Session 03 핸즈온 가이드 (Windows)

> 이 문서는 **Windows PowerShell** 용이다. 두 문서는 **단계 번호와 확인 내용이 같고 명령만 다르다.**

개요

행사 신청 서비스를 동기 처리에서 비동기 처리로 바꾸고, 중복과 실패를 직접 관찰한다. **배포는 명령어로, 확인은 신청 화면과 AWS 콘솔로** 진행한다. 배포는 각 발표의 개념 설명이 시작될 때 미리 실행한다.

실습	시간	내용
실습 1 · 동기	12분	요청 처리와 저장 확인, 발급 지연 관찰
실습 2 · 비동기	23분	접수와 발급 분리, 중복 방지, 실패 복구

모든 명령은 `source.zip` 의 압축을 푼 `source/` 폴더에서 실행한다. 콘솔 리전은 기본값 **서울(** `ap-northeast-2` **)**로 맞춘다. 영어 콘솔 메뉴를 기준으로 안내한다.

AWS 콘솔 화면 언어

콘솔이 한국어로 보이면 아래 표로 메뉴를 찾는다. 리허설 캡처에서 확인한 표기만 적었다. 콘솔 버전에 따라 달라질 수 있다.

이 문서의 표기	한국어 콘솔
Resources, Logical ID, Physical ID	리소스, 논리적 ID, 물리적 ID
Monitoring, Configuration, General configuration	모니터링, 구성, 일반 구성
Environment variables, Triggers, Timeout	환경 변수, 트리거, 제한 시간
Explore items (DynamoDB)	항목 탐색
Visibility timeout (SQS)	기본 표시 제한 시간
maxReceiveCount	최대 수신 수
Dead-letter queue	배달 못한 편지 대기열
Send and receive messages, Poll for messages	메시지 전송 및 수신, 메시지 폴링

---

0. 시작 전 준비

터미널 선택

배포할 터미널은 PowerShell과 CloudShell 중 하나를 고른다.

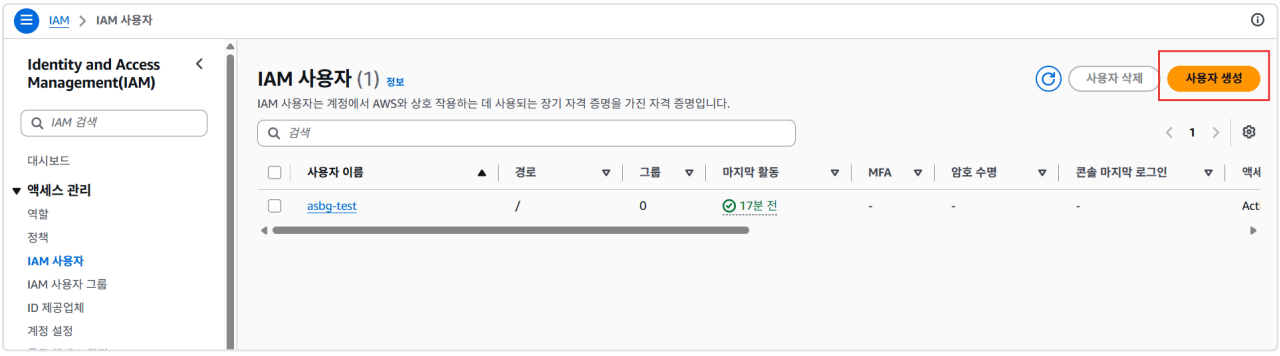
터미널	준비
PowerShell (내 컴퓨터)	AWS CLI v2 로그인, SAM CLI, Python 3.9 이상(`py` 실행기)을 준비한다. `make` **는 필요 없다.** 이 문서는 이 경로를 설명한다
CloudShell	AWS 콘솔의 CloudShell은 Linux 환경이므로 HANDS_ON_GUIDE.md의 CloudShell 경로를 따른다

Git Bash나 WSL처럼 `make`와 `sh`를 쓸 수 있는 환경이면 HANDS_ON_GUIDE.md를 그대로 따라도 된다.

AWS 계정과 CLI 로그인

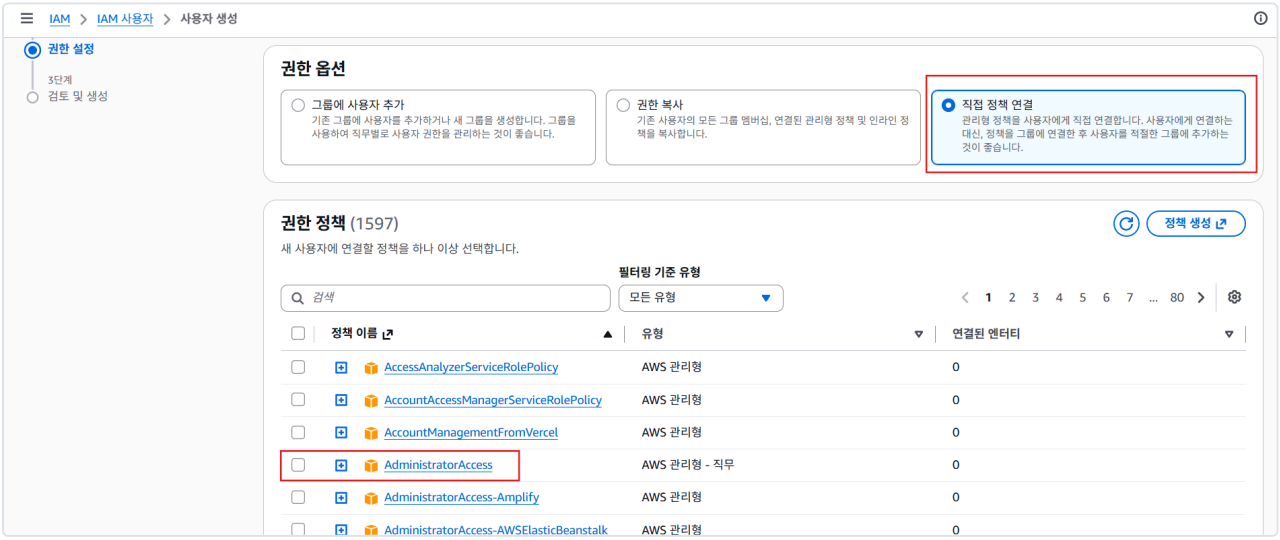
내 컴퓨터의 터미널로 배포하려면 먼저 AWS CLI가 내 계정에 로그인되어 있어야 한다. CloudShell은 콘솔 로그인을 그대로 쓰므로 이 절을 건너뛴다. **세션 시간 안에 하지 않는다.** 세션 전에 `aws sts get-caller-identity`가 성공하는 것까지 확인해 둔다.

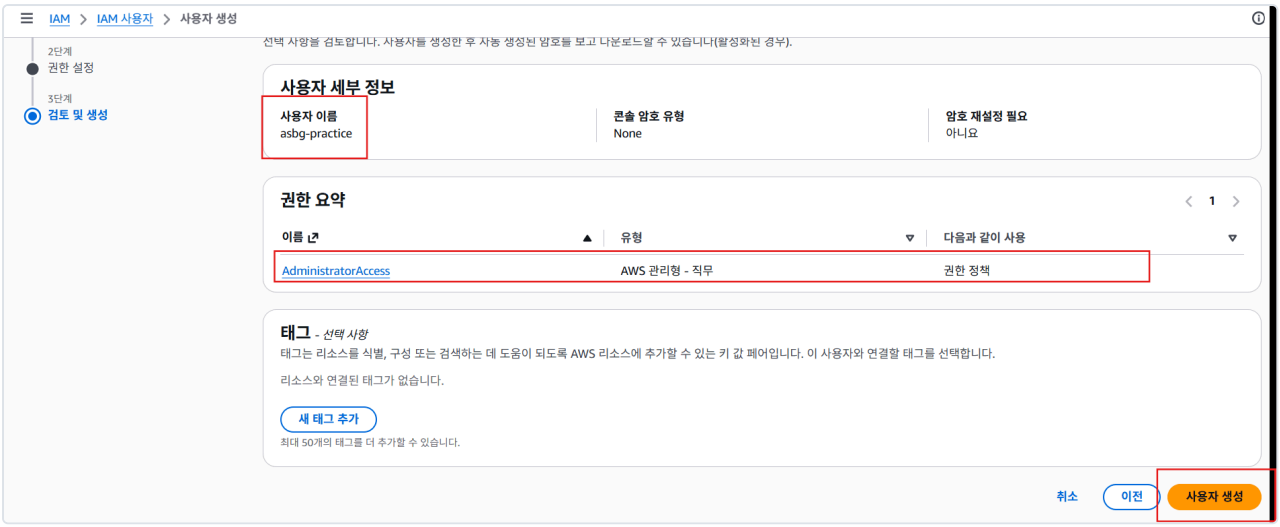
1. **[AWS 콘솔]** 루트 계정 대신 **IAM 사용자**를 만든다. IAM → Users → Create user에서 사용자 이름(예: `asbg-practice`)을 정한다.



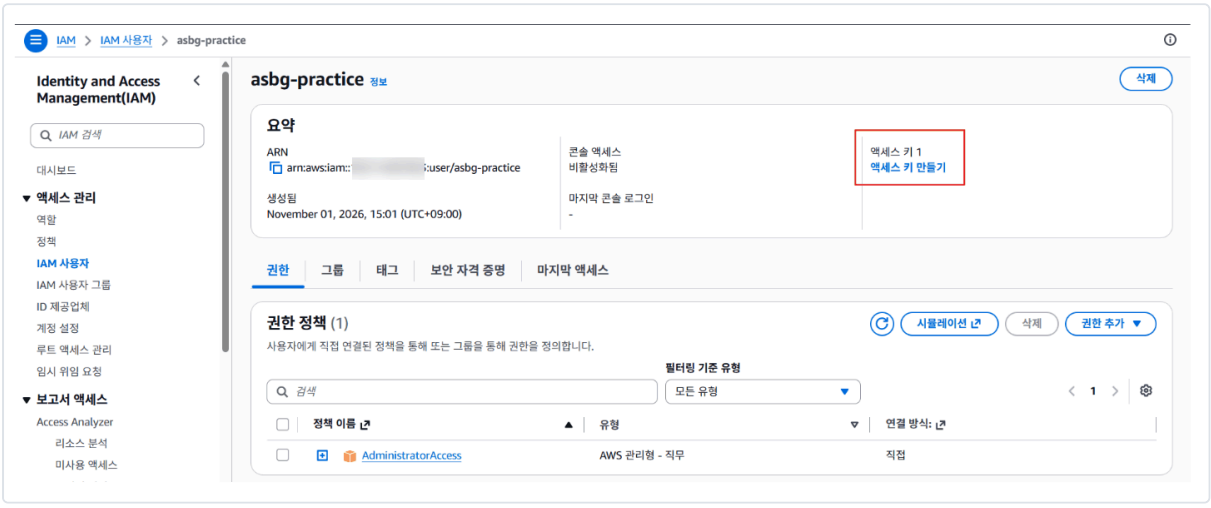


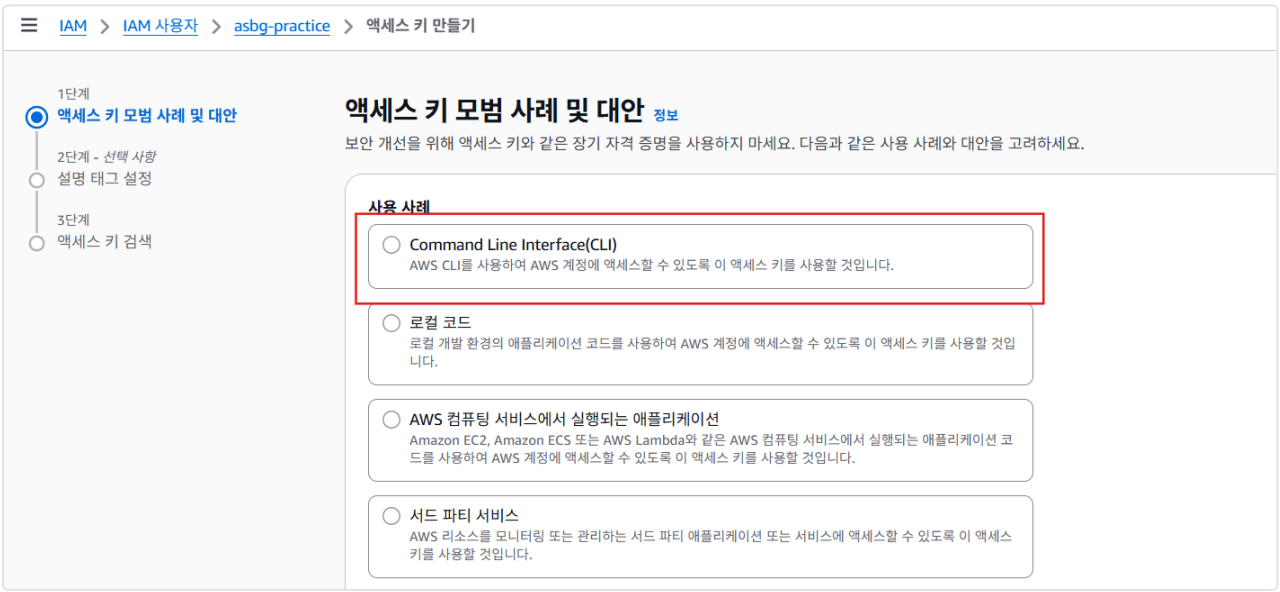
2. **[AWS 콘솔]** Permissions options에서 **Attach policies directly**를 고르고 `AdministratorAccess`를 선택한 뒤 사용자를 만든다. 연습용 계정에서 쓰는 임시 권한이며 4장에서 삭제한다. 이 서비스에 필요한 권한의 종류는 배포 자료의 `README.md`에 있다.



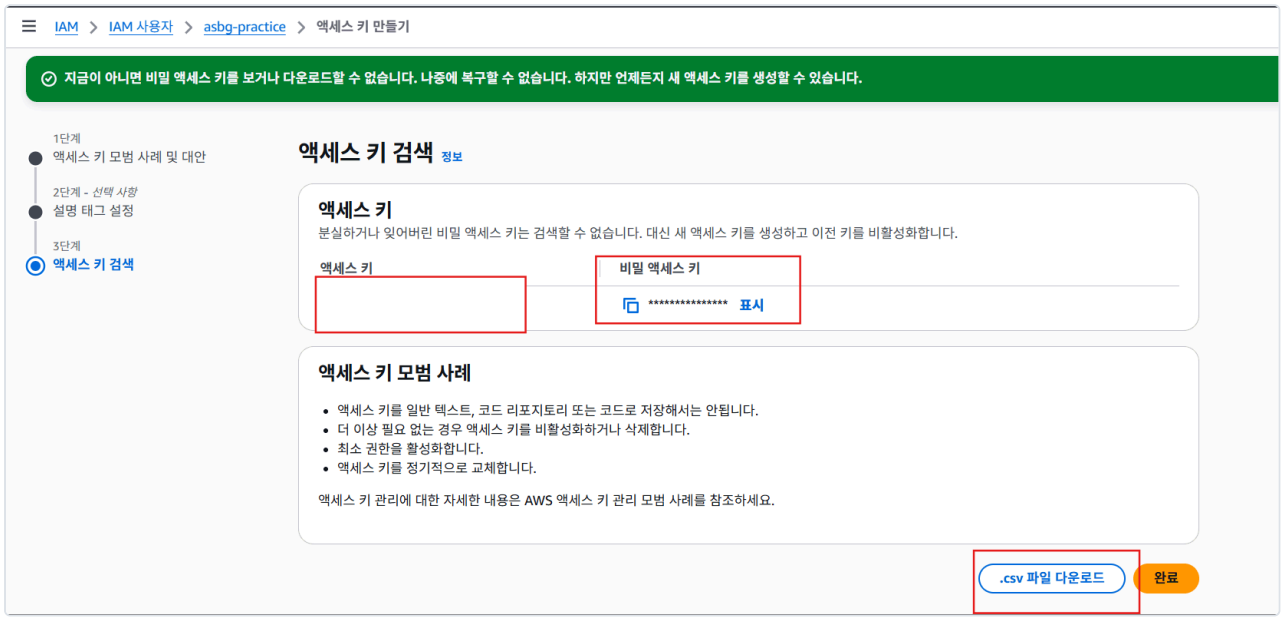


3. **[AWS 콘솔]** 만든 사용자 → Security credentials → Access keys → **Create access key**에서 `Command Line Interface (CLI)`를 고르고 **Download .csv file**로 저장한다. 비밀 액세스 키는 이 창을 닫으면 다시 볼 수 없다.



