# AWS re:INVENT

## Mobile State of the Union

AMIT PATEL
General Manager, AWS Mobile

November 28, 2017

AWS re:Invent



aws

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

# 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

# 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



aws

# 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-e
5
6  Amplify.configure(awsExports);
7  I18n.setLanguage("fr");
8
9  class App extends React.Compor
10   render() {
11     return (
12       <View style={styles.con
13         <Text>Open up App.js
14         <Text>Changes you make
15         <Text>Shake your phone
16       </View>
17     );
18   }
```


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

AWS re:Invent



aws

# TESTING AND DEPLOYING YOUR WEB APP

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

Test your web app on real devices for performance insights

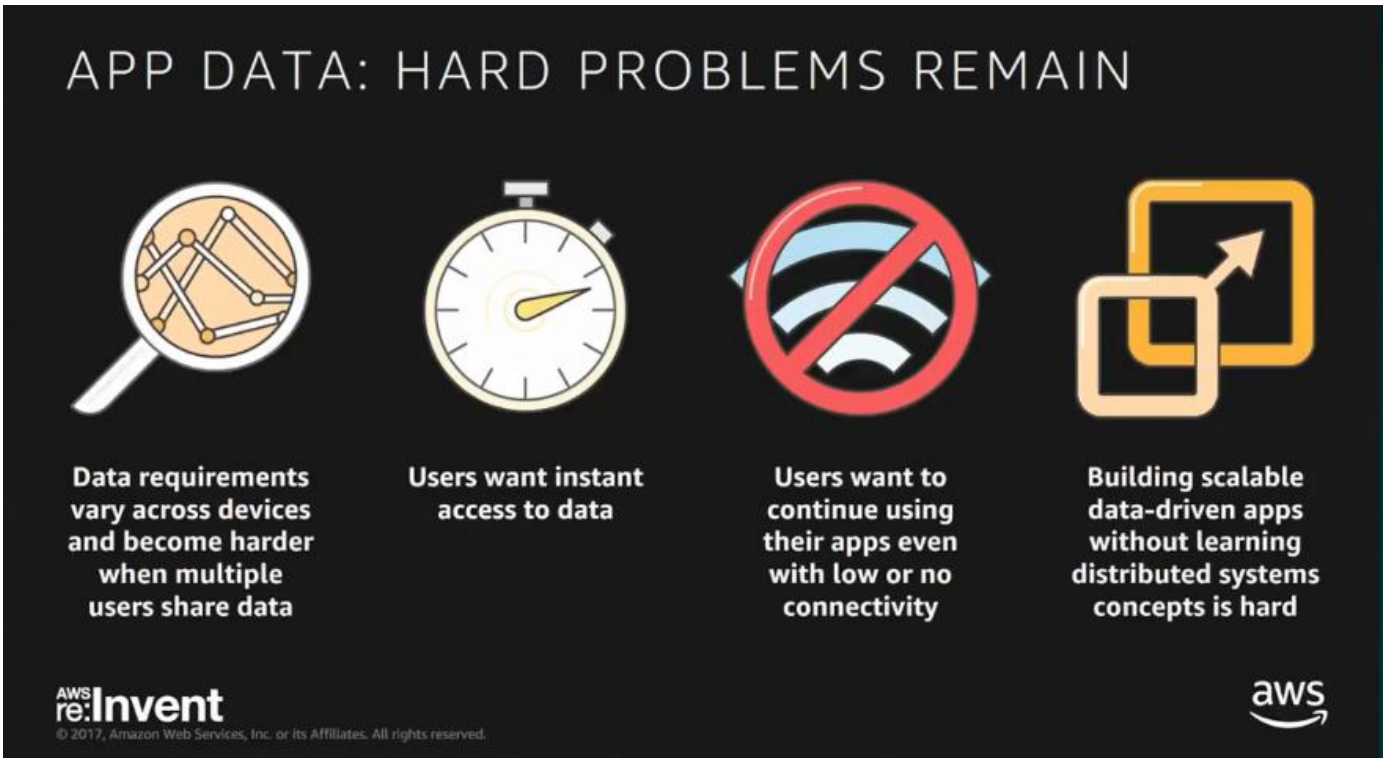
Host your web app on S3 (served through CloudFront)




# THE MOBILE HUB CONSOLE

Use the Mobile Hub console to add more platforms and features

