Hands On Exercises
Working with AWS CI/CD Tools

Lab 1 Working with AWS CodeCommit

Eclipse 를 사용하여 기본 Spring Boot Java project 를 생성한 후 AWS CodeCommit 과 연동 합니다. 이후 local machine 에서 개발을 하고 AWS CodeCommit repository 에 push 해서 commit 를 합니다.

1. Setup Spring Project from Spring Tools Suite 4
   - 1.1 Create project folder. The name of the folder must be the same as your project name. In our case, this is st##-e2edemo
   - 1.2 File > New > Spring Starter Project
   - 1.3 Name = <st##>-e2edemo
   - 1.4 Uncheck -> Use default location
   - 1.5 Location > Browse and point to your Java project folder
   - 1.6 Type = Maven Project
   - 1.7 Packaging = WAR
   - 1.8 Java Version = 17
   - 1.9 Language = Java
   - 1.10 Group: com.<your_company>
   - 1.11 Artifact: <st##>-e2edemo
   - 1.12 Version: 0.0.1-SNAPSHOT
   - 1.13 Description = Demo Project for end2end
   - 1.14 Package = com.<your-company>.e2edemo
   - 1.15 [Next]
   - 1.16 Spring Boot Version = 3.2.2
   - 1.17 Dependencies to add
     - 1.17.1 Spring Web
   - 1.18 [Finish]
2. Create simple “Hello World” application
   - 2.1 src\main\com.wiken.e2edemo 에서 “HelloWorldController.java” 새로 파일 생성
   - 2.2 만들어진 template 에서 아래 위치에 @RestController annotation 을 추가 하고 CTRL-SHIFT-O 하면 자동으로 필요한 “import” 문 추가 합니다

```
package com.wiken.e2edemo;

@RestController
public class HelloWorldController {
}
```

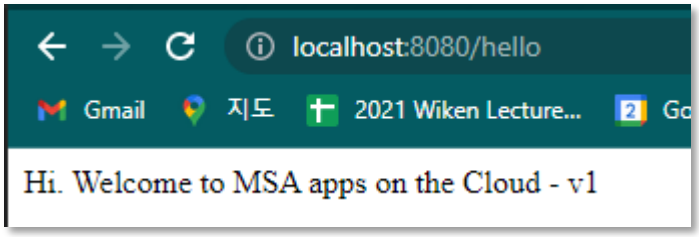
2.3 나머지 코드를 아래와 같이 입력. @GetMapping annotation 을 입력 하면 VS Code 가 자동으로 필요한 "Import" 문을 추가 합니다

```
package com.wiken.e2edemo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hi. Welcome to CNA apps on the Cloud - v1";
    }

}
```

3. Local 에서 실행

3.1 Spring Boot Dashboard 선택 후 상단 "APPS" 에서 e2edemo 실행

3.2 Browser 를 열어서 "localhost:8080/hello" 로 이동 후 정상 적 실행 확인

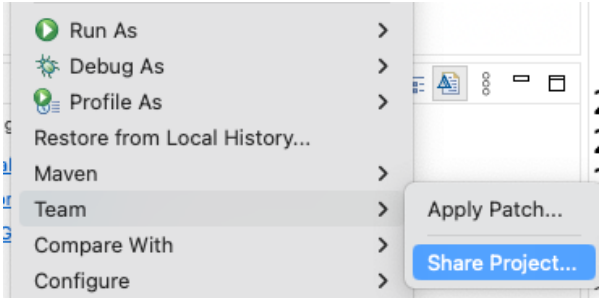


4. Create and link to Github

4.1 Github repository 등록

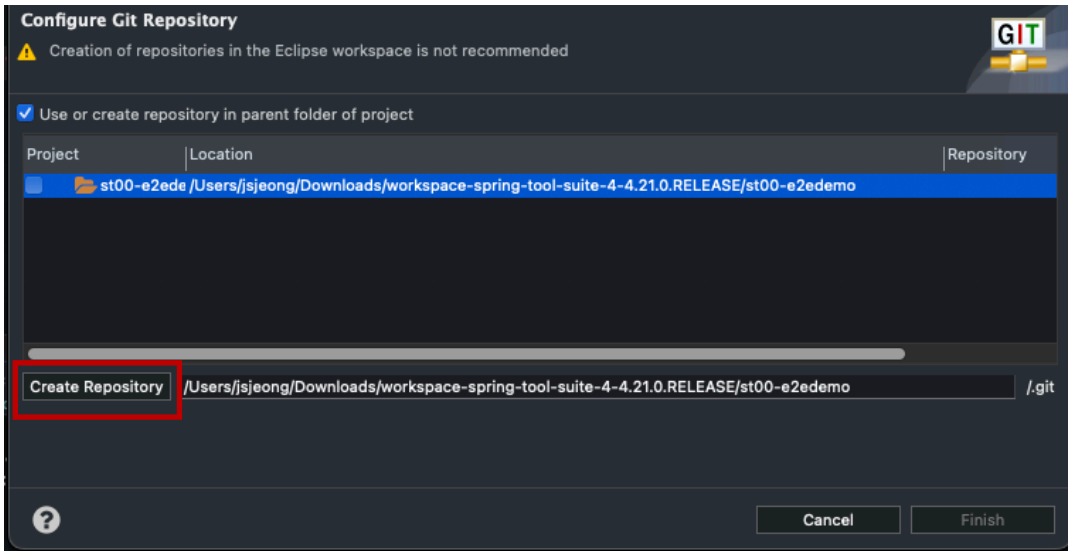
4.1.1 St##-e2edemo project 선택 후, right click

4.1.2 메뉴에서 Team > Share Project 선택



4.1.3 Pop-up 창에서 > Use or create repository in parent folder of project

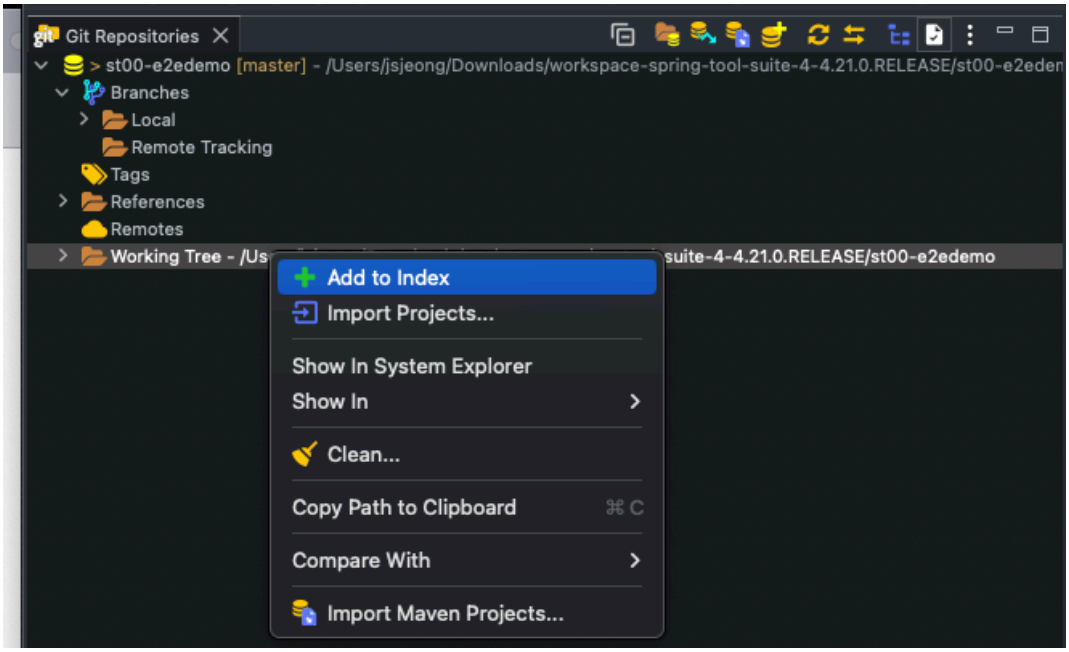
4.1.4 Project 선택 후 [Create Repository]



4.1.5 [Finish]

4.2 local source commit 하기

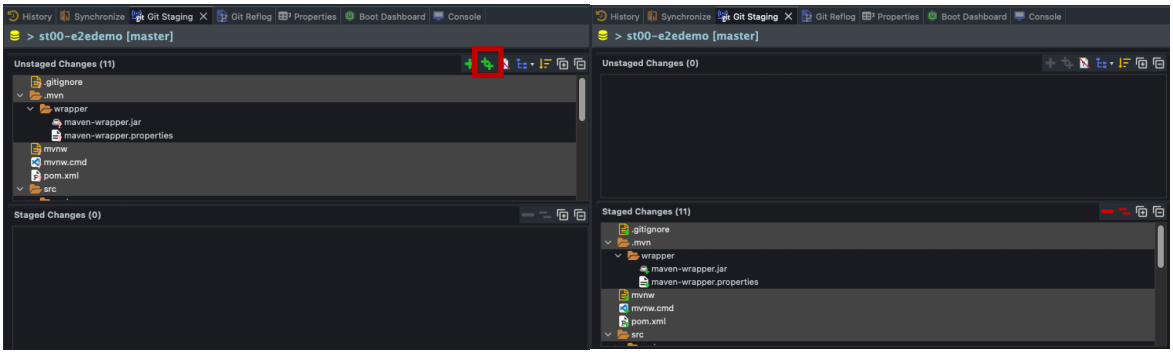
4.2.1 Index working tree (click the right button)에서 Add to index 하기



4.2.2 Git view 선택하고 하단 Git Staging 로 이동

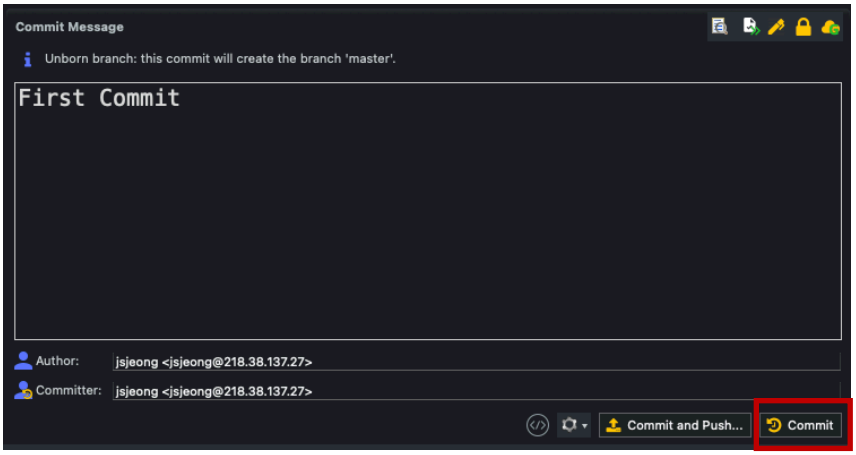
(a) Branches > Local 후 하단에 Git Staging

4.2.3 Select all unstages changes and move to staged change

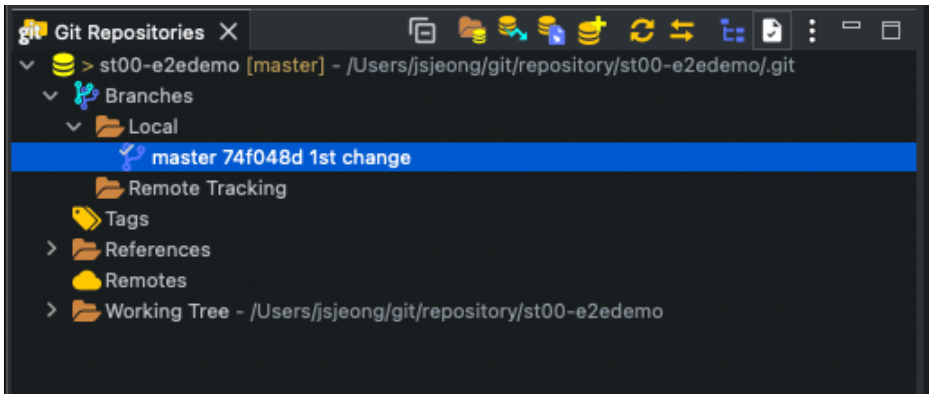


4.2.4 From Commit Message, enter “First Commit”

4.2.5 [Commit] (Be careful. Do not select Commit and Push)