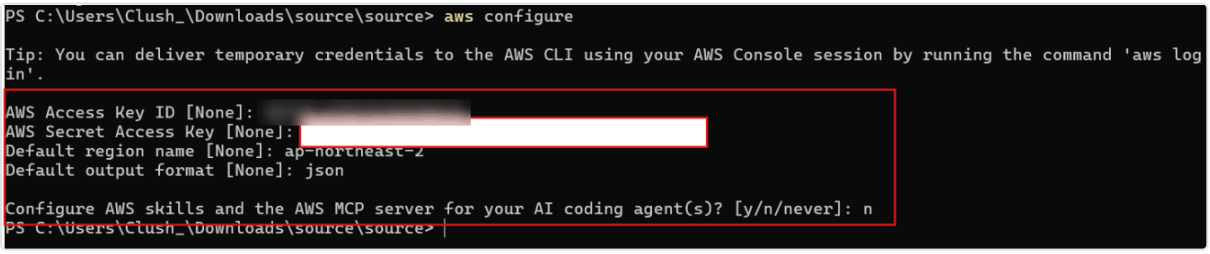






4. **[터미널]** 아래 명령을 실행하고 묻는 값을 입력한다.

```
aws configure
```



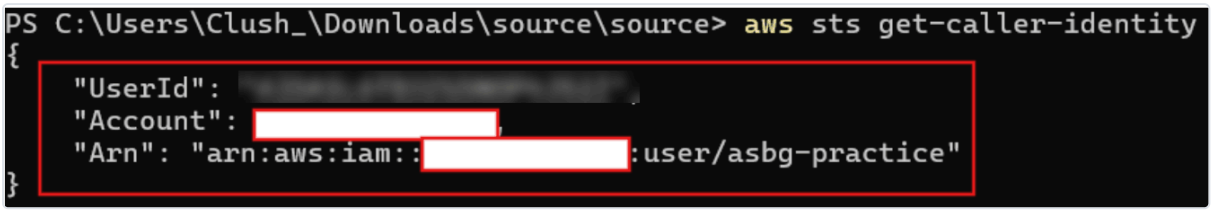
묻는 항목	입력
AWS Access Key ID	받은 `.csv` 의 Access key ID
AWS Secret Access Key	받은 `.csv` 의 Secret access key
Default region name	`ap-northeast-2`
Default output format	`json`

`aws configure` 가 끝나면 `Configure AWS skills and the AWS MCP server` 로 시작하는 `[y/n/never]` 질문이 나올 수 있다. 이 실습에서는 쓰지 않으므로 **`n`** 을 입력한다.

5. **[터미널]** 로그인을 확인한다.

```
aws sts get-caller-identity
```

확인: `Arn`에 방금 만든 사용자(`user/<사용자 이름>`)가 나온다. `Unable to locate credentials`가 나오면 4번을 다시 한다.



액세스 키는 비밀번호와 같다. 채팅, 화면 캡처, 저장소, 과제 README에 넣지 않는다. 노출됐다면 바로 IAM에서 그 키를 비활성화하고 삭제한 뒤 새로 만든다. AWS는 장기 액세스 키 대신 임시 자격 증명을 쓰도록 권장하며, 이 키는 연습이 끝나면 지운다. AWS CLI 사용자 가이드: IAM 사용자 액세스 키로 설정

압축 풀기

```
Expand-Archive source.zip -DestinationPath .
cd source
dir
```

탐색기에서 압축을 풀어도 된다. 이후 모든 명령은 `step1`, `step2`, `tests`, `web` 폴더가 **한 목록에 보이는** `source/` 폴더에서 실행한다. 폴더 안에 `source`가 한 번 더 있으면(`..\source\source`) 안쪽으로 `cd source` 한 번 더 들어간다.

스택 이름과 리전

같은 AWS 계정을 여러 명이 쓰면 각자 다른 스택 이름을 지정한다. 아래 블록을 **PowerShell 창을 열 때마다 통째로 붙여 넣는다.** 설정과 명령은 그 창에서만 유지되므로, 로그를 보려고 창을 하나 더 열면 거기서도 다시 붙여 넣는다. 첫 줄 `event-signup-yourname`은 내 스택 이름으로 바꾼다.



```
$env:AWS_PAGER = ""
$env:STACK = "event-signup-yourname"
$env:REGION = "ap-northeast-2"

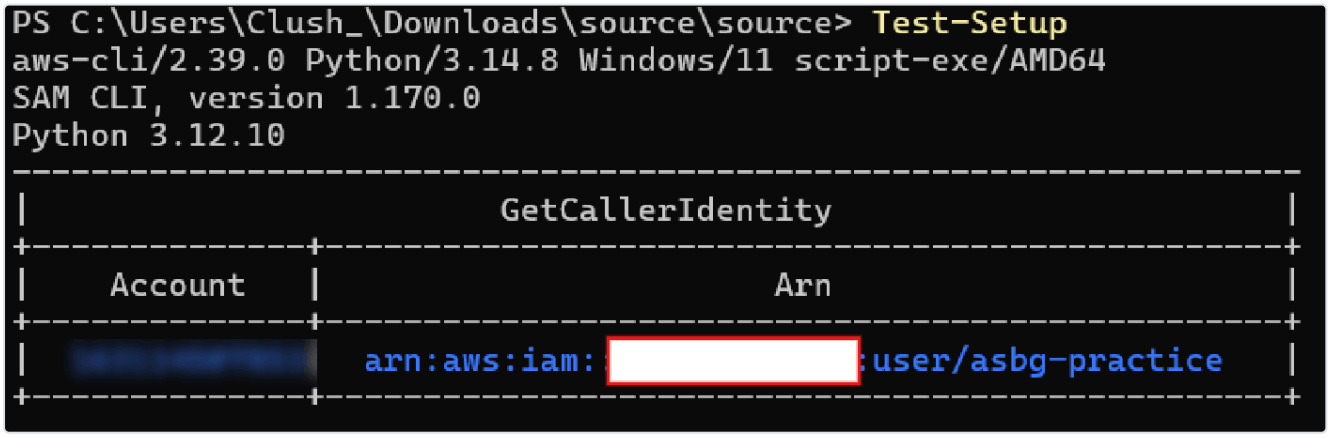
function Get-Out([string]$Key) {
  $v = aws cloudformation describe-stacks--stack-name $env:STACK--region $env:REGION--query
"Stacks[0].Outputs[?OutputKey=='$Key'].OutputValue"--output text
  if (-not $v -or $v -eq "None") { throw "스택$env:STACK 에$Key 출력이 없습니다. 배포 단계와 REGION을
확인하세요." }
  $v
}
function Show-Outputs {
  aws cloudformation describe-stacks--stack-name $env:STACK--region $env:REGION--query
"Stacks[0].Outputs"--output table
}
function Deploy-Stack([string]$Template, [string[]]$Overrides) {
  sam deploy--template-file $Template--stack-name $env:STACK--region $env:REGION--resolve-s3--
s3-prefix $env:STACK--capabilities CAPABILITY_IAM--no-confirm-changeset--no-fail-on-empty-
changeset--parameter-overrides @Overrides
  if ($LASTEXITCODE -eq 0) { Get-Out ApiUrl }
}
function Deploy-Step1([int]$Delay = 1) {
  Deploy-Stack step1/template.yaml @("IssueDelaySeconds=$Delay")
}
function Deploy-Step2([int]$Delay = 1, [string]$Idempotent = "true", [int]$MaxReceives = 3) {
  Deploy-Stack step2/template.yaml @("IssueDelaySeconds=$Delay", "Idempotent=$Idempotent",
"MaxReceiveCount=$MaxReceives")
}
function Show-Logs([string]$Name = "RegisterFunction") {
  aws logs tail (Get-Out "${Name}LogGroup")--region $env:REGION--since 10m--follow--format
short
}
function Show-Db {
  aws dynamodb scan--table-name (Get-Out TableName)--region $env:REGION--query "Items[].
{id:registrationId.S,status:status.S,badge:badgeCode.S,issueCount:issueCount.N,org:organization
.S}"--output table
}
function Show-Dlq {
  aws sqs get-queue-attributes--queue-url (Get-Out DeadLetterQueueUrl)--region $env:REGION--
attribute-names ApproximateNumberOfMessages ApproximateNumberOfMessagesNotVisible--query
Attributes--output table
}
function Show-DlqMessages {
  aws sqs receive-message--queue-url (Get-Out DeadLetterQueueUrl)--region $env:REGION--max-
```

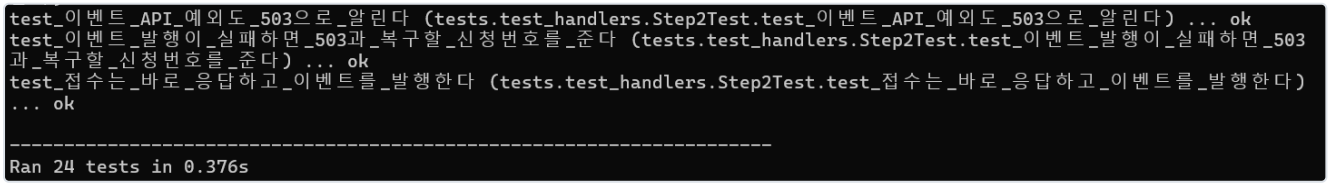
```
 number-of-messages 5--visibility-timeout 0--query "Messages[].Body"--output text
 }
 function Start-Redrive {
   aws sqs start-message-move-task--source-arn (Get-Out DeadLetterQueueArn)--destination-arn
 (Get-Out QueueArn)--max-number-of-messages-per-second 1--region $env:REGION
 }
 function Show-Redrive {
   aws sqs list-message-move-tasks--source-arn (Get-Out DeadLetterQueueArn)--region $env:REGION
--output table
 }
 function Remove-Stack {
   sam delete--stack-name $env:STACK--region $env:REGION--no-prompts
 }
 function Test-Setup {
   aws--version
   sam--version
   py--version
   aws sts get-caller-identity--region $env:REGION--query "{Account:Account,Arn:Arn}"--output
 table
 }
```

가상 이름·이메일을 사용한다. 실습 API에는 인증이 없다. Python 실행기가 `py` 가 아니라 `python` 이면 블록의 `py` 와 이후 명령의 `py` 를 `python` 으로 바꾼다.

점검

```
Test-Setup
py -m unittest -b tests.test_handlers -v
```





- 도구 설치 방법은 배포 자료의 `README.md`에 있다. 명령을 찾을 수 없다는 오류가 나면 그 도구를 설치하고 PowerShell을 새로 연다.
- Lambda는 Python 3.12를 쓰지만 이 예제는 `sam build` 없이 배포하므로 배포 컴퓨터에 Python 3.12나 Docker가 필요하지 않다.
- `Test-Setup`의 마지막 표가 **배포될 계정**을 보여 준다. 캡처할 때는 계정 ID와 ARN을 가린다.
- `test_handlers`는 AWS를 호출하지 않는 기본 테스트이다. 오류 실험 테스트인 `test_exercise`는 **수정 전 실패하는 것이 정상**이다.

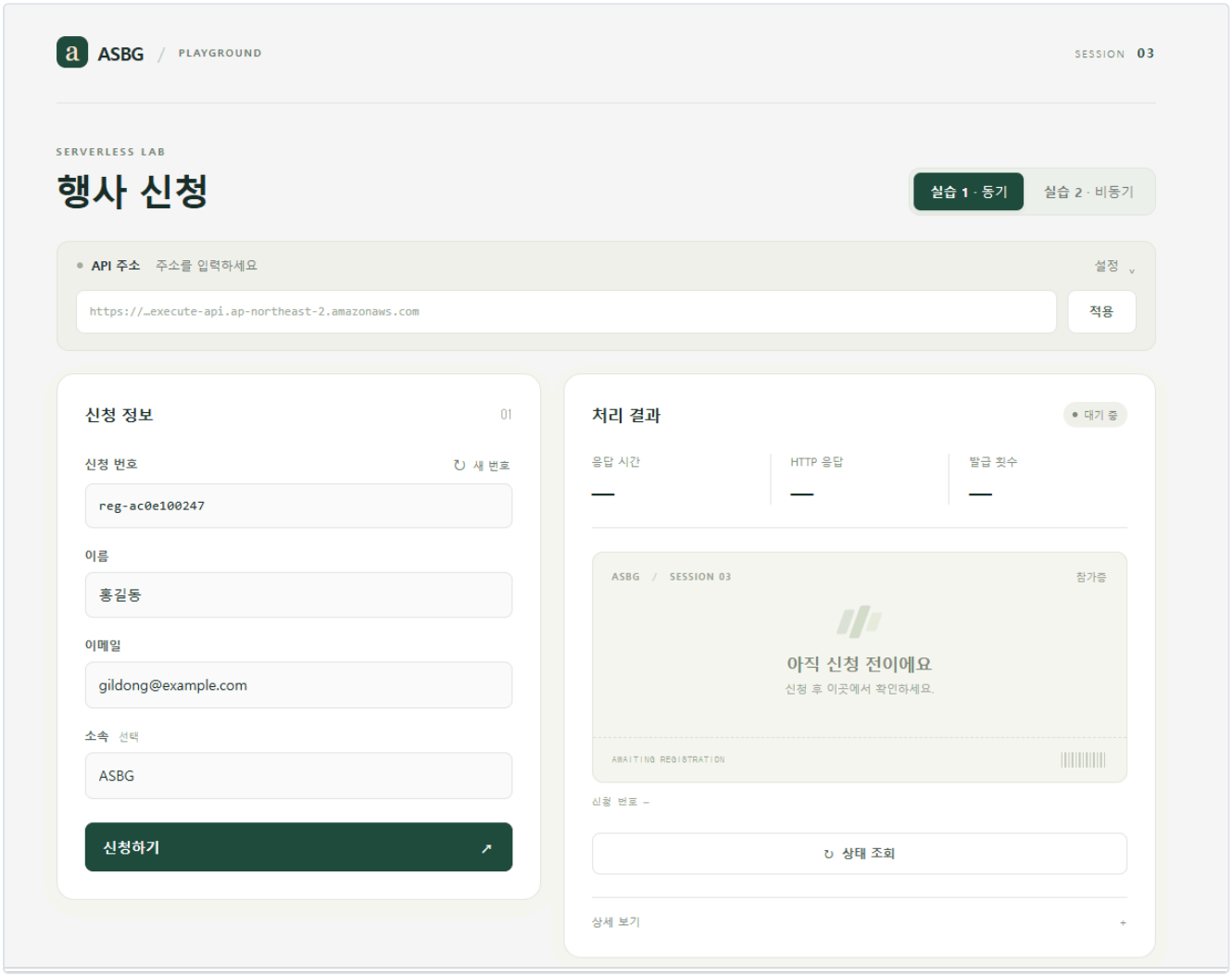
신청 화면 열기

신청 화면은 **내 컴퓨터에 압축을 풀고 로컬 서버로 열어** 사용한다. `web/index.html` 파일을 바로 열면 브라우저가 출처를 `null`로 보내고, 이 실습 API 호출이 CORS 오류(`No 'Access-Control-Allow-Origin' header`)로 막히는 것이 확인되었다. CloudShell에만 업로드하면 내 브라우저에 이 파일이 생기지 않으므로 이 경우에도 내 컴퓨터에서 연다.

새 터미널 창을 하나 더 열고 `source/` 폴더에서 아래를 실행한다. 이 창은 실습이 끝날 때까지 열어 두고, 배포 명령은 원래 창에서 실행한다. 서버를 멈추려면 Ctrl+C를 누른다.

```
py -m http.server 8000--directory web
```

브라우저에서 `http://localhost:8000/`을 연다. Python이 없거나 화면을 사용할 수 없으면 마지막 부록의 curl 예제를 쓴다.



화면 구성

신청 화면의 버튼과 입력칸이 하는 일이다.

화면	하는 일
`실습 1 · 동기` / `실습 2 · 비동기`	해당 단계의 도구를 표시한다. AWS 배포는 별도로 실행해야 한다
`API 주소` → `적용`	`Get-Out ApiUrl` 로 얻은 주소를 사용한다. 주소 적용만으로 배포 성공을 확인한 것은 아니다
`새 번호`	독립된 실험을 시작한다. 이전 번호는 메모해 둔다
`신청하기`	현재 입력으로 신청 요청을 보낸다
`상태 조회`	신청 번호의 현재 DB 상태를 확인한다. 이전 번호를 직접 입력해 조회할 수도 있다
`작업 재전송`	실습 2에서 저장된 신청의 발급 이벤트를 다시 보낸다
`오류 실험` / `정상 신청`	실습 2의 입력 예시를 채운다. `신청하기` 를 눌러야 전송된다
`상세 보기`	요청·응답의 자세한 내용을 펼친다

`응답 시간` 과 `HTTP 응답` 은 마지막 신청 또는 작업 재전송 POST의 결과이다. 상태 조회는 이 기록을 덮어쓰지 않는다. 자동 조회는 신청 후 최대 45초, 재전송 후 최대 15초 동안 진행되며, 이후에는 `상태 조회` 를 눌러 확인한다. 조회가 멈췄다는 것만으로 실패나 DLQ 이동을 판단하지 않는다.

실습 기록표

실습 중 나오는 값이 많아 한곳에 모아 적는다. 과제 기록에 그대로 쓴다.

언제	적을 값	내 값
준비	스택 이름, 리전	
1-2	정상 신청 번호, 응답 코드와 시간, 참가증 코드	
1-3	지연 6초에서의 응답 시간	
2 사전 배포	**실패 신청 번호**	
2-2	정상 신청 번호, 응답 코드와 시간, 최종 상태	
2-3	재전송 전후의 참가증 코드와 발급 횟수	
2-4	실패 신청의 수신 횟수, DLQ에서 확인한 시각	

---

1. 실습 1 · 요청 처리와 저장 확인 (12분)

1-1. 배포와 연결 (3분)

지원 발표 시작 때 아래 배포를 실행한다. 단계 앞의 **[터미널]**, **[AWS 콘솔]**, **[브라우저]**는 그 단계를 하는 곳이다.

