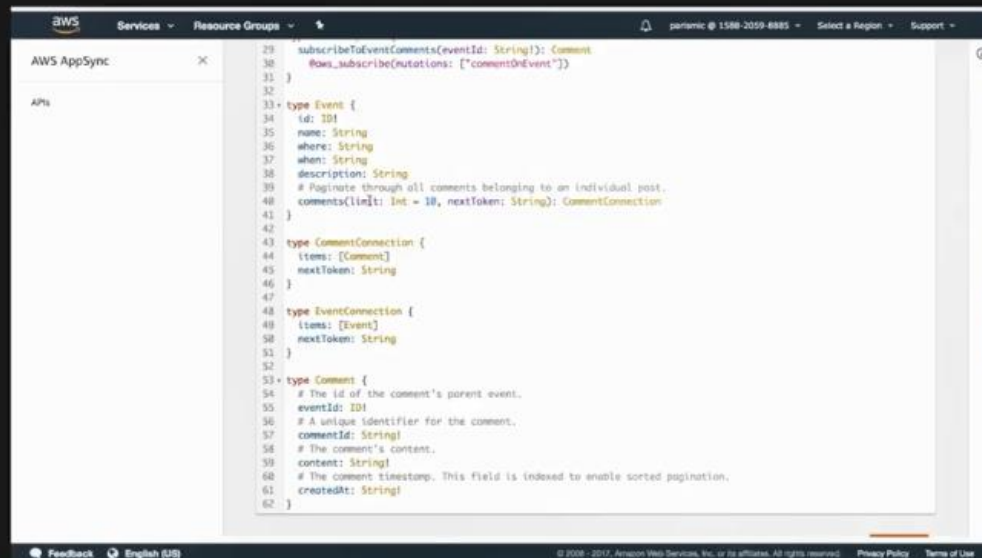
# DEPLOY AN API



The screenshot shows the AWS AppSync console with a GraphQL schema defined. The schema includes types for Event, Comment, Subscription, EventConnection, and CommentConnection. The Event type has fields for id, name, where, when, and description. The Comment type has fields for id, content, and createdAt. The Subscription type has a field for subscribeToEventComments. The EventConnection type has fields for items and nextToken. The CommentConnection type has fields for items and nextToken. The schema is defined as follows:

```
19 where: String!
20 description: String!
21 ): Event
22 # Delete a single event by id.
23 deleteEvent(id: ID!): Event
24 # Comment on an event.
25 commentOnEvent(eventId: ID!, content: String!, createdAt: String!): Comment
26 }
27
28 type Subscription {
29   subscribeToEventComments(eventId: String!): Comment
30   #aws_subscribe(mutations: ["commentOnEvent"])
31 }
32
33 type Event {
34   id: ID!
35   name: String!
36   where: String!
37   when: String!
38   description: String!
39   # Paginate through all comments belonging to an individual post.
40   comments(limit: Int = 10, nextToken: String!): CommentConnection
41 }
42
43 type CommentConnection {
44   items: [Comment]
45   nextToken: String
46 }
47
48 type EventConnection {
49   items: [Event]
50   nextToken: String
51 }
52
53 type Comment {
54   # The id of the comment's parent event.
55   eventId: ID!
56   # A unique identifier for the comment.
57   commentId: String!
58   # The comment's content.
59   content: String!
60   # The comment timestamp. This field is indexed to enable sorted pagination.
61   createdAt: String!
62 }
```

# DEPLOY AN API

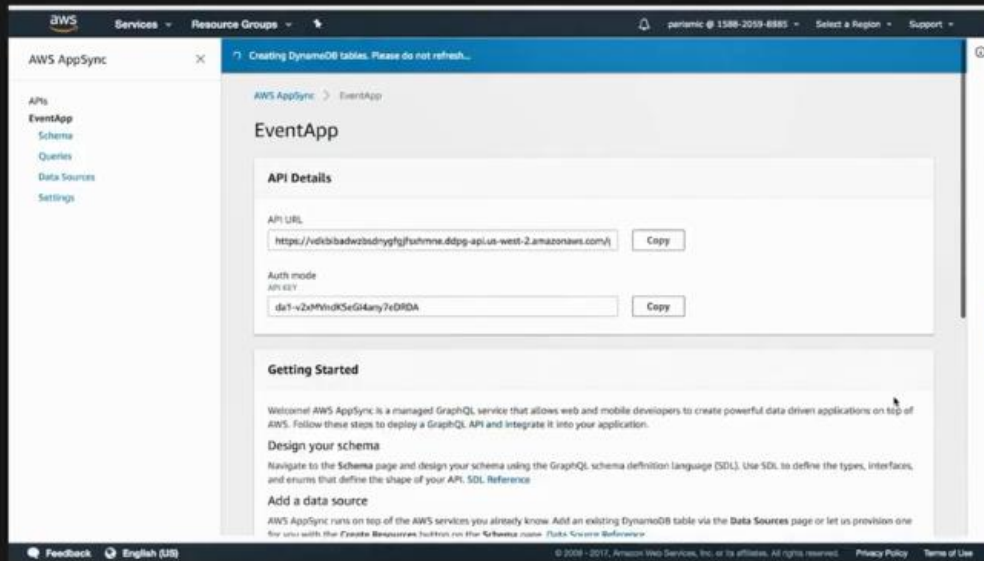


The screenshot shows the AWS AppSync console with a GraphQL schema defined. The schema includes types for Event, Comment, Subscription, EventConnection, and CommentConnection. The Event type has fields for id, name, where, when, and description. The Comment type has fields for eventId, commentId, content, and createdAt. The Subscription type has a field for subscribeToEventComments. The EventConnection type has fields for items and nextToken. The CommentConnection type has fields for items and nextToken. The schema is defined as follows:

```
29 subscribeToEventComments(eventId: String!): Comment
30 #aws_subscribe(mutations: ["commentOnEvent"])
31 }
32
33 type Event {
34   id: ID!
35   name: String!
36   where: String!
37   when: String!
38   description: String!
39   # Paginate through all comments belonging to an individual post.
40   comments(limit: Int = 10, nextToken: String!): CommentConnection
41 }
42
43 type CommentConnection {
44   items: [Comment]
45   nextToken: String
46 }
47
48 type EventConnection {
49   items: [Event]
50   nextToken: String
51 }
52
53 type Comment {
54   # The id of the comment's parent event.
55   eventId: ID!
56   # A unique identifier for the comment.
57   commentId: String!
58   # The comment's content.
59   content: String!
60   # The comment timestamp. This field is indexed to enable sorted pagination.
61   createdAt: String!
62 }
```

