



**BESPIN GLOBAL**

● **HELPING YOU ADOPT CLOUD.**

---

## **AWS 기술지원 가이드**

**AWS CodePipeline 을 활용한 배포 자동화**

2019-09-16

AWS 2 TEAM

최장섭

## 워크 플로우

### AWS Requirement

#### IAM

[AWS CodeCommit User](#)

[AWS IAM Role](#)

#### AWS S3

#### AWS Auto Scaling Group

[AWS AMI Configuration](#)

[AWS CodeDeploy Agent](#)

[AWS Launch Configuration](#)

[AWS Auto Scaling Group Configuration](#)

### AWS Codecommit

[AWS Codecommit Repositories](#)

[IDE Configuration for Repositories](#)

[Source Commit](#)

[MAVEN Configuration](#)

### AWS CodeBuild

[AWS CodeBuild Project](#)

### AWS CodeDeploy

[AWS CodeDeploy Application](#)

[AWS CodeDeploy Deploy Group](#)

[AWS CodeDeploy Deploy](#)

### AWS CodePipeline

## 별첨 문서

[IDE Project Configuration](#)

[기존에 소스가 있는 경우](#)

[기존에 소스가 없는 경우](#)

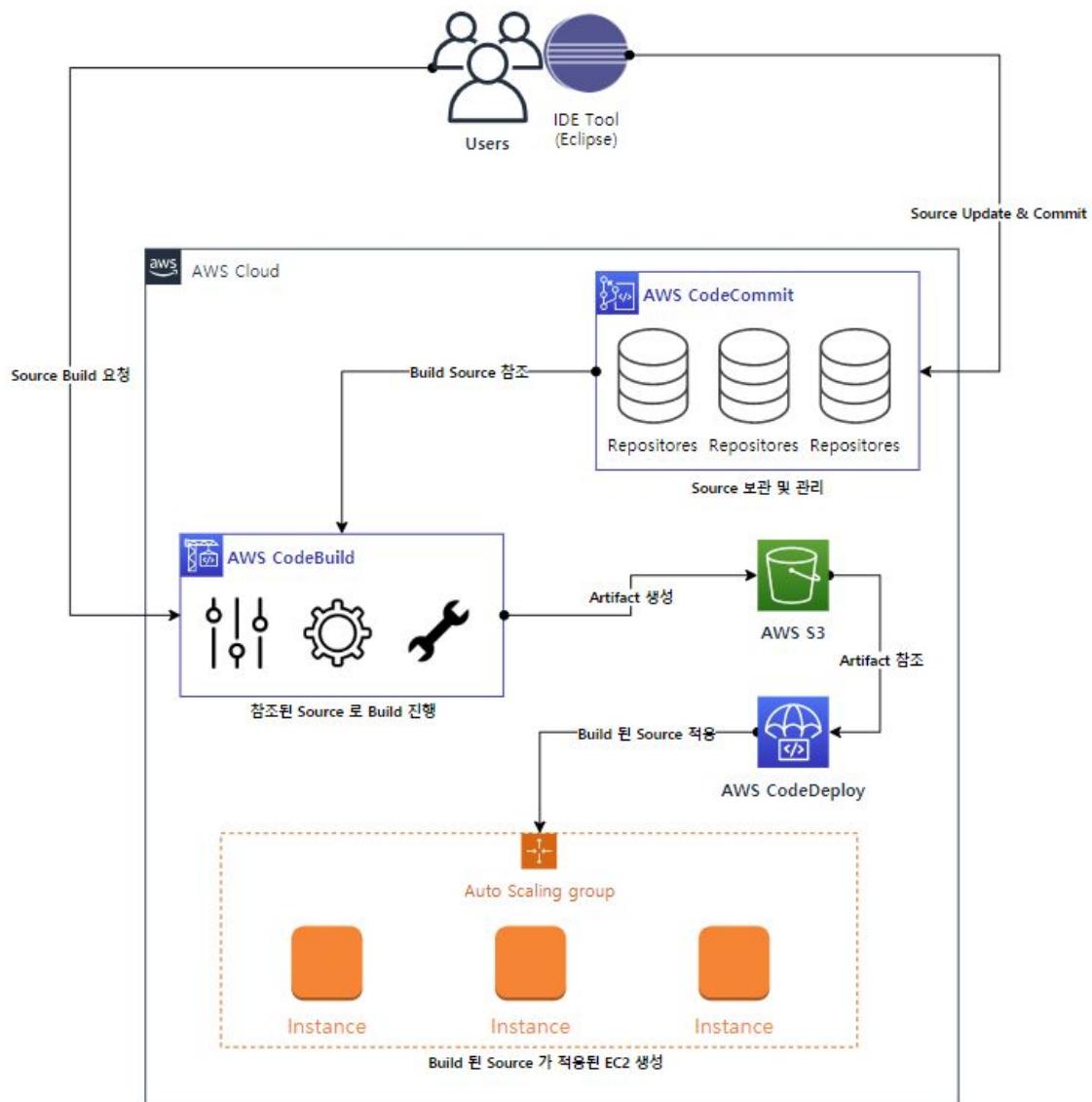
### SAMPLE CODE

[Buildspec.yml](#)

[Appspec.yml](#)

[Application Scripts](#)

# 워크 플로우



# 1. AWS Requirement

## 1.1. IAM

### 1.1.1. AWS CodeCommit User

AWS CodeCommit Repositories 를 사용하기 위해서는 IAM 계정에 대한 설정이 필요합니다.

- IDE 툴에서 Repositories 접근을 위해 AWS CodeCommit 역할을 갖는 User를 생성합니다.

The screenshot shows the IAM Management Console interface. On the left, there's a sidebar with navigation links like 'AWS Account', 'AWS Organizations', 'AWS Organizations activity', and 'Service control policies (SCPs)'. The main area is titled 'jschoi-codecommit-user' under 'Identity and Access Management (IAM)'. It displays the user's ARN (arn:aws:iam::270881836940:user/jschoi-codecommit-user), the creation date (2019-09-16 10:40 UTC+0900), and a permissions tab. Under 'Permissions policies', an inline policy named 'AWSCodeCommitFullAccess' is listed, which grants 'CodeCommit 그룹의 AWS 관리형 정책' (AWS-managed policy for the CodeCommit group). A red box highlights this inline policy.

- 사용자 IDE 툴 (이클립스)에서 AWS Resources 접근을 위한 Access Key를 생성합니다.

The screenshot shows the IAM Management Console again, focusing on access keys. It lists an access key for the user 'jschoi-codecommit-user' with the ID 'AKIAT6EOQZ6GLOEZMHGA', created on '2019-09-16 10:40 UTC+0900', and marked as '활성' (Active). A red box highlights this access key row.

- 해당 user에 대해 AWS CodeCommit에 대한 HTTPS Git 자격 증명을 생성합니다.

The screenshot shows the AWS CodeCommit HTTPS Git credential generation page. It displays the URL 'jschoi-codecommit-user-at-270881836940' and the creation date '2019-09-16 10:42 UTC+0900'. A red box highlights this URL.

## 1.1.2. AWS IAM Role

- EC2/온프레미스 배포의 경우 **AWSCodeDeployRole** 정책 필요 합니다.

Identity and Access Management (IAM)

역할 > jschoi-codedeploy-role

역할 ARN: arn:aws:iam::270881836940:role/jschoi-codedeploy-role

역할 설명: Allows EC2 instances to call AWS services on your behalf. | 편집

인스턴스 프로파일 ARN: arn:aws:iam::270881836940:instance-profile/jschoi-codedeploy-role

경로: /

생성 시간: 2019-09-16 16:24 UTC+0900

최대 CLI/API 세션 기간: 1 시간 편집

권한 | 신뢰 관계 | 태그 | 액세스 관리자 | 세션 취소

Permissions policies (1 정책이 적용됨)

정책 연결 | 인라인 정책 추가

정책 이름	정책 유형
AWSCodeDeployRole	AWS 관리형 정책

Permissions boundary (not set)

## 1.2. AWS S3

- AWS CodeBuild에서 생성되는 아티팩트를 보관 됩니다.

Amazon S3

S3 버킷

jschoi-codebuild 모든 액세스 유형

+ 버킷 만들기 | 퍼블릭 액세스 설정 편집 | 버우기 | 삭제

1 버킷 1 리전

버킷 이름	액세스	리전	생성 날짜
jschoi-codebuild	퍼블릭 아닌 버킷 및 객체	미국 동부(오하이오)	9월 16, 2019 4:06:26 오후 GMT+0900

## 1.3. AWS Auto Scaling Group

- AWS CodeDeploy에서 실제 소스가 배포될 인스턴스들을 생성합니다.

### 1.3.1. AWS AMI Configuration

AWS CodeDeploy에서 Auto Scaling Group을 통해 생성되는 인스턴스에서 CodeDeploy Agent를 사용할 수 있도록 AMI를 생성합니다.

#### 1.3.1.1. AWS CodeDeploy Agent

다음을 참고하여 인스턴스에 CodeDeploy Agent를 설치합니다.

- 다음 명령을 실행하여 Agent에 필요한 패키지를 설치합니다.

```
sudo yum update
sudo yum install -y ruby wget
cd /home/ec2-user
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
```

bucket-name은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름이며, region-identifier는 리전의 식별자입니다.

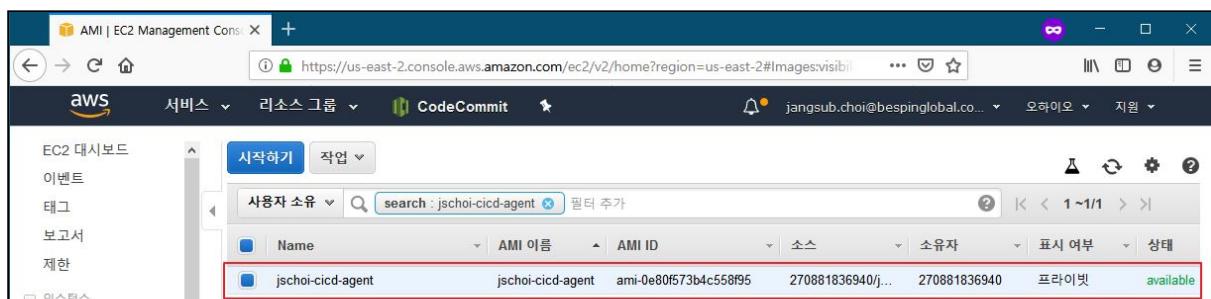
버킷 이름 및 리전 식별자 목록에 대해서는 다음을 참고 합니다. - [참고 링크](#)

- 다음 명령을 실행하여 Agent를 실행하고 확인 합니다.

```
[ec2-user@ip-192-168-20-92 ~]$ sudo service codedeploy-agent start
Starting
[ec2-user@ip-192-168-20-92 ~]$
```

```
[ec2-user@ip-192-168-20-92 ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 3630
[ec2-user@ip-192-168-20-92 ~]$
```

설치가 완료되면 해당 인스턴스를 사용하여 AMI를 생성 합니다.



### 1.3.2. AWS Launch Configuration

위 과정을 통해 생성한 AMI를 사용하여 시작 구성을 생성 합니다.

- AWS Web Console 내 EC2 페이지에서 시작 구성으로 이동하여 시작 구성을 생성합니다.

The screenshot shows two consecutive screenshots of the AWS Management Console for the EC2 service, specifically the Auto Scaling section.

**Screenshot 1:** The 'Launch Configuration' tab is selected. A modal window displays a warning message: "Compute usage up to 90% savings" and "Using a template to create an Auto Scaling group using EC2 On-Demand instances, Spot instances, and reserved instances to automatically create an Auto Scaling group to save computing usage costs." Below the modal, the search bar contains "Launch Configuration" and the results table shows one entry:

Name	AMI ID	Instance Type	Creation Time
jschoi-codedeploy-launch	ami-0e80f573b4c558f95	t2.micro	2019年9月16日 오후 4시 31분...

**Screenshot 2:** The same view, but the table now highlights the first row (the created launch configuration) with a red border. The table header is also highlighted with a red border.

### 1.3.3. AWS Auto Scaling Group Configuration

생성한 시작구성을 사용하여 Auto Scaling Group 을 생성합니다.

- AWS Web Console 내 EC2 페이지에서 Auto Scaling 그룹으로 이동하여 Auto Scaling 그룹을 생성 합니다.

**Auto Scaling 시작**

Auto Scaling을 사용하면 Amazon EC2 용량을 자동으로 관리하고 애플리케이션에 적합한 인스턴스 수를 유지할 수 있으며 정상 인스턴스 그룹을 운영하고 요건에 따라 규모를 조정할 수 있습니다.

**Auto Scaling의 이점**

- 자동화된 프로비저닝**: 인스턴스가 1개 필요하든 1000개 필요하든 상관없이, Auto Scaling 그룹을 균형이 유지되는 정상 상태로 유지합니다.
- 조정 가능 용량**: 고정 그룹 크기를 유지하거나 Amazon CloudWatch 지표에 따라 동적으로 조정합니다.
- 시작 템플릿 지원**: EC2 시작 템플릿을 사용하여 인스턴스를 쉽게 프로비저닝합니다.

**추가 정보**

- 시작 안내서
- 설명서
- 모든 EC2 리소스
- 포럼
- 요금
- 문의처

- CodeDeploy를 위해 최소 1개 이상의 인스턴스가 생성되어야 합니다.

**Auto Scaling 그룹: jschoi-codedeploy-asg**

필터:	jschoi-codedeploy-asg	1~1/1 Auto Scaling 그룹 >						
이름	jschoi-codedeploy-launch	시작 구성 / 템플릿	인스턴스	목표 용량	최소	최대	기본 유지	상태 검사 유예 기간
				0	0	4	us-east-2a, us-east-2c	300

**Auto Scaling 그룹: jschoi-codedeploy-asg**

**세부 정보**

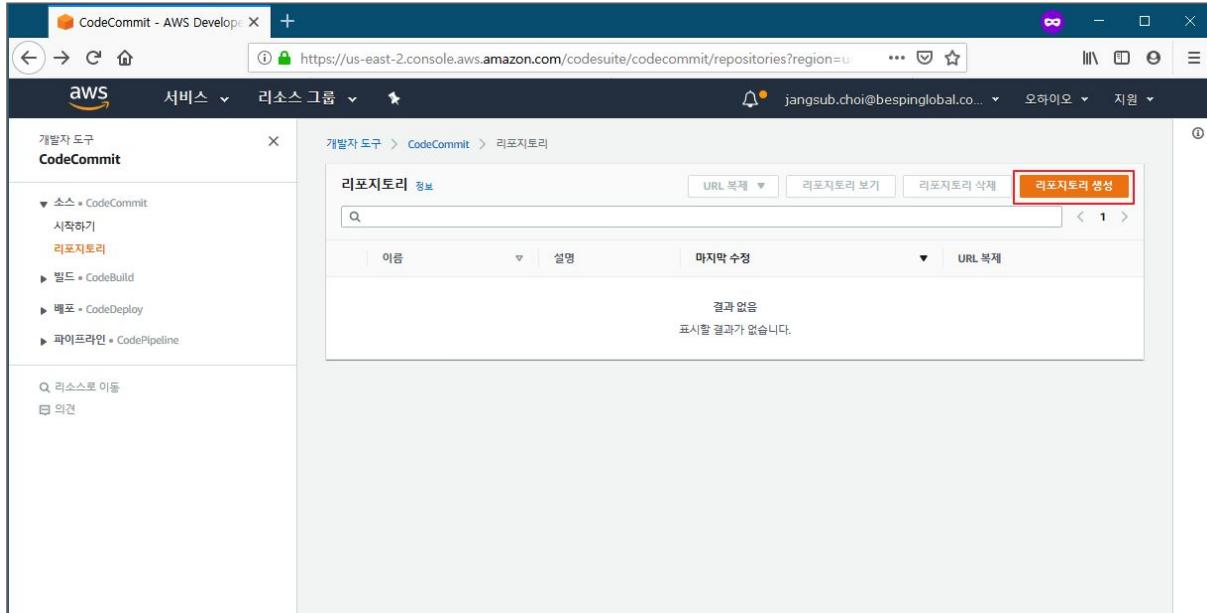
설정	설정 내용
시작 구성	jschoi-codedeploy-launch
목표 용량	0
최소	0
최대	4
클래식 로드 밸런서	①
대상 그룹	①
상태 검사 유예 기간	EC2
상태 검사 유예 기간	300
인스턴스 보호	①
종료 정책	① Default
임시 중지된 프로세스	①
배치 그룹	①
기본 유지	① 300
활성화된 지	①
생성 시간	Mon Sep 16 16:55:19 GMT+900 2019