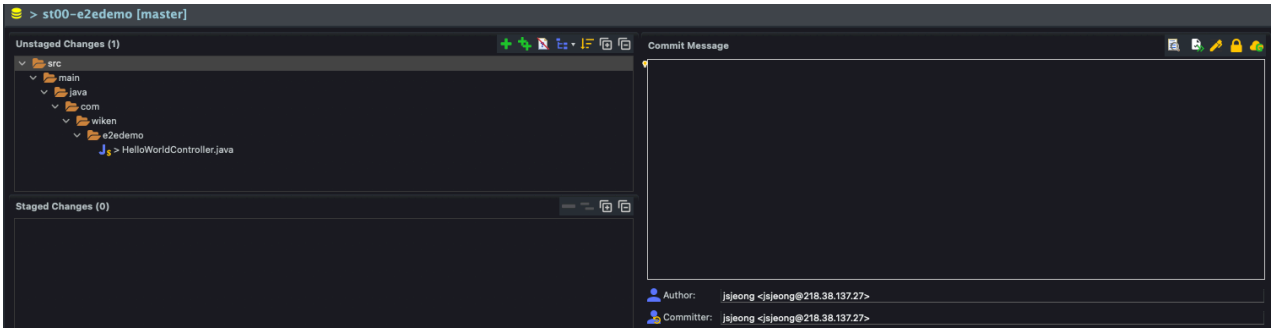


4.2.6 In Git repositories, open Branches > Local again and you will see the First Commit



4.3 Makes changes and commit again

4.3.1 Change the return string to “v2” and saveGo back to Git view > Git Staging. You will see the change reflected



4.3.2 Once again, drag all the changes, enter “Second Commit” for the commit message and [Commit]

4.3.3 You will see the branch updated to Second Commit

5. AWS CodeCommit 과 연결

5.1 AWS CodeCommit repository 생성

5.1.1 AWS Console 에서 CodeCommit 으로 이동

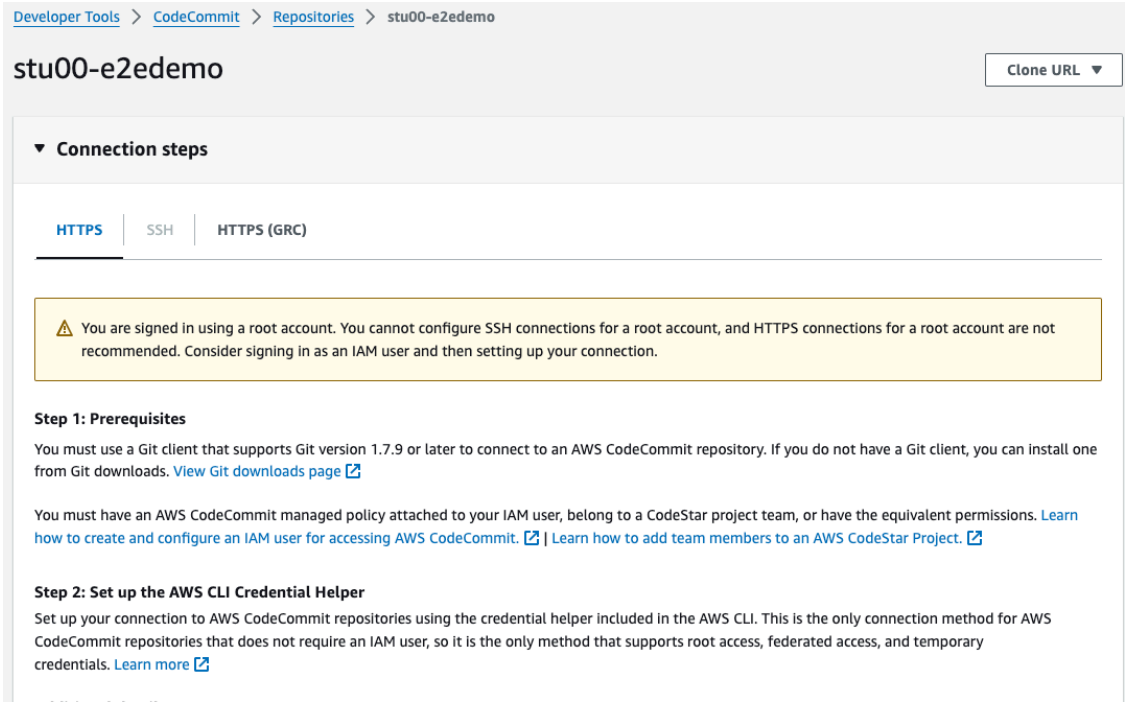
5.1.2 [Create repository] 선택

5.1.3 Repository name => <st##>-e2edemo

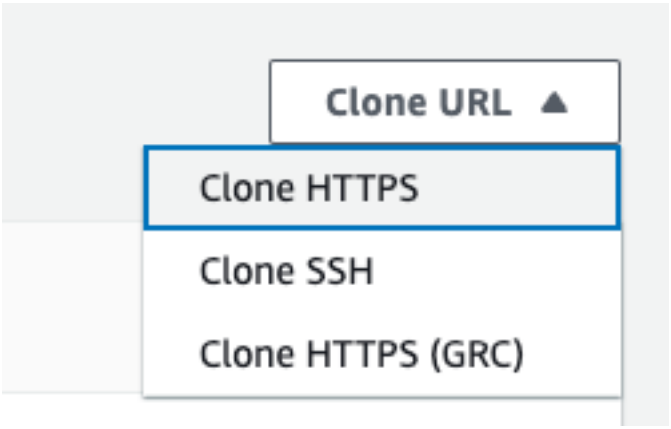
5.1.4 Description => 원하시는 내용 입력

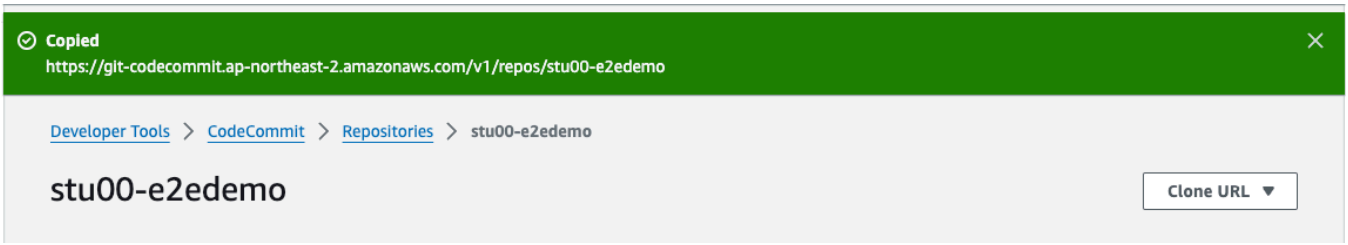
5.1.5 Tags [Add] 선택 후 key = Name, Value = “st##-e2edemo”

5.1.6 [Create] 클릭 하고 나면 설명 page 생성. HTTPS 연결 설명서를 읽고 확인. Eclipse 는 이미 전제 조건 1 단계를 충족합니다.



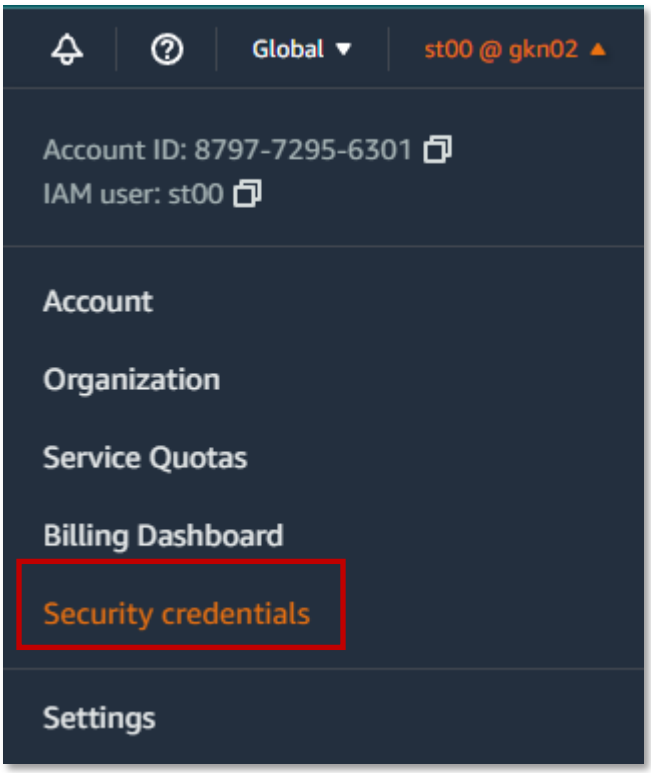
5.1.7 Copy the address to clone the repository



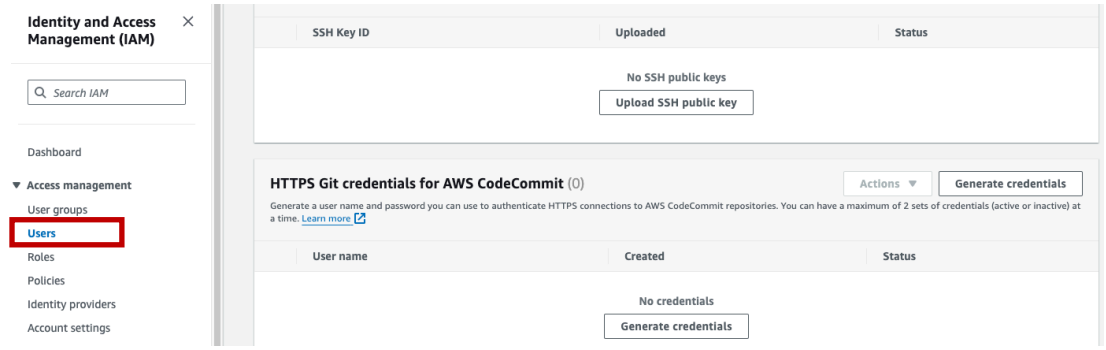


5.2 Security Credential 가지고 오기

5.2.1 AWS Console 에서 우측상단 메뉴에서 본인 유저 선택 후 Security Credentials 선택



5.2.2 My Security Credentials 에서 왼쪽 “Users”를 선택하여 AWS CodeCommit credentials tab 선택 후 [HTTPs Git Credentials for AWS CodeCommit] 에서 [Generate credentials] 선택



5.2.3 생성된 credentials 다운로드 후 사용

5.3 Eclipse 에서 AWS CodeCommit 연결 (전제 조건 3 단계)

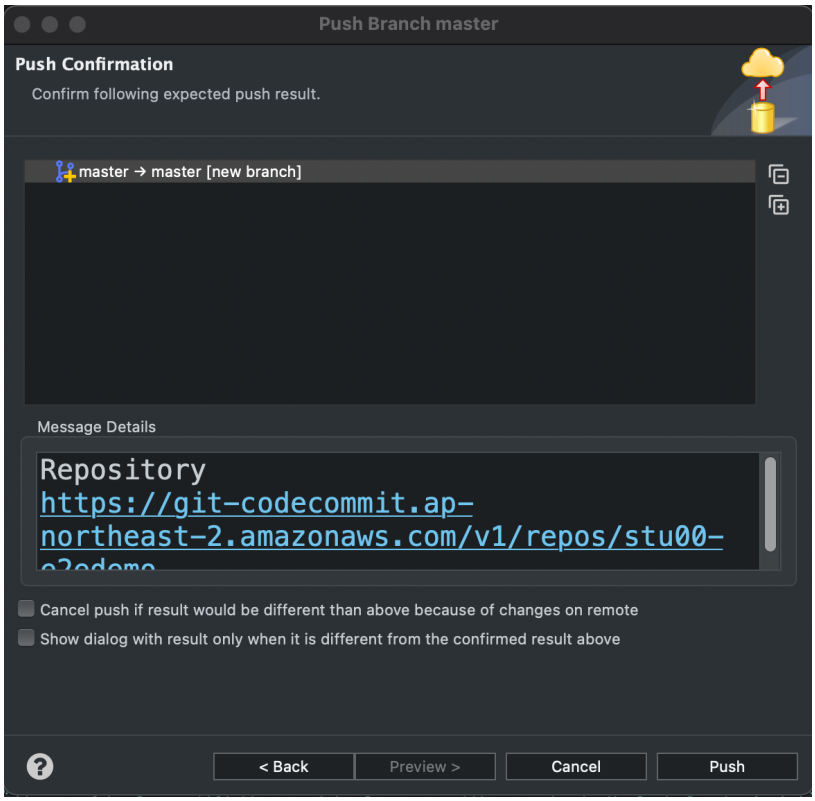
5.3.1 From Java view, right click on project and go to Team

5.3.2 Select Push branch master. A pop-up window open. Enter the address you copied from above into the URI location

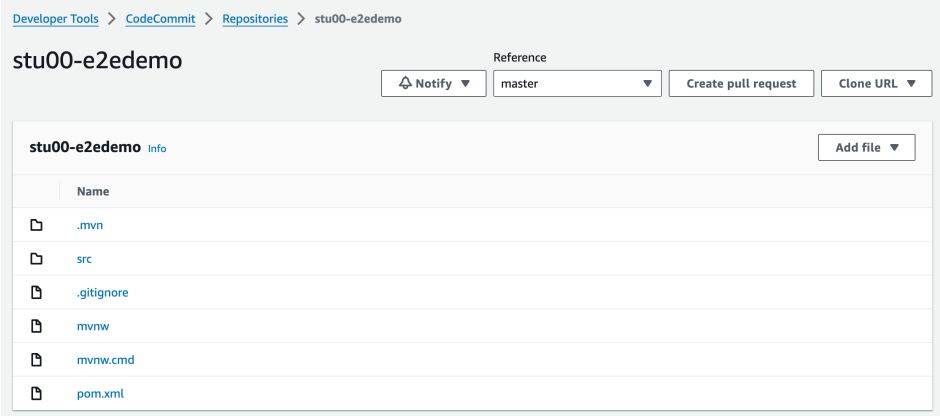
5.3.3 In Authentication, enter the information for credentials you downloaded above. Click on “Store in Secure Store”

