

 AWS Serverless 소모임

AWS Full Serverless 기반 대기열 구현

> 소개



박상운
(spark@rubywave.io)


이력

- Frontend 5년 / Backend 9년 / AWS 8년 경력
- Georgia Institute of Technology 중퇴
- AWS 공인 솔루션즈 아키텍트 / DevOps 엔지니어 프로페셔널
- (前) (주)트위니 솔루션즈 아키텍트 / DevOps 엔지니어
- (주)리온랩스 Dev Lead

특이사항

- 개발 모토 : Serverless로 모든 문제를 풀어보자!
- 유튜브 “AWS 강의실” 운영자

> 소개



AWS 강의실

@AWSClassroom 구독자 1.27만명 동영상 127개

AWS를 쉽게 알려드리는 AWS 강의실입니다. >

구독중

가입

홈

동영상

실시간

재생목록

커뮤니티

채널

정보

🔍

채널 회원

채널 회원이 되어 주셔서 감사합니다

가입

쉽게 설명하는 AWS 기초 강좌

▶ 모두 재생

쉽게 설명하는 AWS 기초 강좌 시리즈 1강 클라우드 컴퓨팅이란?

16:18

세계에서 제일 평균연봉 높은 IT 자격증 알려드립니다. feat. 지금 바로 따라하는 이

6:46

쉽게 설명하는 AWS 기초강좌 시리즈 2강 클라우드 컴퓨팅의 종류

8:36

가용영역 리전 쉽게 설명하는 AWS 기초강좌 시리즈 3강 AWS의 구조

17:54

실습 포함! 쉽게 설명하는 AWS 기초강좌 시리즈 4강 AWS 계정 만들기 & 첫 설정

22:03

쉽게 설명하는 AWS 기초 강좌 1: 클라우드 컴퓨팅이란?

AWS 강의실

조회수 3.4만회 · 1년 전

세계에서 제일 평균연봉 높은 IT자격증 알려드립니다(feat...

AWS 강의실

조회수 3.2만회 · 1년 전

쉽게 설명하는 AWS 기초 강좌 2: 클라우드 컴퓨팅의 종류

AWS 강의실

조회수 1.3만회 · 1년 전

쉽게 설명하는 AWS 기초 강좌 3: AWS의 구조-리전,가용역...

AWS 강의실

조회수 1.5만회 · 1년 전

쉽게 설명하는 AWS 기초 강좌 4: AWS 계정 만들기 및 첫 설...

AWS 강의실

조회수 1.4만회 · 1년 전

AWS Serverless 아키텍처링/개발

▶ 모두 재생

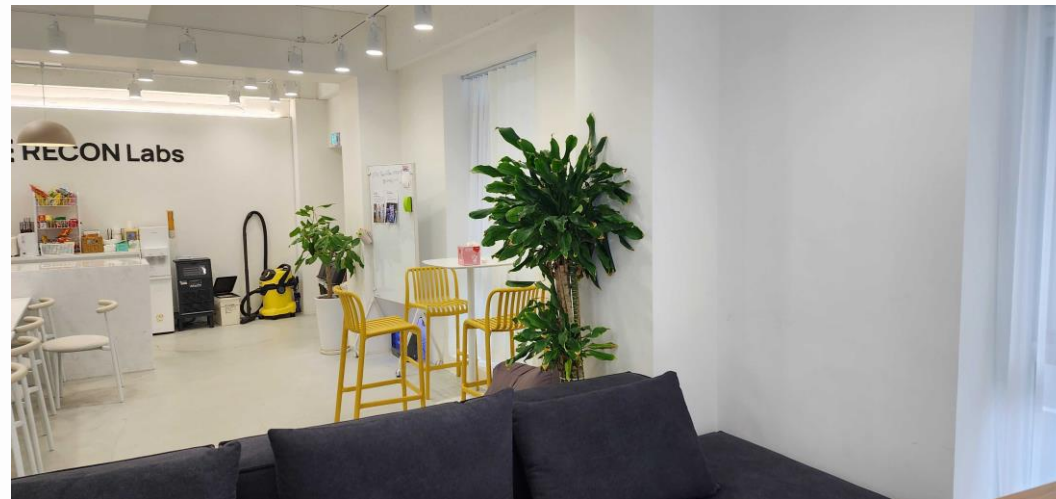
R: RECON Labs

Core Value

WE MAKE 3D CONTENT ACCESSIBLE

리콘랩스는 무궁무진한 가능성이 펼쳐질 디지털 세상과의 새로운 결합, 소통, 탐구를 통해
누구나 쉽게 3D 콘텐츠를 만들고 활용할 수 있는 온라인 서비스를 제공합니다.

> 리콘랩스



AWS Serverless 소모임

AWS Full Serverless 기반 대기열 구현

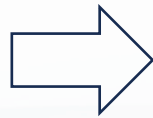
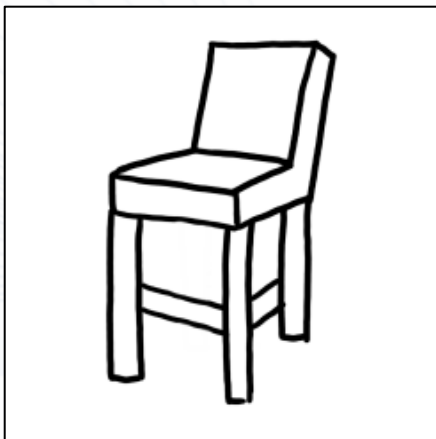
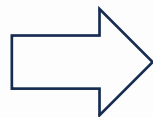
> 리콘랩스



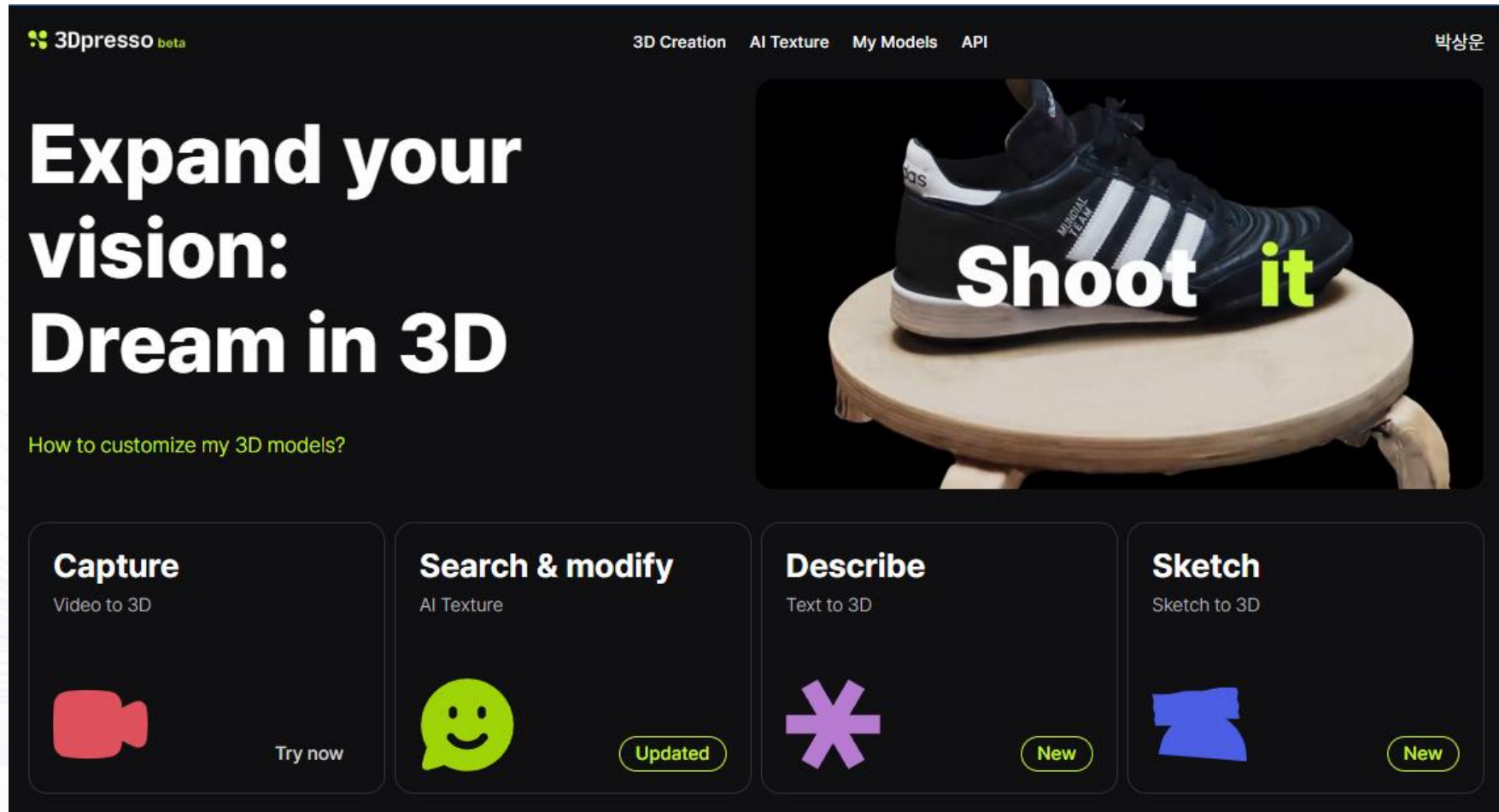
AWS Serverless 소모임

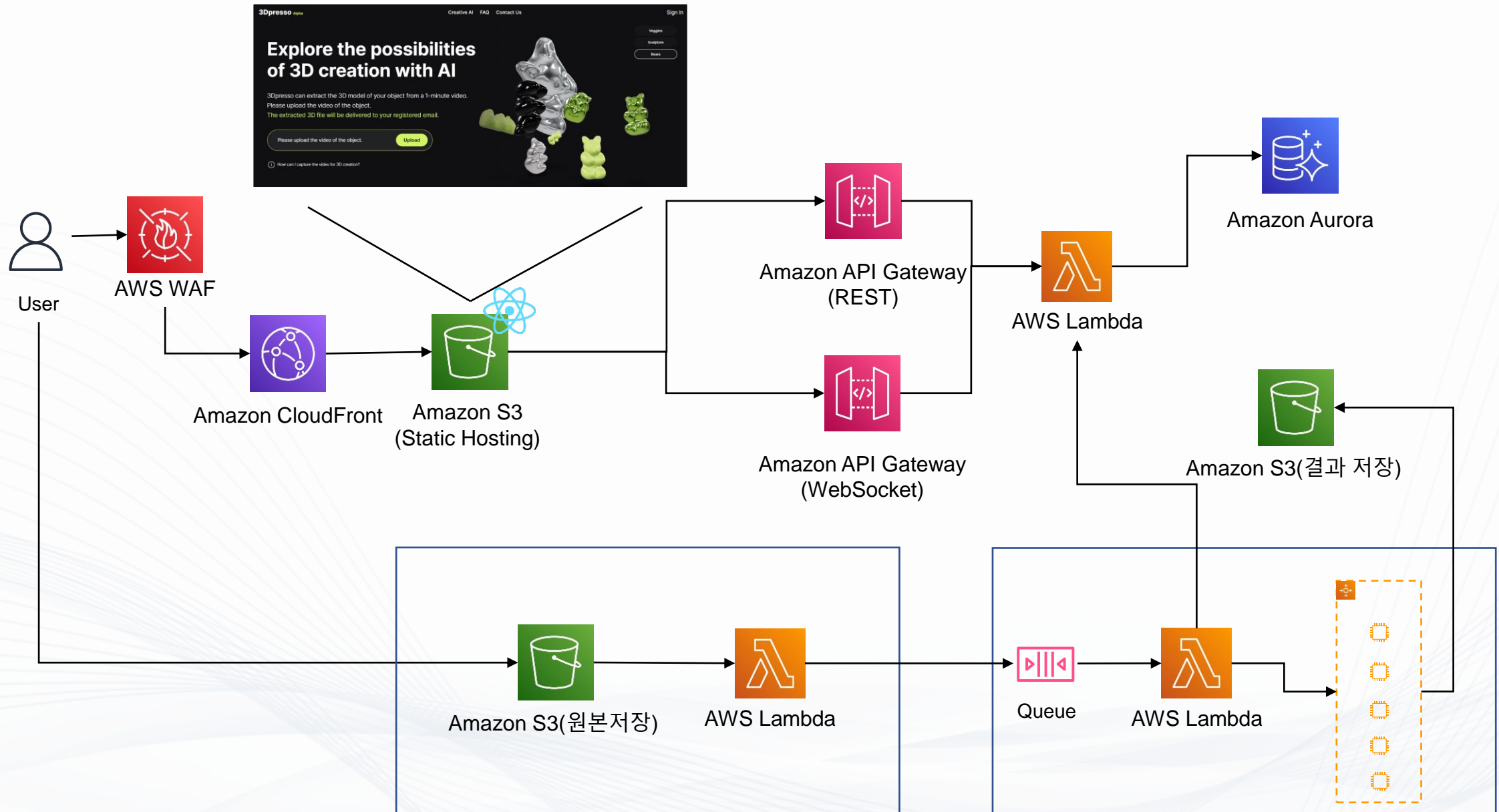
AWS Full Serverless 기반 대기열 구현

> 리콘랩스



> 리콘랩스





> 이번 발표는...

> 이번 발표는...

- AWS Serverless 서비스를 적극적으로 활용한 아키텍처
 - 데모 아키텍처 : 실전에서 사용하는 것을 조금 변형
 - 약간 가상의 상황을 상정 : 실전보다는 “이런것도 가능해요”
 - 물론 조금만 손본다면 다양하게 응용 가능
 - Lambda, DynamoDB, API Gateway, StepFunctions 등의 서비스를 알고 계시다고 가정
- 재미 보다는 내용 위주
 - 좀 설명이 길고 많아요 ππ
 - 궁금하신 점/설명이 더 필요한 부분 있으면 바로 질문!
- Backend/ Frontend로 구성된 Demo
 - 소스코드(Serverless Framework 템플릿) 모두 공유

> 목표

> 목표

- 명절 KTX 예매와 같은 대기열 구성
 - 동시에 X명만 예약 가능
 - 각 사람 당 Y초 만큼만 예약 가능
 - N초 단위로 대기중인 사람 숫자 갱신
- Serverless서비스로만 구성

[닫기] 버튼을 누르면 대기 순서가 다시 부여됩니다.

[닫기] 버튼은 일시적인 통신 장애 등으로 인터넷 접속이 끊겨 대기자 수가 줄지 않고 멈춘 경우에만 사용하시기 바랍니다.

i 예약접속까지 **최장 98분이 소요될 수도 있습니다.**

예약접속 대기 안내

현재 **8385**명이 접속대기 중이며, 대기순서에 따라 자동 접속됩니다.

예약 요청 횟수 및 시간 제한 안내

예약요청 횟수는 **6회**, 전체 예약시간은 **3분**까지입니다.

- 고객님의 선호하는 시간대의 좌석은 예매 시작 초기에 매진될 수 있습니다.
- 예약화면 접속 후 잔여석 부족으로 예약이 어려울 수 있습니다.