1. **[터미널]** 스택 이름과 리전을 확인한다. 비어 있으면 0장의 설정 블록을 이 창에 붙여 넣지 않은 것이다.

```
echo "STACK=$env:STACK REGION=$env:REGION"
```

```
PS C:\Users\Clush_\Downloads\source\source> echo "$env:STACK$env:REGION"
event-signup-rehearsal-soobeenap-northeast-2
```

2. **[터미널]** 배포한다.

```
Deploy-Step1
```

`sam deploy`가 스택을 만들거나 갱신하고, **배포가 성공했을 때만** 마지막 줄에 API 주소를 출력한다. 주소는 `https://<문자열>.execute-api.ap-northeast-2.amazonaws.com` 모양이며 끝에 `/`가 없다.

```
CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------------------------
Outputs
-----------------------------------------------------------------------------------------------------------
Key                 TableName
Description         -
Value               event-signup-rehearsal-soobeen-RegistrationsTable-LUKCGY3IACI5

Key                 ApiUrl
Description         ?? ??(web/index.html)? ??? API ??
Value               https://62vk9lbmx3.execute-api.ap-northeast-2.amazonaws.com

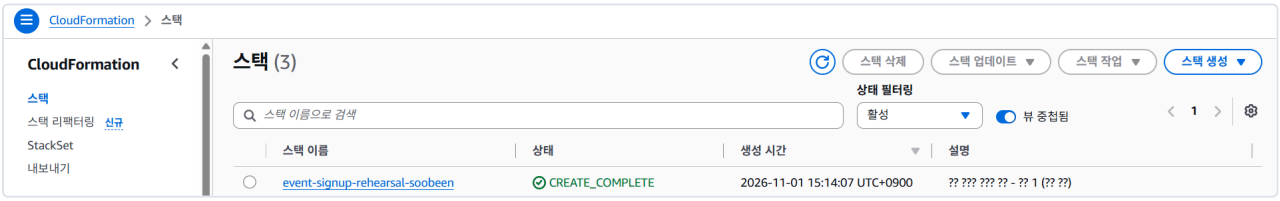
Key                 RegisterFunctionLogGroup
Description         -
Value               /asbg/event-signup-rehearsal-soobeen/register

Key                 StatusFunctionLogGroup
Description         -
Value               /asbg/event-signup-rehearsal-soobeen/status
-----------------------------------------------------------------------------------------------------------

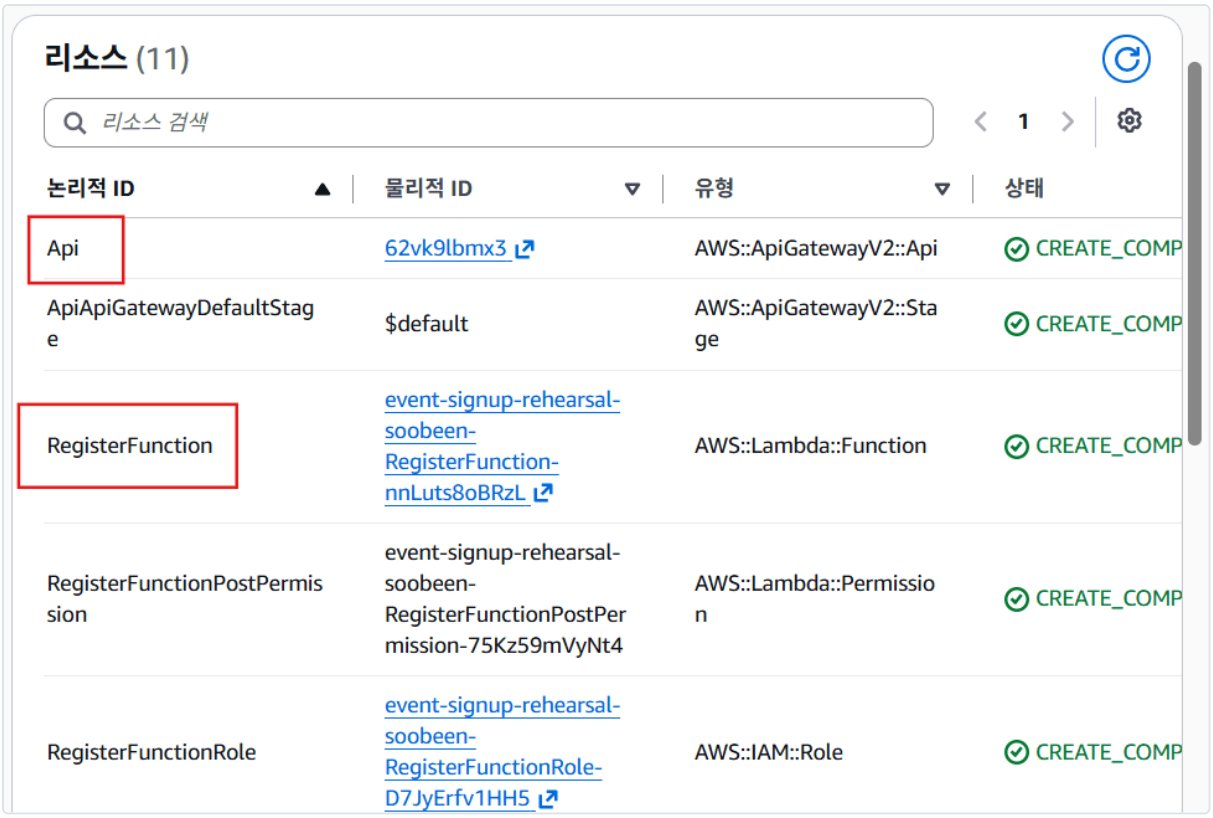
Successfully created/updated stack - event-signup-rehearsal-soobeen in ap-northeast-2

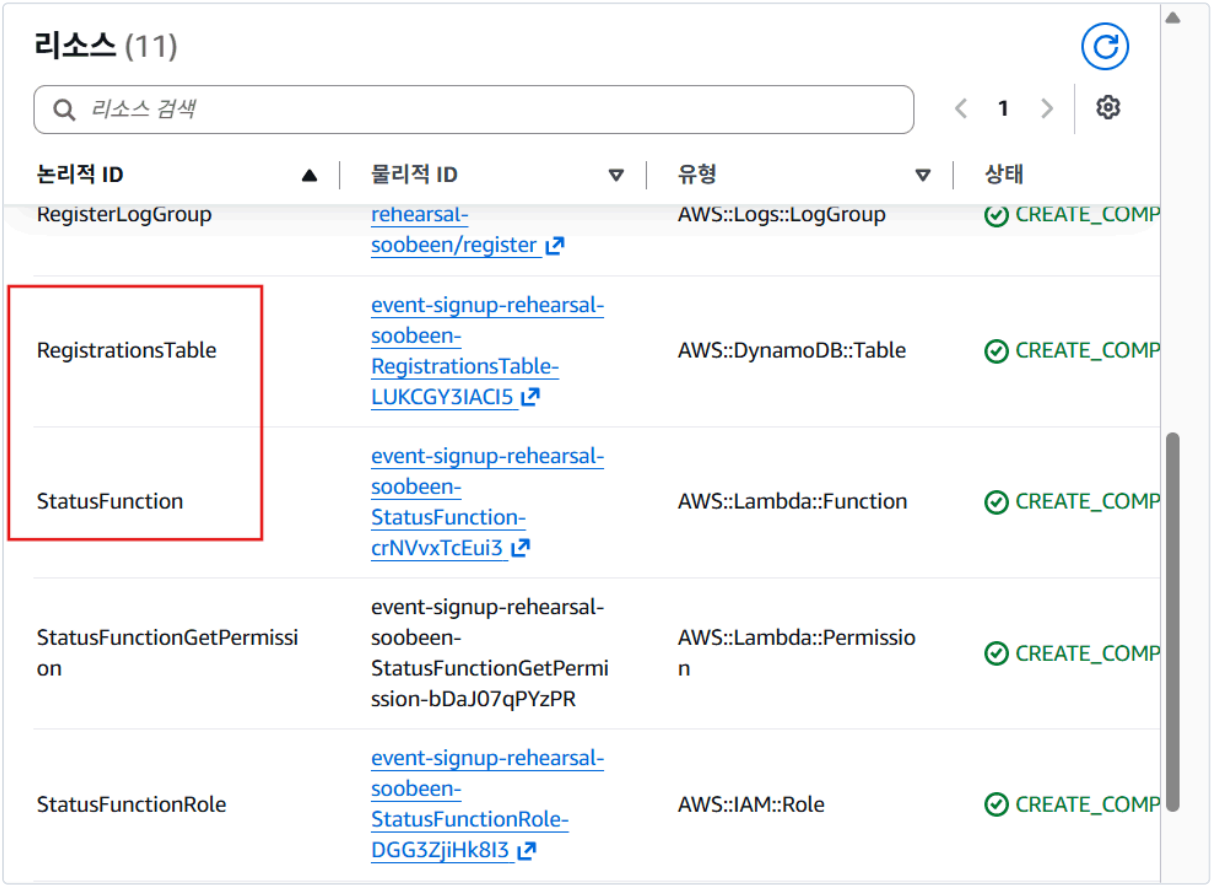
https://62vk9lbmx3.execute-api.ap-northeast-2.amazonaws.com
```

3. **[터미널]** 주소가 보이지 않으면 `Get-Out ApiUrl`을 다시 실행한다. 함수·테이블 이름은 `Show-Outputs`로 본다.
4. **[AWS 콘솔]** CloudFormation → Stacks → 내 스택에서 상태가 `CREATE_COMPLETE`인지 확인한다(이미 있던 스택을 갱신했다면 `UPDATE_COMPLETE`). 실패했다면 **Events** 탭에서 가장 먼저 실패한 이유를 읽는다.



5. **[AWS 콘솔] Resources** 탭에서 `RegisterFunction`, `StatusFunction`, `RegistrationsTable`, `Api`를 찾는다. Physical ID 링크로 각 서비스 화면을 열 수 있다.





6. **[브라우저]** 0장의 로컬 서버를 켜 두고 `http://localhost:8000/` 을 연다. 상단에서 `실습 1 · 동기` 가 선택되어 있는지 보고, `API 주소` 칸에 1–3단계의 주소를 붙여 넣어 `적용` 한다.



체크: 배포 완료 상태와 API 주소를 확인했다. 3분 안에 배포가 준비되지 않으면 발표자 화면을 함께 관찰하고, 설치 문제는 실습 뒤에 해결한다.

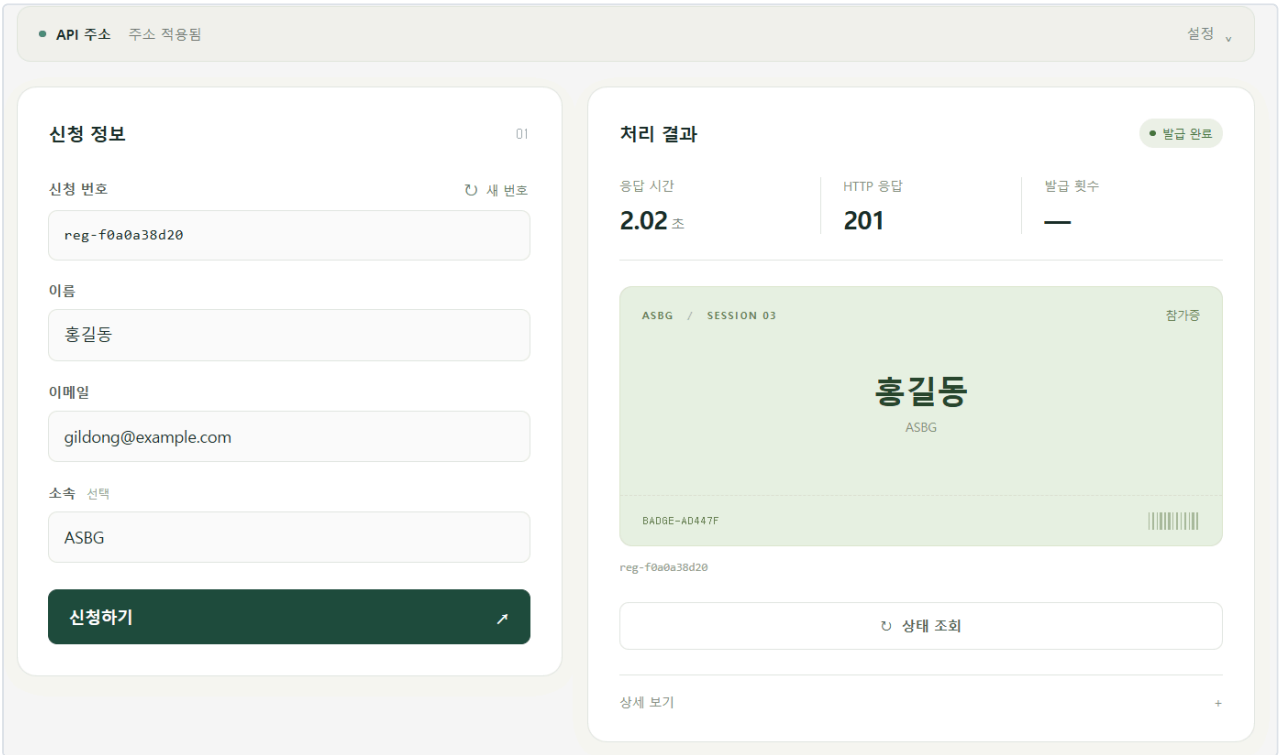
1-2. 응답 · 로그 · DB 확인 (4분)

1. **[터미널, 선택]** 로그를 실시간으로 보려면 **다른 PowerShell 창**을 열어 0장의 설정 블록을 붙여 넣은 뒤(스택 이름은 처음과 같게 쓴다) 먼저 실행해 둔다. `Ctrl+C` 로 끝낼 때까지 그 창을 쓴다.

```
Show-Logs RegisterFunction
```

```
>>   aws sqs list-message-move-tasks --source-arn (Get-Out DeadLetterQueueArn) --region $env:REGION --output table
>> }
PS C:\Users\Clush_> function Remove-Stack {
>>   sam delete --stack-name $env:STACK --region $env:REGION --no-prompts
>> }
PS C:\Users\Clush_> function Test-Setup {
>>   aws --version
>>   sam --version
>>   py --version
>>   aws sts get-caller-identity --region $env:REGION --query "{Account:Account,Arn:Arn}" --output table
>> }
PS C:\Users\Clush_> Show-Logs RegisterFunction
```

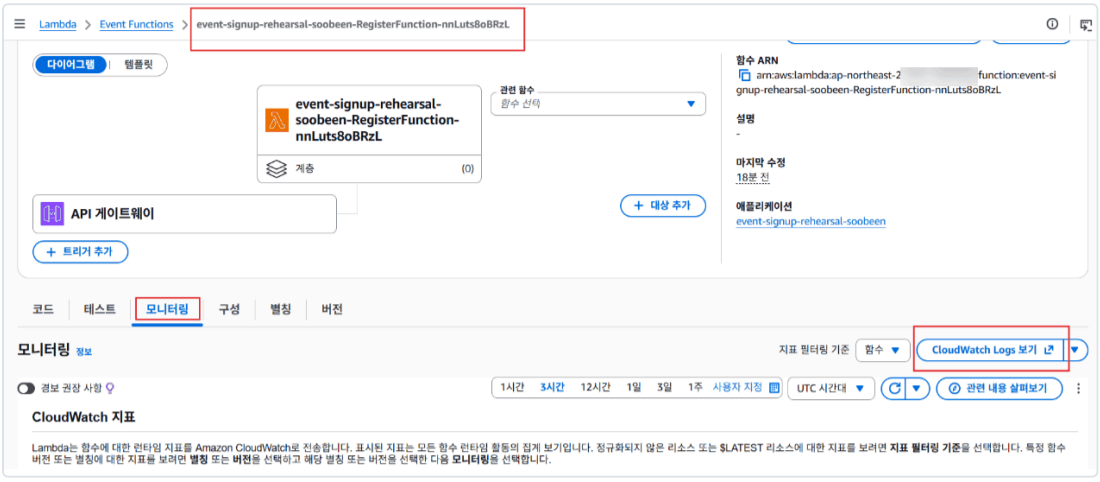
2. **[브라우저]** 입력값(이름 `홍길동` , 소속 `ASBG`)을 그대로 두고 `신청하기` 를 누른다. 화면의 **신청 번호**(`reg-…`)를 기록표에 적는다.