5.3.4 If the login screen pops up again, just reenter the information.

5.3.5 [Push] - You will get a confirmation pop-up that is has been pushed. [Close]



5.3.6 AWS CodeCommit 에서 repository 선택 후 확인

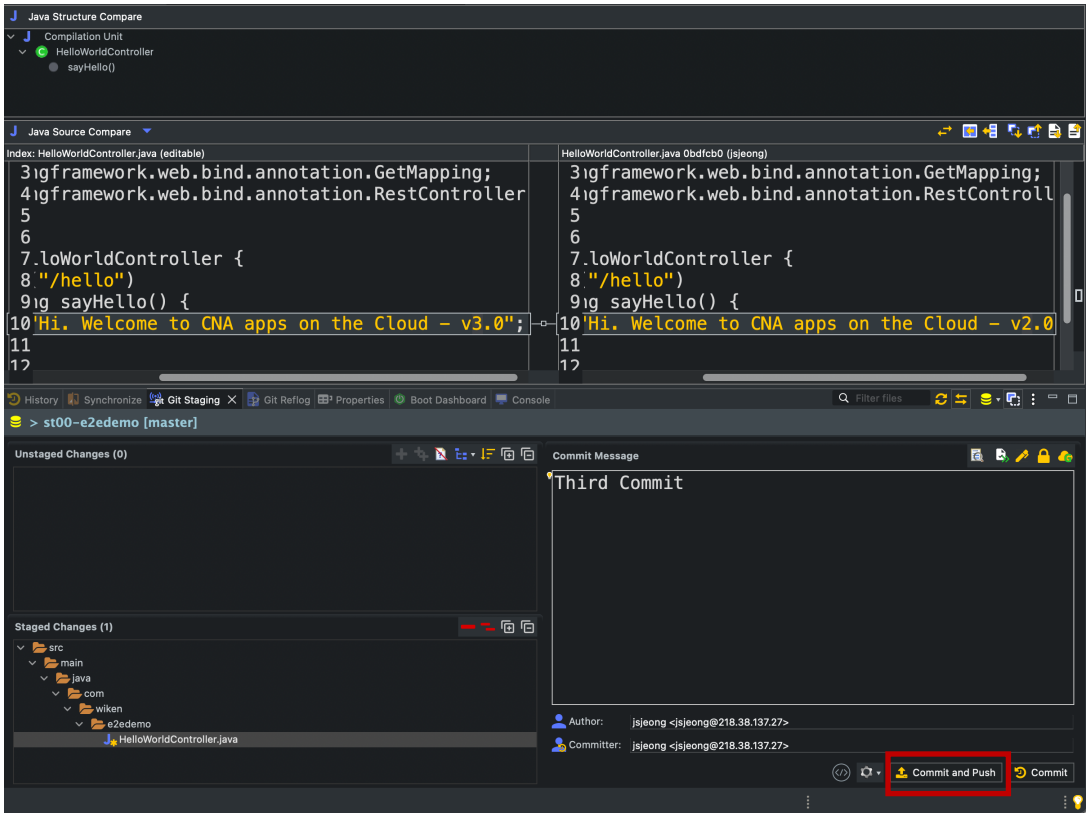


5.3.7 HelloWorldController.java 에서 출력 문자를 다음과 같이 변경 후 저장

```
public String sayHello() {
    return "Hi. Welcome to MSA apps on the Cloud - v3";
```

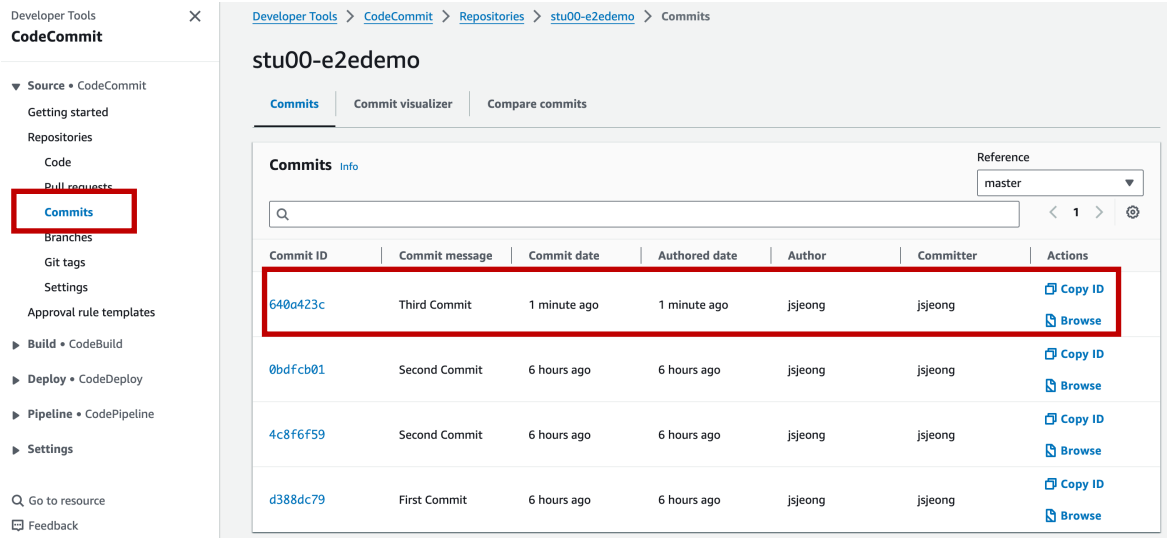
5.3.8 Commit to local as Third Commit and push to AWS CodeCommit

5.3.9 This time, after setting the staged changes, enter “Third Commit”, you will select [Commit and Push] since we now have AWS CodeCommit to push to. A pop up window will confirm success

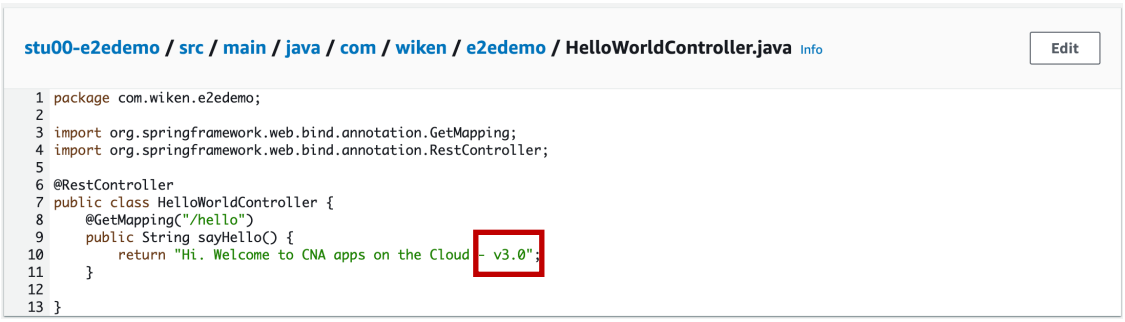


5.4 Go back to AWS CodeCommit and verify the Third Commit

5.4.1 On the left tab menu, select Commits. You should see three commits.



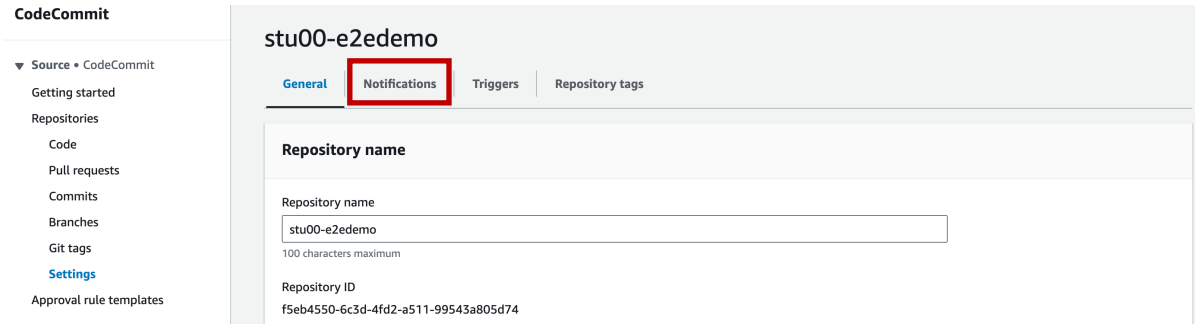
5.4.2 Repository 에서 내용물 update 확인



6. Create notifications

6.1 Navigate to CodeCommit > Repositories > Settings

6.2 Select the Notification tab



6.3 [Create notification rule]

6.4 Notification name = <st##>-e2edemo-notification

6.5 Events that trigger notification

6.5.1 On commits

6.5.2 Status changed

6.5.3 Source updated

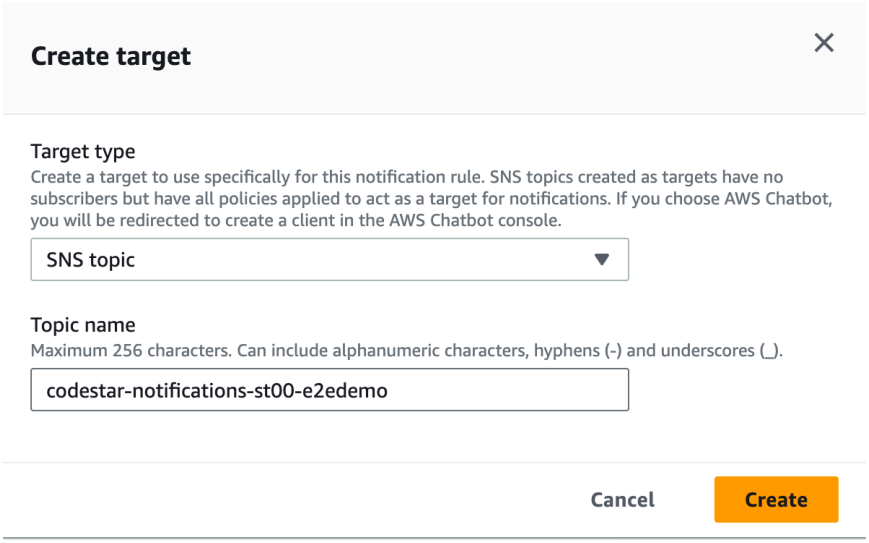
6.6 Targets

6.6.1 Create target

6.6.2 SNS topic

6.6.3 Topic name = codestar-notifications-st##-e2edemo

6.6.4 [Create]

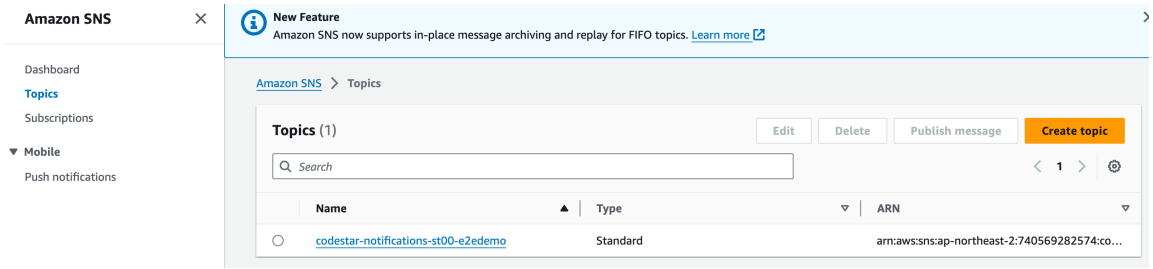


6.6.5 [Submit]