```
> awsmobile console
```





***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



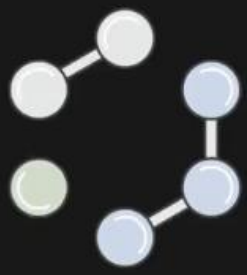
aws

# 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



aws

# START BUILDING FAST


AWS re:Invent



aws

# 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



aws

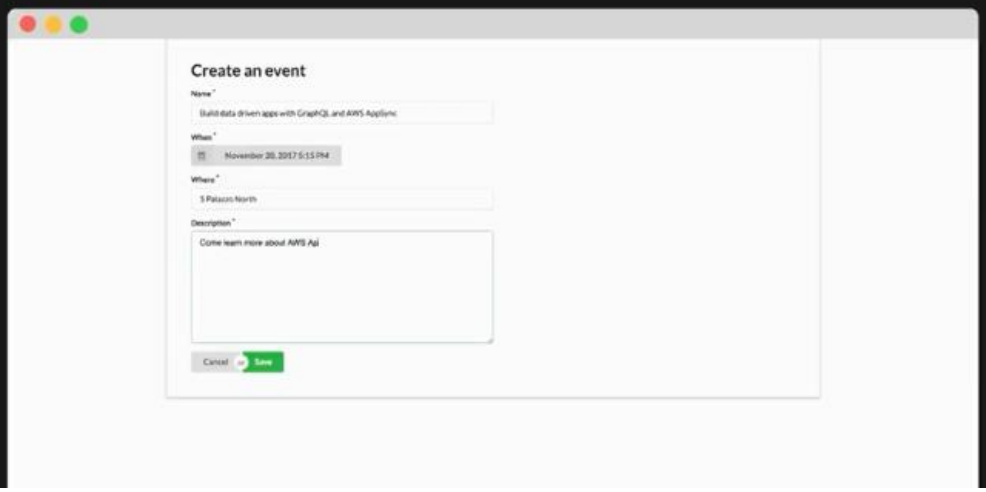
# Getting Started with AWS AppSync

MICHAEL PARIS

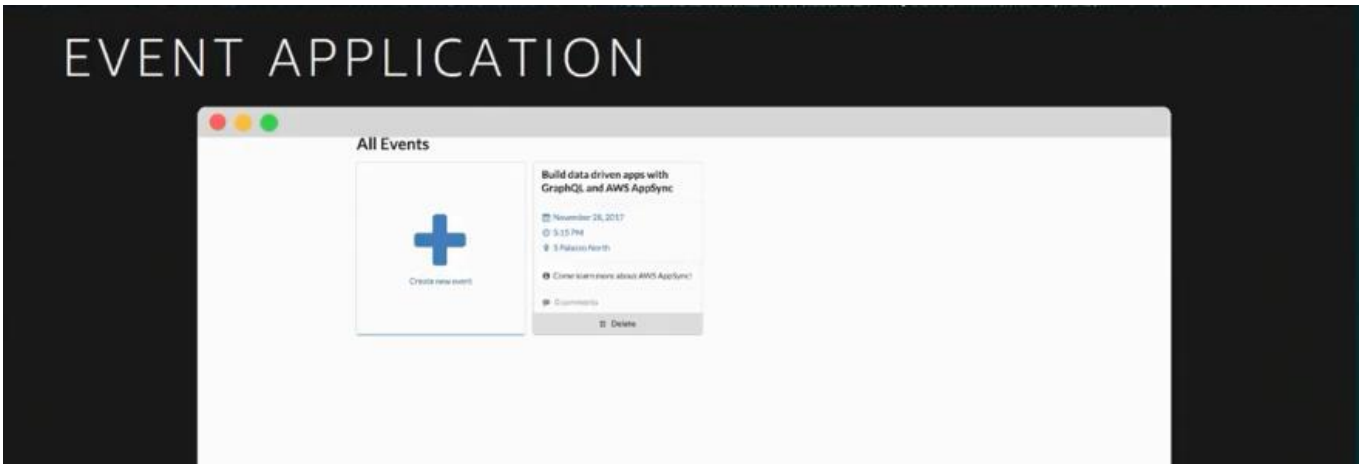
Software Development Engineer, AWS Mobile

# EVENT APPLICATION

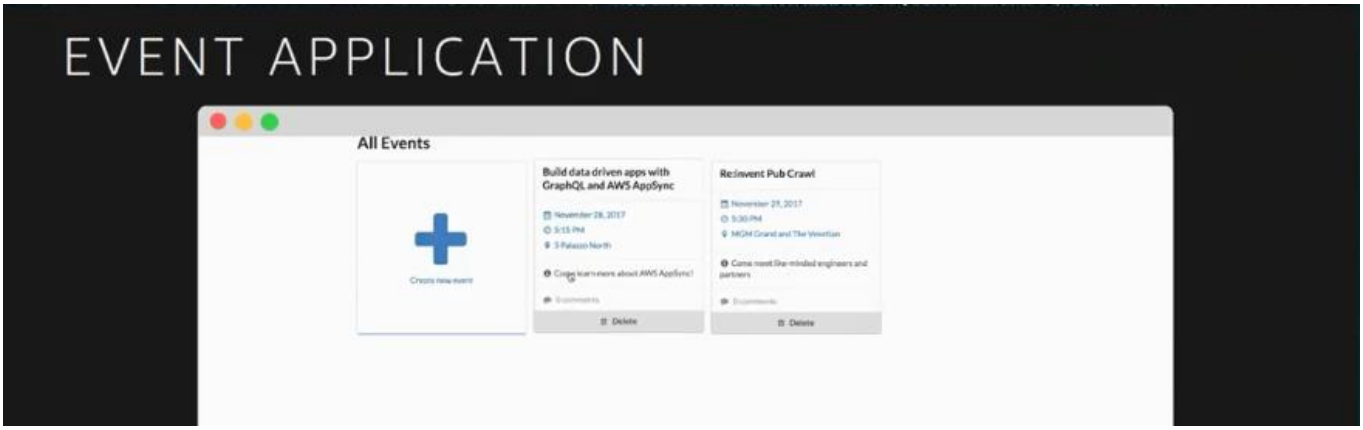

# EVENT APPLICATION

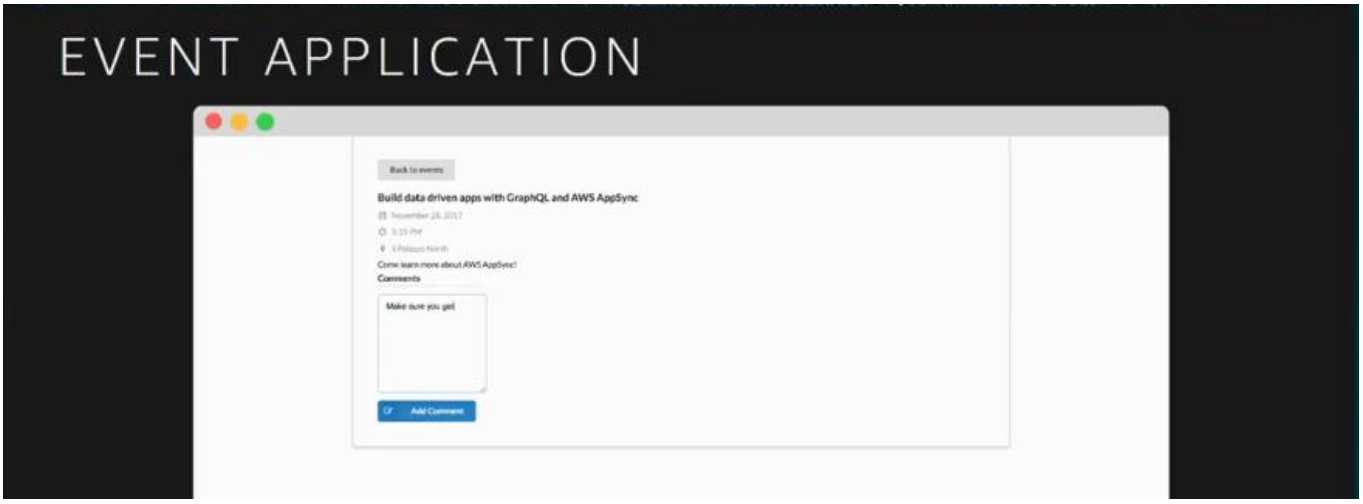



# EVENT APPLICATION

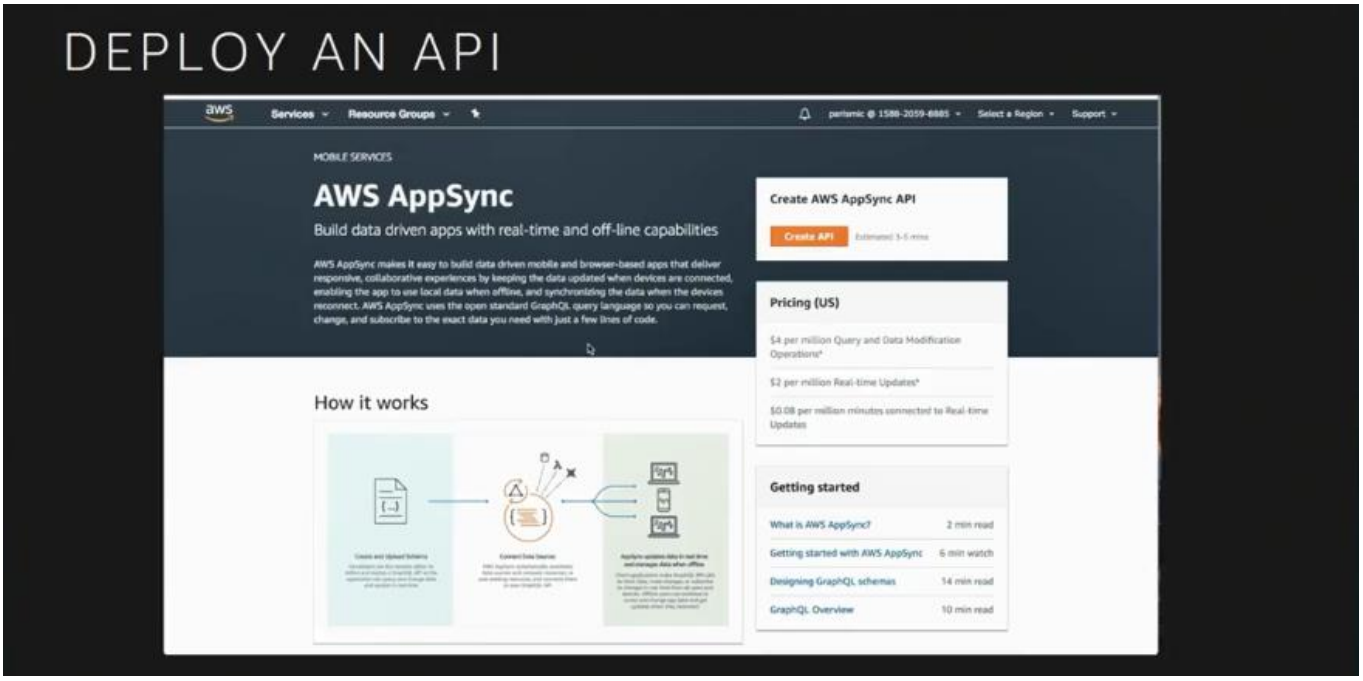



This app uses 2 types of models, an Event model and a Comment model.

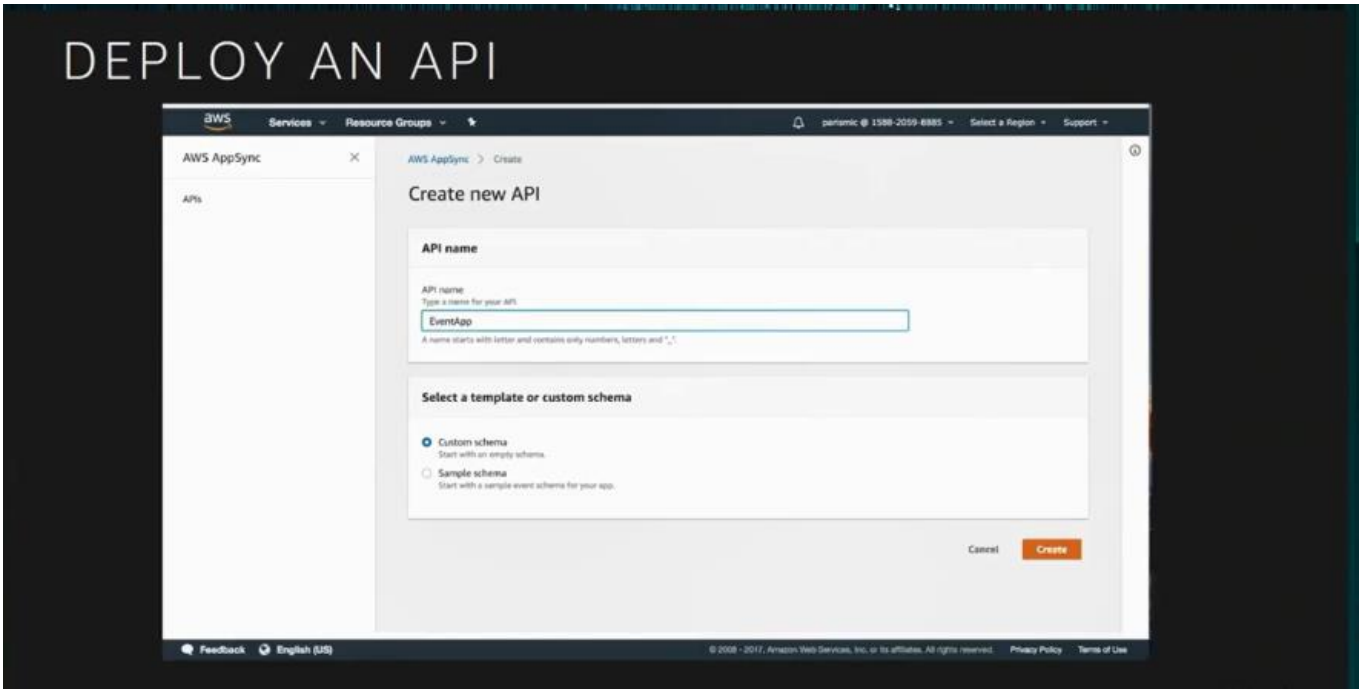



**AWS AppSync helps you build applications faster**




We want to create an API for our app



# DEPLOY AN API

aws Services Resource Groups pariamic @ 1588-2059-8885 Select a Region Support

AWS AppSync

APIs

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.

**Important!**
Each template will automatically deploy an api and provision DynamoDB tables and IAM roles on your behalf.

**Sample event schema**

```
1 schema {
2   query: Query
3   mutation: Mutation
4   subscription: Subscription
5 }
6
7 type Query {
8   # Get a single event by id.
9   getEvent(id: ID!): Event
10  # Paginate through events.
```

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

# DEPLOY AN API

aws Services Resource Groups pariamic @ 1588-2059-8885 Select a Region Support

AWS AppSync

APIs

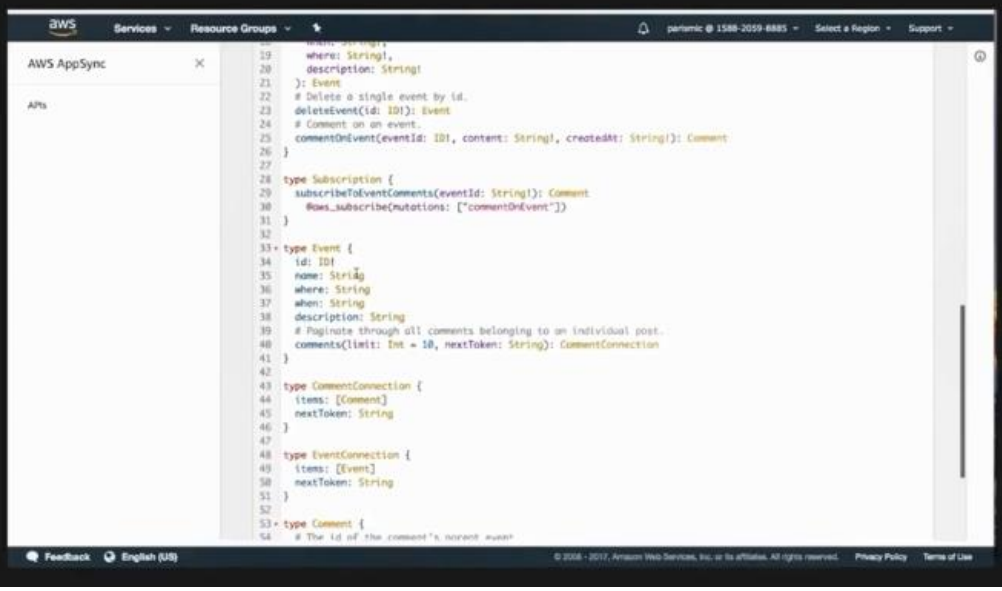
**Sample event schema**