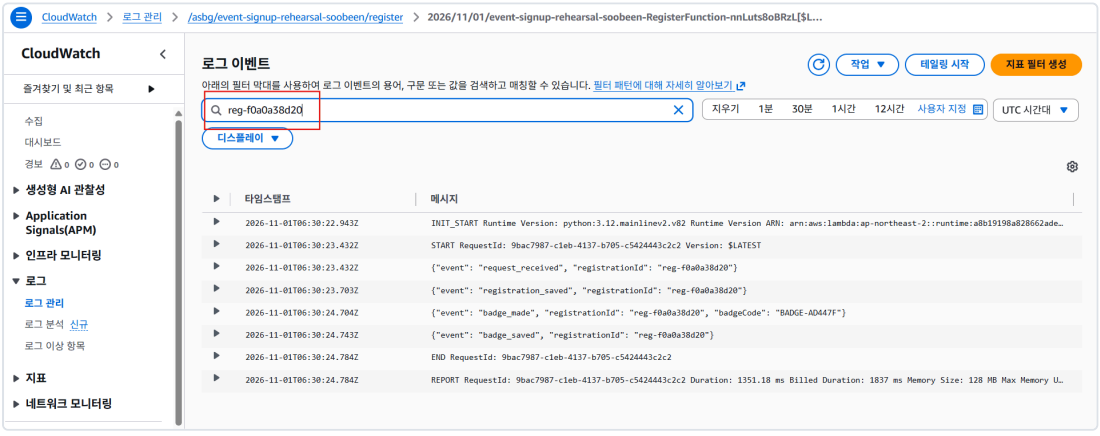


```
PS C:\Users\Clush_> Show-Logs RegisterFunction
2026-11-01T06:30:22 INIT_START Runtime Version: python:3.12.mainlinev2.v82      Runtime Version ARN: arn:aws:lambda:ap-n
ortheast-2::runtime:a8b19198a828662ade07d066a1dce047429a5869166346747ab7d57435583a29
2026-11-01T06:30:23 START RequestId: 9bac7987-c1eb-4137-b705-c5424443c2c2 Version: $LATEST
2026-11-01T06:30:23 {"event": "request_received", "registrationId": "reg-f0a0a38d20"}
2026-11-01T06:30:23 {"event": "registration_saved", "registrationId": "reg-f0a0a38d20"}
2026-11-01T06:30:24 {"event": "badge_made", "registrationId": "reg-f0a0a38d20", "badgeCode": "BADGE-AD447F"}
2026-11-01T06:30:24 {"event": "badge_saved", "registrationId": "reg-f0a0a38d20"}
2026-11-01T06:30:24 END RequestId: 9bac7987-c1eb-4137-b705-c5424443c2c2
2026-11-01T06:30:24 REPORT RequestId: 9bac7987-c1eb-4137-b705-c5424443c2c2      Duration: 1351.18 ms    Billed Duration:
 1837 ms        Memory Size: 128 MB     Max Memory Used: 90 MB  Init Duration: 485.31 ms
```

3. **[브라우저]** 결과 영역**(상세 보기)**에서 `HTTP 응답` 이 `201` 인지, `응답 시간` 과 참가증 코드(`BADGE-…`)를 확인한다. 이 방식에서는 참가증 발급이 끝난 뒤에야 응답이 온다.
4. **[AWS 콘솔]** 로그를 찾는다. 둘 중 편한 방법을 쓴다.
   - Lambda → 내 스택의 `RegisterFunction` → Monitor → View CloudWatch logs → 가장 최근 로그 스트림에서 신청 번호를 검색한다.







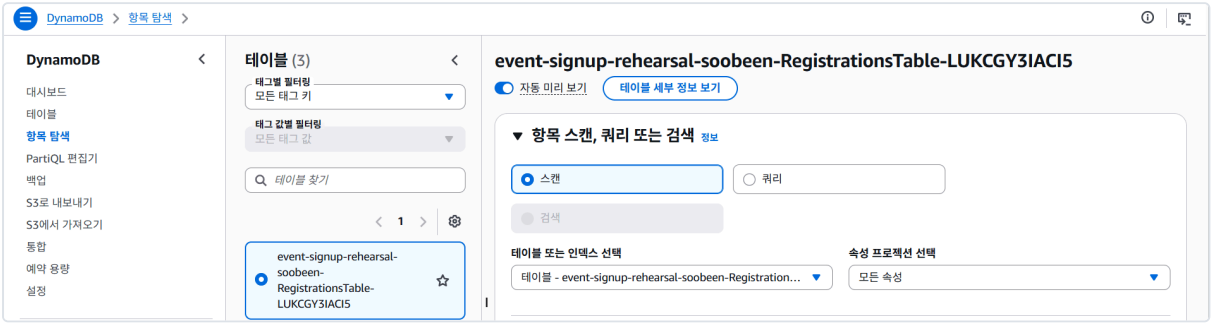
위 터미널 로그값과 동일

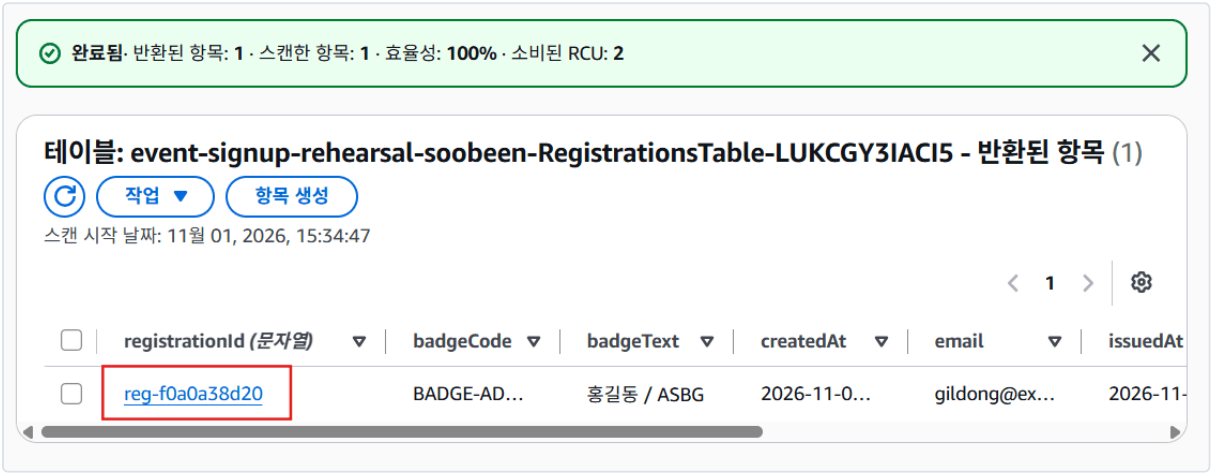
   - CloudWatch → Logs Insights에서 로그 그룹 `/asbg/<스택 이름>/register` 를 선택하고 최근 30분으로 아래 쿼리를 실행한다.

```
fields @timestamp, @message
| filter @message like "여기에 신청 번호"
| sort @timestamp asc
```

`request_received → registration_saved → badge_made → badge_saved`를 찾는다.

5. **[AWS 콘솔]** DynamoDB → Explore items에서 내 테이블을 선택하고 신청 번호를 찾는다. 테이블 이름은 스택 **Outputs → TableName**에도 있다. `status = ISSUED`와 `badgeCode`를 확인한다. **[터미널]**에서는 `Show-Db`가 같은 내용을 표로 보여 준다(작은 실습 테이블 전체를 읽는 명령이다).

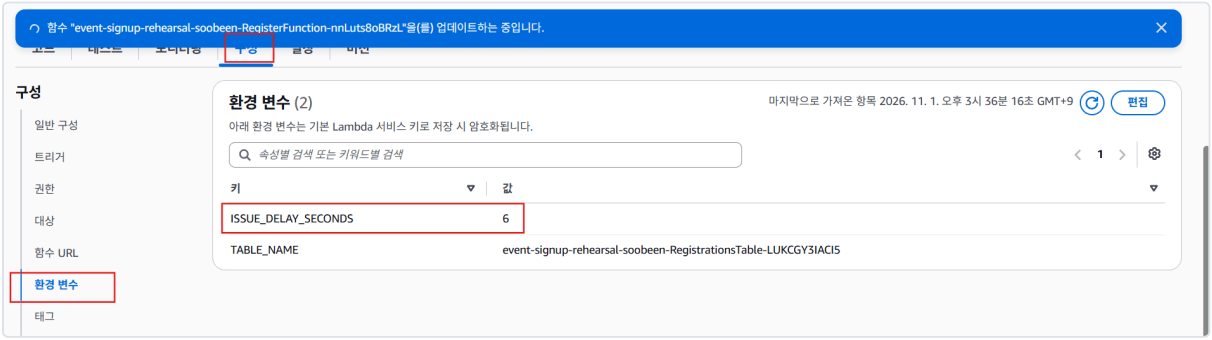




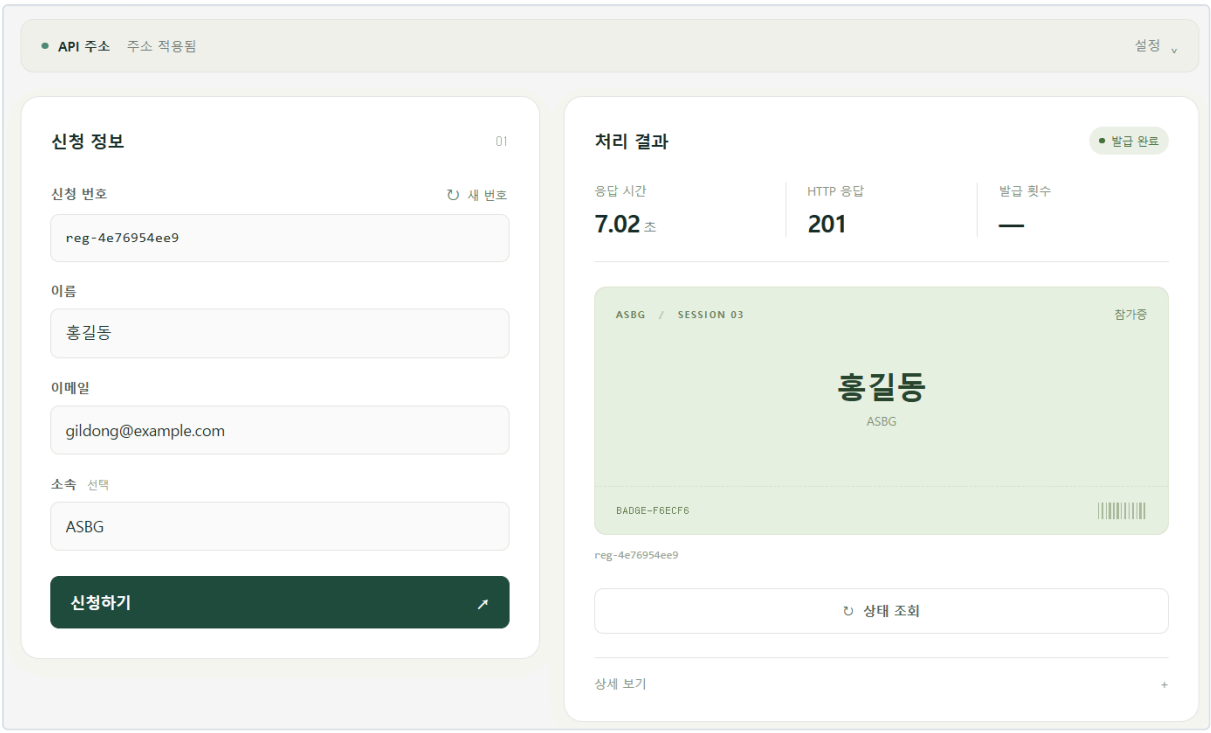
체크: 화면·로그·DB의 신청 번호와 참가증 코드가 같다. 다른 참가자의 스택이나 지난 신청을 보고 있지 않은지 확인한다.

1-3. 발급 지연 실험 (5분)

1. **[AWS 콘솔]** Lambda → 내 스택의 `RegisterFunction` → Configuration 탭 → Environment variables → Edit에서 `ISSUE_DELAY_SECONDS`를 **6**으로 바꾸고 Save한다.



2. **[브라우저]** `새 번호` → `신청하기`를 누른다. `응답 시간`을 1-2와 비교해 기록표에 적는다. 대략 6초의 발급 지연에 호출·저장 시간이 더해진다.



3. 사용자에게 접수만 먼저 알려 줄 수 있는지 생각해 본다.
4. **[AWS 콘솔]** 비교 화면을 기록한 뒤 `ISSUE_DELAY_SECONDS` 를 원래 값 **1**로 되돌리고 Save한다.

콘솔 수정은 실습 중 임시 변경이다. 동일 템플릿으로 재배포해도 항상 덮어써지는 것은 아니므로 직접 복원한다. 과제의 최종 설정은 템플릿·코드에 남긴다.

체크: 발급 지연이 사용자 응답 시간에 포함된다는 점을 설명할 수 있다.

선택: 지연을 15로 올리면 Timeout 10초를 넘는다. 응답 오류, 타임아웃 로그, DB의 `RECEIVED` 를 관찰하고 다시 1로 복원한다. 정확한 HTTP 오류 코드는 실제 결과를 기록한다.

---

2. 실습 2 · 비동기 처리와 실패 실험 (23분)

사전 배포

수빈 발표 시작 때 같은 폴더·스택·리전으로 배포한다. **발급 지연을 6초로 맞춰** 앞 실험과 비교한다.

```
Deploy-Step2 -Delay 6
```

`Deploy-Step2` 도 배포가 성공하면 마지막 줄에 API 주소를 출력한다.

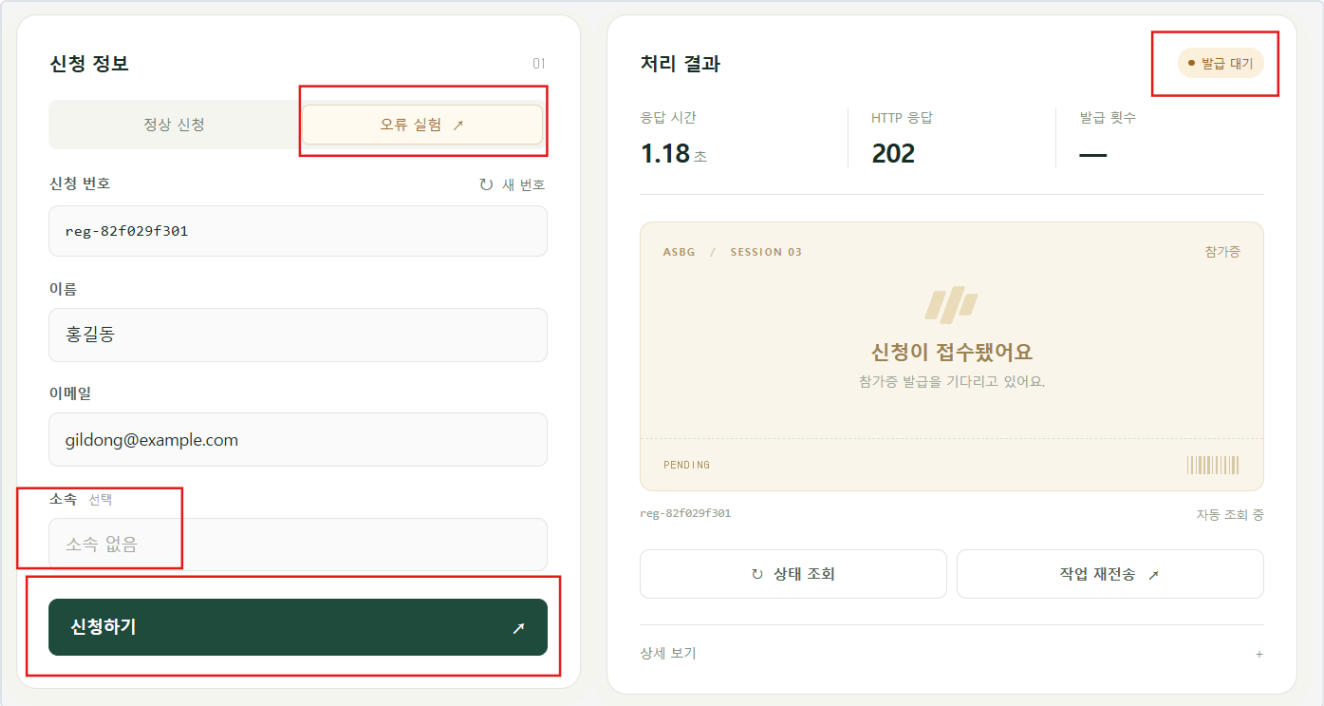
```
Key                 QueueArn
Description         -
Value               arn:aws:sqs:ap-northeast-2              :vent-signup-rehearsal-soobeen-IssueQueue-dVjqk08GYcqK
-------------------------------------------------------------------------------------------------------------------


Successfully created/updated stack - event-signup-rehearsal-soobeen in ap-northeast-2

https://62vk9lbmx3.execute-api.ap-northeast-2.amazonaws.com
PS C:\Users\Clush_\Downloads\source\source>
```

같은 스택의 기존 테이블과 API를 이어 쓰도록 구성되어 있다. 배포 완료 후 Outputs를 다시 확인하고 UI에서 `실습 2 · 비동기` 를 선택한다. 기존 실습 1 데이터와 구별하도록 새 신청 번호를 사용한다.

배포가 준비되면 `오류 실험` → `신청하기` 로 한 건을 먼저 보낸다. 소속이 비어 있는 입력이며 응답은 `202` 이다. **이 실패 신청 번호를 따로 기록한다.** 몇 분 뒤 이 신청을 복구한다. 아직 `issue.py` 를 수정하지 않는다.