7. Subscribe to topic

7.1 Navigate to SNS (Simple Notification Service) console

7.2 From topic, look for your topic and select it

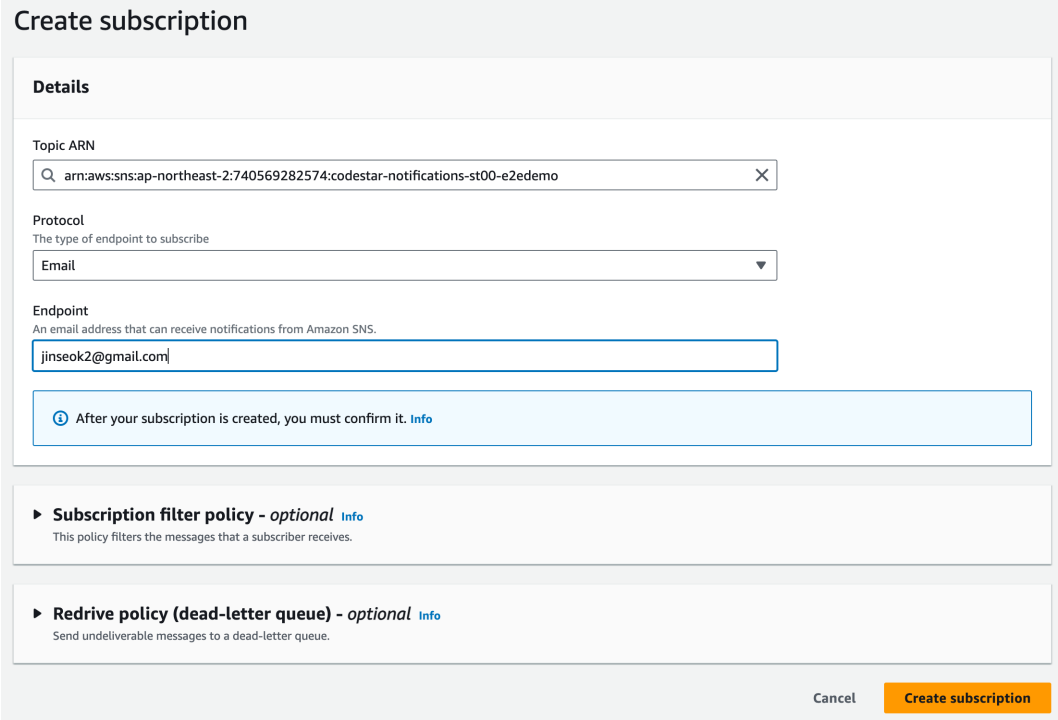


7.3 Create subscription

7.3.1 Protocol = Email

7.3.2 Endpoint = “Enter your email”

7.3.3 [Create subscription]



7.3.4 You will receive an email at the email address you entered. Confirm the subscription

Lab 2 Working with AWS CodeBuild

1. Create New S3 Bucket

If you have created your main-bucket from before, you may skip this step

1.1 General configuration

1.1.1 Bucket name => ktds-<st##>

1.1.2 AWS Region => ap-northeast-2

1.2 Bucket Versioning

1.2.1 Enable Bucket Versioning

1.3 Create bucket

1.4 Create folder

1.4.1 Name => CodeBuild

2. Build New CodeBuild project

2.1 Project Configuration

2.1.1 Project Name => <st##>-e2edemo

2.1.2 Additional configuration

2.1.3 Description => 원하시는 설명

2.1.4 Build badge => enable
(provide an embeddable, dynamically generated image (badge) that displays the status of the latest build for a project. This image is accessible through a publicly available URL generated for your CodeBuild project)

