

AWS Beanstalk

수동 배포

1. AWS Beanstalk 개요

Beanstalk 는 AWS에 어플리케이션을 배포하는 개발자 중심의 관점을 제공.

하나의 인터페이스 화면에서 EC2, S3, ASG, ELB, RDS 등 일반적으로

어플리케이션 배포시 필요한 아키텍처를 자동으로 포함하여 알아서 배포해줌.

관리형 서비스이기 때문에 용량 프로비저닝, 로드밸런서 구성, 스케일링,

어플리케이션 상태 모니터링, 인스턴스 구성등을 자동으로 처리해준다.

따라서 개발자는 코드 자체에만 신경 쓰면 됨.

어플리케이션을 업데이트하는 방법도 제공.

Beanstalk 자체는 무료이지만 ASG 또는 ELB에서 활용하는 인스턴스에 대해서는 요금을 지불해야 된다.

다양한 프로그램 언어와 버전 관리 기능 제공.

Beanstalk 를 생성하며 자동으로 EC2 가 생성되고 필요시 RDS 연동도 가능.

자동으로 탄력적 IP, 오토스케일링 그룹등이 설정됨.

웹브라우저에서 도메인이름으로 접근이 가능.

어플리케이션을 업로드하면 자동으로 EC2에 배포됨.

2. SpringBoot + MyBaits 배포

1) Beanstalk에서 작업하기

(수업시 생성했던 EC2, RDS, S3, ELB, Auto Scaling 은 모두 삭제부터 하자)

가. Beanstalk 위한 IAM 의 역할생성 및 권한 설정

IAM > 액세스 관리 > 역할 > 역할 생성 버튼 클릭



AWS 서비스 체크하고 서비스 또는 사용 사례에서 **EC2** 선택하고 다음 클릭.

신뢰할 수 있는 엔터티 선택
정보

신뢰할 수 있는 엔터티 유형

☒ **AWS 서비스**
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **AWS 계정**
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **웹 자격 증명**
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ **SAML 2.0 연동**
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ **사용자 지정 신뢰 정책**
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

사용 사례
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

서비스 또는 사용 사례

EC2

지정된 서비스에 대한 사용 사례를 선택합니다.
사용 사례

☒ **EC2**
Allows EC2 instances to call AWS services on your behalf.

☐ **EC2 Role for AWS Systems Manager**
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

☐ **EC2 Spot Fleet Role**
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

☐ **EC2 - Spot Fleet Auto Scaling**
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

☐ **EC2 - Spot Fleet Tagging**
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

☐ **EC2 - Spot Instances**
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

☐ **EC2 - Spot Fleet**
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

☐ **EC2 - Scheduled Instances**
Allows EC2 Scheduled Instances to manage instances on your behalf.

취소 다음

권한 추가 화면에서 다음 3가지 권한을 체크한다.

AWSElasticBeanstalkMulticontainerDocker

AWSElasticBeanstalkWebTier

AWSElasticBeanstalkWorkerTier

권한 추가 정보

권한 정책 (3/957) 정보

새 역할에 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

×

☐ 모든 유형

45 개 일치

정책 이름	유형
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS 관리형
<input type="checkbox"/> AWSEC2VssSnapshotPolicy	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkCustomPlatformforEC2Role	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkEnhancedHealth	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy	AWS 관리형
<input checked="" type="checkbox"/> AWSElasticBeanstalkMulticontainerDocker	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkReadOnly	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleCore	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleCWL	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleECS	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleRDS	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleSNS	AWS 관리형
<input type="checkbox"/> AWSElasticBeanstalkRoleWorkerTier	AWS 관리형
<input checked="" type="checkbox"/> AWSElasticBeanstalkWebTier	AWS 관리형
<input checked="" type="checkbox"/> AWSElasticBeanstalkWorkerTier	AWS 관리형

역할 이름은 임의로 지정하고 역할 생성 버튼을 클릭한다.

역할 이름: aws-elasticbeanstalk-ec2-role

이름 지정, 검토 및 생성

역할 세부 정보

학명

이 역할을 식별하는 의미 있는 이름을 입력합니다.

aws-elasticbeanstalk-ec2-role

최대 64자입니다. 영숫자 및 '+', '@', '-' 문자를 사용하세요.

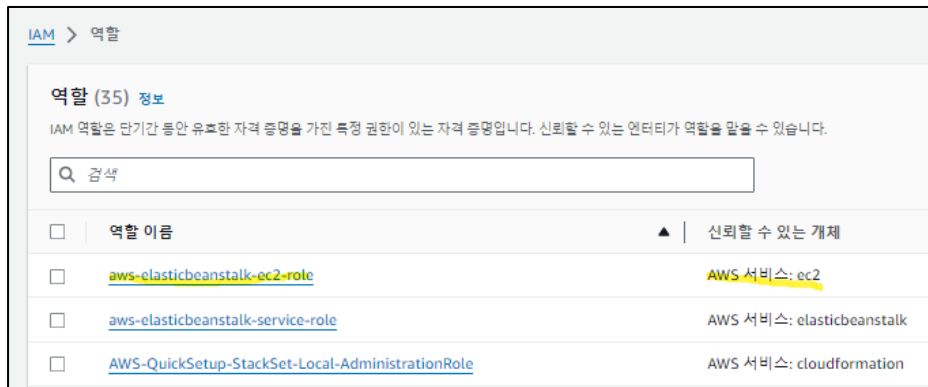
설명

이 역할에 대하여 간단한 설명을 추가합니다.

Allows EC2 instances to call AWS services on your behalf.

최대 문자 수: 1000. 문자(A~Z 및 a~z), 숫자(0~9), 밑줄, 새 줄, 다음 문자를 하나만 사용합니다. +, -, @, /([)]#\$%^&*~;

최종 실행 결과는 다음과 같음.



나. Beanstalk 접근 사용자 생성 및 그룹에 사용자 추가

IAM > 사용자 > 사용자 생성 > 직접 정책 연결 > "AdministratorAccess-AWSElasticBeanstalk" 선택.

사용자 생성을 완료했으면 해당 사용자에게 들어가서 액세스 키를 만들어야 한다.

만들고 나면 액세스키와 비밀번호를 받게 되는데, 이를 저장해두고 Github Actions 와 연동할 때 사용.



정책 이름은 AdministratorAccess-AWSElasticBeanstalk 를 선택.

권한 설정

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 직무별로 사용자의 권한을 관리하려면 그룹을 사용하는 것이 좋습니다. [자세히 알아보기](#)

권한 옵션

☐ 그룹에 사용자 추가

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.

☐ 권한 복사

기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

☒ 직접 정책 연결

관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

권한 정책 (1/1246)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.



정책 생성

필터링 기준 유형

검색

모든 유형

< 1 2 3 4 5 6 7 ... 63 >



정책 이름



유형



AccessAnalyzerServiceRolePolicy

AWS 관리형



AdministratorAccess

AWS 관리형 - 직무



AdministratorAccess-Amplify

AWS 관리형



AdministratorAccess-AWSElasticBeanstalk

AWS 관리형



AdministratorAccessOnlyCalifornia

고객 관리형

검토 및 생성

선택 사항을 검토합니다. 사용자를 생성한 후 자동 생성된 암호를 보고 다운로드할 수 있습니다(활성화된 경우).

사용자 세부 정보

사용자 이름

demo-user-beanstalk

콘솔 암호 유형

None

암호 재설정 필요

아니요

권한 요약

< 1 >

이름



유형



다음과 같이 사용



AdministratorAccess-AWSElasticBeanstalk

AWS 관리형

권한 정책

태그 - 선택 사항

태그는 리소스를 식별, 구성 또는 검색하는 데 도움이 되도록 AWS 리소스에 추가할 수 있는 키 값 페어입니다. 이 사용자와 연결할 태그를 선택합니다.

리소스와 연결된 태그가 없습니다.

새 태그 추가

최대 50개의 태그를 더 추가할 수 있습니다.

취소

이전

사용자 생성

생성된 사용자 체크 > 보안 자격 증명 > 액세스 키와 보안 키 생성한다.

콘솔 로그인

콘솔 액세스 활성화

콘솔 로그인 링크

 <https://285502508151.signin.aws.amazon.com/console>

콘솔 암호


활성화되지 않음

멀티 팩터 인증(MFA) (0)

제거

재동기화

MFA 디바이스 할당

MFA를 사용하여 AWS 환경의 보안을 강화합니다. MFA로 로그인하려면 MFA 디바이스의 인증 코드가 필요합니다. 각 사용자는 MFA 디바이스를 최대 8개까지 할당할 수 있습니다. [자세히 알아보기](#) 

유형

식별자

인증


생성 날짜


MFA 디바이스가 없습니다. MFA 디바이스를 할당하여 AWS 환경 보안 개선하기

MFA 디바이스 할당

엑세스 키 (0)

엑세스 키 만들기

엑세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 엑세스 키(활성 또는 비활성)를 가질 수 있습니다. [자세히 알아보기](#) 

엑세스 키가 없습니다. 엑세스 키와 같은 장기 보안 인증을 사용하지 않는 것이 모범 사례입니다. 대신 단기 보안 인증을 제공하는 도구를 사용하세요. [자세히 알아보기](#) 

엑세스 키 만들기

액세스 키 모범 사례 및 대안 정보

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

사용 사례

☐ **Command Line Interface(CLI)**

AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **로컬 코드**

로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **AWS 컴퓨팅 서비스에서 실행되는 애플리케이션**

Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ **서드 파티 서비스**

AWS 리소스를 모니터링 또는 관리하는 서드 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☒ **AWS 외부에서 실행되는 애플리케이션**

이 액세스 키를 사용하여 AWS 리소스에 액세스해야 하는 AWS 외부의 데이터 센터 또는 기타 인프라에서 실행 중인 워크로드를 인증할 것입니다.

☐ **기타**

귀하의 사용 사례가 여기에 나열되어 있지 않습니다.



권장되는 대안

IAM Roles Anywhere를 사용하여 AWS 서비스에 액세스하는 비 AWS 워크로드에 대한 임시 보안 인증을 생성할 수 있습니다. [AWS 이외의 워크로드에 대한 액세스를 제공하는 방법에 대해 자세히 알아보세요.](#)

취소

다음

설명 태그 설정 - 선택 사항 정보

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

설명 태그 값

이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

demo-beanstalk-accesskey

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 _ . : / = + - @입니다.

취소

이전

액세스 키 만들기

액세스키: AKIAUE6KGRR33CNVU43J

비밀키: xaJiD1+JKF3EYBogB3ob3Jusr82m//URyNQ****

액세스 키 검색 정보

액세스 키

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키

비밀 액세스 키

AKIAUE6KGRR33CNVU43J

xaJiD1+JKF3EYBogB3ob3Jusr82m//URyNQ**** [숨기기](#)

액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안 됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

.csv 파일 다운로드

완료

Admin 과 Student 그룹에 사용자 추가하기

IAM > 사용자 > demo-user-beanstalk

demo-user-beanstalk 정보

요약

ARN
arn:aws:iam::285502508151:user/demo-user-beanstalk

생성됨
October 05, 2024, 18:34 (UTC+09:00)

콘솔 액세스
비활성화됨

마지막 콘솔 로그인
-

권한	그룹 (2)	태그 (2)	보안 자격 증명	Last Accessed
사용자 그룹 멤버십 사용자 그룹은 IAM 사용자의 할당선입니다. 그룹을 사용하여 사용자 할당선에 대한 권한을 지정할 수 있습니다. 사용자는 한 번에 최대 그룹 10개의 멤버가 될 수 있습니다.				
<input type="checkbox"/>	그룹 이름			연결된 정책
<input type="checkbox"/>	Admin	추가할 것		AdministratorAccess
<input type="checkbox"/>	Student			studentAdmin 및 AdministratorAccess

다. 어플리케이션 생성

Elastic Beanstalk > Application > 어플리케이션 생성 버튼 클릭.

Elastic Beanstalk

Elastic Beanstalk > Applications

어플리케이션 (0) 정보

표시 값과 일치하는 결과 필터링

1

어플리케이션 이름

환경

어플리케이션 없음
표시할 어플리케이션이 없음

어플리케이션 생성

환경 구성 정보

환경 티어 정보

Amazon Elastic Beanstalk에는 서로 다른 유형의 웹 애플리케이션을 지원하는 두 가지 유형의 환경 티어가 있습니다.

☒ 웹 서버 환경

HTTP 요청을 처리하는 웹 사이트, 웹 애플리케이션 또는 웹 API를 실행합니다. [자세히 알아보기](#)

☐ 작업자 환경

요청 시 장기 실행 워크로드를 처리하거나 일정에 따라 작업을 수행하는 작업자 애플리케이션을 실행합니다. [자세히 알아보기](#)

애플리케이션 정보 정보

애플리케이션 이름

TodoBootApplication

최대 길이는 100자입니다.

▶ 애플리케이션 태그(선택 사항)

환경 정보 정보

환경의 이름, 하위 도메인 및 설명을 선택합니다. 나중에 변경할 수 없습니다.

환경 이름

TodoBootApplication-env

길이는 4~40자여야 합니다. 이름은 문자, 숫자 및 하이픈만 포함할 수 있습니다. 하이픈으로 시작하거나 끝날 수 없습니다. 이 이름은 계정의 리전 내에서 고유해야 합니다.

도메인

todobootapp .ap-northeast-2.elasticbeanstalk.com

가용성 확인

✓ todobootapp.ap-northeast-2.elasticbeanstalk.com 사용 가능

환경 설명

플랫폼 정보

플랫폼 유형

- ☒ 관리형 플랫폼
Amazon Elastic Beanstalk에서 계서 및 유지 관리하는 플랫폼입니다. [자세히 알아보기](#)
- ☐ 사용자 지정 플랫폼
사용자가 생성하고 소유한 플랫폼입니다. 플랫폼이 없는 경우 이 옵션을 사용할 수 없습니다.

플랫폼

Java Java

플랫폼 브랜치

Corretto 17 running on 64bit Amazon Linux 2

Corretto 17 running on 64bit Amazon Linux 2

플랫폼 버전

3.8.0 (Recommended)

애플리케이션 코드 정보

- ☒ 샘플 애플리케이션
업로드한 애플리케이션 버전입니다.
- ☐ 기존 버전
- ☐ 코드 업로드
컴퓨터에서 소스 번들을 업로드하거나 Amazon S3에서 소스 번들을 복사합니다.

사전 설정 정보

사용 사례와 일치하는 사전 설정에서 시작하거나 사용자 지정 구성을 선택하여 권장 값을 설정 해제하고 서비스의 기본값을 사용합니다.

구성 사전 설정

- ☒ 단일 인스턴스(프리 티어 사용 가능)
- ☐ 단일 인스턴스(스팟 인스턴스 사용)
- ☐고가용성
- ☐고가용성(스팟 및 온디맨드 인스턴스 사용)
- ☐ 사용자 지정 구성

취소

다음



만약 다음의 기존 서비스 역할에 해당되는 `aws-elasticbeanstalk-service-role` 이 없으면 명시적으로 생성해야 됨

IAM > 역할 > 역할 생성 > AWS 서비스 체크 및 사용사례는 Elastic Beanstalk 선택 >

AWS Elastic Beanstalk Enhanced Health

AWS Elastic Beanstalk Managed Updates Customer Role Policy

////////////////////////////////////

서비스 액세스

Elastic Beanstalk에서 서비스 역할로 담당할 IAM 역할 및 EC2 인스턴스 프로파일을 통해 Elastic Beanstalk가 환경을 생성하고 관리할 수 있습니다. IAM 역할과 인스턴스 프로파일 모두 필요한 권한이 포함된 IAM 관리형 정책에 연결되어야 합니다. [자세히 알아보기](#)

서비스 역할

☐ 새 서비스 역할 생성 및 사용
☒ 기존 서비스 역할 사용

기존 서비스 역할

Elastic Beanstalk가 서비스 역할로 담당할 기존 IAM 역할을 선택합니다. 기존 IAM 역할에는 필요한 IAM 관리형 정책이 있어야 합니다.

EC2 키 페어

EC2 키 페어를 선택하여 EC2 인스턴스에 안전하게 로그인합니다. [자세히 알아보기](#)

EC2 인스턴스 프로파일

EC2 인스턴스가 필요한 작업을 수행하도록 허용하는 관리형 정책이 있는 IAM 인스턴스 프로파일을 선택합니다.

네트워킹 화면에서는 퍼블릭 IP 주소 활성화로 설정하고
인스턴스 서브넷은 모두 체크 한다.

네트워킹, 데이터베이스 및 태그 설정 - 선택 사항 정보

Virtual Private Cloud(VPC)

VPC

기본 VPC 대신 사용자 지정 VPC에서 환경을 시작합니다. VPC 관리 콘솔에서 VPC와 서브넷을 생성할 수 있습니다. 자세히 알아보기 [\[?\]](#)

vpc-0a88dc44239219ef5 | (10.0.0.0/16) | demo-vpc ▼

[사용자 지정 VPC 생성](#) [\[?\]](#)

인스턴스 설정

애플리케이션을 실행하는 인스턴스의 각 AZ에서 서브넷을 선택합니다. 인스턴스가 인터넷에 노출되지 않도록 하려면 프라이빗 서브넷에서 인스턴스를 실행하고 퍼블릭 서브넷에서 로드 밸런서를 실행합니다. 로드 밸런서와 인스턴스를 동일한 퍼블릭 서브넷에서 실행하려면 퍼블릭 IP 주소를 인스턴스에 할당합니다. 자세히 알아보기 [\[?\]](#)

퍼블릭 IP 주소

환경의 Amazon EC2 인스턴스에 퍼블릭 IP 주소를 할당합니다.