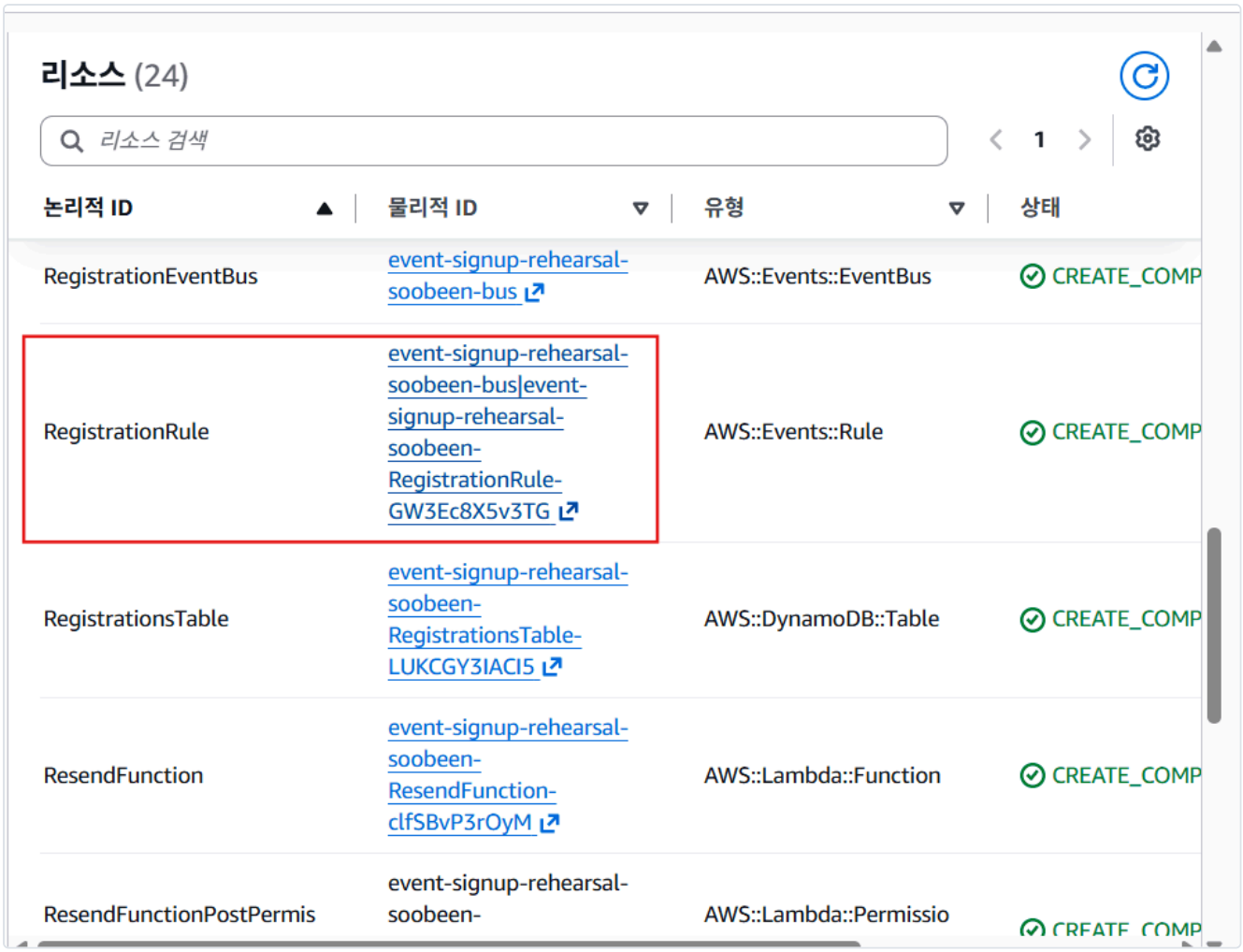
2-1. 연결 확인 (3분)

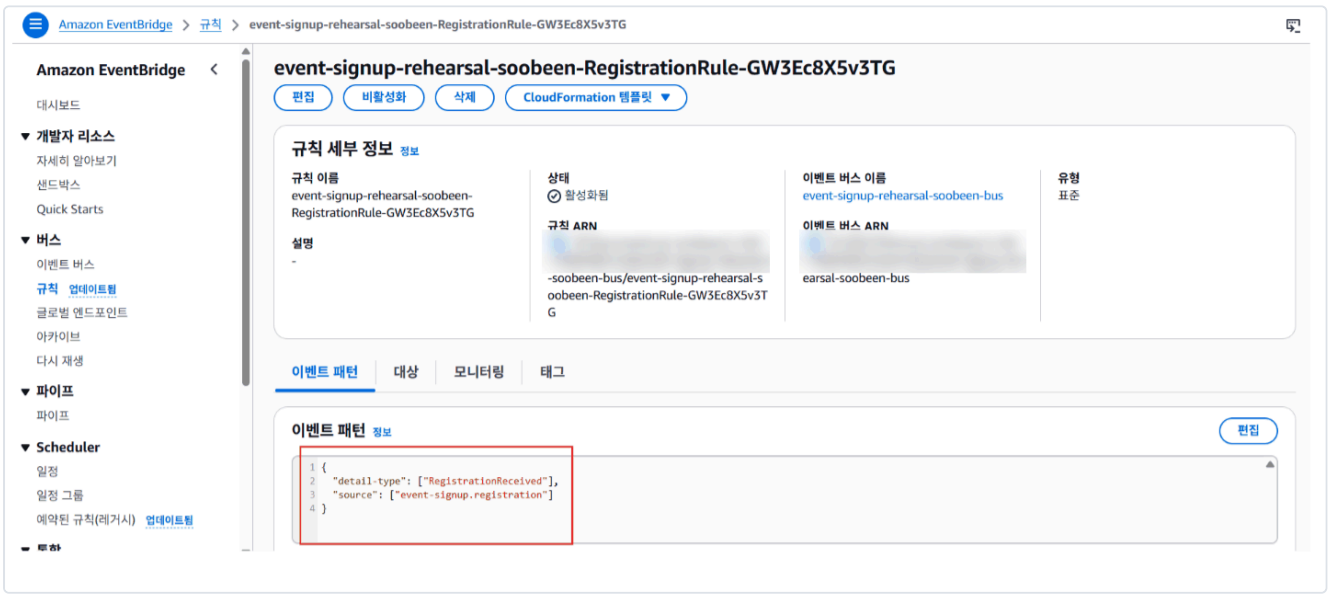
CloudFormation → 내 스택 → Resources 탭에서 아래 표의 Logical ID로 행을 찾고, 같은 행의 Physical ID가 링크이면 눌러 그 리소스로 이동한다. 링크가 아니면 아래 Outputs의 이름을 각 서비스 콘솔에서 검색한다. 흐름은 다음과 같다. 위에서 실패 신청을 못 보냈다면 지금 보낸다.

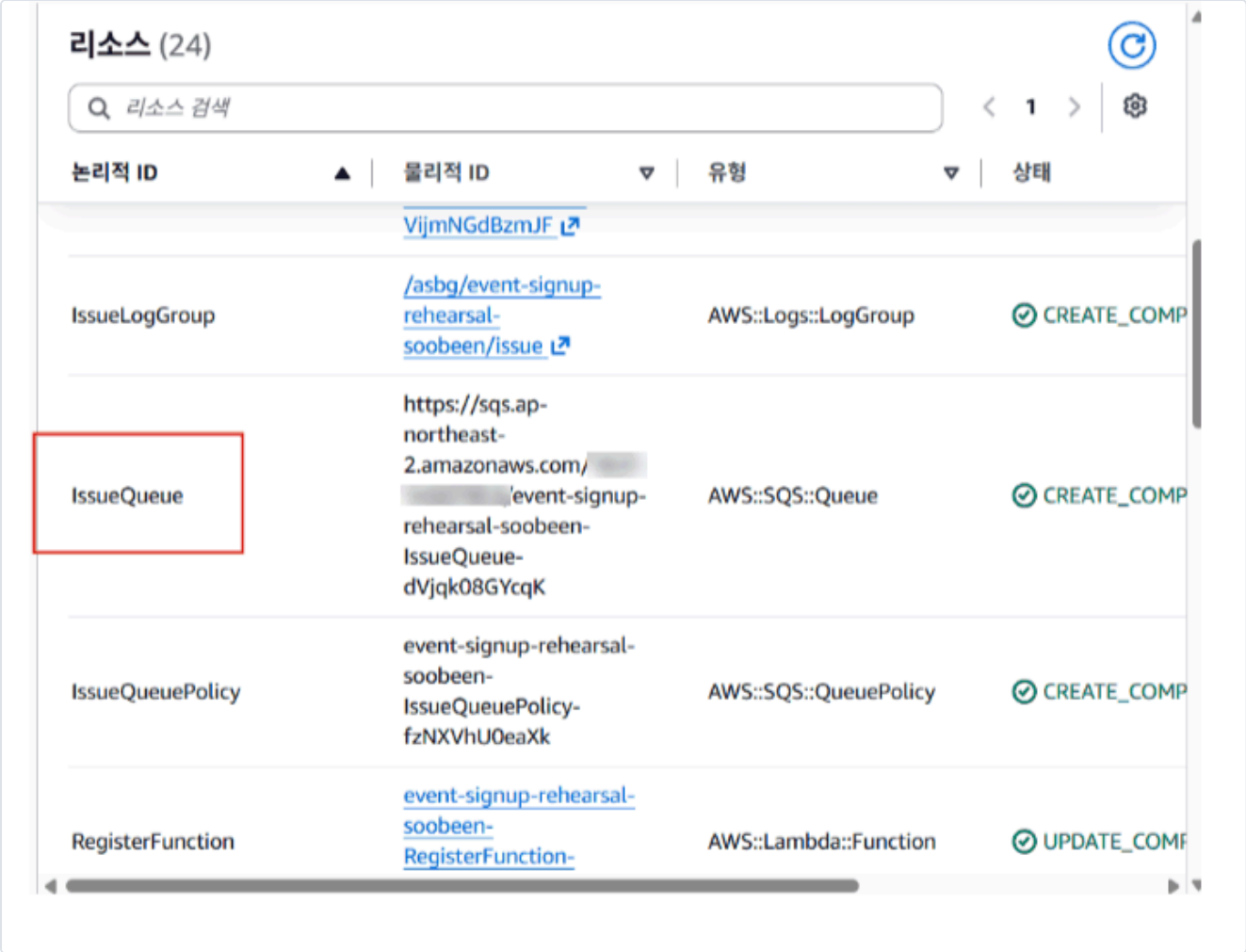
```
신청 함수 → EventBridge 규칙 → SQS 발급 큐 → 발급 함수 → DynamoDB
                                                └ 반복 실패 → DLQ
```

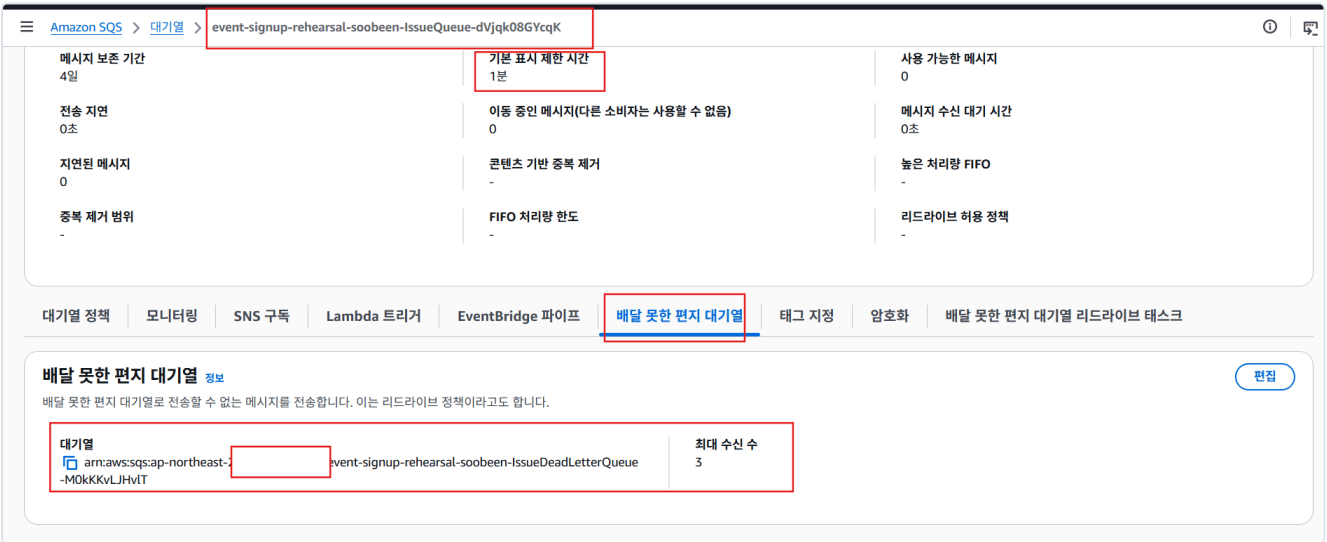
콘솔에서 확인할 위치와 값이다.

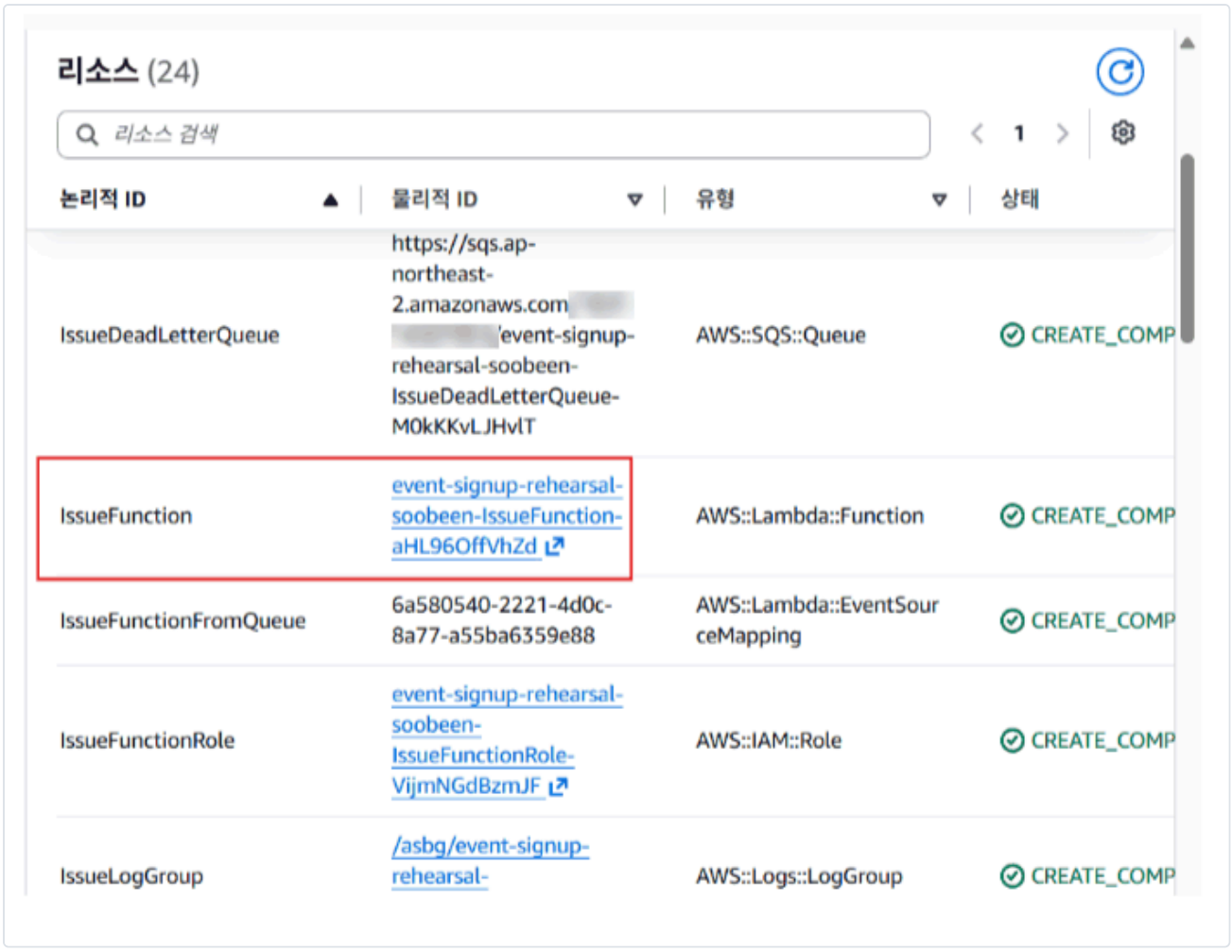
위치	Resources의 Logical ID	확인할 값
EventBridge의 내 이벤트 버스 → 규칙	`RegistrationRule`	`source = event-signup.registration` , `detail-type = RegistrationReceived` , 대상은 발급 큐
SQS 발급 큐 → 설정 / Dead-letter queue	`IssueQueue`	Visibility timeout **60초**, maxReceiveCount **3**, 연결된 DLQ
Lambda `IssueFunction` → Configuration → Triggers	`IssueFunction`	SQS 트리거, 한 번에 메시지 1개
Lambda `IssueFunction` → Configuration	`IssueFunction`	Timeout **10초**, `ISSUE_DELAY_SECONDS = 6` , `IDEMPOTENT = true`