2.2 Source

2.2.1 Source Provider => AWS CodeCommit

2.2.2 Repository => <st##>-e2edemo

2.2.3 Reference type => Branch

2.2.4 Branch => master

2.3 Environment

2.3.1 Keep Managed image

2.3.2 Operating system => Amazon Linux

2.3.3 Runtime => Standard

2.3.4 Image => aws/codebuild/amazonlinux2-x86_64-standard:4.0

2.3.5 Image version: Always use the latest image for this runtime version

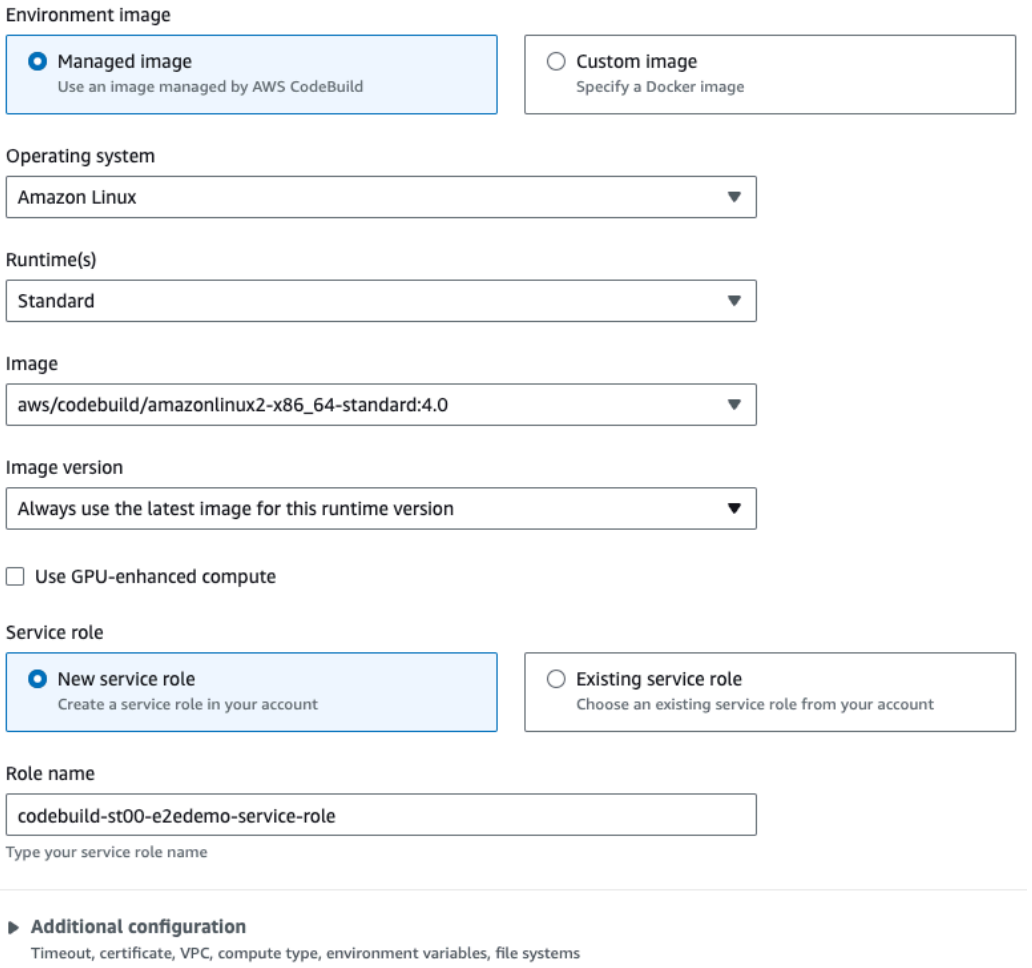
2.3.6 Service role => New service role

2.3.7 Role name => default 값을 사용 / 고유명 필수

2.3.8 Additional configuration 에서:

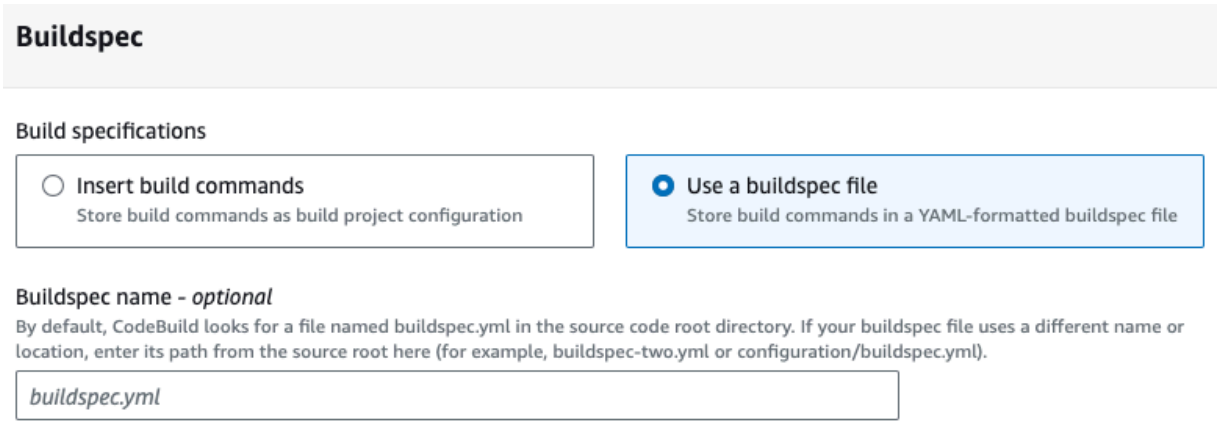
Timeout => 5 분

Queued timeout => 10 분



2.4 Buildspec

2.4.1 Use a buildspec file 선택



2.5 Artifacts

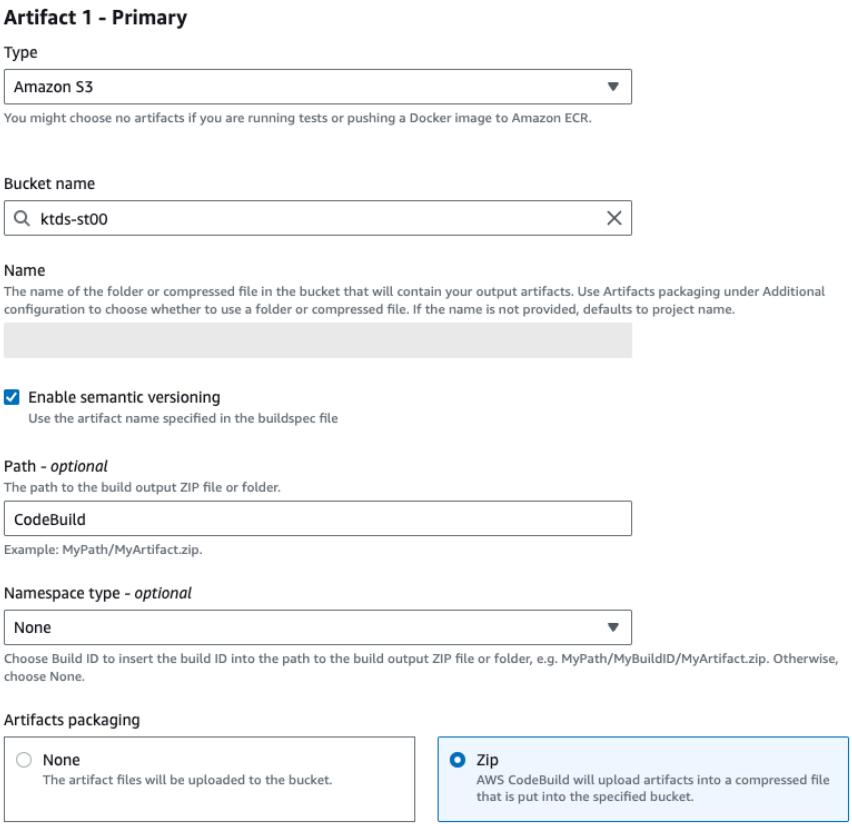
2.5.1 Type => Amazon S3

2.5.2 Bucket name => ktds-<st##>-your bucket name (이전에 생성한 버킷)

2.5.3 Enable semantic versioning => 선택

2.5.4 Path => S3 bucket 생성시 folder 명 => CodeBuild

2.5.5 Artifacts packaging => Zip

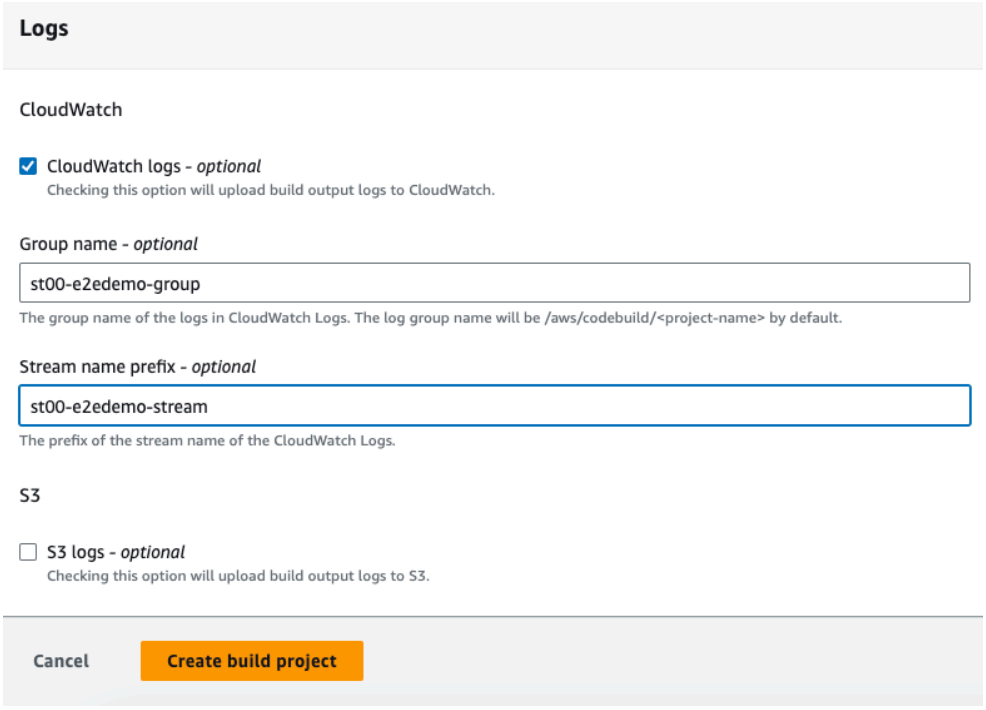


2.6 Logs

2.6.1 Enable CloudWatch logs

2.6.2 Group name => <st##>-e2edemo-group

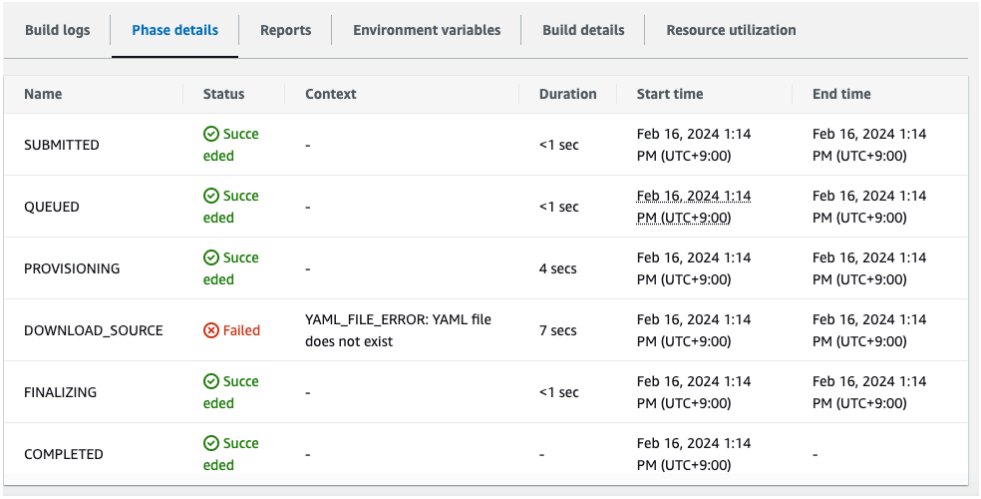
2.6.3 Stream name = <st##>-e2edemo-stream



2.7 Create build project

3. Start CodeBuild Project

지금 상태에서 Start build 를 실행 하면 fail 됩니다. CodeBuild build project 를 생성 할 때 Buildspec 를 파일로 전달한다고 함. (Lab 22.4.1). 아래 “Download Source”에서 Yaml file Fail 을 확인할 수 있습니다.



3.1 Test building WAR file from Spring Boot

3.1.1 From Java view, right click on project, select run as Maven clean.

3.1.2 Repeat with Maven Install this time.

3.1.3 Install 선택 후 실행

Install 이 잘 되는 것 을 확인 했으나 생성된 WAR 파일 이름 을 수정함.

3.1.4 Explorer 에서 pom.xml 선택

3.1.5 아래 build section 에서 finalName 추가 후 저장. Pop-up 창이 떠서 synchronize 를 원하는가 질문하면 [Yes] 클릭

```
38    <build>
39        <plugins>
40            <plugin>
41                <groupId>org.springframework.boot</groupId>
42                <artifactId>spring-boot-maven-plugin</artifactId>
43            </plugin>
44        </plugins>
45        <finalName>${artifactId}</finalName>
46    </build>
47
48 </project>
```

3.1.6 다시 MAVEN => Clean 실행 후 MAVEN => Install

설정한 이름으로 WAR 생성 확인 (st##-e2edemo.war)

3.2 Buildspec.yml 파일 생성

3.2.1 Explorer 에서 root 위치에 파일 추가 (pom.xml 이 있는 작업 디렉토리)

3.2.2 이름 => buildspec.yml **(**주의: 소문자 사용)**

3.2.3 buildspec.yml 에 다음 내용 추가.

** 주위: 마지막 줄에 WAR 출력 파일 명을 본인 명으로 교정하세요

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto17
```

```
    commands:
      - echo Nothing to do in the install phase
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/<st##>-e2edemo.war
```

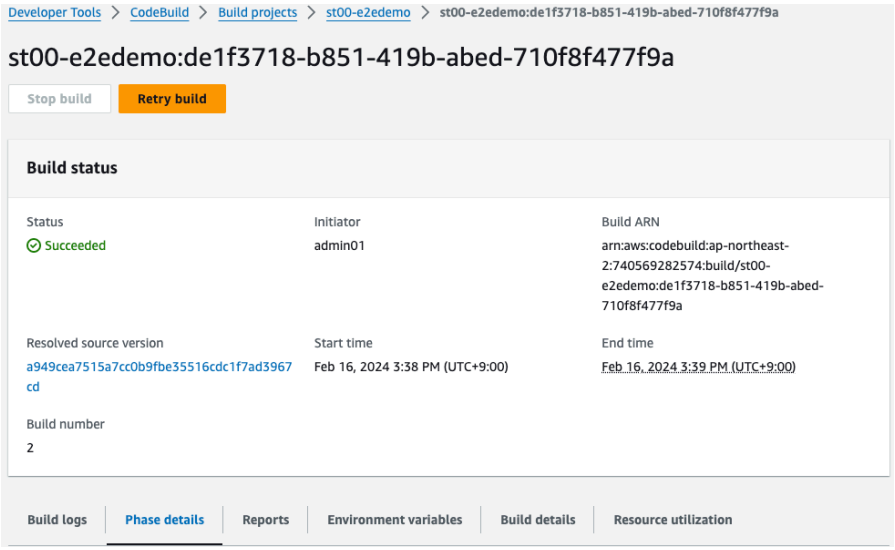
3.3 CodeCommit 와 local repository 에 Commit

3.3.1 위에 단계에서 처럼 pom.xml 과 Buildspec.yml 을 local 과 AWS CodeCommit repository 에 push/commit 합니다. Commit message 에는 “Fourth Commit” 을 사용 합니다. AWS CodeCommit 으로 각각 이동해서 push/commit 확인 합니다.

3.4 AWS CodeBuild 으로 이동 후, 본인 Build project 선택 후 상단에 Start Build => Start now 를 선택해 실행

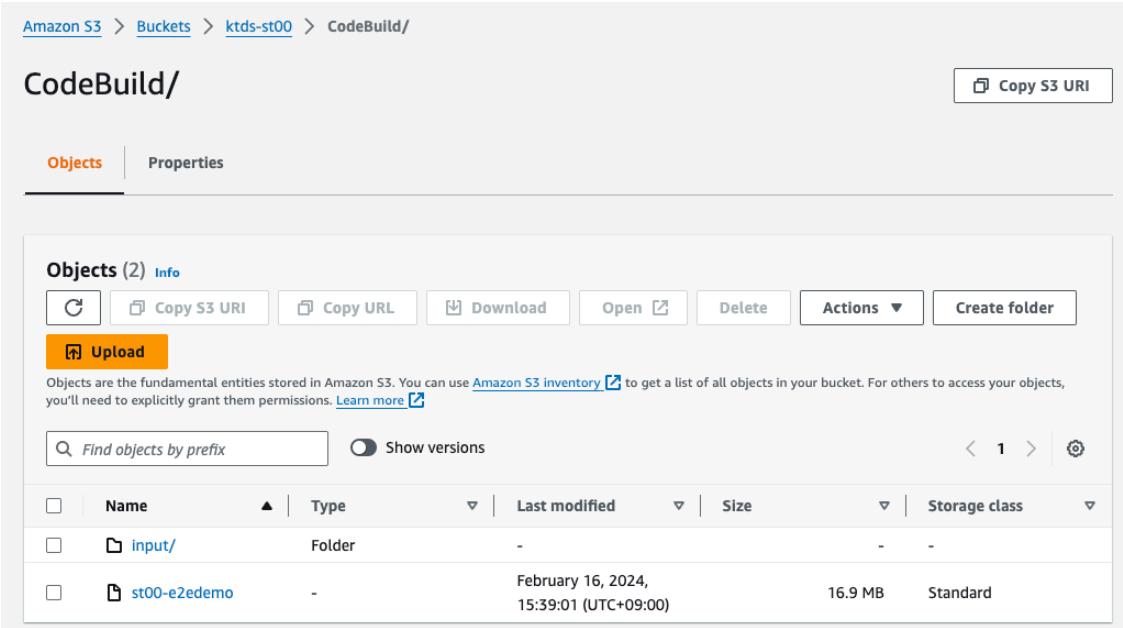
3.4.1 Phase details tab 으로 확인

3.4.2 Build logs tab 선택 후



3.5 AWS S3 에서 Artifact 생성 확인

3.5.1 본인 S3 bucket <ktds-st##> main-bucket 으로 이동 후 CodeBuild 안에 Artifact 가 생성 확인

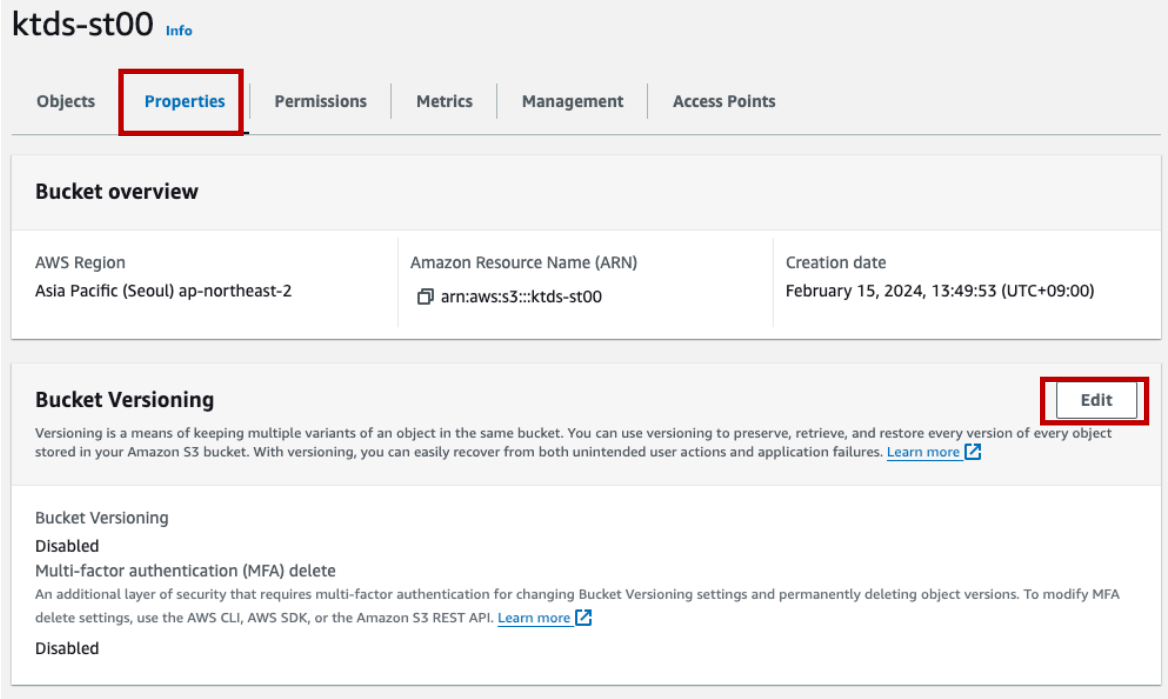


4. CodeBuild 로 새로운 버전 만들기

4.1 <ktds-st##> main bucket 이 버전 기능이 활성화 됨을 확인

4.1.1 Bucket 선택 후 위에 Properties tab 선택

4.1.2 Bucket 버전이 활성화 되어있지 않으면 여기서 교정 가능

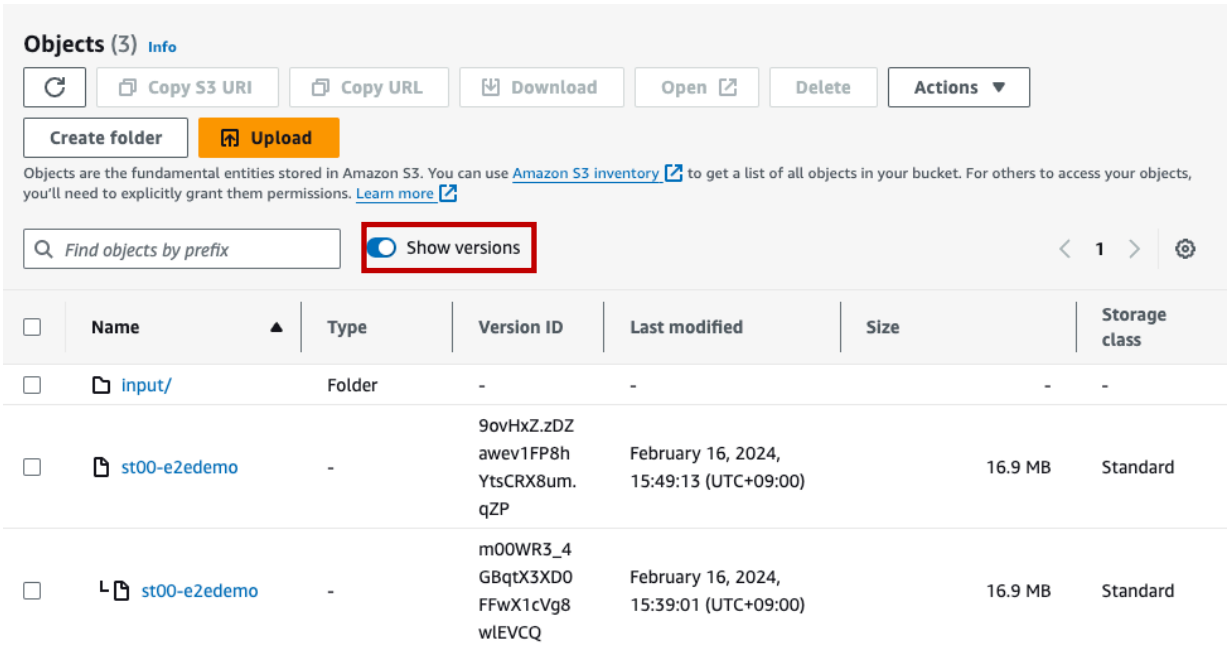


4.2 CodeBuild 로 이동후 다시 빌드 시작

4.3 <ktds-st##> main S3 bucket 에서 버전 확인

4.3.1 <ktds-st##> main bucket/CodeBuild 폴더 로 이동

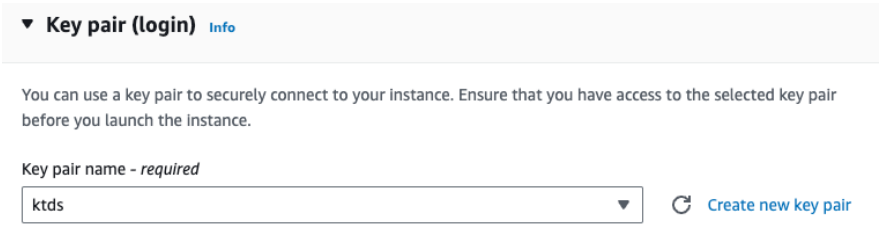
4.3.2 Show versions => 활성화



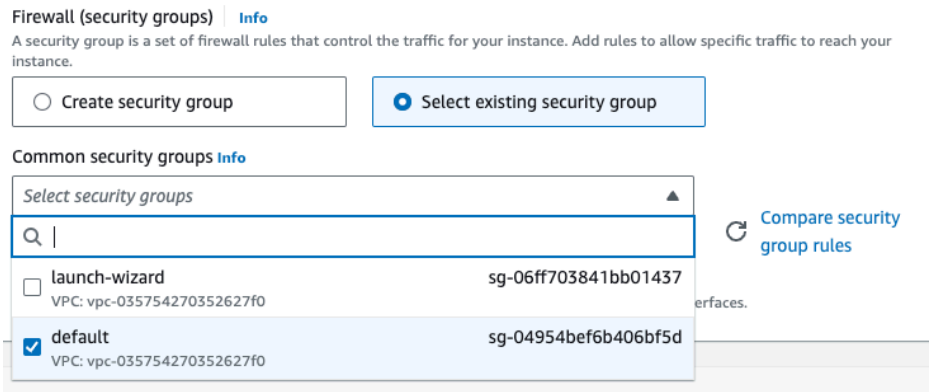
Lab 3 Create AWS EC2 Instance

We will create an EC2 instance to serve the Spring Boot we have created.

1. Set EC2 configurations
   - 1.1 Navigate to EC2 console
   - 1.2 From EC2 Dashboard, [Launch Instance]
   - 1.3 Name and tags
     - 1.3.1 Name => st<##>-e2edemo
   - 1.4 Application and OS Images
     - 1.4.1 Select Amazon Linux
     - 1.4.2 Select Amazon Linux 2 AMI (HVM)
     - 1.4.3 Select 64-bit (x86) architecture
   - 1.5 Instance type
     - 1.5.1 Instance type => t2.micro
   - 1.6 Key pair (login)
     - 1.6.1 Key pair name => ktds



   - 1.7 Network settings
     - 1.7.1 [Select existing security group]
     - 1.7.2 Select default

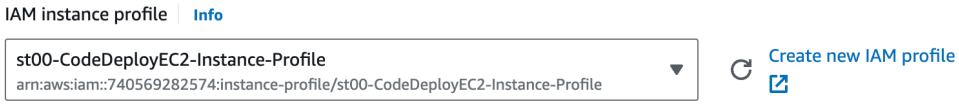


1.8 Configure storage

1.8.1 We can keep 8 GB for now

1.9 Advanced details

1.9.1 IAM instance profile = CodeDeploy-EC2-Instance-Profile



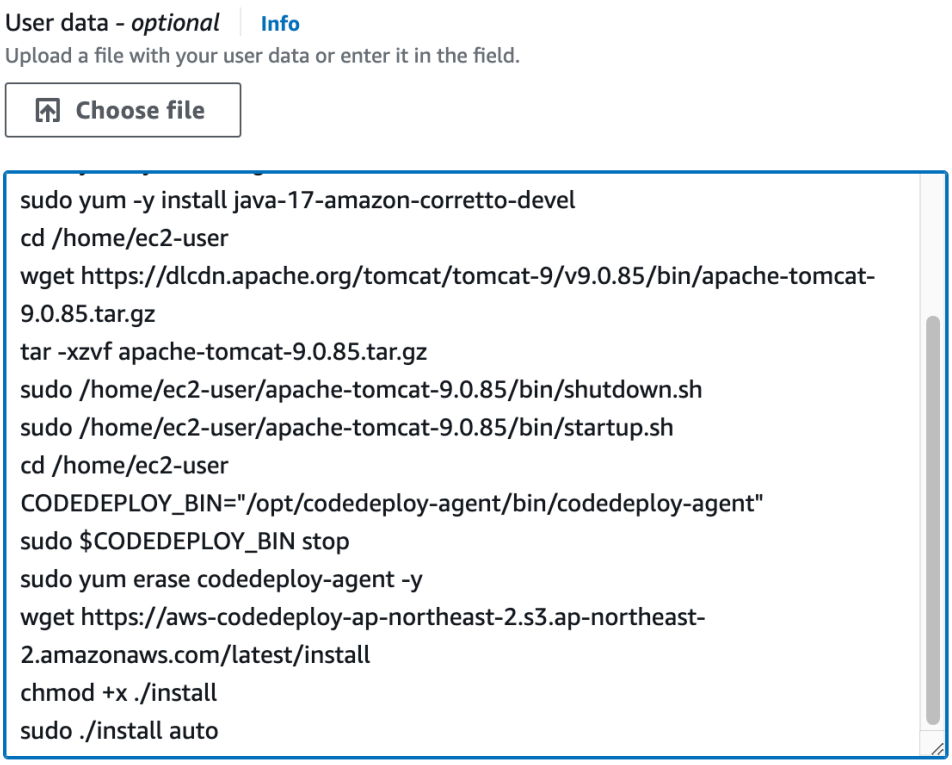
1.9.2 User data => enter the following code

In user data, we will be entering a set of bash commands that we want the EC2 to execute after starting up the EC2 instance. We will perform the following actions via the script:

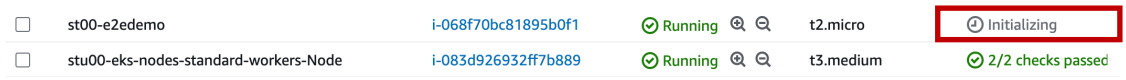
-- update yum
-- install ruby, necessary to run tomcat
-- install wget so we can pull the tomcat9 installation file
-- install java version 17 – we will use the Amazon open source version
-- download and install tomcat 9
-- start tomcat 9
-- install aws-codedeploy-agent, which is the program that the EC2 instance requires to communicate with AWS CodeDeploy