☒ 활성화됨

인스턴스 서브넷

<input checked="" type="checkbox"/>	가용 영역	서브넷 ▲	CIDR	이름
<input checked="" type="checkbox"/>	ap-northeast-2b	subnet-03519a5ce...	10.0.144.0/20	demo-subnet-priv...
<input checked="" type="checkbox"/>	ap-northeast-2b	subnet-03a775489...	10.0.16.0/20	demo-subnet-publ...
<input checked="" type="checkbox"/>	ap-northeast-2a	subnet-0490639ca...	10.0.0.0/20	demo-subnet-publ...
<input checked="" type="checkbox"/>	ap-northeast-2a	subnet-0e622ab9e...	10.0.128.0/20	demo-subnet-priv...

데이터베이스 화면에서는 데이터베이스 활성화로 설정하고 서브넷은 모두 선택한다.

데이터베이스 정보

RDS SQL 데이터베이스를 환경과 통합합니다. 자세히 알아보기

데이터베이스 서브넷

Elastic Beanstalk 환경이 Amazon RDS에 연결되어 있는 경우 데이터베이스 인스턴스의 서브넷을 선택합니다. 자세히 알아보기

데이터베이스 서브넷 선택 (4)

데이터베이스 서브넷 필터링

<input checked="" type="checkbox"/>	가용 영역	서브넷	CIDR	이름
<input checked="" type="checkbox"/>	ap-northeast-2b	subnet-03519a5ce...	10.0.144.0/20	demo-subnet-priv...
<input checked="" type="checkbox"/>	ap-northeast-2b	subnet-03a775489...	10.0.16.0/20	demo-subnet-publ...
<input checked="" type="checkbox"/>	ap-northeast-2a	subnet-0490639ca...	10.0.0.0/20	demo-subnet-publ...
<input checked="" type="checkbox"/>	ap-northeast-2a	subnet-0e622ab9e...	10.0.128.0/20	demo-subnet-priv...

☒ 데이터베이스 활성화

사용자 이름: admin

암호: Passw0rd1234

데이터베이스 설정

환경의 데이터베이스에 대한 엔진 및 인스턴스 유형을 선택합니다.

엔진

mysql ▼

엔진 버전

8.0.35 ▼

인스턴스 클래스

db.t3.small ▼

스토리지

5GB에서 1024GB 사이의 숫자를 선택합니다.

5

GB

사용자 이름

admin

admin

암호

.....

PasswOrd1234

가용성

낮음(AZ 1개) ▼

데이터베이스 삭제 정책

이 정책은 데이터베이스를 분리하거나 데이터베이스와 결합된 환경을 종료할 때 적용됩니다.

☐ 스냅샷 생성

Elastic Beanstalk는 데이터베이스의 스냅샷을 저장한 다음 삭제합니다. Elastic Beanstalk 환경에 DB를 추가하거나 독립 실행형 데이터베이스를 생성할 때 스냅샷에서 데이터베이스를 복원할 수 있습니다. 데이터베이스 스냅샷 저장에 대한 비용이 발생할 수 있습니다.

☐ 유지

분리된 데이터베이스는 Elastic Beanstalk 외부에서 사용 가능하고 작동 가능한 상태로 유지됩니다.

☒ 삭제

Elastic Beanstalk가 데이터베이스를 종료합니다. 데이터베이스를 더 이상 사용할 수 없습니다.

루트 볼륨 유형: 범용3(SSD) 선택하고 최소 크기인 10GB로 설정

인스턴스 트래픽 및 크기 조정 구성 - 선택 사항 정보

▼ 인스턴스 정보

애플리케이션을 실행하는 Amazon EC2 인스턴스를 구성합니다.

루트 볼륨(부팅 디바이스)

루트 볼륨 유형

범용3(SSD)

크기

각 인스턴스에 연결된 루트 볼륨의 기가바이트 수입니다.

10

GB

IOPS

프로비저닝된 IOPS(SSD)볼륨의 초당 입력/출력 작업 수.

3000

IOPS

처리량

환경의 EC2 인스턴스에 연결된 Amazon EBS 루트 볼륨을 프로비저닝하는 데 필요한 처리량

125

MiB/s

Amazon CloudWatch 모니터링

EC2 인스턴스에서 지표가 보고되는 시점 사이의 시간 간격

모니터링 간격

5 minute

인스턴스 메타데이터 서비스(IMDS)

환경의 플랫폼은 IMDSv1과 IMDSv2를 모두 지원합니다. IMDSv2를 적용하려면 IMDSv1을 비활성화합니다. [자세히 알아보기](#)

IMDSv1

현재 설정을 사용하면 환경이 IMDSv2만 활성화합니다.

☒ 비활성화됨

EC2 보안 그룹

트래픽을 제어할 보안 그룹을 선택합니다.

EC2 보안 그룹 (2)

Q 보안 그룹 필터링

☒

그룹 이름

▲

그룹 ID

▼

이름

☒

default

sg-094895c93ffabbeb0

☒

launch-wizard-1

sg-036da95527f7b1587

인스턴스 메타데이터 서비스(IMDS)

환경의 플랫폼은 IMDSv1과 IMDSv2를 모두 지원합니다. IMDSv2를 적용하려면 IMDSv1을 비활성화합니다. [자세히 알아보기](#)

IMDSv1

현재 설정을 사용하면 환경이 IMDSv2만 활성화합니다.

☒ 비활성화됨

EC2 보안 그룹

트래픽을 제어할 보안 그룹을 선택합니다.

EC2 보안 그룹 (1)

☒

그룹 이름

▲

그룹 ID

▼

이름

▼

☒

default,

sg-041285cdd983e69a1

▼ 용량 정보

환경의 컴퓨팅 파워와 오토 스케일링 설정을 구성하여 사용되는 인스턴스 수를 최적화합니다.

기본설정으로 지정

오토 스케일링 그룹

환경 유형

단일 인스턴스 또는 로드 밸런싱된 환경을 선택합니다. 단일 인스턴스 환경에서 애플리케이션을 개발 및 테스트하여 비용을 절감한 다음 애플리케이션이 프로덕션에 투입할 준비가 되면 로드 밸런싱된 환경으로 업그레이드할 수 있습니다. [자세히 알아보기](#)

단일 인스턴스 ▼

인스턴스

1 최솟값

1 최댓값

업데이트, 모니터링 및 로깅 구성 - 선택 사항 정보

▼ 모니터링 정보

이후의 설정은 기본설정으로 지정

상태 보고

강화된 상태 보고는 환경의 인스턴스 및 기타 리소스에 대한 애플리케이션 및 운영 체제 모니터링을 무료로 제공합니다. **EnvironmentHealth** 사용자 지정 지표는 강화된 상태 보고에서 무료로 제공됩니다. 각 사용자 지정 지표에는 추가 요금이 적용됩니다. 자세한 내용은 [Amazon CloudWatch 요금](#) 섹션을 참조하세요.

시스템

- ☐ 기본
- ☒ 강화됨

CloudWatch 사용자 지정 지표 - 인스턴스

지표 선택

CloudWatch 사용자 지정 지표 - 환경

지표 선택

상태 이벤트를 CloudWatch Logs로 스트리밍

환경 상태 이벤트를 CloudWatch Logs로 스트리밍하도록 Elastic Beanstalk을 구성합니다. 보존 기간을 최대 10년까지 설정하고 환경을 종료할 때 로그를 삭제하도록 Elastic Beanstalk을 구성할 수 있습니다.

로그 스트리밍

☐ 활성화됨(표준 CloudWatch 요금이 적용됩니다.)

보존

7

수명 주기

환경 종료 후 로그 유지

다음의 환경 속성까지 기본설정으로 지정함.

환경 속성

다음 속성은 애플리케이션에서 환경 속성으로 전달됩니다. 자세히 [알아보기](#)

이름

GRADLE_HOME

값

/usr/local/gradle

제거

M2

/usr/local/apache-maven/bin

제거

M2_HOME

/usr/local/apache-maven

제거

환경 속성 추가

취소

이전

다음

검토 정보

1단계: 환경 구성

기본설정으로 지정

편집

환경 정보

환경 티어

웹 서버 환경

환경 이름

TodoBootApplication-env

플랫폼

arn:aws:elasticbeanstalk:ap-northeast-2::platform/Corretto 17 running on 64bit Amazon Linux 2023/4.2.7

애플리케이션 이름

TodoBootApplication

애플리케이션 코드

샘플 애플리케이션

가장 마지막에 [제출] 버튼 클릭.

꽤 많은 시간이 걸림.

Elastic Beanstalk가 환경을 시작하고 있습니다. 이 작업은 몇 분 정도 걸립니다.

Elastic Beanstalk > 환경 > TodoBootApplication-env

TodoBootApplication-env 정보

작업 ▼

업로드 및 배포

환경 개요

상태

⌚ Pending

도메인

TodoBootApplication.ap-northeast-2.elasticbeanstalk.com

환경 ID

📄 e-6u37sgqzzy

애플리케이션 이름

TodoBootApplication

작업 진행 상황은 다음과 같이 이벤트 항목을 참고한다.

다음 항목에서 에러가 없어야 됨.

이벤트	상태	로그	모니터링	경보	관리형 업데이트	태그
이벤트 (7) 정보						
Q 텍스트, 속성 또는 값을 기준으로 이벤트를 필터링						
시간	▼	유형	세부 정보			
10월 6, 2024 10:44:55 (UTC+9)		① INFO	Created EIP: 3.37.103.215			
10월 6, 2024 10:44:39 (UTC+9)		① INFO	Creating RDS database named: awseb-e-w76uhzvuda-stack-awsebrds			
10월 6, 2024 10:44:39 (UTC+9)		① INFO	Created security group named: sg-019063cb154326bca			
10월 6, 2024 10:44:39 (UTC+9)		① INFO	Created security group named: sg-07eed5dae00f283da			
10월 6, 2024 10:44:34 (UTC+9)		① INFO	Environment health has transitioned to Pending. Initialization in prog			
10월 6, 2024 10:44:12 (UTC+9)		① INFO	Using elasticbeanstalk-ap-northeast-2-285502508151 as Amazon S3			
10월 6, 2024 10:44:10 (UTC+9)		① INFO	createEnvironment is starting.			

성공적으로 시작이 되었으면 도메인을 복사해서 웹 브라우저에 요청한다.

🔴 환경이 성공적으로 시작되었습니다.

Elastic Beanstalk > 환경 > TodoBootApplication-env

TodoBootApplication-env 정보

🔄 작업 업로드 및 배포

환경 개요

상태
🟢 Ok

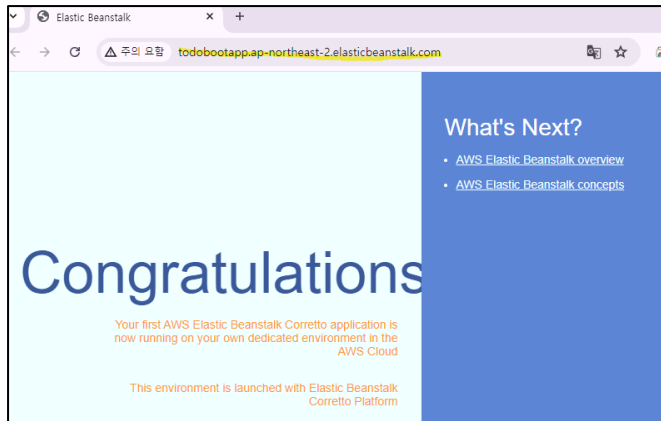
도메인
TodoBootApplication.ap-northeast-2.elasticbeanstalk.com

환경 ID
e-6u37sgqzzy

애플리케이션 이름
TodoBootApplication

샘플 어플리케이션이 다음과 같이 출력됨.

<http://todobootapp.ap-northeast-2.elasticbeanstalk.com/>



////////////////////////////////////

- Beanstalk에 의해서 자동 설정된 서비스 확인하기

EC2 서비스에 가면 다음과 같이 자동으로 EC2 인스턴스가 생성되어 제공됨.

Amazon S3 서비스에 가면 다음과 같이 자동으로 버킷이 생성되어 제공됨.

계정 스냅샷 - 24시간마다 업데이트

모든 AWS 리전

Storage Lens 대시보드 보기

Storage Lens는 스토리지 사용량 및 활동 추세에 대한 가시성을 제공합니다. [자세히 알아보기](#)

범용 버킷

디렉터리 버킷

범용 버킷 (1) 정보

모든 AWS 리전

↺

ARN 복사

비어 있음

삭제

버킷 만들기

범용 버킷 (1) 정보

모든 AWS 리전

버킷은 S3에 저장되는 데이터의 컨테이너입니다.

이름으로 버킷 찾기

<

1

>

⚙

이름	AWS 리전	IAM Access Analyzer	생성 날짜
<div>elasticbeanstalk-ap-northeast-2-285502508151</div>	아시아 태평양(서울) ap-northeast-2	ap-northeast-2에 대한 분석기 보기	2024. 8. 1. am 8:55:19 AM KST

#####

Beanstalk 제거시 S3 의 버킷은 자동으로 제거 안됨.

버킷 > 권한 > 버킷 정책 에서 S3:DeleteBucket의 Effect 값을 Deny에서 Allow로 변경해야 삭제가 가능해짐.

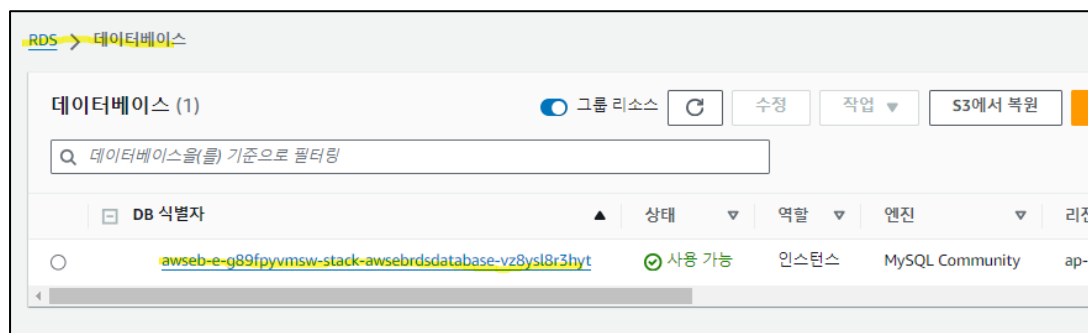

```

    },
    {
      "Sid": "eb-58950a8c-feb6-11e2-89e0-0800277d041b",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:DeleteBucket",
      "Resource": "arn:aws:s3:::elasticbeanstalk-ap-northeast-2-285502508151"
    }
  ]
}

```

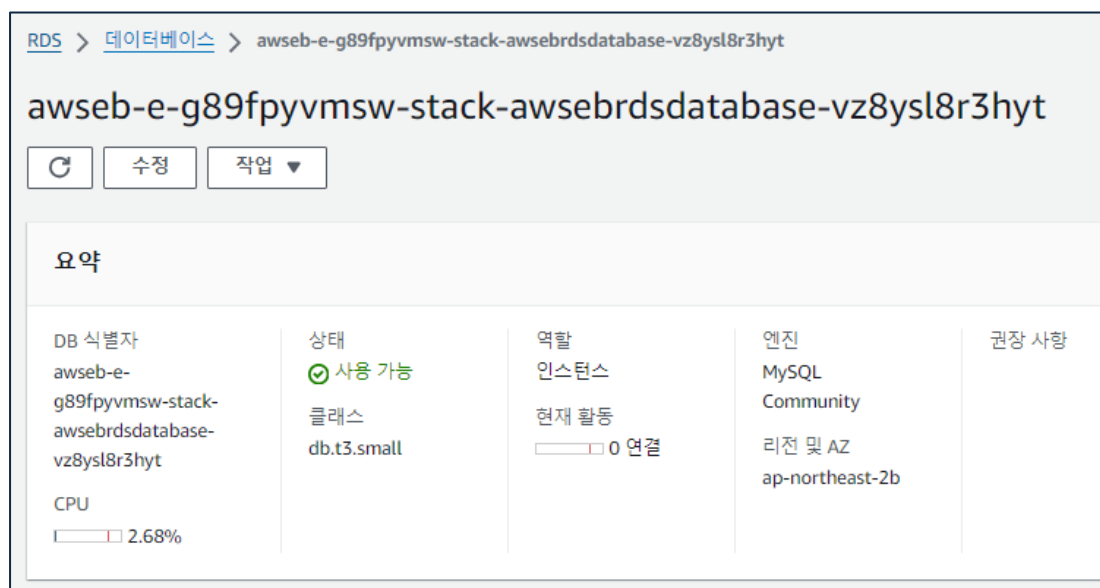
#####

Amazon RDS 서비스에 가면 다음과 같이 생성되어 제공됨



외부(Workbench등)에서 접근을 허용하기 위해서 설정값을 수정한다.

다음 화면에서 [수정]버튼을 선택한다.



연결 > 추가 구성 에서 퍼블릭 액세스 가능하도록 선택한다.

G

연결

네트워크 유형 정보

듀얼 스택 모드를 사용하려면 IPv6 CIDR 블록을 지정한 VPC의 서브넷과 연결해야 합니다.

☒ **IPv4**
리소스는 IPv4 주소 지정 프로토콜을 통해서만 통신할 수 있습니다.

☐ **듀얼 스택 모드**
리소스는 IPv4, IPv6 또는 둘 모두를 통해 통신할 수 있습니다.

DB 서브넷 그룹

aws-e-9g89fpyvmw-stack-awsebrdsdbsubnetgroup-pdk9dxnngwjv ▼

보안 그룹

이 DB 인스턴스와 연결할 DB 보안 그룹 목록입니다.