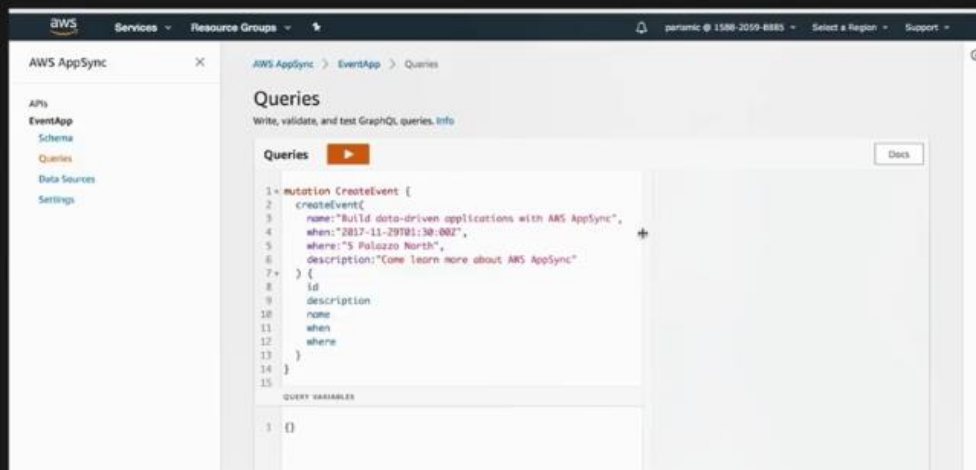


# DEPLOY AN API



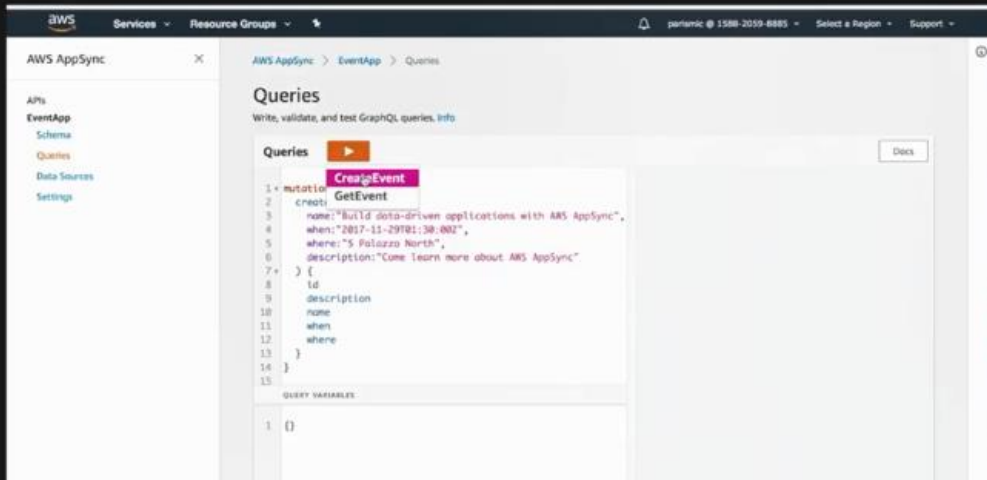
When you click the Create button with the sample schema, AppSync goes and does some things for you. It is going to provision DynamoDB tables in your account, IAM Roles that will give AWS permissions to access those database resources on your behalf,

# DEPLOY AN API

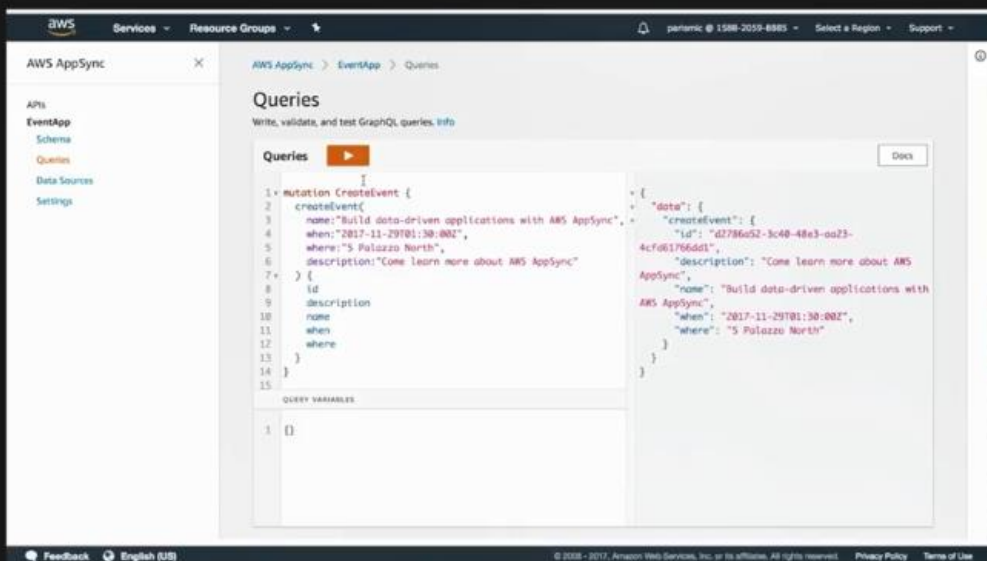


After the tables have been provisioned, the API is now live and ready to be used.