## 2. AWS Codecommit

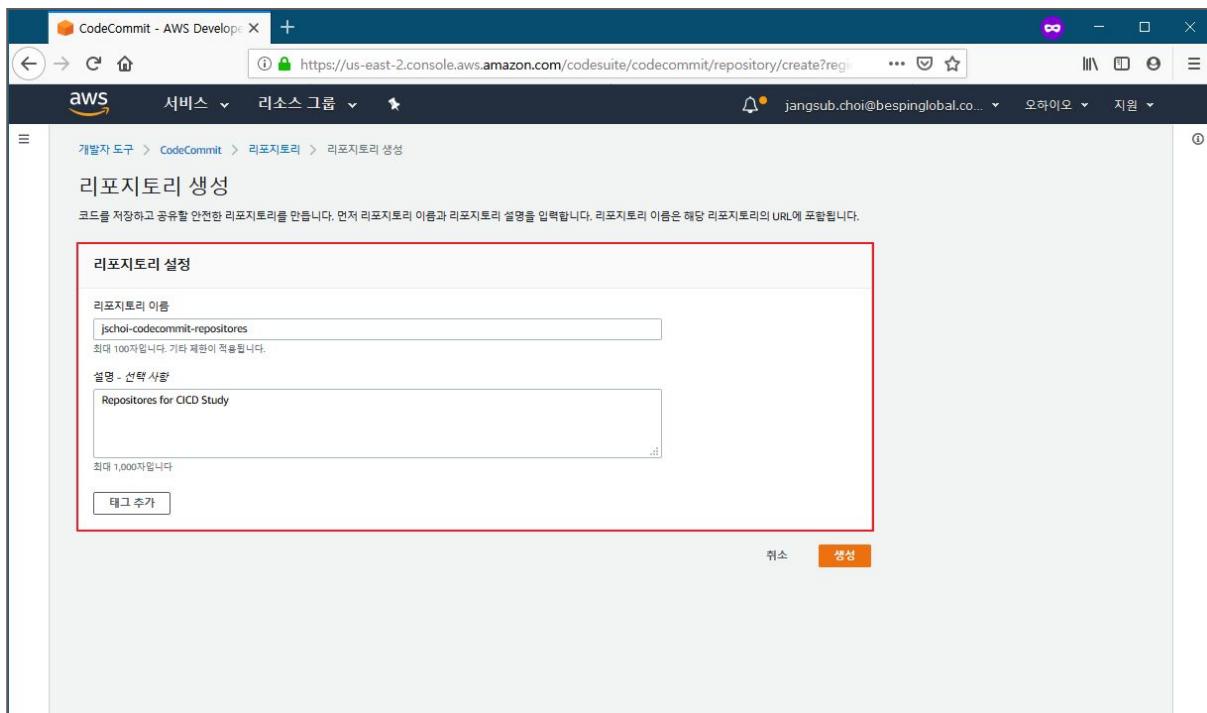
AWS CodeCommit은 안전한 Git 기반 리포지토리를 호스팅하는 완전관리형 소스 제어 서비스입니다.

### 2.1. AWS Codecommit Repositories

- AWS Web Console 내 Codecommit 페이지로 이동한 후 Repositories 를 생성 합니다.



- Repositories 에 대한 이름과 설명(선택 사항)을 지정하고 생성 합니다.



- Repositories 가 생성되면 IDE 툴에 대한 정보와 URL 을 제공 됩니다.

The screenshot shows the AWS CodeCommit console with a success message: "성공 리포지토리가 성공적으로 생성됨". The main content area displays the repository details for "jschoi-codecommit-repositories". It includes sections for "연결 단계" (Connection Steps) with "HTTPS" selected, "1단계: 사전 요구 사항" (Step 1: Prerequisites), "2단계: Git 자격 증명" (Step 2: Git credentials), and "3단계: 리포지토리 복제" (Step 3: Replicate repository). A command line input field contains "git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/jschoi-codecommit-repositories". Below this is a "파일 추가" (File Add) button. At the bottom, there's a note about cloning the repository to a local machine and a "파일 생성" (File Create) button.

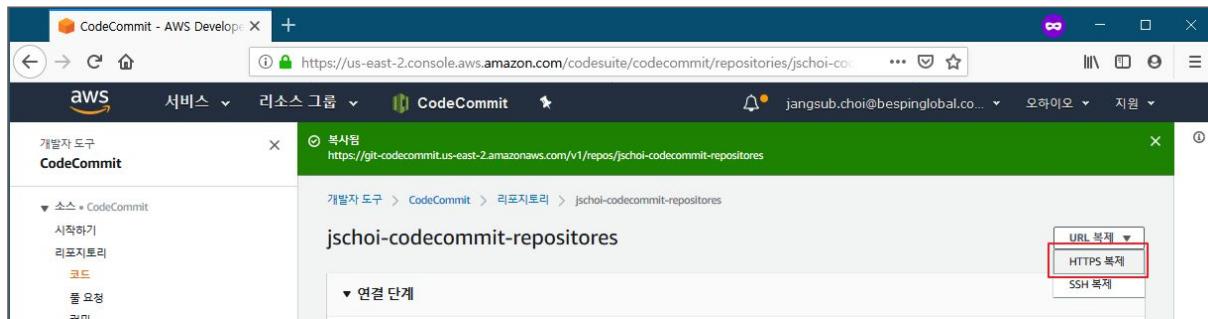
## 2.2. IDE Configuration for Repositories

AWS Codecommit Repositories 가 생성되면 사용자의 IDE 툴과 연결합니다.

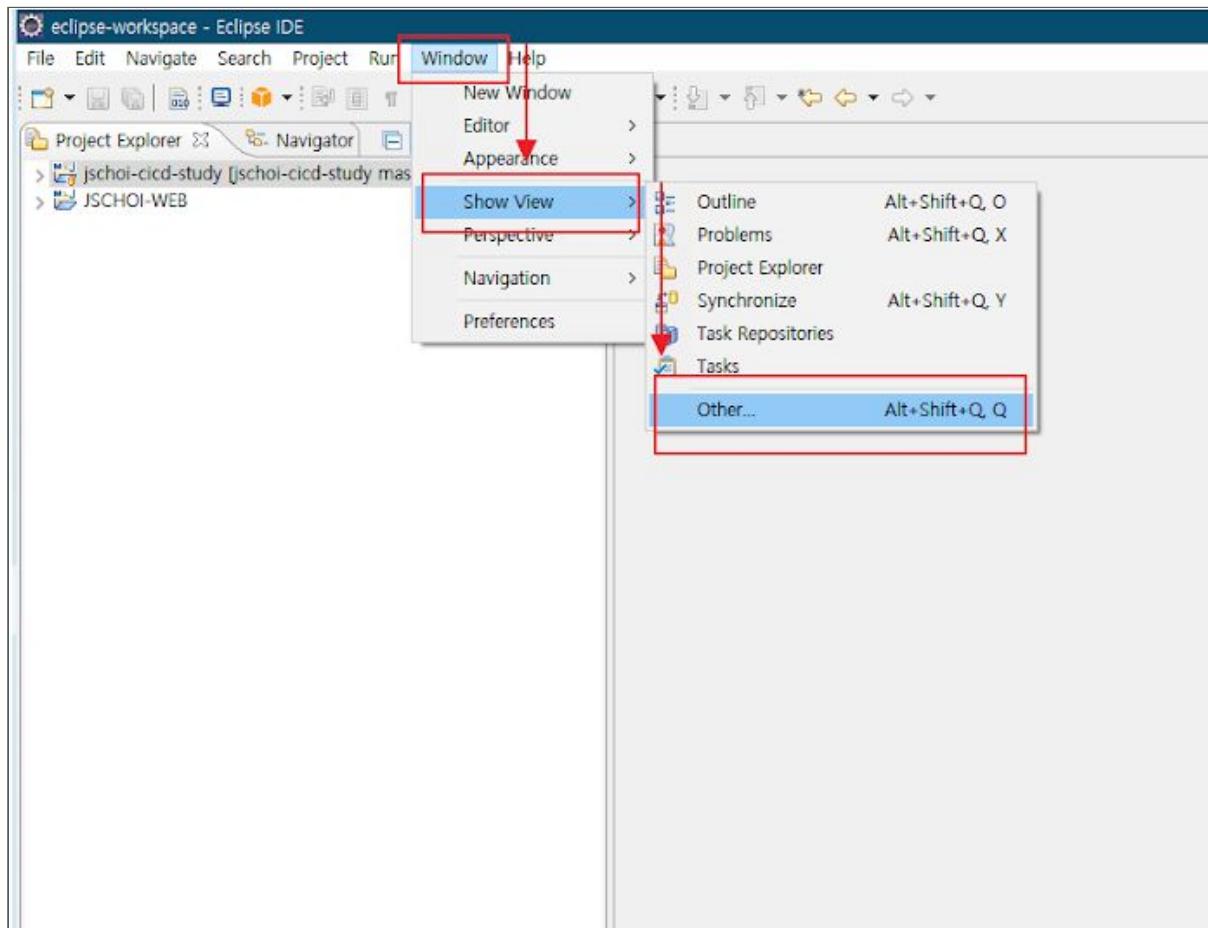
- AWS CodeCommit Repositories 에서 HTTPS 정보, User Git 정보를 확인후 연결합니다.

이번 테스트에서 CodeCommit Repositories 을 사용 위해서는 조건이 한가지 있으며, 반드시 Repositories의 최 상위에 소스코드를 업로드 하도록 설정해야 합니다.

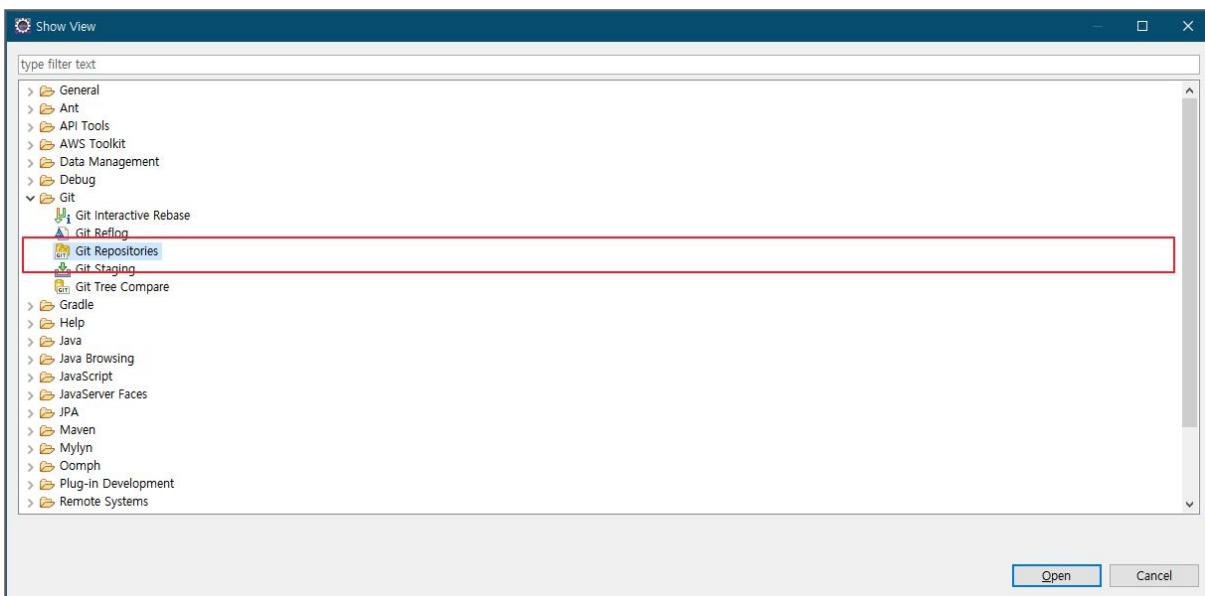
첨부 문서내 “IDE Project Configuration” 참고 합니다.



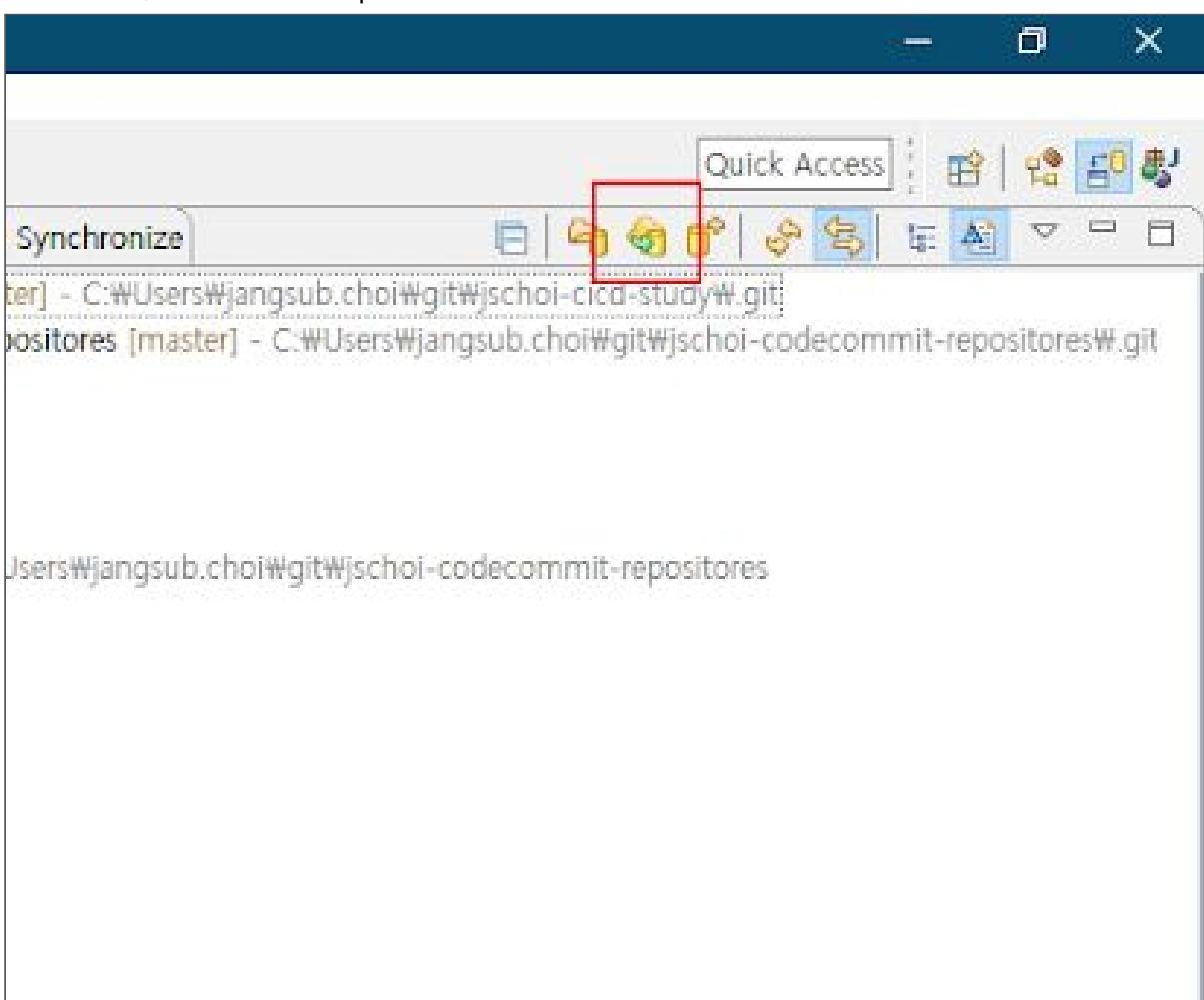
- IDE 툴 (이클립스)의 메뉴의 Windows > Show View > Other.. 를 클릭 합니다.