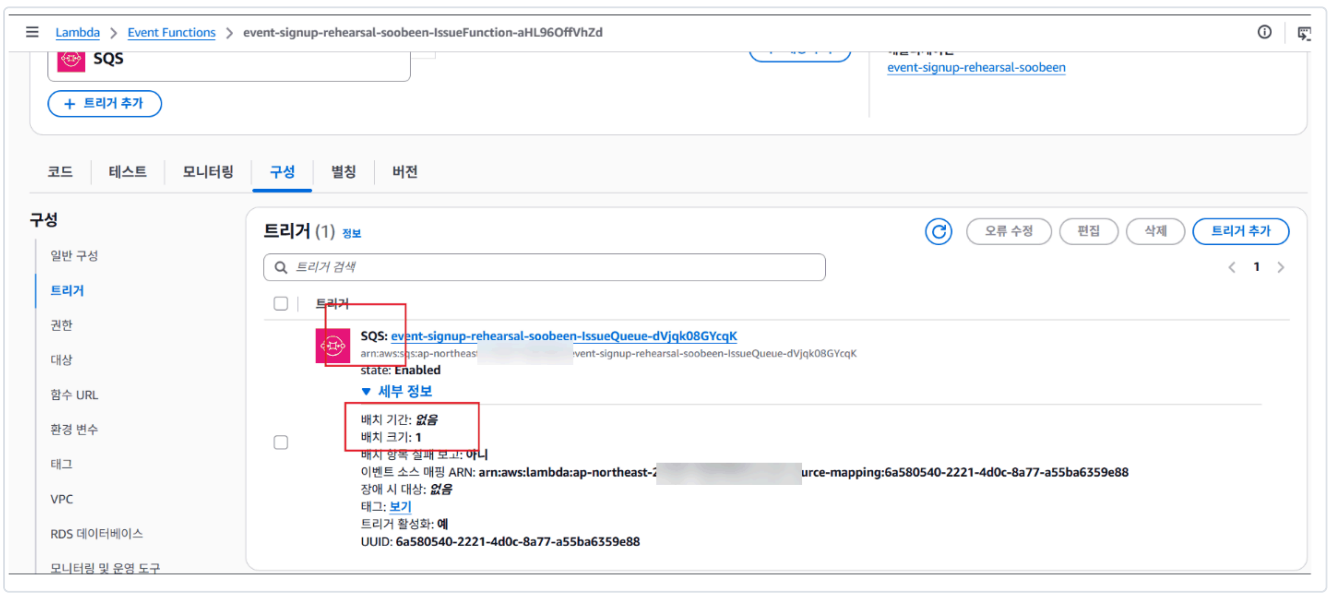

리소스 (24)
리소스 검색
1
논리적 ID
물리적 ID
유형
상태
RegistrationEventBus
event-signup-rehearsal-soobeen-bus
AWS::Events::EventBus
CREATE_COMP
RegistrationRule
event-signup-rehearsal-soobeen-bus|event-signup-rehearsal-soobeen-RegistrationRule-GW3Ec8X5v3TG
AWS::Events::Rule
CREATE_COMP
RegistrationsTable
event-signup-rehearsal-soobeen-RegistrationsTable-LUKCGY3IACI5
AWS::DynamoDB::Table
CREATE_COMP
ResendFunction
event-signup-rehearsal-soobeen-ResendFunction-clfSBvP3rOyM
AWS::Lambda::Function
CREATE_COMP
ResendFunctionPostPermis
event-signup-rehearsal-soobeen-
AWS::Lambda::Permissio
CREATE_COMP

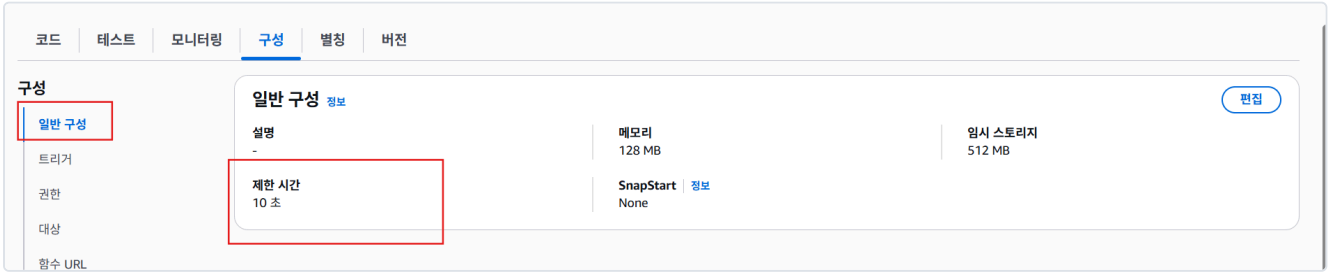

Amazon EventBridge
규칙
event-signup-rehearsal-soobeen-RegistrationRule-GW3Ec8X5v3TG
Amazon EventBridge
대시보드
개발자 리소스
자세히 알아보기
샌드박스
Quick Starts
버스
이벤트 버스
규칙 업데이트됨
글로벌 엔드포인트
아카이브
다시 재생
파이프
파이프
Scheduler
일정
일정 그룹
예약된 규칙(레거시) 업데이트됨
event-signup-rehearsal-soobeen-RegistrationRule-GW3Ec8X5v3TG
편집
비활성화
삭제
CloudFormation 템플릿
규칙 세부 정보 정보
규칙 이름
event-signup-rehearsal-soobeen-RegistrationRule-GW3Ec8X5v3TG
설명
-
상태
활성화됨
규칙 ARN
-soobeen-bus/event-signup-rehearsal-soobeen-RegistrationRule-GW3Ec8X5v3TG
이벤트 버스 이름
event-signup-rehearsal-soobeen-bus
이벤트 버스 ARN
earsal-soobeen-bus
유형
표준
이벤트 패턴
대상
모니터링
태그
이벤트 패턴 정보
편집
{
"detail-type": ["RegistrationReceived"],
"source": ["event-signup.registration"]
}

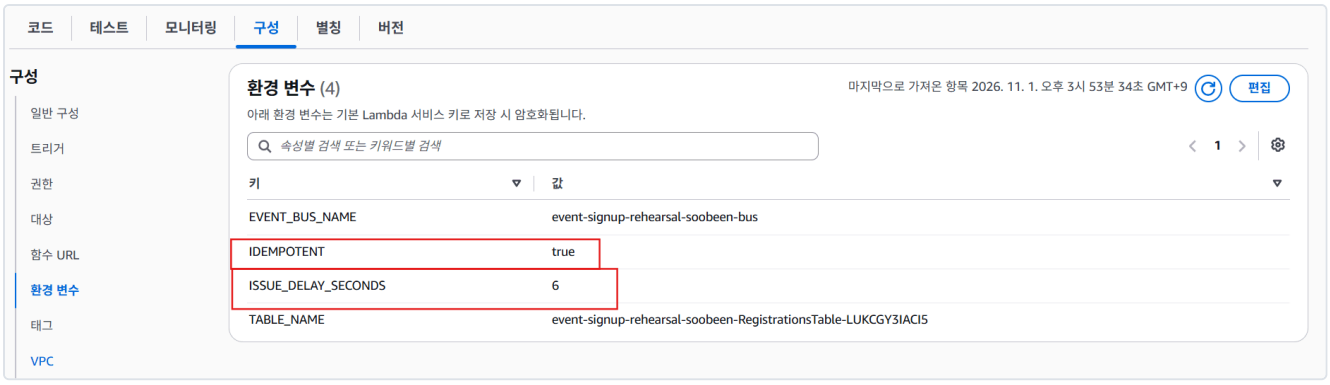

리소스 (24)
리소스 검색
1
논리적 ID
물리적 ID
유형
상태
VijmNGdBzmJF
IssueLogGroup
/asbg/event-signup-rehearsal-soobeen/issue
AWS::Logs::LogGroup
CREATE_COMP
IssueQueue
https://sqs.ap-northeast-2.amazonaws.com/ event-signup-rehearsal-soobeen-IssueQueue-dVjqk08GYcqK
AWS::SQS::Queue
CREATE_COMP
IssueQueuePolicy
event-signup-rehearsal-soobeen-IssueQueuePolicy-fzNXVhU0eaXk
AWS::SQS::QueuePolicy
CREATE_COMP
RegisterFunction
event-signup-rehearsal-soobeen-RegisterFunction-
AWS::Lambda::Function
UPDATE_COMP


Amazon SQS
대기열
event-signup-rehearsal-soobeen-IssueQueue-dVjqk08GYcqK
메시지 보존 기간
4일
기본 표시 제한 시간
1분
사용 가능한 메시지
0
전송 지연
0초
이동 중인 메시지(다른 소비자는 사용할 수 없음)
0
메시지 수신 대기 시간
0초
지연된 메시지
0
콘텐츠 기반 중복 제거
-
높은 처리량 FIFO
-
중복 제거 범위
-
FIFO 처리량 한도
-
리드라이브 허용 정책
-
대기열 정책
모니터링
SNS 구독
Lambda 트리거
EventBridge 파이프
배달 못한 편지 대기열
태그 지정
암호화
배달 못한 편지 대기열 리드라이브 태스크
배달 못한 편지 대기열 정보
편집
배달 못한 편지 대기열로 전송할 수 없는 메시지를 전송합니다. 이는 리드라이브 정책이라고도 합니다.
대기열
arn:aws:sqs:ap-northeast-
event-signup-rehearsal-soobeen-IssueDeadLetterQueue
-M0kKKvLJHvlT
최대 수신 수
3


리소스 (24)
리소스 검색
1
논리적 ID
물리적 ID
유형
상태
IssueDeadLetterQueue
https://sqs.ap-northeast-2.amazonaws.com/event-signup-rehearsal-soobeen-IssueDeadLetterQueue-M0kKKvLJHvlT
AWS::SQS::Queue
CREATE_COMP
IssueFunction
event-signup-rehearsal-soobeen-IssueFunction-aHL96OffVhZd
AWS::Lambda::Function
CREATE_COMP
IssueFunctionFromQueue
6a580540-2221-4d0c-8a77-a55ba6359e88
AWS::Lambda::EventSourceMapping
CREATE_COMP
IssueFunctionRole
event-signup-rehearsal-soobeen-IssueFunctionRole-VijmNGdBzmJF
AWS::IAM::Role
CREATE_COMP
IssueLogGroup
/asbg/event-signup-rehearsal-
AWS::Logs::LogGroup
CREATE_COMP


Lambda > Event Functions > event-signup-rehearsal-soobeen-IssueFunction-aHL96OffVhZd
SQS
+ 트리거 추가
event-signup-rehearsal-soobeen
코드
테스트
모니터링
구성
별칭
버전
구성
일반 구성
트리거
권한
대상
함수 URL
환경 변수
태그
VPC
RDS 데이터베이스
모니터링 및 운영 도구
트리거 (1) 정보
오류 수정
편집
삭제
트리거 추가
트리거 검색
1
트리거
SQS: event-signup-rehearsal-soobeen-IssueQueue-dVjqk08GYcqK
state: Enabled
세부 정보
배치 기간: 없음
배치 크기: 1
이벤트 소스 매핑 ARN: arn:aws:lambda:ap-northeast-
장애 시 대상: 없음
태그: 보기
트리거 활성화: 예
UUID: 6a580540-2221-4d0c-8a77-a55ba6359e88

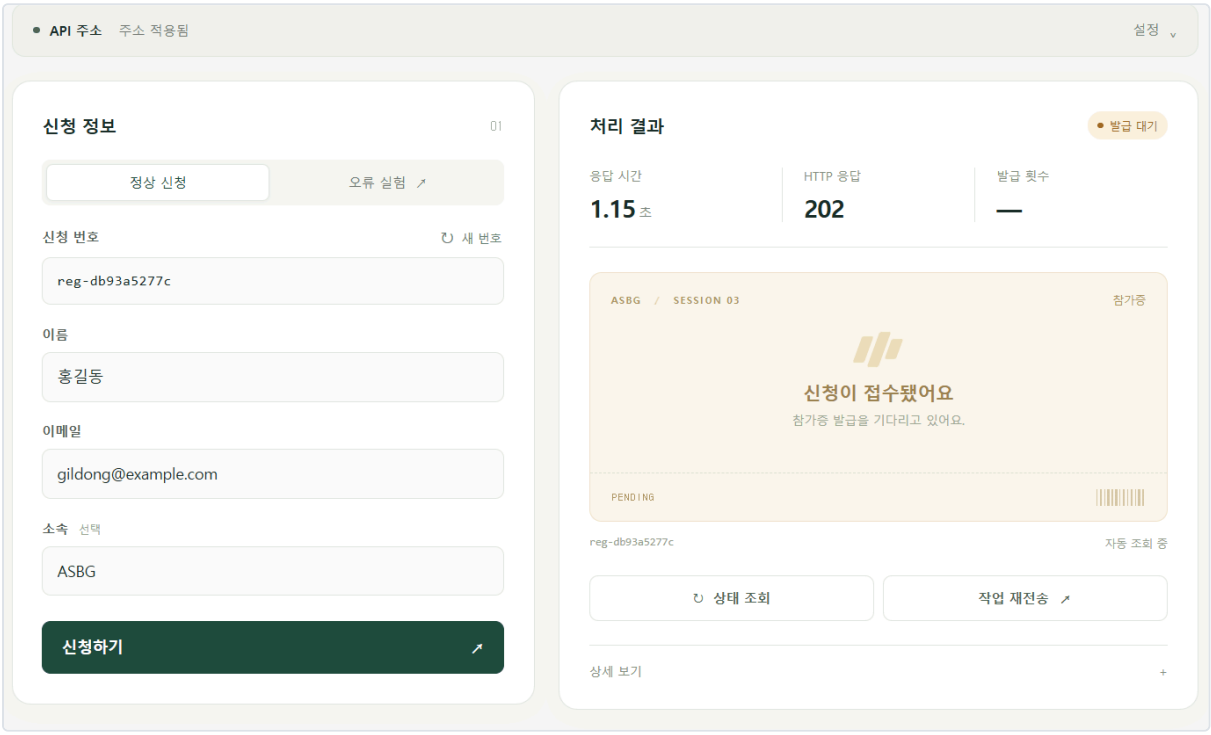

코드
테스트
모니터링
구성
별칭
버전
구성
일반 구성
트리거
권한
대상
함수 URL
일반 구성 정보
편집
설명
-
메모리
128 MB
임시 스토리지
512 MB
제한 시간
10 초
SnapStart 정보
None



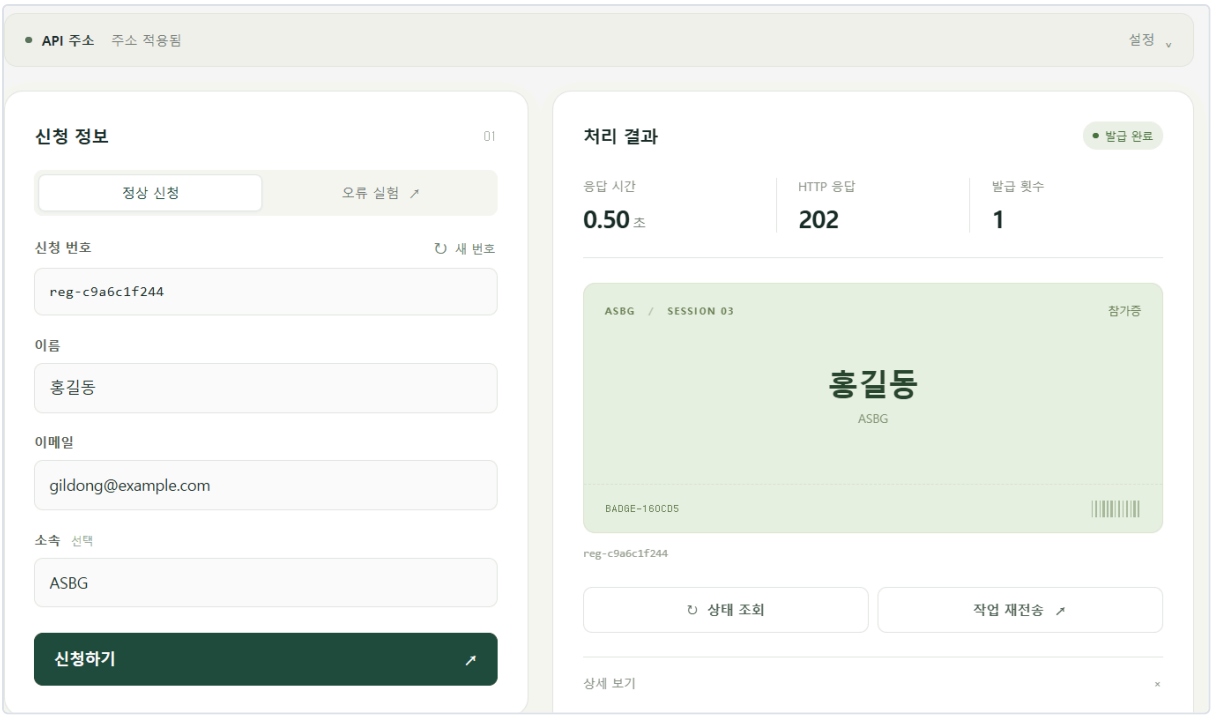
60초는 함수 Timeout 10초의 6배이다. 최대 수신 횟수 3은 **첫 수신을 포함**하며 수업 중 관찰을 위한 값이다. AWS는 최소 5를 권장하므로 운영 값으로 그대로 받아들이지 않는다. [AWS 설정 가이드]

2-2. 접수와 발급의 분리 (4분)

1. `정상 신청` → `신청하기`를 누른다. **새 정상 신청 번호**를 기록한다.



2. **응답:** `202` 와 `PENDING`이 반환된다. 발급 6초가 끝나기를 기다리지 않고 응답한다. 실제 응답 시간은 콜드 스타트·네트워크 등의 영향을 받을 수 있다.