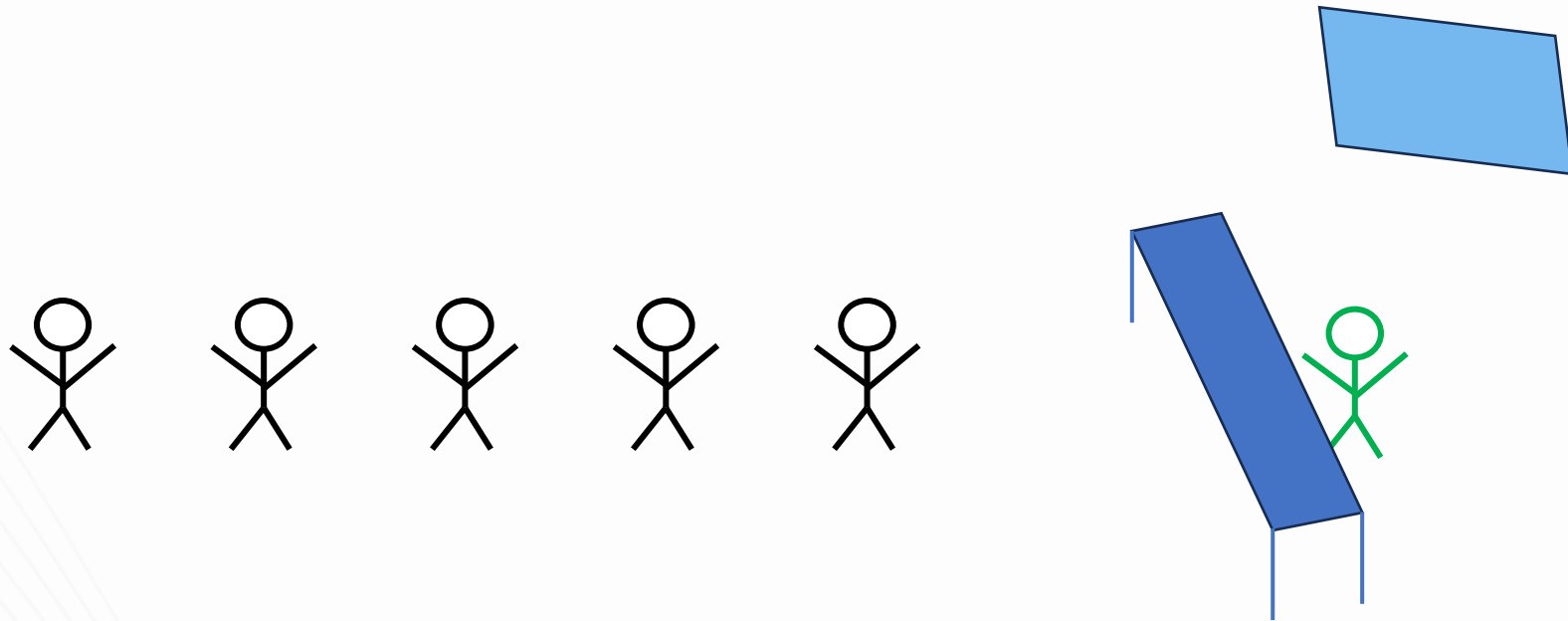
KORAIL

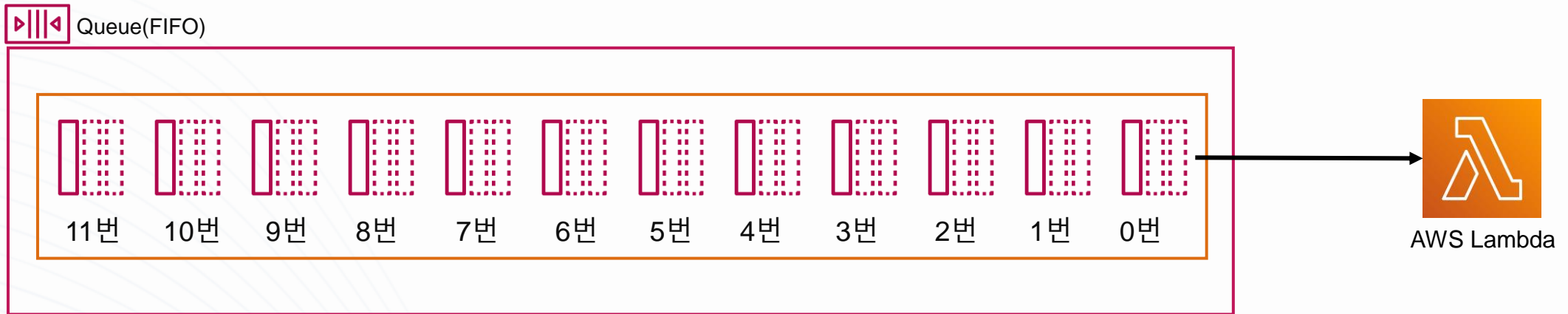
▶ **주요시간별 잔여석 현황**

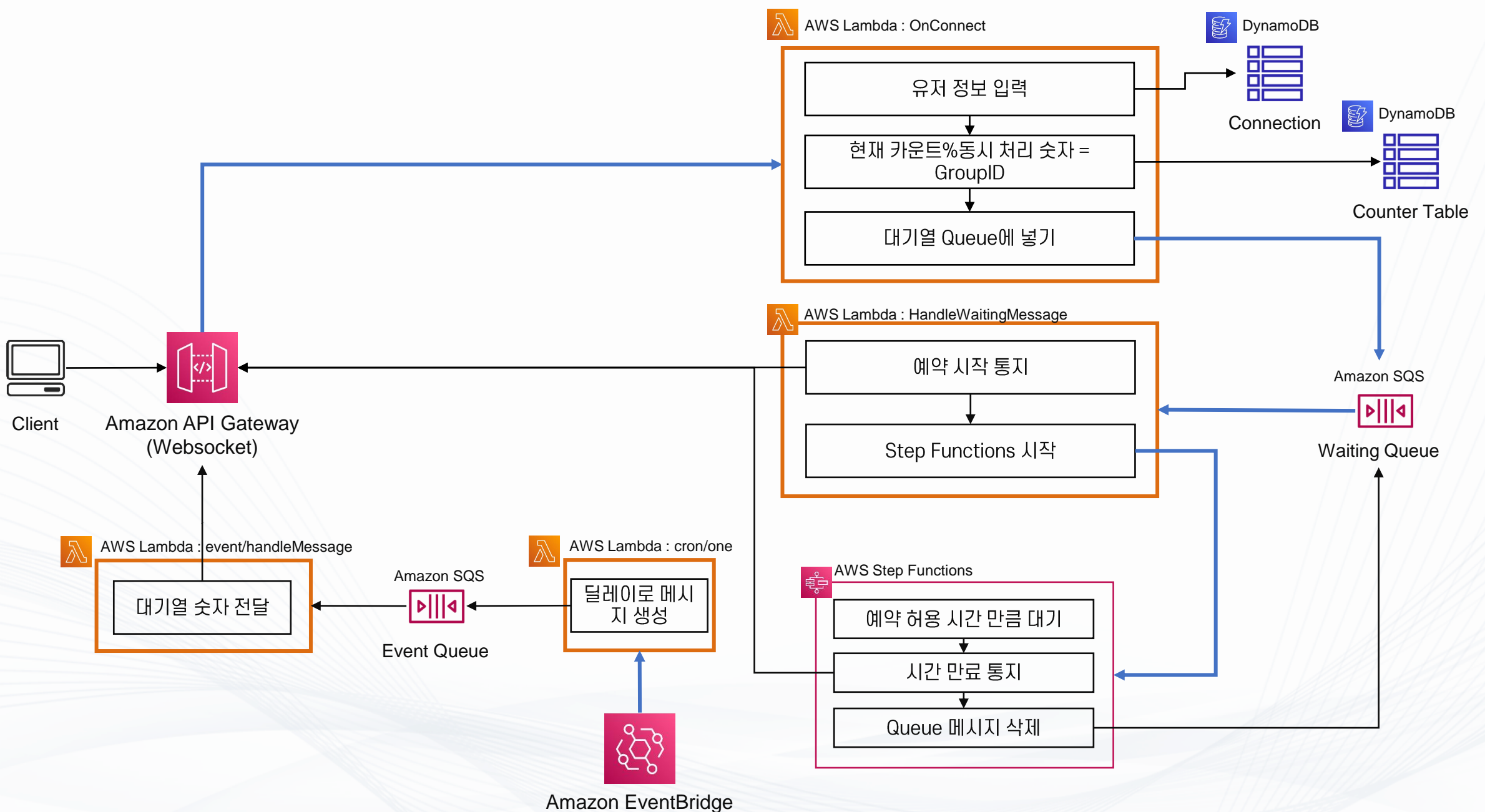
› 기본 아키텍처 핵심

› 기본 아키텍처 핵심

- SQS FIFO큐에 유저 요청을 넣고 Lambda로 순차적으로 처리
 - 이 과정에서 동시에 X 명 예약 / 한 사람당 Y 초 만큼 예약 가능 / N 초 단위로 대기 숫자 갱신 구현
- 페스트푸드 음식점과 같은 구조







› 기본 아키텍처 요구사항

› 대기열 관리

- 대기열 관리는 SQS FIFO 큐로 처리

› 클라이언트와 실시간 통신

- API Gateway Websocket을 활용해 유저와 통신

› 데이터 저장 및 관리

- Connection ID 등 저장을 위해서 DynamoDB 사용

› 기타 로직 처리 및 비동기 처리

- 로직 처리는 당연히 Lambda / 대기 시간 처리등은 Stepfunctions 활용

› Live Demo

› Live Demo

- 백엔드/인프라: Serverless Framework(slsberry) 기반
- 프론트: React 기반 Static 웹사이트
- Full Source 코드 공유 예정!
- 선물 :
 - 선착순
 - 5번,15번 분에게 AWS 강의실 굿즈(키캡)
 - 2번,6번,4번,20번 분에게 기타 선물
- 주의
 - 대기시간 25초니깐 빠르게 입력하셔야 해요!
 - 새로고침 시 대기열 증가하니 새로고침 자제해주세요 ☺



> Live Demo

21:17 68%

Demo - 대기열

사전예약 등록

MessageGroupId:1
남은시간:18 초

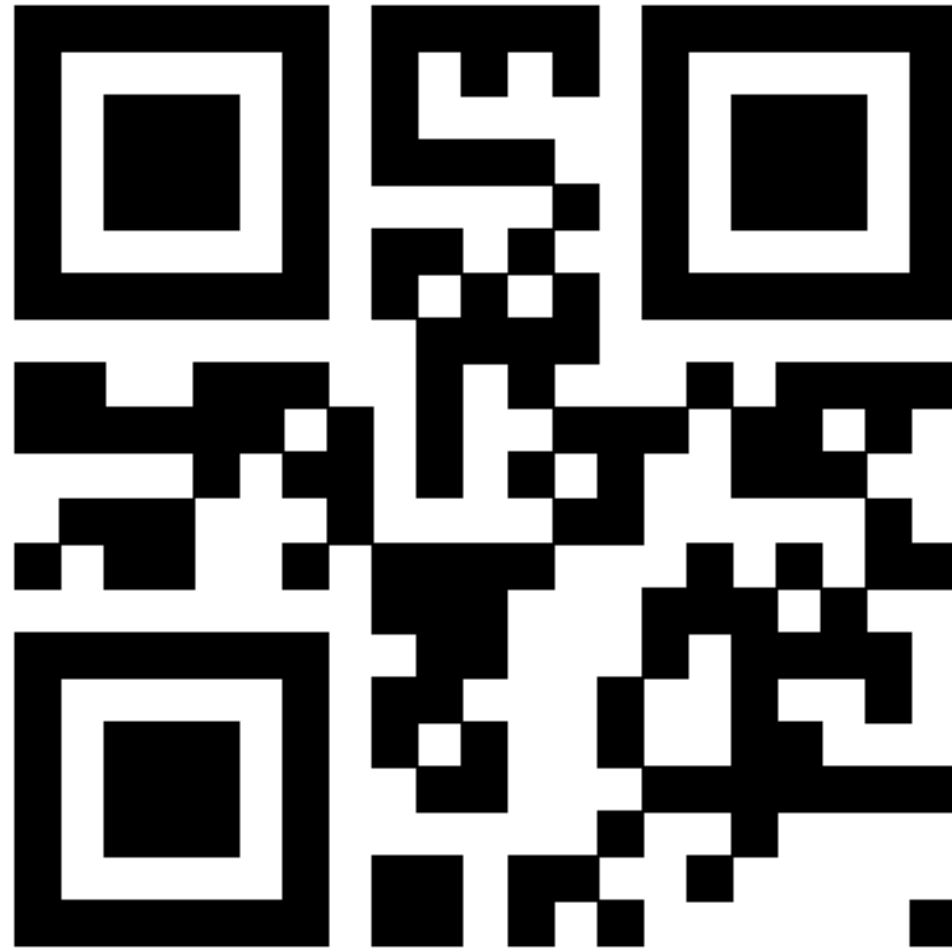
이메일을 입력하세요

등록

Items returned (3)

<input type="checkbox"/>	email ▾	date
<input type="checkbox"/>	test@email.com	2023-07-02 16:44:20
<input type="checkbox"/>	test	2023-07-02 16:55:34
<input type="checkbox"/>	spark@email.com	2023-07-02 21:17:32

› Live Demo



› AWS Lambda의 수행 방식

› AWS Lambda의 수행 방식

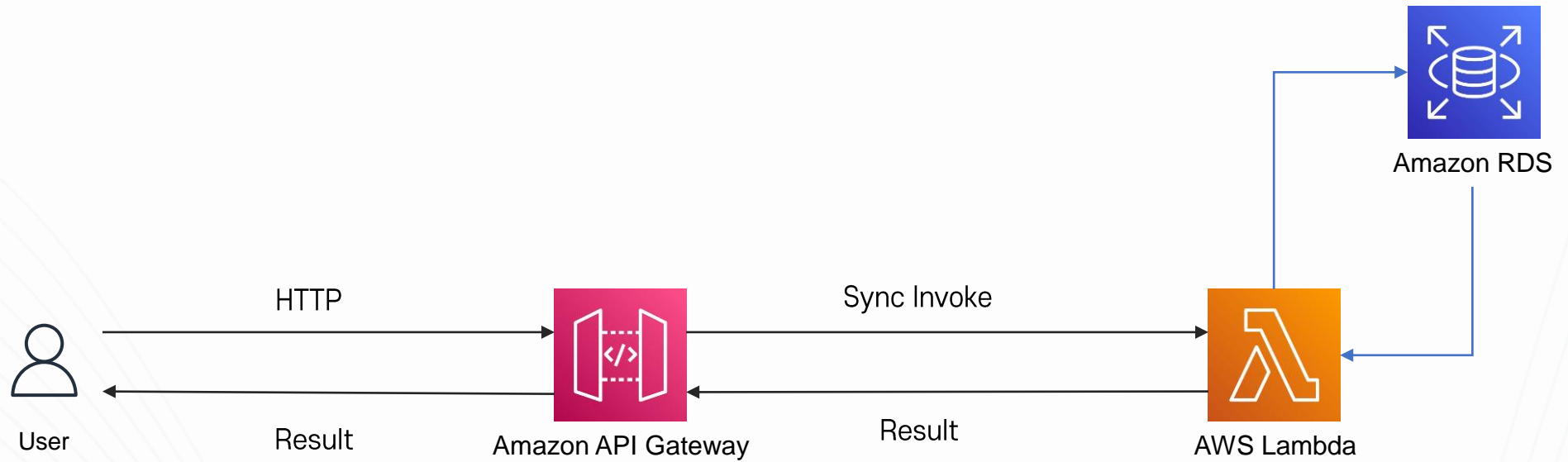
- Synchronous 호출
- Asynchronous 호출
- Event Source Mapping