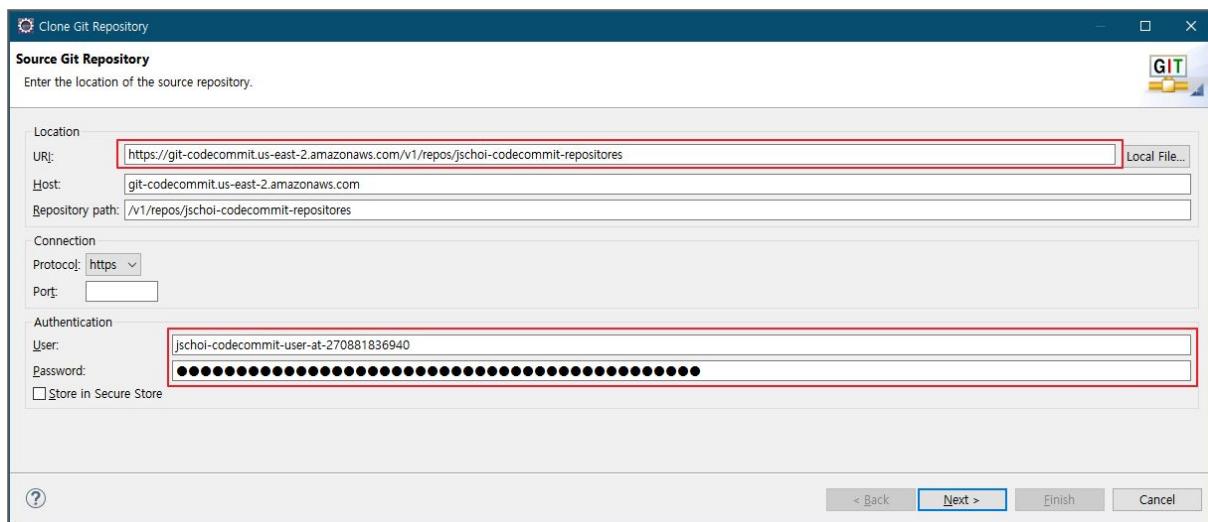
- Git > Git Repositories 를 선택 합니다.



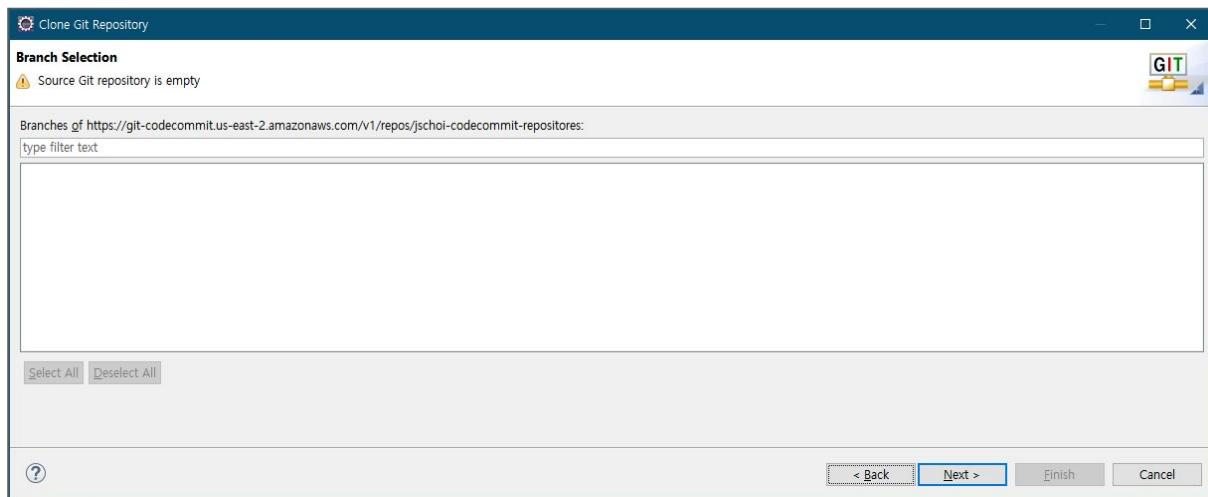
- 아이콘을 클릭하여 Repositories 연결을 진행 합니다.



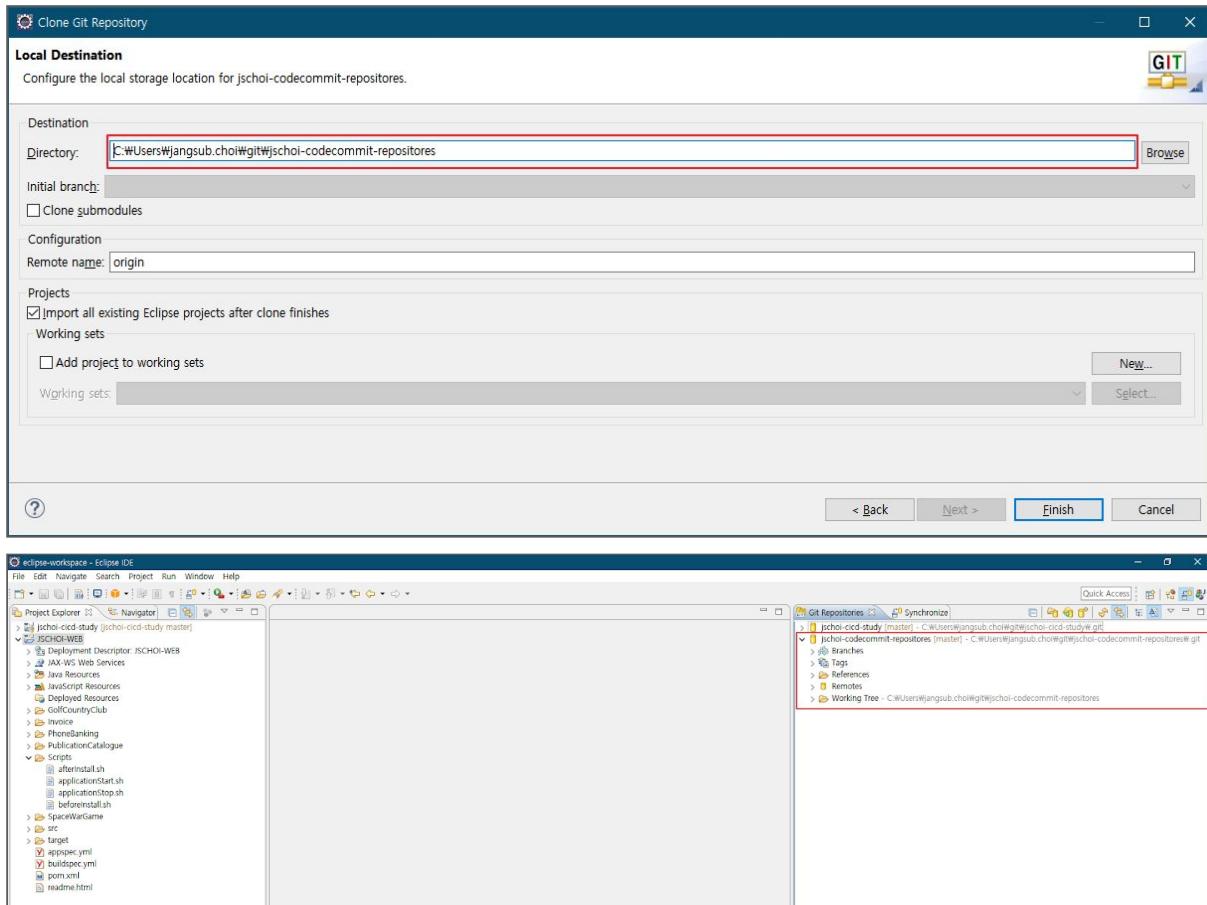
- URL 주소를 확인 후 IDE 를 (이클립스)에서 Repositories 를 연결 합니다.



- 아직 소스가 없기 때문에 아무런 브런치가 표시되지 않으며, 다음으로 진행 합니다.



- AWS CodeCommit Repositories 와 연결할 Local Path 를 확인후 연결 작업을 완료합니다.

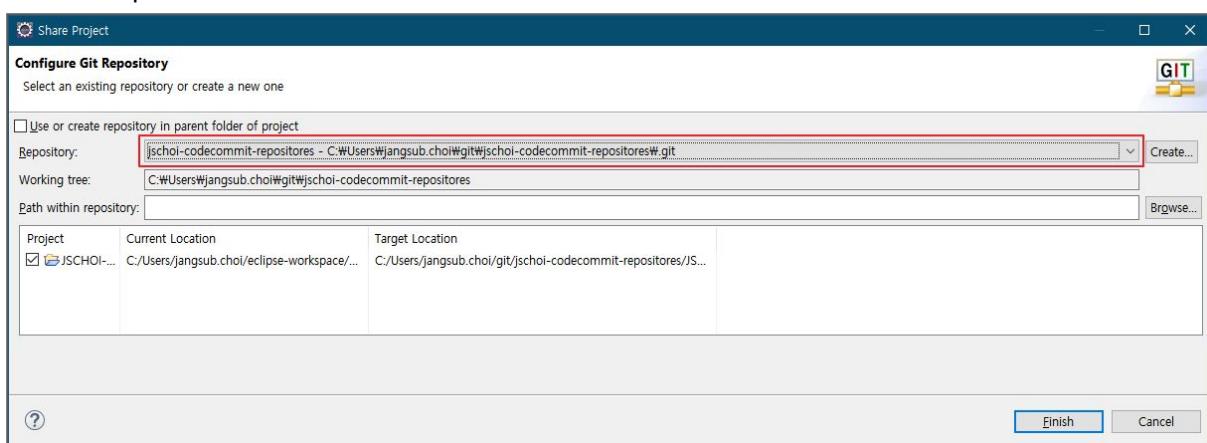


- 사용자의 Project 를 Git 과 연결하기 위해 다음 작업을 진행 합니다.

사용자의 Project 우클릭 > Team > Share Project... 를 클릭합니다.



- Repositories 를 지정하고 완료 합니다.

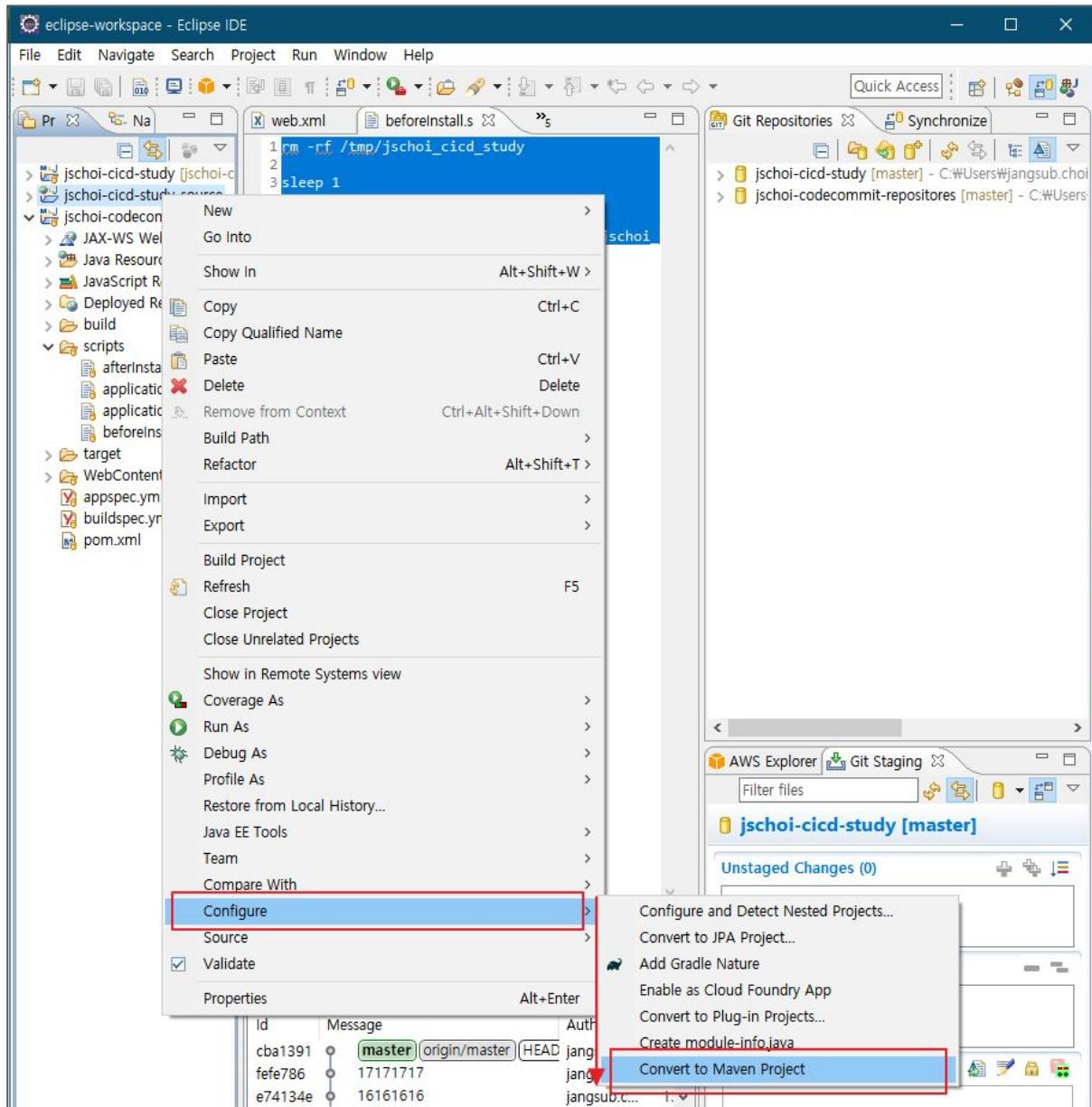


## 2.3. Source Commit

### 2.3.1. MAVEN Configuration

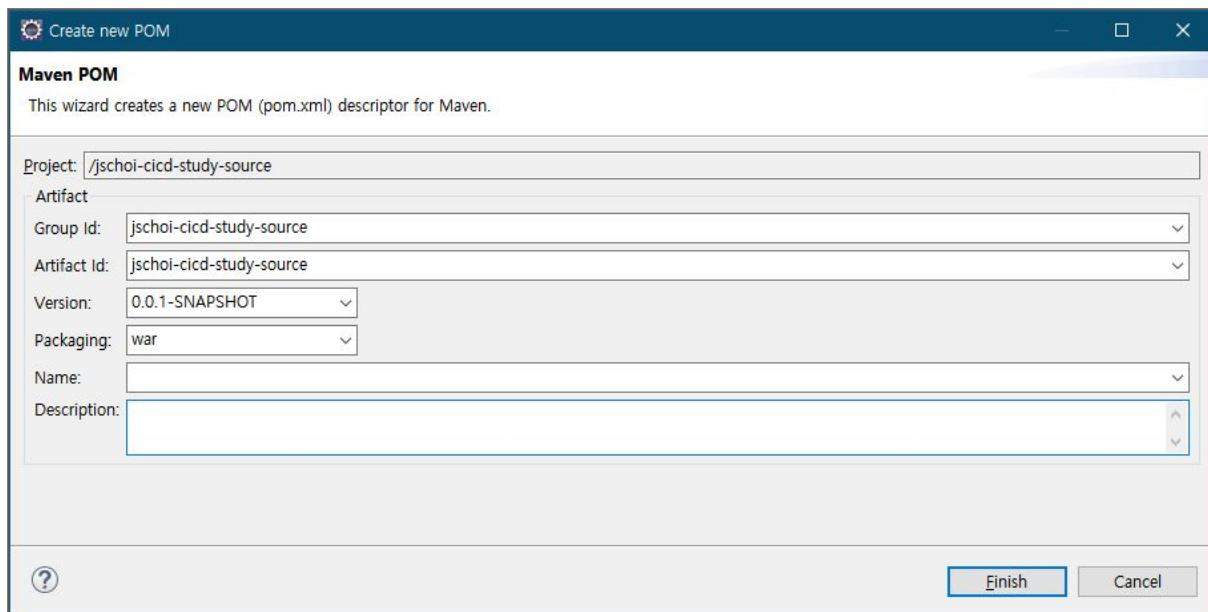
이 문서에서는 Maven<sup>1</sup> 을 사용하며, 다음 과정을 통해 Maven 으로 Convert 합니다.

- 프로젝트를 우클릭 후 **Configure > Convert to Maven Project** 를 클릭 합니다.

