

# 3분만에 생성하는 AI 기반 속품 제작 서비스

SkyWalker Project

SkyWalker

이수민



Presentation Date: 2025-01-03



# 이수민

프로젝트 팀장

lucyi030920@gmail.com

## 맡은 파트

Backend

Docker

ECR

ECS

Bedrock

Rekognition

Lambda

API Gateway

SNS

DynamoDB

IAM

CloudWatch

## 자격증

- 네트워크 관리사 2급
- AWS Certified Solutions Architect - Associate

## 프로젝트 팀원

## 맡은 파트

Frontend

Terraform

Amplify

CI/CD

CloudFront

WAF

Route 53

Cognito

IAM

CloudWatch

## 자격증

- 네트워크 관리사 2급
- 리눅스 마스터 2급

# 주제 선정 이유



행사안내	참여기업등록	전시관람안내	컨퍼런스	어워드	미디어센터
11:30 ~ 13:00					
13:00 ~ 13:30	<ul style="list-style-type: none"><li>총 팔로워 37만 틱톡맨 성장 스토리와 솟풀 크리에이터로 성공하는 노하우</li></ul>	틱톡맨(SOONENT) 석병선			
13:30 ~ 13:50	<ul style="list-style-type: none"><li>광고회사는 창의적 인재를 어떻게 채용할까?</li></ul>	이노션 최인학 파트장			
13:50 ~ 14:10	<ul style="list-style-type: none"><li>CJ 올리브영의 개발자는 어떻게 일하는가?</li></ul>	CJ올리브영 장회수 팀장			
14:10 ~ 14:30	<ul style="list-style-type: none"><li>'내 AI 커리어에 식스팩(6-SPEC) 달기' : SKT AI인재 육성 프로젝트로 살펴보는 6가지 성장 기회</li></ul>	SK텔레콤 역량혁신팀 허세정 매니저			
14:30 ~ 14:50	<ul style="list-style-type: none"><li>생성형AI 기술로 변화하는 업무 환경, 성공적인 적용법</li></ul>	AWS 박혜영 수석 솔루션즈 아키텍트			
14:50 ~ 15:10	<ul style="list-style-type: none"><li>규칙 없음 혁신 있음</li></ul>	넷플릭스 심상기 디렉터			

# 목차

- 01 프로젝트 개요
- 02 프로젝트 설계 및 계획
- 03 프로젝트 구현
- 04 프로젝트 성과
- 05 문제 해결 및 리스크 관리
- 06 향후 계획 및 결론
- 07 Q & A

# | 프로젝트 개요

01-1 프로젝트 배경

01-2 프로젝트 필요성

01-3 프로젝트 목표

01-4 프로젝트 기대효과



# 숏폼 시대



▶ 홈 > 오피니언

## [칼럼] 숏폼 시대, 콘텐츠 혁명의 전환점

▶ 김상기 기자 | Ⓛ 입력 2024.11.18 16:34 | 🗣 댓글 0

짧고 강렬한 콘텐츠, 미디어 환경과 소비자의 일상을 재편하다



출처: 숏폼 시대, 컨텐츠 혁명의 전환점,  
[https://www.k-trendynews.com/news/articleView.html?idxno=175746&utm\\_KtN\\_뉴스, 2024.11.18](https://www.k-trendynews.com/news/articleView.html?idxno=175746&utm_KtN_뉴스, 2024.11.18)

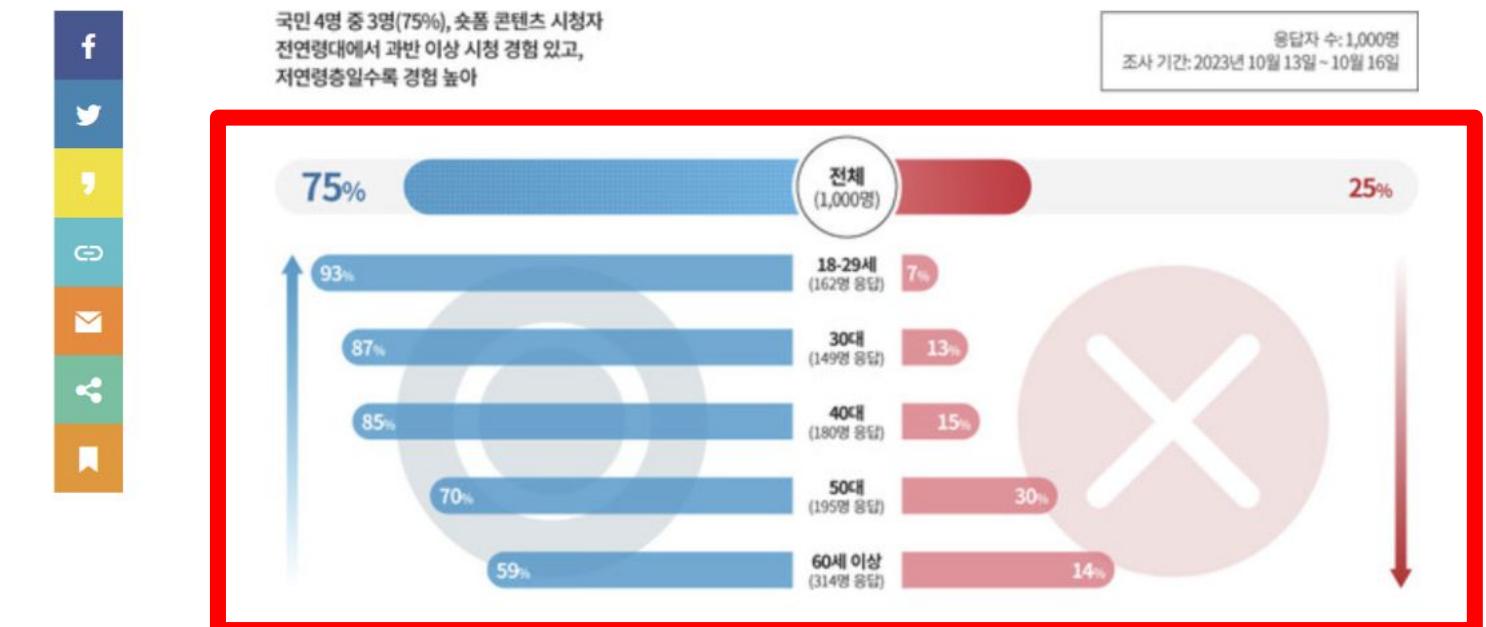
▶ 홈 > 라이프·엔터플랫폼

## 60대 60%도 '숏폼' 본다...숏폼 인기 비결 4가지

▶ 조윤하 기자 | Ⓛ 입력 2024.02.26 11:28 | 🗣 댓글 0

국민 4명 중 3명(75%), 숏폼 콘텐츠 시청자  
전연령대에서 과반 이상 시청 경험 있고,  
저연령층일수록 경험 높아

응답자 수: 1,000명  
조사 기간: 2023년 10월 13일 ~ 10월 16일



'숏폼'이 대세다. 숏폼(short form)은 10~60초 내외 길이의 짧은 영상 콘텐츠를 말한다. 지난해

60대 60%도 '숏폼' 본다... 숏폼 인기 비결 4가지,  
[https://www.stvnews.kr/news/articleView.html?idxno=10744&utm\\_STV\\_뉴스, 2024.02.26](https://www.stvnews.kr/news/articleView.html?idxno=10744&utm_STV_뉴스, 2024.02.26)

## 프로젝트 배경

숏폼 콘텐츠 시장의 현황과 한계점

- Instagram Reels, YouTube Shorts 등 숏폼 플랫폼의 급성장
- 현대 마케팅의 필수 요소로 부상

문제점

- 과도한 시간과 비용 소모
- 타겟 고객 맞춤형 콘텐츠 제작의 어려움
- 다국어 지원의 한계



# 프로젝트 필요성

숏폼 콘텐츠는 마케팅 필수 요소지만, 기업들은 제작 과정에서 과도한 비용과 시간, 맞춤형 콘텐츠 부족, 글로벌 확장성의 한계로 어려움을 겪고 있어 이를 해결하기 위한 플랫폼이 필요



## 비용과 시간 절감

- 콘텐츠 제작 과정의 높은 비용과 긴 제작 시간으로 경쟁력 약화
- 제작 자동화를 통해 시간과 비용 부담 해결

## 맞춤형 콘텐츠 제작

- 기존 도구들은 타겟 고객에 맞는 맞춤형 콘텐츠 제공에 한계
- 개인화된 콘텐츠로 고객 참여 극대화

## 글로벌 확장

- 다국어 지원 부족과 대규모 제작 요구 대응의 어려움
- 확장 가능한 플랫폼으로 글로벌 시장 경쟁력 강화

## 프로젝트 목표

### 콘텐츠 제작 자동화

- AI 기반 시스템을 활용해 사용자의 입력만으로 간단하고 빠르게 콘텐츠를 제작하고, 제작 시간을 단축하여 효율적인 플랫폼 구축

### 맞춤형 콘텐츠 제공

- 고객 데이터를 분석하여 타겟 고객에 최적화된 개인화 콘텐츠를 생성하고, 소비자 참여와 마케팅 효과를 극대화

### 글로벌 확장성 확보

- 다국어 지원 및 확장 가능한 아키텍처를 통해 대규모 콘텐츠 제작 요구를 처리하고, 글로벌 사용자 대상의 마케팅을 지원

### 안정적 인프라 구축

- 클라우드 기반으로 안정적이고 신뢰성 높은 시스템을 운영하며, 장기적인 확장성과 효율성을 갖춘 인프라 제공

## 프로젝트 기대효과

### 비용과 시간 절감

- 콘텐츠 제작 과정의 자동화를 통해 제작 비용을 절감하고, 소규모 비즈니스도 효율적으로 고품질 콘텐츠 제작 가능

### 고객 참여 증대

- 개인화된 맞춤형 콘텐츠 제공으로 소비자와의 연결을 강화하고, 타겟 마케팅 효과 극대화를 통한 전환율 상승

### 글로벌 시장 경쟁력 확보

- 다국어 콘텐츠와 글로벌 확장성을 통해 새로운 시장 진출을 용이하게 하고, 대규모 제작 요구에도 유연하게 대응

### 안정적이고 지속 가능한 운영

- AWS 클라우드 기반으로 안정적이고 확장 가능한 서비스 운영을 지원하며, 장기적인 비용 효율성과 운영 안정성 강화

## | 프로젝트 설계 및 계획

02-1 협업 도구 및 소프트웨어

02-2 주요 AWS 서비스

02-3 추진 일정 및 계획

## 협업 도구 및 소프트웨어



Github

코드 버전 관리와 코드  
리뷰를 통해 팀원 간 협업을  
지원하는 도구



VSCode

코드 작성과 디버깅, Git  
연동을 지원하는 통합  
개발 환경



Slack

실시간 커뮤니케이션과 파일  
공유를 통해 팀원 간 협업을  
지원하는 도구



Notion

문서 작성, 일정 관리,  
프로젝트 정보를 체계적으로  
공유할 수 있는 협업 도구

## 주요 서비스



ECS

컨테이너화된  
애플리케이션을 배포 및  
관리할 수 있는 서비스



Terraform

인프라를 코드로 관리하며  
AWS 리소스를  
프로비저닝할 수 있는 도구



Rekognition

이미지와 동영상을 분석하여  
객체, 텍스트, 얼굴 등을  
인식할 수 있는 서비스



Bedrock

생성형 AI를 활용해 마케팅  
스크립트를 생성할 수 있는  
서비스

## 추진 일정 및 계획

번호	세부 추진 내용	추 진 일 정 (일)							
		12/2~12/6	12/7~12/11	12/12~12/14	12/15~12/19	12/20~12/24	12/25~12/29	12/30~12/31	1/1~1/3
1	프로젝트 비전 및 목표 수립								
2	요구사항 정의 및 기술 검토								
3	네트워크 아키텍처 및 보안 프레임워크 설계								
4	클라우드 인프라 초기 구성								
5	AI 및 비디오 생성 워크플로우 설계								
6	애플리케이션 개발 (Frontend & Backend)								
7	지속적 통합 및 배포 파이프라인 구축								
8	서비스 테스트 및 배포 최적화								
9	최종 발표 준비								

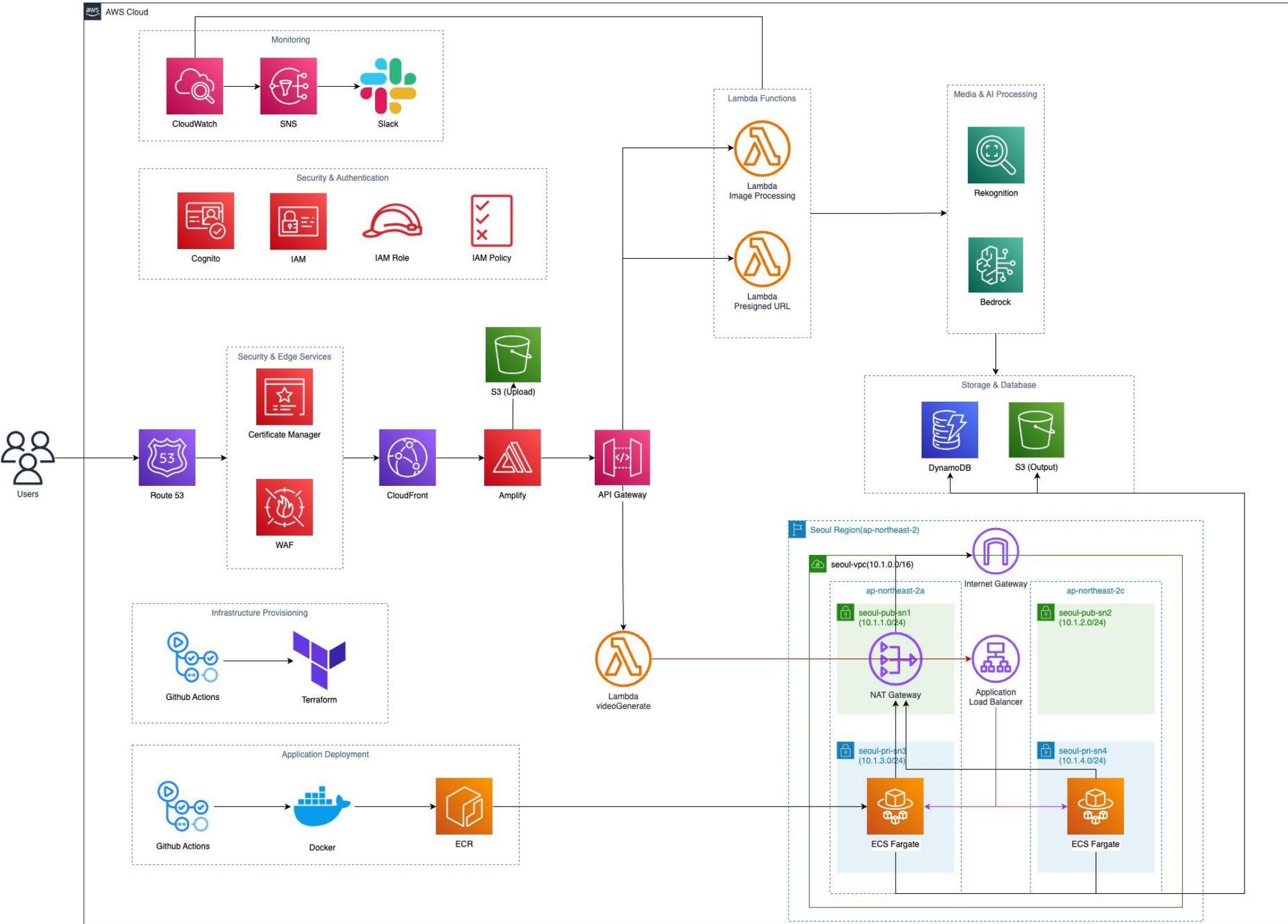
## | 프로젝트 구현

03-1 아키텍처 구성도

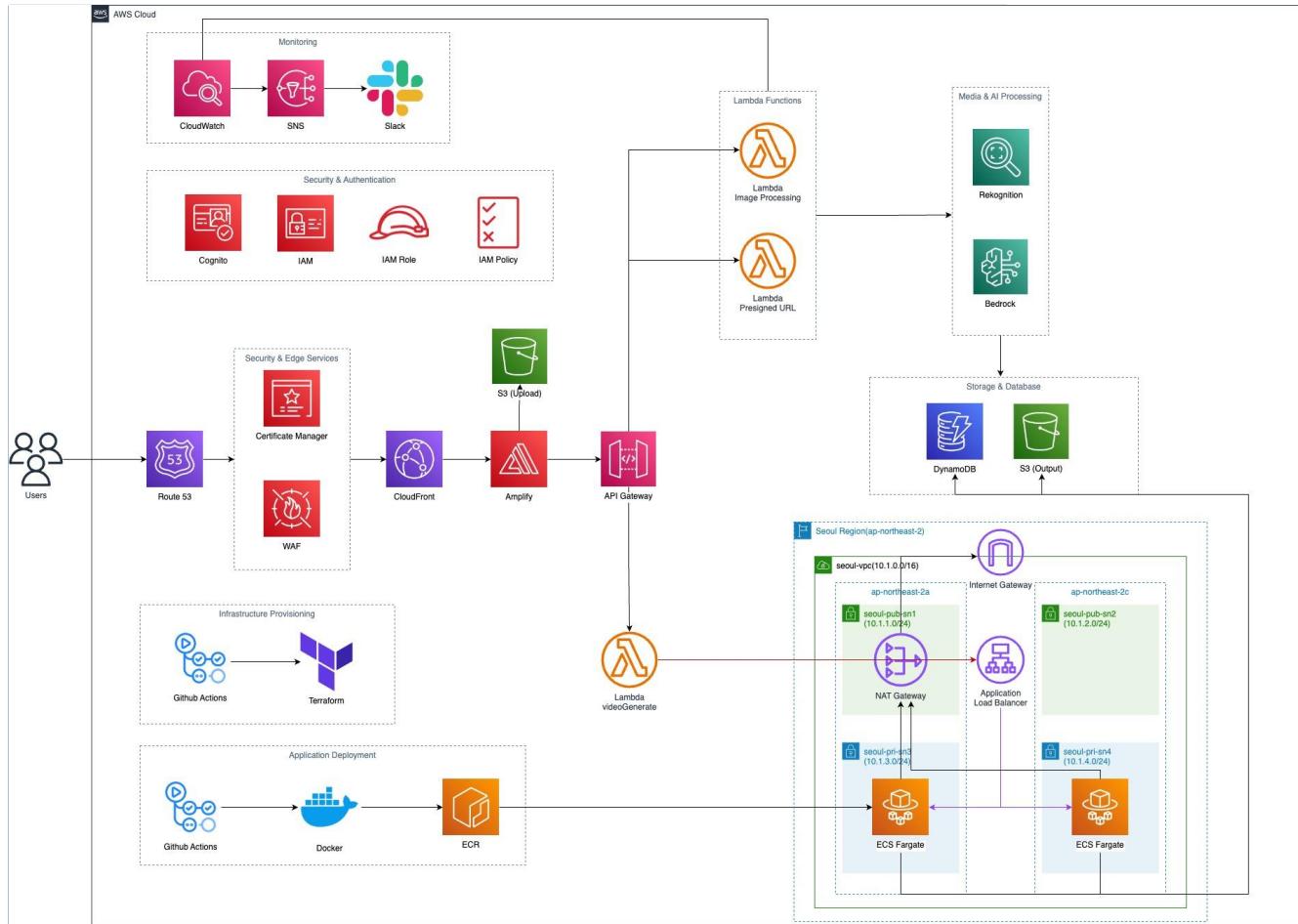
03-2 구현 과정



# 아키텍처 구성도



# 인프라 구축(Terraform)



```

infra.tf
C: > Users > jaehy > terraform2 > infra.tf
148 resource "aws_security_group" "alb_sg" {
149
150 # 10. ALB Target Group 생성
151 resource "aws_lb_target_group" "app_tg" {
152   name      = "app-target-group" # 타겟 그룹 이름
153   port      = 80                 # 대상 포트
154   protocol  = "HTTP"            # 대상 프로토콜
155   vpc_id    = aws_vpc.main.id  # VPC
156
157   health_check {
158     path          = "/"          # 헬스 체크 경로
159     interval     = 30           # 헬스 체크 간격
160     timeout      = 5            # 타임아웃
161     healthy_threshold = 3       # 헬스 체크 성공 임계값
162     unhealthy_threshold = 2     # 헬스 체크 실패 임계값
163   }
164
165   tags = {
166     Name = "app-tg" # 타겟 그룹 이름 태그
167   }
168
169
170 # 11. ALB(Application Load Balancer) 생성
171 resource "aws_lb" "app" {
172   name      = "app-lb"          # ALB 이름
173   internal  = false             # 외부에 노출
174   load_balancer_type = "application" # ALB 타입
175   security_groups = [aws_security_group.alb_sg.id] # 보안 그룹
176   subnets    = [aws_subnet.public_a.id, aws_subnet.public_b.id] # 피블릭 서브넷 두 개 연결
177
178   tags = {
179     Name = "application-load-balancer" # ALB 이름 태그
180   }
181
182
183 # ACM 인증서 생성
184 resource "aws_acm_certificate" "shopreel_cert" {
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
}