› AWS Lambda의 수행 방식 - Synchronous 호출

› AWS Lambda의 수행 방식 - Synchronous 호출

- AWS SDK/CLI/Console 등으로 직접 AWS Lambda를 호출
- Sync 호출이기 때문에 작업 수행 이후 결과값이 반환될 때 까지 기다림
- 사용 사례:
 - API Gateway
 - 테스트 함수 등

> AWS Lambda의 수행 방식 - Synchronous 호출

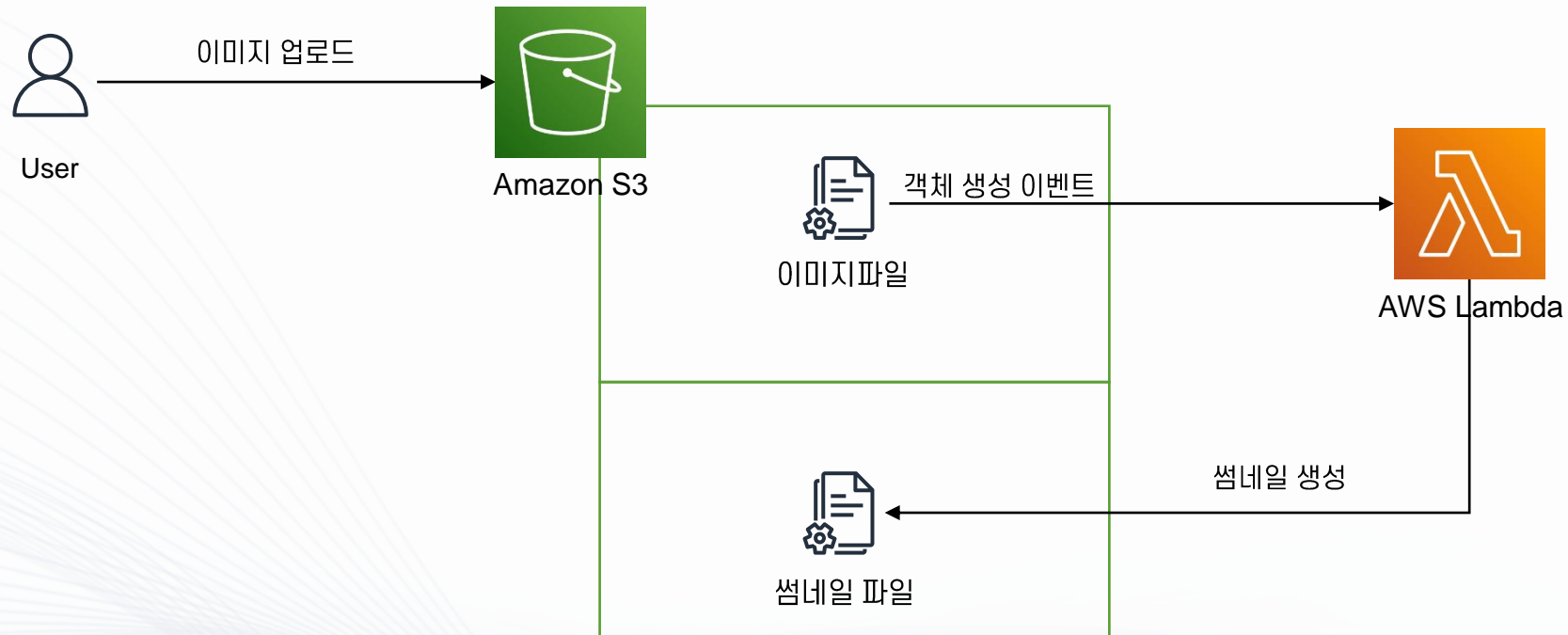


› AWS Lambda의 수행 방식 - Async 호출

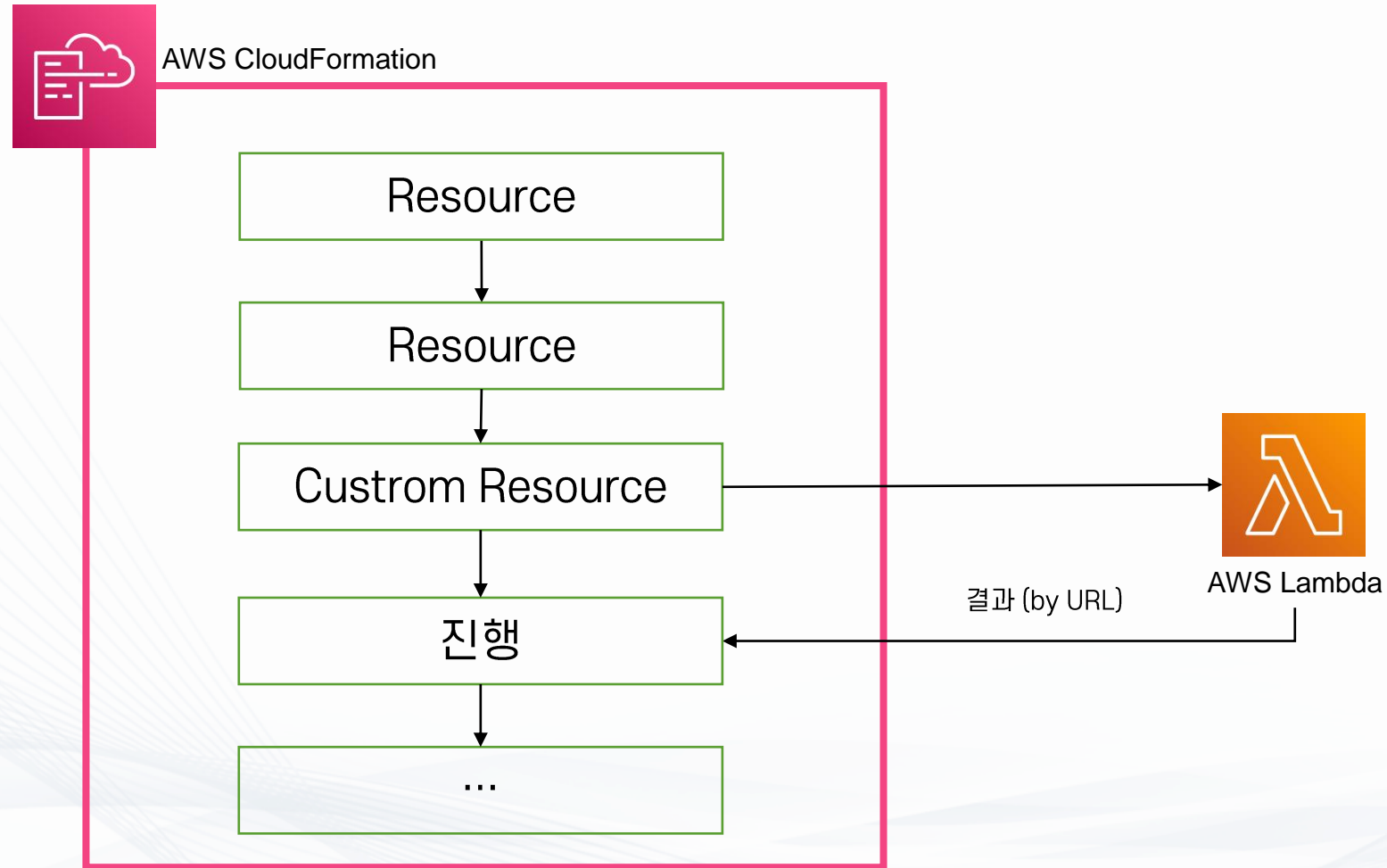
› AWS Lambda의 수행 방식 - Synchronous 호출

- AWS 서비스 등이 특정 이벤트에 반응해서 AWS Lambda를 호출
- Async 호출이기 때문에 작업 수행 이후 결과값 반환에 관심이 없음
- 사용 사례:
 - Amazon S3 Trigger
 - Amazon Eventbridge 이벤트
 - Amazon IOT Core 이벤트 등

> AWS Lambda의 수행 방식 - Asynchronous 호출



> AWS Lambda의 수행 방식 - Asynchronous + Synchronous

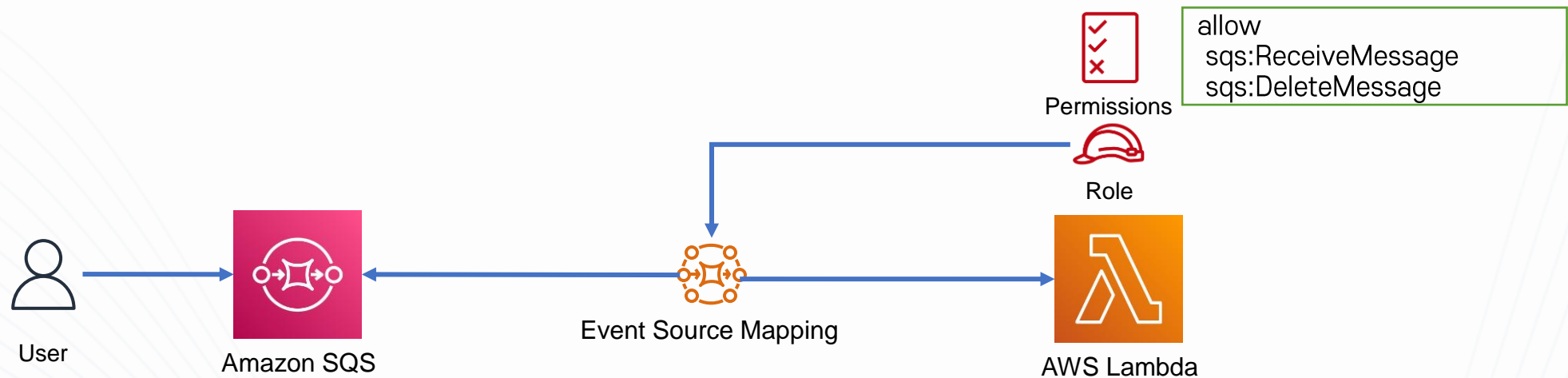


› AWS Lambda의 수행 방식 - Event Source Mapping

› AWS Lambda의 수행 방식 - Event Source Mapping

- Event Source Mapping : 대상에서 데이터를 가져와 Lambda를 호출해주는 리소스
 - 일종의 중계인
 - **Lambda의 권한(Execution Role)을 활용해서 대상의 메시지를 확보**
 - 즉 Lambda자체에서 대상 리소스에 대한 권한이 필요함
 - 주요 대상 서비스
 - Amazon DynamoDB
 - Amazon Kinesis
 - Amazon SQS
 - Amazon MSK

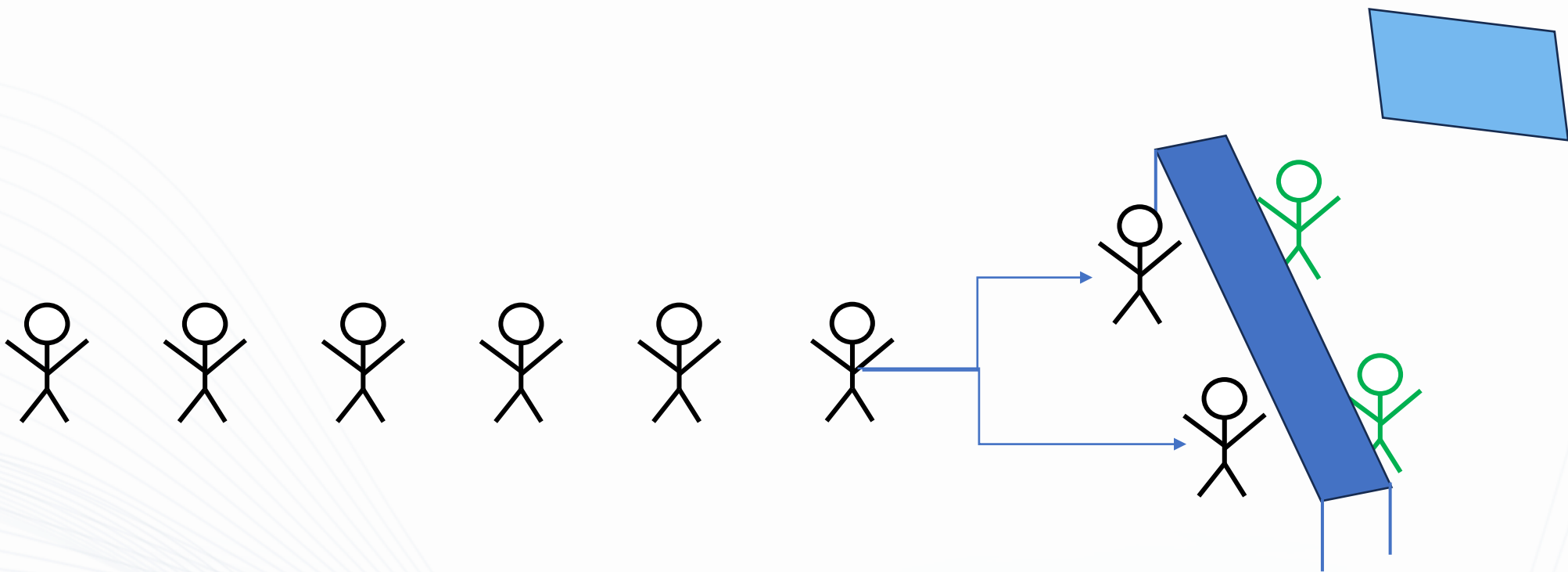
> AWS Lambda의 수행 방식 - Event Source Mapping