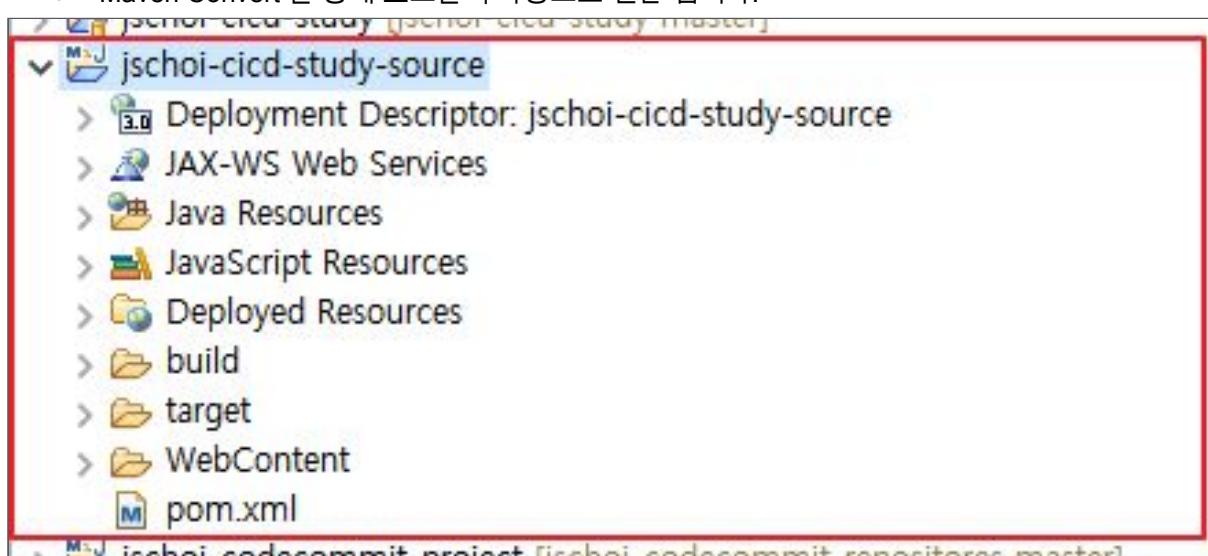


<sup>1</sup> Apache Maven - 자바용 프로젝트 관리 도구로 Apache Ant 대용으로 만들어 졌으며 Apache 라이선스로 배포되는 오픈 소스 소프트웨어입니다.

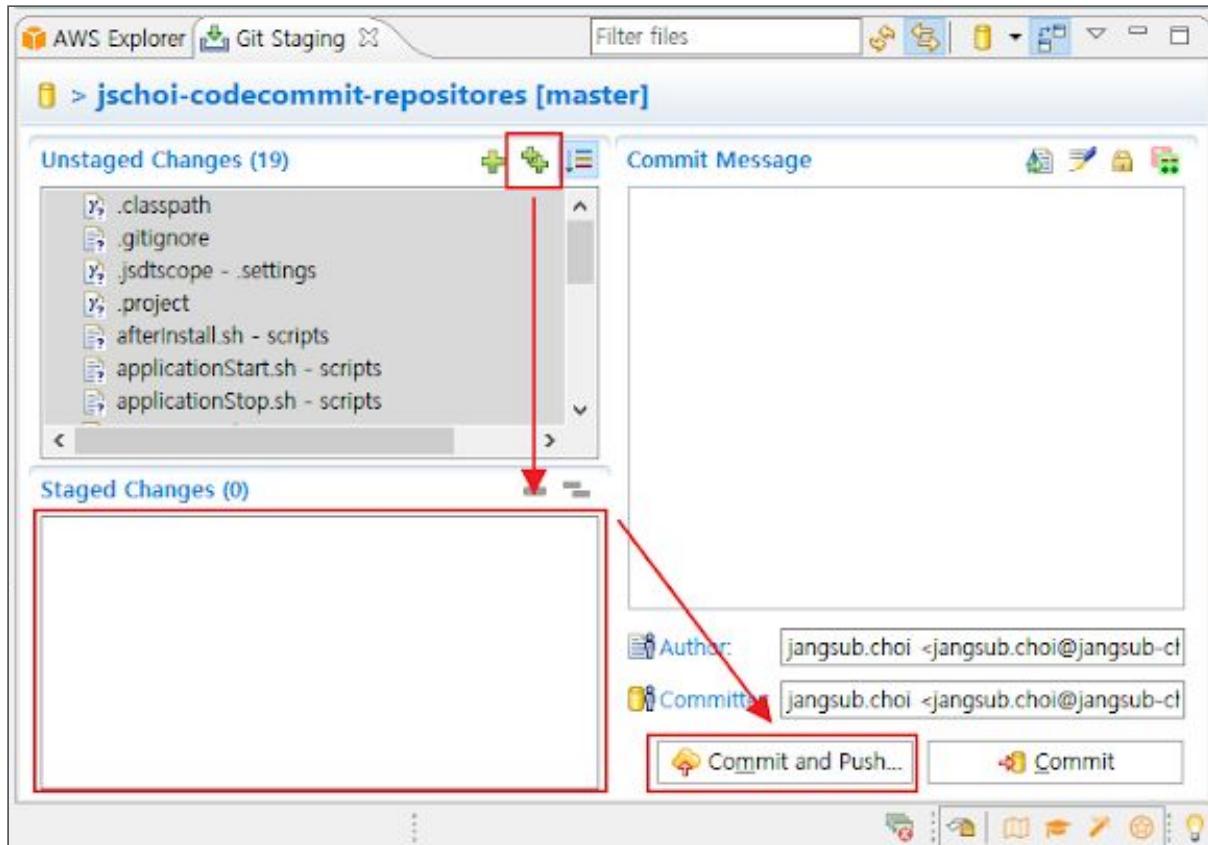
- Maven 사용을 위해 POM 파일이 생성되며 내용을 확인 후 진행 합니다.



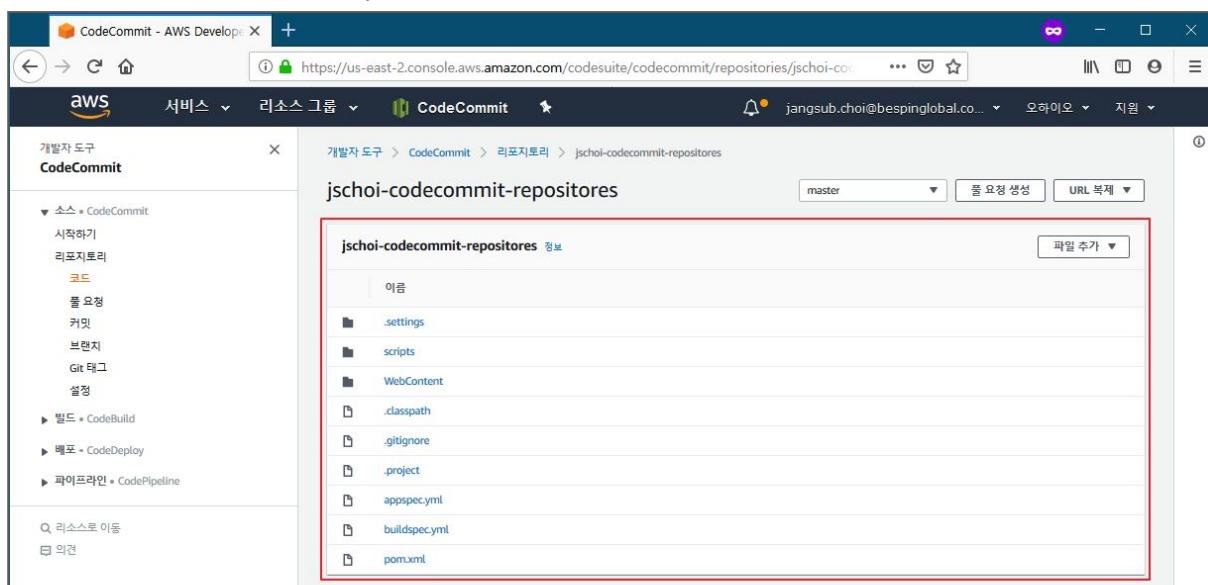
- Maven Convert 를 통해 소스들이 자동으로 변환 됩니다.



- 사용자의 소스가 변경되면 IDE 툴 (이클립스)은 이를 “Unstaged Changes” 상태로 등록하고 사용자가 “Staged Changes”으로 등록하고 “Commit and Push...”를 클릭합니다.



- AWS CodeCommit Repositories에서 업로드된 소스들을 확인할 수 있습니다.



### 3. AWS CodeBuild

- AWS CodeBuild 를 실행하기 전 buildspec.yml 을 생성해야 하며, 이는 CodeBuild 가 빌드를 실행하는 데 사용되는 YAML 형식의 빌드 명령 및 관련 설정의 모음입니다.

소스 코드의 일부로 빌드 사양을 포함할 수 있으며, 빌드 프로젝트를 생성할 때 빌드 사양을 정의할 수도 있습니다.

또한 빌드 사양을 소스 코드의 일부로 포함시키는 경우, 기본적으로 빌드 사양 파일의 이름은 buildspec.yml이어야 하며 소스 디렉터리의 루트에 있어야 합니다.

현재 지원되는 Buildspec 런타임 버전은 다음을 참고 합니다.

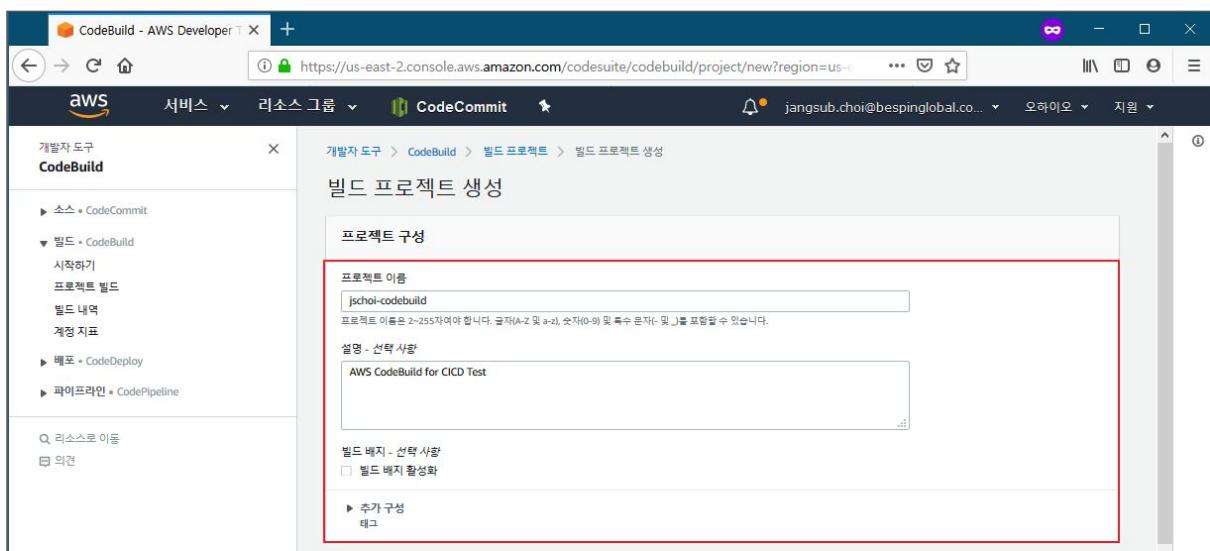
android	28	android : 28
docker	18	docker : 18
dotnet	2.2	dotnet : 2.2
golang	1.12	golang: 1.12
node.js	8, 10	nodejs : 8, nodejs : 10
java	openjdk8, openjdk11	java: openjdk8, java : openjdk11
php	7.3	php : 7.3
python	3.7	python : 3.7
ruby	2.6	ruby : 2.6

### 3.1. AWS CodeBuild Project

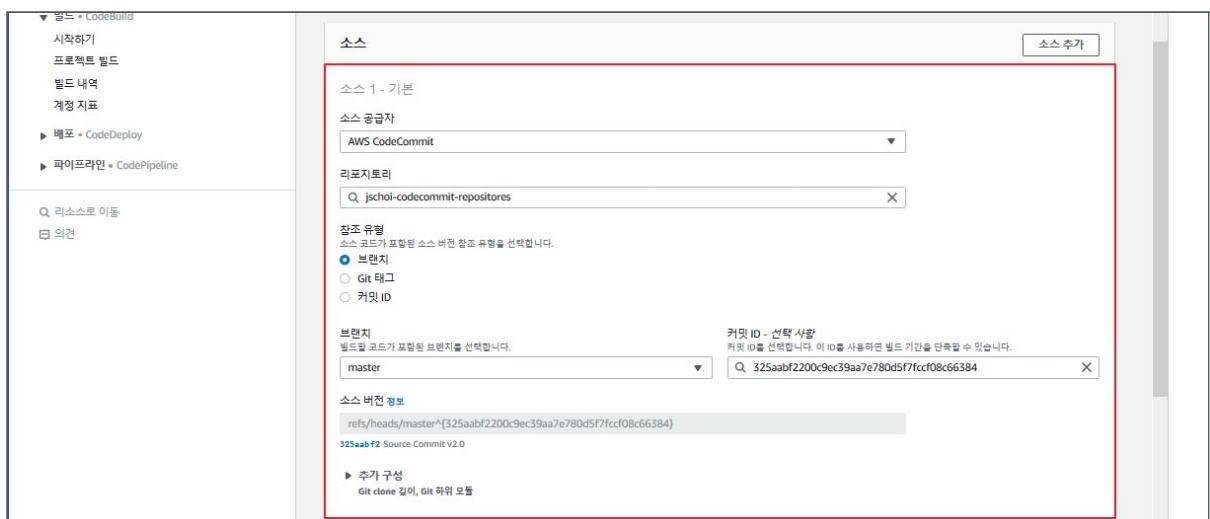
- AWS Web Console 내 AWS CodeBuild 페이지로 이동후 CodeBuild 를 생성 합니다.



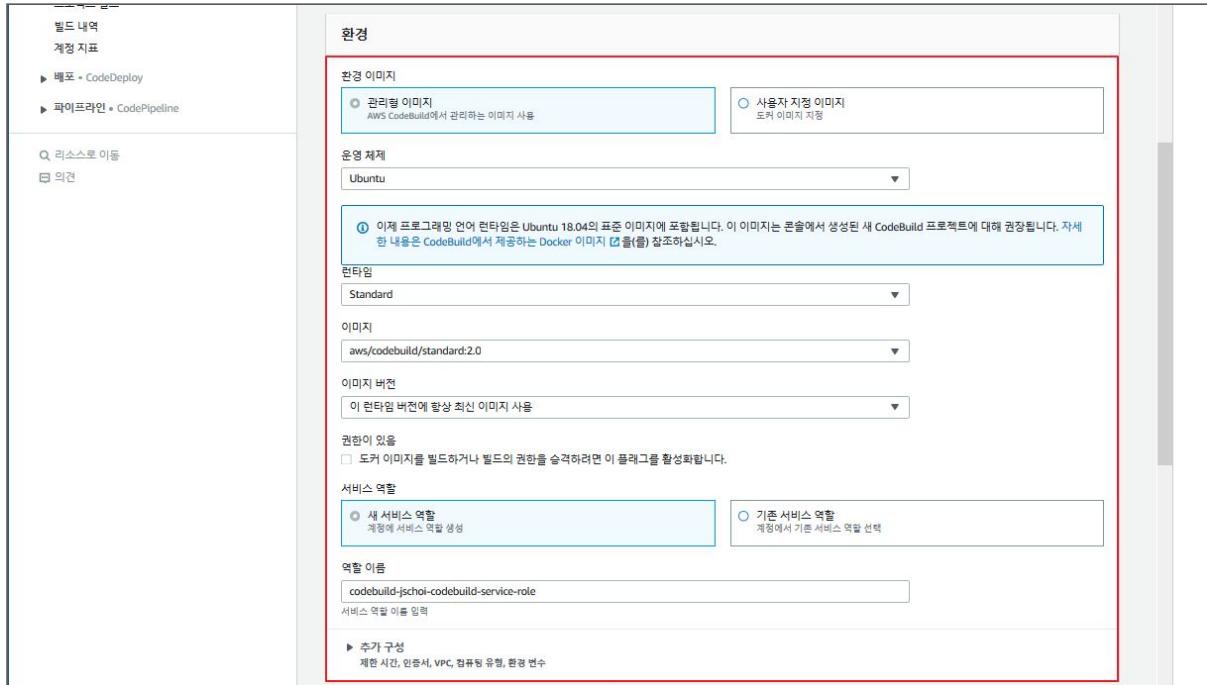
- 다음 사항들을 참고하여 AWS CodeBuild 를 생성 합니다.



- 프로젝트 이름 :** AWS CodeBuild 이름
- 설명 :** AWS CodeBuild 에 대한 설명



- **소스 공급자** : AWS CodeCommit 선택
- **리포지토리** : 사용자의 Repositories
- **참조 유형** : 브랜치 선택
- **브랜치** : master 선택
- **커밋 ID** : 커밋 ID를 선택



- **환경 이미지** : AWS CodeBuild 에서 관리하는 이미지 선택
- **운영 체제** : Ubuntu 선택
- **런타임** : Standard 선택
- **이미지** : aws/codebuild/standard2.0 선택
- **이미지 버전** : 이 런타임 버전에서 항상 최신 이미지 사용 선택
- **서비스 역할** : 새 서비스 역할
- **역할 이름** : (새 서비스 역할 선택시 자동 생성)



- **빌드 사양** : buildspec 파일 사용을 선택
- **buildspec 이름** : buildspec.yml 파일 이름을 지정

**아티팩트**

아티팩트 1 – 기본

**유형**

Amazon S3

테스트를 실행하거나 도커 이미지를 Amazon ECR에 넣는 경우 [아티팩트 읽기/쓰기]를 선택할 수 있습니다.

