Serverless ImmersionDay

순서

1. 설정
2. 의존성 설치
3. 터미널에서 게임 서비스 테스트
4. 웹페이지에서 게임 서비스 테스트

1. 설정

Cloud9 환경 생성

- AWS 웹 콘솔의 서비스페이지에서 Cloud9 검색후 아래 내용으로 IDE 환경 생성
  - 이름: Serverless
  - 인스턴스 타입: **t3.small**
  - 플랫폼: Amazon Linux 2
  - 나머지는 기본설정
- 아래 진행되는 모든 내용은 Cloud9 위에서 실행

툴 설치

C9 IDE 에서 터미널 실행후 아래 명령 입력

```
$ sudo yum install jq httpie -y
$ npm i -g wscat cdk@1.76.0
```

워크샵레포지토리 클론

```
$ git clone https://github.com/haandol/serverless-event-driven-workshop
```

CDK 로 인프라스트럭쳐 프로비전

워크샵 의존성 설치 및 계정에 CDK 초기화

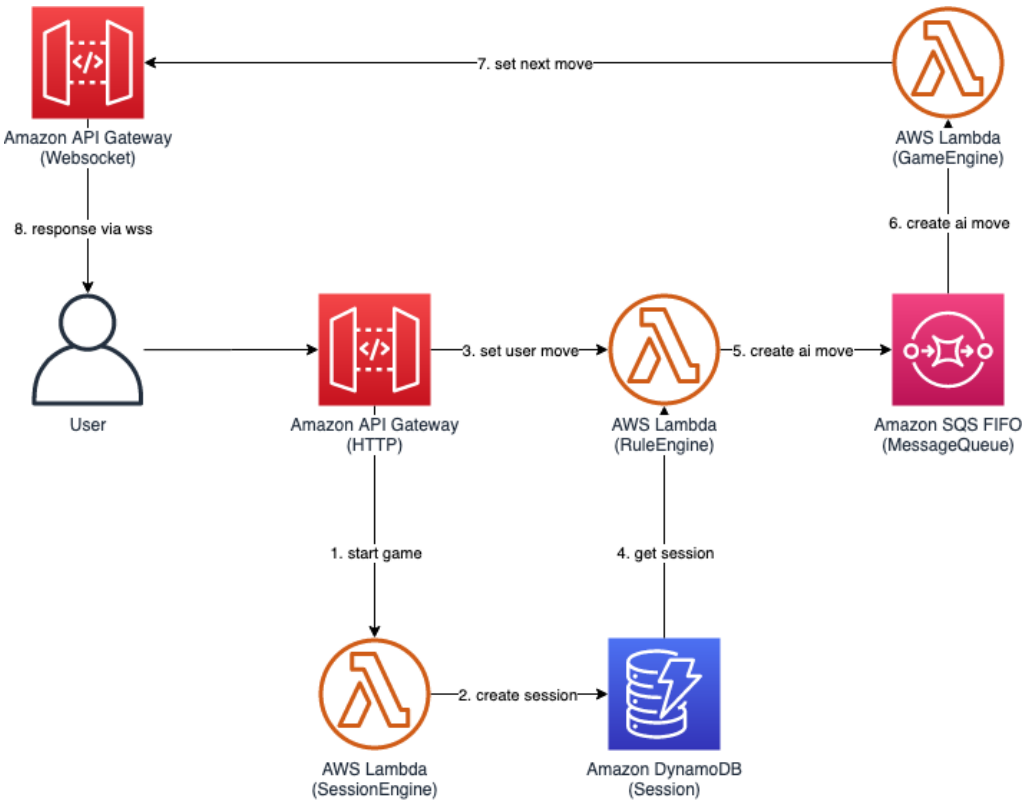
```
$ cd serverless-event-driven-workshop/infra
$ npm i
```

CDK 배포

```
$ cdk bootstrap
$ cdk deploy "*" --require-approval never
```

터미널에서 게임 서비스 테스트

CDK 프로젝트를 배포하고나면 아래와 같은 서버리스 어플리케이션이 배포된다.



잘 알려진 베스킨라빈스 `31` 이라는 게임에서 31을 13으로 바꾼 형태의 게임 구현이다. 게임 서비스의 이용방법은 아래와 같다.