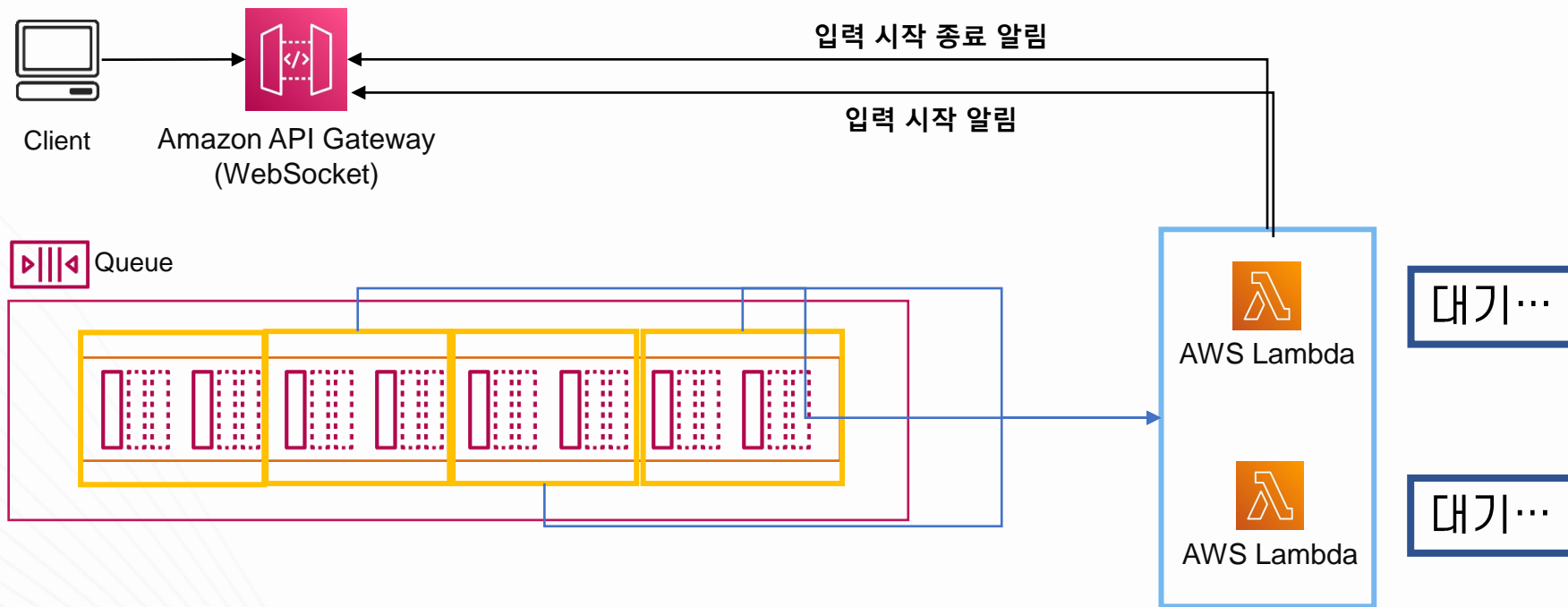
> “동시에 x 명만 예약 가능”

> “동시에 x 명만 예약 가능”

- SQS를 매인 대기열로 정하고 x개의 메시지만 가져와서 처리
- 두 가지 방법
 - 대기 시간이 15분 이하일 경우
 - Lambda 수행 시간이 15분이기 때문에 Lambda 에서 대기 처리
 - Event Source Mapping의 maximumConcurrency Property로 해결
 - SQS에서 Lambda가 처리하는 최대 동시 수행 제한



> 아키텍처

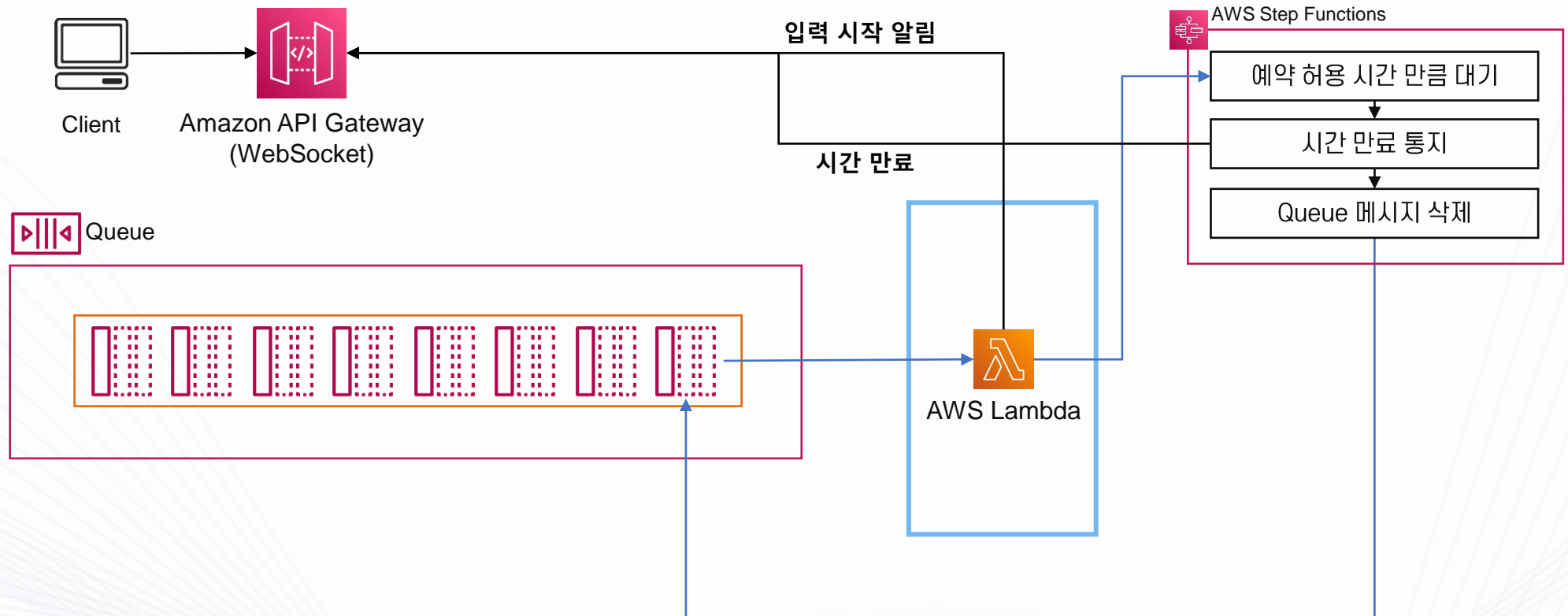


> “동시에 x 명만 예약 가능”

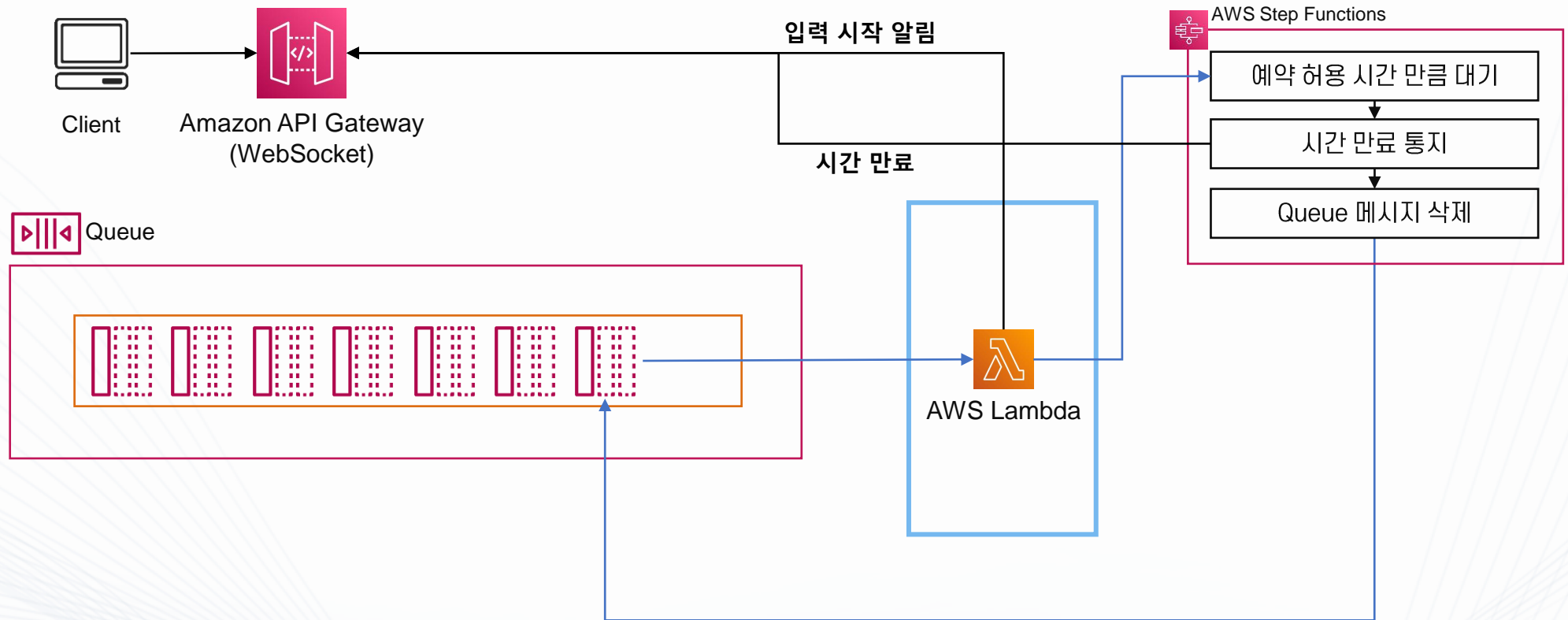
> “동시에 x 명만 예약 가능”

- SQS를 매인 대기열로 정하고 x개의 메시지만 가져와서 처리
- 두 가지 방법
 - 대기 시간이 15분 이하일 경우
 - Lambda 수행 시간이 15분이기 때문에 Lambda 에서 대기 처리
 - maximumConcurrency Property로 해결(SQS에서 Lambda가 처리하는 최대 동시 수행 제한)
 - 대기 시간이 15분 이상일 경우
 - Stepfunction을 활용해서 대기 시간 이후 SQS 큐 메시지 삭제

> 아키텍처



> 아키텍처



> “동시에 x 명만 예약 가능”

> “동시에 x 명만 예약 가능”

- SQS를 매인 대기열로 정하고 x개의 메시지만 가져와서 처리
- 두 가지 방법
 - 대기 시간이 15분 이하일 경우
 - Lambda 수행 시간이 15분이기 때문에 Lambda 에서 대기 처리
 - maximumConcurrency Property로 해결(SQS에서 Lambda가 처리하는 최대 동시 수행 제한)
 - 대기 시간이 15분 이상일 경우
 - Stepfunction을 활용해서 대기 시간 이후 SQS 큐 메시지 삭제

› SQS 메시지의 삭제 로직

› SQS와 Lambda

- SQS에서 Lambda를 직접 트리거할 경우 Lambda가 종료될 때 메시지 자동 삭제
 - 즉 Lambda가 삭제하지 않더라도 Event Source Mapping이 삭제

> SQS 메시지의 삭제 로직

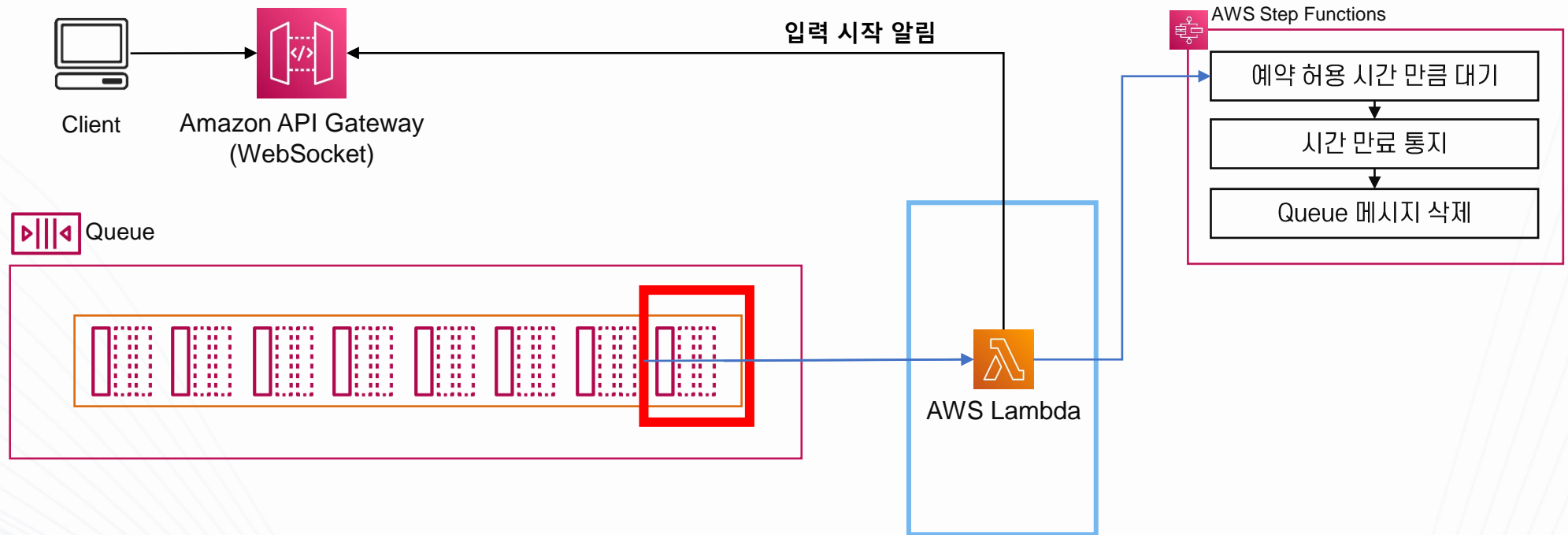
When Lambda reads a batch, the messages stay in the queue but are hidden for the length of the queue's visibility timeout. If your function successfully processes the batch, Lambda deletes the messages from the queue. By default, if your function encounters an error while processing a batch, all messages in that batch become visible in the queue again. For this reason, your function code must be able to process the same message multiple times without unintended side effects. You can modify this reprocessing behavior by including batch item failures in your function response.