# DEPLOY AN API

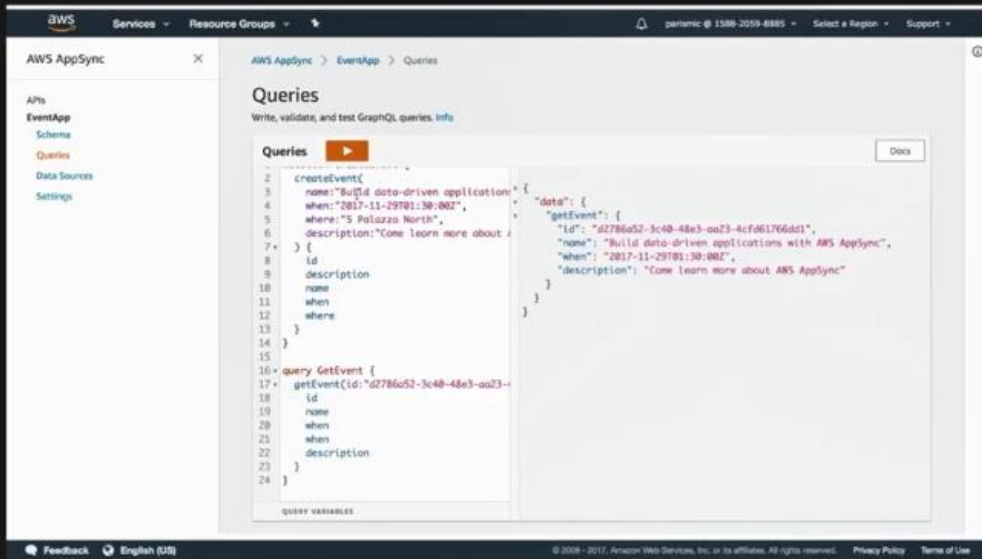


# DEPLOY AN API

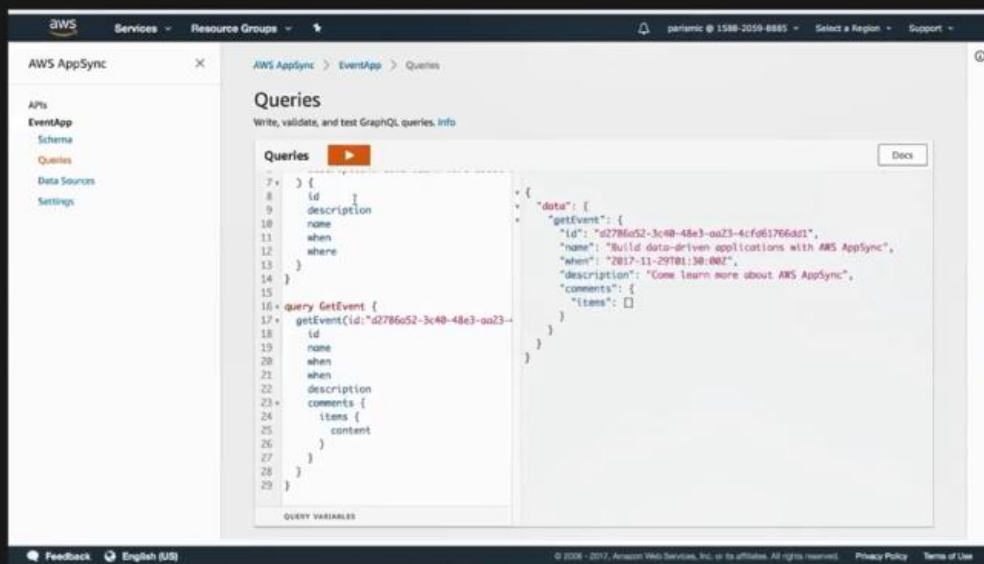


We can now write GraphQL queries against the data in DynamoDB

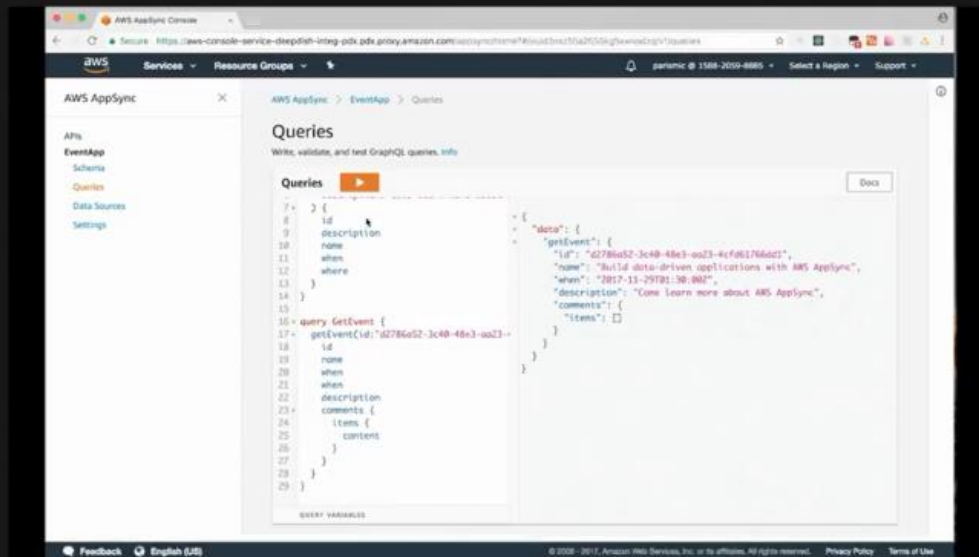
# DEPLOY AN API



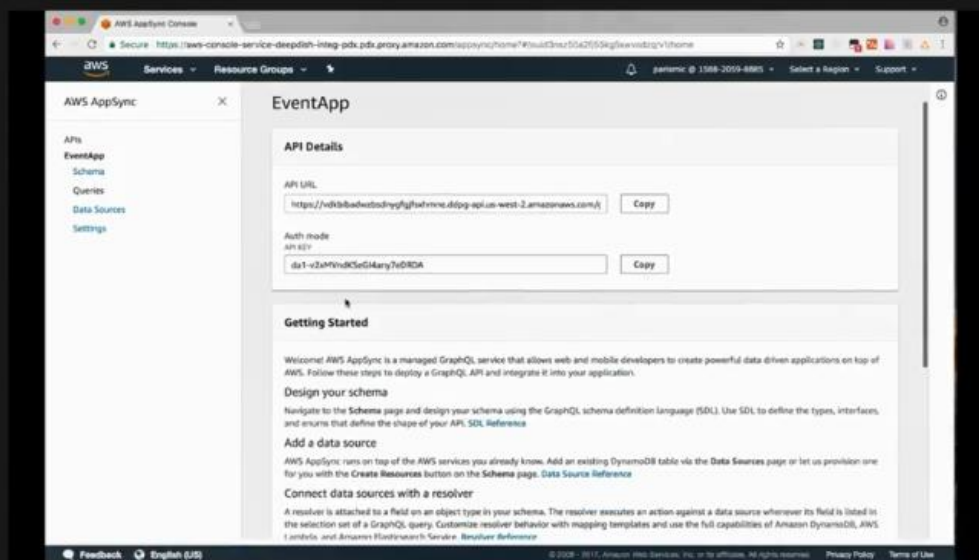
# DEPLOY AN API



# INTEGRATE WITH YOUR APPLICATION

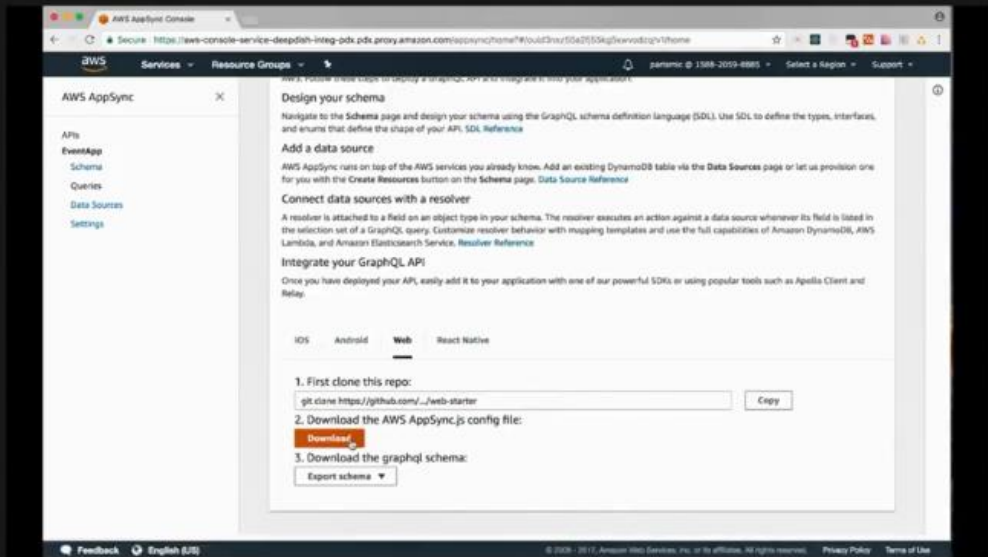


# INTEGRATE WITH YOUR APPLICATION





# INTEGRATE WITH YOUR APPLICATION

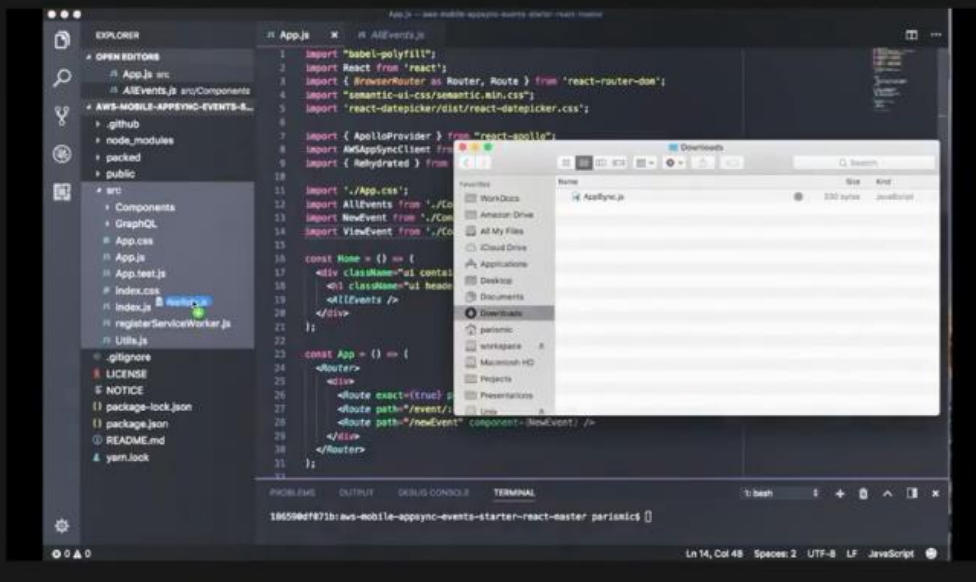


The screenshot displays the AWS AppSync console interface. On the left, a sidebar menu shows the navigation options: APIs, EventApp, Schema, Queries, Data Sources, and Settings. The main content area is titled 'Design your schema' and provides a step-by-step guide. It explains that the Schema page is used to define types, interfaces, and enums using GraphQL SDL. It also covers adding data sources (DynamoDB or other AWS services) and connecting them with resolvers. The 'Integrate your GraphQL API' section is highlighted, showing options for iOS, Android, Web (selected), and React Native. Below this, there are instructions to clone a repository, download the AWS AppSync.js config file (with a 'Download' button), and download the GraphQL schema (with an 'Export schema' button).

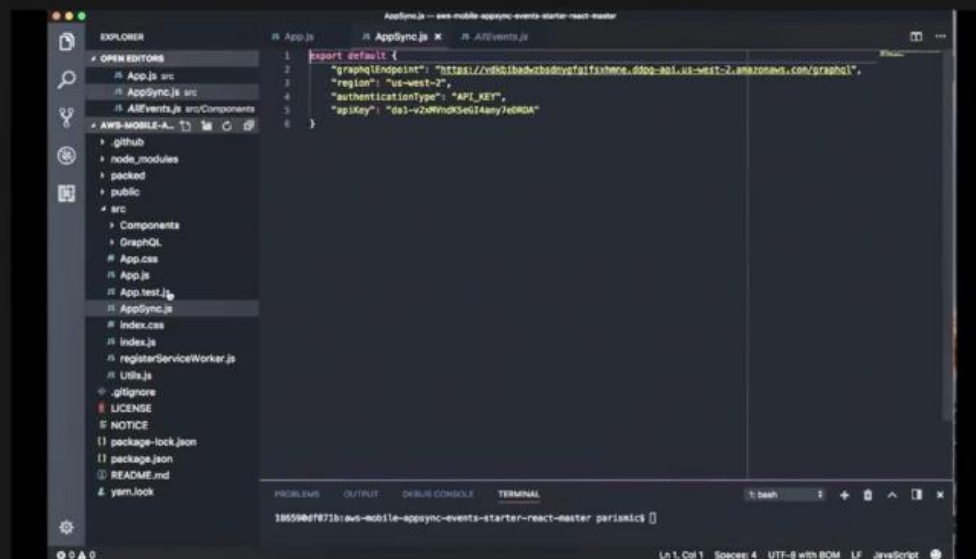
# INTEGRATE WITH YOUR APPLICATION

The screenshot illustrates the integration of AWS Mobile AppSync events into a React application. The Explorer sidebar on the left shows the project structure, including files like `App.js`, `AllEvents.js`, and `AppSyncEvents.js`. The main editor displays the `App.js` file, which imports necessary libraries and defines the `App` component. The `App` component uses `ApolloProvider` and `AWSAppSyncClient` to connect to the AppSync service. The `render` method returns a `Router` with routes for `/`, `/event/:id`, and `/newEvent`. The terminal at the bottom shows the command `npm install` being executed.