보안 그룹 선택 ▼

aws-e-9g89fpyvmw-stack-AWSEBRDSDBSecurityGroup-N1dPhN4aJ7pz ✕

인증 기관 정보

서버 인증서를 사용하면 Amazon 데이터베이스에 대한 연결이 이루어지고 있는지 검증하여 추가 보안 계층을 제공합니다. 프로비저닝하는 모든 데이터베이스에 자동으로 설치되는 서버 인증서를 확인하여 이를 수행합니다.

rds-ca-rsa2048-g1 (기본값) ▼
만료: May 21, 2061

추가 구성

퍼블릭 액세스

☒ 퍼블릭 액세스 가능

RDS는 데이터베이스에 퍼블릭 IP 주소를 할당합니다. VPC 외부의 Amazon EC2 인스턴스 및 다른 리소스가 데이터베이스에 연결할 수 있습니다. VPC 내부의 리소스도 데이터베이스에 연결할 수 있습니다. 데이터베이스에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

☐ 퍼블릭 액세스 불가능

DB 인스턴스에 할당된 IP 주소가 없습니다. VPC 외부의 EC2 인스턴스 및 디바이스는 연결할 수 없습니다.

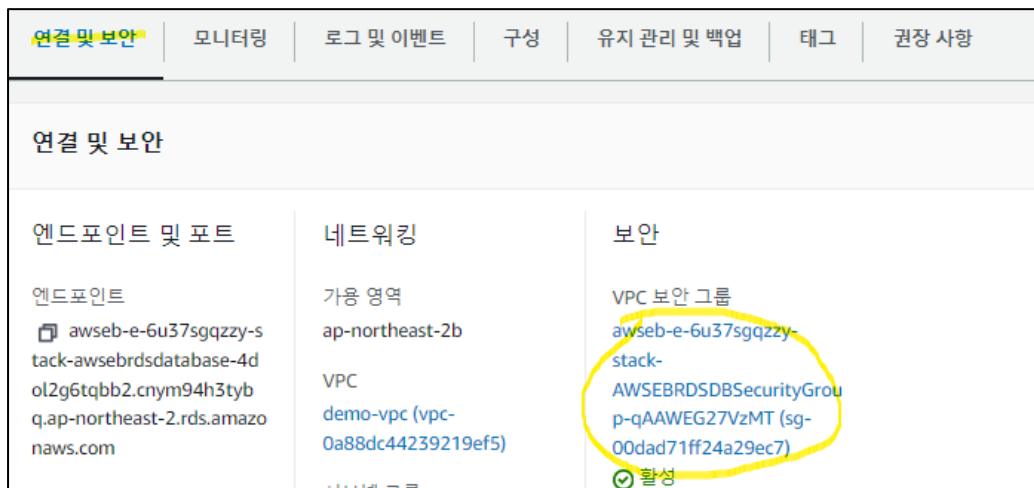
데이터베이스 포트

DB 인스턴스가 애플리케이션 연결에 사용할 TCP/IP 포트를 지정합니다. 애플리케이션 연결 문자열에서 포트 번호를 지정해야 합니다. DB 보안 그룹과 방화벽은 이 포트에 대한 연결을 허용해야 합니다. 자세히 알아보기 [\[?\]](#)

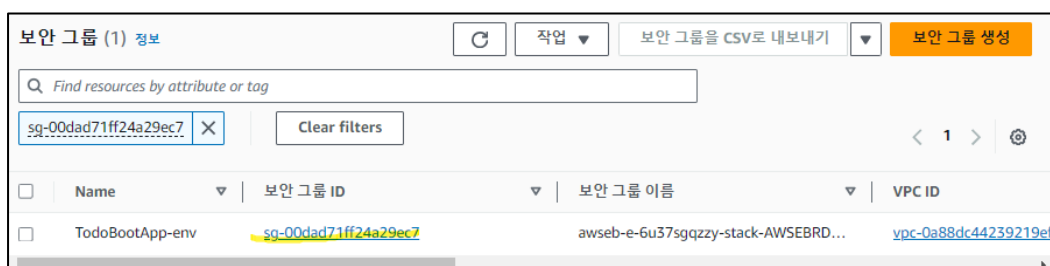
3306



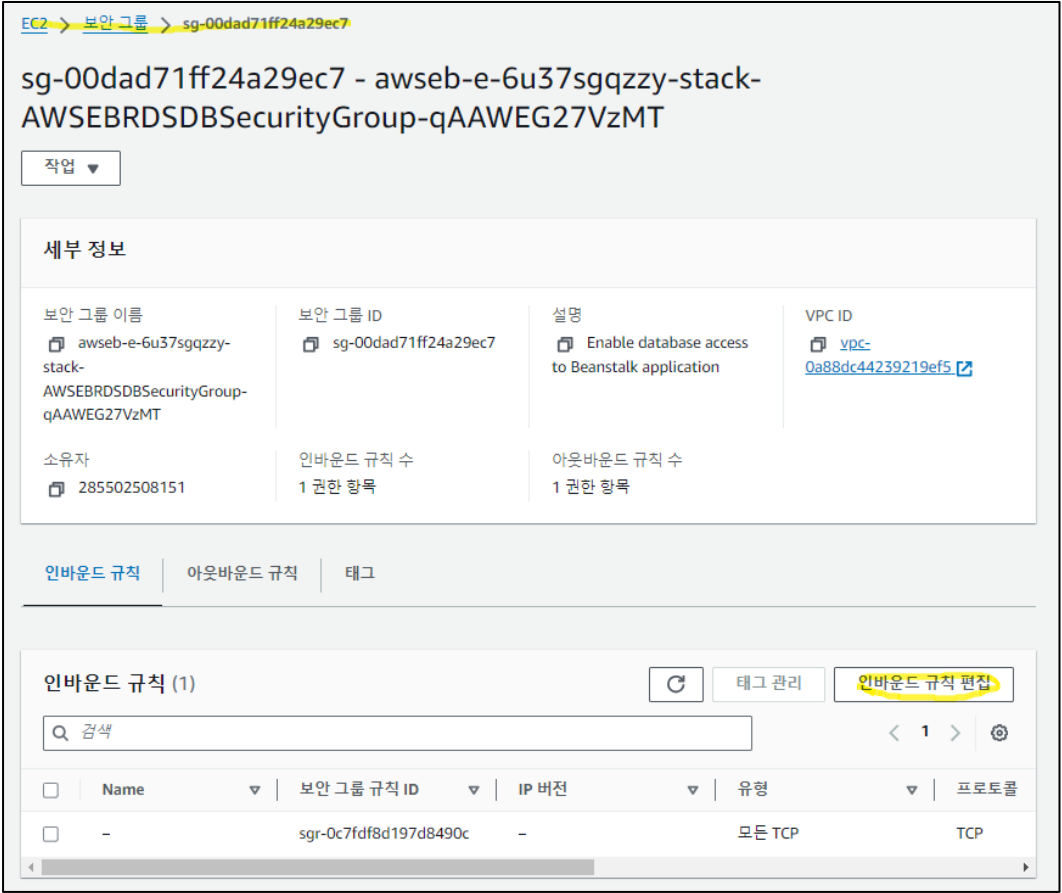
연결 및 보안 > VPC 보안 그룹 클릭하기.



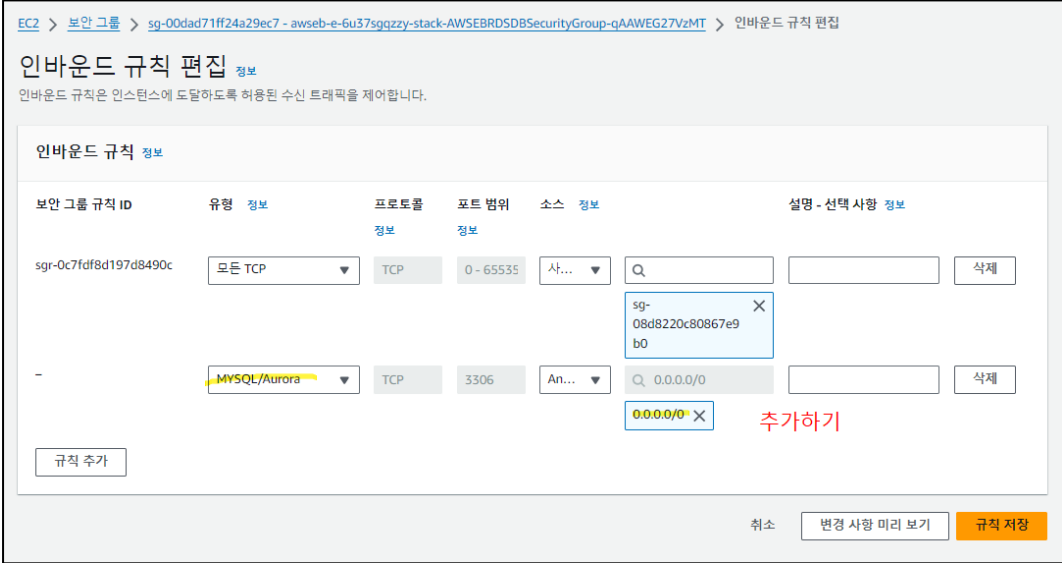
보안 그룹 ID 클릭하기



인바운드 규칙 편집 버튼 클릭하기

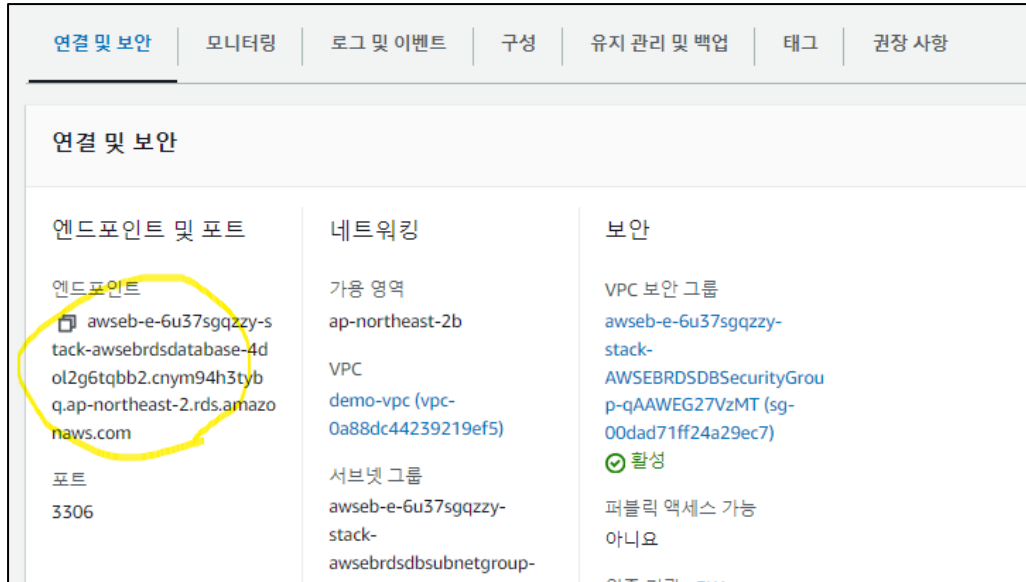


규칙 추가 버튼 클릭 > 유형에서 MySQL/Aurora 선택 및 0.0.0.0/0 추가.

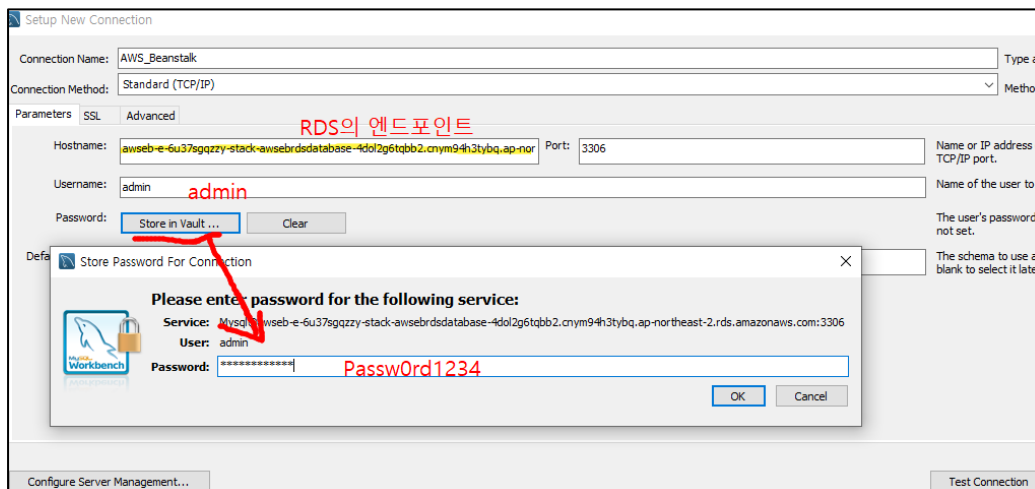


workbench에서 접속해보자.

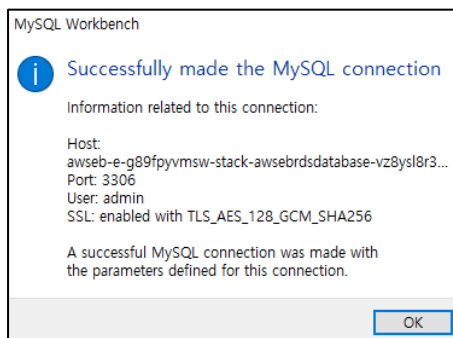
먼저 RDS에 가서 엔드포인트를 복사한다.



workbench에서 다음과 같이 정보를 설정한다.

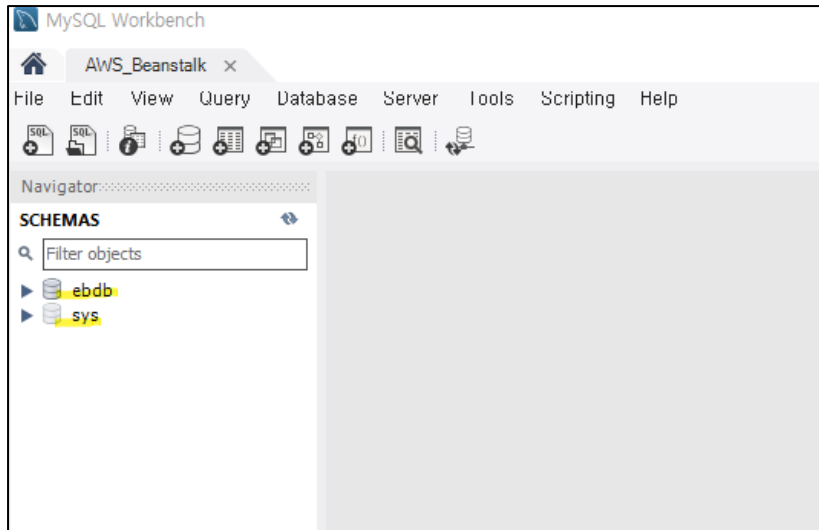


Test Connection 버튼을 클릭하면 다음과 같이 Success 화면이 보임.



beanstalk에서 생성된 RDS의 데이터베이스는 기본적으로 다음과 같다.

필요시 데이터베이스를 명시적으로 생성해서 사용한다.



- RDS에 testdb 및 테이블 생성하기

```
use testdb;
```

```
drop table if exists todo;
```

```
drop table if exists member;
```

```
create table member(
```

```
    userid varchar(255) primary key COMMENT '아이디',
```

```
    passwd varchar(255) not null    COMMENT '비밀번호',
```

```
    username varchar(255) not null  COMMENT 'TODO 작성자'
```

```
);
```

```
create table todo
```

```
( id bigint not null auto_increment COMMENT 'TODO 번호',
```

```
  userid varchar(255) not null COMMENT 'TODO 아이디',
```

```
  description varchar(255) not null COMMENT 'TODO 목록',
```

```
  targetDate date COMMENT 'TODO 목표날짜',
```

```
  done boolean COMMENT 'TODO 완료여부',
```

```
  primary key(id)
```

```
);
```

```
insert into todo ( id, userid, description, targetDate, done ) values ( 1000,'inky4832', 'Learn  
SpringBoot2.1.8', DATE_ADD(NOW(), INTERVAL 1 YEAR), false);
```

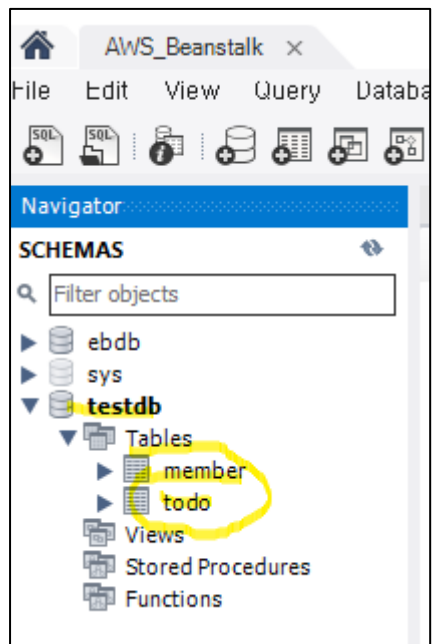
```
insert into todo ( id, userid, description, targetDate, done ) values ( 1001,'inky4832', 'Learn  
Reactjs', DATE_ADD(NOW(), INTERVAL 1 MONTH), false);
```

```
insert into todo ( id, userid, description, targetDate, done ) values ( 1002,'inky4832', 'Learn  
SpringSecurity', DATE_ADD(NOW(), INTERVAL 10 DAY), false);
```

```
insert into todo ( id, userid, description, targetDate, done ) values ( 200,'kim4832', 'Learn Dance',  
DATE_ADD(NOW(), INTERVAL 3 YEAR), false);
```

```
commit;
```

최종 실행결과는 다음과 같다.