› SQS 메시지의 삭제 로직

› SQS와 Lambda

- SQS에서 Lambda를 직접 트리거할 경우 Lambda가 종료될 때 메시지 자동 삭제
 - 즉 Lambda가 삭제하지 않더라도 Event Source Mapping이 삭제
- 문제점
 - 우리는 Lambda에서 메시지를 삭제하지 않고 StepFunctions에서 대기 후 삭제 예정
 - 하지만 Stepfunction실행 후 Lambda가 종료되면 메시지가 삭제됨

> 아키텍처



› SQS 메시지의 삭제 로직

› SQS와 Lambda

- SQS에서 Lambda를 직접 트리거할 경우 Lambda가 종료될 때 메시지 자동 삭제
 - 즉 Lambda가 삭제하지 않더라도 Event Source Mapping이 삭제
- 문제점
 - 우리는 Lambda에서 메시지를 삭제하지 않고 StepFunctions에서 대기 후 삭제 예정
 - 하지만 Stepfunction실행 후 Lambda가 종료되면 메시지가 삭제됨
 - 해결 방법(꿀팁): sqs:DeleteMessage Deny!
 - Event Source Mapping은 Lambda의 권한을 활용하기 때문에 권한이 없으면 삭제 불가능
 - 단 처음에 프로비전시 sqs:DeleteMessage 권한이 없으면 프로비전이 안되기에 프로비전 후 역할 수정

> SQS와 Lambda

```
Version: '2012-10-17'
Statement:
  - Effect: Allow
    Action:
      - s3:*
      - sqs:*
      - logs:*
      - states:*
      - execute-api:*
    Resource: '*'
  - Effect: Deny
    Action:
      - sqs:DeleteMessage
    Resource:
      Fn::GetAtt:
        - WaitingQueue
        - Arn
```

> SQS와 Lambda

```
Error:
UPDATE_FAILED: SqsUnderscorehandleUnderscorewaitingUnderscoremessageEventSourceMappingSQSWaiti
ngQueue (AWS::Lambda::EventSourceMapping)
Resource handler returned message: "Invalid request provided: The provided execution role does
not have permissions to call DeleteMessage on SQS (Service: Lambda, Status Code: 400, Request
ID: 6d2b5e91-1d7b-405a-9f9c-c81f5bb2c2fd, Extended Request ID: null)" (RequestToken: e563f052
-5f51-b089-85da-34d441d41f12, HandlerErrorCode: InvalidRequest)
```

> “동시에 x 명만 예약 가능”

> “동시에 x 명만 예약 가능”

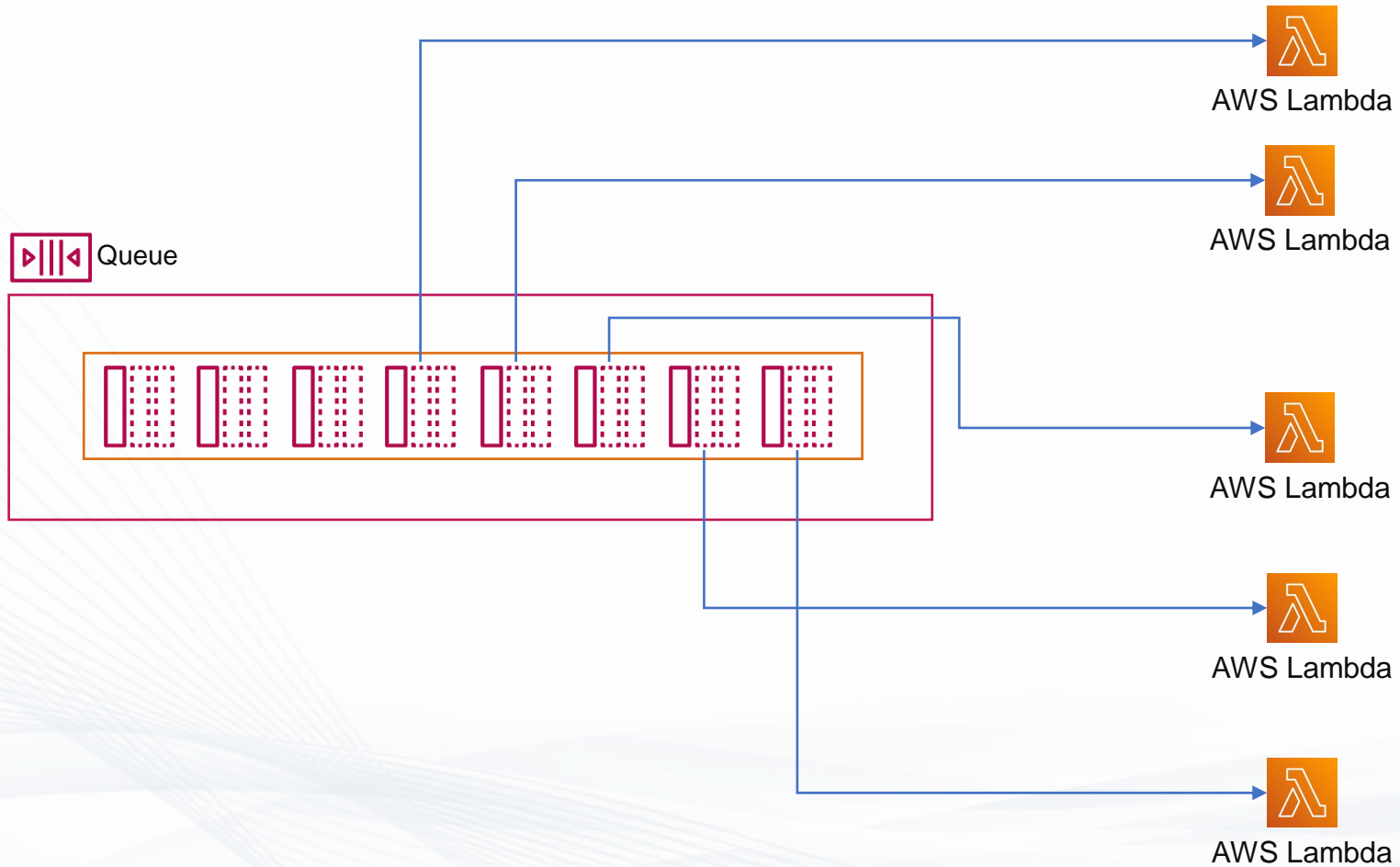
- StepFunctions 사용시 동시에 x명 예약 가능 구현?
 - FIFO Queue의 MessageGroupId 활용

› SQS의 MessageGroupId

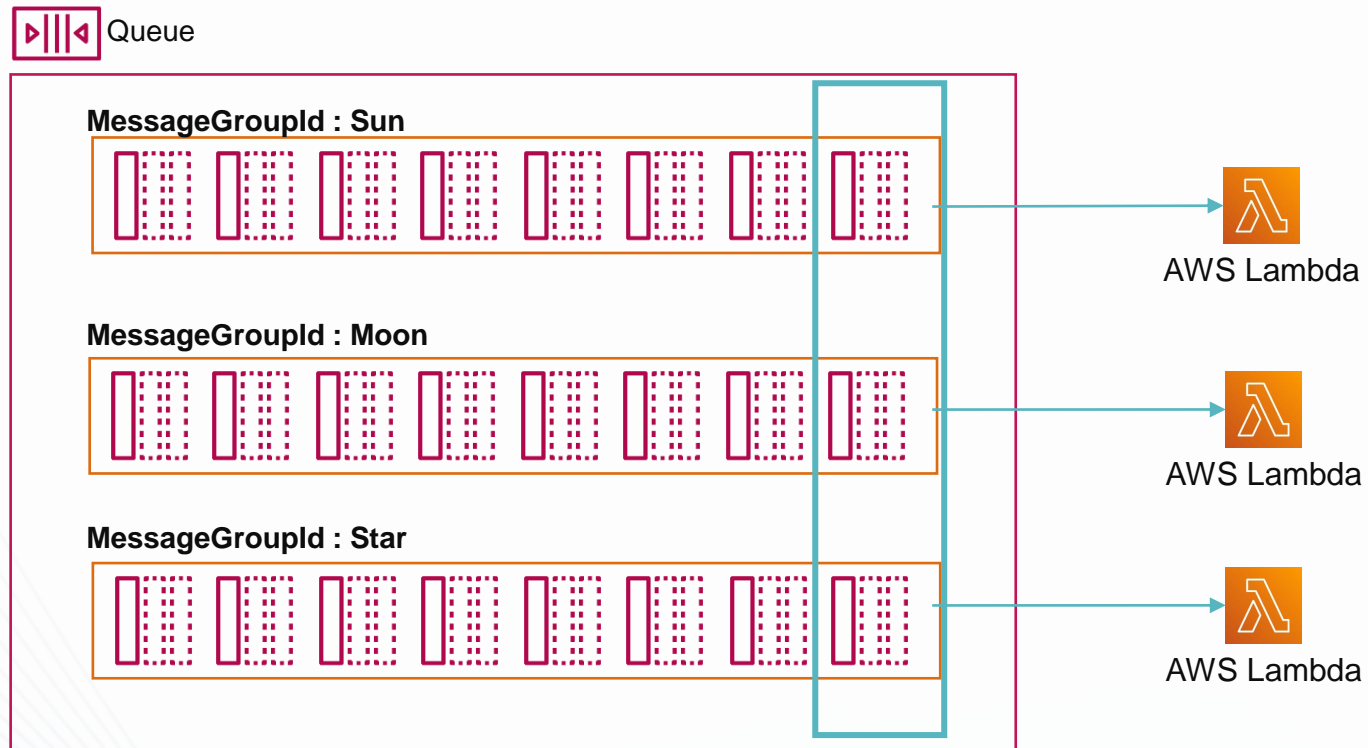
› SQS의 MessageGroupId

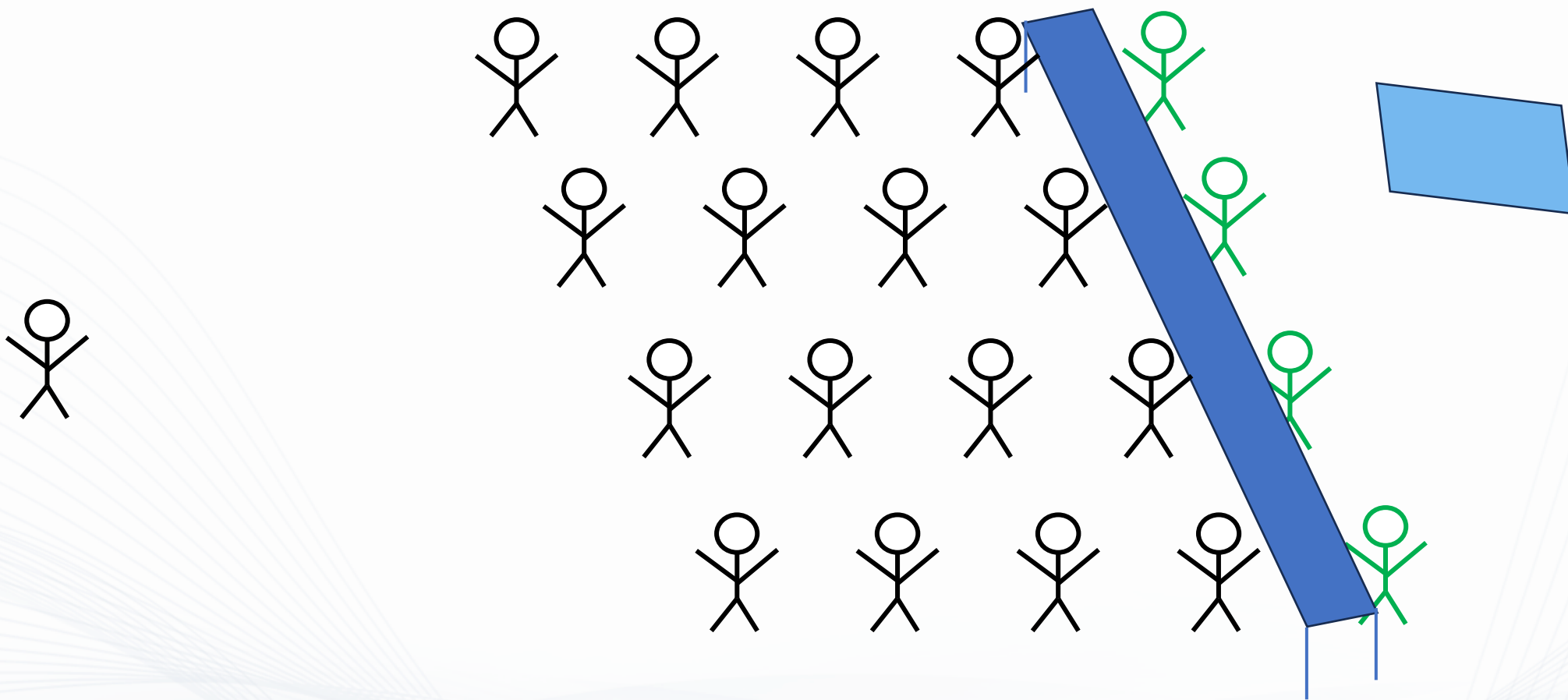
- FIFO모드에서만 동작하는 일종의 SQS안의 채널
- 같은 MessageGroupId는 동시에 한번만 처리 가능
 - 즉 아무리 Consumer가 많아도 현재 특정 MessageGroupId가 처리중이라면, 나머지 같은 MessageGroupId를 가진 쌓여있는 메시지는 Receive 불가능
- 즉 MessageGroupId의 종류 만큼만 동시에 처리 할 수 있도록 제어 가능