```
#!/bin/bash

sudo yum -y update
sudo yum -y install ruby
sudo yum -y install wget
sudo yum -y install java-17-amazon-corretto-devel
cd /home/ec2-user
wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.90/bin/apache-tomcat-9.0.90.tar.gz
tar -xzvf apache-tomcat-9.0.90.tar.gz
sudo /home/ec2-user/apache-tomcat-9.0.90/bin/shutdown.sh
sudo /home/ec2-user/apache-tomcat-9.0.90/bin/startup.sh
cd /home/ec2-user
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
```

```
sudo $CODEDEPLOY_BIN stop
sudo yum erase codedeploy-agent -y
wget https://aws-codedeploy-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
```



2. Create instance and verify
   - 2.1 Click on [Launch instance]
   - 2.2 Navigate to the EC2 instances page to view your progress
     - 2.2.1 Because the new EC2 instance has to install all the items that we requested in the user data section, it will take some time. Be patient



   - 2.3 Once the status check column show 2/2 pass, we can check on the EC2 instance
   - 2.4 Setup SSH public key
     - 2.4.1 Start Windows Terminal you installed above. (Mac users can use Terminal here)
     - 2.4.2 Copy or move the ktds.pem file to your home directory

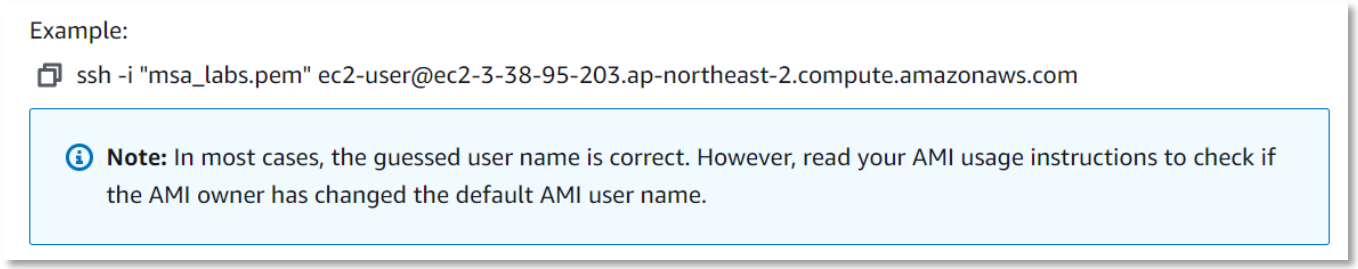
2.4.3 Execute the following to change the security rights to this file. This may not be necessary if you are using Widows Terminal. Do this command only if you are using GitBash.

```
chmod 400 ktds.pem
```

2.5 SSH into your EC2 instance

2.5.1 Click on your instance id.. From there, click on [Connect] to view

2.5.2 From the SSH client tab, copy the example ssh command



2.5.3 Use your terminal program to SSH into your instance. Make sure that ktds.pem file is in your current directory before using the above ssh example command

2.6 Verify your EC2 instance

2.6.1 Make sure that java has been properly installed

```
java -version
```

```
[ec2-user@ip-172-31-37-154 ~]$ java -version
openjdk version "17.0.10" 2024-01-16 LTS
OpenJDK Runtime Environment Corretto-17.0.10.7.1 (build 17.0.10+7-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.10.7.1 (build 17.0.10+7-LTS, mixed mode, sharing)
```

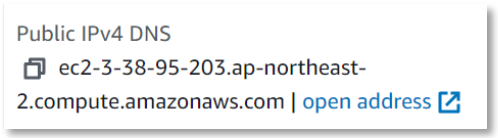
2.7 Make sure that codedeploy-agent has been installed properly.