2) SpringBoot 에서 확인할 사항

pom.xml 파일에서 <packaging> 및 <build> 추가할것.

```
<artifactId>boot_reactjs</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<packaging>jar</packaging>
```

....=

```
<build>
```

```
<finalName>todo</finalName>
```

```
<plugins>
```

```
<plugin>
```

```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-maven-plugin</artifactId>
```

```
</plugin>
```

```
</plugins>
```

```
<resources>
```

```
<resource>
```

```
<directory>src/main/resources</directory>
```

```
<filtering>true</filtering>
```

```
<includes>
```

```
<include>**/application*.properties</include>
```

```
</includes>
```

```
</resource>
```

```
<resource>
```

```
<directory>src/main/java</directory>
```

```
<includes>
```

```
<include> **/*.properties </include>
```

```
<include> **/*.xml </include>
```

```
</includes>
```

```
</resource>
```

```
</resources>
```

```
</build
```

application.properties 파일에서 확인

```
server.port=5000
```

```
server.servlet.context-path=/todo
```


```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.datasource.url=jdbc:mysql://beanstalk의RDB엔드포인트: 3306/testdb
```

```
spring.datasource.username=admin
```

```
spring.datasource.password=Passw0rd1234
```

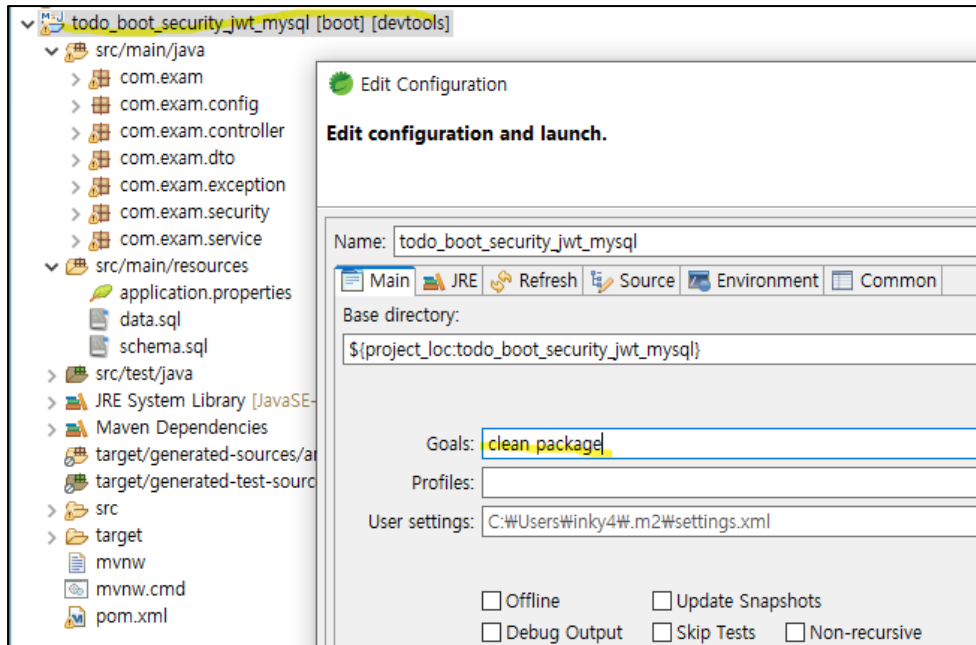
먼저 RDS의 엔드포인트를 복사해서 SpringBoot의 url에 지정해야 된다.

연결 및 보안	모니터링	로그 및 이벤트	구성
연결 및 보안			
엔드포인트 및 포트		네트워킹	
엔드포인트 복사하기  awseb-e-g89fpvmsw-stack-awsebrdsdatabase-vz8ysl8r3hyt.cnym94h3tybq.ap-northeast-2.rds.amazonaws.com		가용 영역 ap-northeast-2b	
포트 3306		VPC demo-vpc (vpc-0a88dc44239219ef5)	
		서브넷 그룹 awseb-e-g89fpvmsw-stack-	

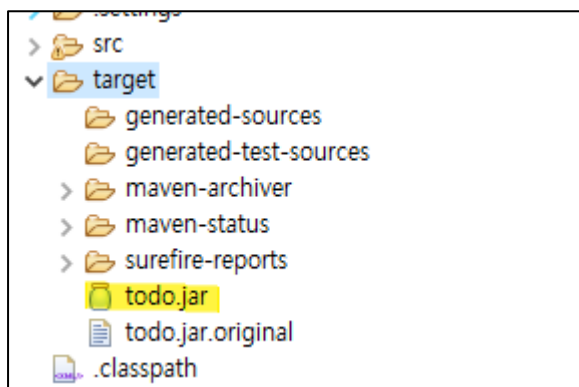
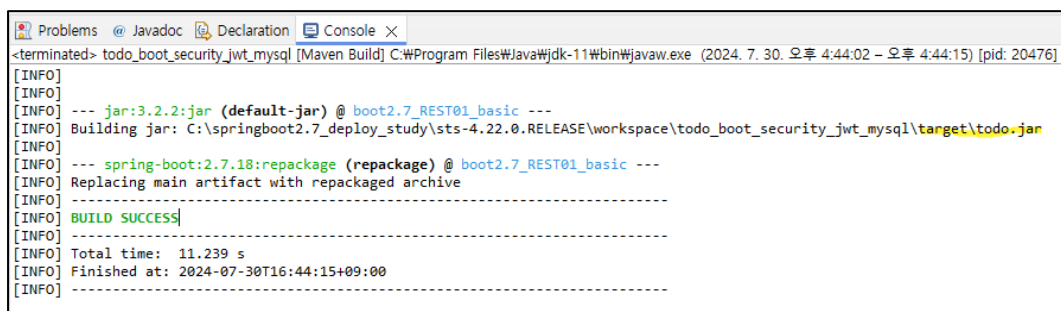
```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://awseb-e-g89fpvmsw-stack-awsebrdsdatabase-vz8ysl8r3hyt.cnym94h3tybq.ap-northeast-2.rds.amazonaws.com:3306/testdb
spring.datasource.username=admin
spring.datasource.password=Passw0rd1234
```

maven 이용하여 jar로 패키징하기

프로젝트 선택 > 오른쪽 클릭 > Run As... > Maven build... 선택

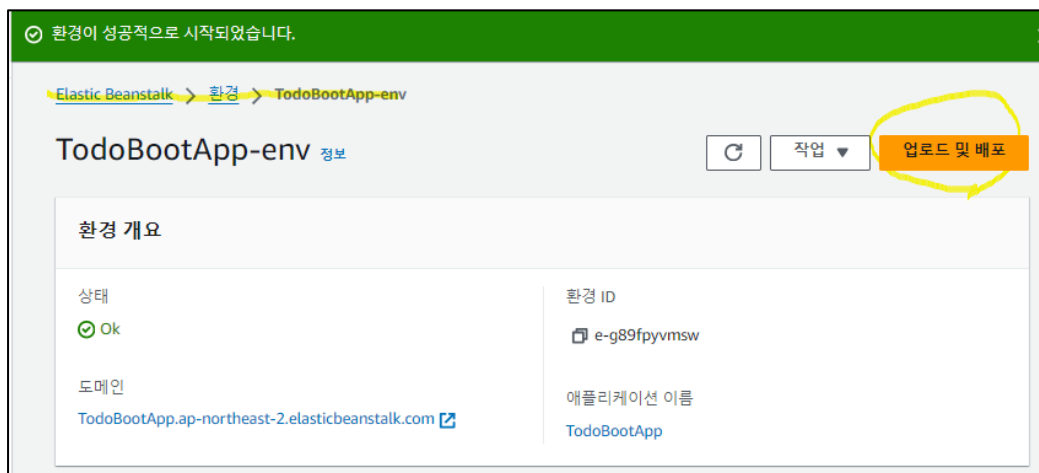


실행결과는 다음과 같이 todo.jar 파일이 생성됨. 나중에 todo.jar 파일을 Beanstalk 에서 업로드하면 배포 완료.



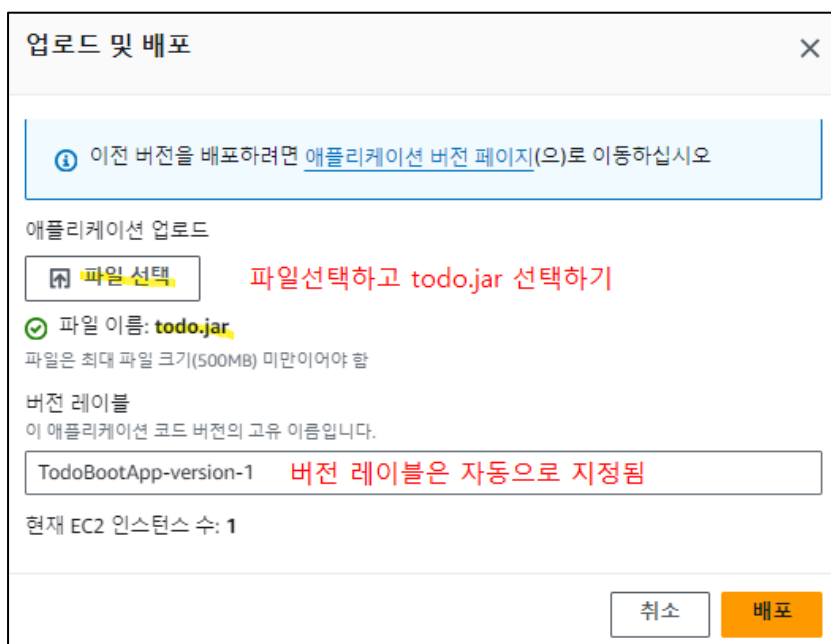
Beanstalk에 todo.jar 업로드 하기

Elastic Beanstalk > 환경 > TodoBootApplication-env > 업로드 및 배포 선택하기

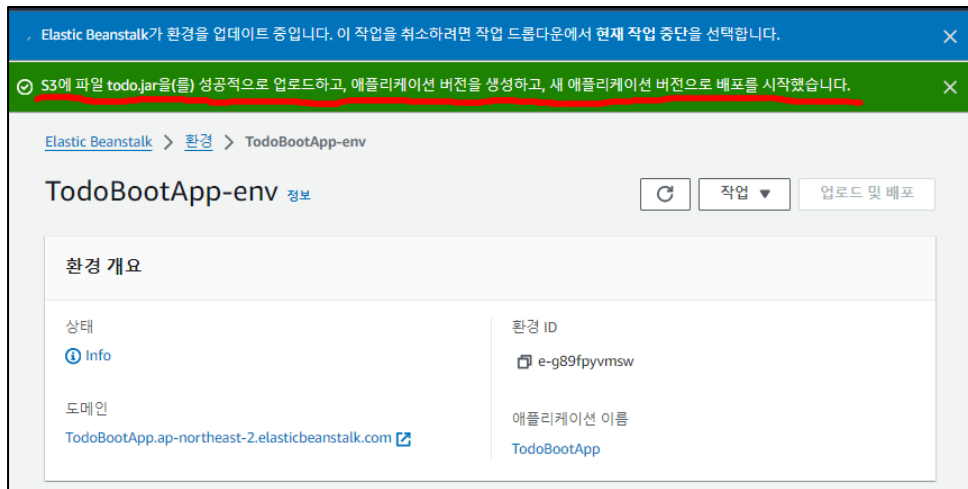


업로드할 todo.jar 파일을 선택하면 자동으로 버전 레이블이 생성됨.

만약 todo.jar 파일을 다시 업로드하면 version-2 가 자동으로 생성됨.

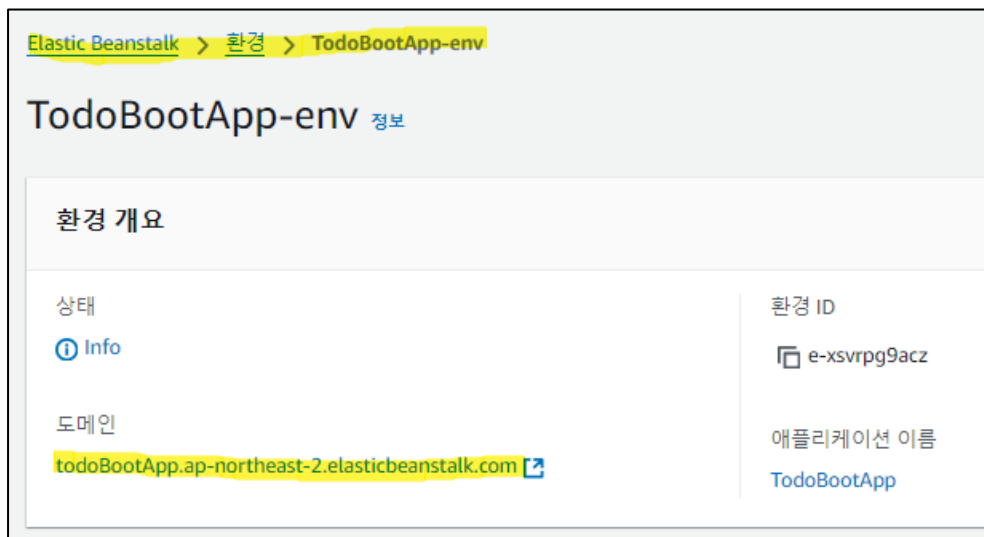


최종적으로 [배포] 버튼을 클릭하면 다음과 같이 배포중임을 확인 할 수 있다.



Talend API 에서 요청하자.

Elastic Beanstalk > 환경 > TodoBootApp-env 에서 도메인 복사해서 다음과 같이 URL을 생성한다.



위의 도메인은 나중에 React.js에서 Boot 요청시 사용될 수 있음.

GET : <http://todobootapp.ap-northeast-2.elasticbeanstalk.com/todo/home>

The screenshot shows a REST client interface with a tab labeled "01_hello". The request method is "GET" and the URL is <http://todobootapp.ap-northeast-2.elasticbeanstalk.com/todo/home>. The response status is "200". The response headers include "Server: nginx", "Date: Tue, 08 Apr 2025 01:41:06 GMT +1s", and "Content-Type: text/plain; charset=UTF-8". The response body is "home".

회원 가입 요청하기

POST: <http://todobootapp.ap-northeast-2.elasticbeanstalk.com/todo/signup>

The screenshot shows a REST client interface with a tab labeled "02_signup". The request method is "POST" and the URL is <http://todobootapp.ap-northeast-2.elasticbeanstalk.com/todo/signup>. The request headers include "Content-Type: application/json". The request body is a JSON object:

```
{  "userid": "kim4832",  "passwd": "1234",  "username": "김유신"}
```

. The response status is "200".

{

"userid":"kim4832",

"passwd":"1234",

"username":"김유신"

}

실행결과는 다음과 같이 201 응답되고 RDS에 새로운 행이 생성됨

The screenshot shows a database management interface. On the left, the 'SCHEMAS' panel displays a tree view with 'testdb' expanded, showing tables 'member' and 'todo'. The main area displays a SQL query: `1 • SELECT * FROM testdb`. Below the query, the 'Result Grid' shows the following data:

	userid	passwd	username
▶	kim4832	\$2a\$10\$41cFQ39l...	김유신
*	NULL	NULL	NULL

3) Beanstalk 에 https 보안 프로토콜 적용하기

가. ELB 생성

EC2 > 로드 밸런서 > 로드 밸런서 생성 버튼 클릭 > ALB 생성

- [EC2](#)
- >
- [로드 밸런서](#)
- >
- Application Load Balancer 생성

① 이제 Application Load Balancer가 퍼블릭 IPv4 IP 주소 관리(IPAM)를 지원합니다.

네트워크 매핑 섹션에서 IP 풀을 구성하여 이 기능을 시작할 수 있습니다.

Application Load Balancer 생성 정보

Application Load Balancer는 수신 HTTP 및 HTTPS 트래픽을 요청 속성을 기반으로 Amazon EC2 인스턴스, 마이크로서비스 및 컨테이너와 같은 여러 대상에 배포합니다. 로드 밸런서를 결정할 다음 해당되는 경우, 대상 그룹에서 규칙 작업의 대상을 선택합니다.

▶ Application Load Balancer의 작동 방식

기본 구성

로드 밸런서 이름

이름은 AWS 계정 내에서 고유해야 하며 로드 밸런서 생성 후에는 변경할 수 없습니다.

☒ DemoALB
 ☐ DemoALB

하이픈을 포함하여 최대 32자의 영숫자 문자를 사용할 수 있지만 이름이 하이픈으로 시작하거나 끝나지 않아야 합니다.

체계 | 정보

로드 밸런서 생성 후에는 스키마를 변경할 수 없습니다.

☒ 인터넷 연결

• 인터넷 경계 트래픽을 처리합니다.
 • 퍼블릭 IP 주소가 있습니다.
 • DNS 이름은 공개적으로 확인할 수 있습니다.
 • 퍼블릭 서브넷이 필요합니다.

☐ 내부

• 내부 트래픽을 처리합니다.
 • 프라이빗 IP 주소가 있습니다.
 • DNS 이름은 공개적으로 확인할 수 없습니다.
 • IPv4 및 듀얼 스택 IP 주소 유형과 호환됩니다.

로드 밸런서 IP 주소 유형 | 정보

로드 밸런서에 할당할 프란트엔드 IP 주소 유형을 선택합니다. 이 로드 밸런서에 매핑된 VPC 및 서브넷에는 선택한 IP 주소 유형이 포함되어야 합니다. 퍼블릭 IPv4 주소에는 추가 비용이 부과됩니다.

☒ IPv4

IPv4 주소만 포함합니다.

☐ 듀얼 스택

IPv4 및 IPv6 주소를 포함합니다.