**버킷 이름**

Q jschoi-deploy X

이름

중복 아티팩트를 포함할 버킷에 있는 둘더 또는 앤惆 파일의 이름입니다. [추가 구성]에서 [아티팩트 패키징]을 사용하여 둘더를 사용할지 아니면 앤惆 파일을 사용할지 선택합니다. 이름을 제공하지 않으면 기본적으로 프로젝트 이름으로 설정됩니다.

jschoi-cicd-build.zip

**의미 제게 버전 관리 활성화**  
buildspec 파일에 지정된 아티팩트 이름 사용

**경로 - 선택 사용**

빌드 출력 ZIP 파일 또는 둘더의 경로입니다.

Build/

예: MyPath/MyArtifact.zip

**네임스페이스 유형 - 선택 사용**

없음

[빌드 ID]를 선택하여 빌드 출력 ZIP 파일 또는 둘더의 경로(예: MyPath/MyBuildID/MyArtifact.zip)에 빌드 ID를 삽입합니다. 또는 [없음]을 선택합니다.

**아티팩트 패키징**

**Zip**  
AWS CodeBuild가 아티팩트를 앤惆 파일로 업로드합니다(이 앤惆 파일은 지정된 버킷에 넣습니다).

**아티팩트 암호화 제거**  
아티팩트를 사용하여 정적 웹 사이트를 게시하거나 다른 사용자와 팀원을 공유할 경우 암호화를 제거합니다.

▶ 추가 구성  
개시, 암호화 키

- **유형 : Amazon S3 를 선택**
- **버킷 이름 : 사용자의 Bucket 이름을 선택**
- **이름 : 아티팩트 파일이 생성될 이름을 지정**
- **경로 : Build 출력 ZIP 파일의 경로를 지정**
- **아티팩트 패키징 : Zip 을 선택**

- **CloudWatch 로그 를 활성화 하여 Build 시 로그를 실시간으로 확인합니다.**

**로그**

CloudWatch

**CloudWatch 로그 - 선택 사용**  
이 옵션을 선택하면 빌드 출력 로그가 CloudWatch에 업로드됩니다.

그룹 이름

codebuild-jschoi-codebuild-service-role

S3

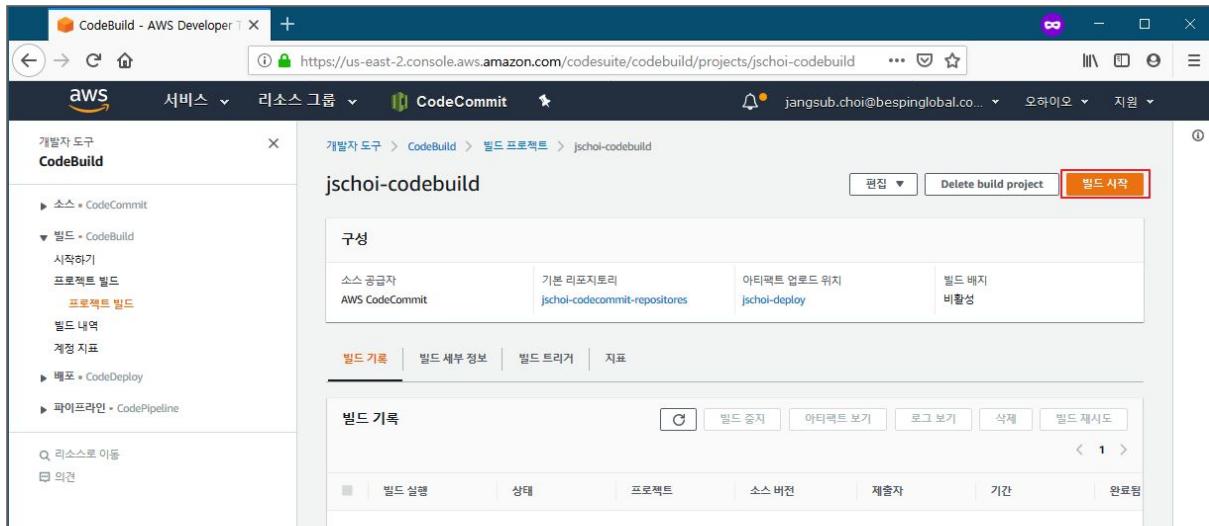
**S3 로그 - 선택 사용**  
이 옵션을 선택하면 빌드 출력 로그가 S3에 업로드됩니다.

취소 **빌드 프로젝트 생성**

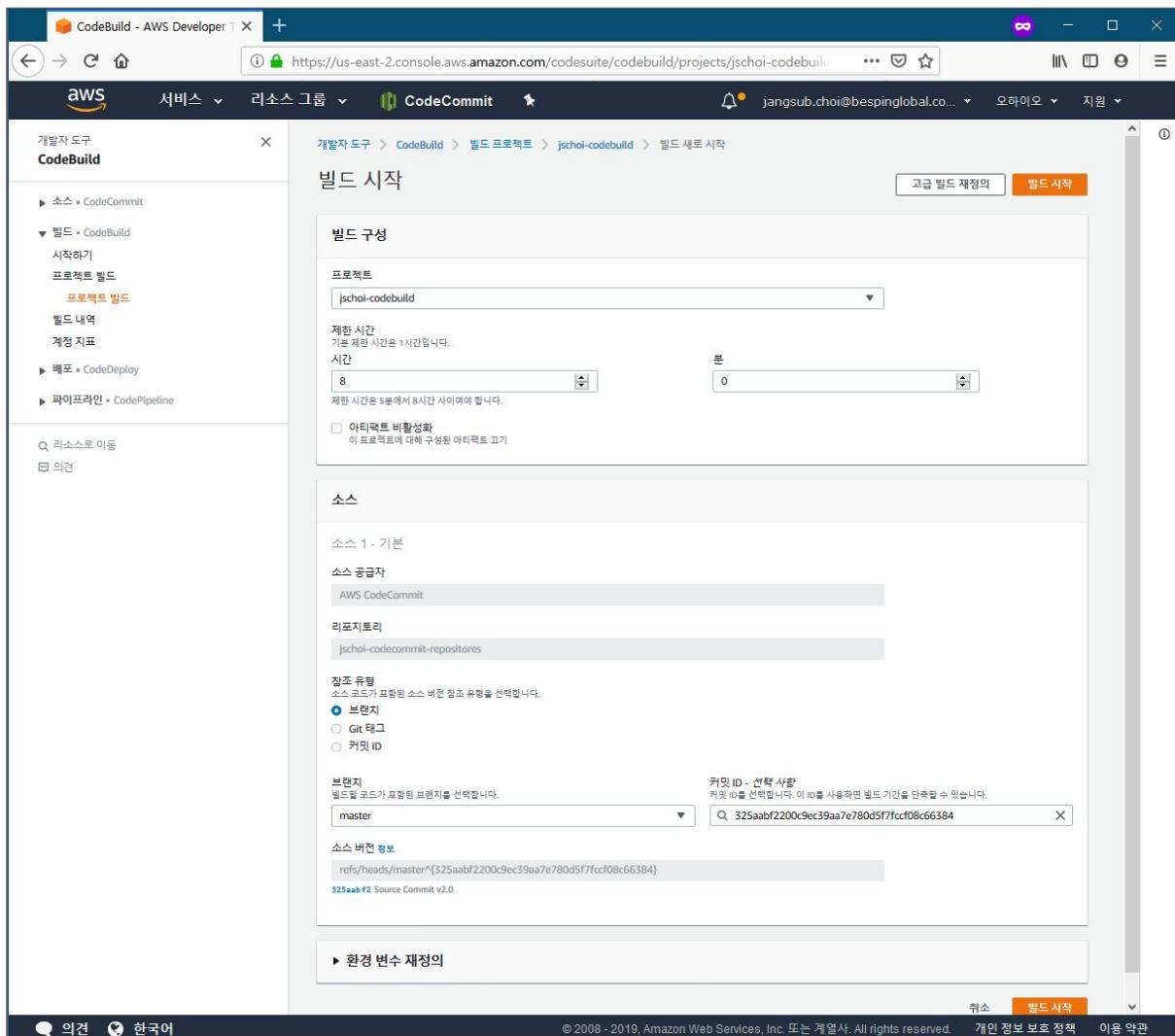
© 2008 - 2019, Amazon Web Services, Inc. 또는 계열사. All rights reserved. 개인 정보 보호 정책 이용 약관

의견 한국어

- AWS CodeBuild 생성이 완료되면 “빌드 시작”을 클릭하여 Build를 시작합니다.



The screenshot shows the AWS CodeBuild console for the project 'jschoi-codebuild'. On the left, the navigation pane shows 'CodeCommit' selected under '소스'. The main area displays the build configuration with 'AWS CodeCommit' as the source provider and 'jschoi-codecommit-repositories' as the repository. At the top right, there is a red box around the '빌드 시작' (Start Build) button.

The screenshot shows the 'Build Start' configuration page for the same project. It includes fields for 'Build Configuration' (using the 'jschoi-codebuild' profile), 'Build Duration' (set to 8 minutes), and 'Source' settings (using 'AWS CodeCommit' and 'jschoi-codecommit-repositories'). At the bottom right, there is a red box around the '빌드 시작' (Start Build) button.

- Build가 실행되고 하단에서 Build 로그를 확인할 수 있습니다.

The screenshot shows the AWS CodeBuild console with a successful build start message. The build status is '진행 중' (In Progress). The 'Build Log' button is highlighted with a red box.

- “테일 로그”를 클릭하면 실시간으로 마지막 로그들을 확인할 수 있습니다.

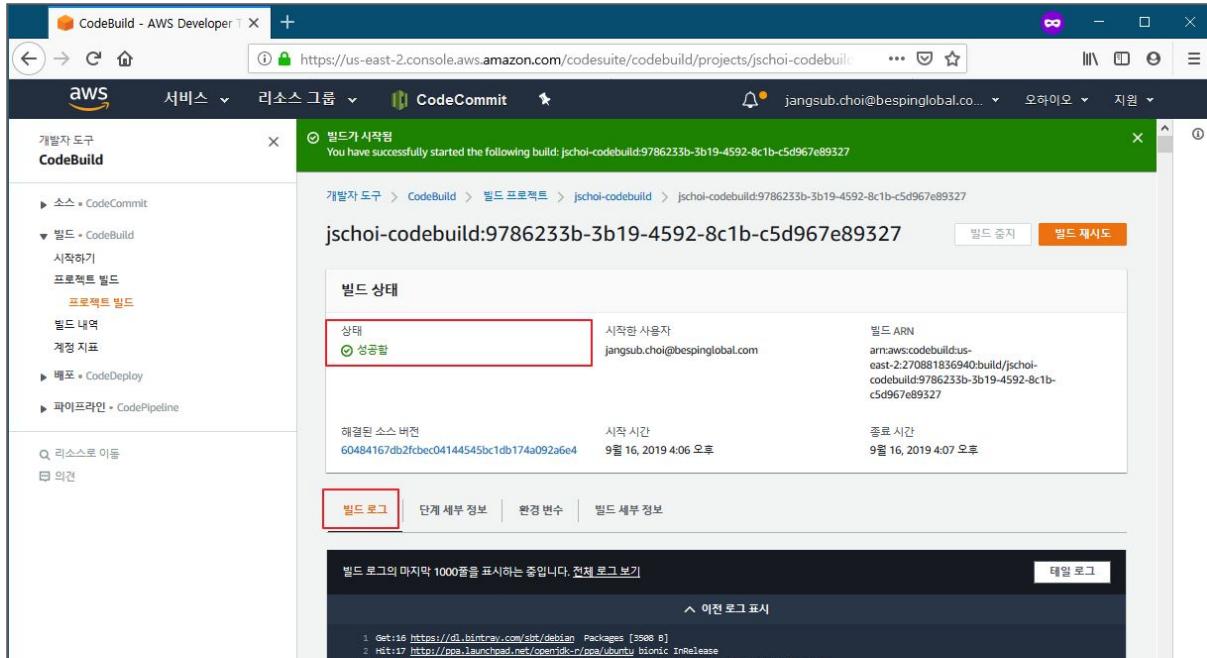
The screenshot shows the AWS CodeBuild console with the 'Tail Log' view open. The log output shows Maven dependency download progress. The 'Tail Log' button is highlighted with a red box.

```

753 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/1.1/maven-filtering-1.1.pom (5.8 kB at 223 kB/s)
753 Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/17/maven-shared-components-17.pom
754 Progress (1): 2.2/8.7 kB
755 Progress (1): 5.6/8.7 kB
756 Progress (1): 7.8/8.7 kB
757 Progress (1): 8.7 kB
758
758 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/17/maven-shared-components-17.pom (8.7 kB at 322 kB/s)
768 Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.pom
761 Progress (1): 2.2/26 kB
762 Progress (1): 5.6/26 kB
763 Progress (1): 7.8/26 kB
764 Progress (1): 11/26 kB
765 Progress (1): 13/26 kB
766 Progress (1): 16/26 kB
767 Progress (1): 19/26 kB
768 Progress (1): 21/26 kB
769 Progress (1): 24/26 kB
770 Progress (1): 26 kB
771
772 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/maven-parent-21.pom (26 kB at 941 kB/s)
773 Downloading from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom
774 Progress (1): 2.2/15 kB
775 Progress (1): 5.6/15 kB
776 Progress (1): 7.8/15 kB
777 Progress (1): 11/15 kB
778 Progress (1): 13/15 kB
779 Progress (1): 15 kB
780
781 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom (15 kB at 569 kB/s)
782 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/1.5.15/plexus-utils-1.5.15.pom
783 Progress (1): 2.2/6.8 kB
784 Progress (1): 5.6/6.8 kB
785 Progress (1): 6.8 kB
786
787 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-util/1.5.15/plexus-util-1.5.15.pom (6.8 kB at 263 kB/s)
788 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-2.0.2/plexus-2.0.2.pom
789 Progress (1): 5.6/11 kB
790 Progress (1): 7.8/11 kB
791 Progress (1): 10/11 kB
792 Progress (1): 11/12 kB
793 Progress (1): 12 kB
794
795 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/2.0.2/plexus-2.0.2.pom (12 kB at 363 kB/s)
796 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.12/plexus-interpolation-1.12.pom
797 Progress (1): 889 B
798
799 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.12/plexus-interpolation-1.12.pom (889 B at 36 kB/s)
800 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-components/1.1.14/plexus-components-1.1.14.pom
801 Progress (1): 2.2/5.8 kB
802 Progress (1): 5.6/5.8 kB
803 Progress (1): 5.8 kB
804
805 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-components/1.1.14/plexus-components-1.1.14.pom (5.8 kB at 216 kB/s)
806 Downloading from central: https://repo.maven.apache.org/maven2/org/songge/plexus/plexus-build-api/0.0.4/plexus-build-api-0.0.4.pom
807

```

- Build 가 정상적으로 완료되면 다음과 같이 확인할 수 있습니다.