```

인프라를 Terraform으로 코드화하여 배포 및 관리 자동화

# 인프라 구축(Terraform)

```
13     |     Name = "shopreel-vpc" # VPC 이름 태그
14   }
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
C:\ cmd + ⌂ ⌂ ...
C:\Users\jaehy\terraform2>git add .

C:\Users\jaehy\terraform2>git commit -m "Add Terraform VPC setup code"
[feature/infra/vpc-setup 36938b7] Add Terraform VPC setup code
 2 files changed, 325 insertions(+)
 create mode 100644 infra.tf
 create mode 100644 infraCICD.yaml

C:\Users\jaehy\terraform2>git push origin feature/infra/vpc-setup
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 22 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.22 KiB | 1.61 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lucyi0920/ShopReelAI.git
  c8a2e0f..36938b7  feature/infra/vpc-setup -> feature/infra/vpc-setup

C:\Users\jaehy\terraform2>
```

## 체계적인 관리를 위해 Github에 Terraform 커밋 및 푸쉬

# 인프라 구축(Terraform)

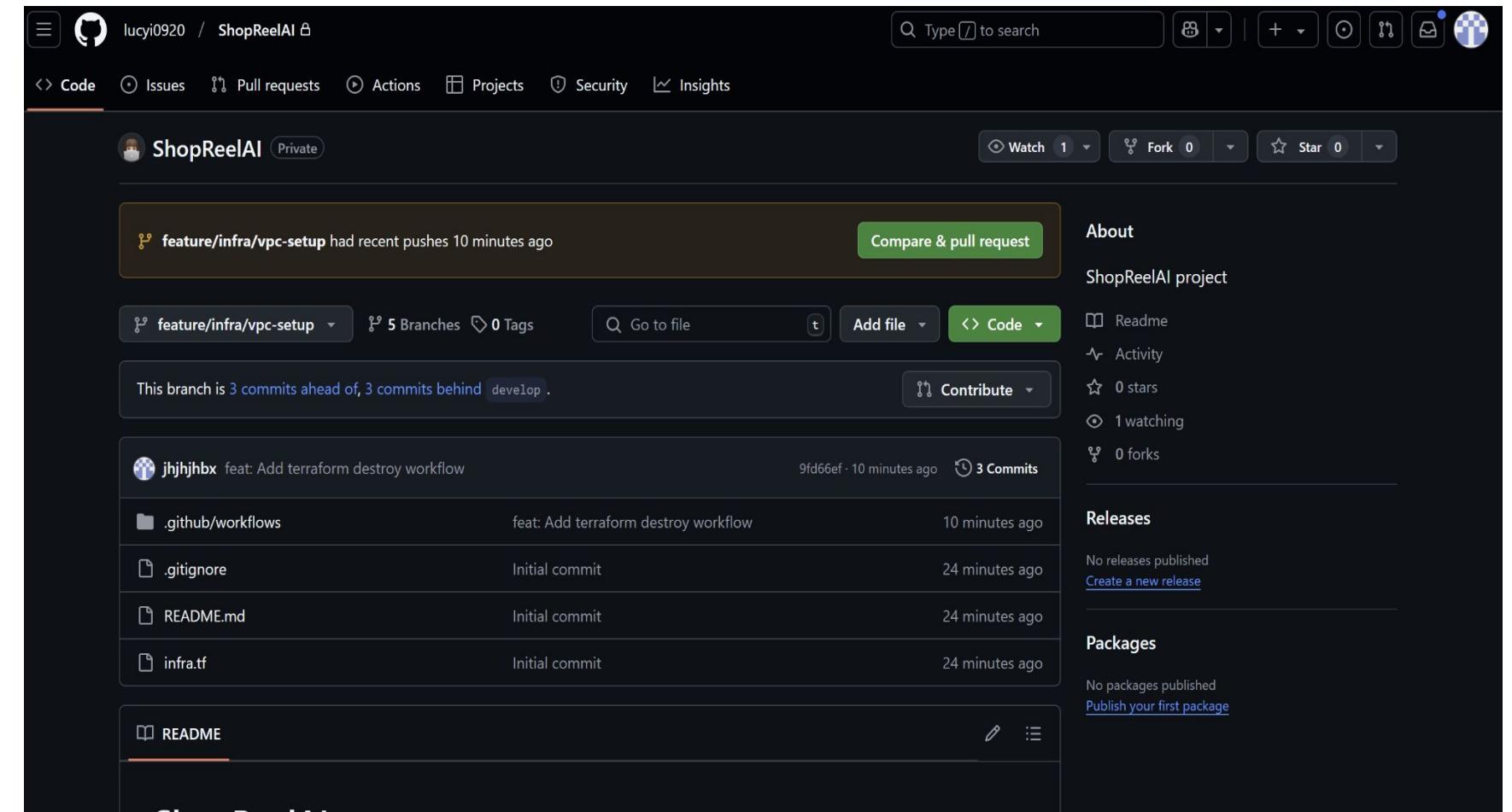
The screenshot shows the GitHub 'Branches' page for a repository. It includes sections for 'Default', 'Your branches', and 'Active branches'. The 'Default' section shows the 'develop' branch updated 'last month' and marked as 'Default'. The 'Your branches' section shows the 'feature/infra/vpc-setup' branch updated '4 days ago'. The 'Active branches' section shows three branches: 'feature/infra/vpc-setup', 'feature/backend/lambda-image-process', and 'feature/infra/ecs-setup', all updated 'last month'. A 'New branch' button is located at the top right.

The screenshot shows the GitHub repository page for 'ShopReelAI'. It features tabs for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights'. The 'Code' tab is active, showing a commit from 'jhjhjhbx' adding Terraform VPC setup code. Other files shown include 'README.md', 'infra.tf', and 'infraCIDC.yaml'. The repository has 5 branches and 0 tags. The 'About' section indicates it's a private project with 1 watch, 0 forks, and 0 stars. The 'Releases' section shows no releases published, and the 'Packages' section shows no packages published.

Github 브랜치를 생성하고 Terraform 코드 업로드

# 자동화 및 CI/CD

```
infraCI_CD.yaml
C: > Users > jaehy > terraform2 > .github > workflows > infraCI_CD.yaml
  5   on:
  6     push:
  7       branches:
  8
  9
 10  jobs:
 11    terraform:
 12      name: Deploy Terraform Infrastructure
 13      runs-on: ubuntu-latest
 14
 15    steps:
 16      # 1. GitHub 리포지토리 체크아웃
 17      - name: Checkout repository
 18        uses: actions/checkout@v3
 19
 20      # 2. AWS 인증 설정 (GitHub Secrets 사용)
 21      - name: Configure AWS credentials
 22        uses: aws-actions/configure-aws-credentials@v3
 23        with:
 24          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
 25          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
 26          aws-region: ap-northeast-2
 27
 28      # 3. Terraform 설치
 29      - name: Setup Terraform
 30        uses: hashicorp/setup-terraform@v2
 31        with:
 32          terraform_version: 1.10.0 # 원하는 Terraform 버전
 33
 34      # 4. Terraform 초기화
 35      - name: Terraform Init
 36        run: terraform init
 37
 38      # 5. Terraform Plan (변경 사항 시뮬레이션)
 39      - name: Terraform Plan
 40        run: terraform plan
 41
 42      # 6. Terraform Apply (실제 리소스 배포)
 43      - name: Terraform Apply
 44        run: terraform apply --auto-approve
```



CI/CD 파이프라인 설정을 위한 Github Actions 워크플로우 파일 생성

# 자동화 및 CI/CD

The screenshot shows a terminal window in Visual Studio Code running on Microsoft Windows. The terminal output is as follows:

```
C:\Users\jaehy\terraform2>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.82.2

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

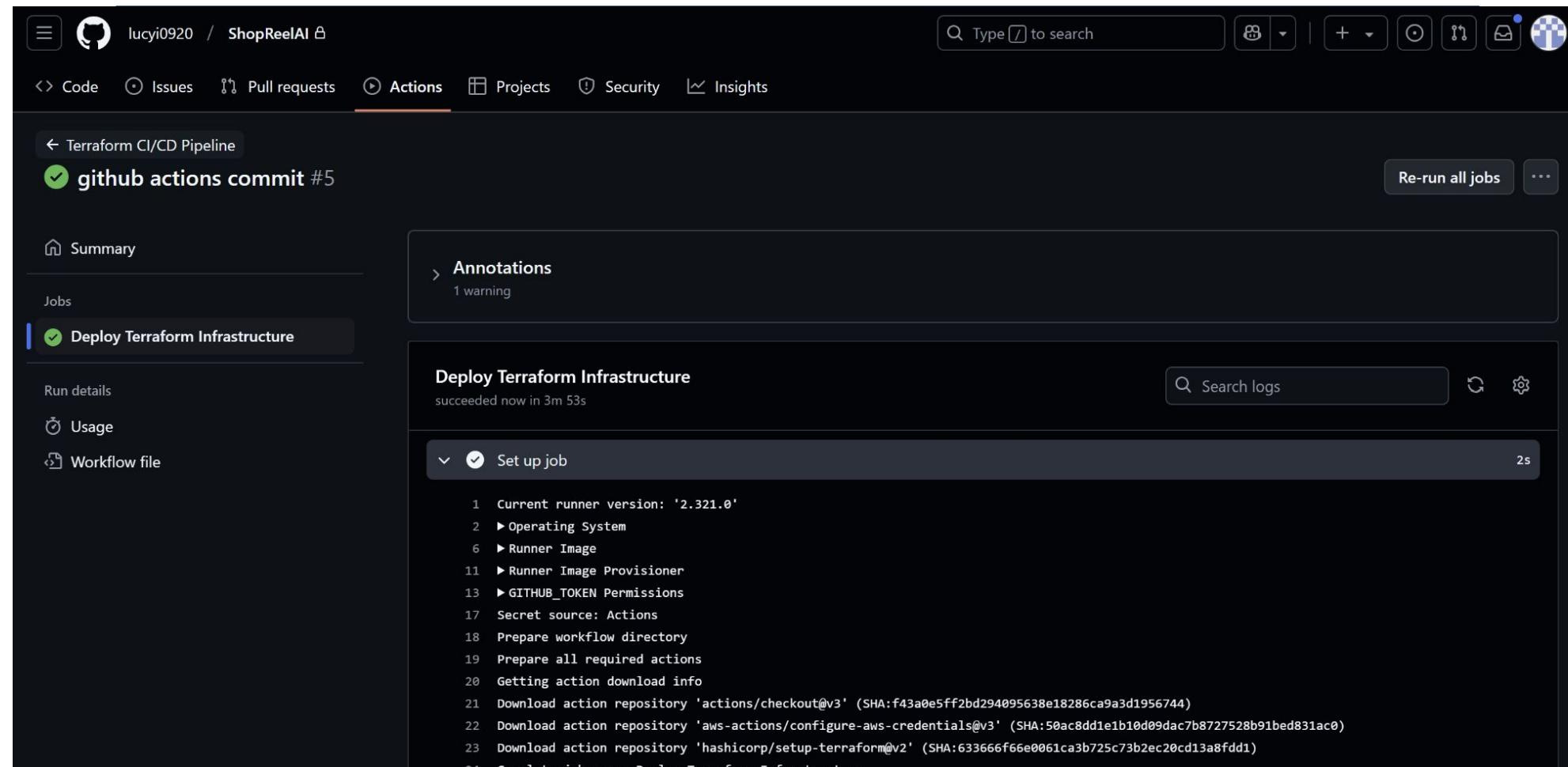
C:\Users\jaehy\terraform2>git add .

C:\Users\jaehy\terraform2>git commit -m "github actions commit"
[feature/infra/vpc-setup 01adbc] github actions commit
 1 file changed, 13 insertions(+), 12 deletions(-)

C:\Users\jaehy\terraform2>git push origin feature/infra/vpc-setup
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 22 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 831 bytes | 277.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lucyi0920/ShopReelAI.git
 9fd66ef..01adbc feature/infra/vpc-setup -> feature/infra/vpc-setup
```

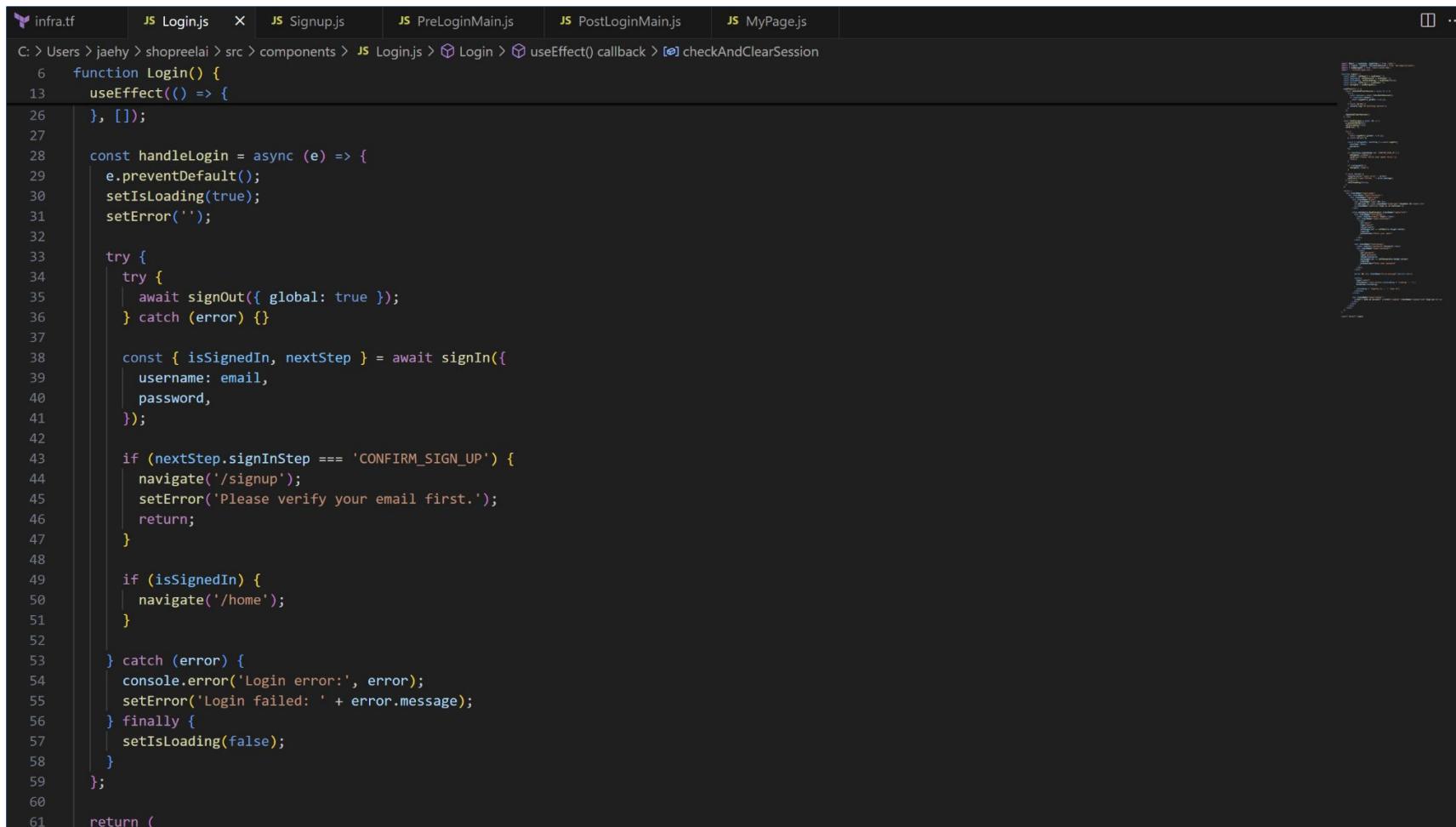
Terraform과 GitHub Actions를 활용하여 CI/CD 프로세스를 자동화하고 버전 관리를 체계적으로 수행

# 자동화 및 CI/CD



GitHub Actions를 통해 인프라 배포 프로세스를 자동화하여 신속하고 효율적인 관리 구현

# 프론트엔드(Login.js)

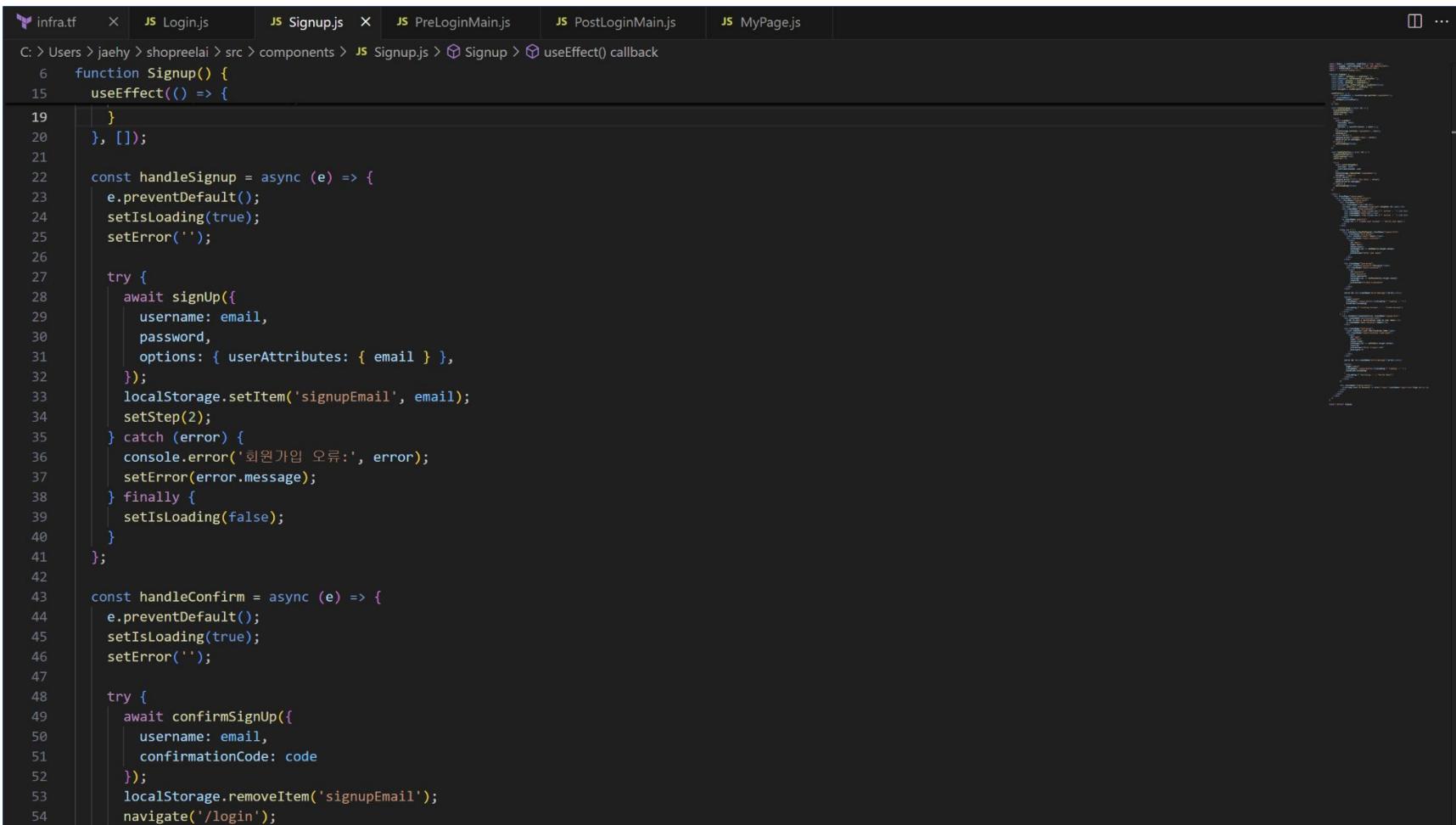


```
infra.tf JS Login.js X JS Signup.js JS PreLoginMain.js JS PostLoginMain.js JS MyPage.js ...
C: > Users > jaehee > shopreelai > src > components > JS Login.js > Login > useEffect() callback > checkAndClearSession
6   function Login() {
13     useEffect(() => {
26       ], [I]);
27
28     const handleLogin = async (e) => {
29       e.preventDefault();
30       setIsLoading(true);
31       setError('');
32
33       try {
34         try {
35           await signOut({ global: true });
36         } catch (error) {}
37
38         const { isSignedIn, nextStep } = await signIn({
39           username: email,
40           password,
41         });
42
43         if (nextStep.signInStep === 'CONFIRM_SIGN_UP') {
44           navigate('/signup');
45           setError('Please verify your email first.');
46           return;
47         }
48
49         if (isSignedIn) {
50           navigate('/home');
51         }
52
53       } catch (error) {
54         console.error('Login error:', error);
55         setError('Login failed: ' + error.message);
56       } finally {
57         setIsLoading(false);
58       }
59     };
60
61     return (