```
sudo service codedeploy-agent status
```

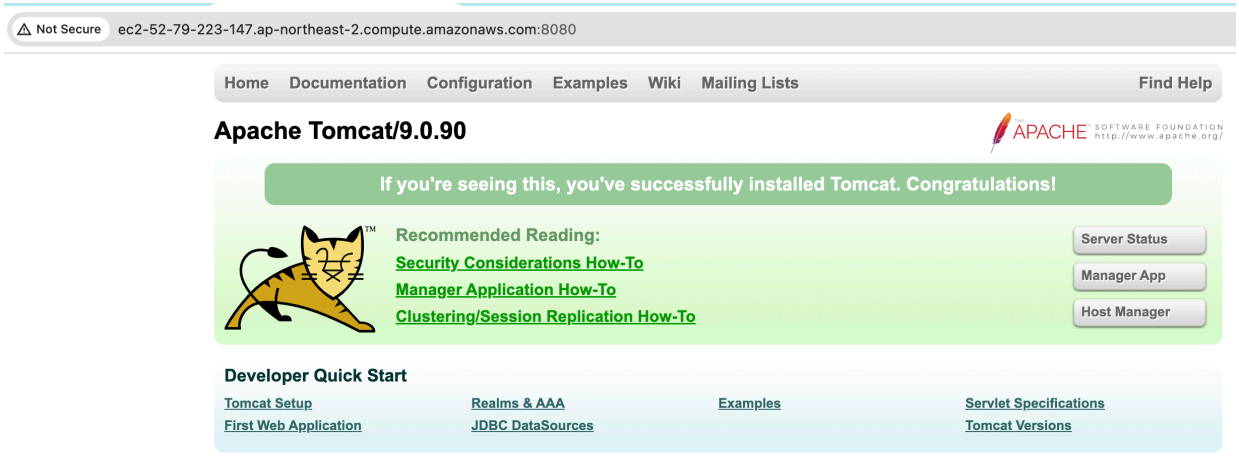
```
[ec2-user@ip-172-31-37-154 ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 6743
```

2.8 Make sure that Tomcat9 has been installed properly and running

2.8.1 From the EC2 dashboard, return to your EC2 instance information and copy the public IPv4 DNS information



2.8.2 From your browser, navigate to <the address you copied above>:8080
Make sure you enter port 8080. If you see the Tomcat screen, you have successfully installed Tomcat on your EC2

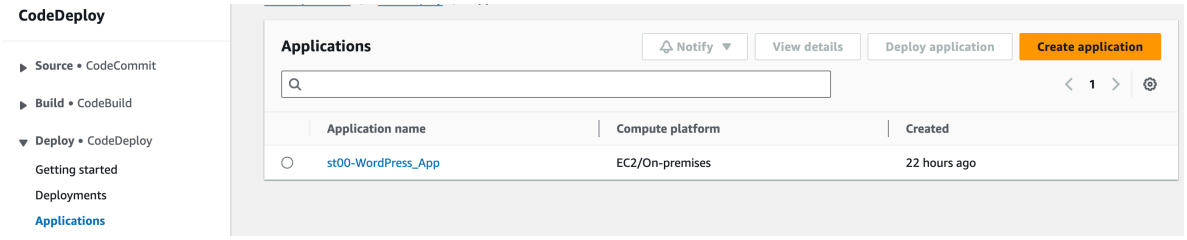


Lab 4 Working with AWS CodeDeploy

1. Create CodeDeploy Application

   1.1 Naviate to AWS CodeDeploy console

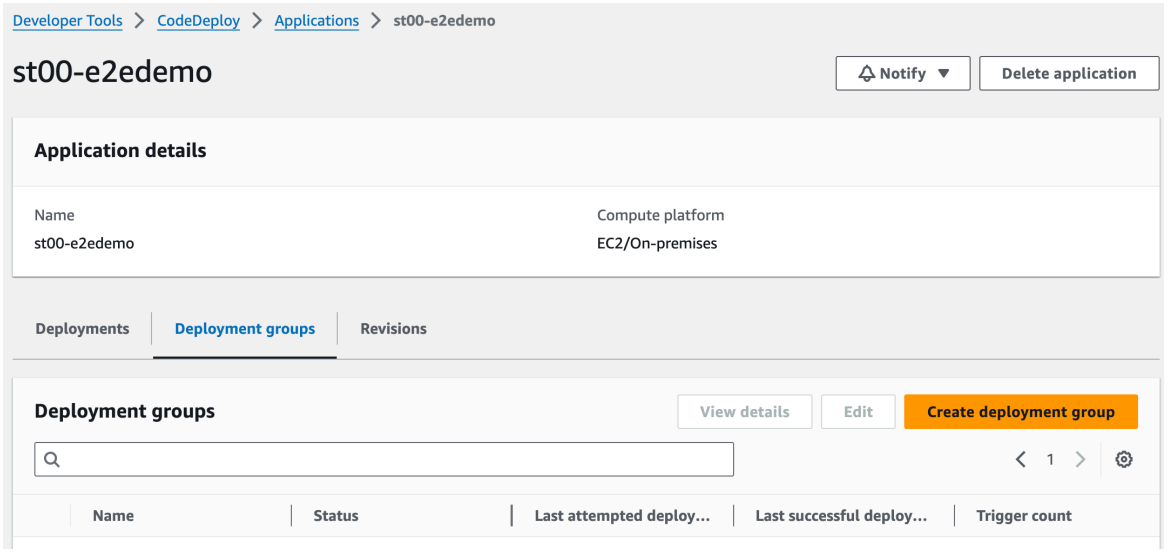
   1.2 Click on [Applications] on left tab and then click [Create applications]



   1.3 Application name => <st##>-e2edemo

   1.4 Compute platform => EC2/On-premises

   1.5 Click [Create application]



2. Create Deployment Group

   2.1 Once you have created a CodeDeploy application, you will have to either select an existing deployment group or create a new one.

   2.2 Select [Create deployment group]

   2.3 Deployment group name => <st##>-e2edemo-deploy-dev-group

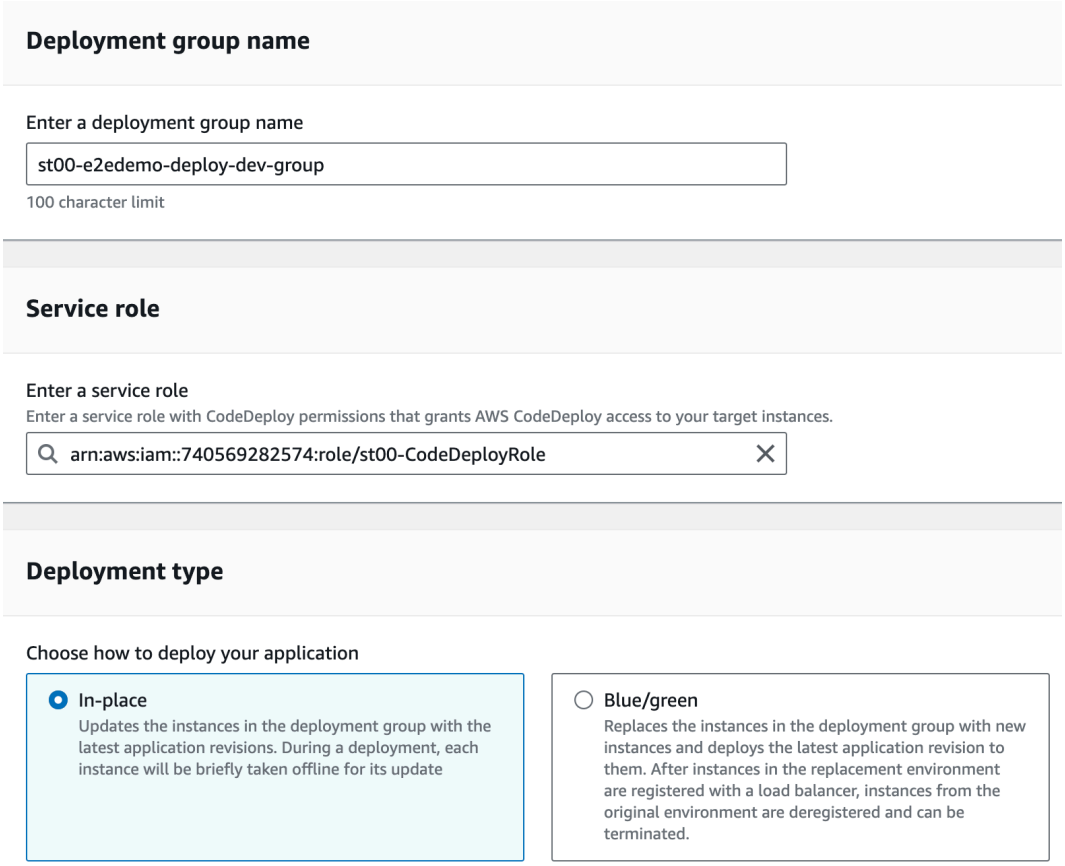
   2.4 Service role => Use CodeDeployServiceRole”

   2.5 Deployment type

   [In-place] 옵션을 선택하여 기존 배포를 업데이트하고 이전 배포를 덮어쓰도록 선택할 수 있습니다. [blue/green] 배포 모드를 사용하도록 선택할 수도 있습니다. Blue/green 배포

모드를 사용하는 경우 blue 세트가 있고, green 세트와 배포는 이 2 개 세트 간에 순환됩니다. 이를 통해 일반 공개하기 전에 최신 배포가 제대로 작동하고 버그가 없는지 확인할 수 있습니다.

For this lab, we will simply choose the [in-place] option.

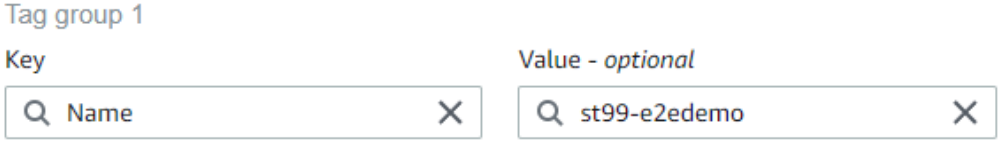


2.6 Environment configuration

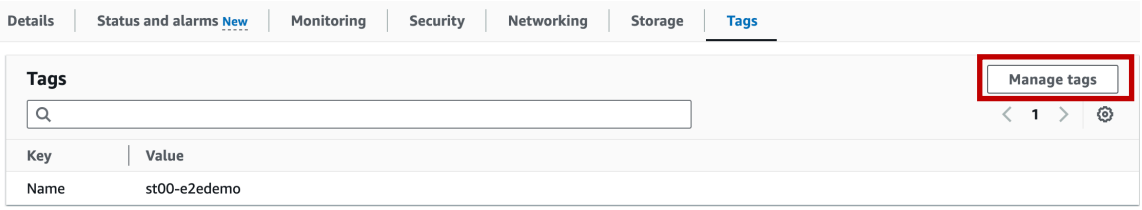
2.6.1 Select [Amazon EC2 instances]

2.6.2 For Tag group 1, Key => Name, Value => <st##>-e2edemo

CodeDeploy 는 태그 정보를 사용하여 배포할 EC2 인스턴스를 결정합니다. 이것이 바로 EC2 인스턴스에 지정된 이름(<st00>-e2edemo)으로 태그를 지정했는지 확인하는 것이 매우 중요한 이유입니다. 키 및 값 상자 내부를 클릭하여 사용 가능한 태그와 값을 표시할 수 있습니다.



2.6.3 EC2 인스턴스에 이름 태그를 설정하는 것을 잊은 경우 지금 설정할 수 있습니다. AWS EC2 콘솔로 이동하여 EC2 인스턴스를 선택합니다. 화면 하단에는 여러 탭이 표시됩니다. [태그] 탭이 있습니다. [태그]를 클릭하세요. [태그 관리]를 클릭하면 새로운 태그를 추가할 수 있습니다.



2.7 Agent configuration with AWS Systems Manager

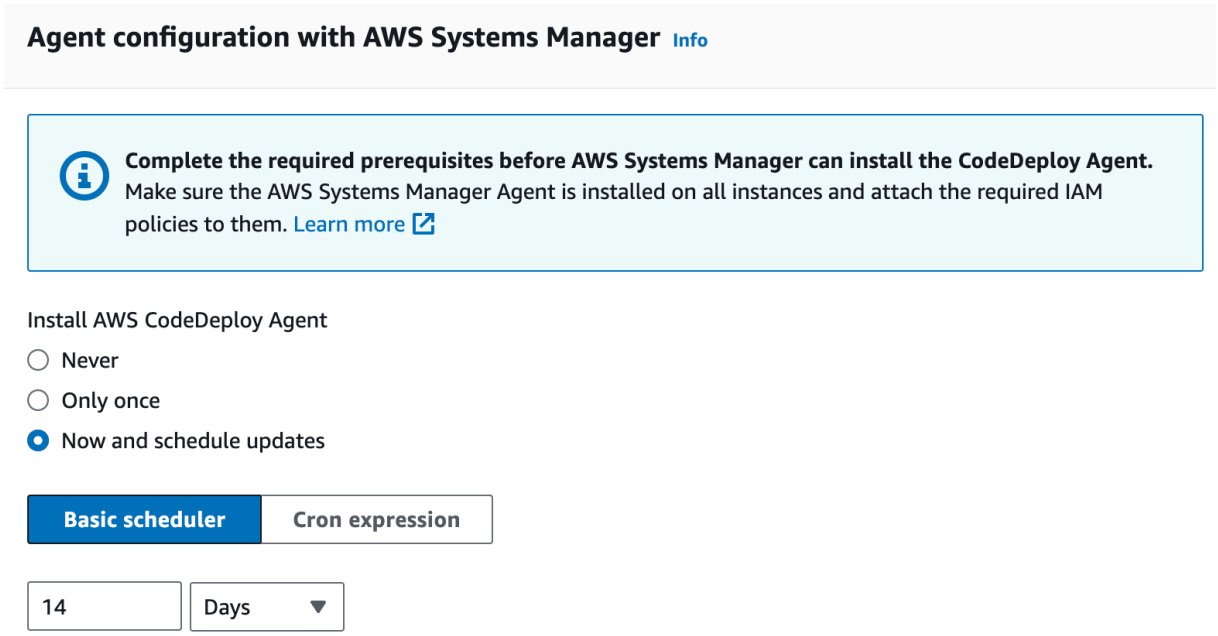
2.7.1 AWS CodeDeploy 에이전트에 대한 업데이트를 예약할 수 있습니다. 하지만 이를 위해서는 AWS SSM 에이전트가 EC2 인스턴스에 설치되어 있어야 합니다. Amazon Linux 2 인스턴스에는 AWS SSM 에이전트가 사전 설치되어 있습니다. 에이전트가 실행 중인지 확인할 수 있습니다.

2.7.2 EC2 인스턴스의 SSH 세션으로 돌아가서 다음 명령을 실행하여 AWS SSM 에이전트가 실행 중인지 확인합니다.