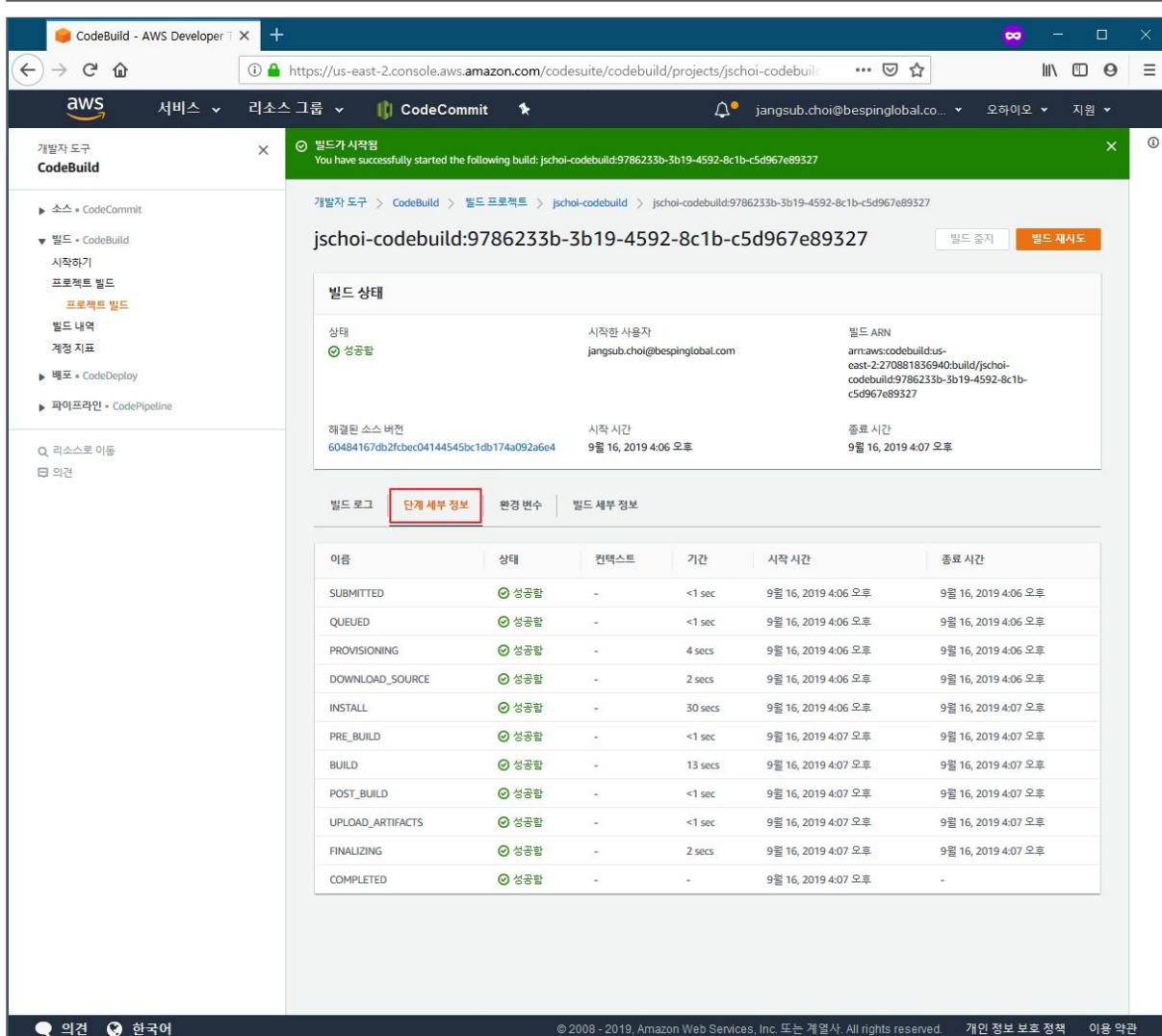


The screenshot shows the AWS CodeBuild console with a successful build summary. The build ID is jschoi-codebuild:9786233b-3b19-4592-8c1b-c5d967e89327. The build state is listed as '성공' (Success). The build ARN is arn:aws:codebuild:us-east-2:270881836940:build/jschoi-codebuild:9786233b-3b19-4592-8c1b-c5d967e89327. The build log section shows the last 1000 lines of the log, which includes several GET requests for package files from various repositories.

```

1 Get:16 https://d1.bintray.com/sbt/debian Packages [3988 B]
2 Hit:17 http://ppa.launchpad.net/openjdk-7/ubuntu bionic InRelease
3 Get:18 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [8385 B]

```

The screenshot shows the AWS CodeBuild console with detailed build logs. The build ID is jschoi-codebuild:9786233b-3b19-4592-8c1b-c5d967e89327. The build state is listed as '성공' (Success). The build ARN is arn:aws:codebuild:us-east-2:270881836940:build/jschoi-codebuild:9786233b-3b19-4592-8c1b-c5d967e89327. The build log section shows the last 1000 lines of the log, which includes several GET requests for package files from various repositories. The detailed log table lists the following steps and their details:

이름	상태	컨택스트	기간	시작 시간	종료 시간
SUBMITTED	성공	-	<1 sec	9월 16, 2019 4:06 오후	9월 16, 2019 4:06 오후
QUEUED	성공	-	<1 sec	9월 16, 2019 4:06 오후	9월 16, 2019 4:06 오후
PROVISIONING	성공	-	4 secs	9월 16, 2019 4:06 오후	9월 16, 2019 4:06 오후
DOWNLOAD_SOURCE	성공	-	2 secs	9월 16, 2019 4:06 오후	9월 16, 2019 4:06 오후
INSTALL	성공	-	30 secs	9월 16, 2019 4:06 오후	9월 16, 2019 4:07 오후
PRE_BUILD	성공	-	<1 sec	9월 16, 2019 4:07 오후	9월 16, 2019 4:07 오후
BUILD	성공	-	13 secs	9월 16, 2019 4:07 오후	9월 16, 2019 4:07 오후
POST_BUILD	성공	-	<1 sec	9월 16, 2019 4:07 오후	9월 16, 2019 4:07 오후
UPLOAD_ARTIFACTS	성공	-	<1 sec	9월 16, 2019 4:07 오후	9월 16, 2019 4:07 오후
FINALIZING	성공	-	2 secs	9월 16, 2019 4:07 오후	9월 16, 2019 4:07 오후
COMPLETED	성공	-	-	9월 16, 2019 4:07 오후	-

## 4. AWS CodeDeploy

AWS CodeDeploy는 Amazon EC2 인스턴스, 온프레미스 인스턴스, 서비스 Lambda 함수 또는 Amazon ECS 서비스로 애플리케이션 배포를 자동화하는 배포 서비스입니다.

### 4.1. AWS CodeDeploy Application

- AWS Web Console 내 CodeDeploy 페이지로 이동한 후 CodeDeploy 를 생성합니다.

- 애플리케이션 이름** : CodeDeploy 애플리케이션의 이름
- 컴퓨팅 플랫폼** : EC2/온프레미스를 선택

## 4.2. AWS CodeDeploy Deploy Group

- CodeDeploy 애플리케이션이 생성되면 “배포 그룹”을 생성합니다.

The screenshot shows the AWS CodeDeploy console with the URL <https://us-east-2.console.aws.amazon.com/codesuite/codedeploy/applications/jschoi-codedeploy-application>. The left sidebar shows 'CodeDeploy' under 'Services'. The main content area displays 'Application details' for 'jschoi-codedeploy-application'. In the 'Deploy Groups' section, there is a table with one row. A red box highlights the 'Create Deploy Group' button at the bottom right of the table.

The screenshot shows the 'Create Deploy Group' wizard step 1, 'Deploy Group Name'. It lists the application 'jschoi-codedeploy-application' and its environment 'EC2/운프레미스'. The 'Deploy Group Name' input field contains 'jschoi-deploy-group' and is highlighted with a red box.

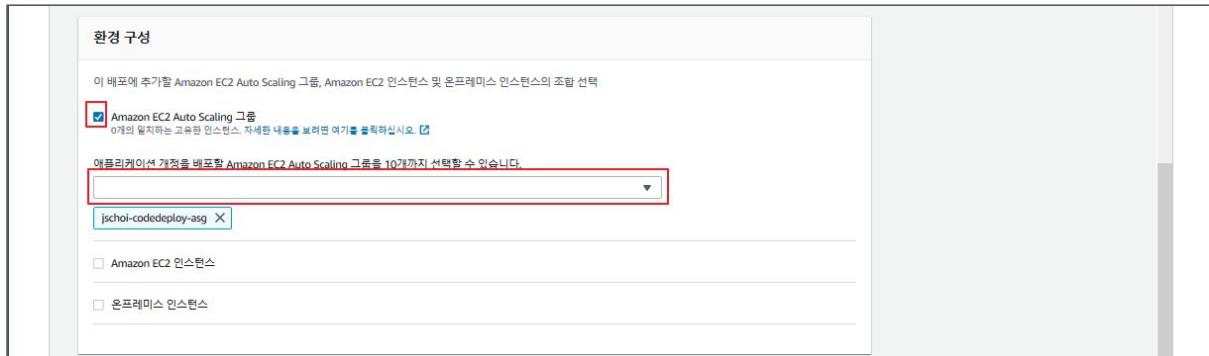
- 배포 그룹 이름 입력 : 배포 그룹의 이름을 지정

The screenshot shows the 'Service Accounts' section of the 'Create Deploy Group' wizard. It lists the role 'arn:aws:iam::270881836940:role/jschoi-codedeploy-role' and includes a note: 'AWS CodeDeploy가 대상 인스턴스에 액세스하도록 허용하는 CodeDeploy 권한이 있는 서비스 역할을 입력합니다.' A red box highlights the input field containing the ARN.

- 서비스 역할 입력 : AWS CodeDeploy가 대상 인스턴스에 액세스하도록 허용하는 CodeDeploy 권한이 있는 서비스 역할을 입력



- 배포 유형 : 현재 위치 (In-place)를 선택



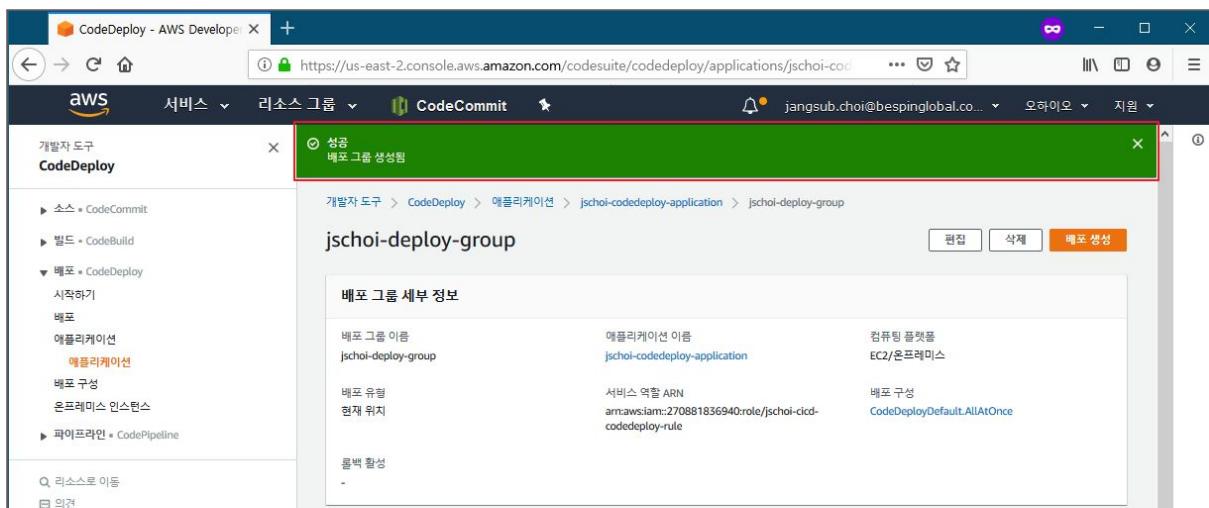
- Amazon EC2 Auto Scaling 그룹 을 체크한 후 Auto Scaling Group을 지정합니다.



- 배포 구성 : CodeDeployDefault.AllAtOnce 를 선택



- 로드 밸런서는 테스트 하지 않으므로 선택하지 않고 배포 그룹을 생성 합니다.



## 4.3. AWS CodeDeploy Deploy

- CodeDeploy 배포 그룹이 생성되면 “배포”를 생성합니다.

The screenshot shows the AWS CodeDeploy console with the URL <https://us-east-2.console.aws.amazon.com/codesuite/codedeploy/applications/jschoi-codedeploy-application/deployment-groups/jschoi-deploy-group>. The left sidebar shows navigation for CodeDeploy, including '소스' (CodeCommit), '빌드' (CodeBuild), '배포' (CodeDeploy), and '파이프라인' (CodePipeline). The main content area displays the 'Deployment Groups' configuration for the 'jschoi-deploy-group'. It includes sections for 'Deployment Group Details' (Deployment group name: jschoi-deploy-group, Application name: jschoi-codedeploy-application, Compute Platform: EC2/OnPremises), 'Service ARN': arn:aws:iam::270881836940:role/jschoi-cicd-codedeploy-rule, and 'Environment Configuration: Amazon EC2 Auto Scaling Group' (Name: jschoi-codedeploy-asg). The 'Triggers' section is empty. A red box highlights the 'Create Deployment' button at the top right of the main content area.

- 개정 위치는 아티팩트가 생성되는 경로를 지정 합니다.

The screenshot shows the AWS CodeDeploy console with the URL <https://us-east-2.console.aws.amazon.com/codesuite/codedeploy/application/jschoi-codedeploy-application/deployment-groups/jschoi-deploy-group/create>. The left sidebar shows navigation for CodeDeploy, including '소스' (CodeCommit), '빌드' (CodeBuild), '배포' (CodeDeploy), and '파이프라인' (CodePipeline). The main content area shows the 'Create deployment' wizard. The 'Deployment Settings' step is active, with the 'Application' set to 'jschoi-codedeploy-application' and the 'Deployment Group' set to 'jschoi-deploy-group'. Under 'Deployment Type', 'Deploy to Lambda Function' is selected. In the 'Deployment Artifacts' section, 'Application artifacts stored in Amazon S3' is selected, and the 'Artifact location' field contains the value 's3://jschoi-codebuild/Build/jschoi-cicd-build.zip'. Other options like 'GitHub' are shown as unselected radio buttons. A red box highlights the 'Artifact location' input field.

- 배포 실패시 상태 복구를 위한 롤백을 설정한 후 배포를 생성 합니다.

배포 설명  
배포 설명 - 선택 사용  
배포에 대한 단락된 설명을 추가합니다.

추가 배포 동작 설정  
ApplicationStop 수령 주기 이벤트 실패 - 선택 사용  
배포 그룹 이름 일련  
□ 인스턴스에서 이 수령 주기 이벤트가 실패하는 경우 해당 인스턴스에 대한 배포에 실패 안 할

콘텐츠 옵션 - 선택 사용  
배포하는 대상 인스턴스의 파일이 애플리케이션 개정의 파일과 동일한 이름인 경우 수행할 작업 선택

배포 실패  
오류가 보고되고 배포 상태가 실패로 변경됩니다.

콘텐츠 업데이쓰기  
애플리케이션 개정의 파일을 인스턴스의 대상 위치에 복사하고 이전 파일을 대체합니다.

콘텐츠 유지  
애플리케이션 개정의 파일을 인스턴스에 복사하지 않습니다. 기존 파일은 대상 위치에 유지되고 새 배포의 일부로 처리됩니다.

▶ 배포 그룹 재정의

▼ 롤백 구성 재정의  
이 배포에 대해서만 배포 그룹 롤백 설정을 재정의할 수 있습니다.  
 배포에 실패하는 경우 롤백  
 경보 임계값이 충족되는 경우 롤백  
 롤백 비활성화

취소 **배포 만들기**

- 배포가 생성되면 자동으로 Deploy 가 진행되고 결과를 확인할 수 있습니다.

aws 서비스 리소스 그룹 CodeCommit

개발자 도구 CodeDeploy

소스 + CodeCommit  
빌드 + CodeBuild  
배포 + CodeDeploy  
서비스  
배포  
애플리케이션  
배포 구성  
온프레미스 인스턴스  
파이프라인 + CodePipeline

리소스로 이동 의견

https://us-east-2.console.aws.amazon.com/codesuite/codedeploy/deployments/d-R5EAW01IN?region=us-east-2

d-R5EAW01IN

배포 상태  
인스턴스에 애플리케이션 설치 중 1/1개의 인스턴스가 준비되었습니다. 성공

배포 세부 정보  
애플리케이션 jchoi-codedeploy-application  
배포 구성 CodeDeployDefault.AllAtOnce  
배포 ID d-R5EAW01IN  
배포 그룹 jchoi-deploy-group  
상태 성공  
시작 user

개정 세부 정보  
개정 위치 s3://jchoi-codedeploy/build/jchoi-coded-build.zip  
개정 수정 7분 전  
설명 Application revision registered by Deployment ID: d-GANIXQON

배포 수령 주기 이벤트  
인스턴스 ID 기간 상태 가장 최근 이벤트 이벤트 시작 시간 종료 시간  
i-0abfa5c78b050cf7f 6초 성공 ValidateService View events 9월 16, 2019 5:34 오후 9월 16, 2019 5:34 오후

취소 **배포 만들기**

The screenshot shows the AWS CodeDeploy console with the following details:

- Deployment ID:** d-R5EAW01N
- Deployment Name:** jschoi-codedeploy-application
- Deployment Group:** jschoi-deploy-group
- Revision:** Application revision registered by Deployment ID: d-6N3E00N
- Logs:** Application revision registered by Deployment ID: d-6N3E00N
- Events:**

이벤트	기간	상태	오류 코드	시작 시간	종료 시간
ApplicationStop	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
DownloadBundle	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
BeforeInstall	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
Install	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
AfterInstall	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
ApplicationStart	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후
ValidateService	1초 미만	성공	-	9월 16, 2019 5:34 오후	9월 16, 2019 5:34 오후

- 배포가 정상적으로 완료되고 Auto Scaling Group 과 실제 EC2 에서도 확인할 수 있습니다.

