AWS Amplify

AWS 서버리스 백엔드 환경을 만들어보자

발표자 : 다람쥐

발표날짜 : 2021년 07월 31일 토요일

서버리스

- 개발자가 서버를 관리할 필요가 없음
- 애플리케이션을 빌드하고 실행하는 클라우드 네이티브 개발 모델

AWS Amplify

- AWS 에서 아이디어부터 프로토타입까지 새로운 웹 애플리케이션을 만드는 가장 빠른 방법 중 하나
- 인증 (AWS Cognito), 스토리지 (AWS S3), REST API, GraphQL 백엔드 요소들을 CLI 로 쉽게 추가할 수 있음
- 배포도 가능
- AWS Amplify SDK 로 클라이언트에서도 쉽게 사용 가능

AWS Amplify 소개 - Amplify CLI

- 설치
  - npm install -g @aws-amplify/cli
  - amplify configure : CLI 용 IAM 생성
  - amplify init : Amplify 프로젝트 생성
- Amplify CLI로 구성요소를 구축할 수 있음
  - api, function, storage, auth, analytics, notifications, hosting 등
  - amplify api add

AWS AppSync

- GraphQL 을 사용하여 애플리케이션이 모든 데이터 소스, REST API 또는 마이크로서비스와 쉽게 상호작용할 수 있도록 하는 관리형 API 계층
- GraphQL 추상화 제공
  - Query (조회)
  - Mutation (생성, 수정)
  - Subscription (실시간 데이터 구독)
  - 스키마로 제공하여 특정 데이터소스에 종속적이지 않음
- 오프라인 (Amplify Client SDK) 동기화 제공, 실시간 (GraphQL Subscription) 제공이 장점

AWS Amplify 명령어 - API 만들기

- AWS API Gateway + Lambda
- AWS AppSync GraphQL + Lambda or DB

AWS Amplify 명령어 - API 만들기

- 서버리스 함수 생성
- amplify add function
  - amplify/backend 디렉터리 안에 생성
  - amplify/backend/function/{이름}/src/app.js 에 작성

AWS Amplify 명령어 - API 만들기

```
var express = require('express')
var bodyParser = require('body-parser')
var awsServerlessExpressMiddleware = require('aws-serverless-express/middleware')

var axios = require('axios')

// declare a new express app
var app = express()
app.use(bodyParser.json())
app.use(awsServerlessExpressMiddleware.eventContext())

// Enable CORS for all methods
app.use(function(req, res, next) {
res.header("Access-Control-Allow-Origin", "*")
res.header("Access-Control-Allow-Headers", "*")
next()
});

/* coin methods */
app.get('/coins', function(req, res) {
let apiUrl = `https://api.coinlore.com/api/tickers?start=0&limit=10`

if (req.apiGateway && req.apiGateway.event.queryStringParameters) {
const { start = 0, limit = 10 } = req.apiGateway.event.queryStringParameters
apiUrl = `https://api.coinlore.com/api/tickers/?start=${start}&limit=${limit}`
}

axios.get(apiUrl)
.then(response => {
res.json({ coins: response.data.data })
})
.catch(err => res.json({ error: err }))
})
```

AWS Amplify 명령어 - API 만들기

- 새 API 생성
- amplify add api
  - REST 설정
  - 이름 설정
  - Path 설정
  - Use a lambda function already added in the current Amplify project
  - 생성한 함수 이름

AWS Amplify 명령어 - API 만들기

- API 와 Lambda 함수 배포
- amplify push

AWS Amplify 명령어 - AppSync GraphQL API 만들기

- amplify init
- amplify add api
  - GraphQL 선택
- amplify/backend/api/notesapi/schema.graphql
  - 스키마 설정
- amplify push

AWS Amplify 명령어 - AppSync GraphQL API 만들기

- AppSync 전용 디렉티브
- @model
- @auth
  - 권한 규칙
- @key
  - GSI 및 정렬 키 설정
- @connection
  - 타입 간의 관계

AWS Amplify 명령어 - AppSync GraphQL API 만들기

```
type Stage @model
  @auth(rules: [
  { allow: public, operations: [read] },
  { allow: groups, groups: [“Admin”] }
]) {
  id: ID!
  name: String!
  performances: [Performance] @connection
  (keyName: “byStageId”, fields: [“id”])
}
```

```
type Performance @model
  @key(name: “byStageId”, fields: [“performanceStageId”])
  @auth(rules: [
    { allow: public, operations: [read] },
    { allow: groups, groups: [“Admin”] }
]) {
  id: ID!
  perfomanceStageId: ID!
  productionID: ID
  performer: String!
  imageUrl: String
  stage: Stage @connection
}
```

AWS Amplify 명령어 - 인증

- amplify init
- amplify add auth
- amplify push

AWS Amplify 명령어 - AWS S3 생성

- amplify add storage
  - Content
  - 리소스 이름
  - 버킷 이름
  - 사용자 접근 범위
  - 람다 트리거 생성

AWS Amplify 명령어 - DynamoDB 생성

- amplify add storage
  - NoSQL Database
  - 리소스 이름
  - 테이블 이름
  - 칼럼 생성
  - Sort key, Global Secondary index 추가
  - 람다 트리거 추가 여부

AWS Amplify 명령어 - Amplify DataStore

- 구성 요소
  - AppSync GraphQL API
  - 오프라인에서 데이터 유지하는 Local Storage
  - 정교한 충돌 감지 및 충돌 해결 가능케 해주는 그래프QL 리졸버
- CLI 로 생성할 때 충돌 탐지를 사용하도록 설정
- 스토리지 저장소와 상호작용하는 모델 생성
  - amplify codegen models

AWS Amplify 명령어 - Amplify DataStore

```
import { Data Store } from '@aws-amplify/datastore'
import { Message } from './models'

// 데이터 저장
await DataStore.save(
  new Message({ title: 'Hello World!', sender: 'Chipmunk' })
)

// 데이터 조회
const posts = await DataStore.query(Post)
// 데이터 삭제
const message = await DataStore.query(Message, '123')
DataStore.delete(message)
// 데이터 수정
const message = await DataStore.query(Message, '123')
await DataStore.save( Post.copyOf(message, updated => {
  updated.title = "My new title"
}))
// 실시간 기능을 위한 데이터 변경 사항 추적
const subscription = DataStore.observe(Message.subscribe(msg => {
  console.log(message.model, message.opType, message.element)
})
```

감사합니다.

참고

1. 풀스택 서버리스 / 한빛미디어, 네이더 다빗 저, 김범준 역 (2021)