# INTEGRATE WITH YOUR APPLICATION

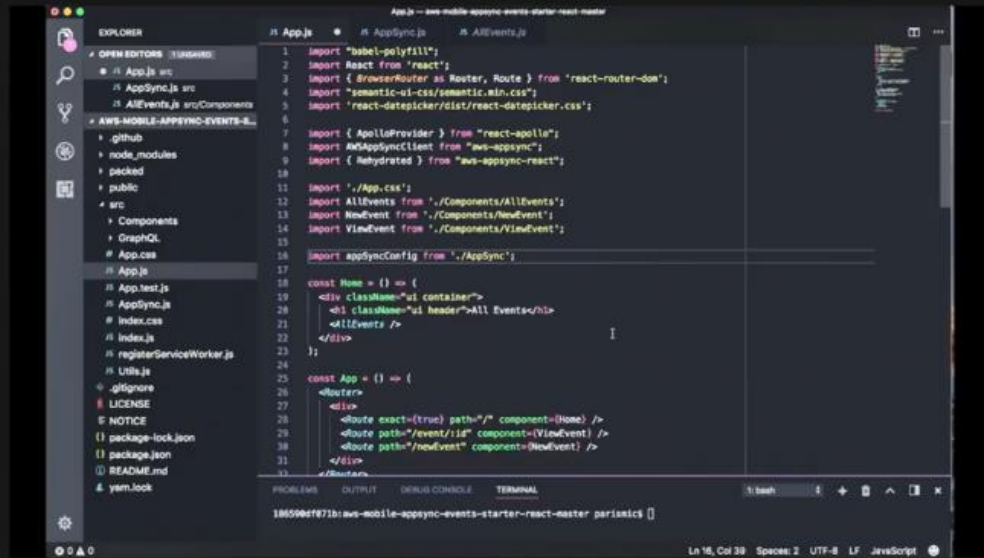


# INTEGRATE WITH YOUR APPLICATION



This is the configuration file that we can use on our app to integrate with the GraphQL backend, it contains things like the URL of our GraphQL server, the entry endpoint, default authorization mode is API Key

# INTEGRATE WITH YOUR APPLICATION



```
1 import "babel-polyfill";
2 import React from "react";
3 import { BrowserRouter as Router, Route } from "react-router-dom";
4 import "semantic-ui-css/semantic.min.css";
5 import "react-datepicker/dist/react-datepicker.css";
6
7 import { ApolloProvider } from "react-apollo";
8 import AWSAppSyncClient from "aws-appsync";
9 import { Rehydrated } from "aws-appsync-react";
10
11 import './App.css';
12 import AllEvents from './Components/AllEvents';
13 import NewEvent from './Components/NewEvent';
14 import ViewEvent from './Components/ViewEvent';
15
16 import appSyncConfig from './AppSync';
17
18 const Home = () => {
19   <div className="ui container">
20     <h1 className="ui header">All Events</h1>
21     <AllEvents />
22   </div>
23 };
24
25 const App = () => {
26   <Router>
27     <div>
28       <Route exact={true} path="/" component={Home} />
29       <Route path="/event/:id" component={ViewEvent} />
30       <Route path="/newevent" component={NewEvent} />
31     </div>
32   </Router>
33 };
34
35 export default App;
```

You just import it and start using as above. Do an ***npm install*** and ***npm start***