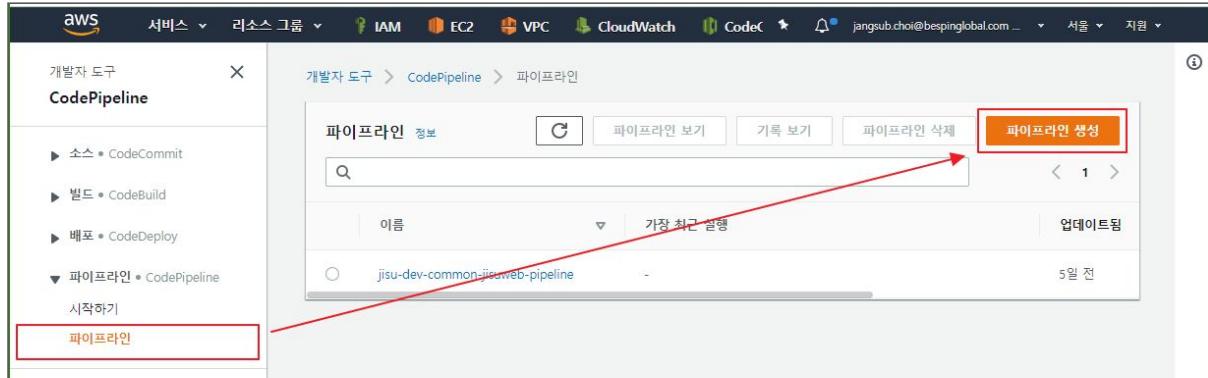
The screenshot shows the AWS Auto Scaling console with the following details:

- Auto Scaling Group:** jschoi-codedeploy-asg
- Launch Task:** jschoi-codedeploy-launch
- InstanceState:** Launching a new EC2 instance: i-0a8a5e78b05bcf77f
- Timestamps:** 2019 September 16 17:28:56 UTC+9, 2019 September 16 17:29:28 UTC+9

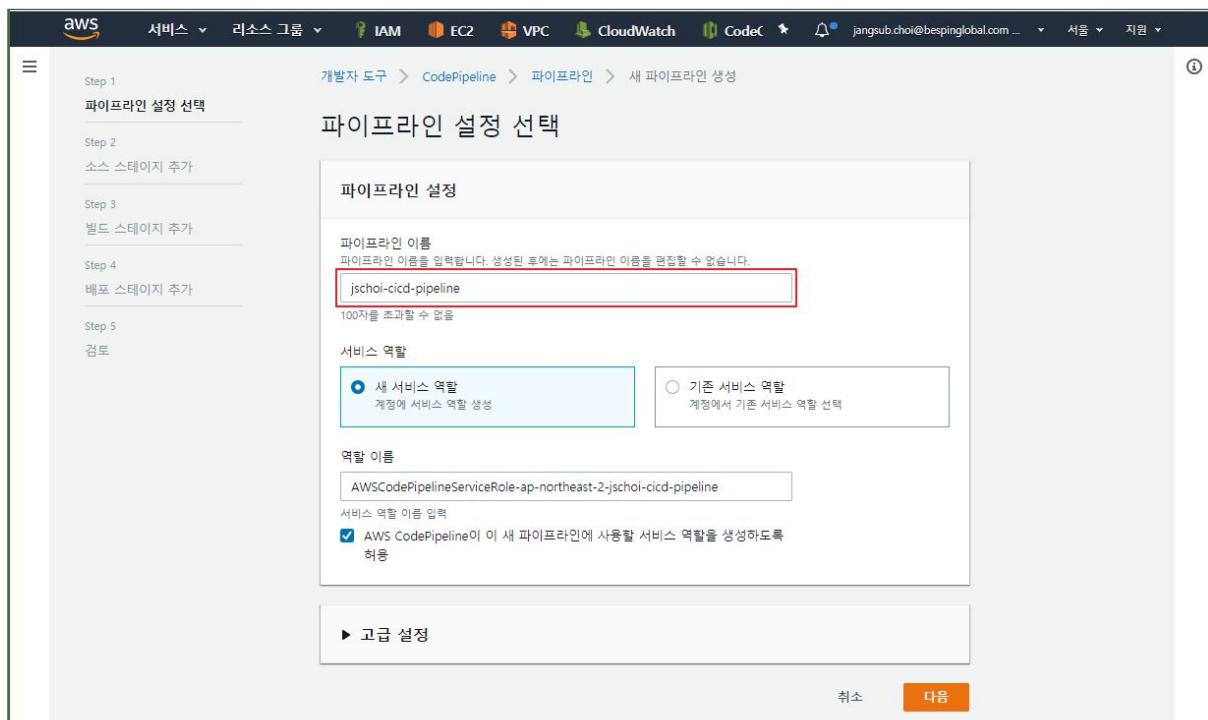
## 5. AWS CodePipeline

AWS CodePipeline은 빠르고 안정적으로 애플리케이션 및 인프라 업데이트를 위해 릴리스 파이프라인을 자동화하는데 도움이 되는 완전 관리형 지속 전달 서비스입니다.

- AWS Web Console 내 CodePipeline 페이지로 이동 후 파이프 라인을 생성 합니다.



- 파이프라인에 대한 이름을 지정 후 다음으로 진행 합니다.



파이프라인에 대한 이름을 지정하면 새로운 서비스 역할의 이름이 자동으로 지정 됩니다.

- 다음 사항을 참고하여 소스 스테이지를 추가 합니다.

소스 스테이지 추가

**소스**

소스 공급자: AWS CodeCommit

리포지토리 이름: jschoi-cicd-study

브랜치 이름: master

변경 감지 옵션: Amazon CloudWatch Events(권장)

- **소스 공급자** : AWS CodeCommit 선택
- **리포지토리 이름** : 사용자의 소스가 보관되어 있는 Repositories
- **브랜치 이름** : 사용자 브랜치를 지정
- **변경 감지 옵션** : Amazon CloudWatch Events 를 선택

- 다음 사항을 참고하여 빌드 스테이지를 추가 합니다.

빌드 스테이지 추가

**빌드 - 선택 사항**

빌드 공급자: AWS CodeBuild

리전: 아시아 태평양(서울)

프로젝트 이름: jschoi-cicd-study-build

- **빌드 공급자** : AWS CodeCommit 선택
- **리전** : 아시아 태평양 (서울) 선택
- **프로젝트 이름** : AWS CodeBuild 프로젝트 이름을 지정

- 다음 사항을 참고하여 배포 스테이지를 추가 합니다.

部署 - 선택 사항

**部署 공급자**  
인스턴스에 배포하는 방법을 선택합니다. 공급자를 선택한 후 해당 공급자에 대한 구성 세부 정보를 제공해십시오.

AWS CodeDeploy

**리전**

아시아 태평양(서울)

**애플리케이션 이름**  
AWS CodeDeploy 콘솔에서 이미 생성한 애플리케이션을 선택합니다. 또는 AWS CodeDeploy 콘솔에서 애플리케이션을 생성한 후 이 작업으로 돌아옵니다.

jschoi-cicd-application

**배포 그룹**  
AWS CodeDeploy 콘솔에서 이미 생성한 배포 그룹을 선택합니다. 또는 AWS CodeDeploy 콘솔에서 배포 그룹을 생성한 후 이 작업으로 돌아옵니다.

jschoi-cicd-group

취소 이전 배포 스테이지 건너뛰기 다음

- **배포 공급자** : AWS CodeDeploy 선택
- **리전** : 아시아 태평양 (서울) 선택
- **애플리케이션 이름** : AWS CodeDeploy 에서 생성한 애플리케이션을 선택
- **배포 그룹** : AWS CodeDeploy 의 애플리케이션에서 생성한 배포 그룹을 선택

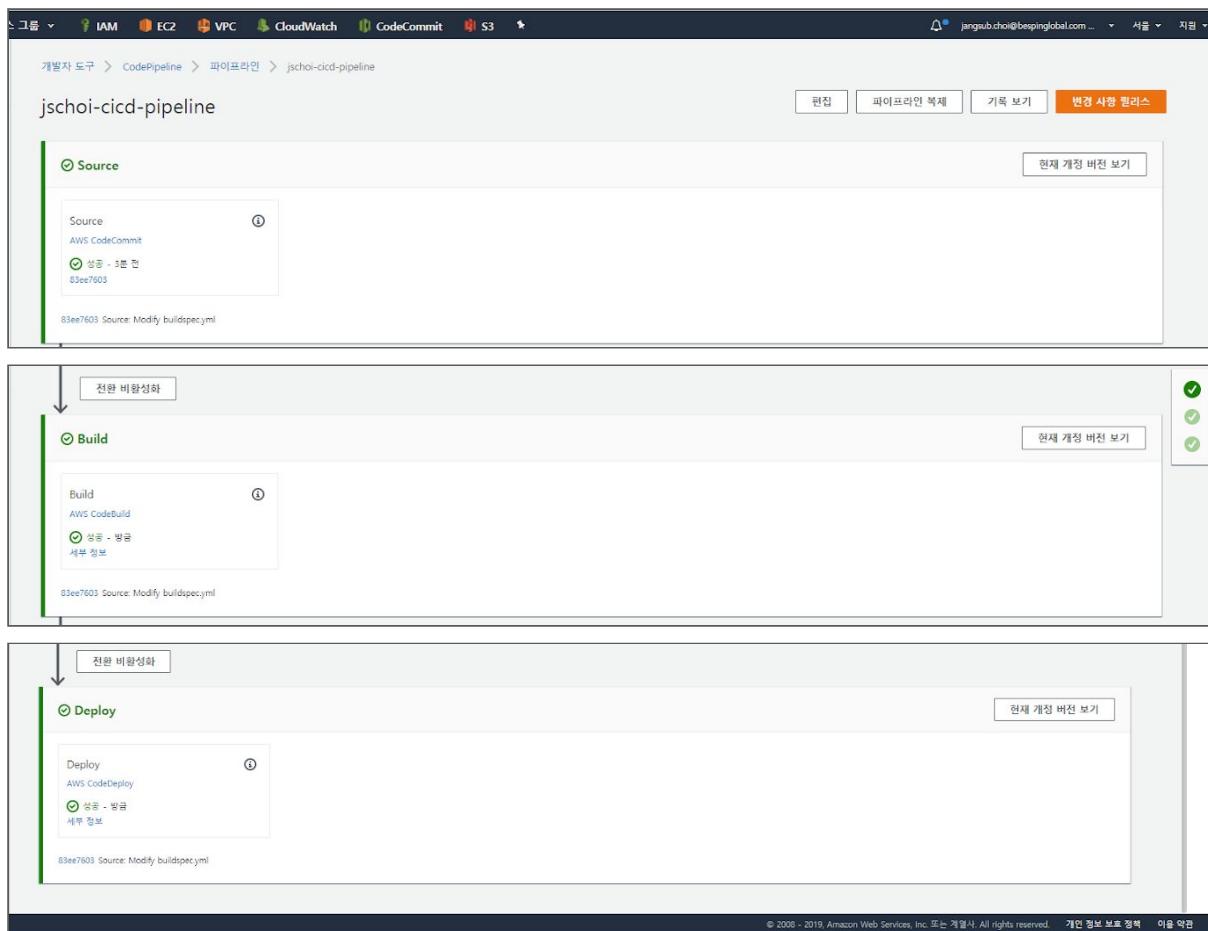
- 파이프 라인을 생성하기 전에 내용을 충분히 검토 한 후 파이프 라인을 생성합니다.

The screenshot shows the AWS CodePipeline console interface for creating a new pipeline. The pipeline is named 'jschoi-cicd-pipeline' and is located in the 'codepipeline-ap-northeast-2-920672512890' account. The pipeline consists of four stages: Step 1 (검토), Step 2 (소스 스테이지 추가), Step 3 (빌드 스테이지 추가), and Step 4 (배포 스테이지 추가). Each stage has its own configuration card:

- Step 1: 검토** (Review)
- Step 2: 소스 스테이지 추가** (Add Source Stage)
  - Source 작업 공급자** (Source Action Provider): AWS CodeCommit
  - RepositoryName**: jschoi-cicd-study
  - BranchName**: master
  - PollForSourceChanges**: false
- Step 3: 빌드 스테이지 추가** (Add Build Stage)
  - 빌드 작업 공급자** (Build Action Provider): AWS CodeBuild
  - ProjectName**: jschoi-cicd-study-build
- Step 4: 배포 스테이지 추가** (Add Deployment Stage)
  - 배포 작업 공급자** (Deploy Action Provider): AWS CodeDeploy
  - ApplicationName**: jschoi-cicd-application
  - DeploymentGroupName**: jschoi-cicd-group

At the bottom of the pipeline configuration screen, there are three buttons: '취소' (Cancel), '이전' (Previous), and a prominent orange button labeled '파이프라인 생성' (Create Pipeline).

- 배포 스테이지 추가까지 완료되면 자동으로 Pipeline 이 생성이 완료와 동시에 파이프 라인 페이지 내에서 Source, Build, Deploy 확인이 가능합니다.



최종적으로 AWS CodeCommit 에 소스가 변경되면 이를 감지하고 CodePipeline 이 자동으로 Commit, Build, Deploy 과정을 진행하여 처리하게 됩니다.

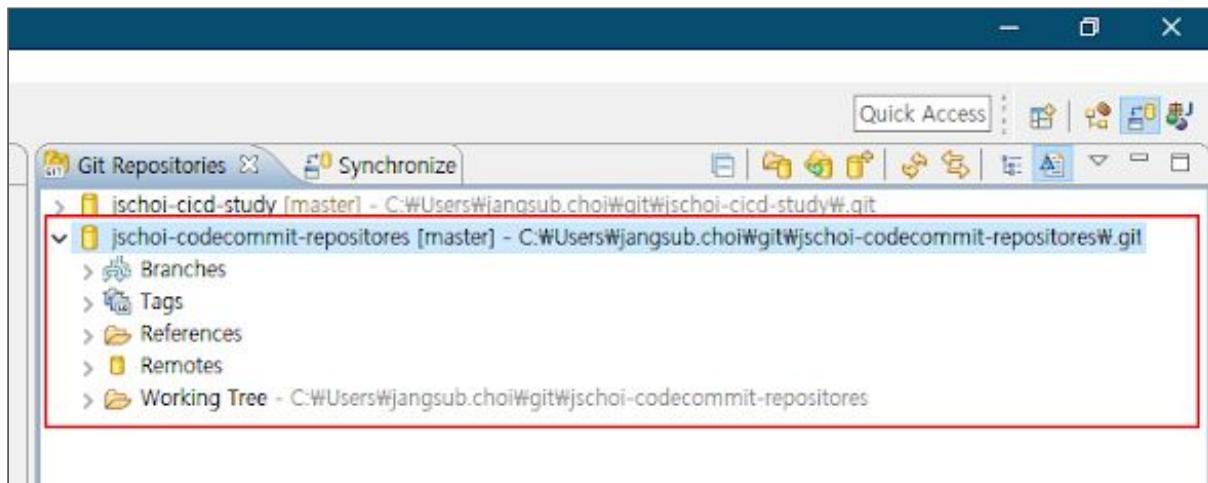
# 별첨 문서

## IDE Project Configuration

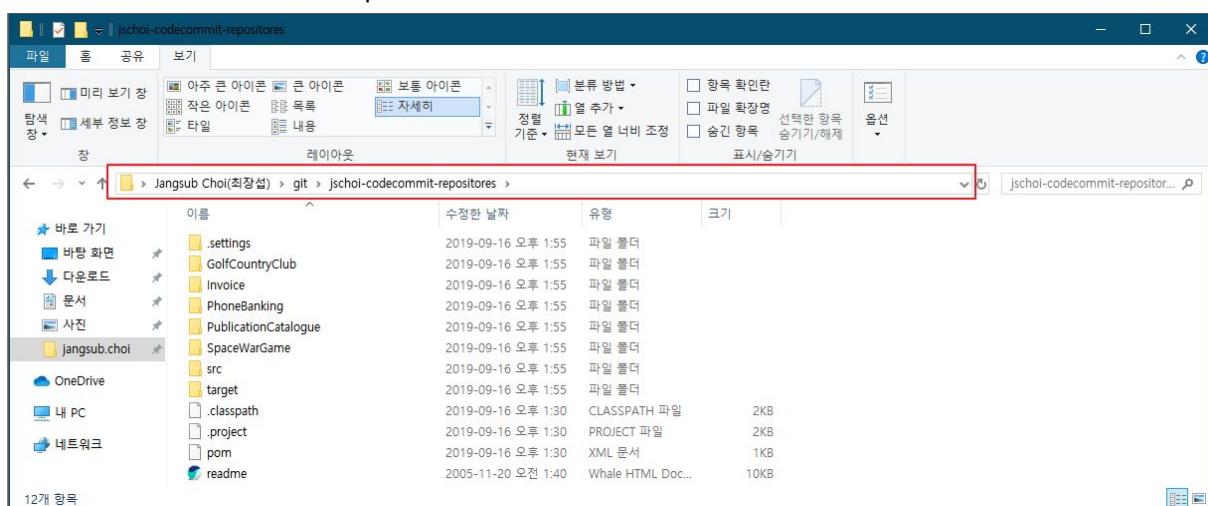
### 기존에 소스가 있는 경우

만약 사용자가 기존에 개발중인 소스가 있다면 다음을 참고 합니다.