☐ 퍼블릭 IPv4가 없는 듀얼 스택

퍼블릭 IPv6 주소와 프라이빗 IPv4 및 IPv6 주소를 포함합니다. 인터넷 연결 로드 밸런서와만 호환됩니다.

네트워크 매핑 정보

로드 밸런서는 IP 주소 설정에 따라 선택한 서브넷의 대상으로 트래픽을 라우팅합니다.

VPC | 정보

로드 밸런서는 선택한 VPC 내에서 존재하고 확장됩니다. 또한 선택한 VPC는 Lambda 또는 온프레미스 대상으로 라우팅하거나 VPC 피어링을 사용하는 경우를 제외하고 로드 밸런서 대상을 로스팅해야 합니다.

VPC를 생성 [?] 하세요.

demo-vpc

vpc-081fbc63cfea64016

IPv4 VPC CIDR: 10.0.0.0/16

IP 풀 - 신규 정보

선택적으로 IPAM 풀을 로드 밸런서 IP 주소의 기본 소스로 구성하도록 선택할 수 있습니다. [Amazon VPC IP 주소 관리자 콘솔](#) [?]에서 풀을 생성하거나 확인합니다.

☐ 퍼블릭 IPv4 주소에 IPAM 풀 사용

선택한 IPAM 풀이 퍼블릭 IPv4 주소의 기본 소스가 됩니다. 풀이 고갈되면 AWS에서 IPv4 주소를 할당합니다.

가용 영역 및 서브넷 | 정보

가용 영역을 2개 이상 선택하고 각 영역에 대해 서브넷을 선택합니다. 로드 밸런서 노드는 선택한 각 영역에 배치되며 트래픽에 따라 자동으로 확장됩니다. 로드 밸런서는 선택한 가용 영역의 대상으로만 배포됩니다.

☒ **ap-northeast-2a (apne2-az1)**

서브넷

로드 밸런서 IP 주소 유형에 해당하는 CIDR 블록만 사용됩니다. 로드 밸런서를 효율적으로 확장하려면 사용 가능한 IP 주소가 8개 이상 필요합니다.

subnet-05067f7c757ea20c3

IPv4 서브넷 CIDR: 10.0.128.0/20

☒ **ap-northeast-2b (apne2-az2)**

서브넷

로드 밸런서 IP 주소 유형에 해당하는 CIDR 블록만 사용됩니다. 로드 밸런서를 효율적으로 확장하려면 사용 가능한 IP 주소가 8개 이상 필요합니다.

subnet-0096fd879ede5e45c

IPv4 서브넷 CIDR: 10.0.16.0/20

보안 그룹 항목에서는 새 보안 그룹을 생성 클릭해서 새로 만들자.

(새로운 창이 열림)

보안 그룹 정보
보안 그룹은 로드 밸런서에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성할 수 있습니다.

보안 그룹 클릭

최대 5개의 보안 그룹 선택

default
sg-094895c93ffabbeb0 VPC: vpc-081fbe63cfea64016

새로운 보안그룹을 만들고 http와 https 트래픽 허용하는 것으로 설정한다

보안 그룹 생성 정보
보안 그룹은 인바운드 및 아웃바운드 트래픽을 관리하는 인스턴스의 가상 방화벽 역할을 합니다. 새 보안 그룹을 생성하려면 아래의 필드를 작성하십시오.

기본 세부 정보

보안 그룹 이름 정보
demo-sg-load-balancer demo-sg-load-balancer
생성 후에는 이름을 편집할 수 없습니다.

설명 정보
demo-sg-load-balancer

VPC 정보
vpc-081fbe63cfea64016 (demo-vpc) demo-vpc 선택

인바운드 규칙 정보

유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명 - 선택 사항 정보
HTTP HTTP 선택	TCP	80	Anywhere-IPv4 0.0.0.0/0	
HTTPS HTTPS 선택	TCP	443	Anywhere-IPv4 0.0.0.0/0	

규칙 추가

이제 보안그룹을 로드 밸런서에 배정해야 된다. (이전 창에서 작업)

(새 보안 그룹을 생성했으면 새로고침 하여 새로운 보안 그룹을 선택한다.)

보안 그룹 정보
보안 그룹은 로드 밸런서에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성할 수 있습니다.

보안 그룹 클릭하여 demo-sg-load-balancer 보안그룹을 선택

최대 5개의 보안 그룹 선택

demo-sg-load-balancer
sg-0ae10379fad7bc3e VPC: vpc-081fbe63cfea64016

리스너 및 라우팅 항목에서는 대상그룹을 정해야 되는데 대상은 EC2 인스턴스임.

현재는 대상그룹이 없기 때문에 대상그룹을 새로 생성하자

리스너 및 라우팅 정보

리스너는 사용자가 구성된 포트 및 프로토콜을 사용하여 연결 요청을 검사하는 프로세스입니다. 리스너에 대해 정의한 규칙에 따라 로드 밸런서가 등록된 대상으로 요청을 라우팅하는 방법이 결정됩니다.

▼ 리스너 HTTP:80

제거

프로토콜

포트

기본 작업 정보

HTTP

:

80

다음으로 전달: 대상 그룹 선택

↻

1-65535

대상 그룹 생성

그룹 세부 정보 지정

로드 밸런서는 요청을 대상 그룹의 대상으로 라우팅하고 대상에 대한 상태 확인을 수행합니다.

기본 구성

대상 그룹이 생성된 후에는 이 섹션의 설정을 변경할 수 없습니다.

대상 유형 선택

☒ 인스턴스

- 특정 VPC 내의 인스턴스에 대한 로드 밸런싱을 지원합니다.
- [Amazon EC2 Auto Scaling](#)을 사용하여 EC2 용량을 관리하고 크기를 조정할 수 있습니다.

☐ IP 주소

- VPC 및 온프레미스 리소스에 대한 로드 밸런싱을 지원합니다.
- 동일한 인스턴스에 있는 여러 IP 주소 및 네트워크 인터페이스로의 라우팅을 지원합니다.
- 마이크로서비스 기반 아키텍처를 통한 유연성을 제공하여 애플리케이션 간 통신을 간소화합니다.
- IPv6 대상을 지원하여 종단 간 IPv6 통신 및 IPv4에서 IPv6로의 NAT를 활성화합니다.

☐ Lambda 함수

- 단일 Lambda 함수로 라우팅을 지원합니다.
- Application Load Balancer에만 액세스할 수 있습니다.

☐ Application Load Balancer

- Network Load Balancer가 특정 VPC 내에서 TCP 요청을 수락하고 라우팅할 수 있는 유연성을 제공합니다.
- Application Load Balancer로 고정 IP 주소 및 PrivateLink를 손쉽게 사용할 수 있습니다.

대상 그룹 이름

demo-tg-alb

demo-tg-alb

하이픈을 포함하여 최대 32자의 영숫자 문자를 사용할 수 있지만 이름이 하이픈으로 시작하거나 끝나지 않아야 합니다.

다음 화면 대상등록에서 '아래에 보류 중인 것으로 포함' 버튼을 클릭.

대상 등록
이는 대상 그룹을 생성하기 위한 선택적 단계입니다. 그러나 로드 밸런서가 이 대상 그룹으로 트래픽을 라우팅하려면 대상을 등록해야 합니다.

사용 가능한 인스턴스 (1/1)

인스턴스 필터링

<input checked="" type="checkbox"/>	인스턴스 ID	이름	상태	보안 그룹	영역
<input checked="" type="checkbox"/>	i-05cfed417ff9edab6	TodoBootApp-env	실행 중	default, awseb-e-c3acnryhpb-stack-A...	ap-northeast-2a

먼저 선택하고..

1개 선택됨

선택한 인스턴스를 위한 포트
선택한 인스턴스로 트래픽을 라우팅하기 위한 포트입니다.

80

1-65535(임포트 여러 포트 구분)

아래에 보류 중인 것으로 포함

클릭

버튼을 클릭하면 다음과 같이 대상에 포함됨. 대상 그룹 생성 버튼 클릭.

대상 보기

대상 (1)

대상 필터링

대기 중인 항목만 보기

보류 중인 모든 항목 제거

인스턴스 ID	이름	포트	상태	보안 그룹	영역
i-033a6ca5917dec95f	TodoBootApp-env	80	실행 중	default, awseb-e-w76uhzvuda-stack-AWSEBSecurityGroup-IVVCcOuBMylK	ap-northeast-2

1개 대기 중

취소

이전

대상 그룹 생성

다음은 로드 밸런서 추가 화면의 리스너 및 라우팅 화면에서 작업한다.

새로고침 클릭하고 이전에 방금 생성한 보안그룹(demo-tg-alb)을 선택한다

리스너 및 라우팅 정보
리스너는 사용자가 구성한 포트 및 프로토콜을 사용하여 연결 요청을 검사하는 프로세스입니다. 리스너에 대해 정의한 규칙에 따라 로드 밸런서가 등록된 대상으로 요청을 라우팅하는 방법이 결정됩니다.

▼ 리스너 HTTP:80

프로토콜 : HTTP : 포트 : 80
1-65535

기본 작업 | 정보
다음으로 전달: 대상 그룹 선택
대상 그룹 생성

리스너 태그 - 선택 사항
리스너에 태그를 추가하는 것을 고려하십시오. 태그를 사용하면 AWS 리소스를 분류하여 좀 더 쉽게 관리할 수 있습니다.

리스너 태그 추가
최대 50개의 태그를 더 추가할 수 있습니다.

리스너 추가

demo-tg-alb
대상 유형: 인스턴스, IPv4
선택

기본으로 제공되는 http 이외의 https 용 리스너를 추가한다.

리스너 및 라우팅 정보
리스너는 사용자가 구성한 포트 및 프로토콜을 사용하여 연결 요청을 검사하는 프로세스입니다. 리스너에 대해 정의한 규칙에 따라 로드 밸런서가 등록된 대상으로 요청을 라우팅하는 방법이 결정됩니다.

▼ 리스너 HTTP:80

프로토콜 : HTTP : 포트 : 80
1-65535

기본 작업 | 정보
다음으로 전달: demo-tg-alb
대상 유형: 인스턴스, IPv4
대상 그룹 생성

리스너 태그 - 선택 사항
리스너에 태그를 추가하는 것을 고려하십시오. 태그를 사용하면 AWS 리소스를 분류하여 좀 더 쉽게 관리할 수 있습니다.

리스너 태그 추가
최대 50개의 태그를 더 추가할 수 있습니다.

제거

▼ 리스너 HTTPS:443 리스너 추가 버튼을 클릭하여 https 용 리스너를 추가한다

프로토콜 : HTTPS : 포트 : 443
1-65535

기본 작업 | 정보
다음으로 전달: demo-tg-alb
대상 유형: 인스턴스, IPv4
대상 그룹 생성

리스너 태그 - 선택 사항
리스너에 태그를 추가하는 것을 고려하십시오. 태그를 사용하면 AWS 리소스를 분류하여 좀 더 쉽게 관리할 수 있습니다.

리스너 태그 추가
최대 50개의 태그를 더 추가할 수 있습니다.

제거

리스너 추가

보안 리스너 설정에서는 인증서(ACM에서)를 선택해서 수업시간에 만든 Route53 도메인을 선택한다. (예> ssg-kdt.click)

보안 리스너 설정 정보

이러한 설정은 모든 보안 리스너에 적용됩니다. 생성된 후에 리스너별로 이러한 설정을 관리할 수 있습니다.

보안 정책 | 정보

로드 밸런서는 보안 정책이라고 하는 Secure Socket Layer(SSL) 협상 구성을 사용해 클라이언트와의 SSL 연결을 관리합니다. [보안 정책 비교](#)

보안 카테고리 정책 이름

모든 보안 정책

ELBSecurityPolicy-TLS13-1-2-2021-06 (권장)

기본 SSL/TLS 서버 인증서

클라이언트가 SNI 프로토콜 없이 연결하거나 일치하는 인증서가 없는 경우에 사용되는 인증서입니다. 이 인증서를 AWS Certificate Manager(ACM), Amazon Identity and Access Management(IAM)에서 소싱하거나 인증서 가져오기를 실시할 수 있습니다. 이 인증서는 리스너 인증서 목록에 자동으로 추가됩니다.

인증서 소스

☒ ACM에서

☐ IAM에서

☐ 인증서 가져오기

인증서(ACM에서)

선택한 인증서는 이 로드 밸런서의 보안 리스너에 대한 기본 SSL/TLS 서버 인증서로 적용됩니다.

*.ssg-kdt.click
33c7b5b0-355e-42dc-933d-e5e47be6a489

[새 ACM 인증서 요청](#)

클라이언트 인증서 처리 | 정보

클라이언트 인증서는 인증된 요청을 원격 서버로 보내는 데 사용됩니다. [자세히 알아보기](#)

☐ 상호 인증(mTLS)

상호 전송 계층 보안(TLS) 인증은 양방향 피어 인증을 제공합니다. TLS를 통한 보안 계층을 추가하고 서비스에서 연결을 설정하는 클라이언트를 확인할 수 있도록 합니다.

이후는 기본설정으로 지정하고 마지막으로 [로드 밸런서 생성] 버튼을 클릭.

다음은 최종으로 설정된 화면이다.

처음에는 상태값이 '프로비저닝 중' 이라고 표시됨. 이후에 '활성'으로 바뀜.

(새로고침 버튼 클릭 필요)

DemoALB

로드 밸런서 유형
애플리케이션

상태
활성 **활성 상태**

VPC
vpc-081fbe63cfea64016

로드 밸런서 IP 주소 유형
IPv4

채계
Internet-facing

호스팅 영역
ZWKZPGT148KDX

가용 영역
subnet-05067f7c757ea20c3 ap-northeast-2a (apne2-az1)
subnet-0096fd879ede5e45c ap-northeast-2b (apne2-az2)

생성된 날짜
2025년 4월 8일, 11:32 (UTC+09:00)

로드 밸런서 ARN
arn:aws:elasticloadbalancing:ap-northeast-2:285502508151:loadbalancer/app/DemoALB/6d3e0f314ce169ce

DNS 이름 정보
DemoALB-1131565993.ap-northeast-2.elb.amazonaws.com (A 레코드)

DNS 복사

리스너 및 규칙

네트워크 매핑

리소스 맵

보안

모니터링

통합

속성

용량

태그

리스너 및 규칙 (2) 정보

리스너는 구성된 프로토콜 및 포트에서 연결 요청을 확인합니다. 리스너가 수신한 트래픽은 기본 작업 및 기타 추가 규칙에 따라 라우팅됩니다.

리스너 필터링

< 1 >

☐ 프로토콜: 포트

☐ 기본 작업

☐ 규칙

☐ ARN

☐ 보안 정책

☐ 기본 SSL/TLS 인증서

☐ mTLS

☐ HTTPS:443

대상 그룹으로 전달
demo-tg-alb 1 (100%)
대상 그룹 고정성: 끄

1개 규칙

ARN

ELBSecurityPolicy-TLS13-1-2-...

*.ssg-kdt.click(인증서 ID: 33c7...

끔

☐ HTTP:80

대상 그룹으로 전달
demo-tg-alb 1 (100%)
대상 그룹 고정성: 끄

1개 규칙

ARN

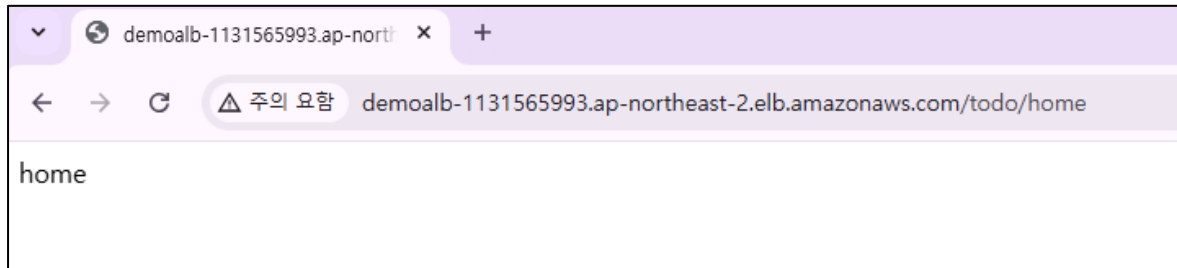
해당되지 않음

해당되지 않음

해당되지 않음

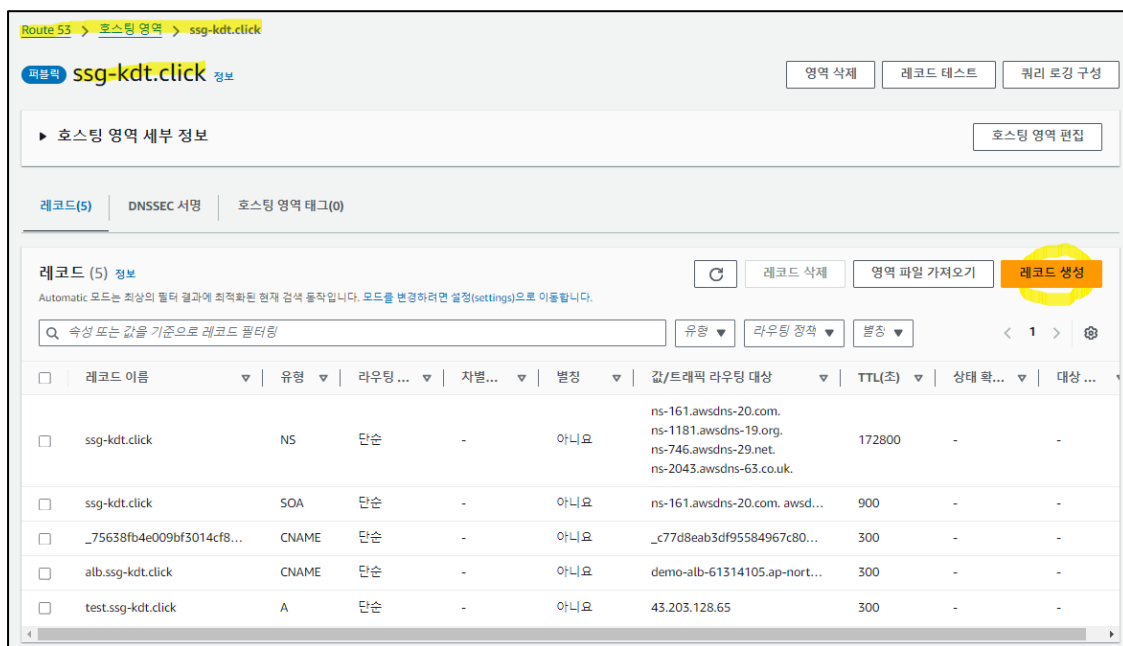
로드 밸런서가 생성되었으면 DNS 이름의 값을 복사해서 웹 브라우저에서 요청한다. (아직은 https 로 요청은 안됨)

http://demoALB-1131565993.ap-northeast-2.elb.amazonaws.com/todo/home



나. ELB와 Route 53 연동 (https 요청 가능)

Route 53 > 호스팅 영역 > ssg-kdt.click 선택해서 새로운 레코드를 생성하자.