```

사용자가 입력한 이메일과 비밀번호로 AWS Amplify API를 호출하여 로그인 성공 시 홈 화면으로 이동, 이메일 인증이 필요한 경우 회원가입 화면으로 이동

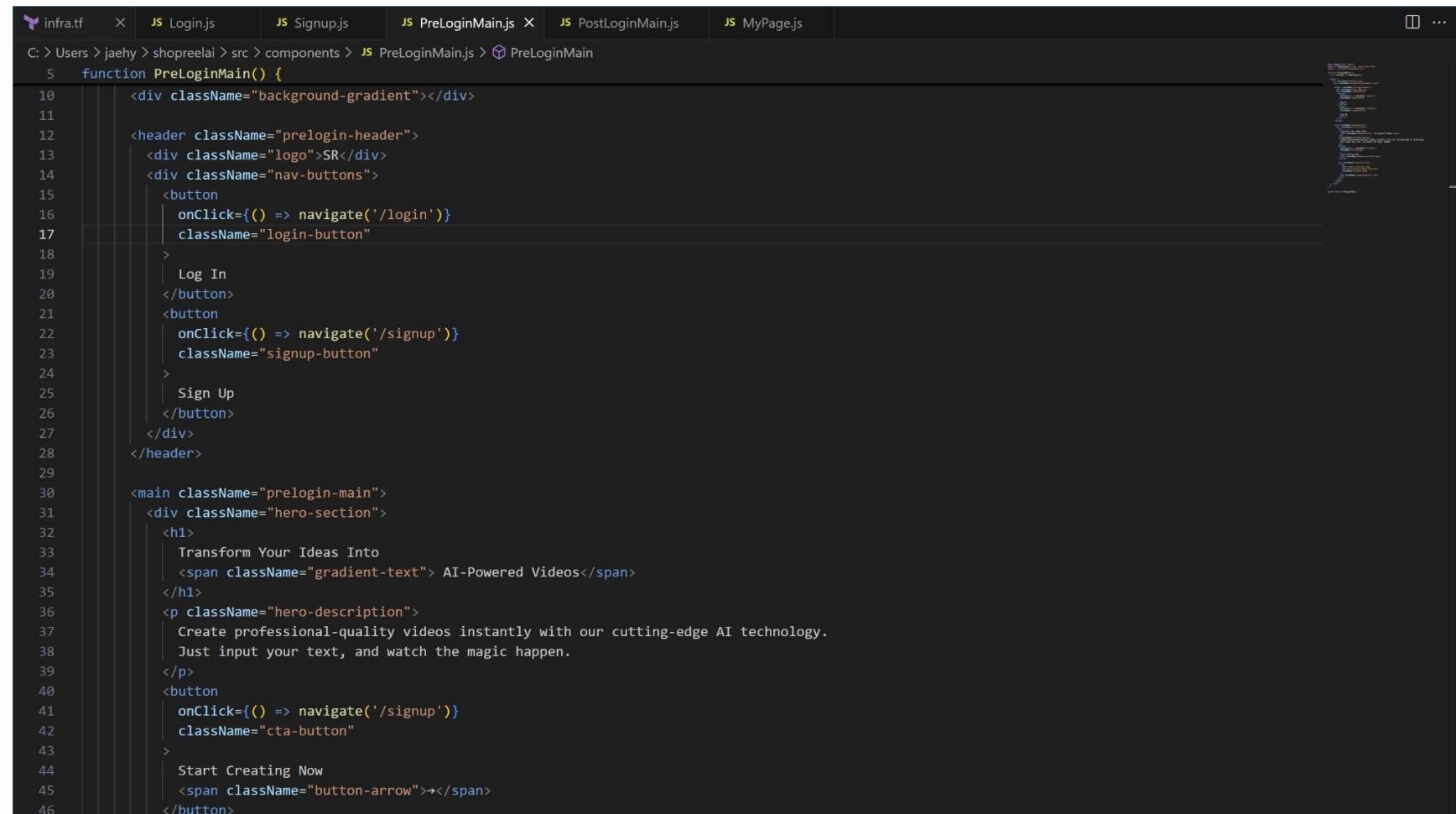
# 프론트엔드(Signup.js)



```
C: > Users > jaehy > shopreelai > src > components > JS Signup.js > Signup > useEffect() callback
6  function Signup() {
15    useEffect(() => {
19      }
20    }, []);
21
22    const handleSignup = async (e) => {
23      e.preventDefault();
24      setIsLoading(true);
25      setError('');
26
27      try {
28        await signUp({
29          username: email,
30          password,
31          options: { userAttributes: { email } },
32        });
33        localStorage.setItem('signupEmail', email);
34        setStep(2);
35      } catch (error) {
36        console.error('회원가입 오류:', error);
37        setError(error.message);
38      } finally {
39        setIsLoading(false);
40      }
41    };
42
43    const handleConfirm = async (e) => {
44      e.preventDefault();
45      setIsLoading(true);
46      setError('');
47
48      try {
49        await confirmSignUp({
50          username: email,
51          confirmationCode: code
52        });
53        localStorage.removeItem('signupEmail');
54        navigate('/login');
55      } catch (error) {
56        console.error('회원가입 오류:', error);
57        setError(error.message);
58      } finally {
59        setIsLoading(false);
60      }
61    };
62
63  };
64
65  export default Signup;
```

사용자의 계정을 생성하고 이메일 인증 코드를 확인하여 계정을 활성화

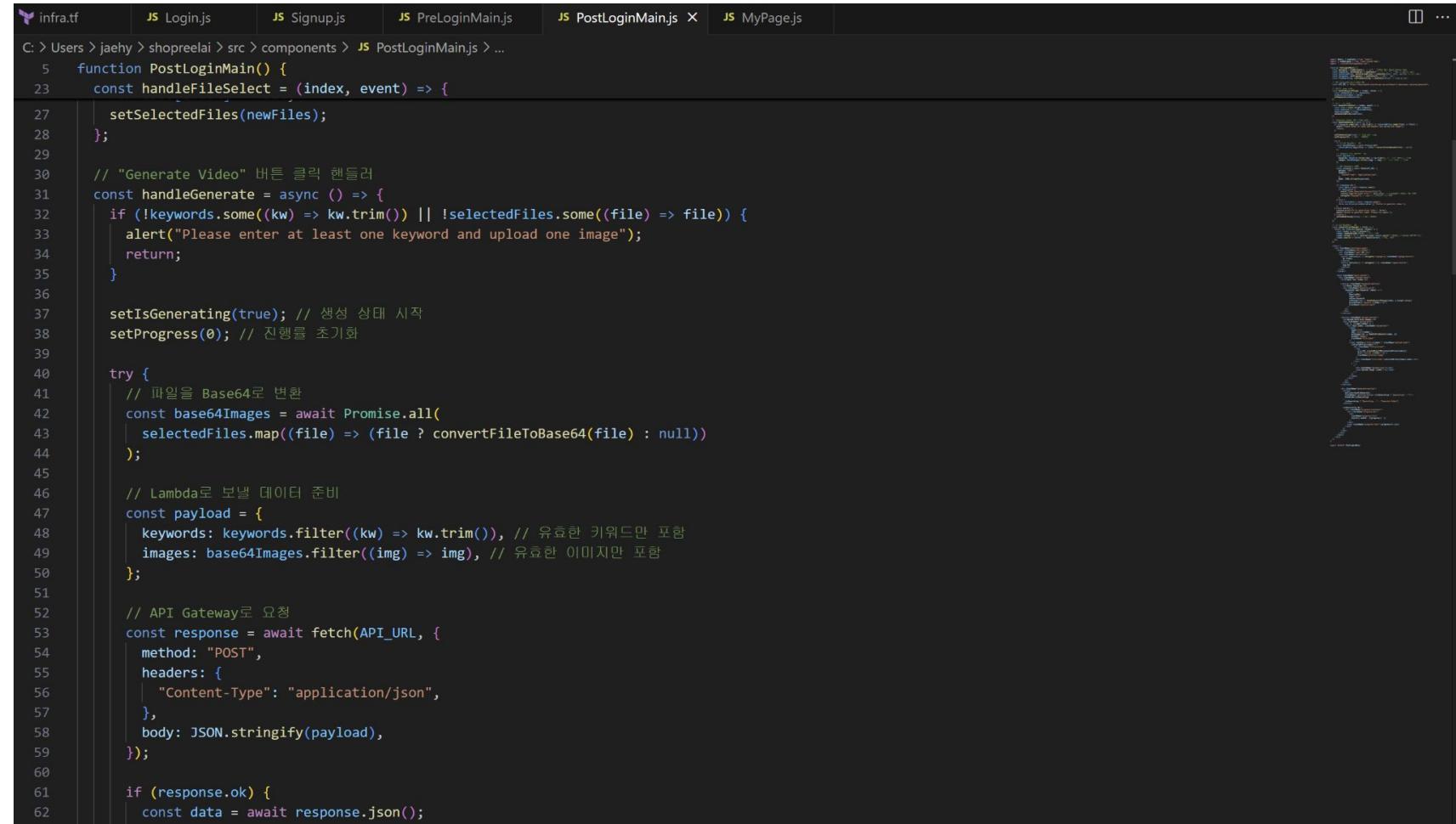
# 프론트엔드(PreLoginMain.js)



```
C: > Users > jaehy > shopreelai > src > components > JS PreLoginMain.js X JS PostLoginMain.js JS MyPage.js
function PreLoginMain() {
  <div className="background-gradient"></div>
  <header className="prelogin-header">
    <div className="logo">SR</div>
    <div className="nav-buttons">
      <button
        onClick={() => navigate('/login')}
        className="login-button"
      >
        Log In
      </button>
      <button
        onClick={() => navigate('/signup')}
        className="signup-button"
      >
        Sign Up
      </button>
    </div>
  </header>
  <main className="prelogin-main">
    <div className="hero-section">
      <h1>
        Transform Your Ideas Into
        <span className="gradient-text"> AI-Powered Videos </span>
      </h1>
      <p className="hero-description">
        Create professional-quality videos instantly with our cutting-edge AI technology.
        Just input your text, and watch the magic happen.
      </p>
      <button
        onClick={() => navigate('/signup')}
        className="cta-button"
      >
        Start Creating Now
        <span className="button-arrow" >></span>
      </button>
    </div>
  </main>
}
```

사용자를 회원가입 또는 로그인 페이지로 안내

# 프론트엔드(PostLoginMain.js)

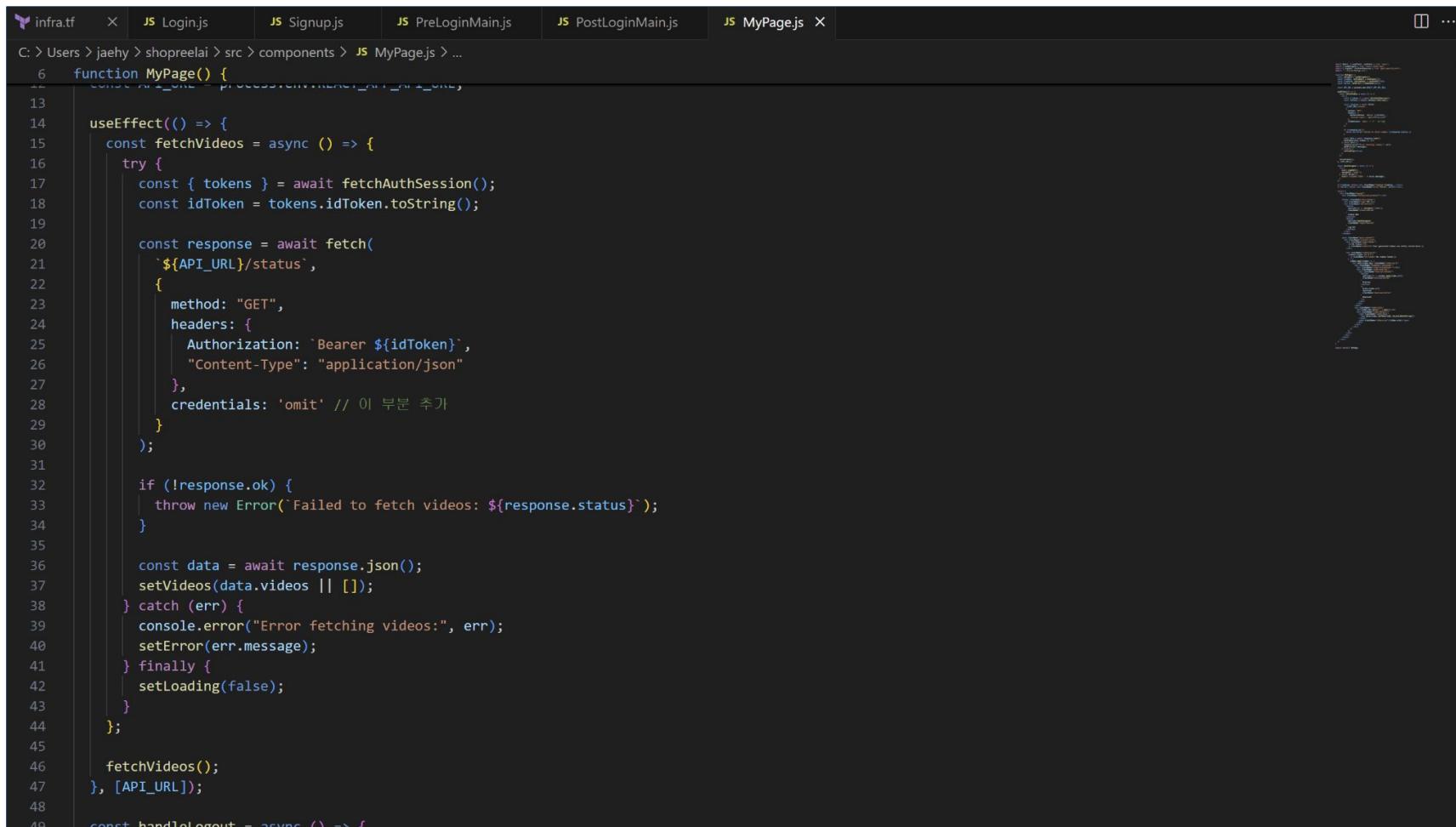


The screenshot shows a code editor window with multiple tabs at the top: infra.tf, JS Login.js, JS Signup.js, JS PreLoginMain.js, JS PostLoginMain.js (which is the active tab), JS MyPage.js, and a ... tab. The code in the PostLoginMain.js tab is as follows:

```
C: > Users > jaehy > shopreelai > src > components > JS PostLoginMain.js > ...
5  function PostLoginMain() {
23    const handleFileSelect = (index, event) => {
27      setSelectedFiles(newFiles);
28    };
29
30    // "Generate Video" 버튼 클릭 핸들러
31    const handleGenerate = async () => {
32      if (!keywords.some((kw) => kw.trim()) || !selectedFiles.some((file) => file)) {
33        alert("Please enter at least one keyword and upload one image");
34        return;
35      }
36
37      setIsGenerating(true); // 생성 상태 시작
38      setProgress(0); // 진행률 초기화
39
40      try {
41        // 파일을 Base64로 변환
42        const base64Images = await Promise.all(
43          selectedFiles.map((file) => (file ? convertFileToBase64(file) : null))
44        );
45
46        // Lambda로 보낼 데이터 준비
47        const payload = {
48          keywords: keywords.filter((kw) => kw.trim()), // 유효한 키워드만 포함
49          images: base64Images.filter((img) => img), // 유효한 이미지만 포함
50        };
51
52        // API Gateway로 요청
53        const response = await fetch(API_URL, {
54          method: "POST",
55          headers: {
56            "Content-Type": "application/json",
57          },
58          body: JSON.stringify(payload),
59        });
60
61        if (response.ok) {
62          const data = await response.json();
63        }
64      } catch (error) {
65        console.error(error);
66      }
67    };
68
69    // 파일 선택 핸들러
70    const handleFileSelectHandler = (event) => {
71      const files = event.target.files;
72      setSelectedFiles(files);
73    };
74
75    // 파일 선택 버튼
76    const fileSelectButton = document.createElement("button");
77    fileSelectButton.textContent = "Select File";
78    fileSelectButton.addEventListener("click", handleFileSelectHandler);
79
80    // Generate Video 버튼
81    const generateButton = document.createElement("button");
82    generateButton.textContent = "Generate Video";
83    generateButton.addEventListener("click", handleGenerate);
84
85    // Container
86    const container = document.createElement("div");
87    container.appendChild(fileSelectButton);
88    container.appendChild(generateButton);
89
90    // Render
91    render(container);
92  };
93
94  // Render
95  const render = (container) => {
96    ReactDOM.render(
97      <PostLoginMain />,
98      container
99    );
100  };
101
102  render();
103
```

사용자가 업로드한 이미지와 입력한 키워드를 바탕으로 AWS API Gateway를 통해 Presigned URL을 생성하고 파일 업로드 후 영상 생성 요청

# 프론트엔드(MyPage.js)



A screenshot of a code editor showing the file `MyPage.js`. The code is a functional component that uses the `useEffect` hook to fetch videos from an API. It includes error handling and loading state management.

```
function MyPage() {
  useEffect(() => {
    const fetchVideos = async () => {
      try {
        const { tokens } = await fetchAuthSession();
        const idToken = tokens.idToken.toString();

        const response = await fetch(`${API_URL}/status`,
        {
          method: "GET",
          headers: {
            Authorization: `Bearer ${idToken}`,
            "Content-Type": "application/json"
          },
          credentials: 'omit' // 이 부분 추가
        });
      if (!response.ok) {
        throw new Error(`Failed to fetch videos: ${response.status}`);
      }
      const data = await response.json();
      setVideos(data.videos || []);
    } catch (err) {
      console.error("Error fetching videos:", err);
      setError(err.message);
    } finally {
      setLoading(false);
    };
  };
  fetchVideos();
}, [API_URL]);
const handleLogout = async () => {
  await signOut();
  window.location.href = "/";
}
```

AWS API Gateway를 통해 서버에서 사용자 계정에 연결되어있는 생성된 영상을 불러오고 오류 발생시 사용자에게 알림

# Amplify

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS node + - ×

● soomin@soomin-uiMacBookPro shopreelai % amplify init
⚠️ For new projects, we recommend starting with AWS Amplify Gen 2, our new code-first developer experience. Get started at https://docs.amplify.aws/react/start/quickstart
/
✓ Do you want to continue with Amplify Gen 1? (y/N) · yes
✓ Why would you like to use Amplify Gen 1? · Prefer not to answer
Note: It is recommended to run this command from the root of your app directory
? Enter a name for the project shopreelai
The following configuration will be applied:

Project information
| Name: shopreelai
| Environment: dev
| Default editor: Visual Studio Code
| App type: javascript
| Javascript framework: react
| Source Directory Path: src
| Distribution Directory Path: build
| Build Command: npm run-script build
| Start Command: npm run-script start

? Initialize the project with the above configuration? Yes
Using default provider awscloudformation
? Select the authentication method you want to use: AWS access keys
? accessKeyId: *****
? secretAccessKey: *****
? region: ap-northeast-2
Adding backend environment dev to AWS Amplify app: disbmtjzvhi1g

Deployment completed.
Deploying root stack shopreelai [ ===== ] 1/4
  amplify-shopreelai-dev-4a632  AWS::CloudFormation::Stack    CREATE_IN_PROGRESS   Sat Dec 28 2024 18:10:07...
    AuthRole                  AWS::IAM::Role             CREATE_IN_PROGRESS   Sat Dec 28 2024 18:10:09...
    DeploymentBucket          AWS::S3::Bucket           CREATE_COMPLETE      Sat Dec 28 2024 18:10:23...
    UnauthRole                AWS::IAM::Role             CREATE_IN_PROGRESS   Sat Dec 28 2024 18:10:09...