> 아키텍처

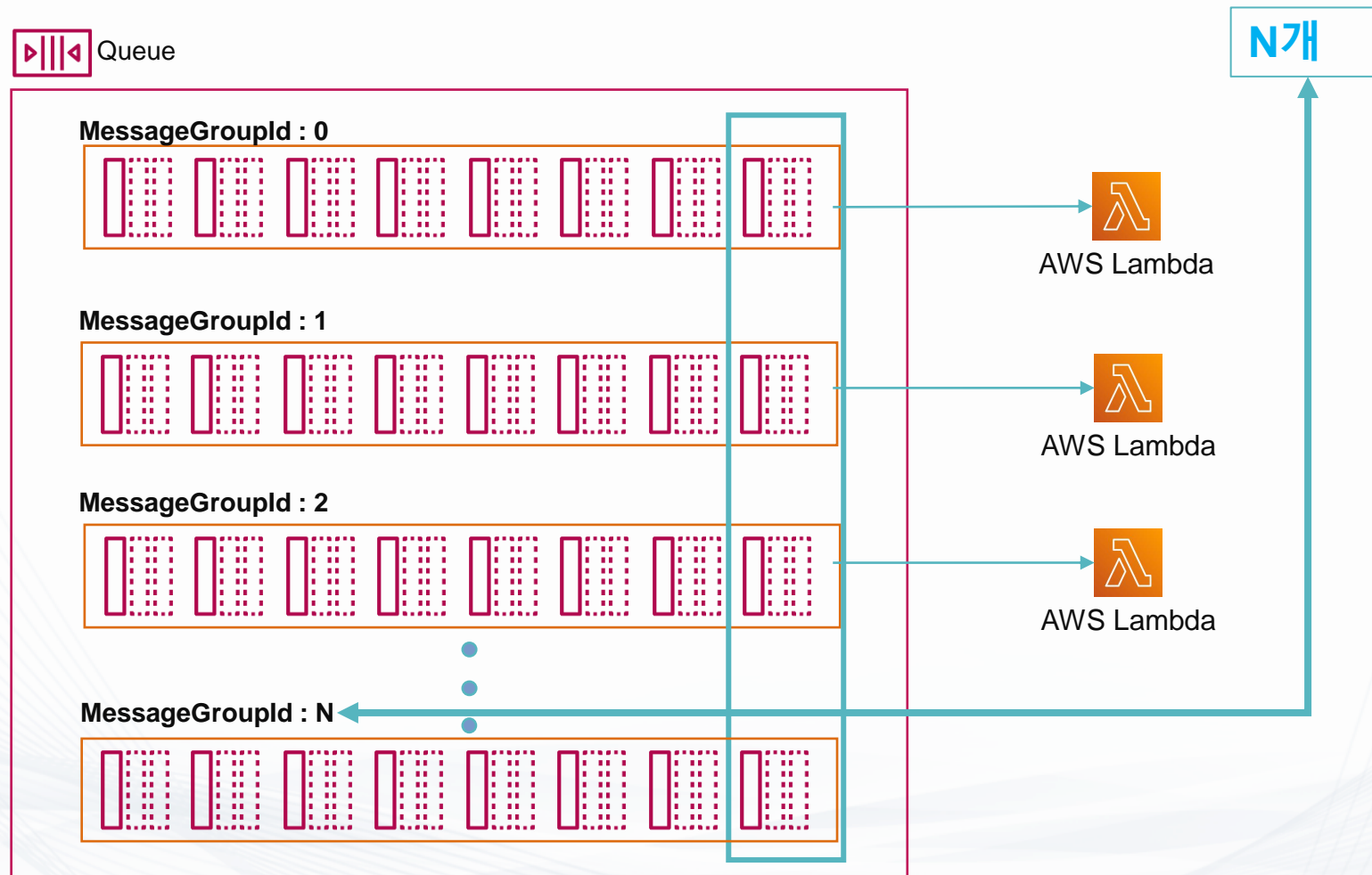


> 아키텍처





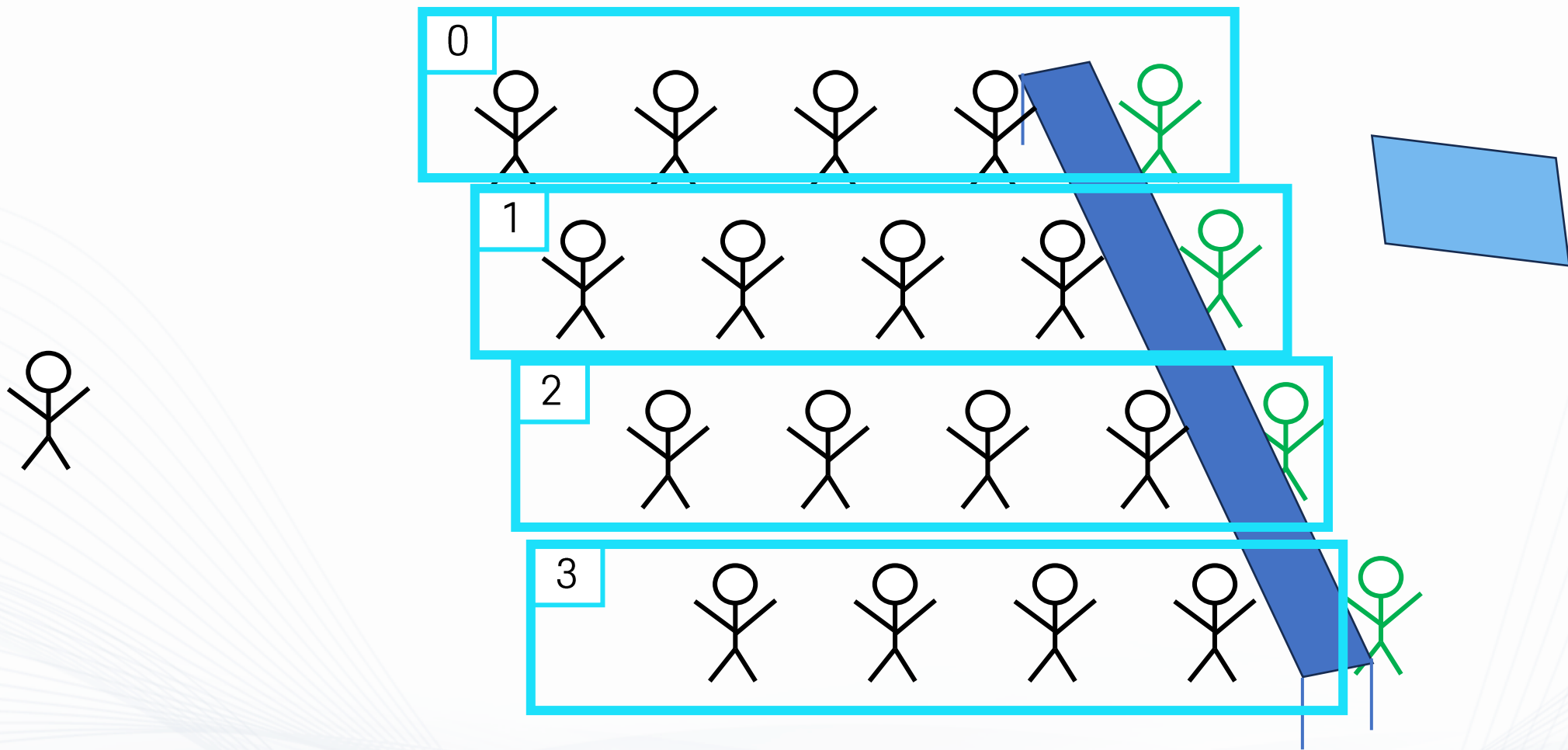
> 아키텍처



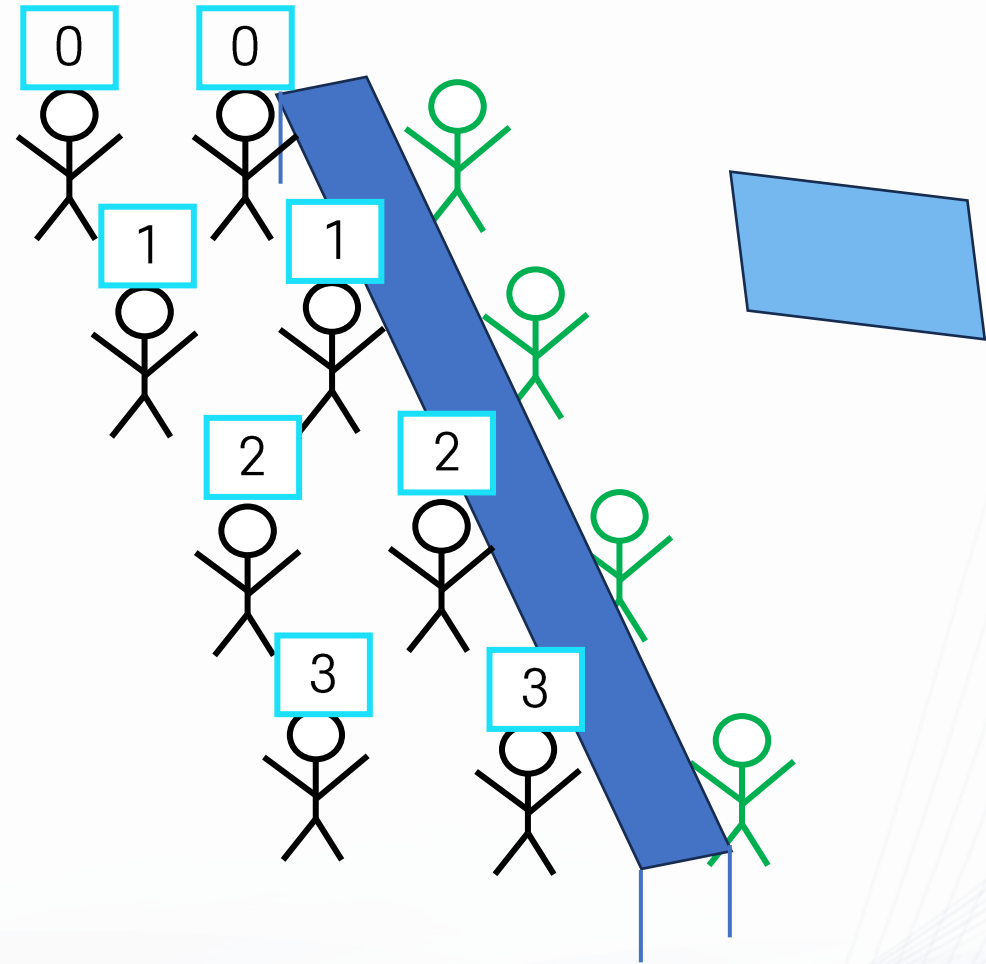
> “동시에 x 명만 예약 가능”

> “동시에 x 명만 예약 가능”

- 한번에 처리 가능한 유저 숫자를 정하고, 들어오는 순서대로 MessageGroupID를 부여해서 SQS에 넣기



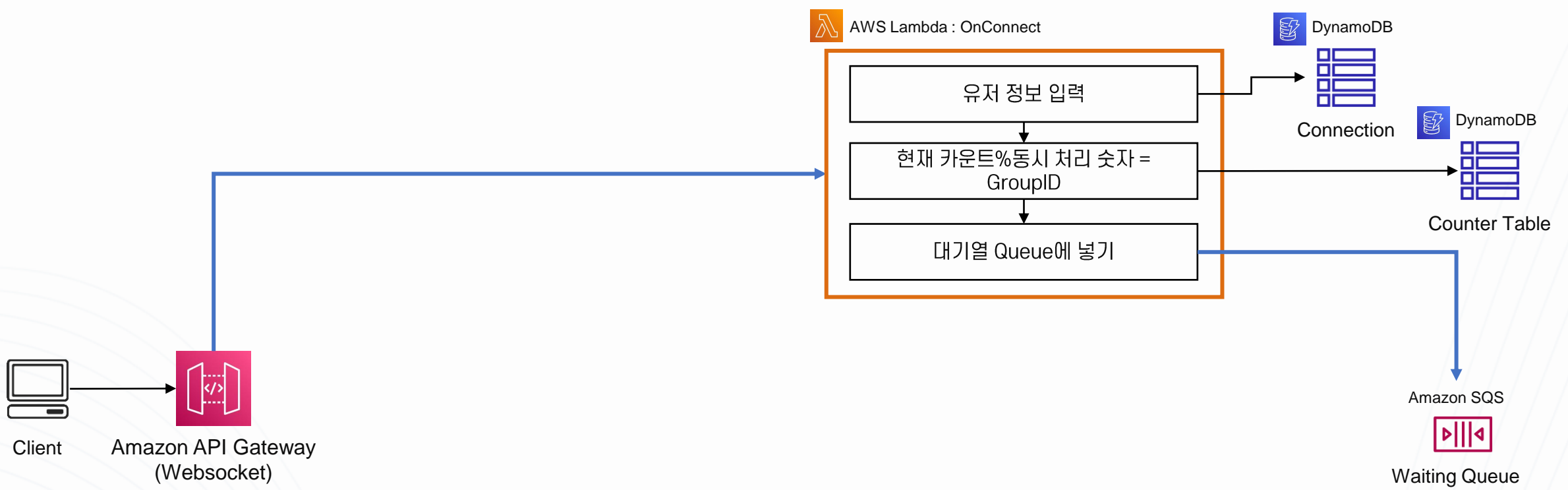
MessageGroupId = 순서%4



> “동시에 x 명만 예약 가능”

> “동시에 x 명만 예약 가능”

- 한번에 처리 가능한 유저 숫자를 정하고, 들어오는 순서대로 MessageGroupId를 부여해서 SQS에 넣기
 - 유저 카운터는 DynamoDB에 저장(Increment)
 - MessageGroupId = 유저 카운터%(한번에 처리할 숫자)
 - 예: 한번에 처리할 숫자가 5라면, 256번째 유저는 MessageGroupId = 1
- 유저 액션이 끝나면 StepFunctions가 Queue에서 메시지 삭제-> 다음 MessageGroupId에 해당하는 대기열 처리 시작

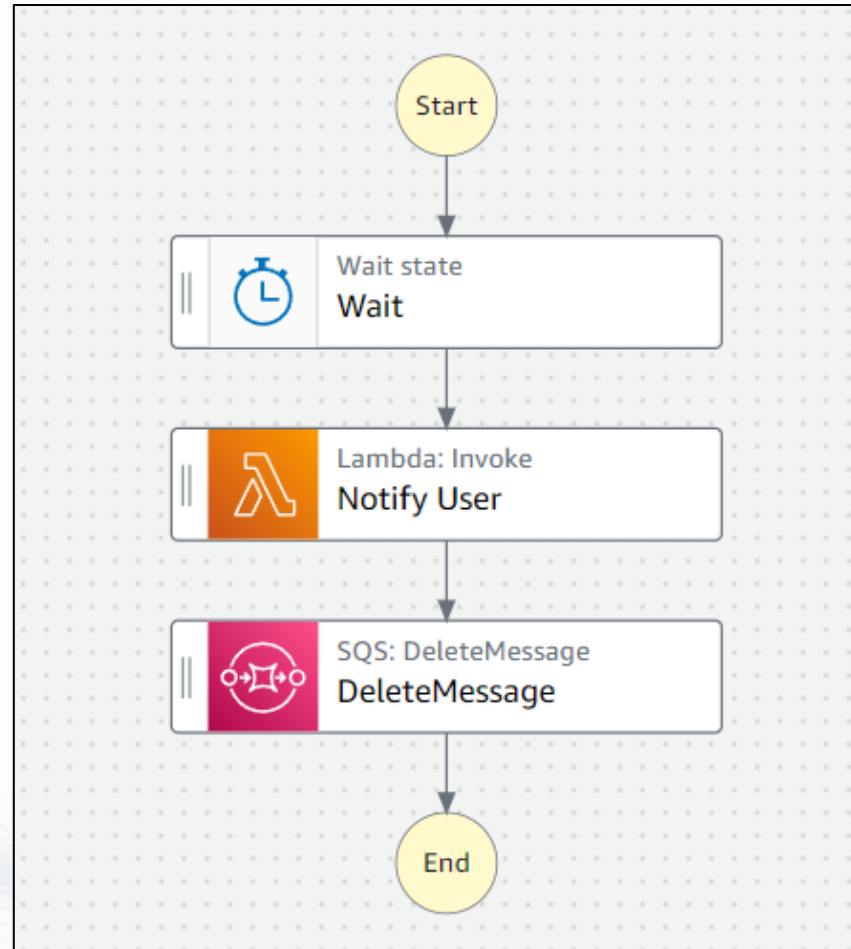


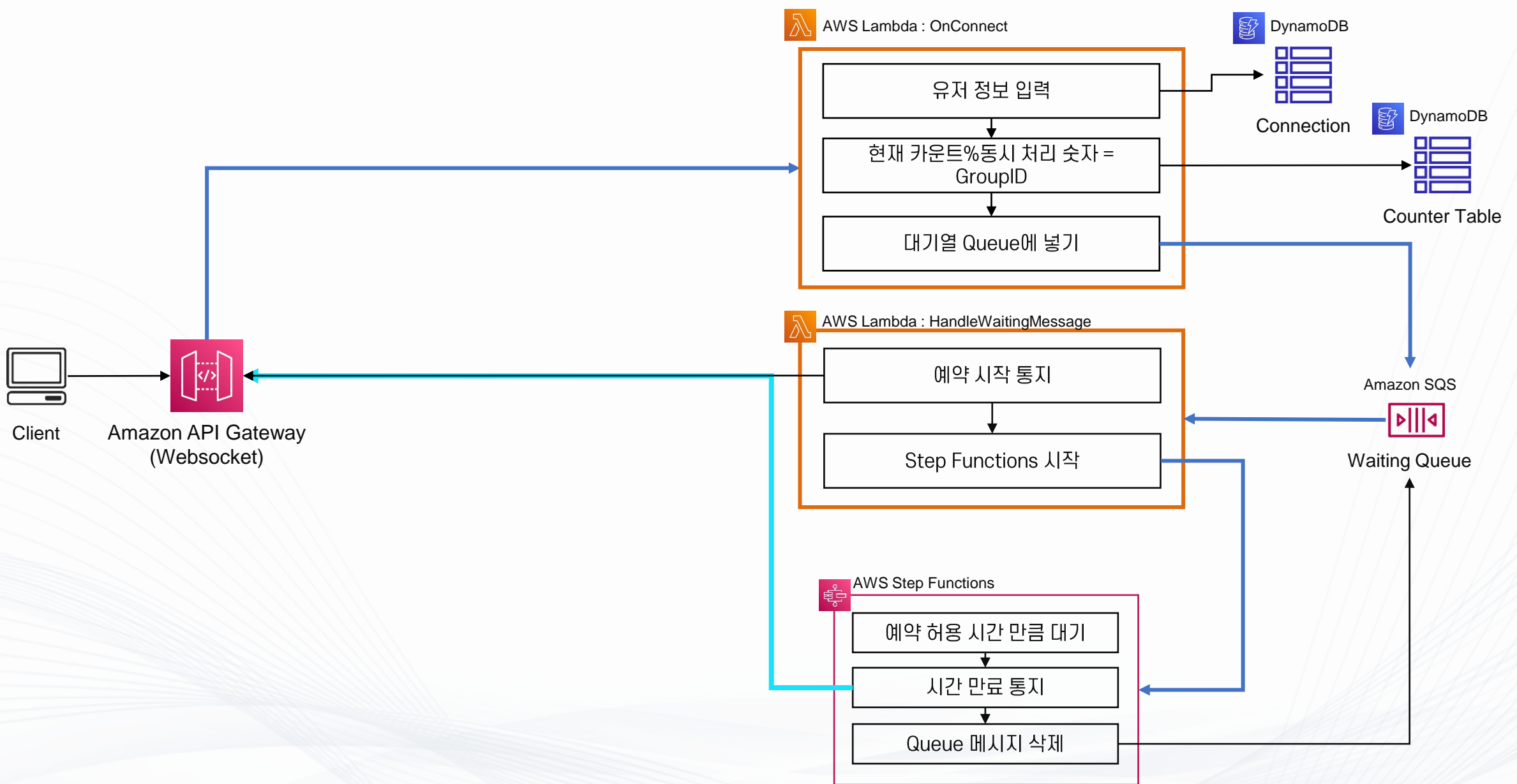
› “한 사람당 Y초 만큼 예약 가능”