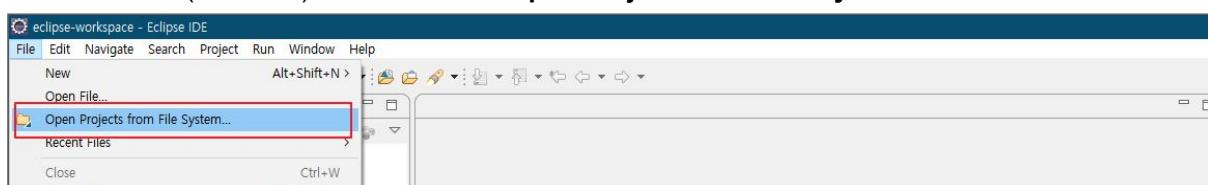
- 먼저 AWS CodeCommit Repositories 를 IDE (이클립스) 에 연결 합니다.



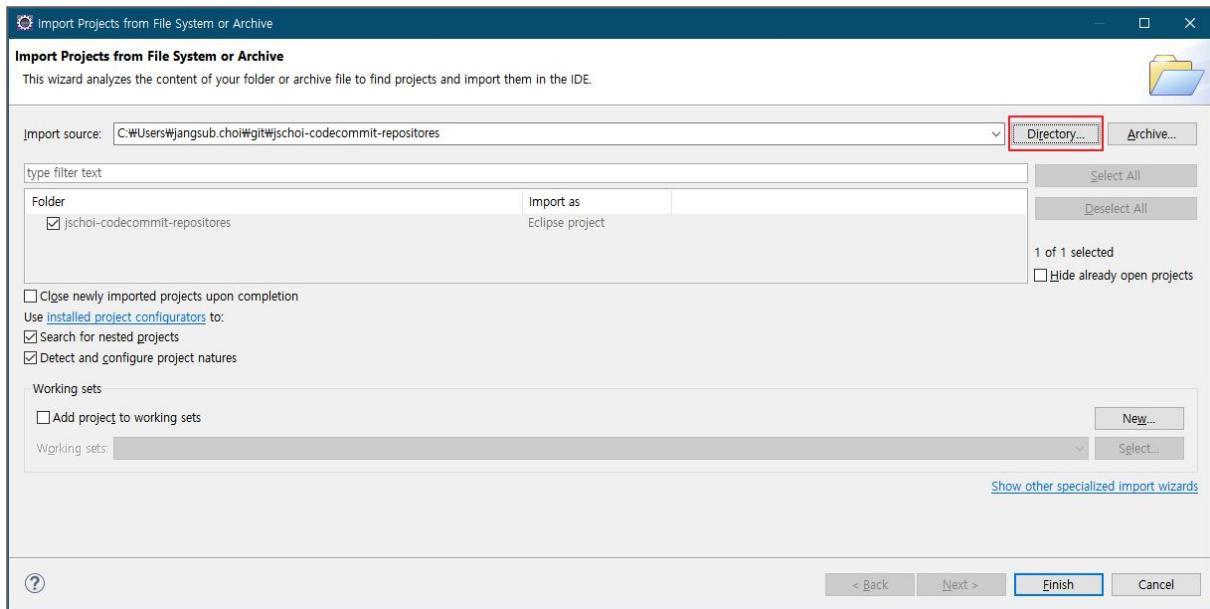
- AWS CodeCommit Repositories 가 연결된 디렉토리에 기존의 개발 소스를 복사 합니다.



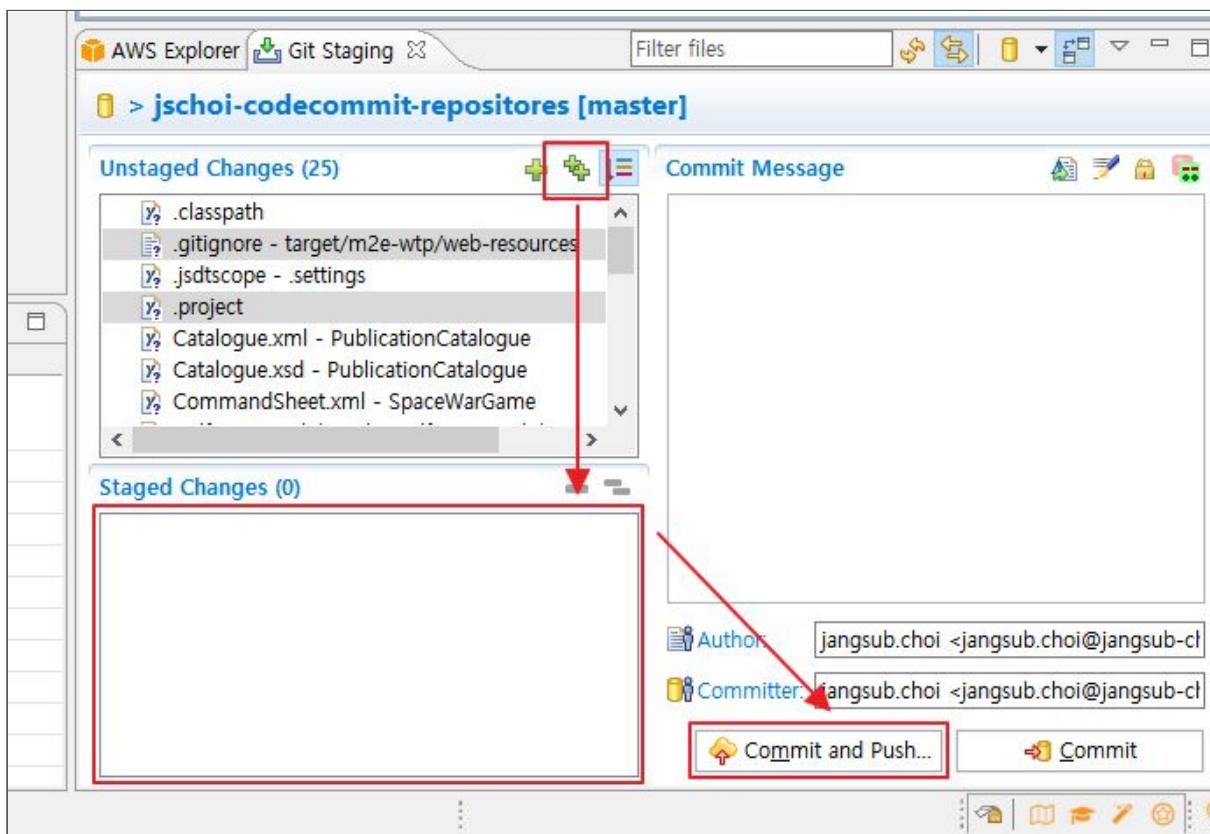
- IDE 툴 (이클립스) 메뉴에서 **File > Open Projects from File System...** 을 클릭 합니다.



- Directory 를 클릭하여 위치를 확인하고 완료 합니다.



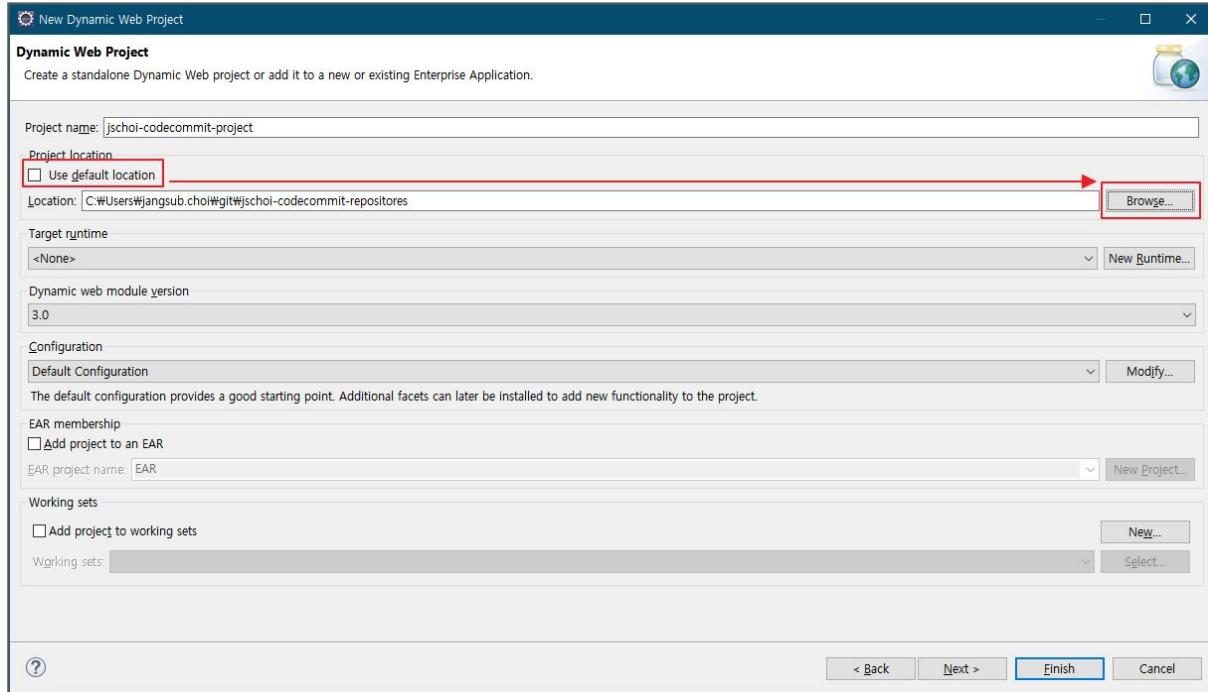
- 변경 사항들이 AWS CodeCommit Repositories 에 Commit 할 수 있게 됩니다.



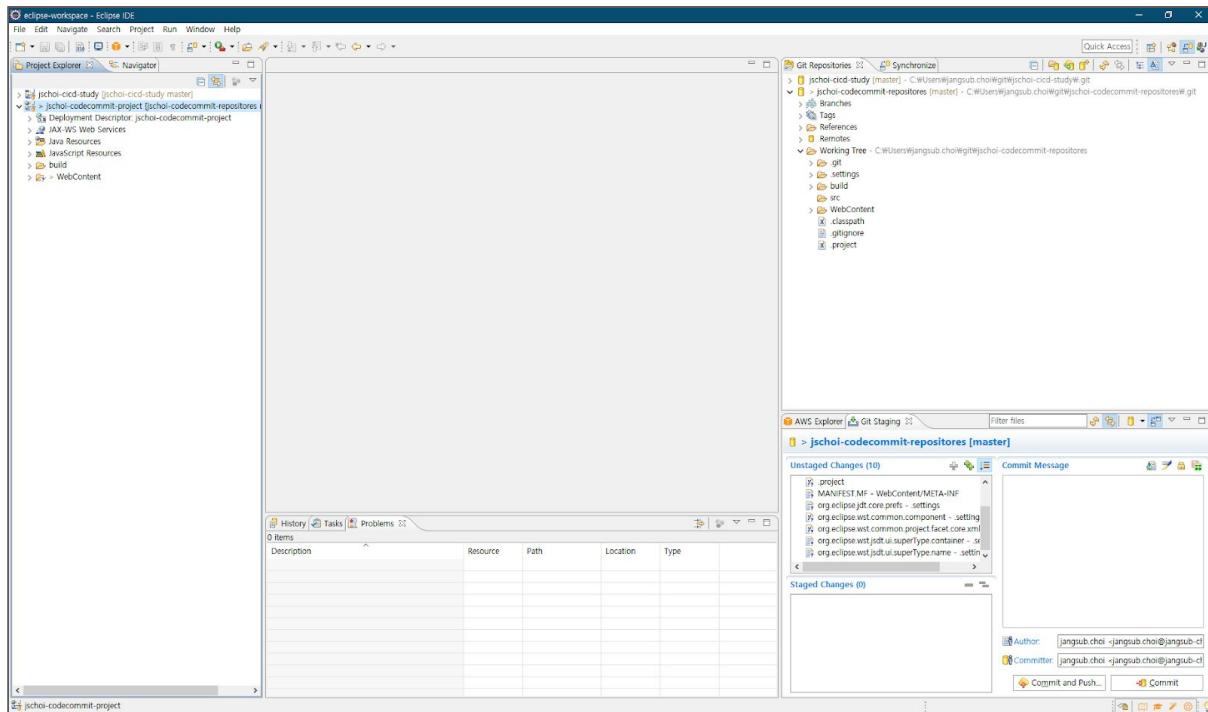
## 기존에 소스가 없는 경우

만약 사용자가 기존에 개발중인 소스가 없다면 다음을 참고 합니다.

- 사용자가 IDE 툴 (이클립스)에서 Project를 생성할 때 Local 경로를 AWS CodeCommit Repositories 경로로 변경하여 생성 합니다.



- 사용자의 Project 가 생성되고 Repositories 에서 생성된 파일들을 자동으로 인식하여 AWS CodeCommit Repositories 에 Commit 할 수 있게 됩니다.



# SAMPLE CODE

## Buildspec.yml

```

version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk8
    commands:
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
    finally:
      - echo This always runs even if the update or install command fails
  pre_build:
    commands:
      - echo Entered the pre_build phase...
    finally:
      - echo This always runs even if the login command fails
  build:
    commands:
      - echo Entered the build phase...
      - echo Build started on `date`
      - mvn install
    finally:
      - echo This always runs even if the install command fails
  post_build:
    commands:
      - echo Entered the post_build phase...
      - echo Build completed on `date`

## 빌드 후 결과물을 생성 ##
artifacts:
  files:
    - appspec.yml
    - scripts/**
    - target/*

```

## Appspec.yml

```

version: 0.0
os: linux
files:
  - source: target/jschoi-codecommit-project-0.0.1-SNAPSHOT.war
    destination: /root/

permissions:
  - object: /
    pattern: "*"
    owner: root
    group: root

hooks:
  #AppSpec file은 인스턴스에 배포하기 전에는 인스턴스에 존재하지 않습니다. 때문에 인스턴스에
  처음으로 배포하면 ApplicationStop 휴크는 실행되지 않습니다. 인스턴스에 두 번째 배포하면
  ApplicationStop 휴크를 사용
  ApplicationStop:
    - location: scripts/applicationStop.sh
      timeout: 120
      runas: root
    #이 배포 수명 주기 이벤트는 파일 암호 해독 및 현재 버전의 백업 만들기 등 사전 설치/작업에
    사용
  BeforeInstall:
    - location: scripts/beforeInstall.sh
      timeout: 180
      runas: root
    #이 배포 수명 주기 이벤트는 애플리케이션 구성 또는 파일 권한 변경 등의 작업에 사용할 수
    있습니다
  #AfterInstall:
  #  - location: scripts/afterInstall.sh
  #    runas: appuser1
  #이 배포 수명 주기 이벤트는 일반적으로 ApplicationStop 중 종지된 서비스를 다시 시작하는 데
  사용
  ApplicationStart:
    - location: scripts/applicationStart.sh
      timeout: 180
      runas: root
    #마지막 배포 수명 주기 이벤트입니다. 배포가 성공적으로 완료되었는지 확인하는 데 사용 [ ALB
    health check ]
  #ValidateService:
  #  - location: scripts/basic_health_check.sh
  #    timeout: 180

```

## Application Scripts

```
## Application Stop Scripts
echo "Application Stop Script.." > /tmp/jschoi_cicd_study/application_stop.log

## Application Start Scripts
echo "Application Start Script.." > /tmp/jschoi_cicd_study/application_Start.log

## Application Afterinstall Scripts
echo "Afterinstall Scripts.." > /tmp/jschoi_cicd_study/application_afterinstall.log

## Application Beforinstall Scripts
rm -rf /tmp/jschoi_cicd_study

sleep 1

mkdir /tmp/jschoi_cicd_study/
echo "Before install Script.." > /tmp/jschoi_cicd_study/application_beforinstall.log
```