```
1 schema {
2   query: Query
3   mutation: Mutation
4   subscription: Subscription
5 }
6
7 type Query {
8   # Get a single event by id.
9   getEvent(id: ID!): Event
10  # Paginate through events.
11  listEvents(limit: Int = 10, nextToken: String): EventConnection
12 }
13
14 type Mutation {
15   # Create a single event.
16   createEvent(
17     name: String!,
18     when: String!,
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
```

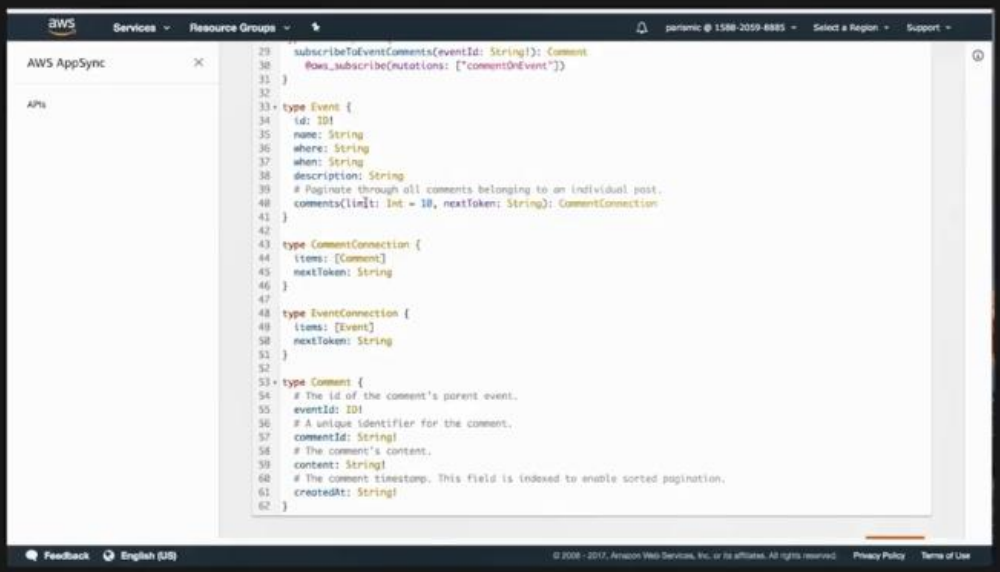
Feedback English (US) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

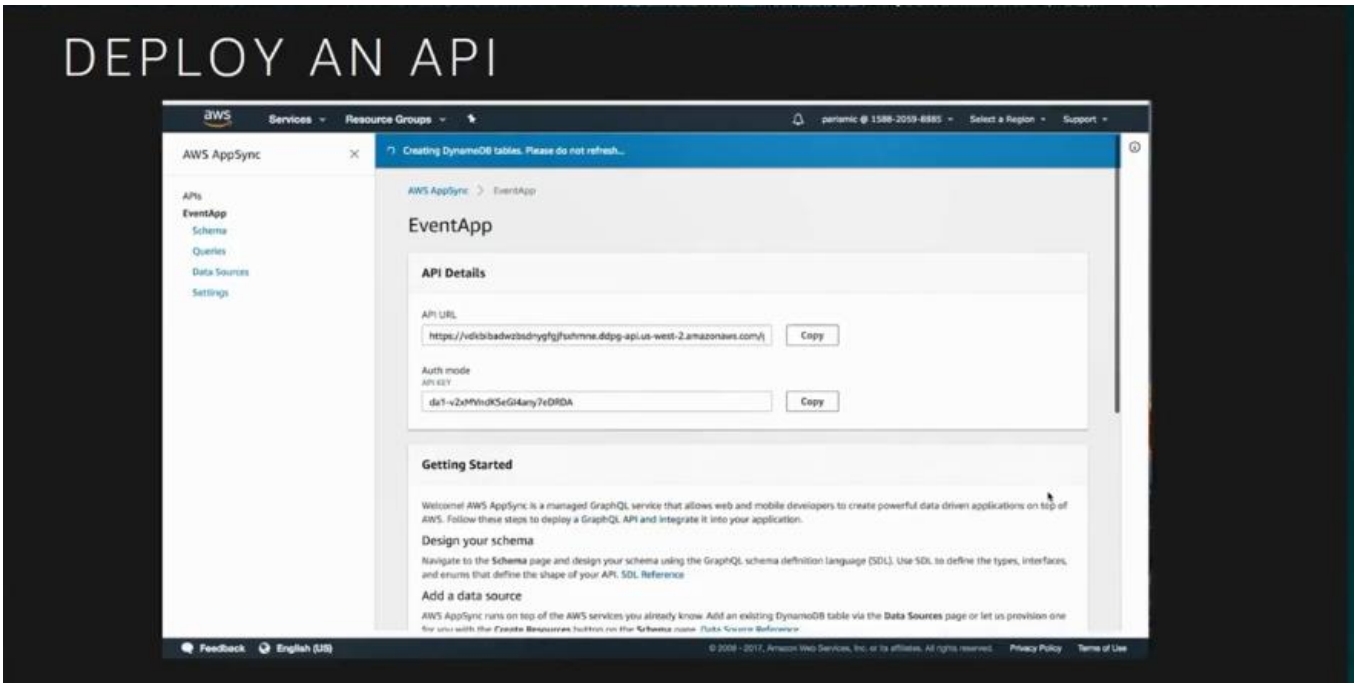
A schema in GraphQL defines the data model and Types, as well as the operations that you have access to within your API

# DEPLOY AN API

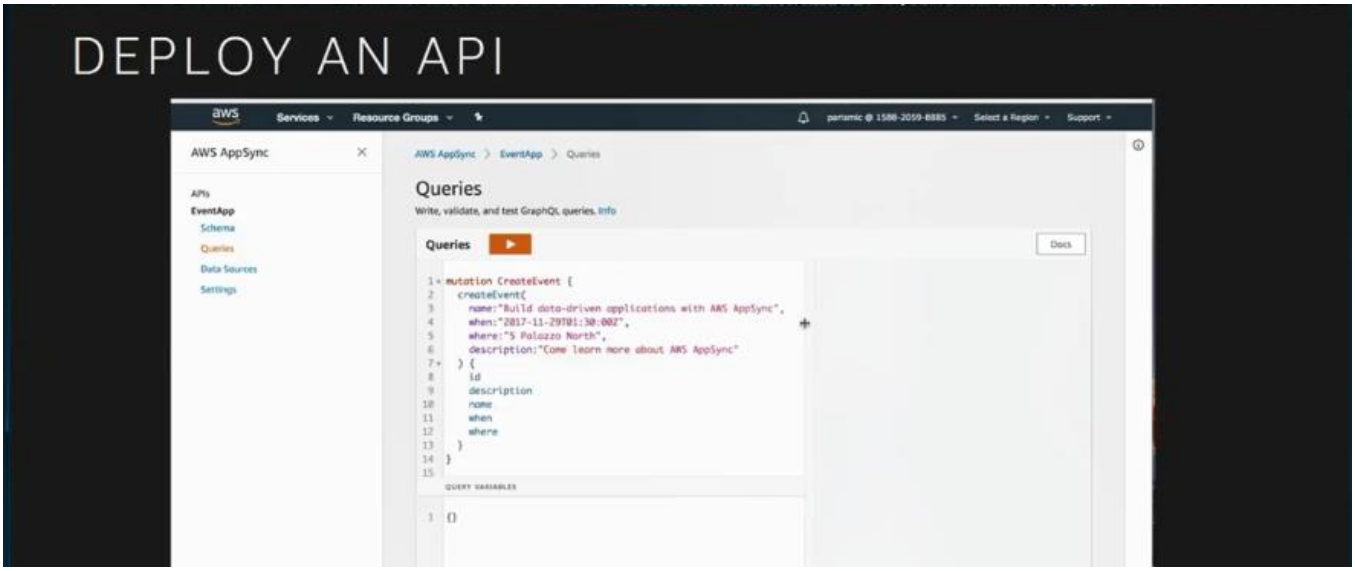



# DEPLOY AN API

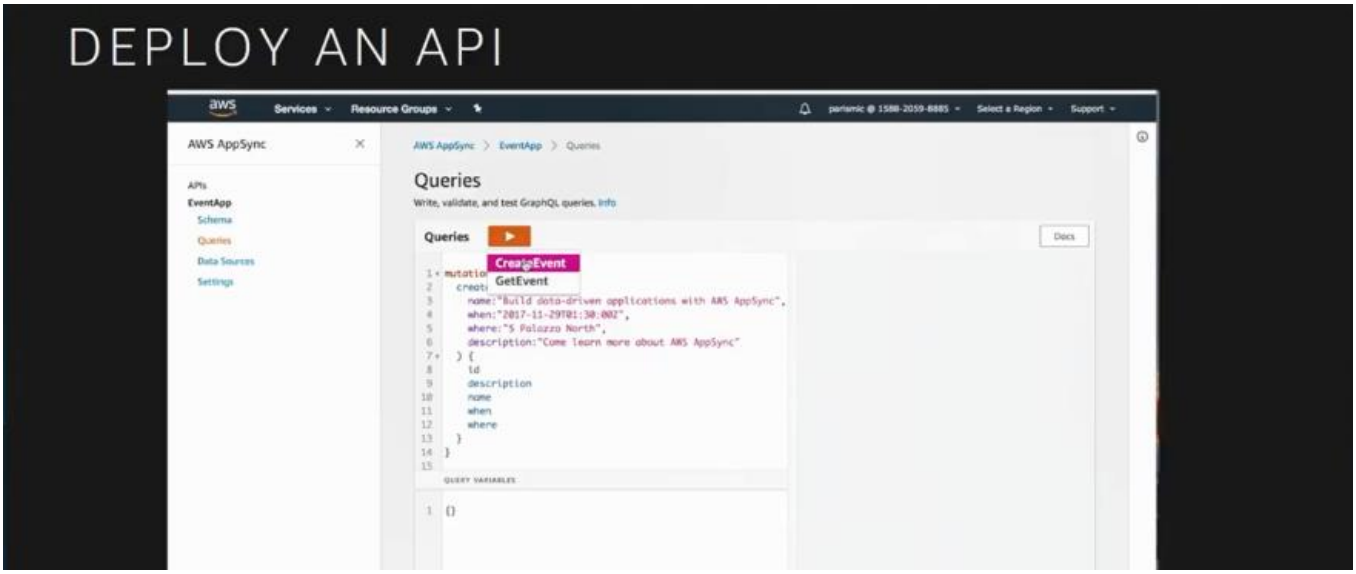


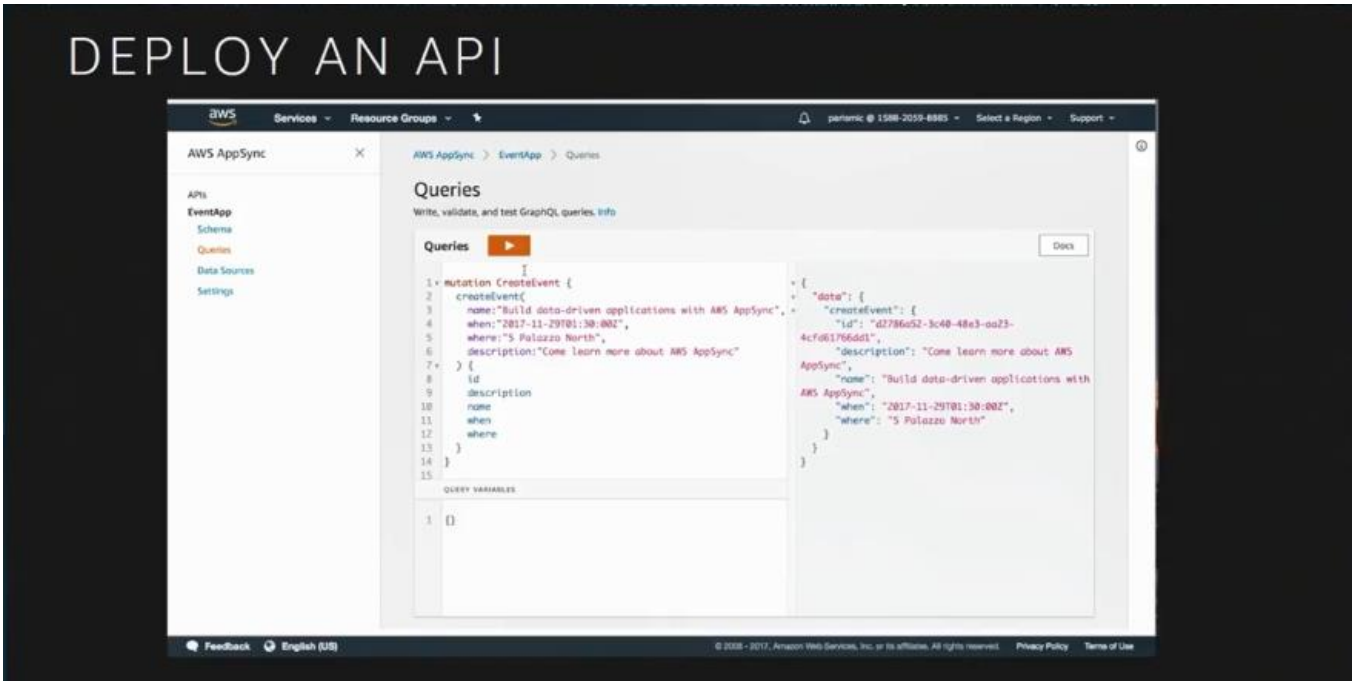



When you click the Create button with the sample schema, AppSync goes and does some things for you. It is going to provision DynamoDB tables in your account, IAM Roles that will give AWS permissions to access those database resources on your behalf,