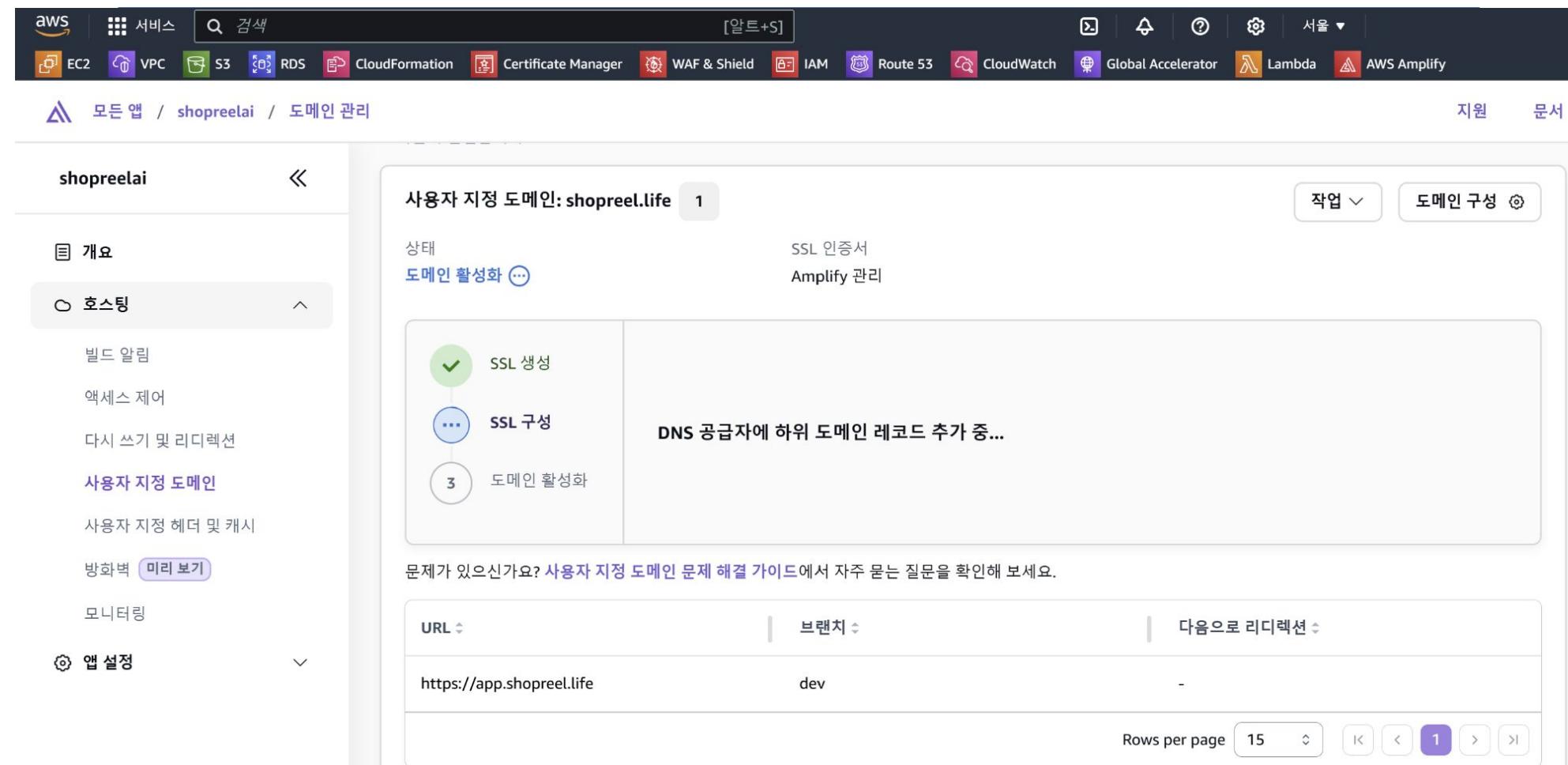
✓ Help improve Amplify CLI by sharing non-sensitive project configurations on failures (y/N) · no

  You can always opt-in by running "amplify configure --share-project-config-on"
Deployment state saved successfully.
✓ Initialized provider successfully.
✓ Initialized your environment successfully.
✓ Your project has been successfully initialized and connected to the cloud!
Some next steps:

"amplify status" will show you what you've added already and if it's locally configured or deployed
"amplify add <category>" will allow you to add features like user login or a backend API
"amplify push" will build all your local backend resources and provision it in the cloud
"amplify console" to open the Amplify Console and view your project status
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud
```

프론트엔드와 백엔드 통합 배포를 자동화하고 손쉽게 관리 가능

# Amplify



도메인과 인증서를 손쉽게 설정하여 사용자 친화적인 URL과 보안을 확보

# Cognito

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ×

"amplify add <category>" will allow you to add features like user login or a backend API  
"amplify push" will build all your local backend resources and provision it in the cloud  
"amplify console" to open the Amplify Console and view your project status  
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud

**Pro tip:**  
Try "amplify add api" to create a backend API and then "amplify push" to deploy everything

- soomin@soomin-iMacBookPro shopreelai % amplify add auth  
Using service: Cognito, provided by: awscloudformation

The current configured provider is Amazon Cognito.

Do you want to use the default authentication and security configuration? Default configuration  
Warning: you will not be able to edit these selections.  
How do you want users to be able to sign in? Email  
Do you want to configure advanced settings? No, I am done.  
 Successfully added auth resource shopreelai0727bb8b locally

Some next steps:  
"amplify push" will build all your local backend resources and provision it in the cloud  
"amplify publish" will build all your local backend and frontend resources (if you have hosting category added) and provision it in the cloud

- soomin@soomin-iMacBookPro shopreelai % amplify push  
 Successfully pulled backend environment dev from the cloud.

Current Environment: dev

Category	Resource name	Operation	Provider plugin
Auth	shopreelai0727bb8b	Create	awscloudformation

Are you sure you want to continue? (Y/n) · yes

Deployment completed.

```
Deploying root stack shopreelai [ ===== ] 1/2
  amplify-shopreelai-dev-4a632 AWS::CloudFormation::Stack UPDATE_IN_PROGRESS
  authshopreelai0727bb8b AWS::CloudFormation::Stack CREATE_COMPLETE
Deployed auth shopreelai0727bb8b [ ===== ] 6/6
  UserPool AWS::Cognito::UserPool CREATE_COMPLETE
  UserPoolClientRole AWS::IAM::Role CREATE_IN_PROGRESS
  UserPoolClient AWS::Cognito::UserPoolClient CREATE_COMPLETE
  UserPoolClientWeb AWS::Cognito::UserPoolClient CREATE_COMPLETE
  IdentityPool AWS::Cognito::IdentityPool CREATE_COMPLETE
  IdentityPoolRoleMap AWS::Cognito::IdentityPoolRoleMap CREATE_COMPLETE
```

Sat Dec 28 2024 18:11:07...  
Sat Dec 28 2024 18:11:46...  
Sat Dec 28 2024 18:11:16...  
Sat Dec 28 2024 18:11:14...  
Sat Dec 28 2024 18:11:18...  
Sat Dec 28 2024 18:11:18...  
Sat Dec 28 2024 18:11:19...  
Sat Dec 28 2024 18:11:22...

Deployment state saved successfully.

The screenshot shows the AWS Amazon Cognito User Pools console. The top navigation bar includes links for EC2, VPC, S3, RDS, CloudFormation, Certificate Manager, WAF & Shield, IAM, Route 53, CloudWatch, Global Accelerator, Lambda, and AWS Amplify. The main page displays a message about Amazon Verified Permissions, followed by a table listing user pools. The table has columns for User Pool Name, User Pool ID, Creation Time, and Last Update Time. One row is selected, showing the name 'shopreelai0727bb8b\_userpool\_0727bb8b-dev', ID 'ap-northeast-2\_paHrxwPRm', creation time '28초 전', and last update time '28초 전'. Action buttons for 'Edit' (pencil), 'Delete' (trash), and 'Create User Pool' (plus) are visible.

Amazon Cognito

Amazon Verified Permissions의 새로운 소식! API 게이트웨이용 Cognito 사용자 그룹 인증  
이제 애플리케이션에 대한 세분화된 권한 부여 서비스인 Amazon Verified Permissions를 사용하여 API의 그룹 인식 권한 부여 정책을 생성할 수 있습니다. [자세히 알아보기](#)

Amazon Verified Permissions로 이동

사용자 풀 (1) 정보

사용자 풀을 보고 구성합니다. 사용자 풀은 연동 및 로컬 사용자 프로필의 디렉토리이며, 사용자에게 인증 옵션을 제공합니다.

사용자 풀 이름	사용자 풀 ID	생성 시간	마지막 업데이트 시간
<a href="#">shopreelai0727bb8b_userpool_0727bb8b-dev</a>	ap-northeast-2_paHrxwPRm	28초 전	28초 전

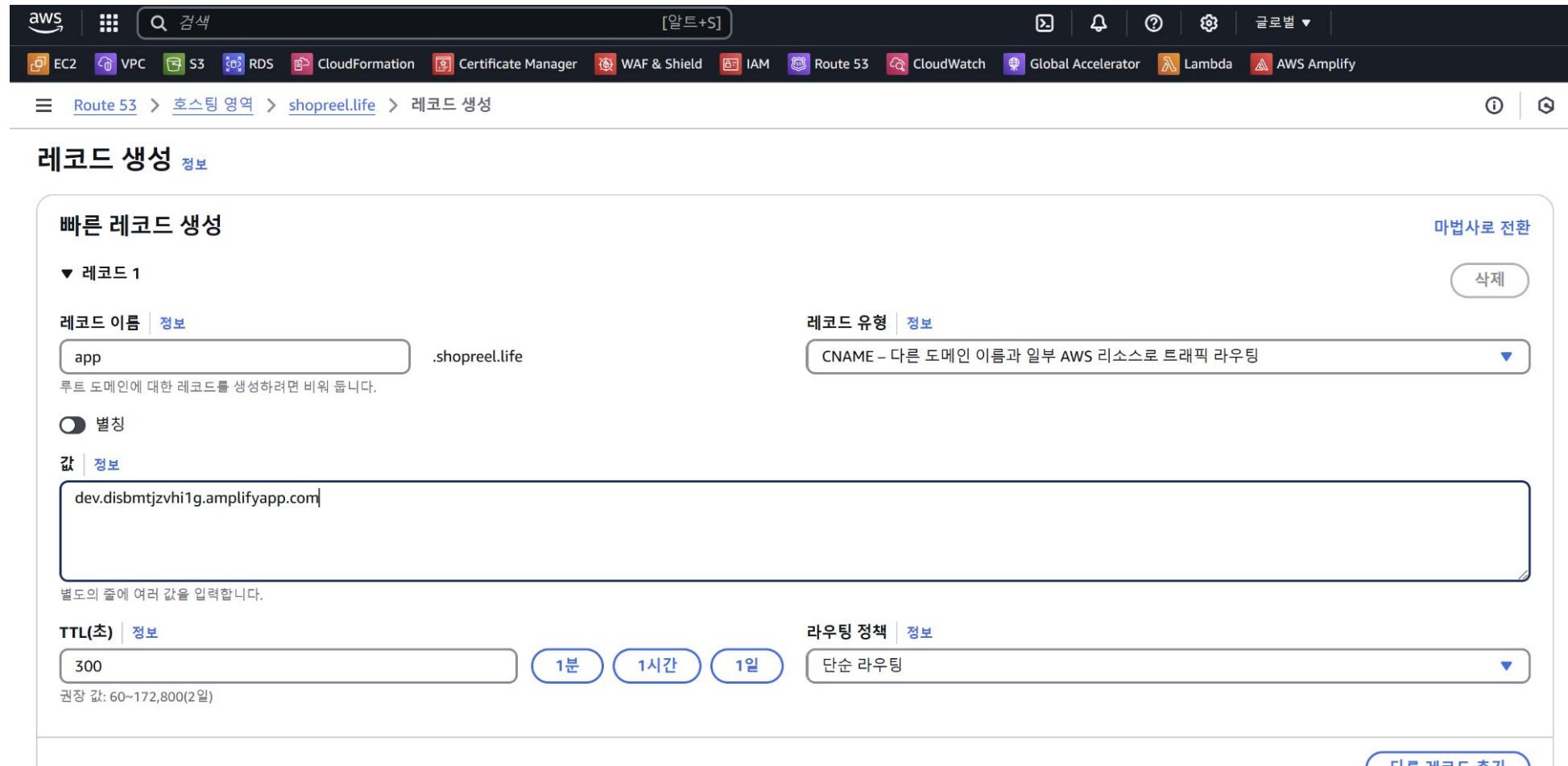
안전하고 확장 가능한 사용자 인증 및 권한 관리 기능을 구현

# Cognito

The screenshot shows the Amazon Cognito console interface. The top navigation bar includes links for EC2, VPC, S3, RDS, CloudFormation, Certificate Manager, WAF & Shield, IAM, Route 53, CloudWatch, Global Accelerator, Lambda, and AWS Amplify. The main navigation on the left lists 'Amazon Cognito', '현재 사용자 풀' (Current User Pool), and sections for '개요' (Overview), '애플리케이션' (Applications), '사용자 관리' (User Management), '인증' (Authentication), and '보안' (Security). The central panel displays two main sections: '사용자 정보' (User Information) and '사용자 가져오기' (Import Users). The '사용자 정보' section shows one user entry: '24c85dac-4051-70e7...' with email 'jaehyunbox@naver.com' and status '확인됨' (Verified). The '사용자 가져오기' section indicates '0' entries and provides a search bar and filter options for '작업 이름', '상태', '가져온 사용자', '건너뛴 사용자', '실패한 사용자', and '생성 시간'.

사용자 인증 및 권한 관리를 중앙에서 손쉽게 처리

# Route 53



사용자 지정 도메인을 Amplify 애플리케이션과 매핑하여 도메인 관리를 간소화

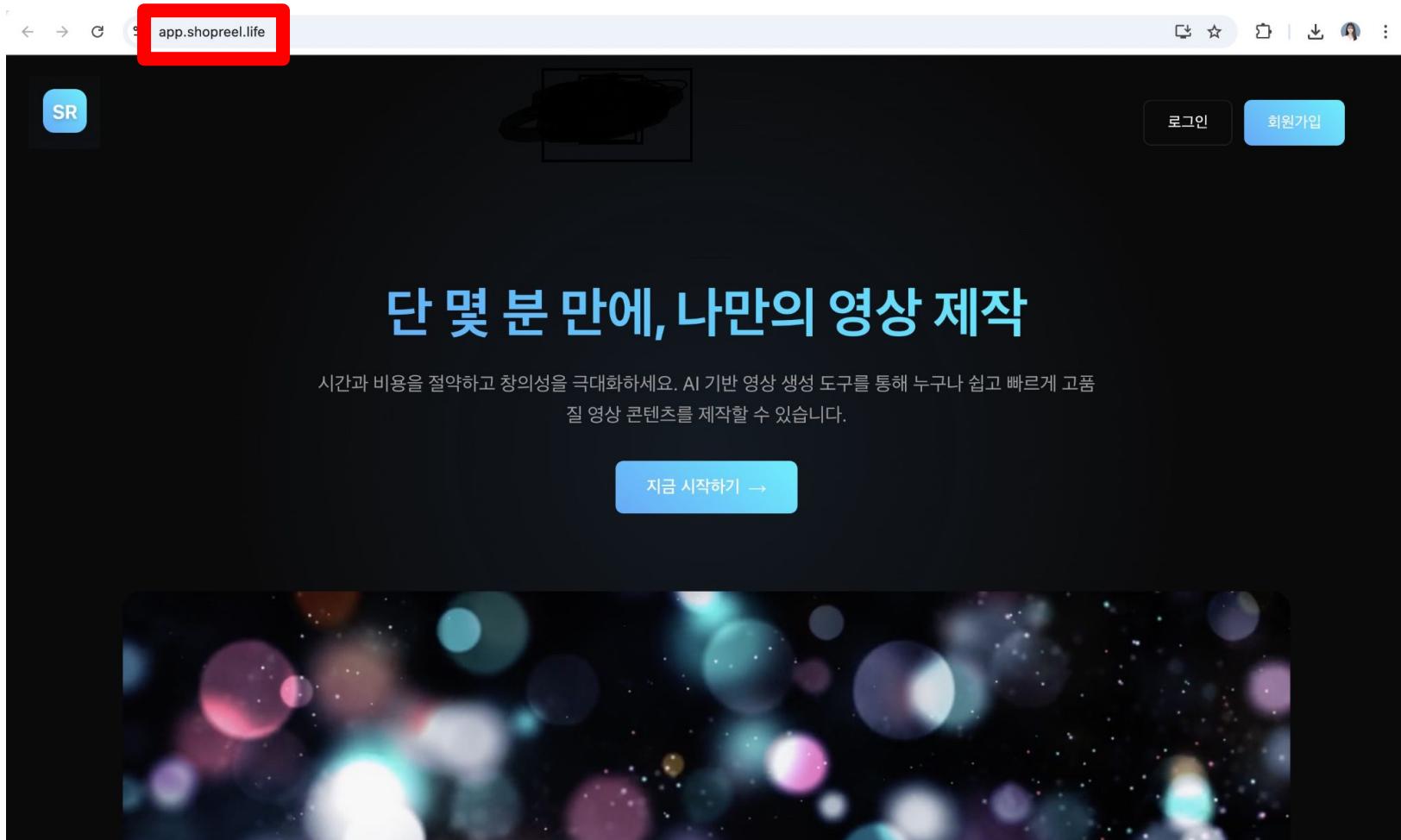
# Route 53

The screenshot shows the AWS Route 53 console interface. The top navigation bar includes links for EC2, VPC, S3, RDS, CloudFormation, Certificate Manager, WAF & Shield, IAM, Route 53, CloudWatch, Global Accelerator, Lambda, and AWS Amplify. The main navigation path is Route 53 > 호스팅 영역 > shopreel.life. Below this, there are tabs for 퍼블릭 (selected), shopreel.life 정보, and three buttons: 영역 삭제, 레코드 테스트, and 쿼리 로깅 구성. A large button labeled 호스팅 영역 편집 is visible. The main content area is titled '▶ 호스팅 영역 세부 정보' and shows a table of DNS records. The table has columns for Record ID, Name, Type, TTL, and Value. It lists three NS records for shopreel.life pointing to various AWS nameservers, one SOA record, and one CNAME record for app.shopreel.life pointing to a specific URL.

Record ID	Name	Type	TTL	Value
ns-1780.awsdns-30.co.uk.	shopreel.life	NS	-	ns-363.awsdns-45.com. ns-1134.awsdns-13.org. ns-838.awsdns-40.net.
ns-1780.awsdns-30.co.uk. a...	shopreel.life	SOA	-	900
https://dev.disbmtjzvhi1g.a...	app.shopreel.life	CNAME	-	300

DNS 관리 및 도메인 레코드 구성을 효율적으로 처리

# Route 53



사용자 지정 도메인 연결하여 서비스 접근성 강화

# CloudFront

**원본**

**Origin domain**  
Choose an AWS origin, or enter your origin's domain name. Learn more [\[링크\]](#)

Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

**Origin path - optional**  
Enter a URL path to append to the origin domain name for origin requests.

**이름**  
이 원본의 이름을 입력합니다.

**원본 액세스** | 정보

**공개**  
버킷은 공개 액세스를 허용해야 합니다.

**원본 액세스 제어 설정(권장)**  
버킷은 CloudFront에 대한 액세스만 제한할 수 있습니다.

**Legacy access identities**  
CloudFront 원본 액세스 ID(OAI)를 사용하여 S3 버킷에 액세스합니다.

**기본 캐시 동작**

**경로 패턴** | 정보  
기본값(\*)

**자동으로 객체 압축** | 정보  
 Yes

**뷰어**

**뷰어 프로토콜 정책**

- HTTP and HTTPS
- Redirect HTTP to HTTPS
- HTTPS only

**허용된 HTTP 방법**

- GET, HEAD
- GET, HEAD, OPTIONS
- GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

**뷰어 액세스 제한**  
뷰어 액세스를 제한하는 경우 뷰어는 CloudFront 서명된 URL 또는 서명된 쿠키를 사용하여 사용자의 콘텐츠에 액세스해야 합니다.

- No
- Yes

Amplify 배포를 전 세계 사용자에게 빠르고 안전하게 제공

# CloudFront

**설정**

**Anycast static IP list - optional** | 정보  
Deliver traffic from a small set of IP addresses  
There are no Anycast static IP lists available

**가격 분류** | 정보  
지불하려는 최고가와 연관된 가격 분류를 선택합니다.  
 모든 엣지 로케이션에서 사용(최고의 성능)  
 북미 및 유럽만 사용  
 북미, 유럽, 아시아, 중동 및 아프리카에서 사용

**대체 도메인 이름(CNAME) - 선택 사항**  
이 배포에서 제공하는 파일에 대해 URL에서 사용하는 사용자 정의 도메인 이름을 추가합니다.

**Custom SSL certificate - optional**  
AWS Certificate Manager의 인증서를 연결합니다. 인증서는 반드시 미국 동부(버지니아 북부) 리전(us-east-1)에 있어야 합니다.

**E3EIIIBTY6XQHCF**

**일반** | 보안 | 원본 | 동작 | 오류 페이지 | 무효화 | 태그 | Logging

**세부 정보**

**배포 도메인 이름**  
 ARN  
 마지막 수정

**설정**

<b>설명</b> -	<b>대체 도메인 이름</b> -	<b>표준 로깅</b> Off
<b>가격 분류</b> 모든 엣지 로케이션에서 사용(최고의 성능)	<b>Cookie logging</b> <input type="button" value="고기"/>	<b>기본 루트 객체</b>
<b>지원되는 HTTP 버전</b> HTTP/2 HTTP/1.1 HTTP/1.0		

사용자 지정 도메인과 SSL 인증서를 사용해 안전하고 빠른 콘텐츠 전송을 구현

# WAF & CloudWatch

Set rule priority

Name	Capacity	Action
AWS-AWSManagedRulesSQLiRuleSet	200	Use rule actions
AWS-AWSManagedRulesKnownBadInputsRuleSet	200	Use rule actions
AWS-AWSManagedRulesAmazonIpReputationList	25	Use rule actions
AWS-AWSManagedRulesCommonRuleSet	700	Use rule actions
AWS-AWSManagedRulesLinuxRuleSet	200	Use rule actions

Step 1: Describe web ACL and associate it to AWS resources  
Step 2: Add rules and rule groups  
Step 3: Set rule priority  
Step 4: Configure metrics  
Step 5: Review and create web ACL

Cancel Previous Next

Web ACLs

Name	Description	ID
seoul-waf	seoul-waf	8ec052f8-4b6b-477d-90bf-d0038ac1027c

Getting started  
Web ACLs  
Bot control dashboard  
Application integration  
IP sets  
Regex pattern sets  
Rule groups  
AWS Marketplace managed rules  
Switch to AWS WAF Classic  
AWS Shield Getting started Overview

Configure metrics

Amazon CloudWatch metrics

CloudWatch metrics allow you to monitor web requests, web ACLs, and rules.

Rules	CloudWatch metric name
<input checked="" type="checkbox"/> AWS-AWSManagedRulesSQLiRuleSet	AWS-AWSManagedRulesSQLiRuleSet
<input checked="" type="checkbox"/> AWS-AWSManagedRulesKnownBadInputsRuleSet	AWS-AWSManagedRulesKnownBadInputsRuleSet
<input checked="" type="checkbox"/> AWS-AWSManagedRulesAmazonIpReputationList	AWS-AWSManagedRulesAmazonIpReputationList
<input checked="" type="checkbox"/> AWS-AWSManagedRulesCommonRuleSet	AWS-AWSManagedRulesCommonRuleSet
<input checked="" type="checkbox"/> AWS-AWSManagedRulesLinuxRuleSet	AWS-AWSManagedRulesLinuxRuleSet

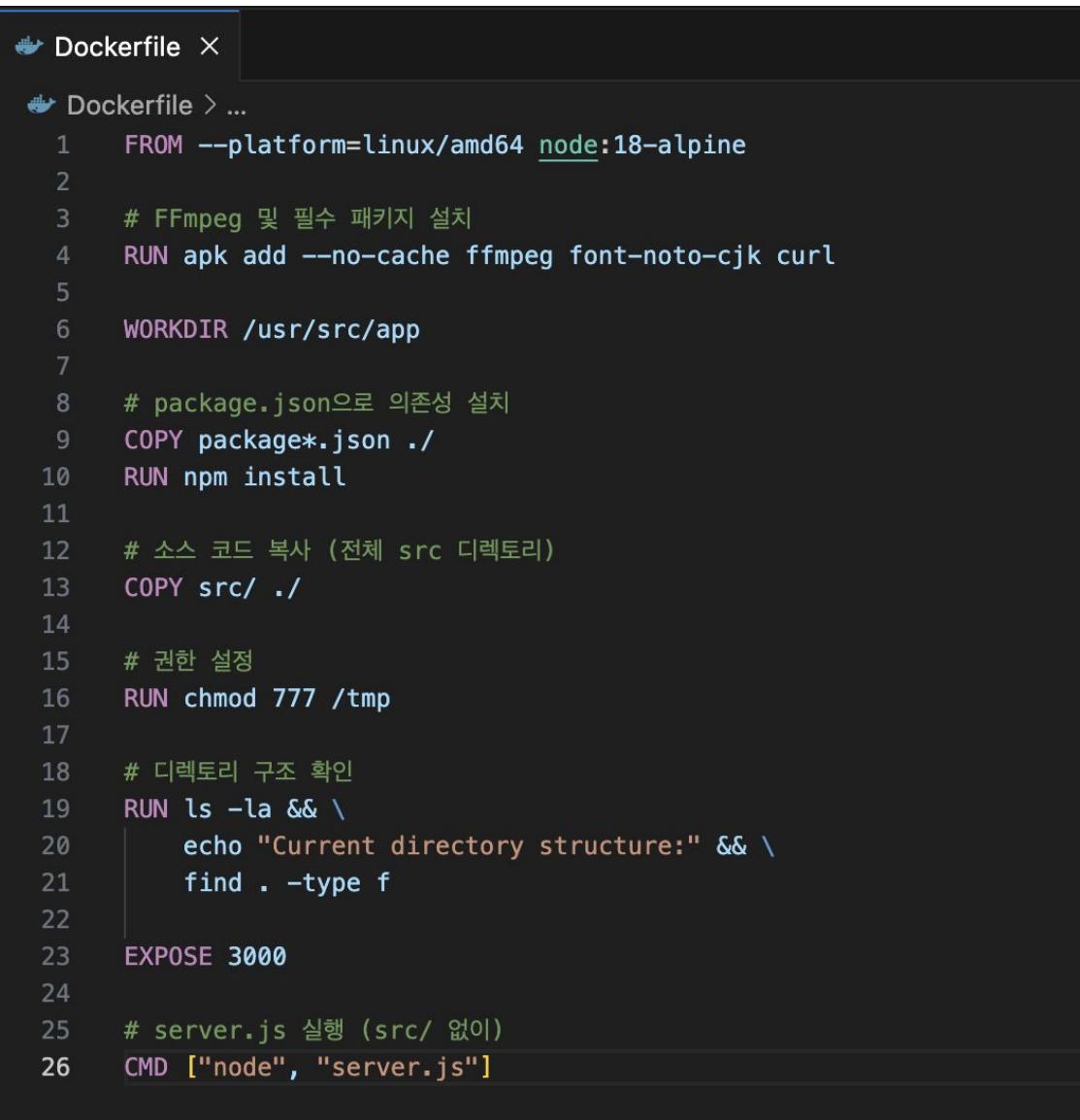
Request sampling options

If you disable request sampling, you can't view requests that match your web ACL rules.

Step 1: Describe web ACL and associate it to AWS resources  
Step 2: Add rules and rule groups  
Step 3: Set rule priority  
Step 4: Configure metrics  
Step 5: Review and create web ACL

AWS WAF와 CloudWatch를 통합하여 웹 애플리케이션의 보안을 강화하고, 실시간 트래픽 모니터링을 통해 위협에 신속 대응

# Dockerfile



A screenshot of a code editor showing a Dockerfile. The file contains 26 lines of Docker build instructions. The code uses color-coded syntax highlighting for commands like FROM, RUN, COPY, and EXPOSE. The Dockerfile specifies a Node.js base image, installs FFmpeg and other dependencies via apk add, sets the working directory to /usr/src/app, copies package.json and source code, installs npm dependencies, and exposes port 3000.