레코드 생성 정보

빠른 레코드 생성

▼ 레코드 1

레코드 이름 정보 레코드 유형 정보 CNAME - 다른 도메인 이름과 일부 AWS 리소스로 트래픽 라우팅
루트 도메인에 대한 레코드를 생성하려면 비워 둡니다. **CNAME 선택**

☐ 별칭

값 정보 **ELB의 도메인**

별도의 줄에 여러 값을 입력합니다.

TTL(초) 정보 라우팅 정책 정보
권장 값: 60~172,800(2일)

생성후 다음과 같이 beanstalk.ssg-kdt.click 레코드가 생성된 것을 확인 가능.

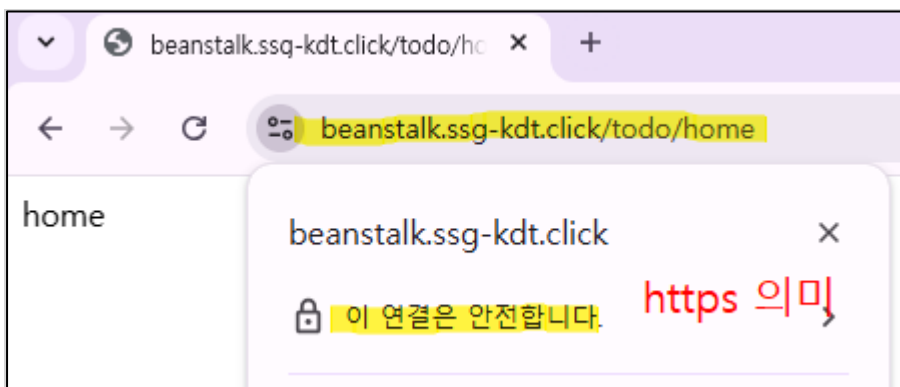
레코드 (6) 정보 레코드 삭제

Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. [모드를 변경하려면 설정\(settings\)으로 이동합니다](#)

🔍 속성 또는 값을 기준으로 레코드 필터링 유형 ▼ 라우팅 정책 ▼

<input type="checkbox"/>	레코드 이름	유형	라우팅 ...	자별...	별칭
<input type="checkbox"/>	ssg-kdt.click	NS	단순	-	아니요
<input type="checkbox"/>	ssg-kdt.click	SOA	단순	-	아니요
<input type="checkbox"/>	_75638fb4e009bf3014cf86d...	CNAME	단순	-	아니요
<input type="checkbox"/>	alb.ssg-kdt.click	CNAME	단순	-	아니요
<input checked="" type="checkbox"/>	beanstalk.ssg-kdt.click	CNAME	단순	-	아니요
<input type="checkbox"/>	test.ssg-kdt.click	A	단순	-	아니요

<https://beanstalk.ssg-kdt.click/todo/home> 요청하기



지금까지 AWS Beanstalk 이용해서 SpringBoot 어플리케이션을 수동으로 배포하는 방법이였음

4. GithubActions(CI/CD) 이용한 자동 배포

1) Github 에 저장소 생성하기

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

inky4832

 /

Repository name *

todo_boot


todo_boot

✓ todo_boot is available.

Great repository names are short and memorable. Need inspiration? How about [verbose-tribble](#) ?


Description (optional)

☒

 Public

Anyone on the internet can see this repository. You choose who can commit.

☐

 Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

README 파일 체크하지 말것.

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

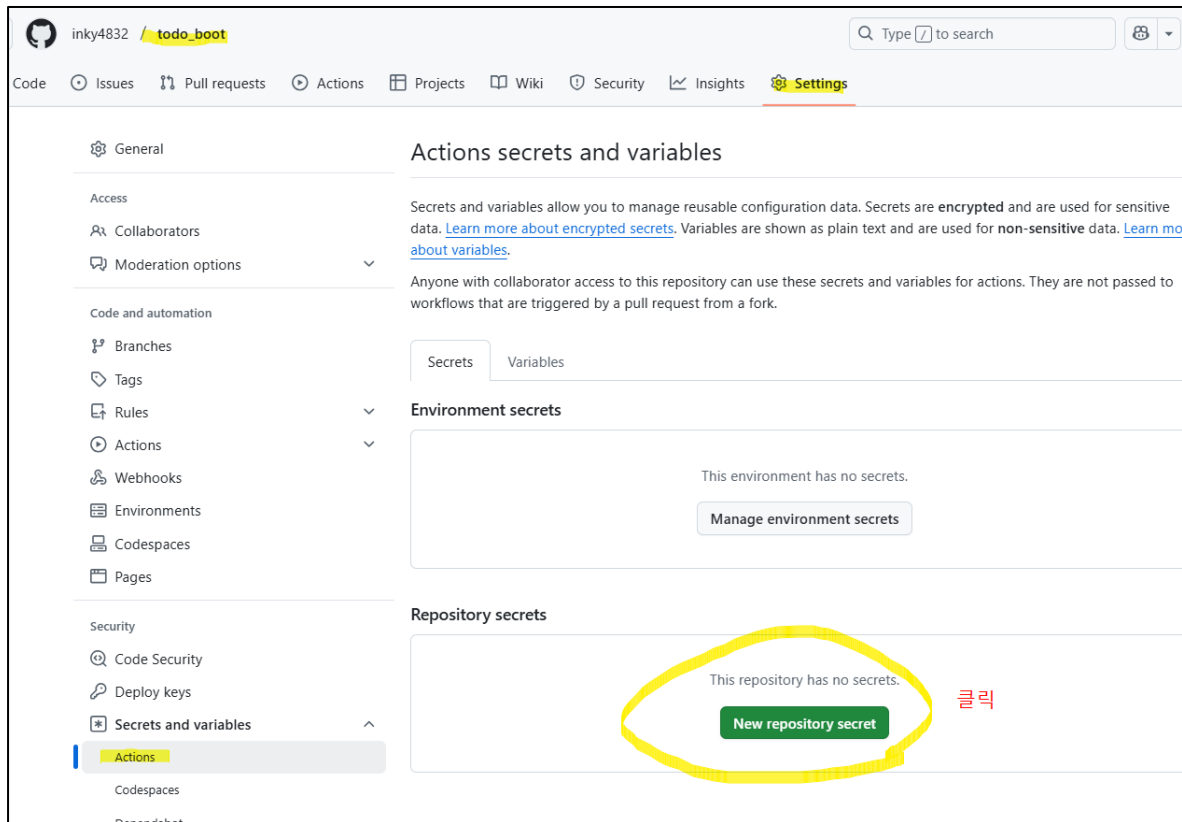
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

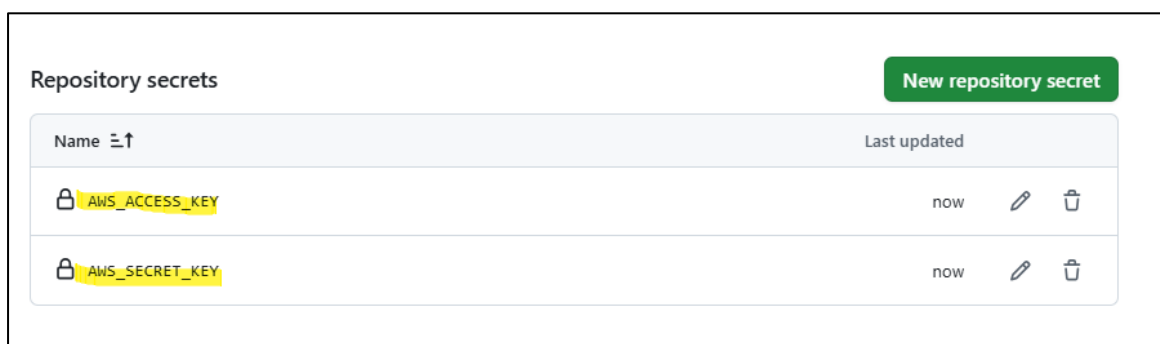
Create repository

2) Beanstalk 에서 생성한 액세스 키 설정하기



AWS_ACCESS_KEY 와 AWS_SECRET_KEY 이름으로 만들자.

제공된 deploy.xml 의 60라인과 61라인의 `${{secrets.AWS_ACCESS_KEY}}` 및 `${{secrets.AWS_SECRET_KEY}}` 와 반드시 일치해야 된다.



3) .github/workflows/deploy.xml 파일 생성

4) 다음 정보를 이용해서 github 저장소에 push.

...or create a new repository on the command line

```
echo "# todo_boot" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/inky4832/todo_boot.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/inky4832/todo_boot.git
git branch -M main
git push -u origin main
```

git init

git add *

git commit -m "first commit"

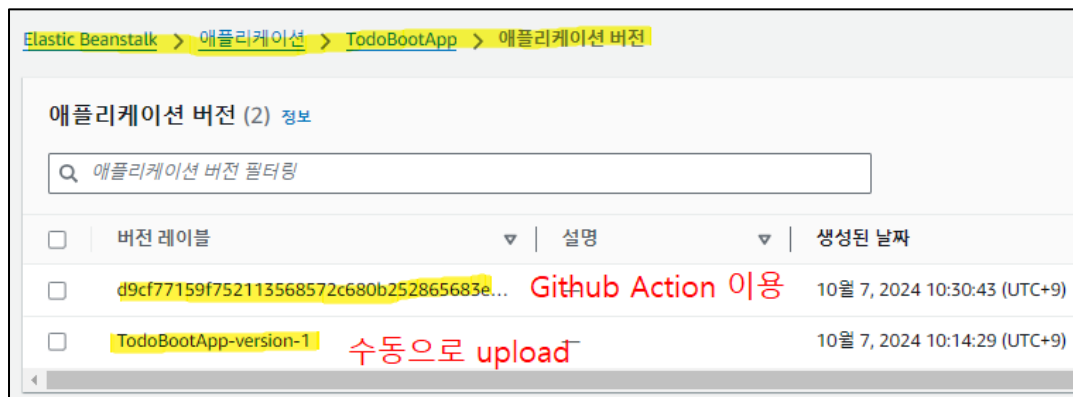
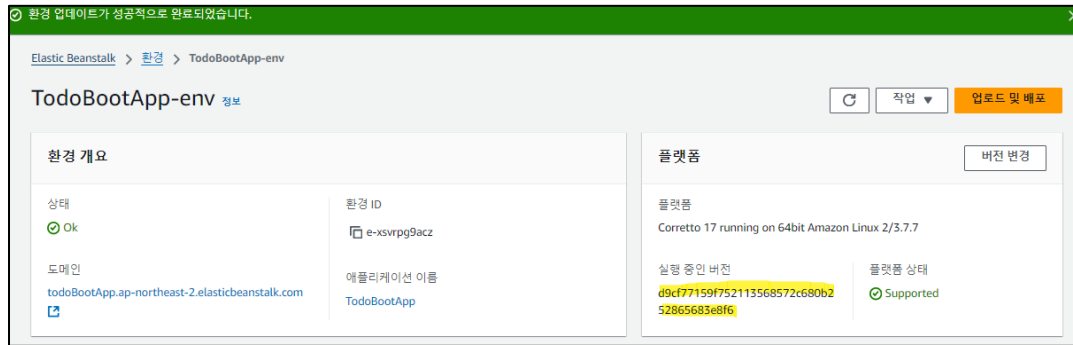
git branch -M main

git remote add origin https://github.com/inky4832/todo_boot.git

git push -u origin main

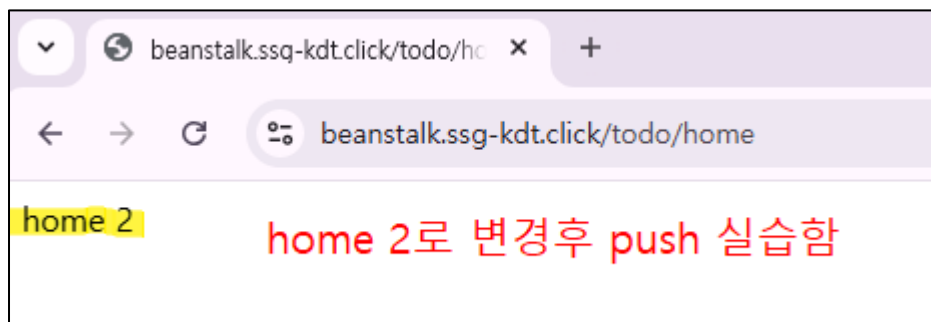
5) 자동 배포 확인하기

workflow 작업 성공후 Beanstalk에서 다음과 같이 실행중인 버전이 변경된 것을 확인할 수 있음.



6) 다시 요청하기

<https://beanstalk.ssg-kdt.click/todo/home> 요청하기



```
////////////////////////////////////////////////////////////////  
////////////////////////////////////////////////////////////////
```

```
# .github/workflows/deploy.yml
```

```
name: SpringBoot Todo CI/CD Pipeline
```

```
on:
```

```
  workflow_dispatch:
```

```
    # this will trigger workflow whenever a change is pushed to main branch
```

```
  push:
```

```
jobs:
```

```
  build:
```

```
    name: Build Archive
```

```
    # Will run steps on latest version of ubuntu
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      # Check-out your repository under $GITHUB_WORKSPACE, so your workflow can access it
```

```
      - uses: actions/checkout@v4
```

```
      # Set up JDK 17
```

```
      - name: Set up JDK
```

```
        uses: actions/setup-java@v4
```

```
        with:
```

```
          distribution: 'temurin'
```

```
          java-version: '17'
```

```
      # Set up Maven cache
```

```
      - name: Cache Maven packages
```

```
        # This action allows caching dependencies and build outputs to improve workflow execution time.
```

```
        uses: actions/cache@v4
```

with:

path: ~/.m2

key: \${runner.os}-m2-\${hashFiles('**/pom.xml')}

restore-keys: \${runner.os}-m2

Build the application using Maven

- name: Build with Maven

run: mvn -B package -DskipTests --file pom.xml

- name: Find name

run: find . -name "*.jar"

- name: Upload JAR

We upload so we can re-use same jar in next job.

uses: actions/upload-artifact@v4

with:

Name of artifact can be anything

name: artifact

Relative path to jar file

path: target/*.jar

Deploy's job

deploy:

Depends on build's job

needs: build

name: Deploy to Elastic Beanstalk

Will run steps on latest version of ubuntu

runs-on: ubuntu-latest

steps:

- name: Download JAR

```
# Download the artifact which was uploaded in the Build Archive's job

uses: actions/download-artifact@v4

with:

  name: artifact

# Deploy the artifact (JAR) into AWS Beanstalk

# https://github.com/marketplace/actions/elastic-beanstalk-deployer

- name: Deploy to EB

  uses: einaregilsson/beanstalk-deploy@v20

  with:

    aws_access_key: ${secrets.AWS_ACCESS_KEY} # This is referred from Github Secrets

    aws_secret_key: ${secrets.AWS_SECRET_KEY} # This is referred from Github Secrets

    use_existing_version_if_available: true

    application_name: TodoBootApplication # Application name we created in Elastic Beanstalk

    environment_name: TodoBootApplication-env # Environment name we created in Elastic Beanstalk

    version_label: ${github.SHA}

    region: ap-northeast-2

    deployment_package: todo.jar # Download artifacts from previous job
```

4. React.js 수동 배포

1) React에서 확인할 사항

http://localhost:8090/todo/home 형태의 URL 값을 AWS Beanstalk 의 도메인 값 또는 Route 53 레코드값으로 모두 변경해야 된다.