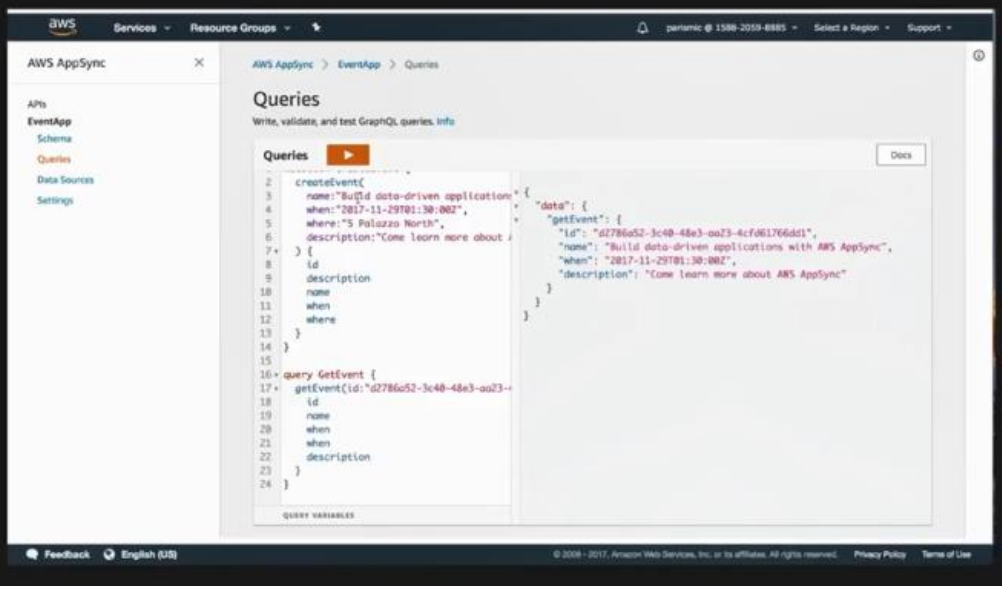

After the tables have been provisioned, the API is now live and ready to be used.



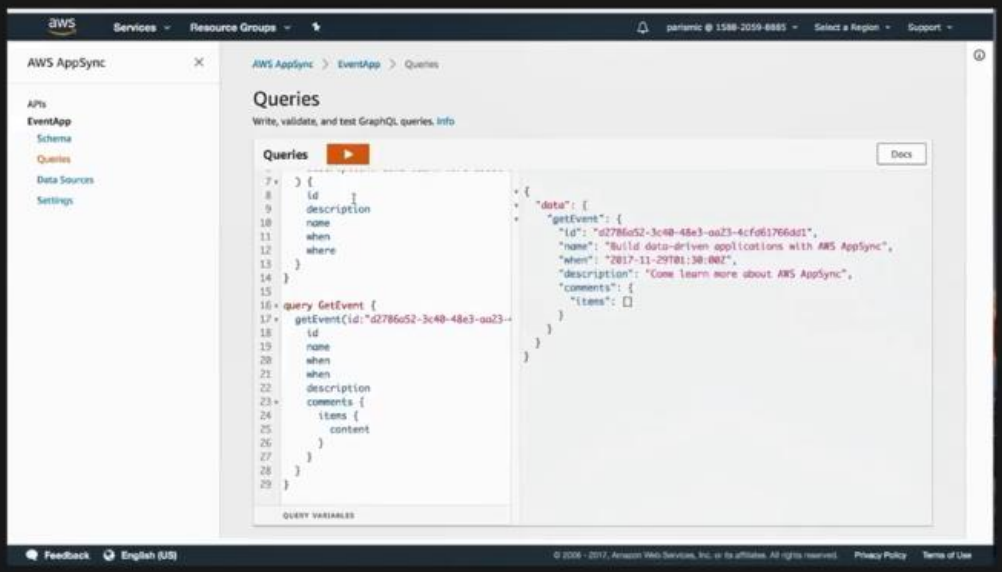




We can now write GraphQL queries against the data in DynamoDB

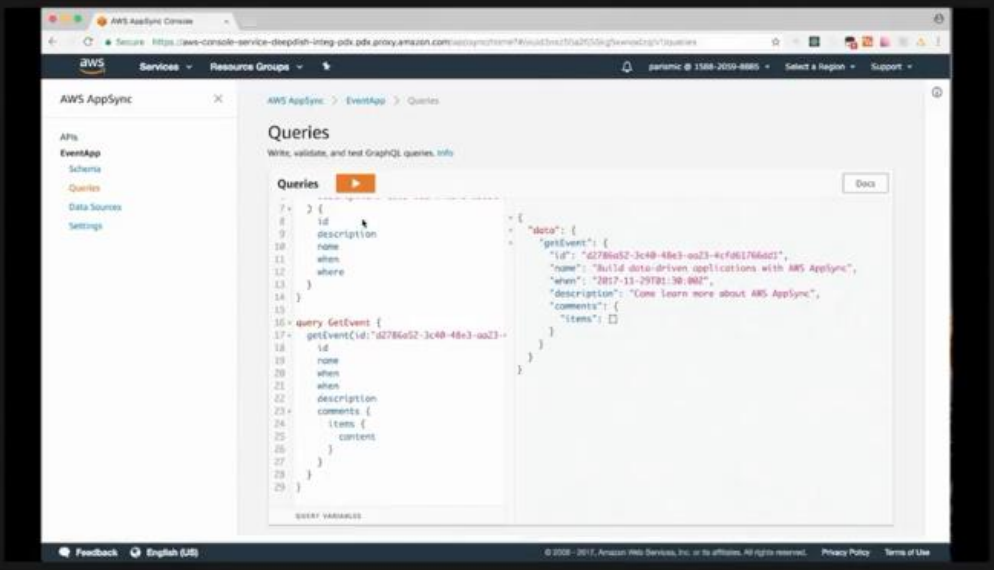
# DEPLOY AN API



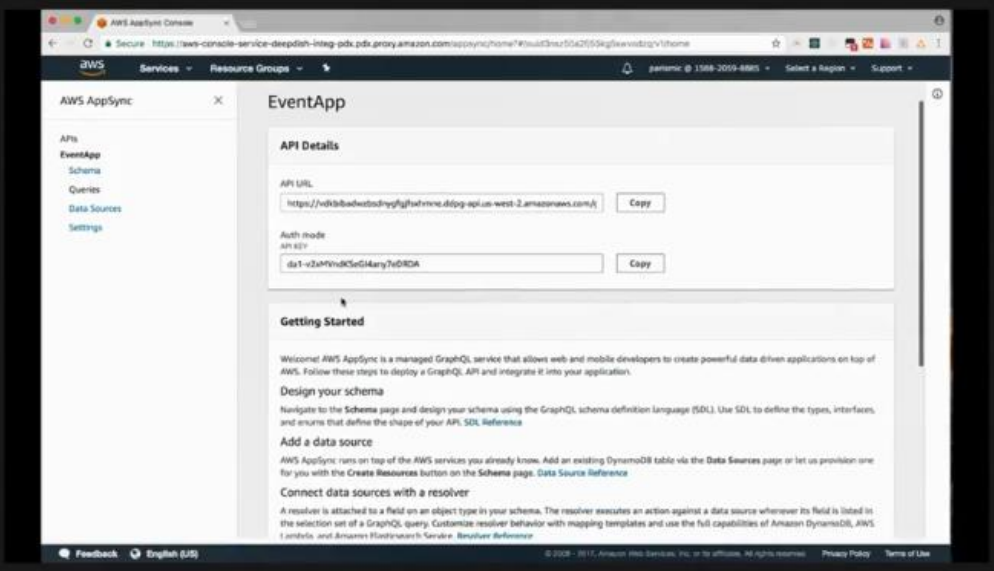

# DEPLOY AN API



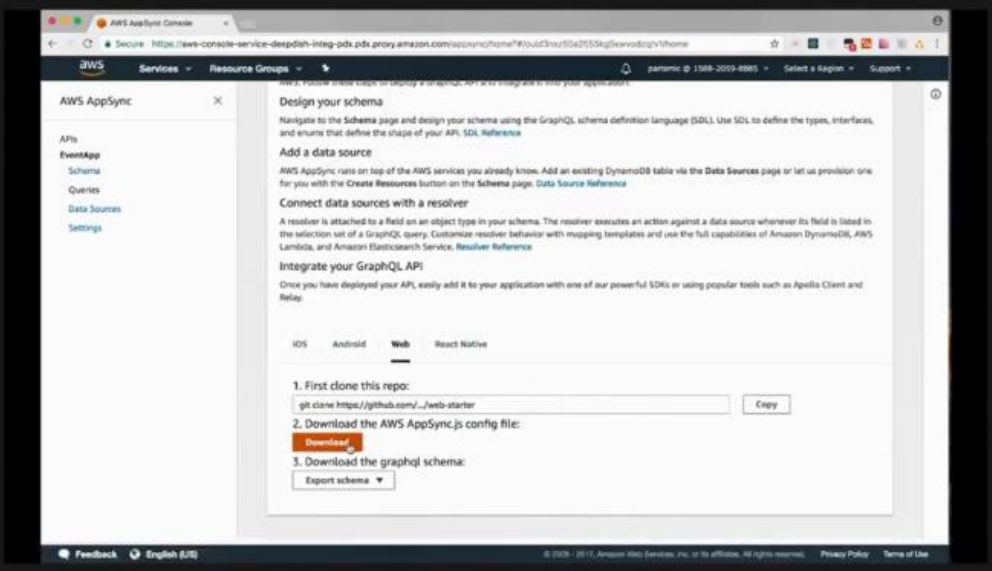
# INTEGRATE WITH YOUR APPLICATION



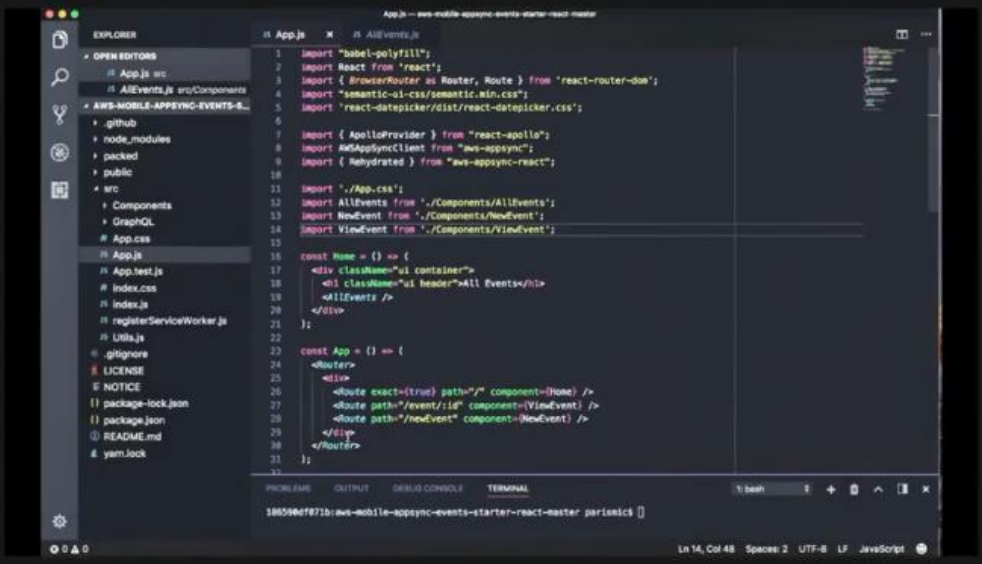

# INTEGRATE WITH YOUR APPLICATION

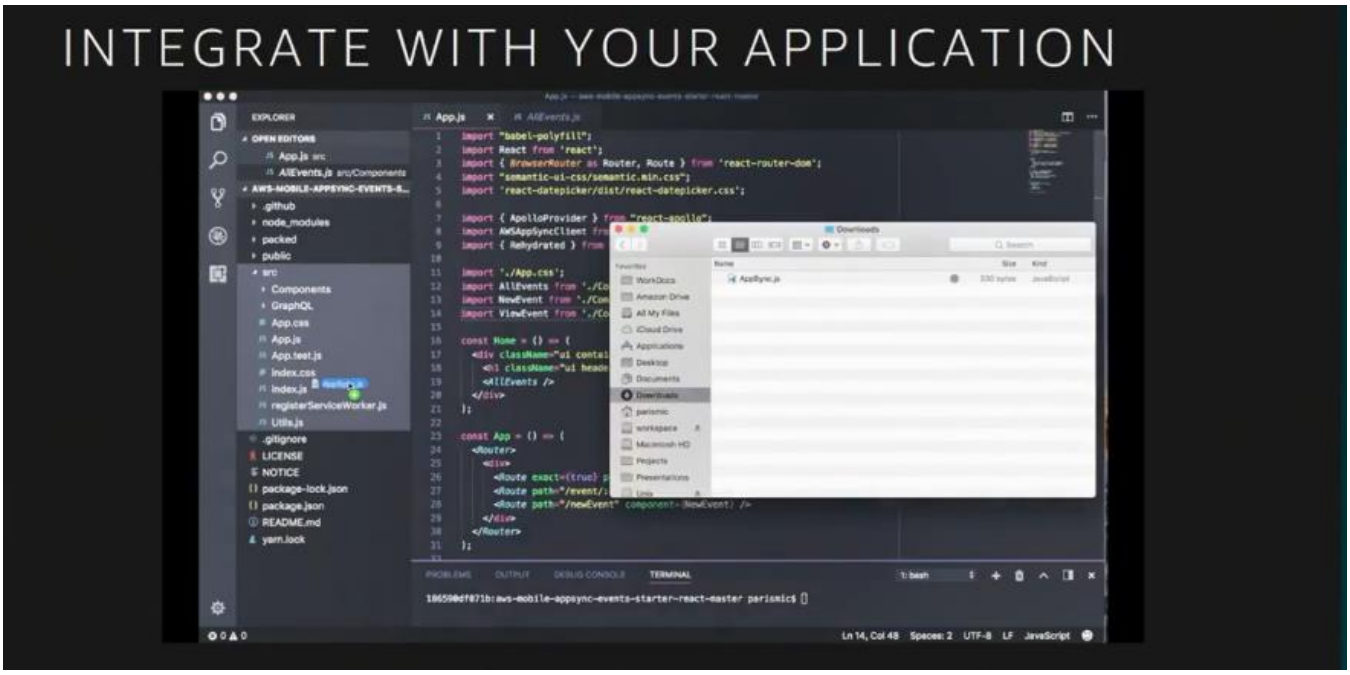


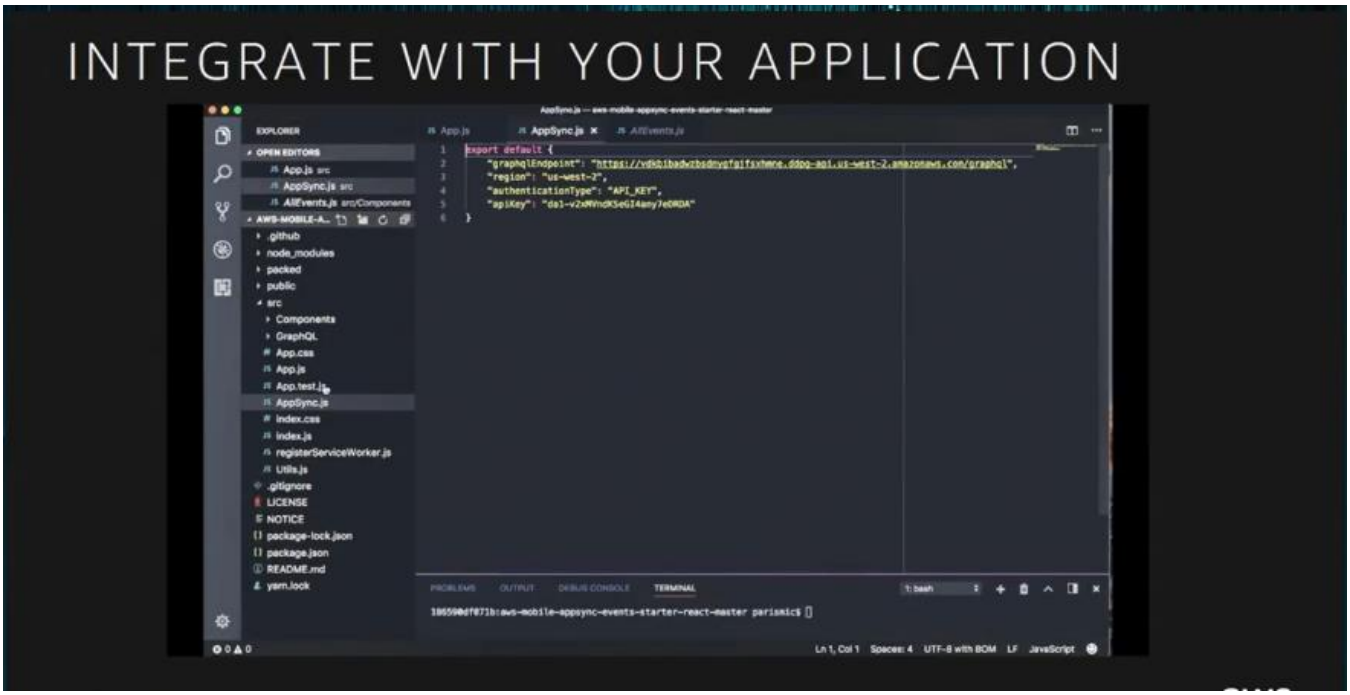
# INTEGRATE WITH YOUR APPLICATION



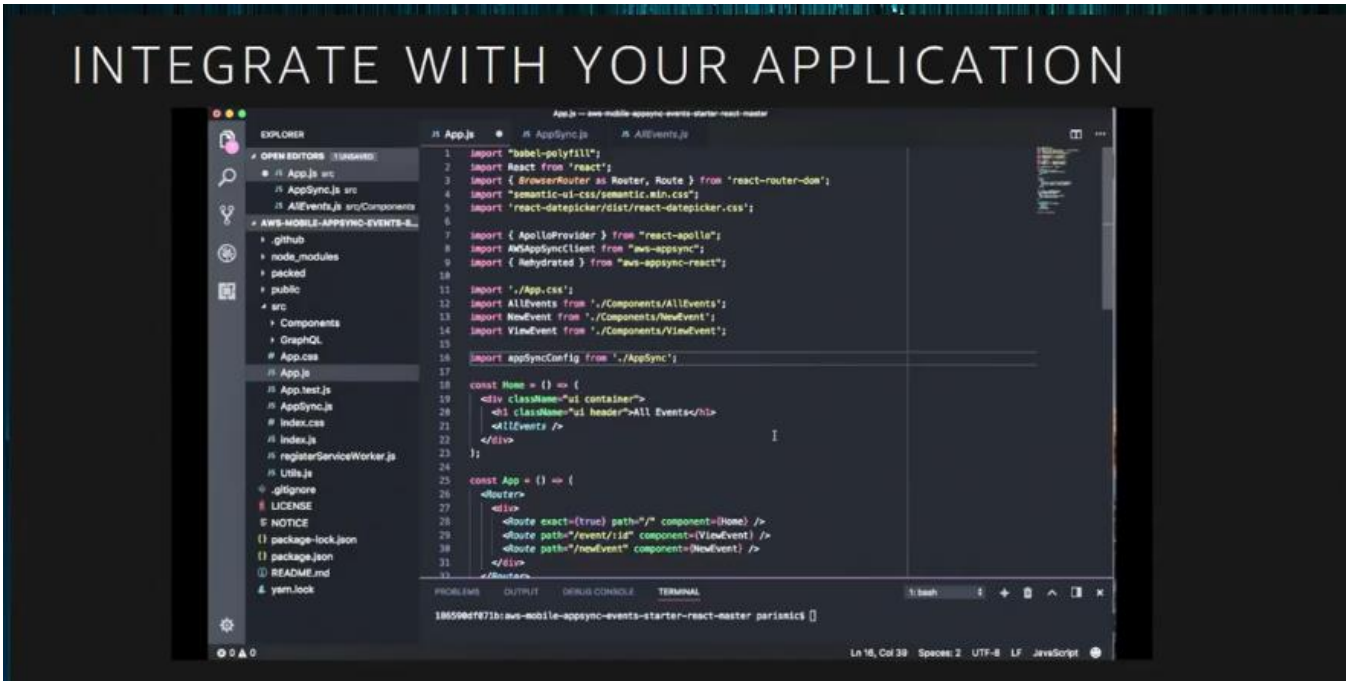

# INTEGRATE WITH YOUR APPLICATION



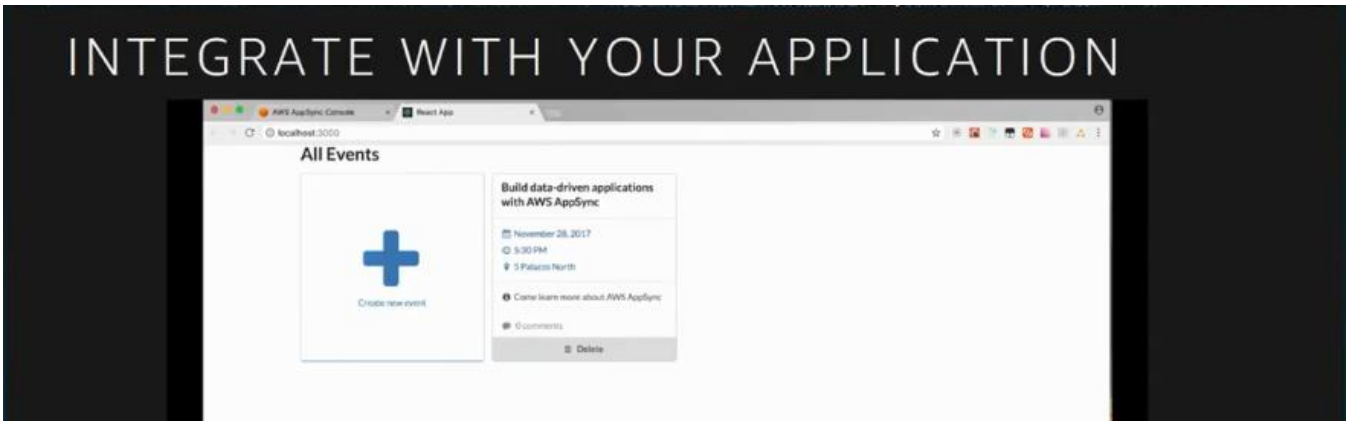


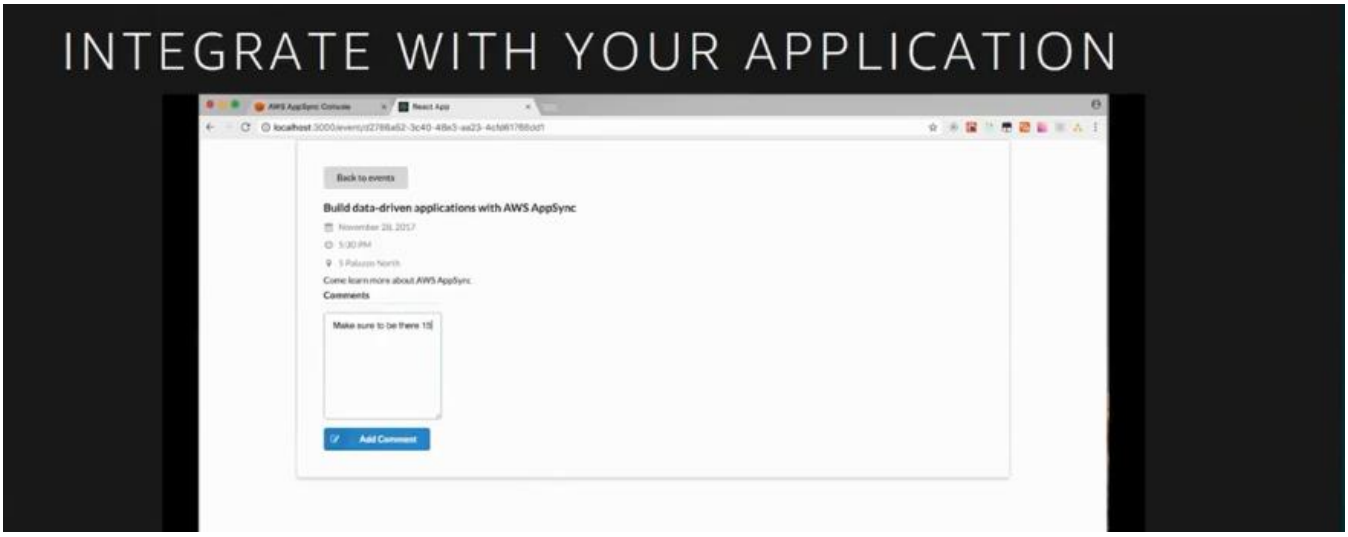



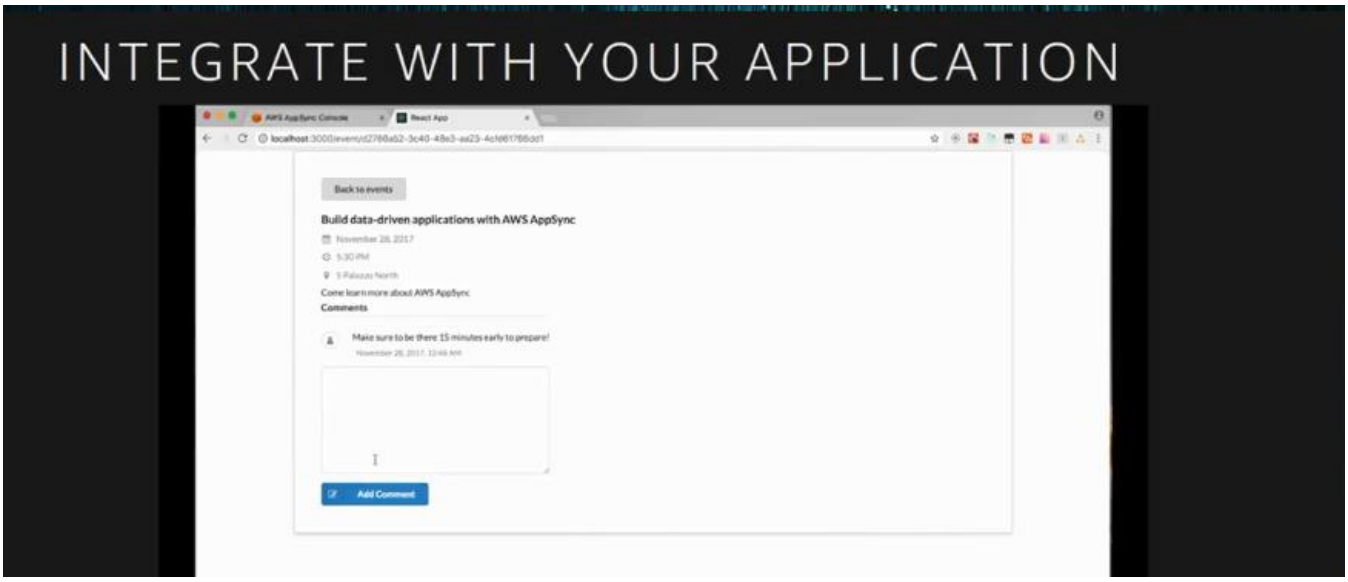