› “한 사람당 Y초 만큼 예약 가능”

- 처음 Queue메세지를 처리할 때(대기열이 돌아왔을 때) 시작 통지
- 일정 기간 동안 기다린 후, 서버에서 알림 전달
 - Lambda의 실행 시간은 최대 15분
 - 이전에 언급한 대로 15분 이하면 그냥 Lambda에서 대기
 - 15분 이상이라면
 - Step Functions
 - Step Functions의 역할
 - Y초동안 대기
 - 이후 유저에게 만료 알림(Websocket)
 - 이후 SQS 메시지 삭제

> “한 사람당 Y시간 만큼 예약 가능”





› “5초 단위로 대기 중인 유저 숫자 알림”

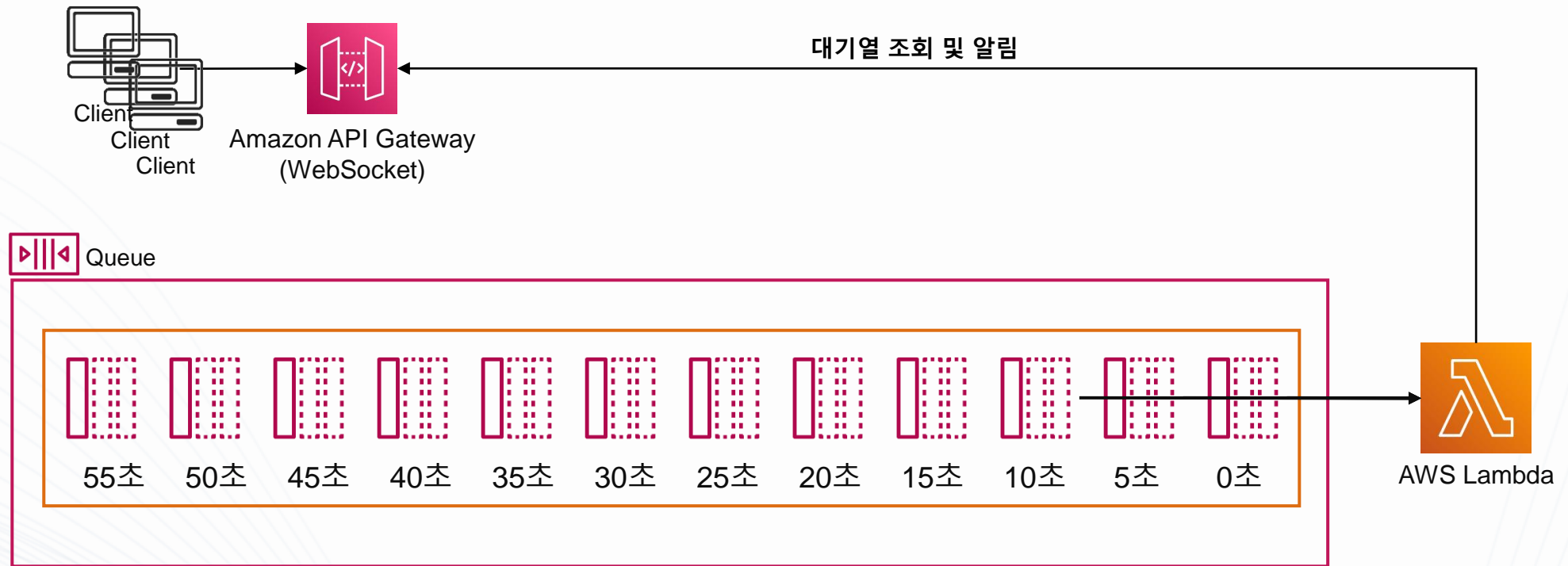
› “5초 단위로 대기 중인 유저 숫자 알림”

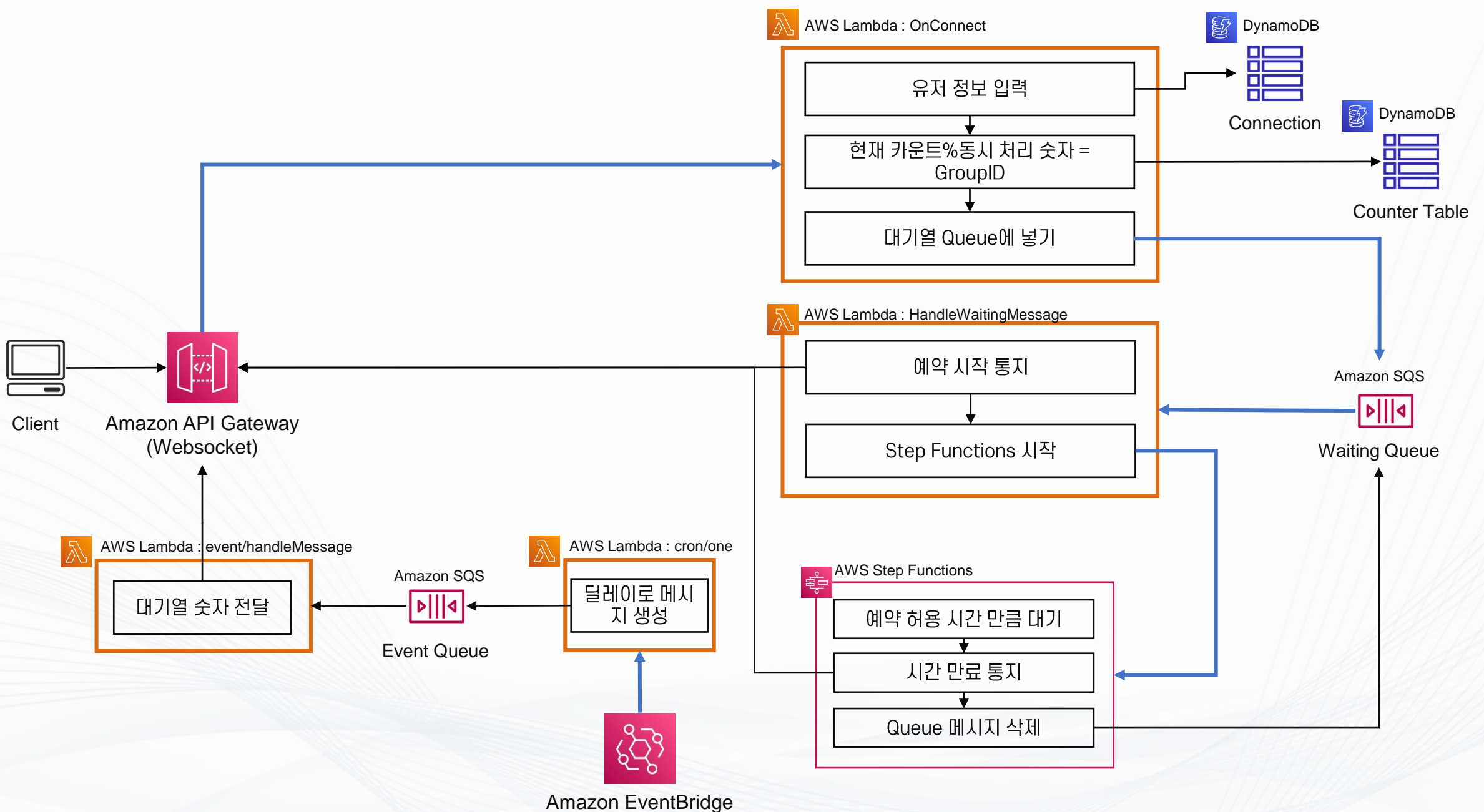
- EventBridge Cron 이벤트로 일정 시간마다 대기열 숫자(SQS ApproximateNumberOfMessages) 전달
- 문제점 : EventBridge Cron 이벤트는 최소 단위가 1분단위
 - 해결방법: SQS의 DelaySeconds 활용
- DelaySeconds: x초 후에 메시지 처리 가능
 - 즉 1분 마다 0,5,10,15,20,25,30,35,40,45,50,55 초 딜레이를 가진 메시지 12개를 Queue에 넣어 처리
 - 해당 메시지가 처리될 때 마다 모든 유저에게 현재 대기열 숫자 업데이트

> “5초 단위로 대기 중인 유저 숫자 알람”

```
console.log(event)
var sqs = new AWS.SQS({ apiVersion: '2012-11-05' });
var params = {
  |   MessageBody: JSON.stringify({ notification: "test" }),
  |   QueueUrl: process.env.event_queue_url
  | }
  //Interval만큼 SQS Queue에 메세지 생성
  let interval = process.env.notification_interval;
  for (var i = 0; i < (60 / interval); i++) {
    |   if (i !== 0) {
    |       |   params["DelaySeconds"] = i * interval;
    |       |   }
    |       try {
    |       |   await sqs.sendMessage(params).promise();
    |       |   }
    |       catch (e) {
    |       |   console.log(e);
    |       |   }
    |   }
  }
```

> SQS의 DelaySeconds

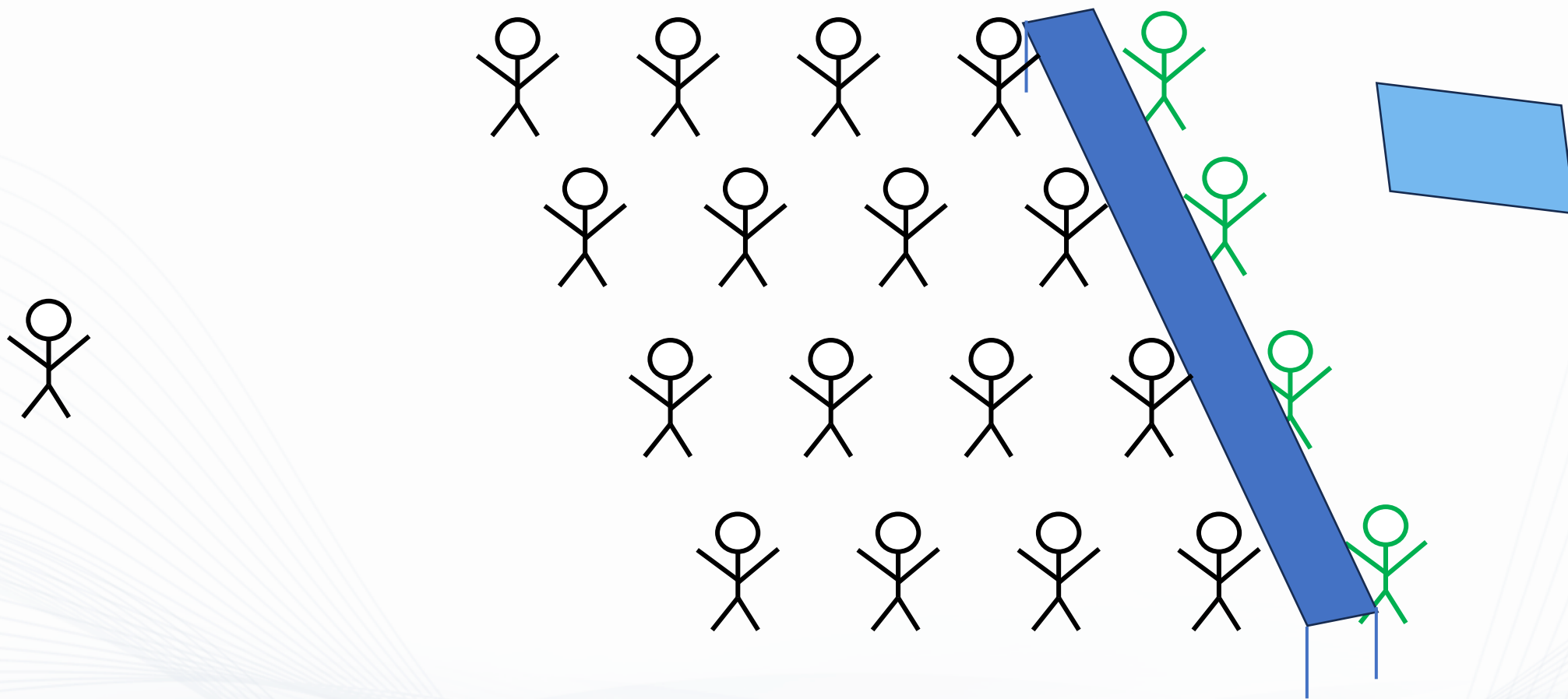




› 생각해 볼 점

› 생각해 볼 점

- 이 아키텍처대로라면 완전히 FIFO는 아님
 - MessageGroupID에 따라 순서 변동 가능



› 생각해 볼 점

› 생각해 볼 점

- 이 아키텍처대로라면 완전히 FIFO는 아님
 - MessageGroupID에 따라 순서 변동 가능
- 15초 이후에 유저가 강제로 이메일을 입력했다면?
 - 별도로 체크 로직 필요
- 기타 최적화 (예: SQS 대기열 메트릭 대신 실제 레코드, 실제 대기시간 계산 등)

> 소스코드

> 소스코드

- Github : <https://github.com/spark323/slsberry>
- slsberry 라는 자체 framework로 warpping되어 있음
 - api spec 정의하면 자동으로 serverless.yml 만들어주고 openapi export 해주고 notion정리도 해주고...