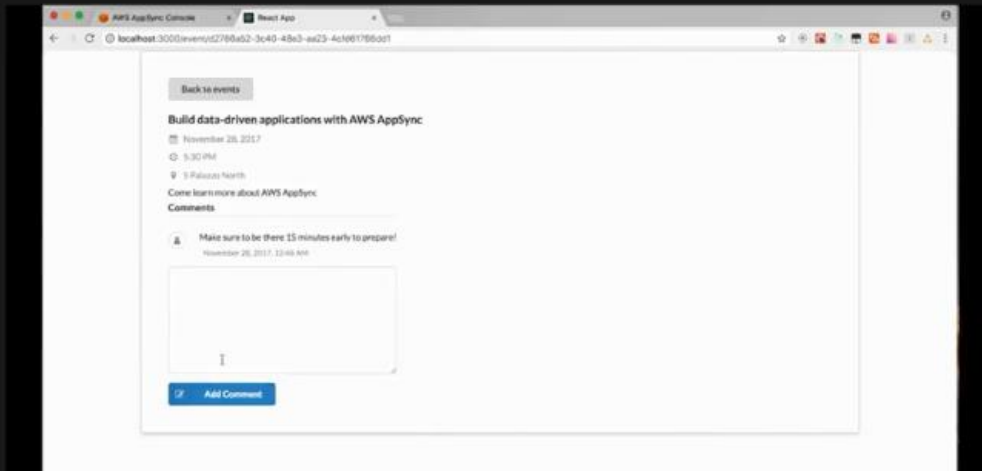
# INTEGRATE WITH YOUR APPLICATION



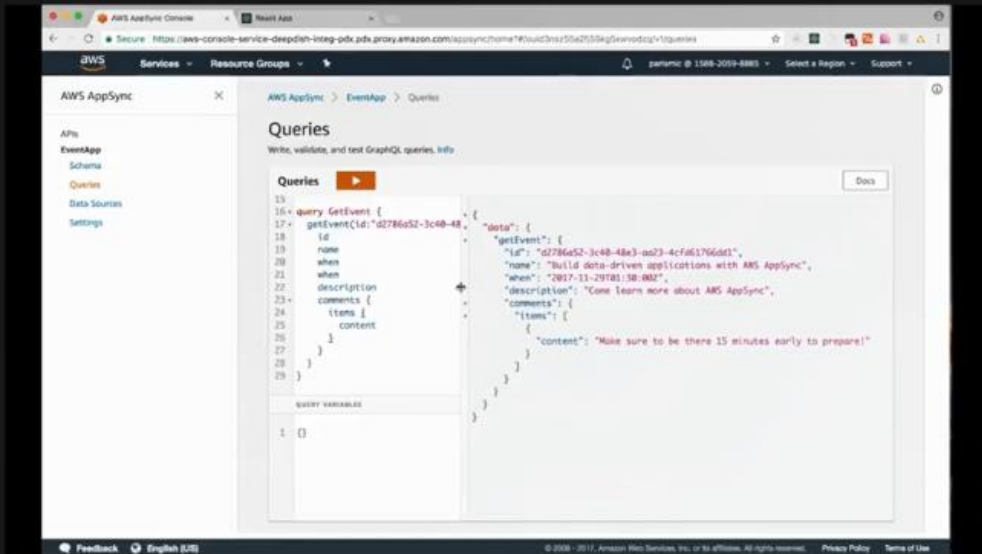
# INTEGRATE WITH YOUR APPLICATION



# INTEGRATE WITH YOUR APPLICATION



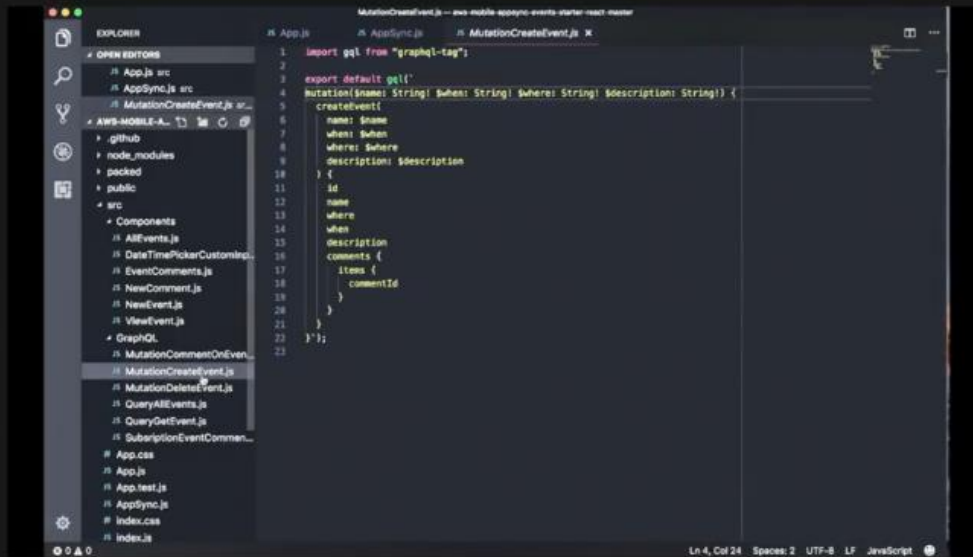
# INTEGRATE WITH YOUR APPLICATION



We now can see the newly added comment



# INTEGRATE WITH YOUR APPLICATION



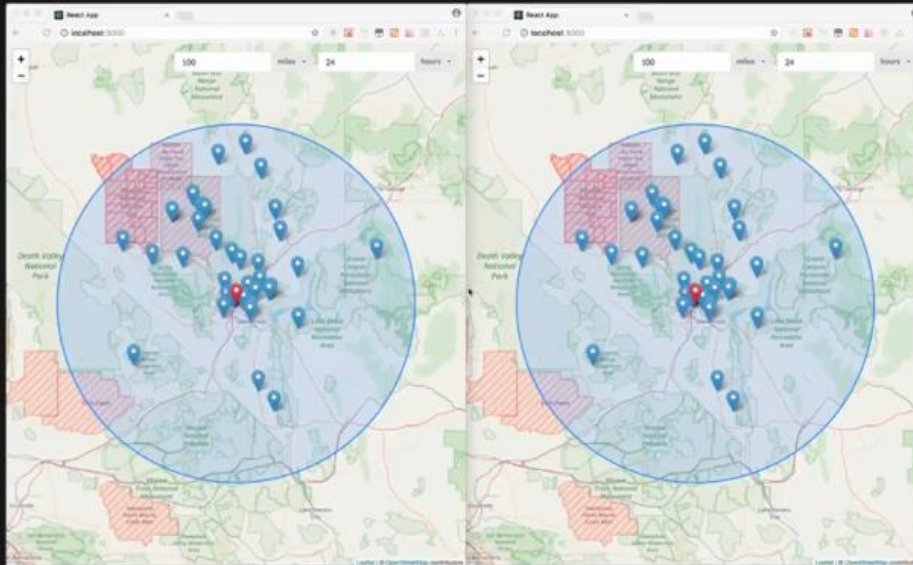
```
1 import gql from "graphql-tag";
2
3 export default gql`
4   mutation($name: String! $when: String! $where: String! $description: String!) {
5     createEvent(
6       name: $name
7       when: $when
8       where: $where
9       description: $description
10     ) {
11       id
12       name
13       where
14       when
15       description
16       comments {
17         items {
18           commentId
19         }
20       }
21     }
22   }
23 `;
```

## RECAP

- Deployed two Amazon DynamoDB tables to hold Event and Comment records
- Created two AWS IAM service roles giving AWS AppSync access to data operations on those tables
- Deployed a scalable GraphQL API
- Ran a single GraphQL query that interacted with data from multiple Amazon DynamoDB tables
- Integrated AWS AppSync with our front-end application and enabled offline and real-time support