```
1  FROM --platform=linux/amd64 node:18-alpine
2
3  # FFmpeg 및 필수 패키지 설치
4  RUN apk add --no-cache ffmpeg font-noto-cjk curl
5
6  WORKDIR /usr/src/app
7
8  # package.json으로 의존성 설치
9  COPY package*.json .
10 RUN npm install
11
12 # 소스 코드 복사 (전체 src 디렉토리)
13 COPY src/ .
14
15 # 권한 설정
16 RUN chmod 777 /tmp
17
18 # 디렉토리 구조 확인
19 RUN ls -la && \
20     echo "Current directory structure:" && \
21     find . -type f
22
23 EXPOSE 3000
24
25 # server.js 실행 (src/ 없이)
26 CMD ["node", "server.js"]
```

FFmpeg와 Node.js 환경을 포함한 이미지 빌드 설정과 AMD64 플랫폼 명시적 지정

# ECR

The screenshot shows the AWS Amazon ECR console. The left sidebar has sections for 'Private registry' (Repositories, Features & Settings) and 'Public registry' (Repositories, Settings). The main area is titled 'Private repositories (1)' and lists a single repository: 'video-generation-service'. The table columns are Repository name, URI, Created at, Tag immutability, and Encryption type. The repository details are: video-generation-service, 941377161069.dkr.ecr.ap-northeast-2.amazonaws.com/video-generation-service, December 23, 2024, 15:43:37 (UTC+09), Mutable, AES-256.

Repository name	URI	Created at	Tag immutability	Encryption type
video-generation-service	941377161069.dkr.ecr.ap-northeast-2.amazonaws.com/video-generation-service	December 23, 2024, 15:43:37 (UTC+09)	Mutable	AES-256

비디오 생성 서비스를 위한 프라이빗 Docker 이미지 저장소 구성

# ECS

The screenshot shows the AWS Management Console interface for the Amazon Elastic Container Service (ECS). The top navigation bar includes the AWS logo, a search bar, and various service icons such as VPC, EC2, S3, RDS, CloudFormation, Lambda, Route 53, Certificate Manager, CloudFront, Amazon Rekognition, Amazon Bedrock, DynamoDB, and CloudWatch. The region is set to Seoul. The main title is "Amazon Elastic Container Service > Clusters". Below the title, it says "Clusters (1) Info" and "Last updated December 29, 2024 at 02:00 (UTC+9:00)". A prominent orange button labeled "Create cluster" is visible. The main table displays one cluster: "video-generation-cluster". The table columns are: Cluster, Services, Tasks, Container instances, CloudWatch monitoring, and Capacity provider strategy. The cluster details are: 0 Services, 0 Tasks, 0 Container instances (0 EC2), Default CloudWatch monitoring, and No default found for Capacity provider strategy.

비디오 생성을 위한 ECS 클러스터 생성

# ECS

```
task-definition.json
deploy > task-definition.json > ...
1  [
2    "family": "video-generation-service",
3    "networkMode": "awsvpc",
4    "requiresCompatibilities": ["FARGATE"],
5    "cpu": "2048",
6    "memory": "4096",
7    "executionRoleArn": "arn:aws:iam:941377161069:role/ecsTaskExecutionRole",
8    "taskRoleArn": "arn:aws:iam:941377161069:role/videoServiceTaskRole",
9    "volumes": [],
10   "containerDefinitions": [
11     {
12       "name": "video-generator",
13       "image": "941377161069.dkr.ecr.ap-northeast-2.amazonaws.com/video-generation-service:latest",
14       "essential": true,
15       "privileged": false,
16       "readonlyRootFilesystem": false,
17       "portMappings": [
18         {
19           "containerPort": 3000,
20           "protocol": "tcp"
21         }
22       ],
23       "environment": [
24         {
25           "name": "NODE_ENV",
26           "value": "production"
27         },
28         {
29           "name": "AWS_REGION",
30           "value": "ap-northeast-2"
31         }
32       ],
33       "logConfiguration": {
34         "logDriver": "awslogs",
35         "options": {
36           "awslogs-group": "/ecs/video-generation",
37           "awslogs-region": "ap-northeast-2",
38           "awslogs-stream-prefix": "ecs",
39           "awslogs-create-group": "true"
40         }
41       },
42       "healthCheck": {
43         "command": ["CMD-SHELL", "curl -f http://localhost:3000/health || exit 1"],
44         "interval": 30
45       }
46     }
47   ]
48 ]
```

Fargate 기반 컨테이너 설정과 환경 변수, 로깅 등 ECS 태스크 상세 정의

# ECS

```
{} ecs-service.json ×
deploy > {} ecs-service.json > ...
1  {
2    "cluster": "video-generation-cluster",
3    "serviceName": "video-generation-service",
4    "taskDefinition": "video-generation-service",
5    "desiredCount": 0,
6    "launchType": "FARGATE",
7    "platformVersion": "LATEST",
8    "networkConfiguration": {
9      "awsvpcConfiguration": {
10        "subnets": [
11          "subnet-03f55db5e52cb706a",
12          "subnet-09d8fd378d8dbc2d2"
13        ],
14        "securityGroups": ["sg-0ff5654960154de1a"],
15        "assignPublicIp": "ENABLED"
16      }
17    }
18 }
```

subnet-09d8fd378d8dbc2d2 / shopreel-private-subnet-a

[Details](#) | Flow logs | Route table | Network ACL | CIDR reservations | Sharing | Tags

Details	
Subnet ID	<a href="#">subnet-09d8fd378d8dbc2d2</a>
IPv4 CIDR	<a href="#">10.1.3.0/24</a>
Availability Zone	<a href="#">ap-northeast-2a</a>
Route table	<a href="#">rtb-0e330bbca0c2e3639   private-route-table</a>
Auto-assign IPv6 address	No
Subnet ARN	<a href="#">arn:aws:ec2:ap-northeast-2:941377161069:subnet/subnet-09d8fd378d8dbc2d2</a>
State	<span>Available</span>
IPv6 CIDR	-
Available IPv4 addresses	250
Network border group	<a href="#">ap-northeast-2</a>
Availability Zone ID	<a href="#">apne2-az1</a>
Default subnet	No
Network ACL	<a href="#">acl-077254096efd99e5a</a>
Customer-owned IPv4 pool	-
Auto-assign customer-owned IPv4	-
Block Public Access	<input checked="" type="checkbox"/> Off
IPv6 CIDR association ID	-
VPC	<a href="#">vpc-075eec3359d88133f   shopreel-vpc</a>
Auto-assign public IPv4 address	No
Outpost ID	-
Hostname type	-

sg-0d5de6b5a0768cf7d - video-generation-service-sg

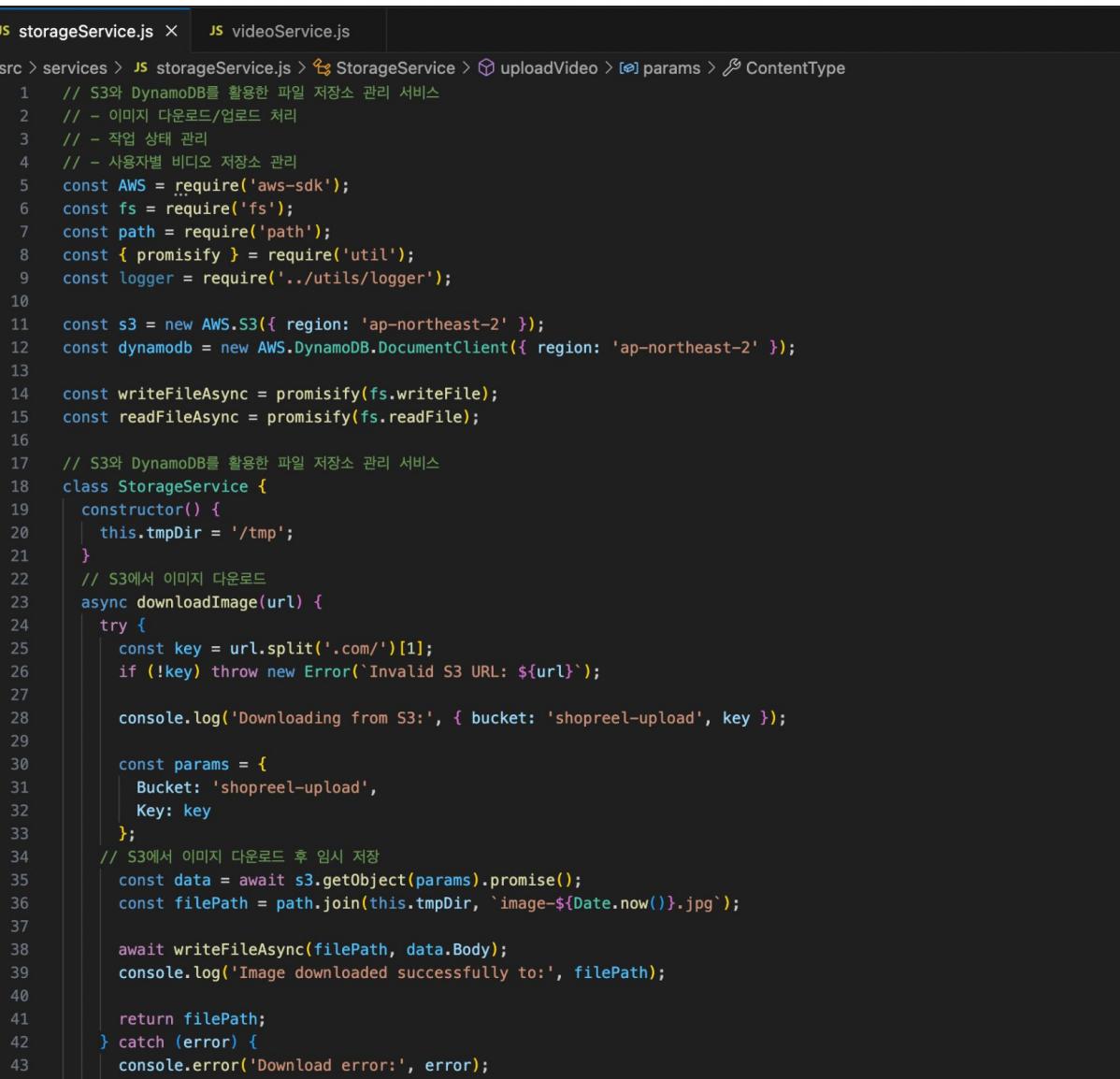
[Details](#) | [Inbound rules](#) | [Outbound rules](#) | [Sharing - new](#) | [VPC associations - new](#) | [Tags](#)

**Inbound rules (1)**

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-04aed9d5d95b86b1d	IPv4	Custom TCP	TCP	3000

프라이빗 서브넷과 보안 그룹 설정으로 안전한 ECS 서비스 환경 구성

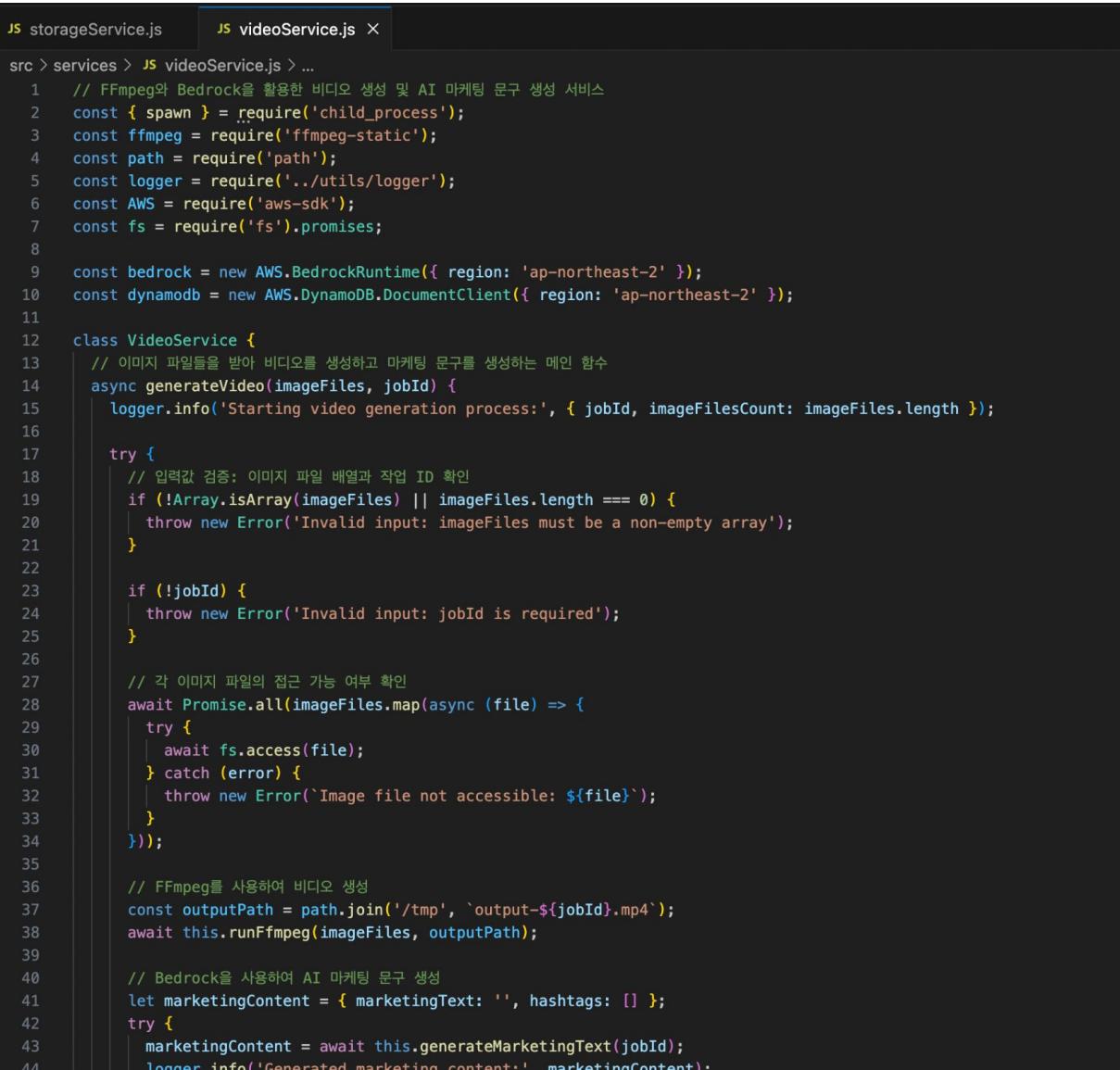
# ECS(Backend)