This is the configuration file that we can use on our app to integrate with the GraphQL backend, it contains things like the URL of our GraphQL server, the entry endpoint, default authorization mode is API Key

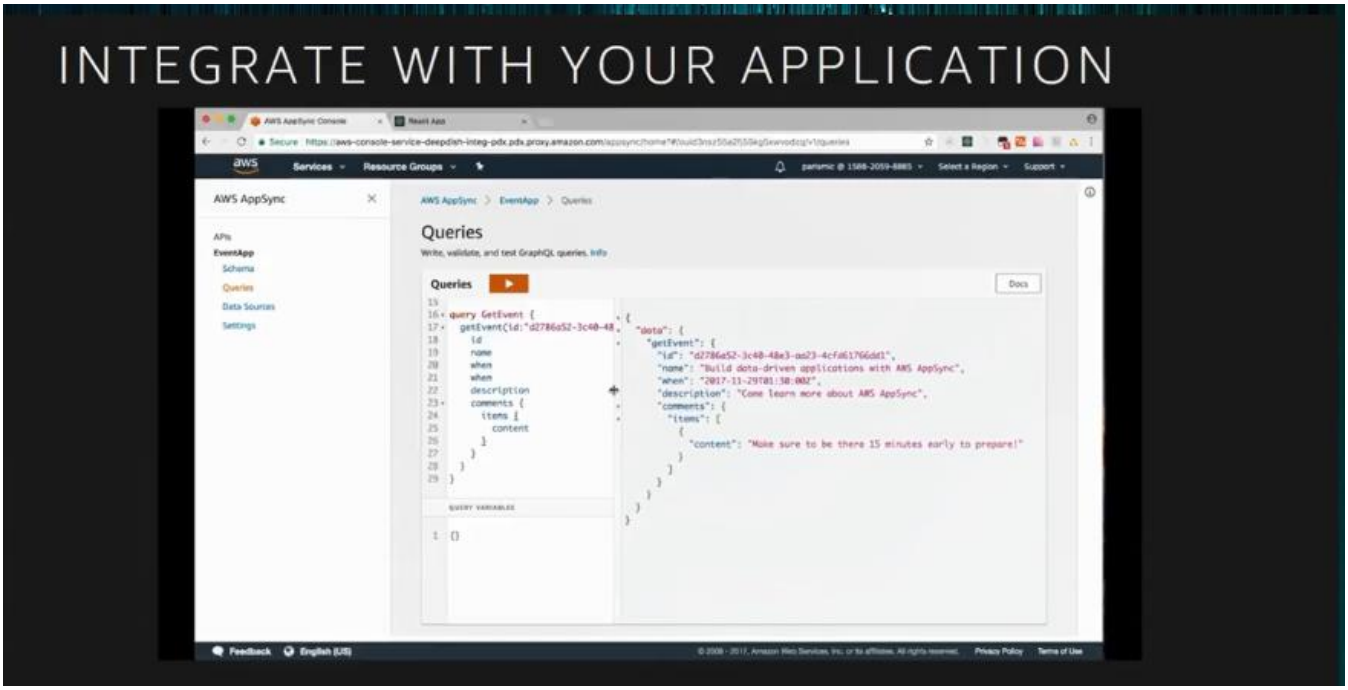



You just import it and start using as above. Do an ***npm install*** and ***npm start***



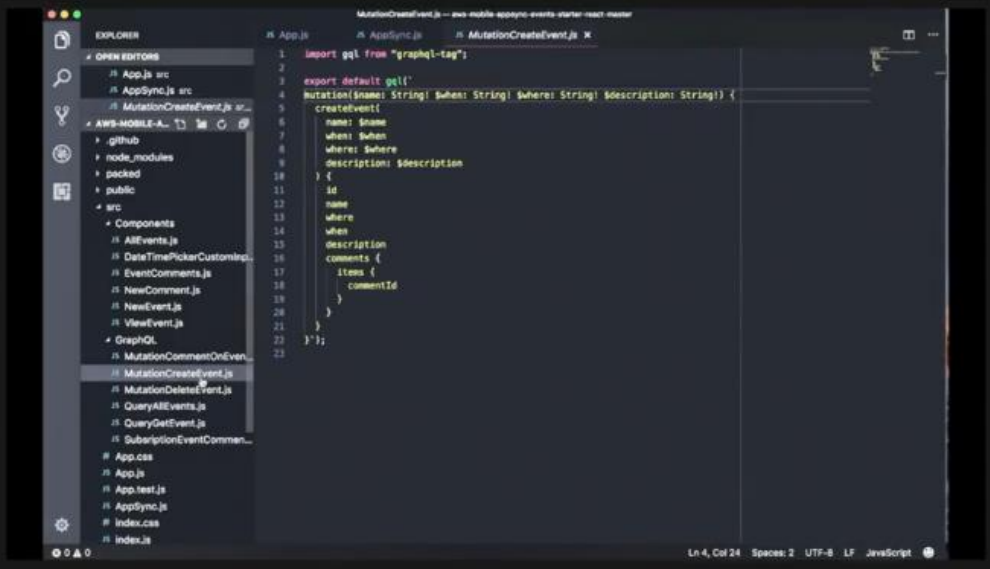






We now can see the newly added comment

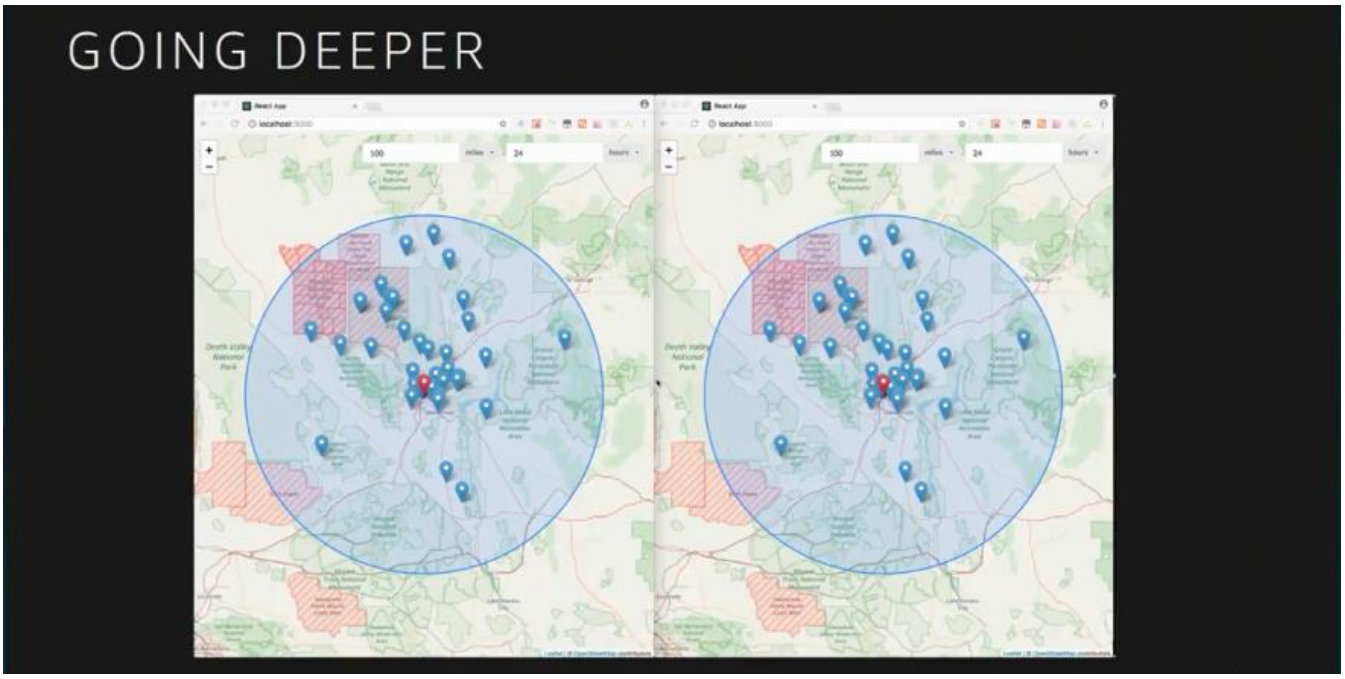
# INTEGRATE WITH YOUR APPLICATION



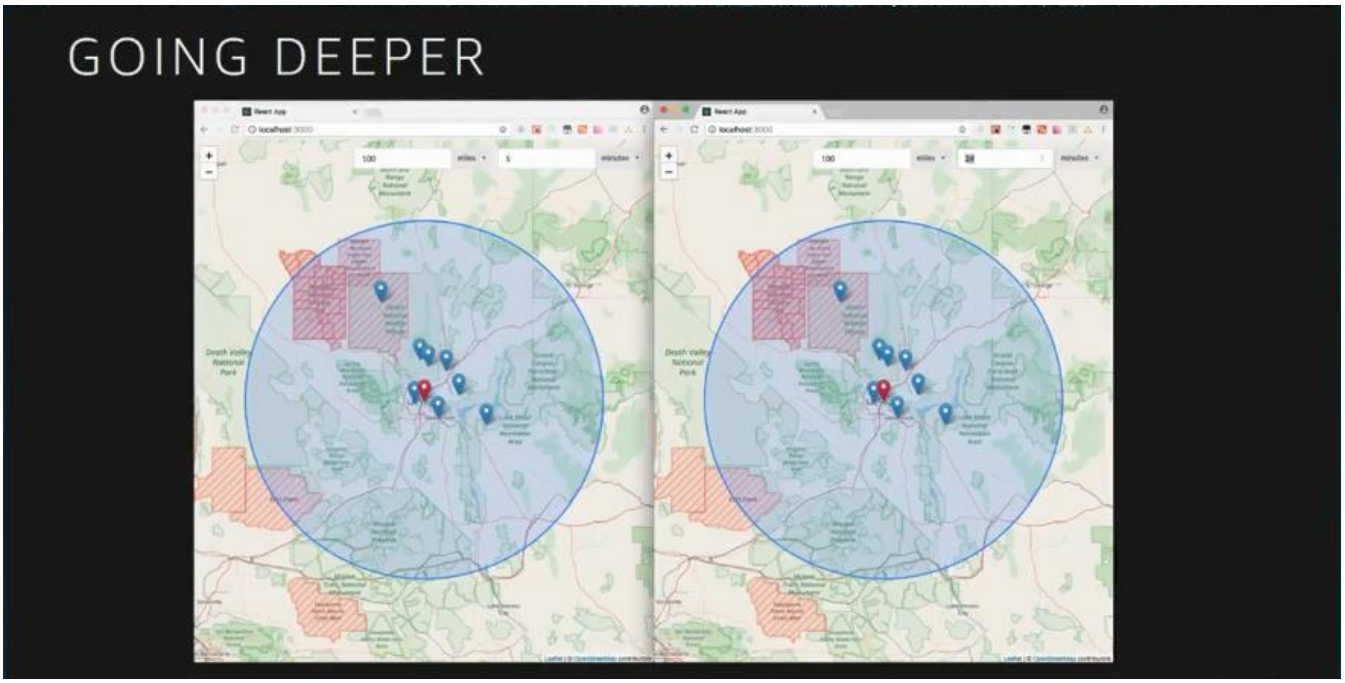

# RECAP

- 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

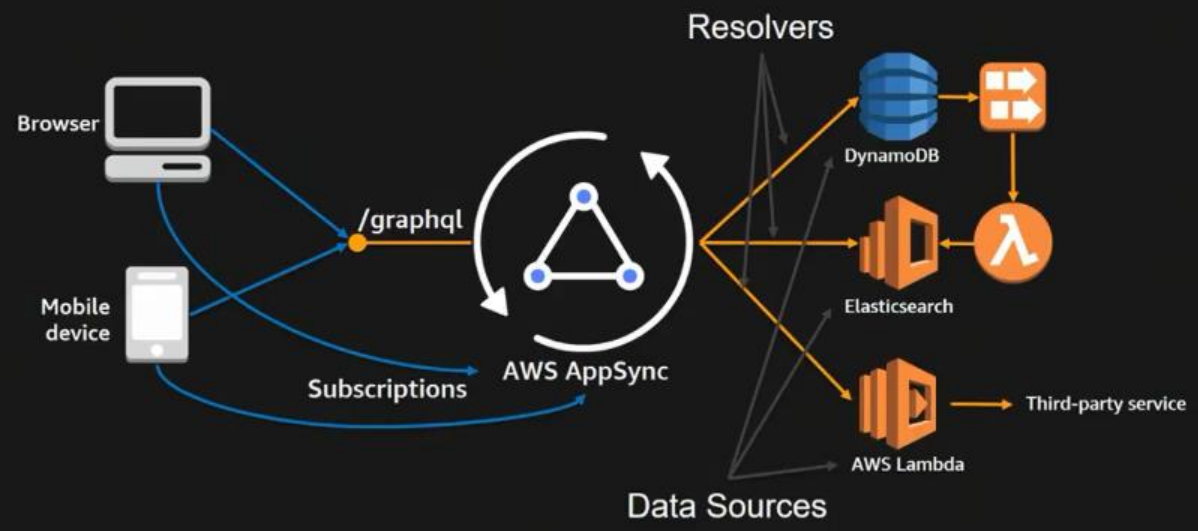
**AWS AppSync helps you build global scale and powerful applications**




We have an AWS AppSync app that is integrated with DynamoDB, Elasticsearch and a Lambda function that calls out to a 3rd 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.



# MAPTAP WITH AWS APPSYNC




AWS re:Invent



aws

# 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



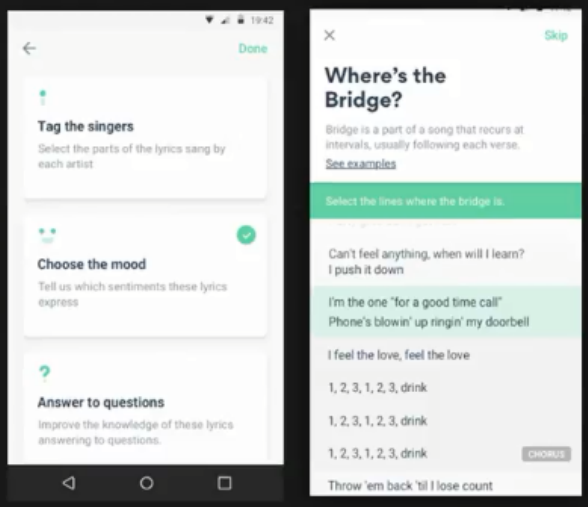
aws