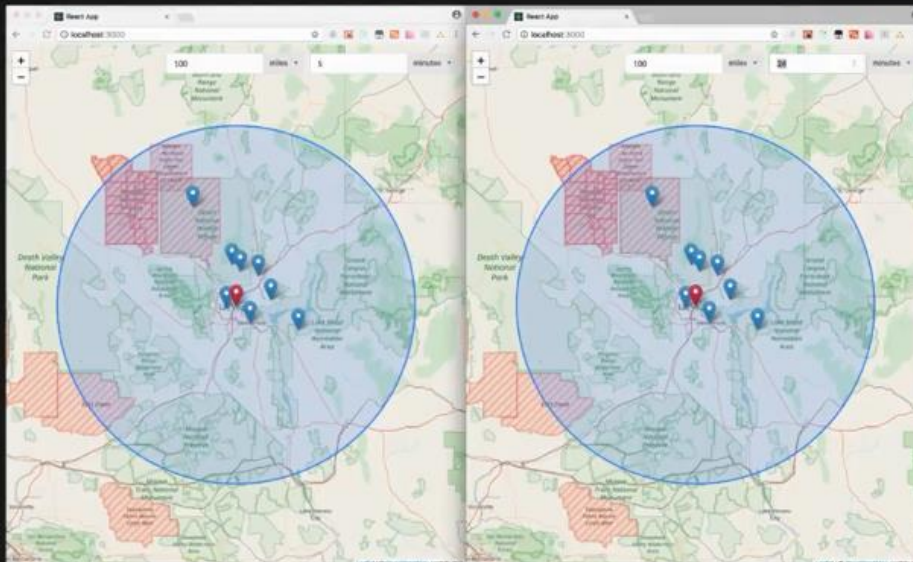
**AWS AppSync helps you build global scale and powerful applications**

## GOING DEEPER

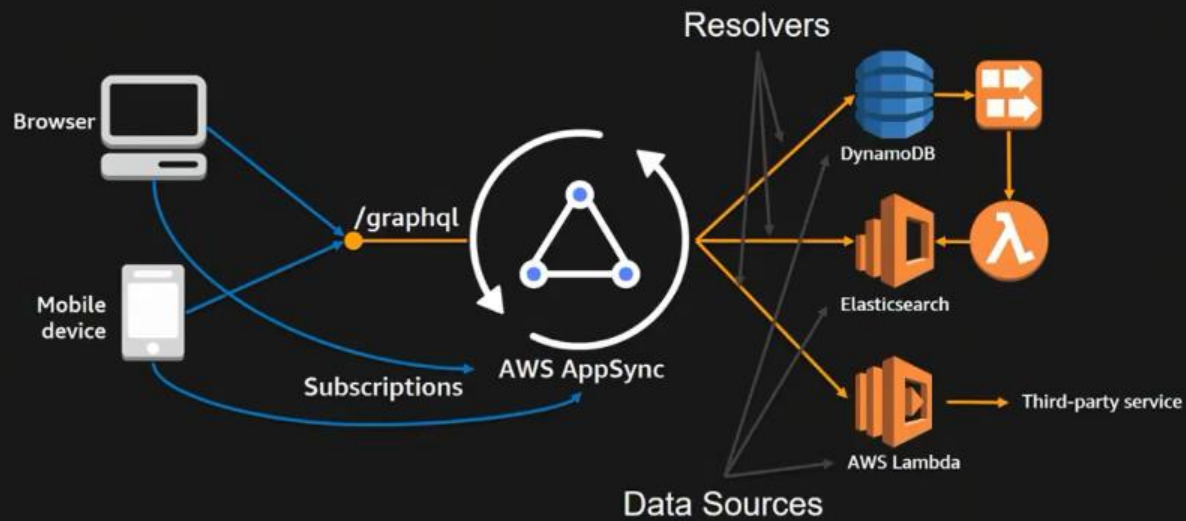


We have an AWS AppSync app that is integrated with DynamoDB, Elasticsearch and a Lambda function that calls out to a 3<sup>rd</sup> party service to find out if a payment we submitted has been processed or not. The page above is been satisfied with 1 single query that gets resent when we move the center cursor on the screen and redraws the results on the screen.

## GOING DEEPER



# MAPTAP WITH AWS APPSYNC



aws  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



## Musixmatch

FRANCESCO DELFINO

CTO & Co-Founder, Musixmatch

### A new way to enjoy music

Founded in 2010. Head quarters in Bologna, Italy

Lyrics API available to third-party music services and websites

Mobile apps for Android and iOS

Desktop widget (Floating Lyrics) running on Windows, Mac, and Linux



## World's largest lyrics catalog



Worldwide coverage with more than 60 different languages



67,11%



6,15%



5,7%



2,65%



2,62%



2,39%



2,36%



1,84%



0,81%



0,72%



0,68%



0,67%



0,61%



0,52%



0,51%

...

## SYNCED LYRICS

Available for over 90% of the lyrics requested

## TRANSLATIONS

Available for over 70% of the lyrics requested

## WORD BY WORD SYNC

Available for in beta. GA Q1 2018



**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.





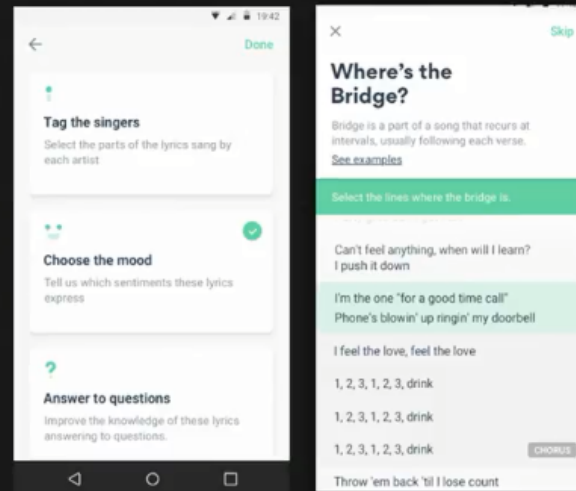
# Understanding the music consumption



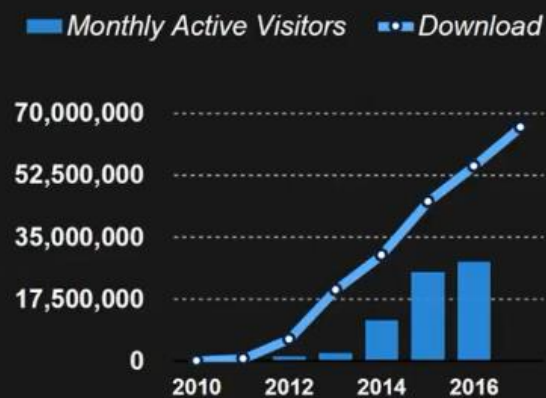
Musixmatch plugs in 3rd party music players  
(Spotify, iTunes, Youtube etc).