› Serverless Framework



“All-in-one development & monitoring of auto-scaling apps on AWS Lambda”

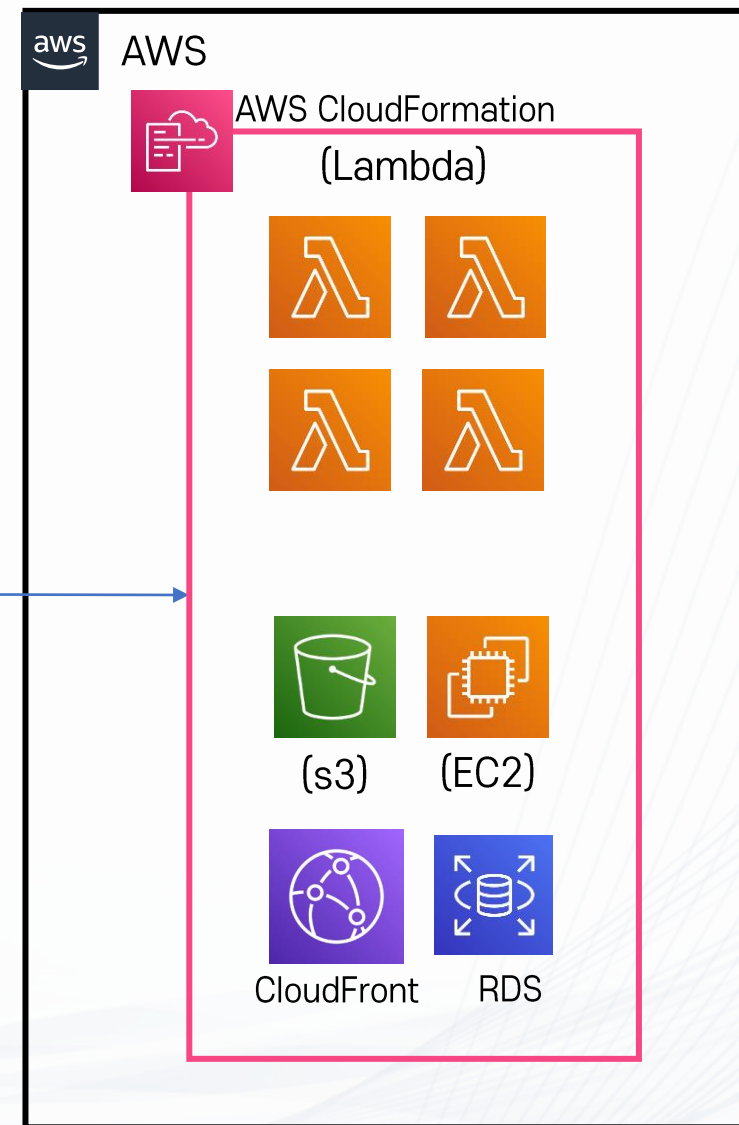


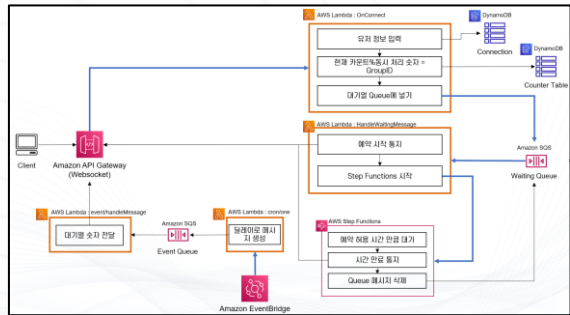
Lambda Source Code

AWS 인프라
(CloudFormation)

serverless.yml

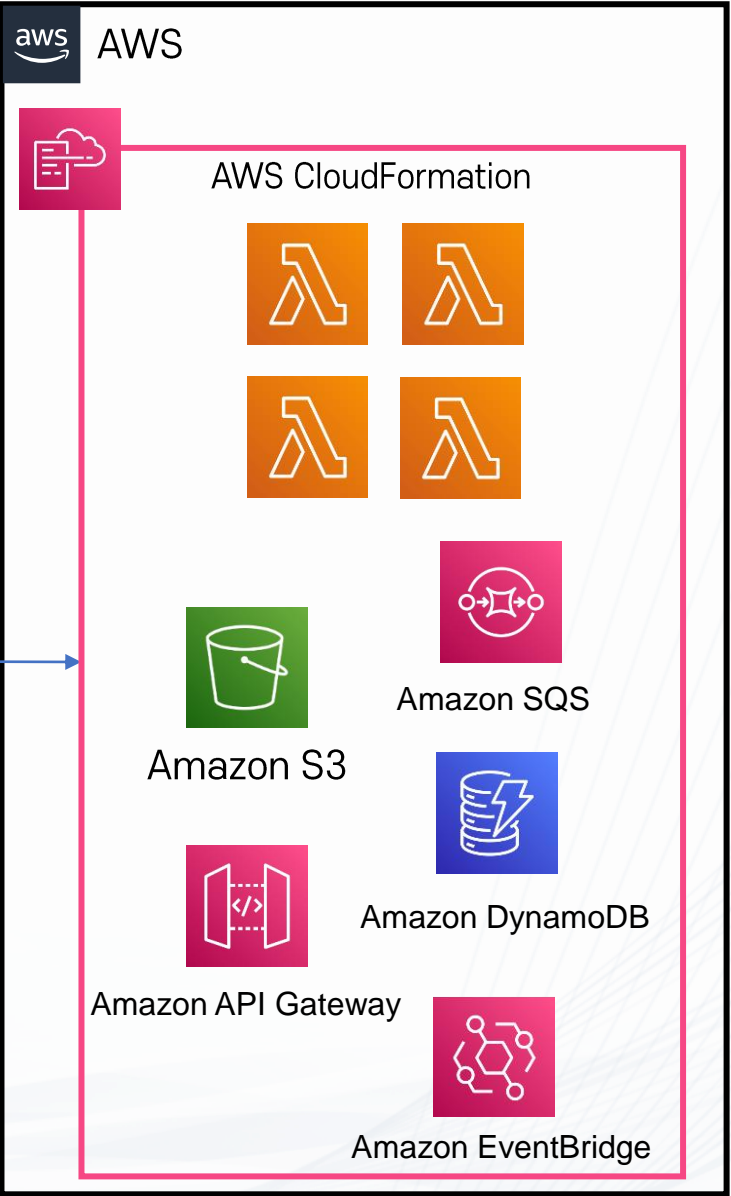
serverless



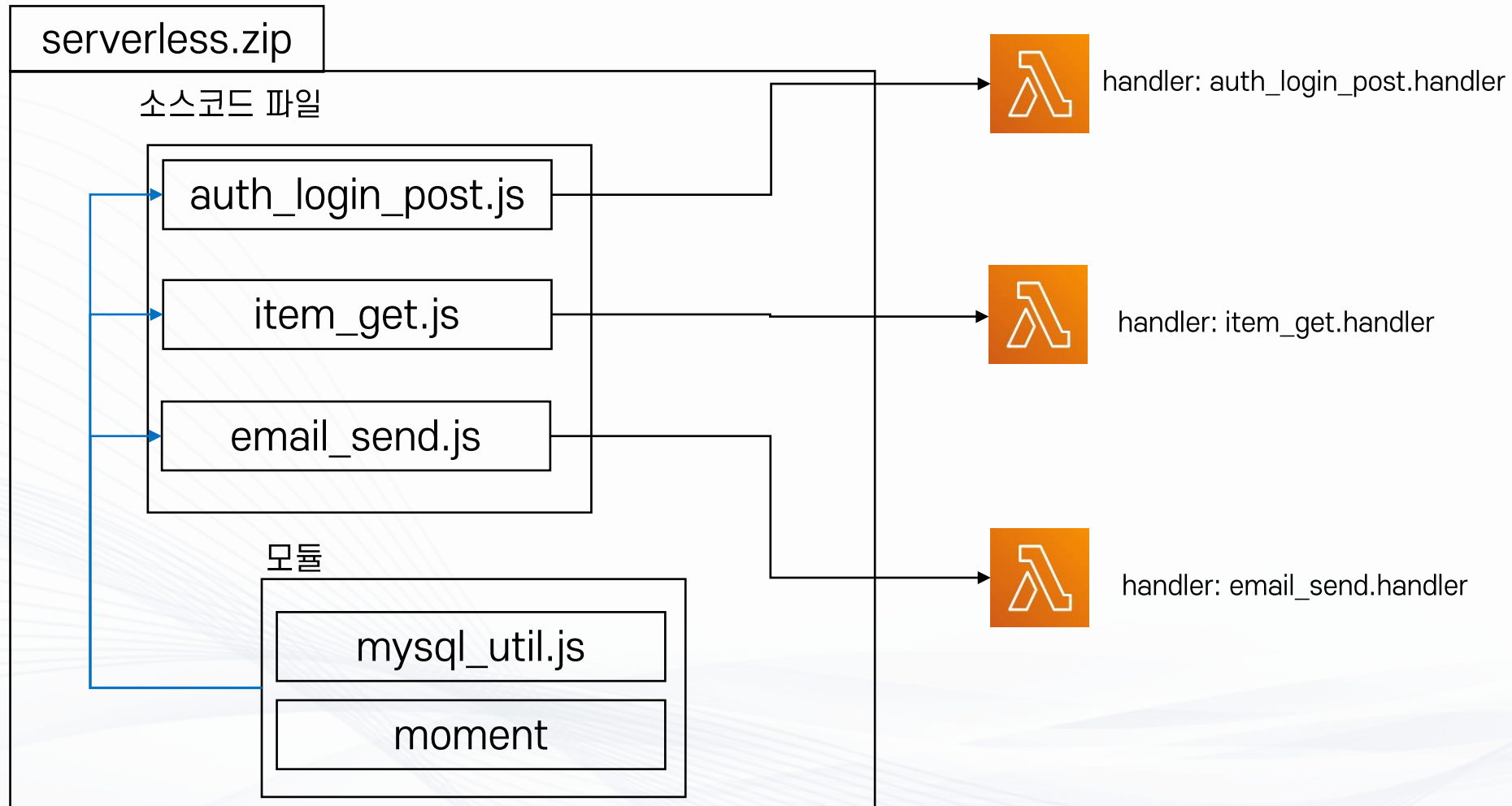


serverless.yml

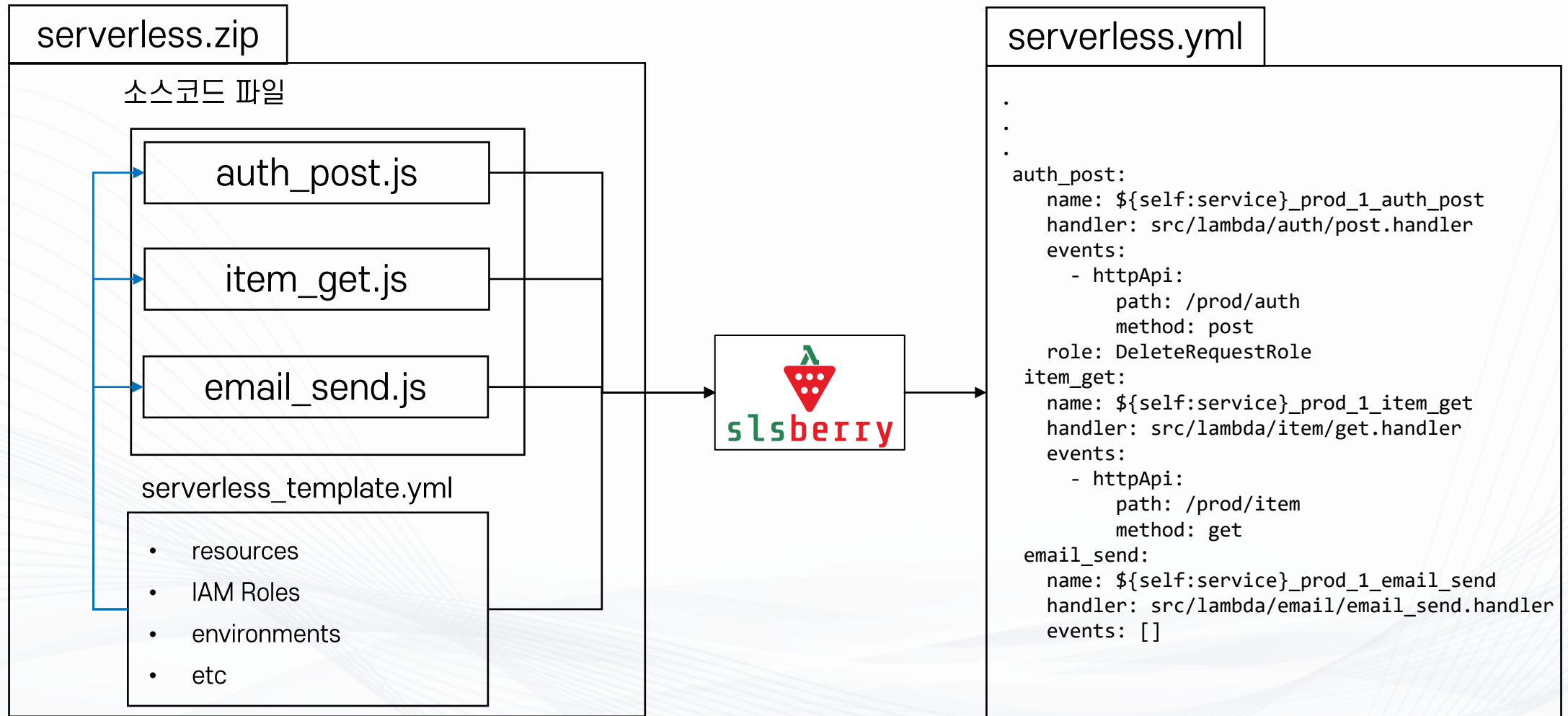
serverless



> slsberry



> slsberry



› slsberry 기능 Demo

› slsberry 기능 Demo

- serverless.yml 생성
- 로컬 Lambda Test
- API Export



› 참고 자료

› 참고 자료

- Demo
 - Github
 - backend: <https://github.com/spark323/demo-waiting-queue-backend.git>
 - frontend: <https://github.com/spark323/demo-waiting-queue-frontend.git>
 - rwlecture.com
 - 코드 “slsdemo-311” 입력 (backend/frontend/PDF 포함)

> 참고 자료

RubyWave 강의자료 다운로드

로그 아웃

slsdemo-311

다운로드

› 참고 자료

› 참고 자료

- Demo
 - Github
 - backend: <https://github.com/spark323/demo-waiting-queue-backend.git>
 - frontend: <https://github.com/spark323/demo-waiting-queue-frontend.git>
 - rwlecture.com
 - 코드 “slsdemo-311” 입력 (backend/frontend/PDF 포함)
- 기타
 - slsberry
 - <https://github.com/spark323/slsberry.git>



rubywave.io

spark@rubywave.io