# Understanding the music consumption

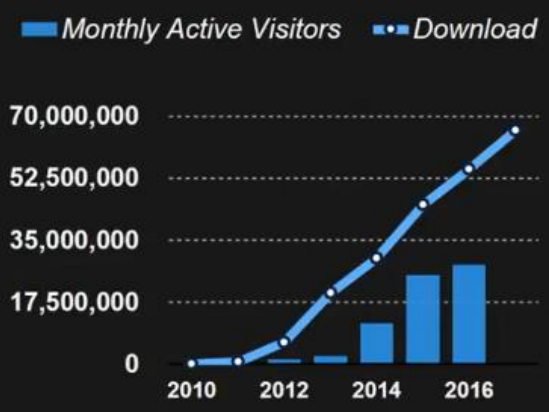
Musixmatch plugs in 3rd party music players (Spotify, iTunes, Youtube etc).

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

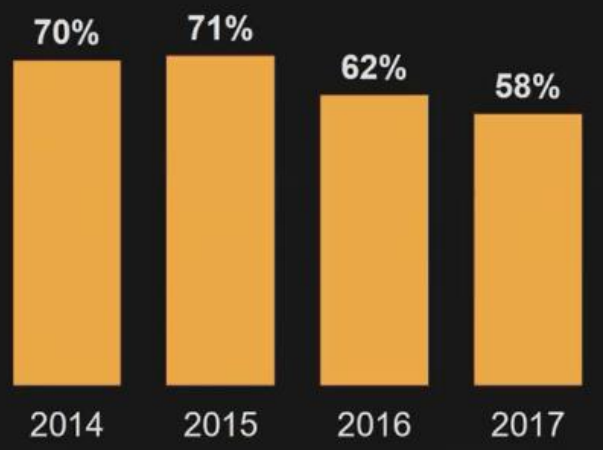



## Over 400k requests per minute, low at 200k

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




The split between EC2 / PaaS services changed over time

# Simplifying our app development and backend

## Multidevice development: React Native

| PROS | CONS |
| --- | --- |
| Vibrant ecosystem | Too many options |
| Development tools, e.g. Expo | Versioning problems |
| Leverage web skills | Development tools sometimes difficult to tweak |
| Easily pluggable in existing apps | |

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

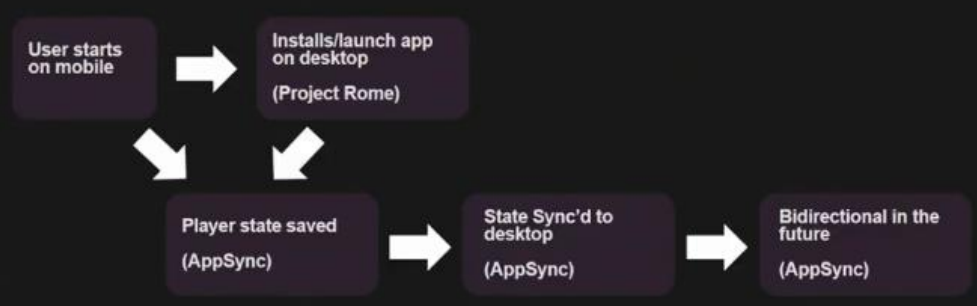
**What we tried:**

Open Source tools: SignalR or Socket.IO

**Challenges:**

Dealing with mobile connection reliability is hard

# Enabling multidevice experience 2/2

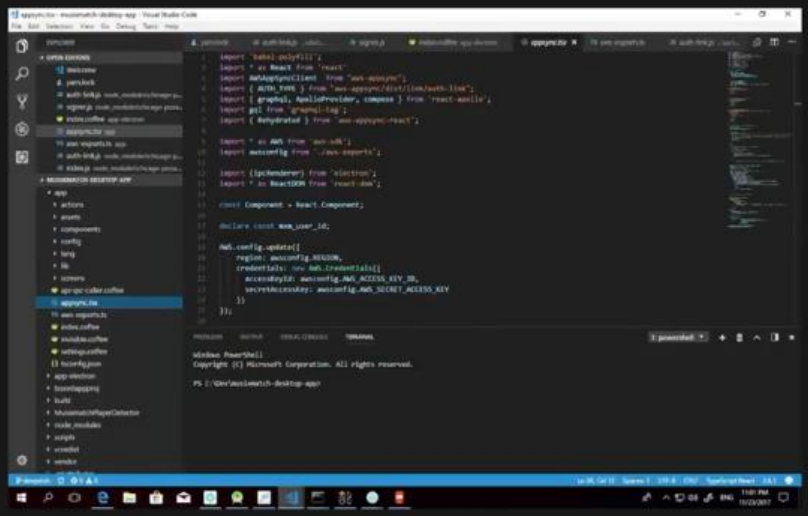