3. **상태:** 잠시 뒤 `ISSUED`와 참가증이 나타나는지 확인한다. 자동 조회가 멈췄거나 결과가 늦으면 `상태 조회`를 누른다.
4. **로그:** 터미널 로그의 시각은 UTC이다(한국 시간보다 9시간 이르다). 터미널에서는 `Show-Logs IssueFunction`으로 발급 함수 로그를 실시간으로 볼 수 있다.

```
PS C:\Users\Clush_> Show-Logs IssueFunction
2026-11-01T06:58:18 INIT_START Runtime Version: python:3.12.mainlinev2.v82      Runtime Version ARN: arn:aws:lambda:ap-n
ortheast-2::runtime:a8b19198a828662ade07d066a1dce047429a58691663467 47ab7d57435583a29
2026-11-01T06:58:18 START RequestId: f387cd74-f59a-5572-ab8c-ae1410d07f5b Version: $LATEST
2026-11-01T06:58:18 {"event": "issue_started", "registrationId": "reg-db93a5277c", "messageId": "5985570c-b671-4493-8413
-e4f496b729ff", "receiveCount": "1"}
2026-11-01T06:58:25 {"event": "badge_saved", "registrationId": "reg-db93a5277c", "badgeCode": "BADGE-F3C626"}
2026-11-01T06:58:25 END RequestId: f387cd74-f59a-5572-ab8c-ae1410d07f5b
2026-11-01T06:58:25 REPORT RequestId: f387cd74-f59a-5572-ab8c-ae1410d07f5b      Duration: 6295.98 ms    Billed Duration:
 6786 ms        Memory Size: 128 MB     Max Memory Used: 92 MB  Init Duration: 489.80 ms
2026-11-01T07:02:06 START RequestId: c31c8982-40e2-5702-80d2-eae2a6fef0ce Version: $LATEST
2026-11-01T07:02:06 {"event": "issue_started", "registrationId": "reg-c9a6c1f244", "messageId": "3941fbd0-6626-4501-ae26
-460a34328256", "receiveCount": "1"}
2026-11-01T07:02:12 {"event": "badge_saved", "registrationId": "reg-c9a6c1f244", "badgeCode": "BADGE-160CD5"}
2026-11-01T07:02:12 END RequestId: c31c8982-40e2-5702-80d2-eae2a6fef0ce
2026-11-01T07:02:12 REPORT RequestId: c31c8982-40e2-5702-80d2-eae2a6fef0ce      Duration: 6215.80 ms    Billed Duration:
 6216 ms        Memory Size: 128 MB     Max Memory Used: 92 MB
```

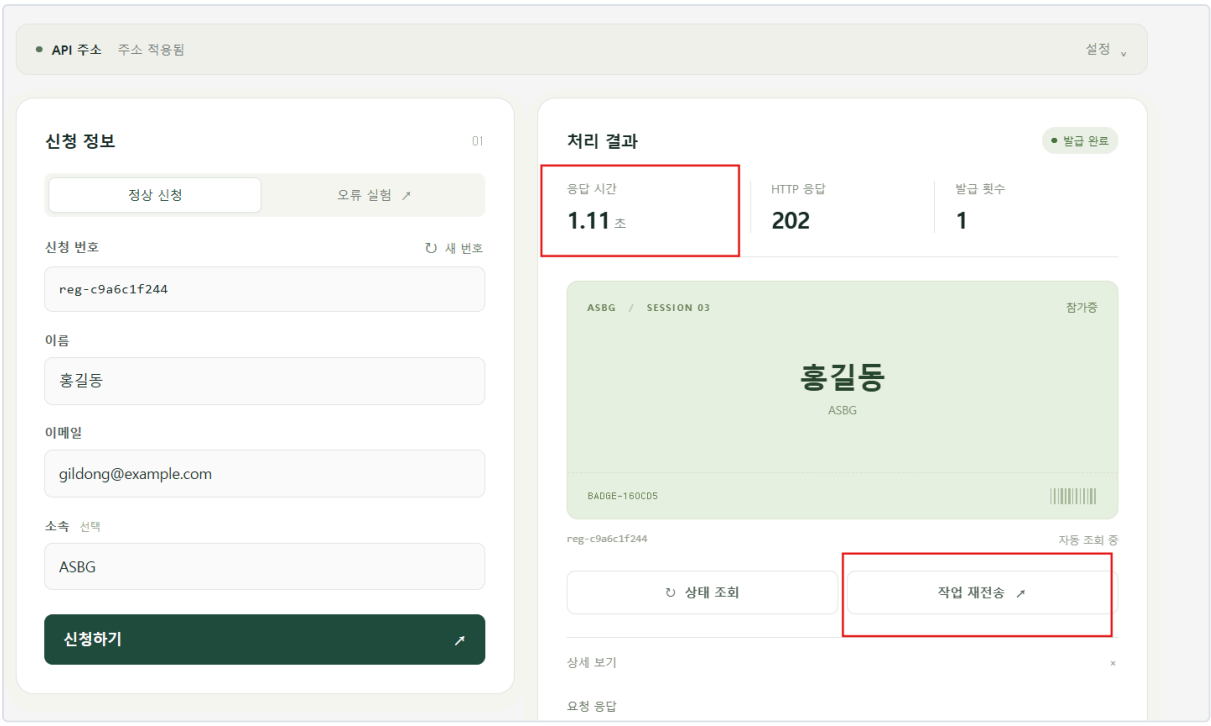
체크: 최초 응답은 `202`, 나중의 DB 결과는 `ISSUED`, `issueCount = 1`이다. 접수 성공과 작업 완료의 차이를 설명할 수 있다.

2-3. 작업 재전송 (4분)

1. 정상 신청이 `ISSUED`인지 확인하고 `badgeCode`, `issueCount`를 기록한다.



2. `작업 재전송`을 누른다. 저장된 신청으로 새 이벤트를 보내 같은 작업이 다시 들어온 상황을 만든다.



3. 잠시 뒤 `상태 조회`와 DB에서 **기존 참가증 코드 유지,** `issueCount = 1` **유지**를 확인한다.

요청 응답


```
{
  "message": "같은 작업을 한 번 더 보냈습니다. (실습용)",
  "registrationId": "reg-c9a6c1f244"
}
```


DB 조회 결과


```
{
  "badgeCode": "BADGE-160CD5",
  "badgeText": "홍길동 / ASBG",
  "createdAt": "2026-11-01T07:02:05+00:00",
  "email": "gildong@example.com",
  "issueCount": 1,
  "issuedAt": "2026-11-01T07:02:12+00:00",
  "name": "홍길동",
  "organization": "ASBG",
  "registrationId": "reg-c9a6c1f244",
  "status": "ISSUED"
}
```


진행 기록

1. 0.0s 신청 전송 · reg-c9a6c1f244
2. 0.5s HTTP 202 · 0.50초
3. 2.3s DB · PENDING
4. 7.0s DB · ISSUED · 발급 1회
5. 114.8s 작업 재전송 · reg-c9a6c1f244
6. 115.9s HTTP 202 · 1.11초
7. 131.7s 자동 조회 종료

4. 발급 함수 로그의 `badge_already_issued_skip` 과 `step2/src/issue.py` 의 `save_badge` 조건을 찾아본다.

체크: 이벤트는 다시 왔지만 DB의 발급 결과는 한 번만 반영되었다. 이것이 이 예제가 보장하는 멱등성의 범위이다.

선택 실험은 세 가지이다.

선택	방법	결과
접수 중복 비교	같은 내용·번호로 `신청하기` 를 다시 누른다	접수 단계의 중복 방어로 `200` 을 반환하고 새 이벤트를 만들지 않는다
충돌 비교	같은 번호의 이름·이메일·소속을 바꿔 보낸다	`409` 이며 원래 신청은 바뀌지 않는다. 새로운 사람은 `새 번호` 로 신청한다
멱등성 끄기	발급 함수의 `IDEMPOTENT` 를 `false` 로 바꿔 새 정상 신청과 작업 재전송을 비교한다	발급 횟수가 증가할 수 있다. 끝나면 반드시 `true` 로 복원한다

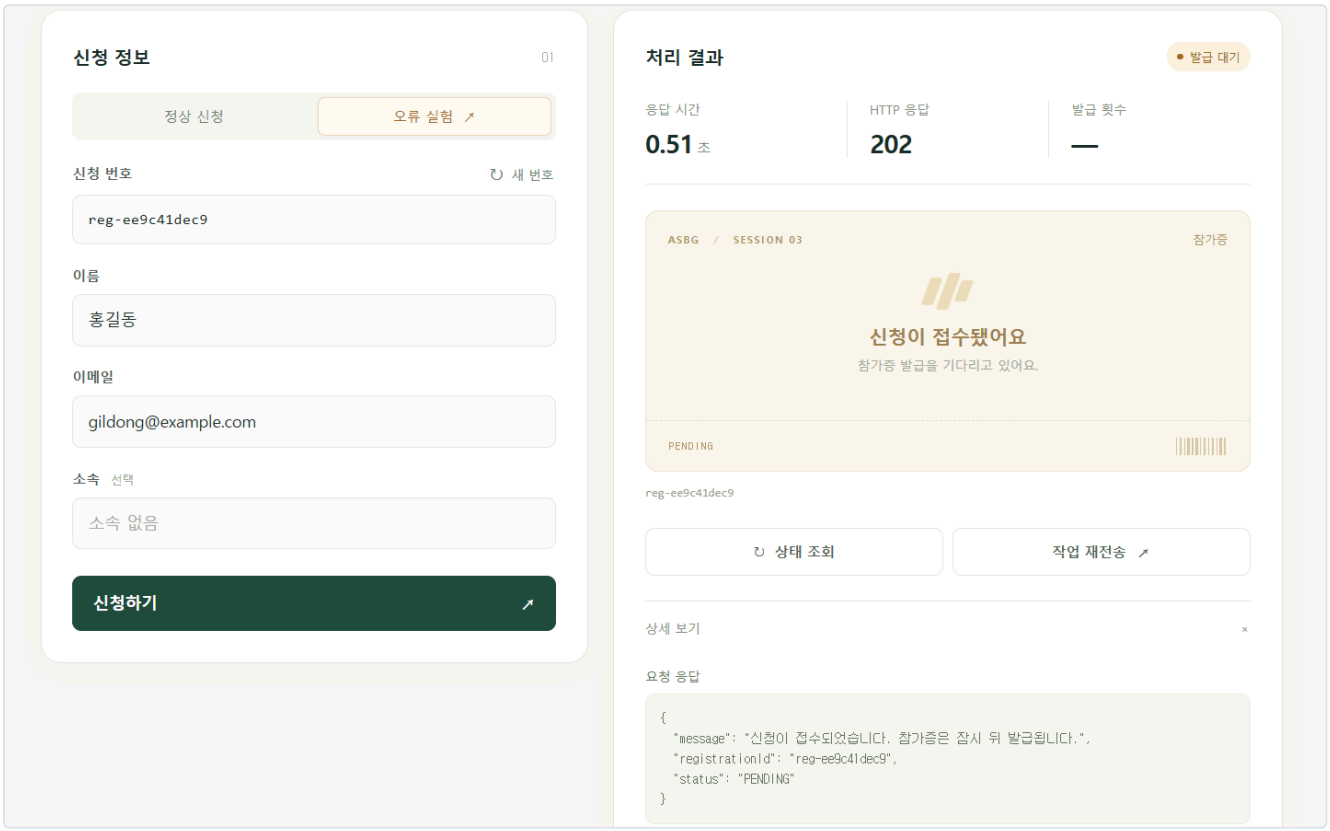
2-4. 원인 수정과 복구 (12분)

실패한 신청을 원래 번호 그대로 복구하는 여섯 단계이다. 현장 시간 안에 모두 끝내기 어려워 둘로 나눈다.

구간	현장에서	이후
①–③ 재조회, 로그, DLQ	각자 직접 한다	과제 C4의 필수 범위이다
④–⑥ 수정, 재처리, 검증	발표자가 시연한다. 따라 하지 않아도 된다	과제 C4의 "권장 · 원인 수정과 재처리"로 각자 한다

④–⑥을 과제로 하려면 **스택을 지우지 않고 유지한다.** 실패 번호와 DLQ 메시지가 그 스택에 있기 때문이다. 복구를 확인한 뒤에 "4. 리소스 정리"를 한다.

① 실패 신청 재조회. 신청 번호 칸에 미리 기록한 실패 번호를 입력하고 `상태 조회` 한다. DB에는 `PENDING` 이 남아 있다. 상태 조회만으로 실패 원인이 나타나지는 않는다.

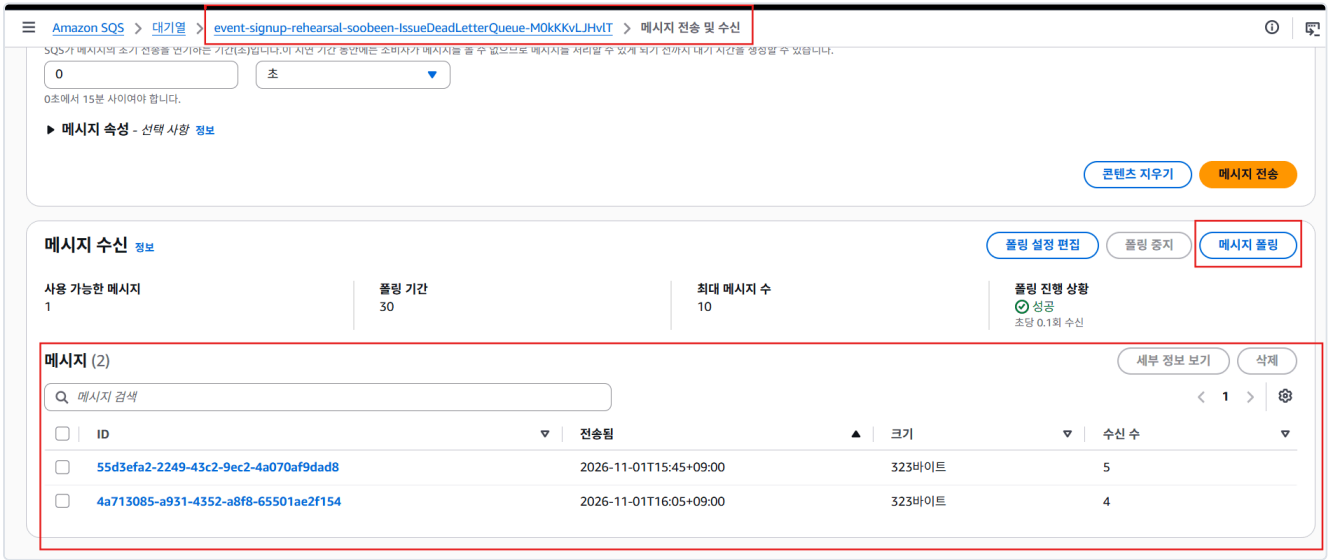


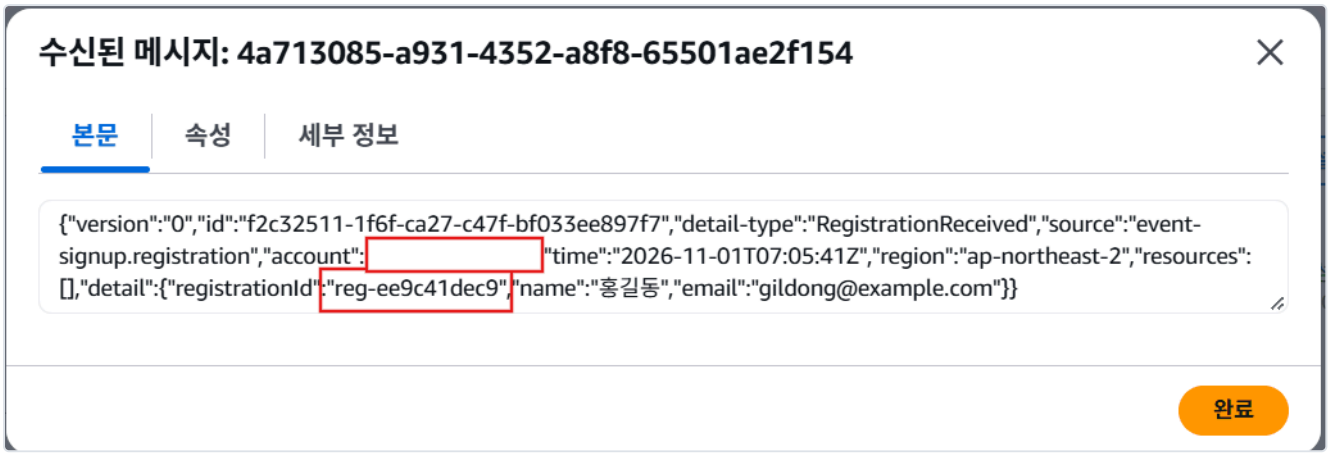
② 로그 확인. `Show-Logs IssueFunction` 으로 `issue_started` 의 `receiveCount` 가 1, 2, 3으로 늘고, 매번 `issue_failed` 에 `errorType` 과 `error` 가 남는지 확인해 오류 종류를 기록한다. 함수가 던진 오류의 추적 정보에서 파일·줄도 찾는다. 입력의 선택 항목과 발급 코드가 기대하는 데이터가 일치하는지 비교한다.

```
2026-11-01T07:04:01 START RequestId: 85835281-cf9a-57c1-abd9-31e1290cc690 Version: $LATEST
2026-11-01T07:04:01 {"event": "issue_started", "registrationId": "reg-c9a6c1f244", "messageId": "b8ac7b06-ce01-4748-9ce1
-294e50da2c61", "receiveCount": "1"}
2026-11-01T07:04:07 {"event": "badge_already_issued_skip", "registrationId": "reg-c9a6c1f244"}
2026-11-01T07:04:07 END RequestId: 85835281-cf9a-57c1-abd9-31e1290cc690
2026-11-01T07:04:07 REPORT RequestId: 85835281-cf9a-57c1-abd9-31e1290cc690      Duration: 6289.05 ms    Billed Duration:
 6290 ms        Memory Size: 128 MB     Max Memory Used: 92 MB
2026-11-01T07:05:41 START RequestId: fb3fab35-f1b0-539b-a956-c85516b2a40c Version: $LATEST
2026-11-01T07:05:41 {"event": "issue_started", "registrationId": "reg-ee9c41dec9", "messageId": "4a713085-a931-4352-a8f8
-65501ae2f154", "receiveCount": "1"}
2026-11-01T07:05:47 {"event": "issue_failed", "registrationId": "reg-ee9c41dec9", "messageId": "4a713085-a931-4352-a8f8-
65501ae2f154", "errorType": "KeyError", "error": "'organization'"}
2026-11-01T07:05:47 [ERROR] KeyError: 'organization'
Traceback (most recent call last):
  File "/var/task/issue.py", line 88, in handler
    badge_code, badge_text = make_badge(detail)
  File "/var/task/issue.py", line 35, in make_badge
    return code, build_badge_text(detail)
  File "/var/task/issue.py", line 28, in build_badge_text
    return f"{detail['name']} / {detail['organization']}"
2026-11-01T07:05:47 END RequestId: fb3fab35-f1b0-539b-a956-c85516b2a40c
2026-11-01T07:05:47 REPORT RequestId: fb3fab35-f1b0-539b-a956-c85516b2a40c      Duration: 6003.85 ms    Billed Duration:
 6004 ms        Memory Size: 128 MB     Max Memory Used: 92 MB
```

```
2026-11-01T07:06:41 START RequestId: e46f8625-73c9-54da-92ef-e54594f73314 Version: $LATEST
2026-11-01T07:06:41 {"event": "issue_started", "registrationId": "reg-ee9c41dec9", "messageId": "4a713085-a931-4352-a8f8
-65501ae2f154", "receiveCount": "2"}
2026-11-01T07:06:47 {"event": "issue_failed", "registrationId": "reg-ee9c41dec9", "messageId": "4a713085-a931-4352-a8f8-
65501ae2f154", "errorType": "KeyError", "error": "'organization'"}
2026-11-01T07:06:47 [ERROR] KeyError: 'organization'
Traceback (most recent call last):
  File "/var/task/issue.py", line 88, in handler
    badge_code, badge_text = make_badge(detail)
  File "/var/task/issue.py", line 35, in make_badge
    return code, build_badge_text(detail)
  File "/var/task/issue.py", line 28, in build_badge_text
    return f"{detail['name']} / {detail['organization']}"
2026-11-01T07:06:47 END RequestId: e46f8625-73c9-54da-92ef-e54594f73314
2026-11-01T07:06:47 REPORT RequestId: e46f8625-73c9-54da-92ef-e54594f73314      Duration: 6021.73 ms    Billed Duration:
 6022 ms        Memory Size: 128 MB     Max Memory Used: 92 MB
```