Over a billion data points processed daily, profile user  
musical preferences

Semantic analysis and structural analysis for tailored  
recommendation (Q1 2018)



## 30 million active users per month



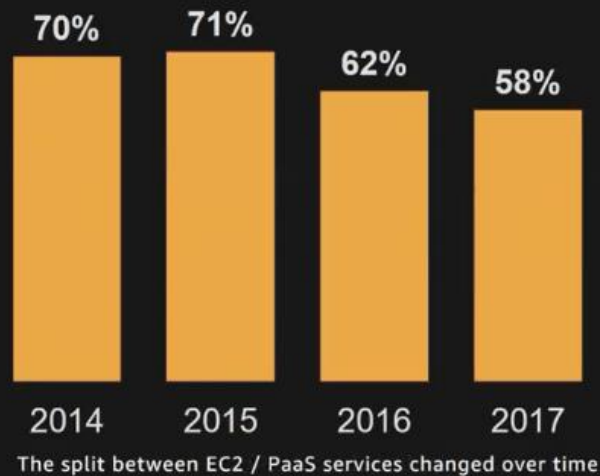
## Over 400k requests per minute, low at 200k



Total spending lowered over time

10x increase in the same period (sessions)

Lessons learned: don't be scared by continuously re-engineering your product to exploit new possibilities



## Simplifying our app development and backend

### Multidevice development: React Native



#### PROS

Vibrant ecosystem

Development tools, e.g. Expo

Leverage web skills

Easily pluggable in existing apps

#### CONS

Too many options

Versioning problems

Development tools sometimes difficult to tweak

Our stack: Typescript, ReactXP, NGROK (for creating our own Expo-like experience)

## Enabling multidevice experience 1/2



Most users move from mobile to desktop for editing the lyrics

We already support interprocess communication (Floating lyrics, deeplinks)

### What we needed:

Extending to multiple device requires a real-time messaging hub

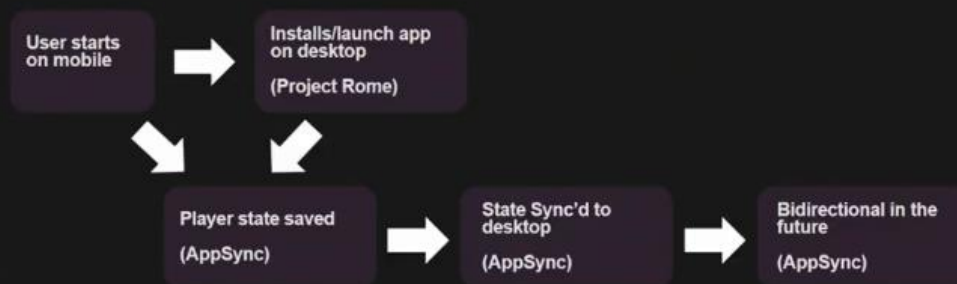
### What we tried:

Open Source tools: SignalR or Socket.IO

### Challenges:

Dealing with mobile connection reliability is hard

## Enabling multidevice experience 2/2

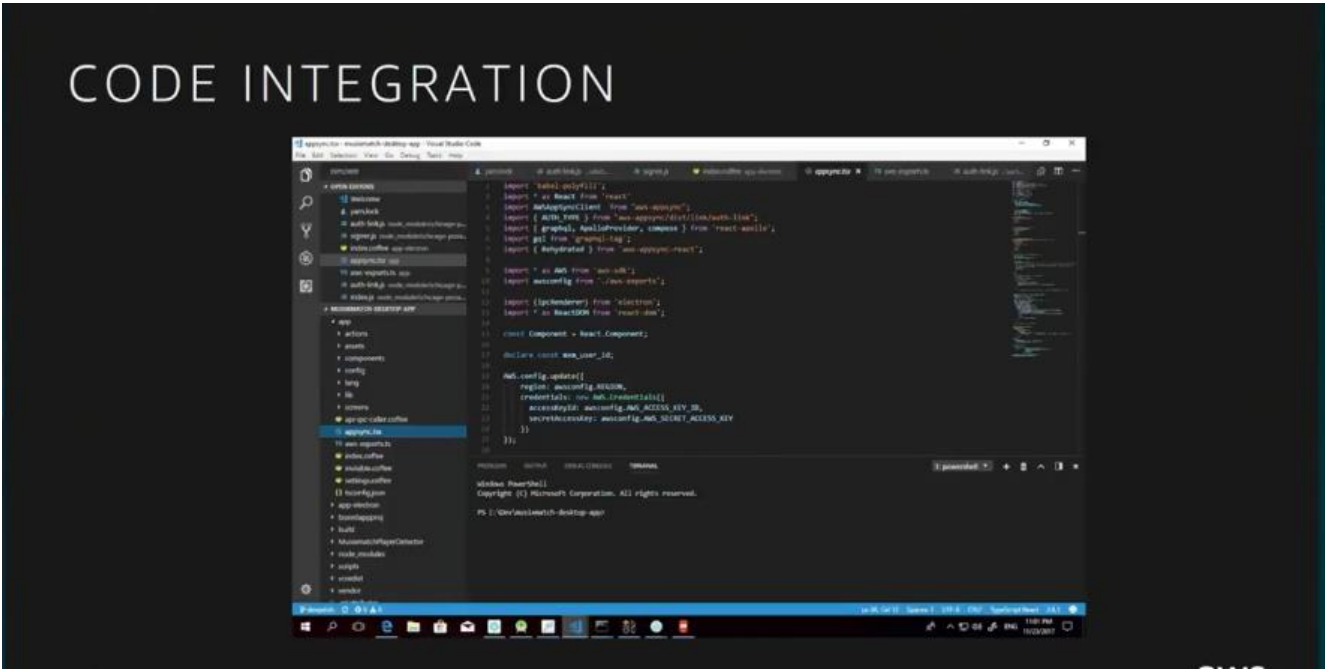


```
Type UserNowPlaying {  
  user_id: ID! UserPlayerState  
  guid?: String!  
  protocolUri: String!  
}
```

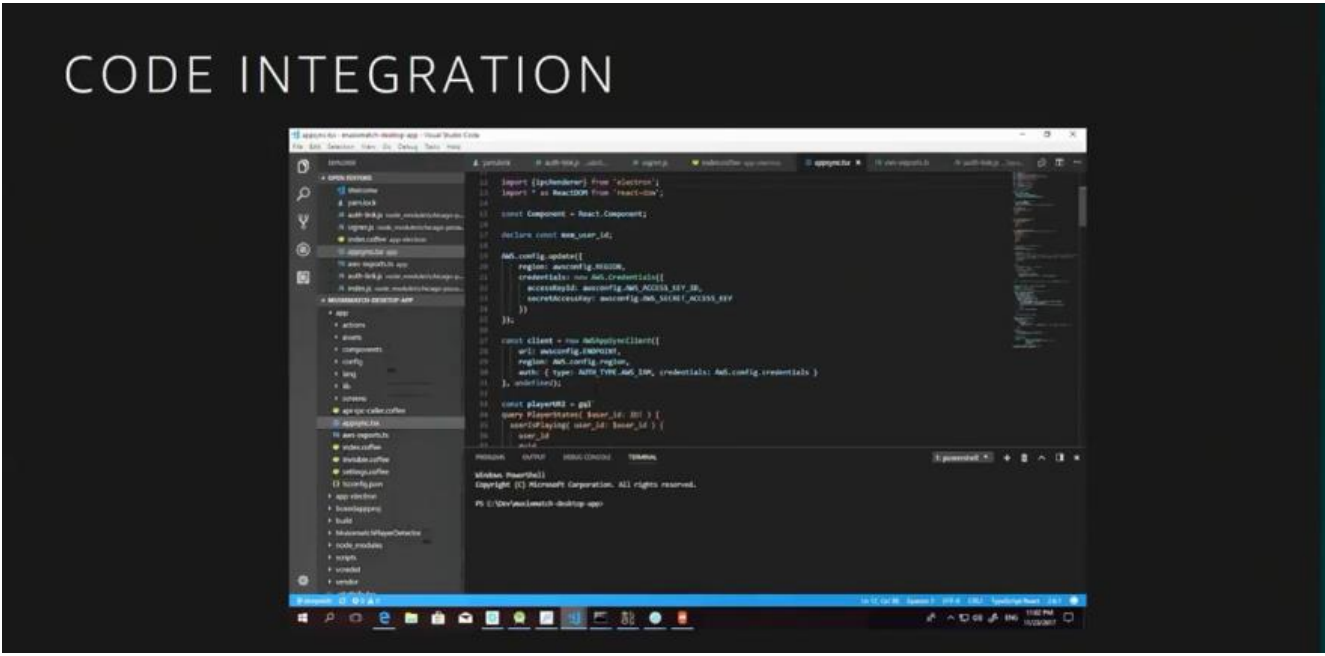
The custom protocol  
URI contains

```
q_track  
q_artist  
duration  
position  
start_time
```