```
Type UserNowPlaying {
    user_id: ID! UserPlayerState
    guid: String!
    protocolUri: String!
}
```

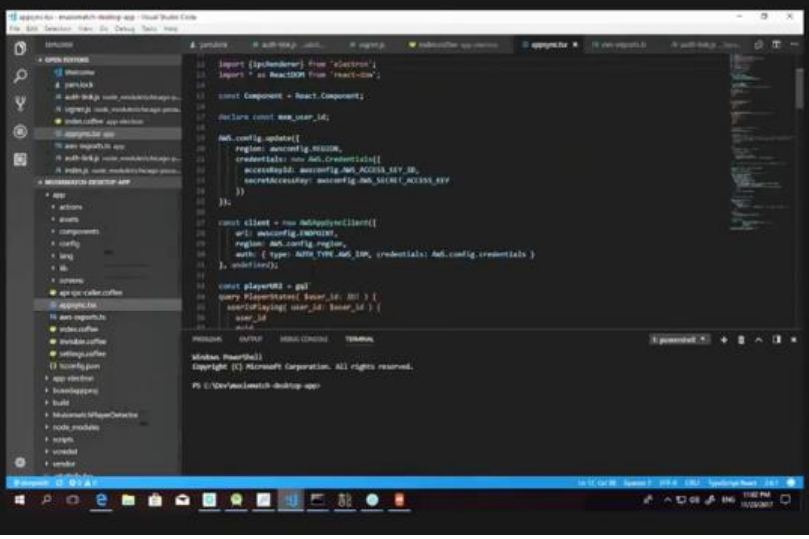
```
The custom protocol
URI contains