```
storageService.js
src > services > storageService.js > StorageService > uploadVideo > params > Content-Type
1 // S3와 DynamoDB를 활용한 파일 저장소 관리 서비스
2 // - 이미지 다운로드/업로드 처리
3 // - 작업 상태 관리
4 // - 사용자별 비디오 저장소 관리
5 const AWS = require('aws-sdk');
6 const fs = require('fs');
7 const path = require('path');
8 const { promisify } = require('util');
9 const logger = require('../utils/logger');
10
11 const s3 = new AWS.S3({ region: 'ap-northeast-2' });
12 const dynamodb = new AWS.DynamoDB.DocumentClient({ region: 'ap-northeast-2' });
13
14 const writeFileAsync = promisify(fs.writeFile);
15 const readFileSync = promisify(fs.readFile);
16
17 // S3와 DynamoDB를 활용한 파일 저장소 관리 서비스
18 class StorageService {
19   constructor() {
20     this.tmpDir = '/tmp';
21   }
22   // S3에서 이미지 다운로드
23   async downloadImage(url) {
24     try {
25       const key = url.split('.com')[1];
26       if (!key) throw new Error(`Invalid S3 URL: ${url}`);
27
28       console.log('Downloading from S3:', { bucket: 'shopreel-upload', key });
29
30       const params = {
31         Bucket: 'shopreel-upload',
32         Key: key
33       };
34       // S3에서 이미지 다운로드 후 임시 저장
35       const data = await s3.getObject(params).promise();
36       const filePath = path.join(this.tmpDir, `image-${Date.now()}.jpg`);
37
38       await writeFileAsync(filePath, data.Body);
39       console.log('Image downloaded successfully to:', filePath);
40
41       return filePath;
42     } catch (error) {
43       console.error('Download error:', error);
44     }
45   }
46 }
```

S3 버킷과 DynamoDB를 활용한 파일 저장 및 작업 상태 관리 서비스

# ECS(Backend)



```
JS storageService.js JS videoService.js X
src > services > JS videoService.js > ...
1 // FFmpeg와 Bedrock을 활용한 비디오 생성 및 AI 마케팅 문서 생성 서비스
2 const { spawn } = require('child_process');
3 const ffmpeg = require('ffmpeg-static');
4 const path = require('path');
5 const logger = require('../utils/logger');
6 const AWS = require('aws-sdk');
7 const fs = require('fs').promises;
8
9 const bedrock = new AWS.BedrockRuntime({ region: 'ap-northeast-2' });
10 const dynamodb = new AWS.DynamoDB.DocumentClient({ region: 'ap-northeast-2' });
11
12 class VideoService {
13   // 이미지 파일들을 받아 비디오를 생성하고 마케팅 문서를 생성하는 메인 함수
14   async generateVideo(imageFiles, jobId) {
15     logger.info('Starting video generation process:', { jobId, imageFilesCount: imageFiles.length });
16
17     try {
18       // 입력값 검증: 이미지 파일 배열과 작업 ID 확인
19       if (!Array.isArray(imageFiles) || imageFiles.length === 0) {
20         throw new Error('Invalid input: imageFiles must be a non-empty array');
21       }
22
23       if (!jobId) {
24         throw new Error('Invalid input: jobId is required');
25       }
26
27       // 각 이미지 파일의 접근 가능 여부 확인
28       await Promise.all(imageFiles.map(async (file) => {
29         try {
30           await fs.access(file);
31         } catch (error) {
32           throw new Error(`Image file not accessible: ${file}`);
33         }
34       }));
35
36       // FFmpeg를 사용하여 비디오 생성
37       const outputPath = path.join('/tmp', `output-${jobId}.mp4`);
38       await this.runFfmpeg(imageFiles, outputPath);
39
40       // Bedrock을 사용하여 AI 마케팅 문서 생성
41       let marketingContent = { marketingText: '', hashtags: [] };
42       try {
43         marketingContent = await this.generateMarketingText(jobId);
44         logger.info(`Generated marketing content: ${marketingContent}`);
45       } catch (error) {
46         logger.error(`Error generating marketing content: ${error.message}`);
47       }
48     } catch (error) {
49       logger.error(`Error generating video: ${error.message}`);
50     }
51   }
52
53   runFfmpeg(imageFiles, outputPath) {
54     const command = `ffmpeg -i ${imageFiles} -c:v libx264 -c:a aac -b:a 128k -t 10 -f mp4 ${outputPath}`;
55     const spawnOptions = { stdio: 'inherit' };
56     const child = spawn(command, spawnOptions);
57     child.on('error', (err) => {
58       logger.error(`FFmpeg error: ${err.message}`);
59     });
60     child.on('close', (code) => {
61       if (code !== 0) {
62         logger.error(`FFmpeg exited with code ${code}`);
63       }
64     });
65   }
66
67   generateMarketingText(jobId) {
68     const marketingText = `Job ID: ${jobId} - Marketing Content`;
69     const hashtags = ['#JobID', '#MarketingContent'];
70     return { marketingText, hashtags };
71   }
72
73   static async generateVideoFromImageFiles(imageFiles, jobId) {
74     const videoService = new VideoService();
75     await videoService.generateVideo(imageFiles, jobId);
76   }
77 }
```

FFmpeg를 활용한 비디오 생성과 AI 마케팅 문서 생성을 담당하는 핵심 비디오 처리 서비스

# Lambda

The screenshot shows the AWS Lambda Functions list page. The left sidebar has a 'Lambda' section with 'Dashboard', 'Applications', and 'Functions' selected. Below it are sections for 'Additional resources' (Code signing configurations, Event source mappings, Layers, Replicas) and 'Related AWS resources' (Step Functions state machines). The main area displays a table titled 'Functions (7)' with columns: Function name, Description, Package type, Runtime, and Last modified. The functions listed are:

Function name	Description	Package type	Runtime	Last modified
<a href="#">presignedUrl</a>	-	Zip	Node.js 18.x	3 days ago
<a href="#">getUserVideos</a>	-	Zip	Node.js 18.x	3 days ago
<a href="#">generateVideo</a>	-	Zip	Node.js 18.x	2 days ago
<a href="#">errors-notification</a>	-	Zip	Node.js 18.x	3 weeks ago
<a href="#">alerts-notification</a>	-	Zip	Node.js 18.x	3 weeks ago
<a href="#">checkJobStatus</a>	-	Zip	Node.js 18.x	1 week ago
<a href="#">imageProcessing</a>	-	Zip	Node.js 18.x	1 week ago

7개의 Lambda 함수를 구성하여 이미지 처리, 비디오 생성, 알림 등 각 기능별 역할 분담

# Lambda(presignedUrl)

```
1 // S3 파일 업로드를 위한 서명된 URL을 생성하는 Lambda 함수
2 import { S3Client, PutObjectCommand } from "@aws-sdk/client-s3";
3 import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
4
5 const s3 = new S3Client({ region: "ap-northeast-2" });
6
7 export const handler = async (event) => {
8     // CORS 헤더 설정
9     const headers = {
10         "Access-Control-Allow-Origin": "https://app.shopreel.life",
11         "Access-Control-Allow-Headers": "Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token",
12         "Access-Control-Allow-Methods": "OPTIONS,GET,POST",
13         "Access-Control-Allow-Credentials": "true"
14     };
15
16     // OPTIONS 요청 처리 (CORS preflight)
17     if (event.httpMethod === "OPTIONS") {
18         return {
19             statusCode: 200,
20             headers,
21             body: null,
22         };
23     }
24
25     try {
26         // 요청 바디 파싱하여 파일 개수 추출
27         const body = JSON.parse(event.body);
28         const { fileCount } = body;
29
30         // 파일 개수 유효성 검사
31         if (!fileCount || fileCount < 1) {
32             return {
33                 statusCode: 400,
34                 headers,
35                 body: JSON.stringify({ error: "Invalid fileCount" })
36             };
37         }
38
39         // 인증 헤더 검증
40         const authHeader = event.headers.Authorization || event.headers.authorization;
41         if (!authHeader || !authHeader.startsWith('Bearer ')) {
42             return {
43                 statusCode: 401,
44                 headers,
```

클라이언트의 안전한 S3 직접 업로드를 위한 Presigned URL 생성 및 제공

# Lambda(getUserVideos)

```
1 // 특정 사용자의 생성된 비디오 목록을 S3에서 조회하는 Lambda 함수
2 // 사용자별 비디오 파일을 검색하고 관련 메타데이터를 DynamoDB에서 함께 조회
3 import { S3Client, ListObjectsV2Command } from "@aws-sdk/client-s3";
4 import { DynamoDBClient, GetItemCommand } from "@aws-sdk/client-dynamodb";
5
6 const s3 = new S3Client({ region: "ap-northeast-2" });
7 const dynamodb = new DynamoDBClient({ region: "ap-northeast-2" });
8
9 // JWT 토큰을 디코딩하여 페이로드를 추출하는 유ти리티 함수
10 const parseJwt = (token) => {
11   try {
12     const base64Url = token.split('.')[1];
13     const base64 = base64Url.replace(/-/g, '+').replace(/_/g, '/');
14     const jsonPayload = decodeURIComponent(atob(base64).split('').map(c => {
15       return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);
16     }).join(''));
17     return JSON.parse(jsonPayload);
18   } catch (error) {
19     console.error('JWT parsing error:', error);
20     return null;
21   }
22 };
23
24 // Authorization 헤더에서 Bearer 토큰을 추출하고 사용자 ID를 반환하는 함수
25 const extractUserIdFromToken = (authHeader) => {
26   if (!authHeader || !authHeader.startsWith('Bearer ')) {
27     return null;
28   }
29   const token = authHeader.slice(7);
30   const decodedToken = parseJwt(token);
31   return decodedToken?.sub || null;
32 };
33
34 export const handler = async (event) => {
35   // CORS 설정을 위한 응답 헤더 정의
36   const headers = {
37     "Access-Control-Allow-Origin": "https://app.shopreel.life",
38     "Access-Control-Allow-Headers": "Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token",
39     "Access-Control-Allow-Methods": "OPTIONS,GET,POST",
40     "Access-Control-Allow-Credentials": "true"
41   };
42
43   // CORS preflight 요청 처리
44   if (event.httpMethod === "OPTIONS") {
45     return {
46       statusCode: 200,
47       headers,
48       body: ""
49     };
50   }
51
52   const userId = extractUserIdFromToken(event.headers.authorization);
53   if (!userId) {
54     return {
55       statusCode: 401,
56       headers,
57       body: "Unauthorized"
58     };
59   }
60
61   const s3Client = new S3Client();
62   const dynamoDBClient = new DynamoDBClient();
63
64   const listObjectsCommand = new ListObjectsV2Command({
65     Bucket: `user-videos-${userId}`,
66     MaxKeys: 10
67   });
68   const [listObjectsResult] = await s3Client.send(listObjectsCommand);
69   const objects = listObjectsResult.Contents || [];
70
71   const itemsPromises = objects.map(async (object) => {
72     const getItemCommand = new GetItemCommand({
73       TableName: "user-video-metadatas",
74       Key: { videoId: object.Key }
75     });
76     const [getItemResult] = await dynamoDBClient.send(getItemCommand);
77     return { ...object, ...getItemResult.Item };
78   });
80   const items = await Promise.all(itemsPromises);
81
82   return {
83     statusCode: 200,
84     headers,
85     body: JSON.stringify(items)
86   };
87 }
```

사용자별 생성된 비디오 목록과 관련 마케팅 문구를 조회하여 제공

# Lambda(generateVideo)

```
1 // 사용자 요청에 따라 이미지를 동영상으로 변환하는 ECS 작업을 시작하는 Lambda 핸들러
2 // Cognito 인증을 통해 사용자 검증 후, ECS Fargate에서 비디오 생성 작업을 실행
3 import { ECSClient, RunTaskCommand } from "@aws-sdk/client-ecs";
4
5 const ecs = new ECSClient({ region: "ap-northeast-2" });
6
7 export const handler = async (event) => {
8     // CORS 헤더 설정
9     const headers = {
10         "Access-Control-Allow-Origin": "https://app.shopreel.life",
11         "Access-Control-Allow-Headers": "Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token",
12         "Access-Control-Allow-Methods": "OPTIONS,GET,POST",
13         "Access-Control-Allow-Credentials": "true",
14         "Content-Type": "application/json"
15     };
16
17     // OPTIONS 요청 처리 (CORS preflight)
18     if (event.httpMethod === "OPTIONS") {
19         return {
20             statusCode: 200,
21             headers,
22             body: null,
23         };
24     }
25
26     try {
27         // 요청 바디에서 이미지 URL과 키워드 추출
28         const { imageUrls, keywords } = JSON.parse(event.body);
29
30         // 입력값 유효성 검사
31         if (!imageUrls || !Array.isArray(imageUrls) || imageUrls.length === 0) {
32             throw new Error('Invalid input: imageUrls must be a non-empty array');
33         }
34
35         // Cognito 인증 토큰에서 사용자 ID 추출
36         const authHeader = event.headers.Authorization || event.headers.authorization;
37         if (!authHeader) {
38             throw new Error('Authorization header is required');
39         }
40
41         const token = authHeader.split(' ')[1];
42         const tokenParts = token.split('.');
43         const payload = JSON.parse(Buffer.from(tokenParts[1], 'base64').toString());
44         const userId = payload.sub;
```

사용자 이미지를 받아 ECS Fargate에서 비디오 생성 작업을 실행하고 관리

# Lambda(errors-notification)

```
1 // 시스템 오류 발생 시 Slack으로 긴급 알림을 전송하는 Lambda 핵수
2 // 일반 알림과 달리 경고 아이콘과 함께 오류의 심각도를 표시
3 import https from 'https';
4 import { URL } from 'url';
5
6 // Slack webhook URL을 환경 변수로 설정
7 const SLACK_WEBHOOK_URL = process.env.SLACK_WEBHOOK_URL;
8
9 export const handler = async (event) => {
10   try {
11     // SNS 메시지 파싱
12     const message = event.Records[0].Sns;
13
14     // Slack 메시지 포맷 (에러 알림용)
15     const slackMessage = {
16       text: message.Subject ? `*${message.Subject}*${message.Message}` : message.Message,
17       username: 'ShopReel 오류',
18       icon_emoji: ':warning:',
19       blocks: [
20         {
21           type: "header",
22           text: {
23             type: "plain_text",
24             text: "⚠ ShopReel 오류 발생",
25             emoji: true
26           }
27         },
28         {
29           type: "section",
30           text: {
31             type: "mrkdwn",
32             text: message.Subject ? `*${message.Subject}*${message.Message}` : message.Message
33           }
34         },
35         {
36           type: "section",
37           fields: [
38             {
39               type: "mrkdwn",
40               text: `*발생 시간*: ${new Date().toLocaleString('ko-KR', { timeZone: 'Asia/Seoul' })}`
41             },
42             {
43               type: "mrkdwn",
44               text: `*주제*: ${message.Subject}`

45         
```

시스템 오류 발생 시 즉시 Slack으로 알림을 전송하여 신속한 대응 지원

# Lambda(alerts-notification)

```
1 // 일반적인 시스템 알림을 Slack 채널로 전송하는 Lambda 함수
2 // 알림 메시지를 구조화된 형식으로 변환하여 Slack webhook을 통해 전달
3 import https from 'https';
4 import { URL } from 'url';
5
6 // Slack webhook URL을 환경 변수에서 가져옴
7 const SLACK_WEBHOOK_URL = process.env.SLACK_WEBHOOK_URL;
8
9 export const handler = async (event) => {
10   try {
11     // SNS에서 전달된 메시지를 파싱
12     const message = event.Records[0].Sns;
13
14     // Slack 메시지 포맷 구성
15     // - username: 알림 발신자 이름
16     // - icon_emoji: 알림에 표시될 이모지
17     // - blocks: Slack의 Block Kit을 사용한 메시지 구조화
18     const slackMessage = {
19       text: message.Subject ? `*${message.Subject}*\\n${message.Message}` : message.Message,
20       username: 'ShopReel 알림',
21       icon_emoji: ':bell:',
22       blocks: [
23         // 헤더 섹션: 알림 타이틀 표시
24         {
25           type: "header",
26           text: {
27             type: "plain_text",
28             text: "⚠ ShopReel 알림",
29             emoji: true
30           }
31         },
32         // 본문 섹션: 알림 내용 표시
33         {
34           type: "section",
35           text: {
36             type: "mrkdwn",
37             text: message.Subject ? `*${message.Subject}*\\n${message.Message}` : message.Message
38           }
39         },
40         // 컨텍스트 섹션: 알림 시간 표시 (한국 시간대 기준)
41         {
42           type: "context",
43           elements: [
44             {
45               type: "text",
46               text: "작성일: " + new Date().toLocaleString('ko-KR')
47             }
48           ]
49         }
50       ]
51     }
52
53     // Slack webhook URL로 POST 요청 보내기
54     const response = await https.post(SLACK_WEBHOOK_URL, JSON.stringify(slackMessage));
55
56     if (response.statusCode === 200) {
57       console.log('Slack webhook call successful');
58     } else {
59       console.error(`Slack webhook call failed with status code ${response.statusCode}`);
60     }
61   } catch (error) {
62     console.error(`Error processing SNS message: ${error.message}`);
63   }
64 }
```

시스템 일반 알림을 Slack 채널로 전송하여 실시간 모니터링 제공

# Lambda(checkJobStatus)