1. wscat 으로 웹소켓 API 에 접속후 `connectionID` 확인
2. `POST /start` API 호출하여 게임세션 시작
3. `POST /game` API 호출하여 게임 진행
4. 웹소켓으로 서버응답 오는지 확인
5. 사용자가 13입력하기 전까지 3~4 단계 진행

웹소켓 API 주소가져오기

터미널에서 아래와같이 웹소켓 주소를 가져온다

```
$ export WS_ENDPOINT=$(aws cloudformation describe-stacks --stack-name ServerlessWorks
$ echo $WS_ENDPOINT

wss://xxx.execute-api.ap-northeast-2.amazonaws.com/dev
```

wscat 을 이용하여 웹소켓에 접속하기

위에서 설치한 wscat 을 이용하여 웹소켓에 접속해보고, connectionId 를 확인한다.

해당 connectionId 는 웹소켓 접속시마다 바뀌는 세션아이디이다.

```
$ wscat -c $WS_ENDPOINT

Connected (press CTRL+C to quit)
> hi
< {"message": "Forbidden", "connectionId":"XDmENcnwIE0CFEQ=", "requestId":"XDmGEE2RoE0
>
```

게임세션 시작하기

새로운 터미널을 띄우고 Http API 주소를 가져온다

```
$ export HTTP_ENDPOINT=$(aws cloudformation describe-stacks --stack-name ServerlessWor
$ echo $HTTP_ENDPOINT

https://xxxx.execute-api.ap-northeast-2.amazonaws.com/dev
```

위에서 복사해둔 connectionId 를 이용하여 `POST /start` 요청을 진행한다

```
$ export CONN_ID=CONNECTION_ID_YOU_COPIED   # e.g. XDqRFclbIE0CJPw=
$ http post $HTTP_ENDPOINT/start sessionId=$CONN_ID accountId=1

HTTP/1.1 200 OK
Apigw-Requestid: XZRxtgVAoE0EPnQ=
Connection: keep-alive
Content-Length: 16
Content-Type: text/plain; charset=utf-8
Date: Fri, 11 Dec 2020 15:55:30 GMT

Session created: XDqRFclbIE0CJPw=
```

게임진행하기

`POST /game` 요청을 통해 실제로 게임을 진행해본다.

```
$ http post $HTTP_ENDPOINT/game sessionId=$CONN_ID accountId=1 userMove=1,2

HTTP/1.1 200 OK
Apigw-Requestid: XDqYKi2WoE0EPdg=
Connection: keep-alive
Content-Length: 45
Content-Type: text/plain; charset=utf-8
Date: Sat, 05 Dec 2020 02:31:30 GMT
{
    "lastMove": 2,
    "sessionId": "XDqRFclbIE0CJPw="
}
```

해당 요청을 보내고 정상적인 응답을 받았다면 웹소켓 터미널에 다음과 같이 서버의 응답이 표시된다.

```
> hi
< {"message": "Forbidden", "connectionId":"XDqRFclbIE0CJPw=", "requestId":"XDqROEJuoE0
< {"data":"3,4"}
```

계속해서 `POST /game` 요청을 통해 13까지 게임을 진행해보자. 웹소켓 터미널에 다음과 같이 표시된다.

```
> hi
< {"message": "Forbidden", "connectionId":"XEEuXcgPoE0Acmw=", "requestId":"XEFD0E2foE0
< {"data":"3,4"}
< {"data":"7,8"}
< {"data":"10,11,12"}
< {"data":"You lose.."}
```

웹페이지를 통해 게임 테스트

위에서 사용한 내용을 웹페이지로 구성해서 테스트해본다.

웹페이지 수정

1. IDE 에서 public/index.html 페이지를 연다.
2. 17,18 라인의 script 태그 안에있는 wsUrl 과 httpUrl 을 위에서 확인한 WS_ENDPOINT 와 HTTP_ENDPOINT 로 각각 수정한다.

웹서버 실행

1. 새 터미널을 띄우고, 클론받은 레포지토리의 public 폴더로 이동한다.
2. 파이썬에서 제공하는 간단한 정적파일 호스팅을 이용하여 8080 포트로 웹서버를 띄운다

```
$ cd serverless-event-driven-workshop/public/
$ python -m http.server 8080
```

3. IDE 상단에 있는 Preview > Preview Running Application 을 클릭하여 브라우저에서 게임을 진행해본다.