q_track
q_artist
duration
position
start_time
```

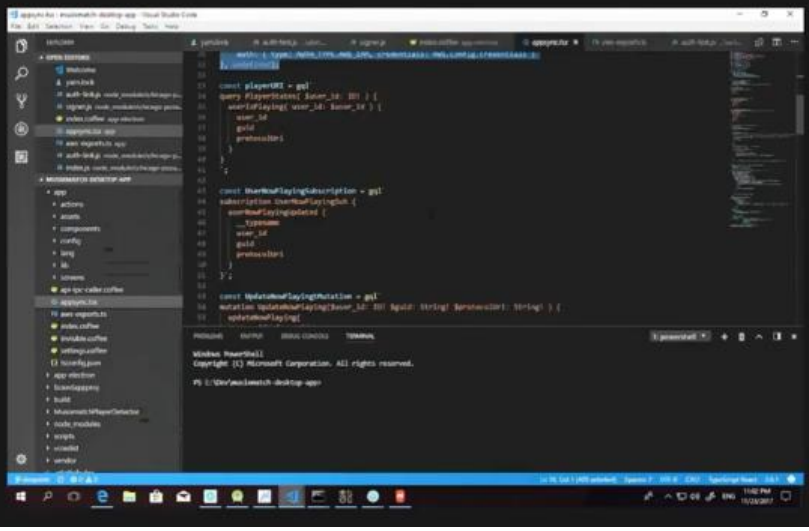
# CODE INTEGRATION



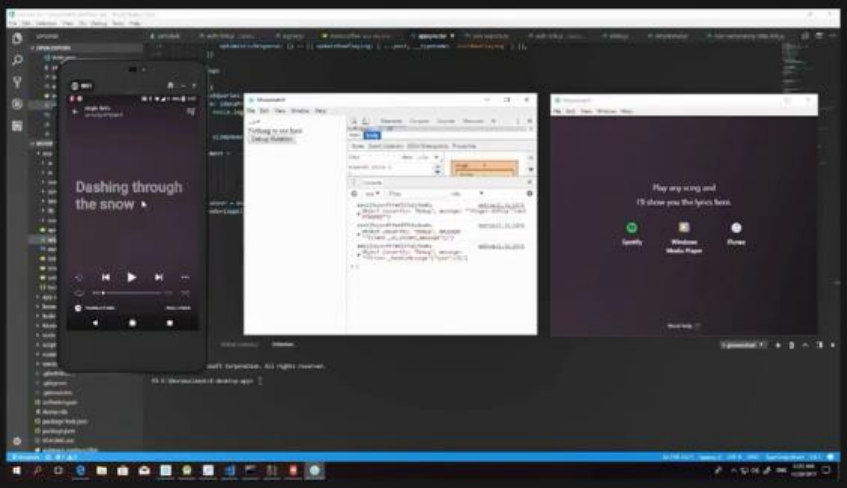
# CODE INTEGRATION



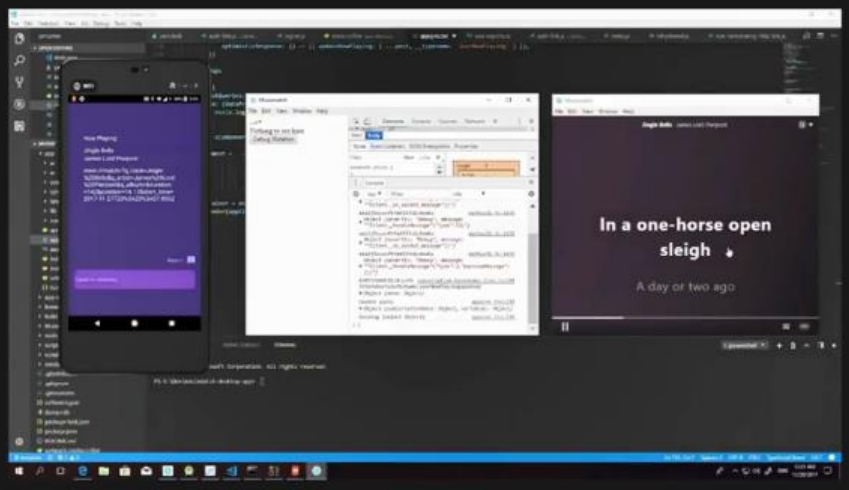
# CODE INTEGRATION



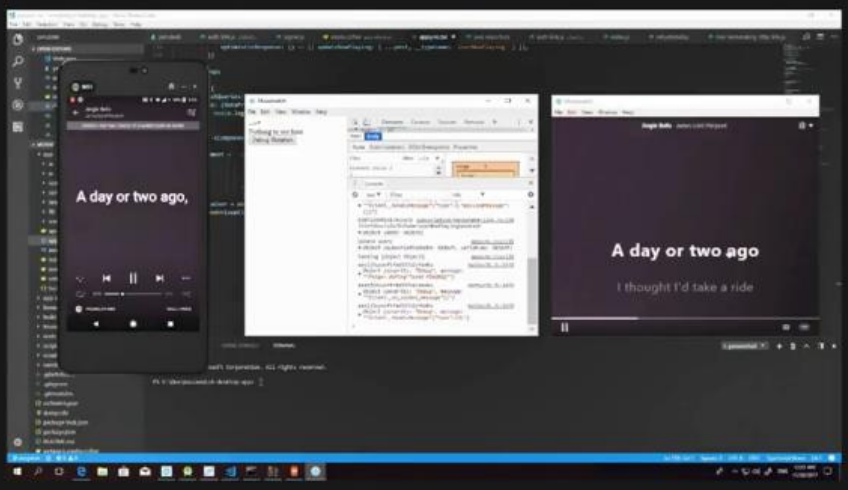
# APP DEMO



# 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



aws