```
1 // 비디오 생성 작업의 상태를 확인하는 Lambda 함수
2 // DynamoDB에서 jobId를 기반으로 작업 진행 상태와 결과를 조회
3 import { DynamoDBClient, GetItemCommand } from "@aws-sdk/client-dynamodb";
4 const dynamodb = new DynamoDBClient();
5
6 export const handler = async (event) => {
7   try {
8     // URL 경로 파라미터에서 jobId 추출
9
10    const jobId = event.pathParameters.jobId;
11
12    // DynamoDB에서 해당 작업 상태 조회
13    const response = await dynamodb.send(new GetItemCommand({
14      TableName: "JobStatus",
15      Key: {
16        jobId: { S: jobId }
17      }
18    }));
19
20    // 작업을 찾지 못한 경우 404 반환
21    if (!response.Item) {
22      return {
23        statusCode: 404,
24        headers: {
25          "Access-Control-Allow-Origin": "*"
26        },
27        body: JSON.stringify({ error: "Job not found" })
28      };
29    }
30
31    // 작업 상태 정보를 클라이언트에 반환
32    return {
33      statusCode: 200,
34      headers: {
35        "Access-Control-Allow-Origin": "*"
36      },
37      body: JSON.stringify({
38        jobId: response.Item.jobId.S,
39        status: response.Item.status.S,
40        videoUrl: response.Item.videoUrl?.S,
41        updatedAt: response.Item.updatedAt.S,
42        error: response.Item.error?.S
43      })
44    };
45  }
46}
```

영상 생성 작업의 현재 상태를 실시간으로 조회하여 유저에게 제공

# Lambda(imageProcessing)

```
1 // 업로드된 이미지를 Rekognition으로 분석하고 결과를 처리하는 Lambda 함수
2 // 이미지 레이블 감지 후 분석 결과를 저장하고 비디오 생성 작업을 트리거
3 import { RekognitionClient, DetectLabelsCommand } from "@aws-sdk/client-rekognition";
4 import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";
5 import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";
6 import { LambdaClient, InvokeCommand } from "@aws-sdk/client-lambda";
7 import { ECSClient, RunTaskCommand } from "@aws-sdk/client-ecs";
8
9 const rekognition = new RekognitionClient();
10 const dynamodb = new DynamoDBClient();
11 const s3 = new S3Client();
12 const lambda = new LambdaClient();
13 const ecs = new ECSClient();
14
15 // URL에서 S3 키 추출하는 함수
16 const getKeyFromUrl = (url) => {
17   try {
18     if (url.startsWith('s3://')) {
19       // s3:// 형식 처리
20       return url.replace('s3://shopreel-upload/', '');
21     } else {
22       // https:// 형식 처리
23       const matches = url.match(/^(https?:\/\/[^\/]+\/(.*)$/);
24       return matches ? matches[1] : null;
25     }
26   } catch (error) {
27     console.error('Error extracting key from URL:', error);
28     throw error;
29   }
30 };
31
32 export const handler = async (event) => {
33   let jobId; // 상단에서 변수 선언
34
35   try {
36     const { urls } = event.body ? JSON.parse(event.body) : event;
37     jobId = `job-${Date.now()}`; // 먼저 jobId 생성
38
39     // Dynamodb에 초기 작업 상태 저장
40     await dynamodb.send(new PutItemCommand({
41       TableName: "JobStatus",
42       Item: {
43         jobId: { S: jobId },
44         status: { S: "PROCESSING" },
45       }
46     }));
47   } catch (error) {
48     console.error(`Error processing job ${jobId}: ${error}`);
49     return {
50       statusCode: 500,
51       body: `Error processing job ${jobId}: ${error}`,
52     };
53   }
54 }
```

Rekognition으로 이미지를 분석하고 결과를 기반으로 비디오 생성 작업 시작

# API Gateway

The screenshot shows the AWS API Gateway interface. The top navigation bar includes links for VPC, EC2, S3, RDS, CloudFormation, Lambda, Route 53, Certificate Manager, CloudFront, Amazon Rekognition, Amazon Bedrock, DynamoDB, and CloudWatch. The search bar contains the text "[Option+S]". The breadcrumb navigation shows: API Gateway > APIs > shopreelapi (uyuys91qe5) > Stages.

**Stages**

- prod**
  - /
  - /api/generate
    - OPTIONS
    - POST
  - /api/generate-video
    - OPTIONS
    - POST
  - /api/status/{jobId}
    - GET
    - OPTIONS
  - /api/upload
    - OPTIONS
    - POST
  - /api/videos
    - GET
    - OPTIONS

**Stage details**

**Stage name:** prod

**Rate Info:** 10000

**Burst Info:** 5000

**Web ACL:** -

**Client certificate:** -

**Invoke URL:** https://uyuys91qe5.execute-api.ap-northeast-2.amazonaws.com/prod

**Active deployment:** a8jf69 on December 29, 2024, 02:29 (UTC+09:00)

**Logs and tracing**

**CloudWatch logs:** Inactive

**Detailed metrics:** Inactive

**Data tracing:** Inactive

## 주요 엔드포인트

- /api/generate: 이미지 분석 및 영상 생성 시작
- /api/generate-video: 영상 생성 작업 실행
- /api/status/{jobId}: 작업 상태 조회
- /api/videos: 사용자별 생성된 영상 목록 조회
- /api/upload: Presigned URL 발급

영상 생성, 상태 확인, 업로드 등을 위한 REST API 엔드포인트 생성, CORS 설정을 통해 특정 도메인의 요청만 허용하도록 제한

# S3

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various options like General purpose buckets, Directory buckets, Table buckets, Access Grants, etc. The main area displays an account snapshot and a list of General purpose buckets. There are two buckets listed:

Name	AWS Region	IAM Access Analyzer	Creation date
<a href="#">shopreel-output</a>	Asia Pacific (Seoul) ap-northeast-2	<a href="#">View analyzer for ap-northeast-2</a>	December 10, 2024, 15:56:26 (UTC+09:00)
<a href="#">shopreel-upload</a>	Asia Pacific (Seoul) ap-northeast-2	<a href="#">View analyzer for ap-northeast-2</a>	December 10, 2024, 11:00:21 (UTC+09:00)

업로드된 이미지와 생성된 비디오를 저장하는 shopreel-upload, shopreel-output 버킷 생성

# S3(shopreel-upload)

Amazon S3

General purpose buckets

Directory buckets

Table buckets [New](#)

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

**Storage Lens**

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight [10](#)

AWS Marketplace for S3

uploads/

Objects Properties

Objects (2) Info

Actions ▾ Create folder Upload

Copy S3 URI Copy URL Download Open Delete

Find objects by prefix Show versions

Name	Type	Last modified	Size	Storage class
<a href="#">24c87dec-0091-7014-030f-a6855b177c54/</a>	Folder	-	-	-
<a href="#">8478ad2c-0011-70ed-d8c3-7a316cd39caf/</a>	Folder	-	-	-

Amazon S3

General purpose buckets

Directory buckets

Table buckets [New](#)

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

**Storage Lens**

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight [10](#)

AWS Marketplace for S3

24c87dec-0091-7014-030f-a6855b177c54/

Objects (3) Info

Actions ▾ Create folder Upload

Copy S3 URI Copy URL Download Open Delete

Find objects by prefix Show versions

Name	Type	Last modified	Size	Storage class
<a href="#">1735404098266-0.jpg</a>	jpg	December 29, 2024, 01:41:39 (UTC+09:00)	76.2 KB	Standard
<a href="#">1735404098266-1.jpg</a>	jpg	December 29, 2024, 01:41:39 (UTC+09:00)	93.0 KB	Standard
<a href="#">1735404098266-2.jpg</a>	jpg	December 29, 2024, 01:41:39 (UTC+09:00)	49.6 KB	Standard

사용자가 업로드한 이미지를 저장하기 위한 버킷으로, uploads/ 경로에 이미지 파일 보관

# S3(shopreel-output)

Amazon S3

**General purpose buckets**

- Directory buckets
- Table buckets [New](#)
- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight [10](#)

▶ AWS Marketplace for S3

Search [Option+S] | Bell | Help | Seoul | [Copy S3 URI](#)

Amazon S3 > Buckets > shopreel-output > users/

**Objects** Properties

**Objects (2) Info**

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">24c87dec-0091-7014-030f-a6855b177c54/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">8478ad2c-0011-70ed-d8c3-7a316cd39caf/</a>	Folder	-	-	-

Amazon S3

**General purpose buckets**

- Directory buckets
- Table buckets [New](#)
- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight [10](#)

▶ AWS Marketplace for S3

Search [Option+S] | Bell | Help | Seoul | [Copy S3 URI](#)

Amazon S3 > Buckets > shopreel-output > users/ > [24c87dec-0091-7014-030f-a6855b177c54...](#) > videos/ > [24c87dec-0091-7014-030f-a6855b177c54...](#)

**24c87dec-0091-7014-030f-a6855b177c54-1735404099547/**

**Objects** Properties

**Objects (1) Info**

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">output.mp4</a>	mp4	December 29, 2024, 01:42:35 (UTC+09:00)	191.4 KB	Standard

생성된 비디오를 사용자별로 구분하여 저장하는 버킷으로, users/{userId}/videos/{jobId} 구조로 관리

# DynamoDB

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. The main area is titled 'Tables (2) Info' and lists two tables: 'ContentMetadata' and 'JobStatus'. Both tables are active and have a partition key of type S (String). The 'ContentMetadata' table has a sort key '-' and 0 indexes, while the 'JobStatus' table has a sort key '-' and 1 index. Both tables have 0 replication regions and deletion protection off. There are also filters for tag keys and values, and a search bar at the top.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection
ContentMetadata	Active	contentId (S)	-	0	0	Off
JobStatus	Active	jobId (S)	-	1	0	Off

작업 상태와 콘텐츠 메타데이터를 저장하는 JobStatus, ContentMetadata 테이블 생성

# DynamoDB(JobStatus)

The screenshot shows the AWS DynamoDB 'Edit item' interface. At the top, there's a navigation bar with various AWS services like VPC, EC2, S3, RDS, CloudFormation, Lambda, Route 53, Certificate Manager, CloudFront, Amazon Rekognition, Amazon Bedrock, and CloudWatch. Below that, the path 'DynamoDB > Explore items: JobStatus > Edit item' is visible. The main area is titled 'Edit item' with a sub-section 'Attributes'. It shows a hierarchical structure of attributes:

- Attribute name:** jobId - Partition key, Value: 8478ad2c-0011-70ed-d8c3-7a316cd39caf-1735403543454, Type: String.
- marketingContent:** A Map attribute with one item:
  - hashtag:** A List attribute with four items:
    - 0: 프리미엄라이프
    - 1: 럭셔리
    - 2: MZ취향저격
    - 3: 품격있는선택
    - 4: FLEX

At the bottom right of the form, there are buttons for 'Form' (selected), 'JSON view', 'Add new attribute', and 'Remove'.

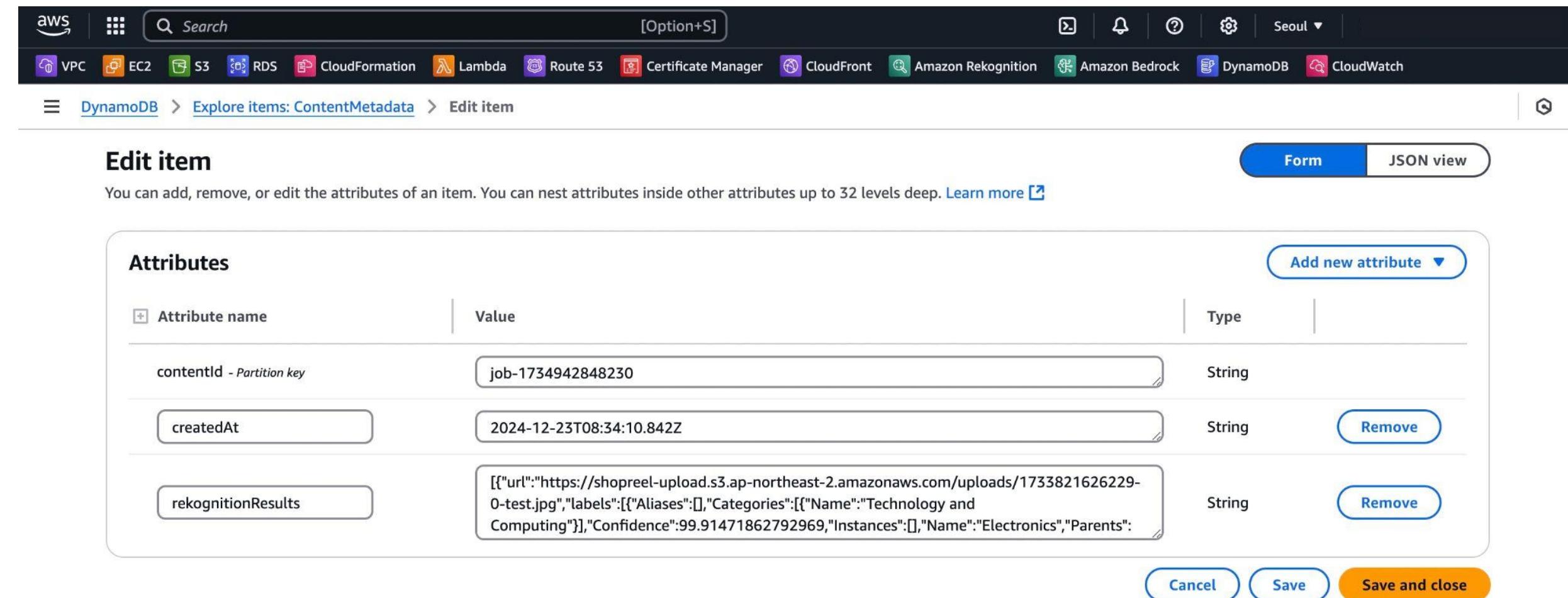
This screenshot shows the same 'Edit item' interface, but the attributes are listed flatly without nesting. The attributes and their values are:

Attribute name	Value	Type	Action
jobId	MZ취향저격	String	Remove
marketingContent	품격있는선택	String	Remove
status	FLEX	String	Remove
marketingText	럭셔리의 새로운 기준, 여기 있어요✨ 남들과는 다른 특별함을 원한다면? 이 제품으로 당신의 품격을 한층 업그레이드하세요🔥	String	Remove
updatedAt	COMPLETED	String	Remove
userId	2024-12-28T16:33:25.617Z	String	Remove
videoUrl	8478ad2c-0011-70ed-d8c3-7a316cd39caf	String	Remove
	https://shopreel-output.s3.ap-northeast-2.amazonaws.com/users/8478ad2c-0011-70ed-d8c3-7a316cd39caf/videos/8478ad2c-0011-70ed-d8c3-7a316cd39caf-1735403543454/output.mp4	String	Remove

At the bottom right, there are buttons for 'Cancel', 'Save', and 'Save and close'.

작업 ID를 기준으로 비디오 생성 상태, 생성된 URL, 마케팅 문구 등 작업 관련 정보 저장

# DynamoDB(ContentMetadata)



The screenshot shows the AWS DynamoDB 'Edit item' interface. The top navigation bar includes the AWS logo, a search bar, and links for various services like VPC, EC2, S3, RDS, CloudFormation, Lambda, Route 53, Certificate Manager, CloudFront, Amazon Rekognition, Amazon Bedrock, DynamoDB, and CloudWatch. The location bar indicates the user is in the 'DynamoDB > Explore items: ContentMetadata > Edit item' section. Below the navigation is a toolbar with 'Form' (selected) and 'JSON view' buttons. The main area is titled 'Edit item' and contains a table for managing attributes. The table has columns for 'Attribute name', 'Value', and 'Type'. There are three rows: 1) 'contentId - Partition key' with value 'job-1734942848230' and type 'String'. 2) 'createdAt' with value '2024-12-23T08:34:10.842Z' and type 'String'. 3) 'rekognitionResults' with a JSON string value representing a Rekognition analysis result and type 'String'. A blue 'Add new attribute' button is located at the top right of the table. At the bottom are 'Cancel', 'Save', and 'Save and close' buttons.

Rekognition 분석 결과를 저장하여 마케팅 문구 생성을 위한 이미지 특성 정보 관리

# CloudWatch

The screenshot shows the AWS CloudWatch Alarms page with three active alarms listed:

Name	State	Last state update (UTC)	Conditions	Actions
<a href="#">lambda-duration-monitoring</a>	OK	2024-12-30 18:32:03	Duration > 720000 for 1 datapoints within 5 minutes	Actions enabled
<a href="#">alerts-notification-monitoring</a>	OK	2024-12-30 20:14:16	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled
<a href="#">generateVideo-error-monitoring</a>	OK	2024-12-30 20:04:10	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled

Lambda 함수 실행 시간, 오류 발생 여부 등을 모니터링하고 임계값 초과 시 SNS 알림 트리거

# SNS

The screenshot shows the AWS SNS Topics page. On the left, there's a sidebar with 'Amazon SNS' selected. The main area displays a table titled 'Topics (2)'. The table has columns for 'Name', 'Type', and 'ARN'. Two rows are listed:

- Name:** shopreel-alerts-topic, **Type:** Standard, **ARN:** arn:aws:sns:ap-northeast-2:941377161069:shopreel-alerts-topic
- Name:** shopreel-errors-topic, **Type:** Standard, **ARN:** arn:aws:sns:ap-northeast-2:941377161069:shopreel-errors-topic

Action buttons at the top right include 'Edit', 'Delete', 'Publish message', and 'Create topic'.

The screenshot shows the AWS SNS Subscription details page for the 'shopreel-alerts-topic' subscription, identified by ARN: 'arn:aws:sns:ap-northeast-2:941377161069:shopreel-alerts-topic:0dac1a68-f9b2-4794-b768-1dd9f936abec'. The page is titled 'Subscription: 0dac1a68-f9b2-4794-b768-1dd9f936abec'.

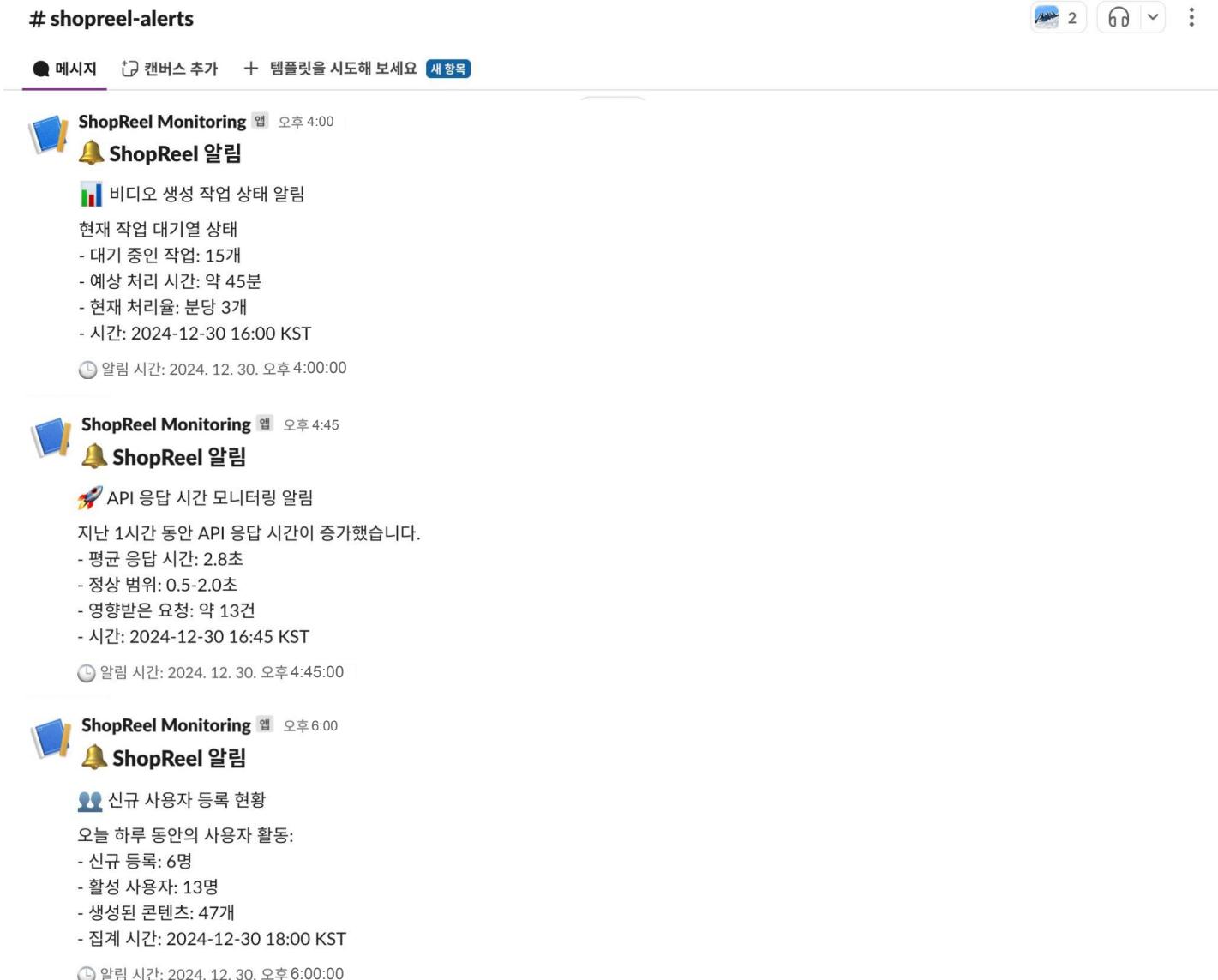
**Details:**

- ARN:** arn:aws:sns:ap-northeast-2:941377161069:shopreel-alerts-topic:0dac1a68-f9b2-4794-b768-1dd9f936abec
- Status:** Confirmed
- Protocol:** LAMBDA
- Endpoint:** arn:aws:lambda:ap-northeast-2:941377161069:function:alerts-notification
- Topic:** shopreel-alerts-topic
- Subscription Principal:** arn:aws:iam::941377161069:user/lucy0920

**Subscription filter policy:** No filter policy configured for this subscription. To apply a filter policy, edit this subscription.

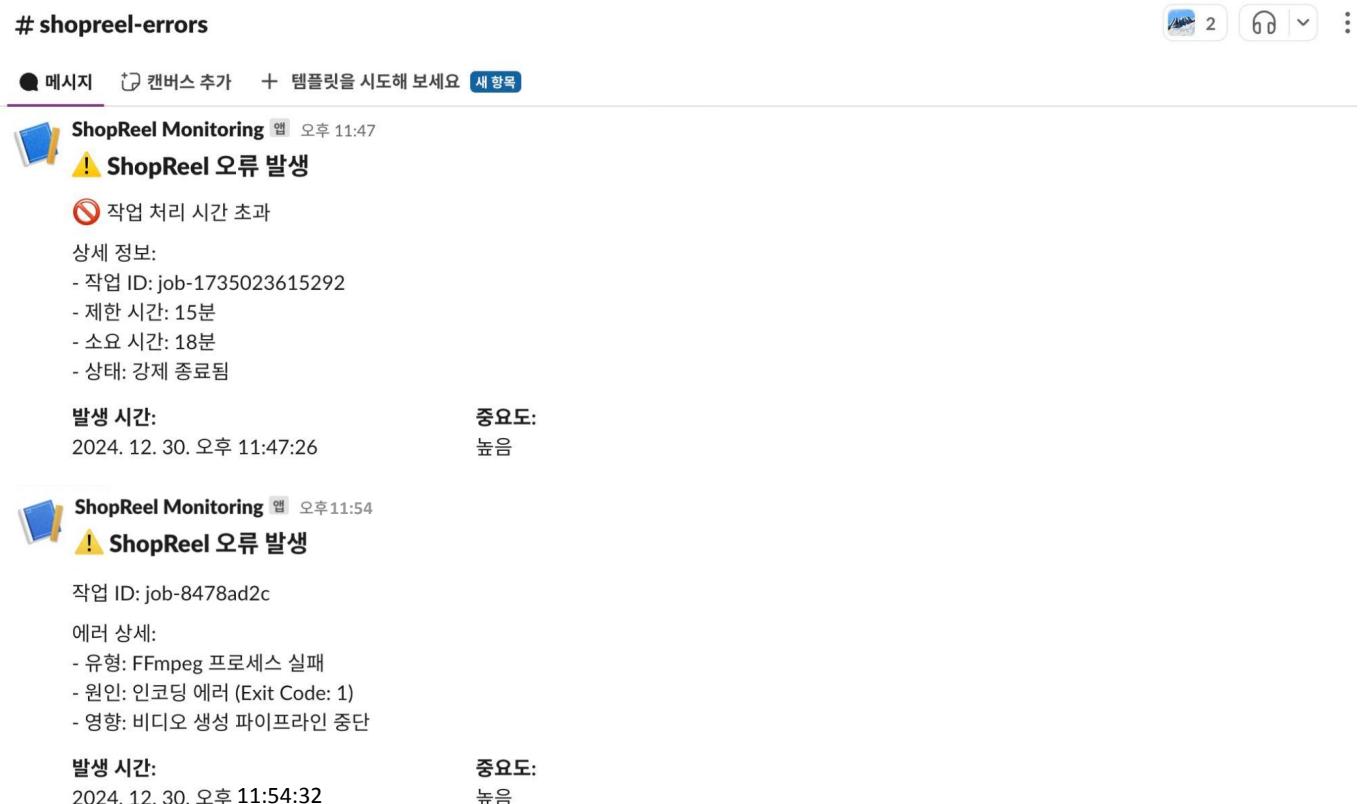
알림 종류별 SNS 토픽 생성 및 Lambda 함수와의 구독 설정

# SNS



#shopreel-alerts 채널에 비디오 작업 상태 알림, API 응답 지연 등 일반 상태 알림 전송

# SNS



#shopreel-errors 채널에 작업 실패, FFmpeg 오류 등 시스템 에러 상황을 실시간 알림

# Rekognition