<http://todobootapp.ap-northeast-2.elasticbeanstalk.com/todo/home>

<https://beanstalk.ssg-kdt.click/todo/home>

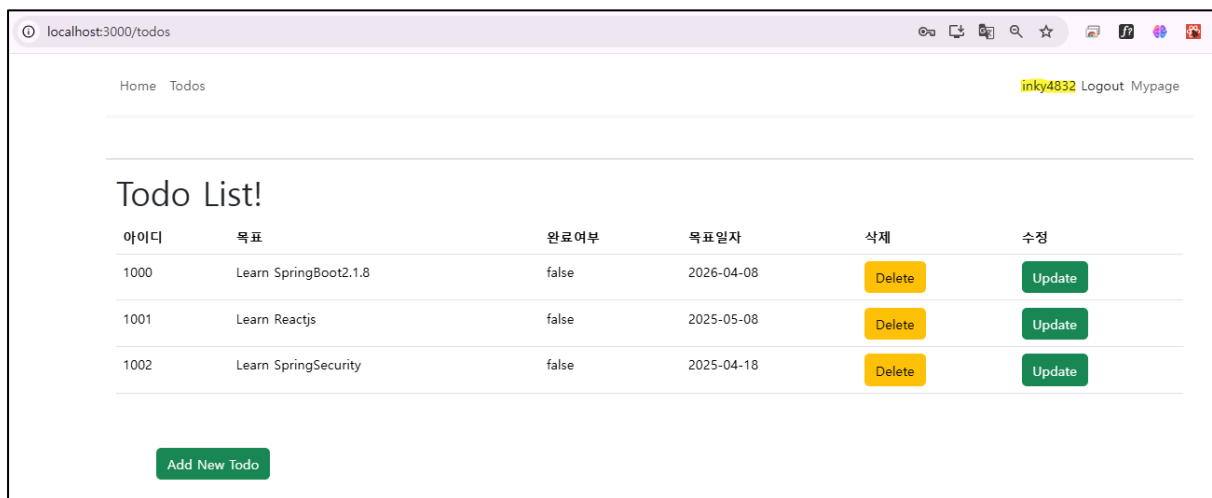
src/api/httpMemberService.js

```
JS httpMemberService.js U X JS httpTodoService.js JS App.js U
my-app > src > api > JS httpMemberService.js > fetchSignup
1 // httpMemberService.js
2
3 import axios from 'axios'
4
5 const instance = axios.create({
6   baseURL: 'https://beanstalk.ssg-kdt.click/todo', // Boot 서버에 반드시 CORS 설정 필수
7   timeout: 1000,
8   headers: { 'Content-Type': 'application/json' }
9 });
10
11 // 회원가입
12 export async function fetchSignup(user) {
13
14   console.log("fetchSignup 요청")
15
16   const response = await instance.post(`/signup`, user);
17
18   return response;
19 }
20
```

src/api/httpTodoService.js


```
JS httpMemberService.js U JS httpTodoService.js X JS App.js U
my-app > src > api > JS httpTodoService.js > ...
1 // httpTodoService.js
2
3
4 import axios from 'axios'
5
6 const instance = axios.create({
7   baseURL: 'https://beanstalkk.ssg-kdt.click/todo', // Boot 서버에 반드시 CORS 설정 필수
8   timeout: 1000,
9   headers: { 'Content-Type': 'application/json' }
10 });
11
12 // Todo 목록보기
13 export async function fetchFindAll(token) {
14
15   const response = await instance.get(`/todos`, {
16     headers: {
17       Authorization: `Bearer ${token}`
18     }
19   });
20
21   console.log("findAll.response:", response)
22
23   return response;
24 }
```

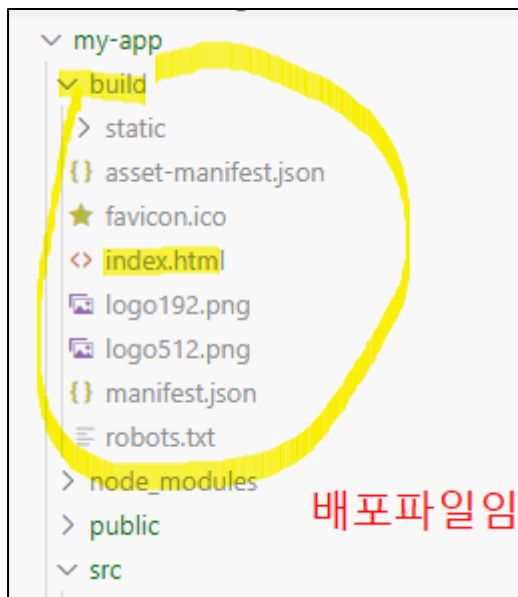
2) 배포전에 VSC 에서 실행해보자.



3) react.js 빌드하자.

`npm run build`

다음과 같이 my-app 프로젝트내에 build 폴더가 생성되고 그 안에 배포파일이 저장되어 있음.



4) Amazon S3 에서 버킷 생성하기

다음과 같이 기본적으로 Beanstalk에서 생성해준 버킷이 있으나 추가로 새로운 버킷을 생성함.



버킷 생성시 버킷 이름은 중복되지 않도록 지정해야 된다.

버킷 만들기

정보

버킷은 S3에 저장되는 데이터의 컨테이너입니다.

일반 구성

AWS 리전

아시아 태평양(서울) ap-northeast-2

버킷 이름

정보

todoreactjsbucket

todoreactjsbucket

버킷 이름은 3~63자여야 하며 글로벌 네임스페이스 내에서 고유해야 합니다. 또한 버킷 이름은 문자나 숫자로 시작하고 끝나야 합니다. 유효한 문자는 a-z, 0-9, 마침표(.), 하이픈(-)입니다. [자세히 알아보기](#)

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

형식: s3://bucket/prefix

객체소유권은 다음과 같이 비활성화로 설정함.

객체 소유권

정보

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

☒ ACL 비활성화됨(권장)

이 버킷의 모든 객체는 이 계정이 소유합니다. 이 버킷과 그 객체에 대한 액세스는 정책을 통해서만 지정됩니다.

☐ ACL 활성화됨

이 버킷의 객체는 다른 AWS 계정에서 소유할 수 있습니다. 이 버킷 및 객체에 대한 액세스는 ACL을 사용하여 지정할 수 있습니다.

객체 소유권

버킷 소유자 적용

외부에서 접근해야 되기 때문에 모든 퍼블릭 액세스 차단을 비활성화 시킴.

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

☒ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

- ☐ **새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**
S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.
- ☐ **임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.
- ☐ **새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.
- ☐ **임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단**
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.



모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

- ☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

버킷 버전 관리

버전 관리는 객체의 여러 버전을 동일한 버킷에서 관리하기 위한 수단입니다. 버전 관리를 사용하여 Amazon S3 버킷에 저장된 모든 객체의 각 버전을 보존, 검색 및 복원할 수 있습니다. 버전 관리를 통해 의도치 않은 사용자 작업과 애플리케이션 장애를 모두 복구할 수 있습니다. [자세히 알아보기](#)

버킷 버전 관리

☒ 비활성화

☐ 활성화

태그 - 선택 사항 (0)

버킷 태그를 사용하여 스토리지 비용을 추적하고 버킷을 구성할 수 있습니다. [자세히 알아보기](#)

이 버킷과 연결된 태그가 없습니다.

태그 추가

기본 암호화 정보

서버 측 암호화는 이 버킷에 저장된 새 객체에 자동으로 적용됩니다.

암호화 유형 | 정보

- ☒ Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화
- ☐ AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
- ☐ AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)
두 개의 개별 암호화 계층으로 객체를 보호합니다. 요금에 대한 자세한 내용은 [Amazon S3 요금 페이지](#)의 스토리지 탭에서 DSSE-KMS 요금을 참조하세요.

버킷 키

SSE-KMS를 S3 버킷 키를 사용하면 AWS KMS에 대한 호출을 줄여 암호화 비용이 절감됩니다. DSSE-KMS에는 S3 버킷 키가 지원되지 않습니다. [자세히 알아보기](#)

- ☐ 비활성화
- ☒ 활성화

고급 설정

버킷을 생성한 후 파일과 폴더를 해당 버킷에 업로드할 수 있고, 추가 버킷 설정도 구성할 수 있습니다.

취소

버킷 만들기

범용 버킷

디렉터리 버킷

범용 버킷 (2) 정보 모든 AWS 리전

버킷은 S3에 저장되는 데이터의 컨테이너입니다.

이름으로 버킷 찾기

이름

▲

AWS 리전

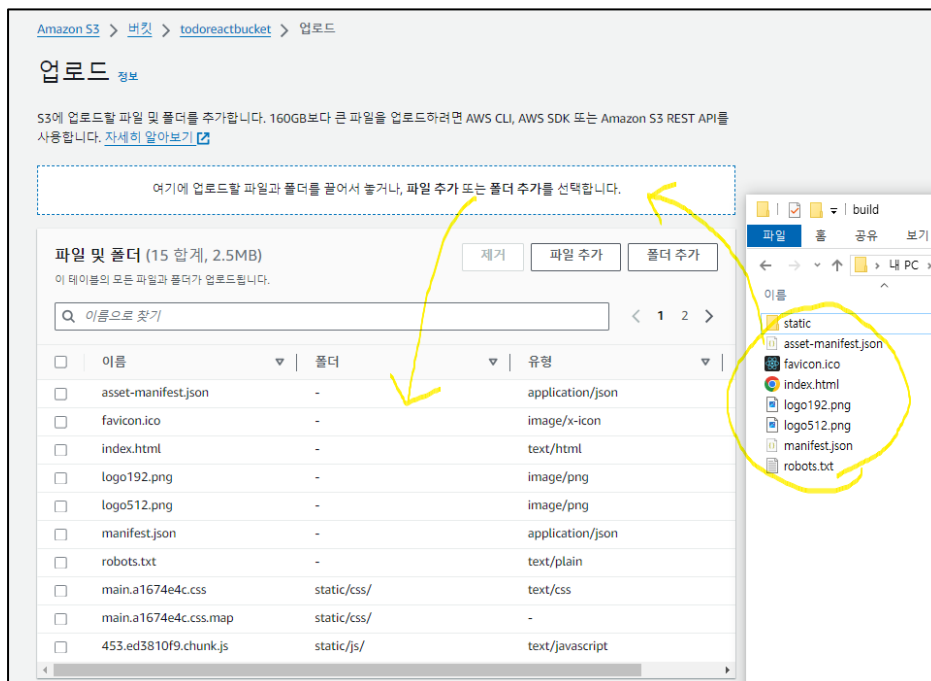
- ☐ [elasticbeanstalk-ap-northeast-2-285502508151](#) 아시아 태평양(서울) ap-northeast-2
- ☐ [todoreactjsbucket](#) **생성됨** 아시아 태평양(서울) ap-northeast-2

다음과 같이 상세 페이지로 가면 객체 정보를 확인할 수 있고 업로드도 가능함.



업로드 > 여기에 업로드할 파일과 폴더를 끌어서 놓거나~ 에 드래그 드롭함.

my-app 프로젝트의 build 폴더내 파일 복사한다.



업로드 버튼 클릭.

업로드 성공

자세한 내용은 파일 및 폴더 테이블을 참조하세요.

업로드: 상태

이 페이지에서 나가면 다음 정보를 더 이상 확인할 수 없습니다.

요약

대상

s3://todoreactjsbucket

성공

13개 파일, 2.7MB (100.00%)

실패

0개 파일, 0B (0%)

파일 및 폴더

구성

파일 및 폴더 (13 합계, 2.7MB)

이름으로 찾기

이름	폴더	유형	크기	상태
index.html	-	text/html	644.0B	성공
logo192.png	-	image/png	5.2KB	성공
logo512.png	-	image/png	9.4KB	성공
manifest.json	-	application/json	492.0B	성공
robots.txt	-	text/plain	67.0B	성공
asset-manifest.json	-	application/json	469.0B	성공
favicon.ico	-	image/x-icon	3.8KB	성공
main.e612c2fd.css	static/css/	text/css	229.1KB	성공
main.e612c2fd.css.map	static/css/	-	532.2KB	성공
main.336d1eff.js	static/js/	text/javascript	251.9KB	성공

[닫기] 버튼을 클릭하자.

정적 웹사이트 호스팅 설정한다.

버킷 > **todoreactjsbucket**> 속성

가장 하단에 정적 웹사이트 호스팅 항목이 있음. 편집버튼 클릭.

요청자 지불

활성화하면, 요청자가 요청 및 데이터 전송 비용을 지불하고 이 버킷에 대한 읽을 액세스는 비활성화됩니다. [자세히 알아보기](#)

요청자 지불

비활성화됨

정적 웹 사이트 호스팅

이 버킷을 사용하여 웹 사이트를 호스팅하거나 요청을 리디렉션합니다. [자세히 알아보기](#)

정적 웹 사이트 호스팅

비활성화됨

정적 웹 사이트 호스팅 편집 정보

정적 웹 사이트 호스팅

이 버킷을 사용하여 웹 사이트를 호스팅하거나 요청을 리디렉션하십시오. [자세히 알아보기](#)

정적 웹 사이트 호스팅

☐ 비활성화

☒ 활성화

호스팅 유형

☒ 정적 웹 사이트 호스팅

버킷 엔드포인트를 웹 주소로 사용합니다. [자세히 알아보기](#)

☐ 객체에 대한 요청 리디렉션

요청을 다른 버킷 또는 도메인으로 리디렉션합니다. [자세히 알아보기](#)

- ① 고객이 웹 사이트 엔드포인트의 콘텐츠에 액세스할 수 있게 하려면 모든 콘텐츠를 공개적으로 읽기 가능하도록 설정해야 합니다. 이렇게 하려면, 버킷에 대한 S3 퍼블릭 액세스 차단 설정을 편집하면 됩니다. 자세한 내용은 [Amazon S3 퍼블릭 액세스 차단 사용](#) 참조하십시오.

인덱스 문서

웹 사이트의 홈 페이지 또는 기본 페이지를 지정합니다.

index.html

오류 문서 - 선택 사항

오류가 발생하면 반환합니다.

error 페이지가 현재는 없기 때문에 그냥 index.html 입력

리디렉션 규칙 - 선택 사항

JSON으로 작성된 리디렉션 규칙은 특정 콘텐츠에 대한 웹 페이지 요청을 자동으로 리디렉션합니다. [자세히 알아보기](#)

정적 웹 사이트 호스팅

이 버킷을 사용하여 웹 사이트를 호스팅하거나 요청을 리디렉션하십시오. [자세히 알아보기](#)

- ① 정적 웹 사이트 호스팅에는 **AWS Amplify Hosting**을 사용하는 것이 좋습니다. AWS Amplify Hosting을 사용하여 빠르고 안전하며 안정적인 웹 사이트를 빠르게 배포하거나 [기존 Amplify 앱을 확인하세요](#).

S3 정적 웹 사이트 호스팅

활성화됨

호스팅 유형

버킷 호스팅

버킷 웹 사이트 엔드포인트

버킷을 정적 웹 사이트로 구성하면, 해당 웹 사이트를 버킷의 AWS 리전별 웹 사이트 엔드포인트에서 사용할

<http://todoreactjsbucket.s3-website.ap-northeast-2.amazonaws.com>

[속성] 탭으로 가서 가장 하단을 확인하자.

버킷 웹 사이트 엔드포인트 URL이 제공됨

<http://todoreactjsbucket.s3-website.ap-northeast-2.amazonaws.com>

다음은 버킷정책을 설정하는 화면이다. 현재는 비워져 있음

버킷 > todoreactjsbuckt > 권한 > 버킷 정책의 편집 > 정책 생성기 클릭



AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to [Amazon Web Services \(AWS\)](#) products and resources. Here are [sample policies](#).

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service Amazon S3 ☐ All Services (**)

Use multiple statements to add permissions for more than one service.

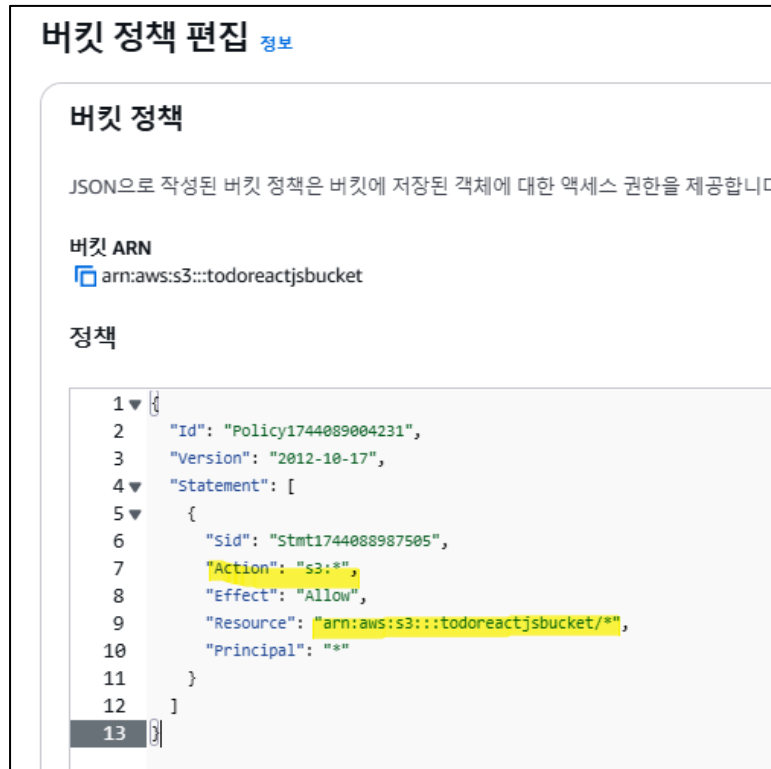
Actions -- Select Actions -- ☒ All Actions (**)

Amazon Resource Name (ARN) arn:aws:s3:::todoreactjsbuckt

ARN should follow the following format: `arn:aws:s3:::{BucketName}/{KeyPrefix}`.
Use a comma to separate multiple ARNs.