③ **DLQ 확인.** SQS → 해당 DLQ → Send and receive messages → Poll for messages에서 Body의 신청 번호를 확인한다. 반복 실패한 메시지가 격리되어 있고 다른 정상 신청은 처리된다. **메시지를 삭제하지 않는다.** 터미널에서는 `Show-Dlq` 가 DLQ의 메시지 수를 보여 주고, `Show-DlqMessages` 가 내용을 보여 준다. `Show-DlqMessages` 는 메시지를 지우지 않지만 수신 횟수를 늘리므로 반복해서 실행하지 않는다.





- 가시성 60초와 최대 수신 횟수 3에서 도착까지 **대략 3분 이상** 잡되 정확한 시간을 보장하지 않는다. Lambda 백오프·폴링과 상태 반영에 따라 더 늦을 수 있다.
- 원본 발급 큐를 콘솔로 직접 Poll하면 수신 횟수와 처리를 방해할 수 있으니 로그로 관찰한다.

④ 코드 수정. 소속은 선택 입력으로 유지하고, 비어 있어도 정상 발급되도록 `step2/src/issue.py`를 수정한다. 로컬에서 검증하고 배포한다.

```
py -m unittest -b tests.test_handlers -v
py -m unittest -b tests.test_exercise -v (전후 체크)
Deploy-Step2 -Delay 6
```

수정 전에는 두 번째 테스트가 실패하고, 수정 후에는 모두 통과해야 한다. 시간이 부족하면 Lambda 콘솔 Code에서 수정 후 Deploy할 수 있다. 이 경우 같은 변경을 로컬 코드에도 반영하고 과제 제출 전 테스트한다.

```
KeyError: 'organization'

Stdout:
{"event": "request_received", "registrationId": "reg-9"}
{"event": "registration_saved", "registrationId": "reg-9"}
{"event": "event_published", "registrationId": "reg-9"}
{"event": "issue_started", "registrationId": "reg-9", "messageId": "msg-2",
{"event": "issue_failed", "registrationId": "reg-9", "messageId": "msg-2",
ion'"}
{"event": "request_received", "registrationId": "reg-9"}
{"event": "registration_saved", "registrationId": "reg-9"}
{"event": "event_published", "registrationId": "reg-9"}
{"event": "issue_started", "registrationId": "reg-9", "messageId": "msg-4",
{"event": "issue_failed", "registrationId": "reg-9", "messageId": "msg-4",
ion'"}

----------------------------------------------------------------------
Ran 1 test in 0.030s

FAILED (errors=2)
```

```
PS C:\Users\Clush_\Downloads\source\source> py -m unittest -b tests.test_exercise -v
test_소속을_비우고_신청해도_참가증이_발급되어야_한다 (tests.test_exercise.OrganizationOptionalTest.test_소속을_비우고_신청해도_참가증이_발급되어야_한다) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.026s

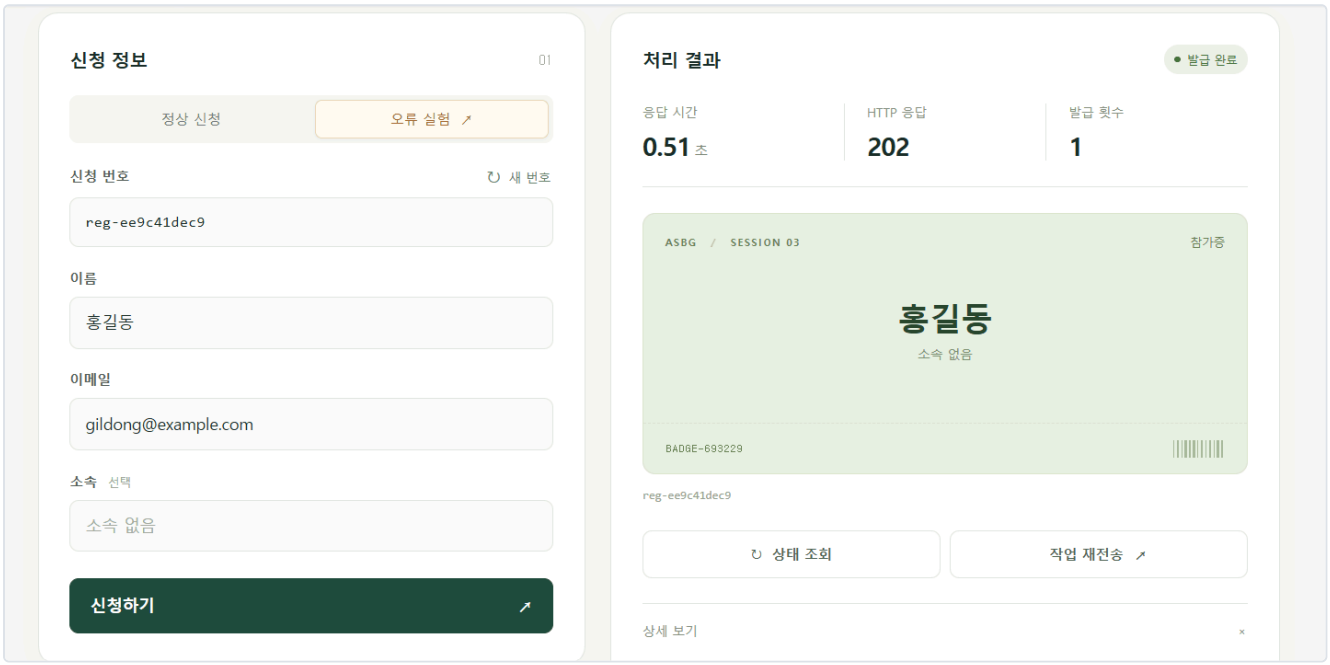
OK
```

⑤ 재처리. 아래 명령은 DLQ의 메시지를 이 스택의 발급 큐로 되돌린다.

```
Start-Redrive
Show-Redrive
```

콘솔에서는 DLQ → Start DLQ redrive에서 **목적지 큐를 지정**하고 이 스택의 `IssueQueue`를 선택한다. 대상 ARN은 `Show-Outputs`의 `QueueArn`으로 확인한다.

⑥ 원래 번호로 검증. 새 신청을 만드는 대신 `상태 조회`를 눌러 `ISSUED`, `issueCount = 1`을 확인한다. 큐의 메시지 수는 근삿값이므로 DB 결과와 로그도 함께 확인한다.



DB 조회 결과


```
{
  "badgeCode": "BADGE-693229",
  "badgeText": "홍길동 / -",
  "createdAt": "2026-11-01T07:05:41+00:00",
  "email": "gildong@example.com",
  "issueCount": 1,
  "issuedAt": "2026-11-01T07:19:09+00:00",
  "name": "홍길동",
  "registrationId": "reg-ee9c41dec9",
  "status": "ISSUED"
}
```


지해 기록

체크: 실패 입력 → 오류 로그 → 재시도 → DLQ의 증거를 남겼다(①–③). 수정 → 재처리 → 원래 신청 복구는 과제로 이어서 증거를 남긴다(④–⑥). DLQ가 늦으면 발표자 시연을 관찰하고 자신의 미검증 항목을 기록한다. 시연 화면은 내 증거로 쓰지 않는다.

3. 과제 기록

배포 자료의 `ASSIGNMENT.md`와 `SUBMISSION_TEMPLATE.md`를 사용한다. 오늘 기록할 것은 네 가지이다.

- 내 아키텍처 그림과 각 서비스의 역할
- Timeout 10초, Visibility 60초, 최대 수신 횟수 3 등 설정값과 선택 이유
- 정상·중복·실패의 입력, 예상 결과, 실제 결과, 신청 번호와 로그/화면
- 수정한 코드와 검증한 범위, 아직 확인하지 못한 항목

공식 과제는 자율 참여이며 제출 마감은 **2026-11-17 21:00 KST**이다. 공식 제출 안내의 템플릿과 PR 규칙을 따른다.

4. 리소스 정리

증거를 저장한 뒤 배포 때와 같은 스택 이름·리전으로 실행한다. 2-4의 ④–⑥을 과제로 이어서 할 때는 복구를 확인한 뒤에 한다.

```
Remove-Stack
```

CloudFormation에서 스택 삭제 완료를 확인한다. 이 버전의 Lambda 로그 그룹도 스택이 관리하므로 함께 삭제된다.

이전 코드로 배포했던 자동 생성 로그 그룹이나 SAM의 공용 배포 보관용 스택 `aws-sam-cli-managed-default`는 별개이다. 다른 실습에서 사용하는 공용 리소스를 함께 지우지 않도록 대상을 확인한다.

삭제됐는지 확인한다. 명령이 끝났다는 출력만 믿지 않고 콘솔에서 직접 본다.

1. **[AWS 콘솔]** 리전이 서울인지 확인하고 CloudFormation → Stacks를 연다. 내 스택이 목록에서 사라졌고, Filter status를 **Deleted**로 바꾸면 `DELETE_COMPLETE`로 보이면 된다. `DELETE_IN_PROGRESS`는 아직 삭제 중이므로 기다린다. `DELETE_FAILED`이면 Events 탭에서 지워지지 않은 리소스와 이유를 읽는다. 삭제된 스택 보기
2. **[AWS 콘솔]** 서비스별 목록에서 이름에 내 스택 이름이 든 항목이 없는지 본다. 검색창에 스택 이름을 넣으면 빠르다.

서비스	메뉴	없어야 하는 것
Lambda	함수 목록	`<스택 이름>-RegisterFunction-…`, `StatusFunction`, `ResendFunction`, `IssueFunction`
API Gateway	APIs	내 API. API ID는 주소 `https://<API ID>.execute-api…`의 앞부분이다. 삭제 뒤에는 `Get-OutApiUrl`로 얻을 수 없으니 **`Remove-Stack` 전에 적어 둔다**
DynamoDB	Tables	`<스택 이름>-RegistrationsTable-…`
SQS	Queues	`<스택 이름>-IssueQueue-…`, `<스택 이름>-IssueDeadLetterQueue-…`
EventBridge	Event buses	`<스택 이름>-bus`
CloudWatch	Logs → Log groups	`/asbg/<스택 이름>/register`, `status`, `resend`, `issue`

3. **[터미널]** 스택이 없다는 오류가 나오면 삭제된 것이다.

```
Show-Outputs
```

콘솔 목록은 새로 고침해야 바뀌는 경우가 있다. 항목이 남아 있으면 이름이 내 스택 것인지 먼저 확인하고, 다른 사람의 스택 리소스는 지우지 않는다. `aws-sam-cli-managed-default` 스택과 그 S3 버킷은 위 표에 없는 것이 정상이며 이 실습의 삭제 대상이 아니다.

연습용 자격 증명도 지운다. 0장에서 만든 IAM 사용자의 액세스 키와 임시로 붙인 `AdministratorAccess`가 남아 있으면 계정이 계속 열려 있는 셈이다.

1. **[AWS 콘솔]** IAM → Users → 해당 사용자 → Security credentials → Access keys에서 키를 **비활성화한 뒤 삭제**한다.
2. **[AWS 콘솔]** Permissions에서 `AdministratorAccess`를 제거한다. 연습용 사용자를 더 쓰지 않으면 사용자도 삭제한다.
3. **[터미널]** 내 컴퓨터에 남은 키를 지운다. `%USERPROFILE%\.aws\credentials`의 `[default]` 항목을 삭제한다.

문제 해결

증상별로 먼저 확인할 것이다.

증상	다음 확인
도구나 로그인이 없음	`Test-Setup`. 명령을 찾을 수 없다고 나오면 도구를 설치하고 PowerShell을 새로 연다
배포 시작에서 `AccessDenied`, `is not authorized to perform`	오류의 `User:` 줄이 내가 만든 IAM 사용자인지 `aws sts get-caller-identity`로 확인하고, 그 사용자에 권한(0장의 2번)이 붙어 있는지 본다. 이 오류는 스택이 만들어지기 전에 터미널에만 나오므로 CloudFormation Events에는 보이지 않는다. 첫 배포는 SAM의 공용 스택 `aws-sam-cli-managed-default`를 먼저 만들어서 CloudFormation 권한이 필요하다
`Get-Out`, `Deploy-Step1` 같은 명령을 찾을 수 없음	새 창이거나 설정 블록을 붙여 넣지 않았다. 0장의 블록을 다시 붙여 넣는다
배포 실패 / 리소스가 안 보임	CloudFormation Events, 같은 계정·리전·스택인지 확인한다
브라우저 요청 실패	신청 화면을 파일(`file://`)로 열었다면 0장의 로컬 서버(`http://localhost:8000/`)로 다시 연다. F12 Console에 `origin 'null' has been blocked by CORS policy`가 보이면 이 경우이다. 그 외에는 `Get-Out ApiUrl` 주소를 다시 적용하고 브라우저 Network/Console에서 오류를 확인한다. 주소 적용은 연결 검사가 아니다
실습 1에서 재전송이 안 됨	재전송 API는 실습 2 배포 후에만 생긴다. 화면 탭만 바꿔서는 배포되지 않는다
`409`가 나옴	이미 사용한 번호에 다른 입력을 넣었다. 원래 내용으로 재요청하거나 `새 번호`를 사용한다
`503` 이후 `PENDING`	이벤트 발행 실패일 수 있다. 접수 함수 로그와 권한을 확인하고 원인 해결 후 그 번호의 `작업 재전송`을 사용한다
로그가 없거나 늦음	로그 그룹·시간 범위·요청 여부를 확인하고 새로 고친다. 권한·설정 문제도 확인한다
DLQ에 아직 없음	발급 함수 오류/수신 횟수와 큐 설정을 확인한다. 3분 안에 반드시 도착하는 것은 아니다
redrive 뒤에도 메시지가 남음	먼저 수정 배포 여부를 확인한다. 콘솔에서 읽은 메시지는 잠시 보이지 않을 수 있으니 가시성 시간이 지난 뒤 재확인한다

부록. curl 사용

화면 대신 `curl.exe`로 같은 실험을 할 수 있다. PowerShell에서 `curl`은 다른 명령의 별칭이므로 **`curl.exe`**를 쓴다. 본문은 파일로 만들어 보내며(영문으로 쓴다), 재실행 시 신청 번호를 바꾼다.

```
$API = Get-Out ApiUrl

# 정상 신청: 실습 1은 201, 실습 2는 202
@'
{"registrationId":"reg-cli-normal-
1","name":"Hong","email":"hong@example.com","organization":"ASBG"}
'@ | Set-Content -Encoding ascii normal.json
curl.exe -s -w "`nHTTP %{http_code} / %{time_total}s`n" -X POST "$API/registrations" -H
"Content-Type: application/json"--data-binary "@normal.json"

# 조회
curl.exe -s "$API/registrations/reg-cli-normal-1"

# 작업 재전송: 실습 2
curl.exe -s -X POST "$API/registrations/reg-cli-normal-1/resend"

# 오류 실험: 실습 2에서 소속을 비운 새 신청
@'
{"registrationId":"reg-cli-failure-1","name":"Kim","email":"dev@example.com","organization":""}
'@ | Set-Content -Encoding ascii failure.json
curl.exe -s -X POST "$API/registrations" -H "Content-Type: application/json"--data-binary
"@failure.json"
```