```
JS imageProcessing.mjs > ...
JS imageProcessing.mjs > ...
1 // 업로드된 이미지를 Rekognition으로 분석하고 결과를 처리하는 Lambda 함수
2 // 이미지 레이블 감지 후 분석 결과를 저장하고 비디오 생성 작업을 트리거
3 import { RekognitionClient, DetectLabelsCommand } from "@aws-sdk/client-rekognition";
4 import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";
5 import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";
6 import { LambdaClient, InvokeCommand } from "@aws-sdk/client-lambda";
7 import { ECSClient, RunTaskCommand } from "@aws-sdk/client-ecs";
8
9 const rekognition = new RekognitionClient();
10 const dynamodb = new DynamoDBClient();
11 const s3 = new S3Client();
12 const lambda = new LambdaClient();
13 const ecs = new ECSClient();
14
15 // Rekognition으로 이미지 분석
16 const rekognitionResponse = await rekognition.send(new DetectLabelsCommand({
17   Image: {
18     Bytes: imageBytes
19   },
20   MaxLabels: 10,
21   MinConfidence: 70
22 });
23
24 console.log('Rekognition analysis completed');
25
26 return {
27   url,
28   labels: rekognitionResponse.Labels
29 };
30 } catch (error) {
31   console.error(`Error processing image ${url}:`, error);
32   return {
33     url,
34     error: error.message
35   };
36 }
37 );
```

업로드된 이미지에서 객체와 장면을 감지하여 레이블을 추출하고, 마케팅 문구 생성을 위한 데이터 제공

# Bedrock

```
98 // Bedrock API 호출
99 const response = await bedrock.invokeModel({
100   modelId: 'anthropic.claude-3-5-sonnet-20240620-v1:0',
101   contentType: 'application/json',
102   accept: 'application/json',
103   body: JSON.stringify({
104     anthropic_version: "bedrock-2023-05-31",
105     max_tokens: 1000,
106     messages: [
107       {
108         role: "user",
109         content: [
110           {
111             type: "text",
112             text: `다음 제품 특성을 바탕으로 SNS 마케팅용 홍보 문구를 생성해주세요:
113             제품 특성: ${labels}
114             요구사항:
115               1. 2~3개의 짧은 문장으로 구성
116               2. MZ세대를 타겟으로 한 트렌디한 표현 사용
117               3. 해시태그 3~5개 포함
118               4. 이모지 적절히 활용
119               5. 100자 이내로 작성
120
121             출력 형식:
122             본문: (마케팅 문구)
123             해시태그: (해시태그 목록)`
124           }
125         ]
126       }
127     ]
128   })
129 }).promise();
```

Rekognition에서 분석된 이미지 특성을 기반으로 Claude 모델을 활용하여 맞춤형 마케팅 문구와 해시태그 생성

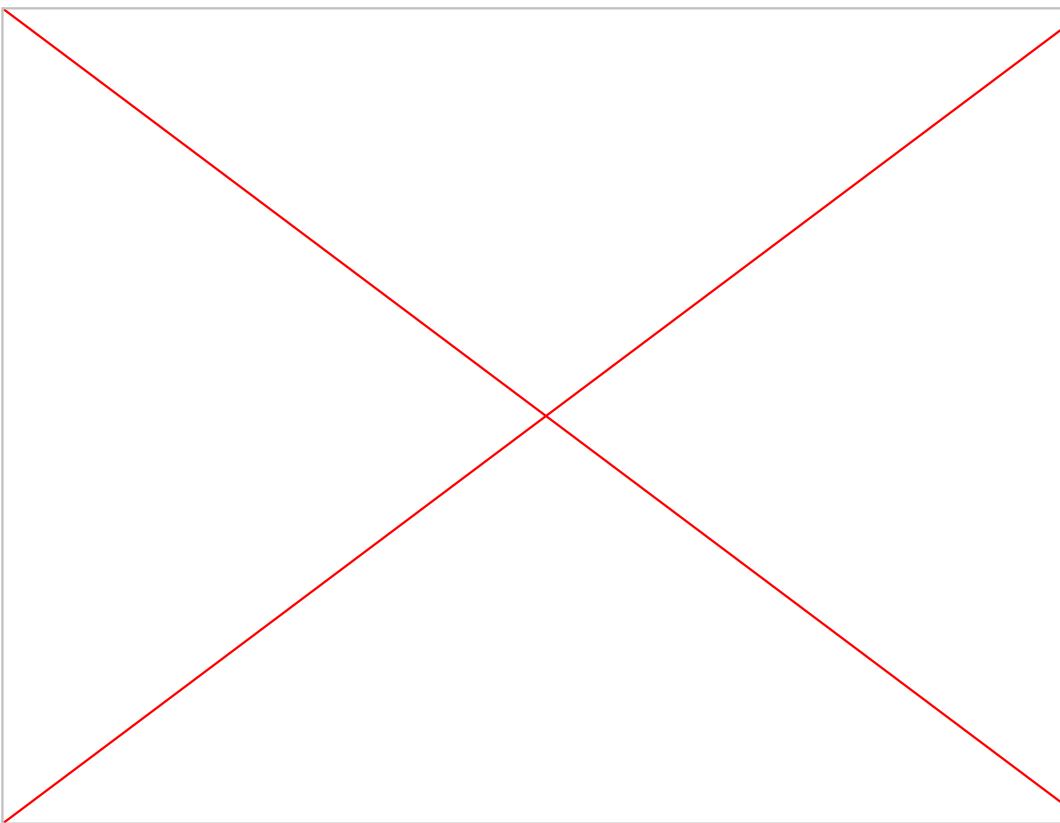
## | 프로젝트 성과

04-1 시연 영상

04-2 성과



## 시연 영상



<https://youtu.be/j2bdr9thSdc>

## 성과

운영 비용 최적화



AWS 서비스 아키텍처(Fargate, Lambda 등)를 활용해 고정 비용을 최소화하고, 사용량 기반 요금제를 통해 비용 효율성을 극대화

콘텐츠 제작 시간 단축



AI 기반 자동화로 콘텐츠 제작과 업로드 시간을 기존 대비 약 90% 단축하며, 빠른 마케팅 실행 가능

직관적 인터페이스 제공



키워드와 이미지만으로 콘텐츠를 생성할 수 있는 직관적 UI 설계로, 중소규모 비즈니스와 개인 창작자의 접근성 증대

## 성과

맞춤형 콘텐츠 강화



AI와 Amazon Rekognition을 활용한  
분석으로 고품질의 타겟 맞춤형 콘텐츠를  
제작하며, 다양한 플랫폼별 템플릿 제공

데이터 기반 운영  
최적화



DynamoDB에 저장된 데이터를 통해  
서비스 개선과 개인화 추천 기능 강화.  
AWS CloudWatch 및 Slack 알림을  
활용한 운영 효율성 제고

글로벌 확장성 지원



Amazon Bedrock으로 다국어 콘텐츠  
지원 및 AWS 글로벌 인프라를 활용해 국제  
시장 대응력을 강화

## | 문제 해결 및 리스크 관리

05-1 문제 해결 사례

05-2 리스크 관리



# 문제 해결 및 극복 과정

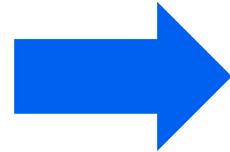
## 문제점

CORS(Cross-Origin Resource Sharing) 이슈  
와일드카드(\*)를 사용한 보안 설정 불가  
API Gateway preflight 요청 처리 실패

## 해결 방안

Lambda 함수에 표준화된 CORS 헤더 설정

ECS Task 실행 문제  
컨테이너 정상 실행됐으나 프로세스 중단  
실제 영상 생성 시작 오류



새로운 Lambda 함수로 영상 생성 프로세스 구현

M1/M2 맥에서 Docker 호환성 문제  
M1/M2 맥(ARM64)과 AWS ECS(AMD64)의 아키텍처 차이  
Docker 이미지 빌드 시 플랫폼 미지정으로 인한 호환성 오류

Dockerfile에 AMD64 플랫폼 명시  
FROM --platform=linux/amd64 node:18-alpine

## 리스크 관리

Lambda 함수나 ECS Task 실패 시 서비스가 중단될 가능성

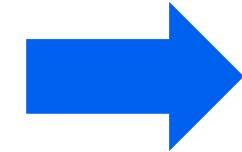
ECS Task 자동 재시작 설정  
CloudWatch 경보와 SNS/Slack 알림 연동

S3 버킷 내 데이터 삭제/손상으로 인한 데이터 손실 가능성

버전 관리 및 Lifecycle Policy 설정  
데이터 복구 기능과 저장소 최적화

트래픽 증가 시 기존의 리소스로 인한 성능 저하 가능성

Auto Scaling으로 ECS/Lambda 동적 확장  
리소스 사용량 모니터링 및 최적화



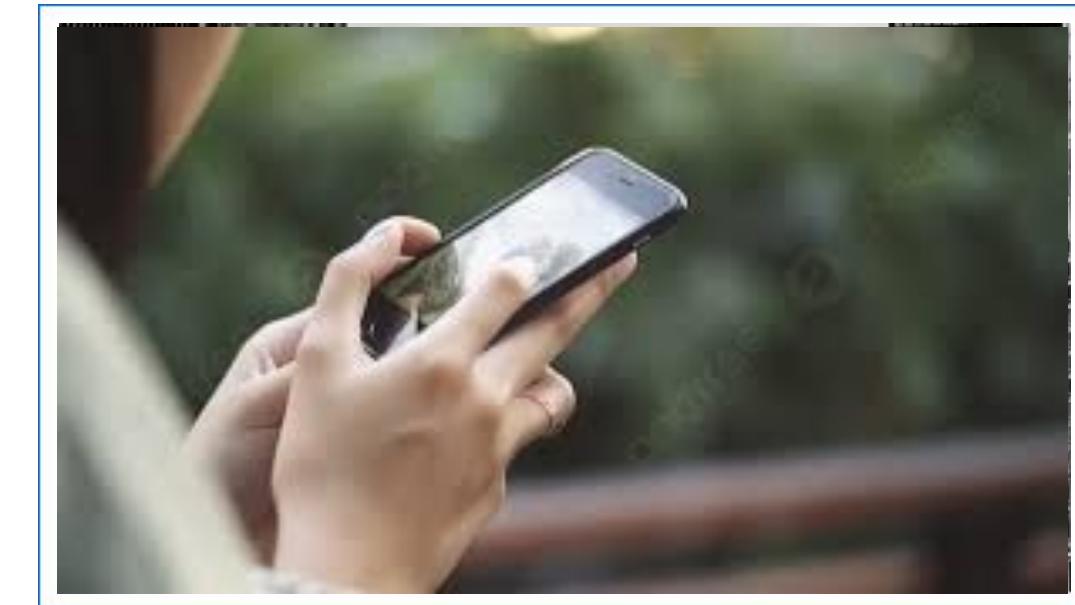
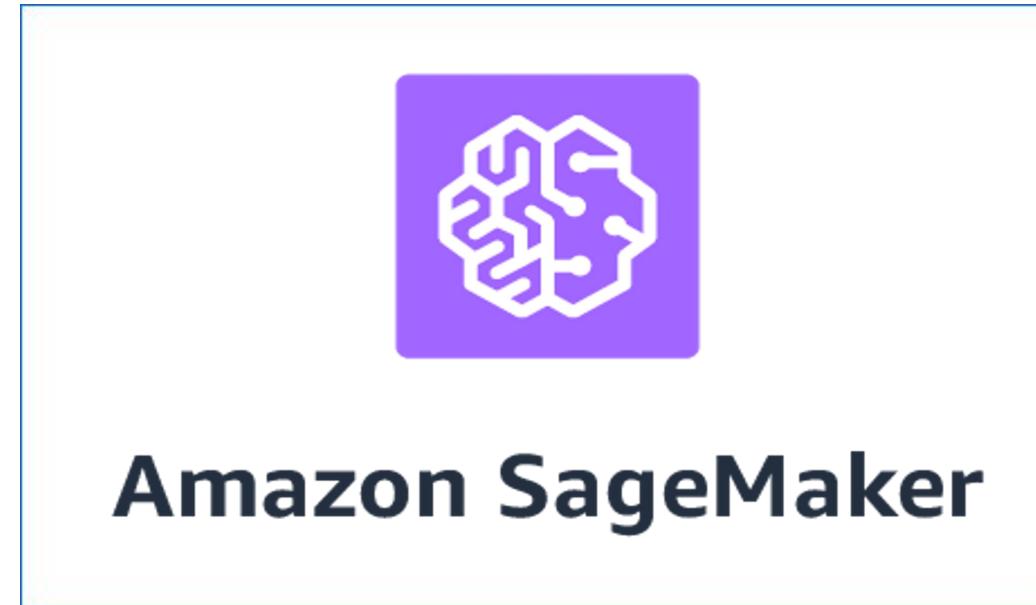
## | 향후 계획 및 결론

06-1 향후 계획

06-2 결론



# 향후 계획



## 서비스 기능 확장

- 다국어 지원 강화: Amazon Bedrock 연계로 다양한 언어의 마케팅 문구와 해시태그 생성
- 소셜 미디어 플랫폼 통합: Instagram, YouTube, TikTok API 연동으로 자동 업로드 기능 제공

## 인프라 고도화

- 멀티 리전 배포: AWS 글로벌 인프라를 활용해 전 세계 사용자 대상 성능 최적화
- AI 모델 고도화: Rekognition 및 SageMaker로 개인화된 마케팅 콘텐츠 품질 향상

## 사용자 기반 확대

- 커뮤니티 활성화: 사용자 참여를 높이기 위한 비디오 제작 튜토리얼 및 사례 공유
- 모바일 앱 출시: 모바일 기기에서도 콘텐츠 제작 및 관리가 가능한 앱 개발

# 결론

## 시장 문제 해결

- 높은 비용과 긴 소요 시간 개선
- 맞춤형 콘텐츠 제작 어려움 해소

## 개발 솔루션

- AWS 클라우드와 AI 기반 플랫폼 구축
- 간단한 입력으로 고품질 영상 제작
- 비용/시간 절감으로 경쟁력 확보





# Q & A

프레젠테이션을 마치겠습니다.

감사합니다.