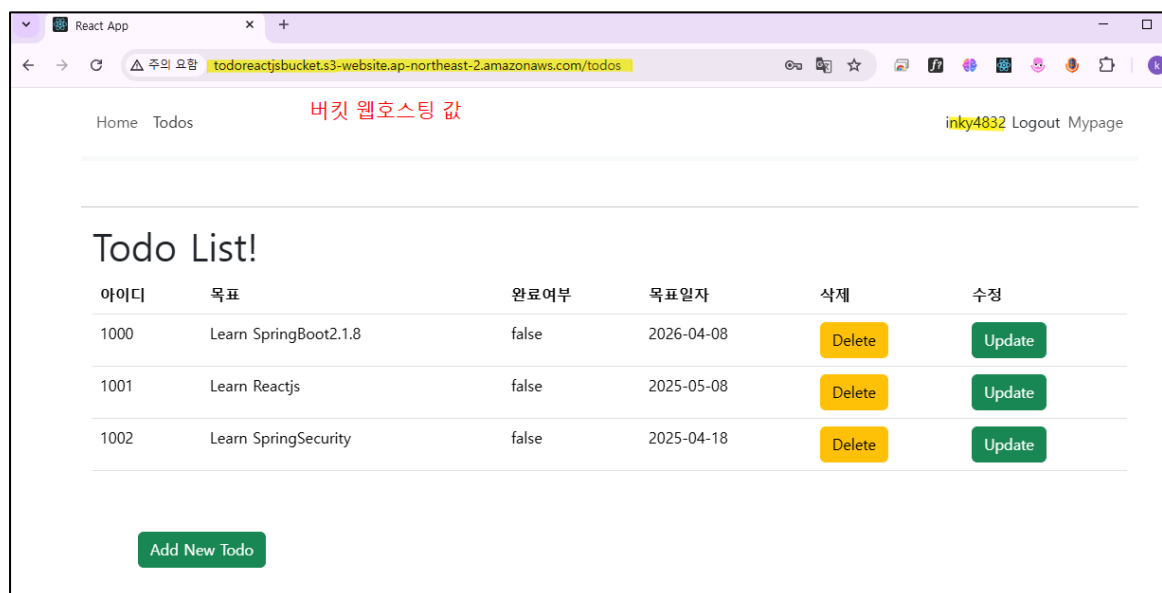
ARN/*
ARN 값은 버킷의 속성탭에서 확인 가능

[Add Conditions \(Optional\)](#)



브라우저에서 요청하기

<http://todoreactjsbucket.s3-website.ap-northeast-2.amazonaws.com>



지금까지 AWS Beanstalk 이용해서 Reactjs 어플리케이션을 수동으로 배포하는 방법이었습니다

5. GithubActions(CI/CD) 이용한 자동 배포

1) Github 에 저장소 생성하기

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

Repository name *

inky4832

/

todo_react

todo_react

✓

todo_react is available.

Great repository names are short and memorable. Need inspiration? How about [musical-octo-fishstick](#) ?

Description (optional)

☒

Public

Anyone on the internet can see this repository. You choose who can commit.

☐

Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒

Add a README file

README 파일은 생성안함

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

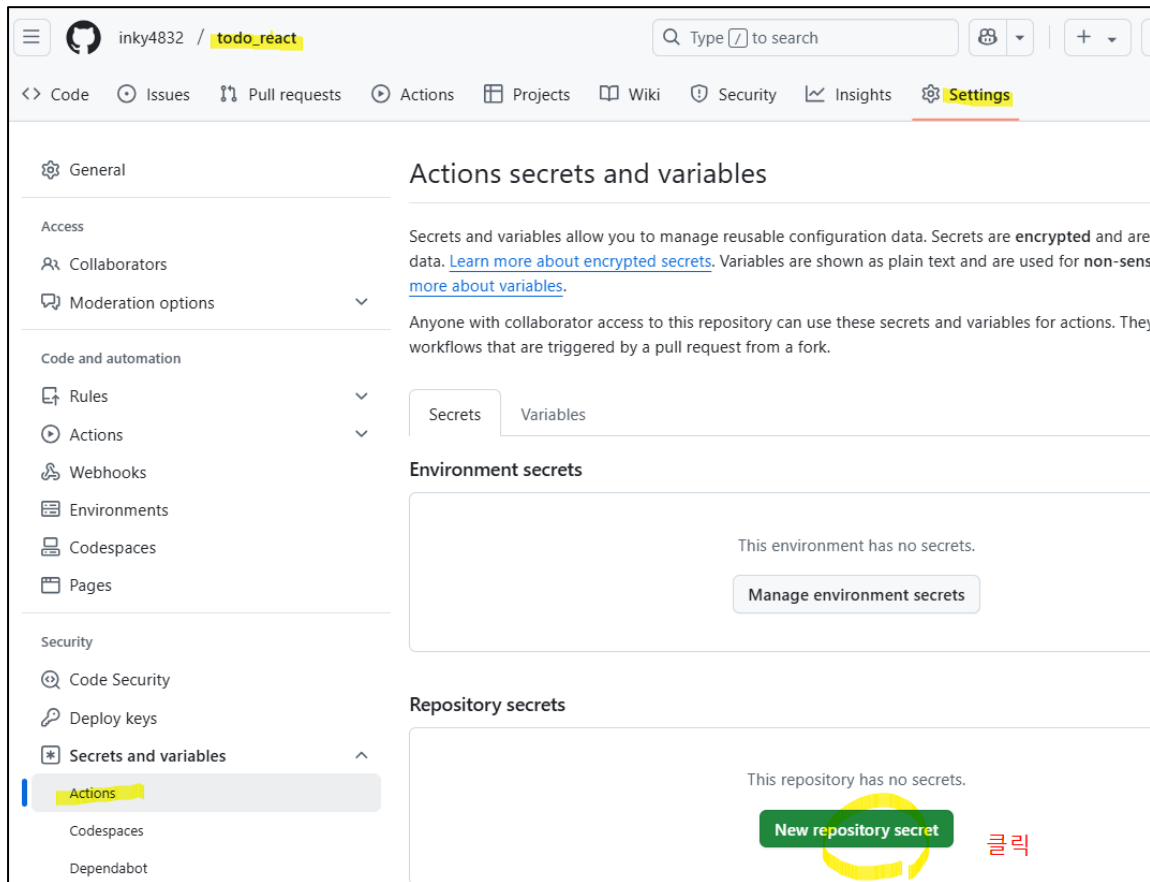
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ

You are creating a public repository in your personal account.

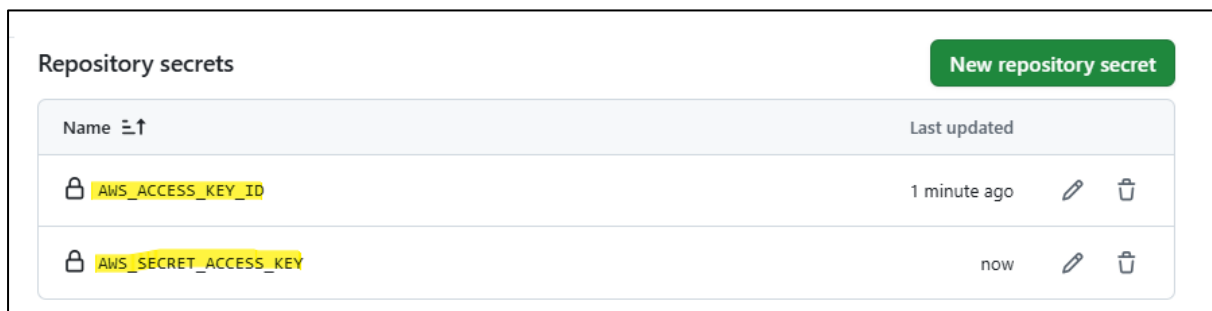
Create repository

2) Beanstalk 에서 생성한 액세스 키 설정하기



AWS_ACCESS_KEY_ID 와 AWS_SECRET_ACCESS_KEY 이름으로 만들자.

제공된 deploy.xml 의 54라인과 55라인의 `${{secrets.AWS_ACCESS_KEY_ID}}` 및 `${{secrets.AWS_SECRET_ACCESS_KEY}}` 와 반드시 일치해야 된다.



3) `.github/workflows/deploy.xml` 파일 생성

4) 다음 정보를 이용해서 github 저장소에 push.

...or create a new repository on the command line

```
echo "# todo_react" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/inky4832/todo_react.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/inky4832/todo_react.git
git branch -M main
git push -u origin main
```

git init

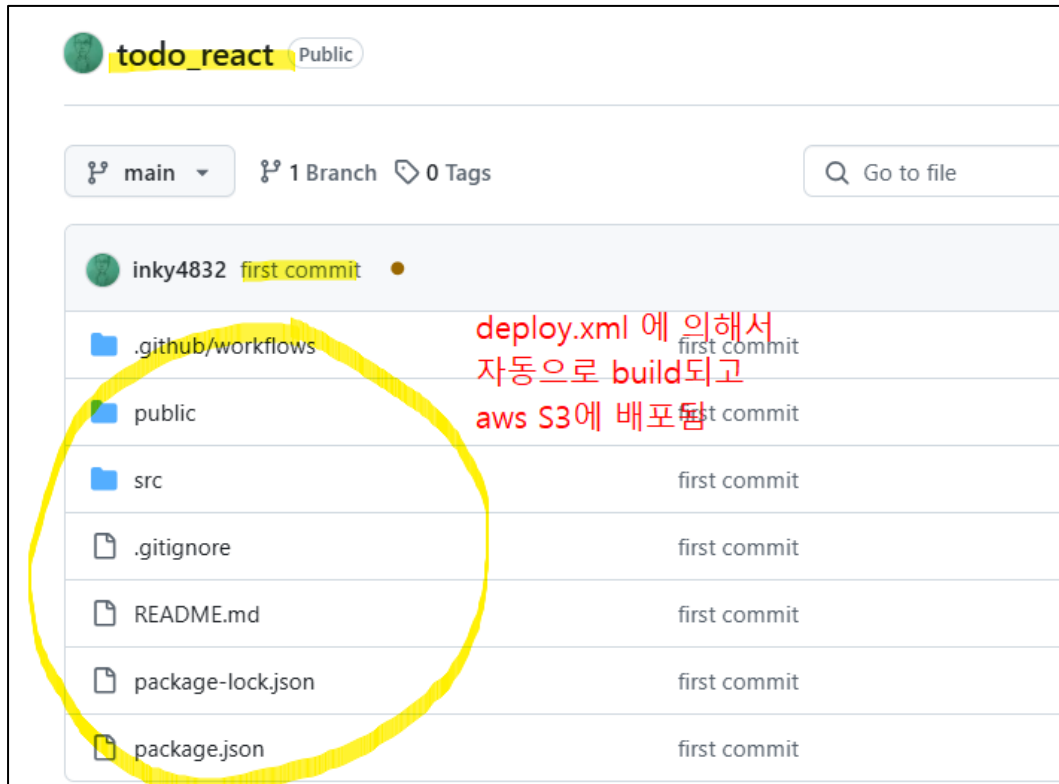
git add *

git commit -m "first commit"

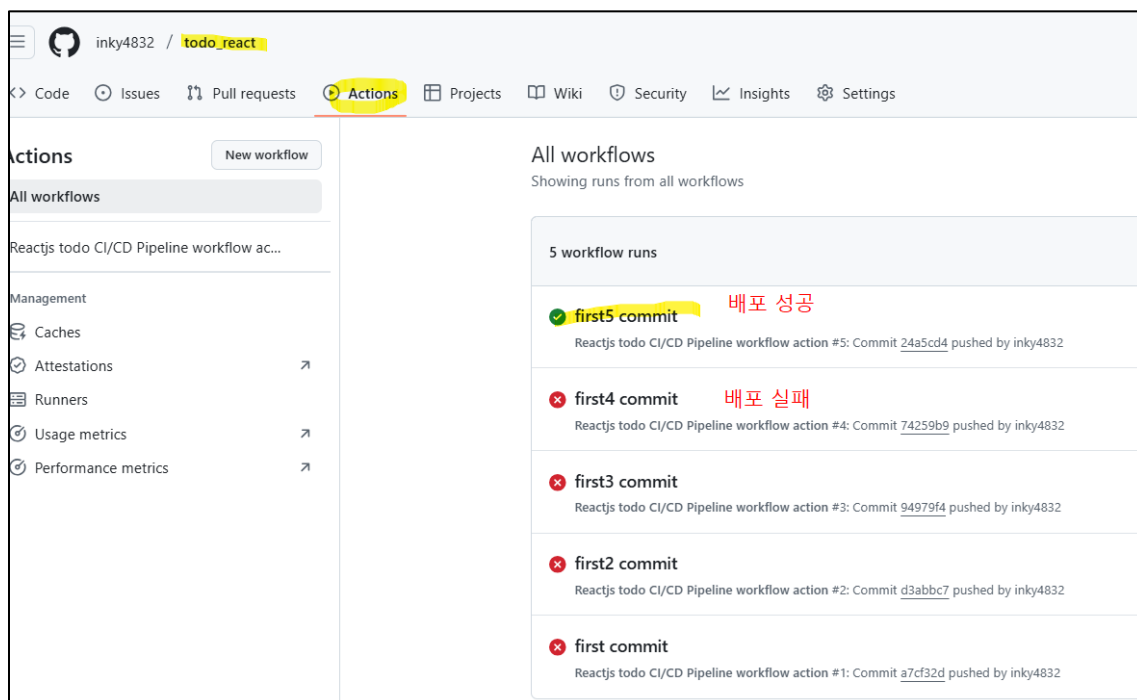
git branch -M main

git remote add origin https://github.com/inky4832/todo_react.git

git push -u origin main



5) 자동 배포 확인하기



todoreactjsbucket 정보

객체 | 속성 | 권한 | 지표 | 관리 | 액세스 지점

객체 (8) 🔄 📄 S3 URI 복사 📄 URL 복사 📄 다운로드

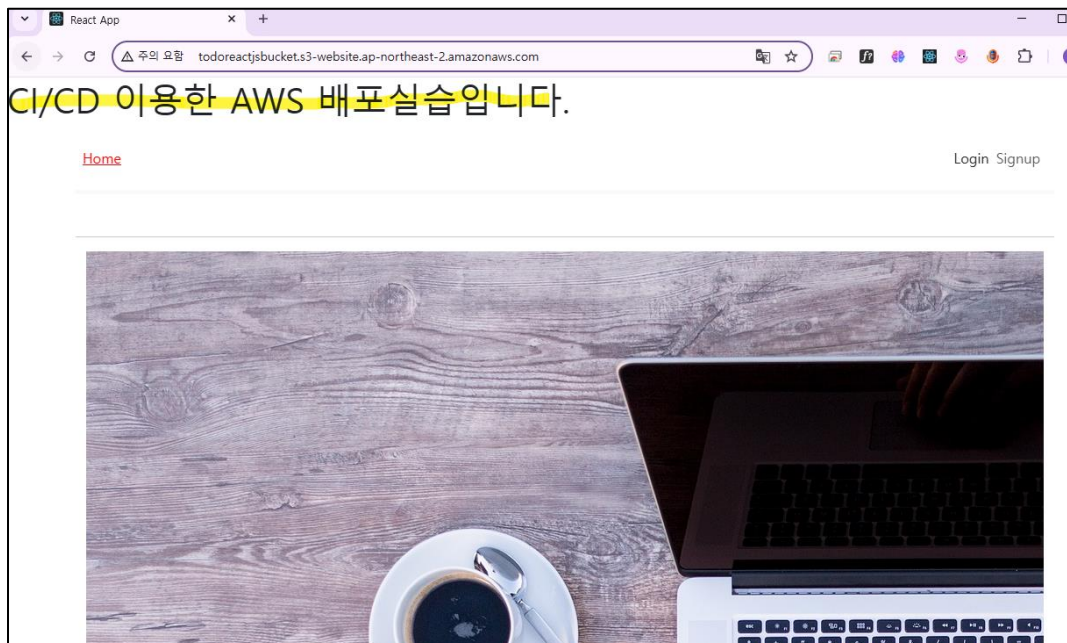
객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. [Amazon S3 인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할

🔍 접두사로 객체 찾기

<input type="checkbox"/>	이름	▲ 유형	▼ 마지막 수정	배포 시간 확인	▼ 크기
<input type="checkbox"/>	asset-manifest.json	json	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	favicon.ico	ico	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	index.html	html	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	logo192.png	png	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	logo512.png	png	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	manifest.json	json	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	robots.txt	txt	2025. 4. 8. pm 2:52:41 PM KST		
<input type="checkbox"/>	static/	폴더	-		

6) 다시 요청하기

<https://beanstalkk.ssg-kdt.click/todo/home> 요청하기



```
# .github/workflows/deploy.yml
```

name: Reactjs todo CI/CD Pipeline workflow action

on: push

jobs:

build:

name: Build

```
runs-on: ubuntu-latest
```

steps:

- name: Checkout source code

uses: actions/checkout@v4

- name: Setup Node

uses: actions/setup-node@v4

with:

node-version: 20

- name: Clean npm cache and reinstall dependencies

run: |

```
npm install --save-dev @babel/plugin-proposal-private-property-in-object
```


- name: Build project

run: |

CI=false npm run build

ls \${{github.workspace}}

- name: Upload production-ready build files

uses: actions/upload-artifact@v4

with:

name: artifact

path: ./build

deploy:

name: Deploy

needs: build

runs-on: ubuntu-latest

steps:

- name: Download artifact

Download the artifact which was uploaded in the Build Archive's job

uses: actions/download-artifact@v4

with:

name: artifact

path: ./build

Deploy the artifact into AWS S3

- name: Deploy to s3

uses: jakejarvis/s3-sync-action@master

with:

args: --delete

env:

AWS_S3_BUCKET: todoreactjsbucket

AWS_ACCESS_KEY_ID: \${ secrets.AWS_ACCESS_KEY_ID } # This is referred from Github

Secrets

AWS_SECRET_ACCESS_KEY: \${ secrets.AWS_SECRET_ACCESS_KEY } # This is referred from

Github Secrets

AWS_REGION: ap-northeast-2

SOURCE_DIR: ./build

////////////////////////////////////

////////////////////////////////////