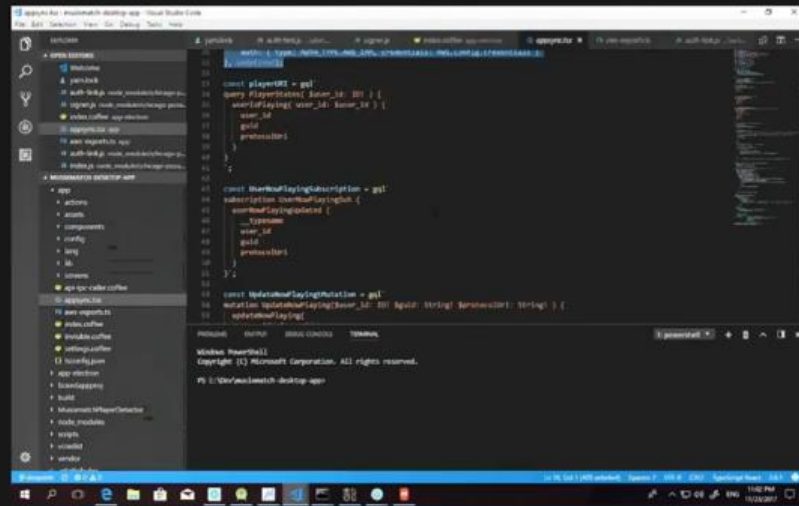
# CODE INTEGRATION



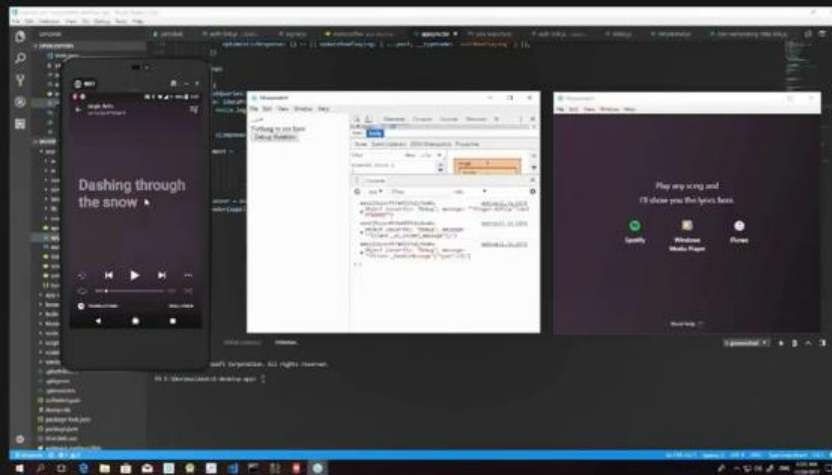
# CODE INTEGRATION



# CODE INTEGRATION

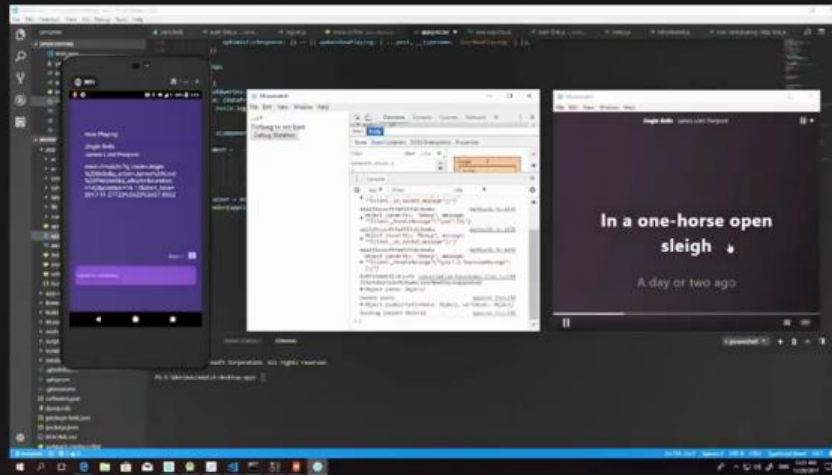


# APP DEMO

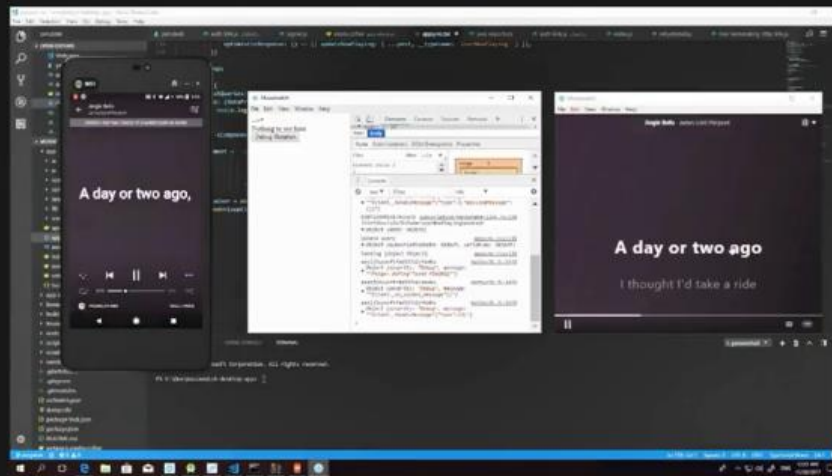




# APP DEMO



# APP DEMO



## Wrapping Up

AMIT PATEL  
General Manager, AWS Mobile

# START BUILDING TODAY!

AWS Mobile Hub and CLI available now

- Learn more and get started: <https://aws.amazon.com/mobile/>

AWS Amplify is also available now

- Find it on GitHub: <https://github.com/aws/aws-amplify>

AWS AppSync Preview available for registration later today

- Learn more and register: <https://aws.amazon.com/appsync/>

## AWS re:Invent

### Thank you!

1. MBL310 (Tues @ 12:15 p.m.)—Building Hybrid and Web apps using JavaScript with AWS Mobile
2. MBL404 (Tues @ 1:45 p.m.)—Real-time and Offline application development using GraphQL with AWS AppSync
3. MBL402 (Tues @ 5:30 p.m.)—Data Driven Apps with GraphQL: AWS AppSync Deep Dive

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