```
sudo systemctl status amazon-ssm-agent
```

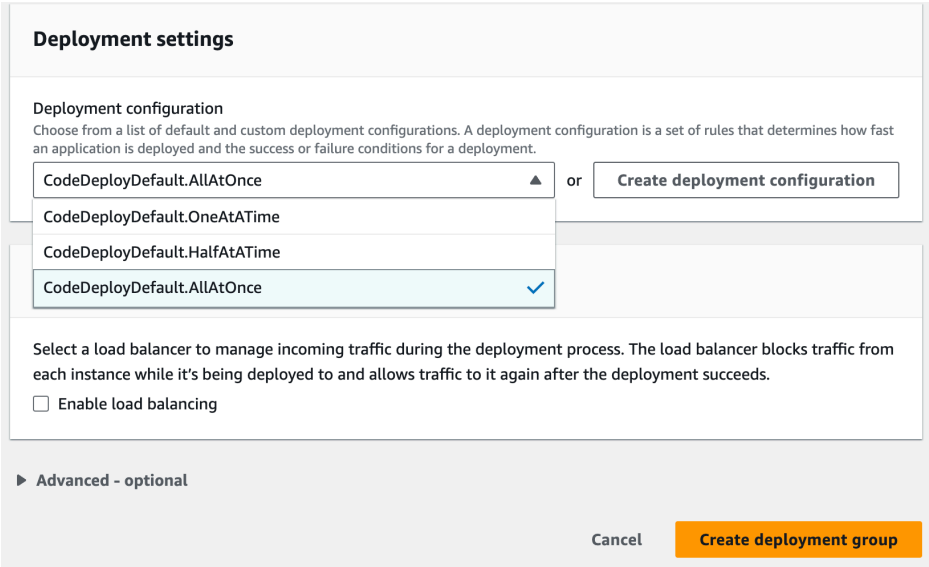
```
[ec2-user@ip-172-31-36-105 ~]$ sudo systemctl status amazon-ssm-agent
● amazon-ssm-agent.service - amazon-ssm-agent
   Loaded: loaded (/usr/lib/systemd/system/amazon-ssm-agent.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2024-02-17 09:16:19 UTC; 49min ago
 Main PID: 3167 (amazon-ssm-agen)
   CGroup: /system.slice/amazon-ssm-agent.service
           ├─3167 /usr/bin/amazon-ssm-agent
           └─3225 /usr/bin/ssm-agent-worker
```

2.7.3 Leave the setting to [Now and schedule updates] Leave the schedule to 14 days



2.8 Deployment settings

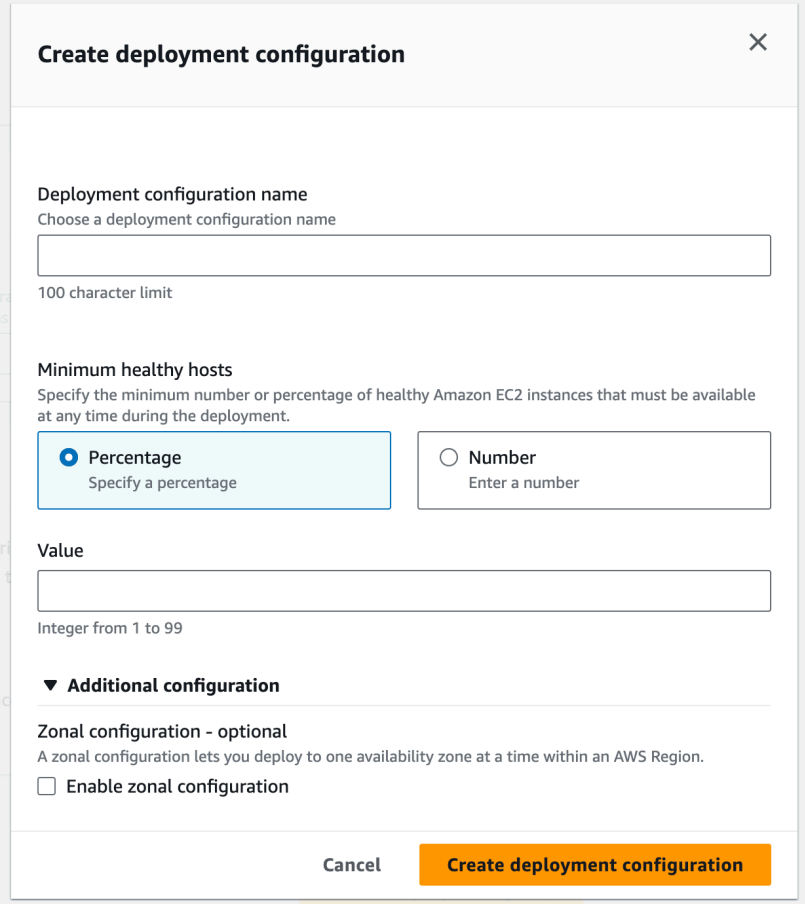
We can choose how AWS CodeDeploy deploys the new build. It provides us with 3 options.



CodeDeployDefault.AllAtOnce 는 일치하는 태그 값(이 경우 이름: <st##>-e2edemo)을 사용하여 모든 인스턴스에 새 빌드를 배포합니다. 이는 실행 중인 모든 인스턴스를 새 빌드로 변경하므로 매우 공격적인 전략입니다.

CodeDeployDefault.HalfAtATime 을 선택하면 한 번에 클러스터의 절반을 업데이트하는 보수적인 옵션을 선택할 수 있습니다.

가장 보수적인 방법은 CodeDeployDefault.OneAtATime 옵션을 사용하여 한 번에 1 개의 서버를 업데이트하는 것이지만, 대규모 클러스터의 경우 시간이 오래 걸릴 수 있습니다. 또는 [Create deployment configuration] 옵션을 사용하여 배포 전략을 사용자 정의할 수도 있습니다. 그런 다음 아래와 같이 전략을 선택할 수 있습니다.



2.8.1 Select CodeDeployDefault.AllAtOnce

2.9 Load balancer

배포 프로세스 중에 들어오는 트래픽을 관리하기 위해 로드 밸런싱을 활성화할 수 있습니다. 단일 EC2 인스턴스에만 배포하므로 이 기능을 비활성화하겠습니다.

2.9.1 Disable load balancing

2.10 Advanced – optional

2.10.1 Click [Create trigger] – 트리거를 설정하고 해당 트리거를 기반으로 알람을 보낼 수 있습니다. 선택할 수 있는 다양한 트리거가 있습니다.

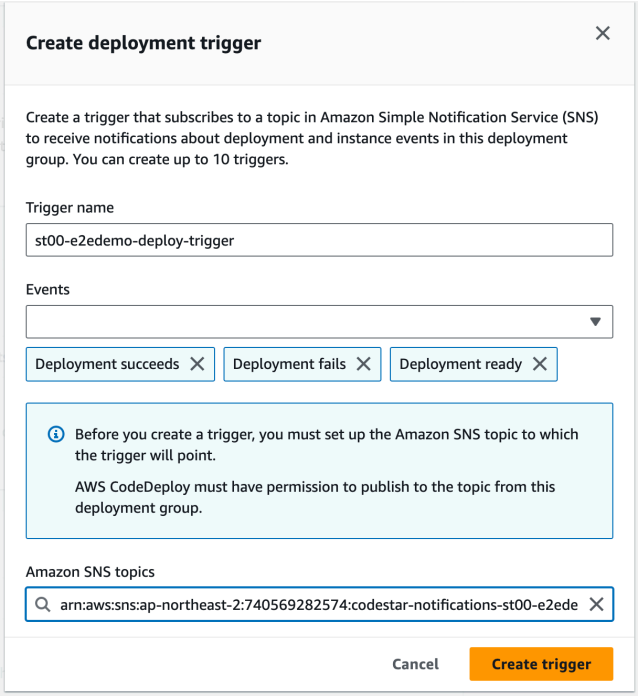
2.10.2 Trigger name => <st##>-e2edemo-deploy-trigger

2.10.3 Events => 드롭다운 메뉴에서 트리거를 생성하려는 이벤트를 선택합니다.

For now, we will select [Deployment succeeds, Deployment fails, Deployment ready]

2.10.4 [Amazon SNS topics] => We must select a SNS topic. Choose the one you created in an earlier lab.

2.10.5 Click on [Create trigger]



2.10.6 [Alarms] allows us to setup CloudWatch alarms. We will skip this step for now.

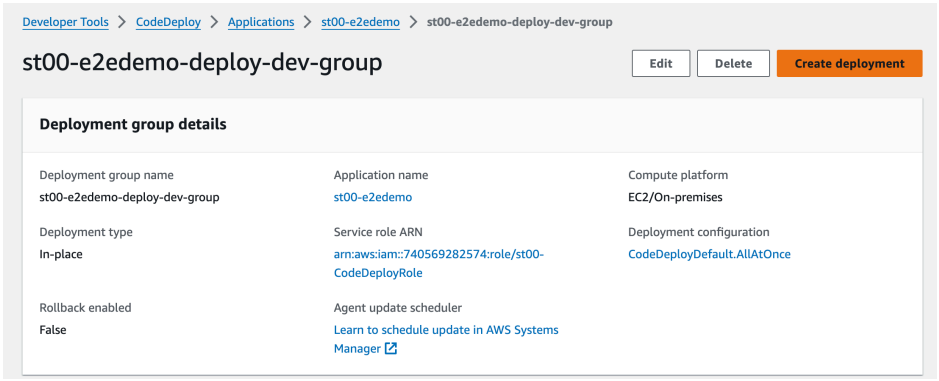
2.10.7 Click [Create deployment group]

3. Create deployment

지금까지 배포 애플리케이션과 배포 그룹을 만들었습니다. 이제 실제 배포를 만들어 보겠습니다.

The wizard should have brought you to the [Create deployment] page.

3.1 Click [Create deployment]

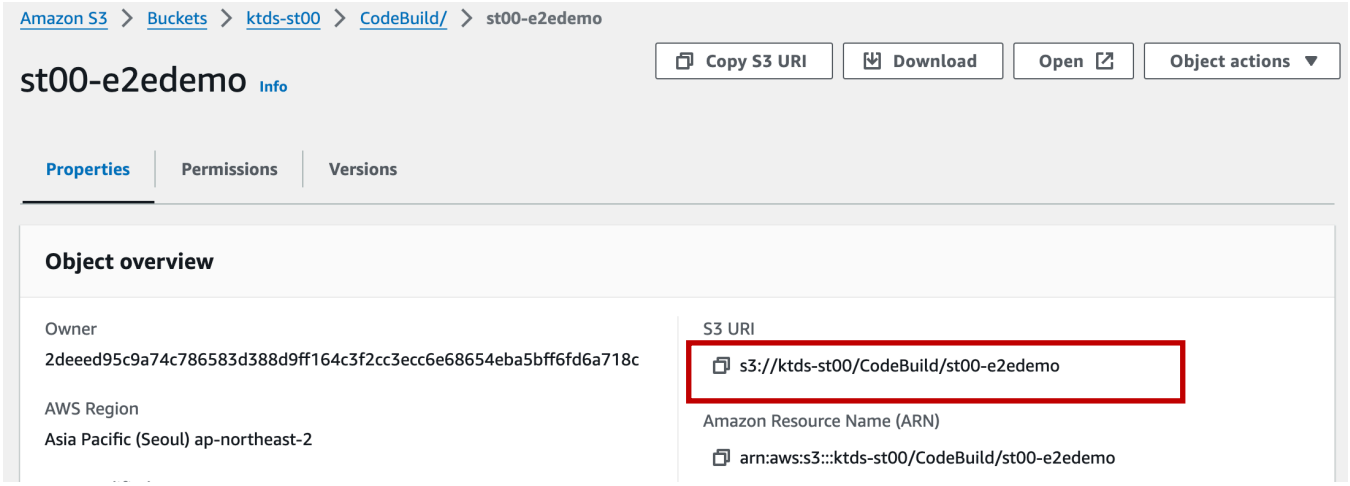


3.2 Deployment settings

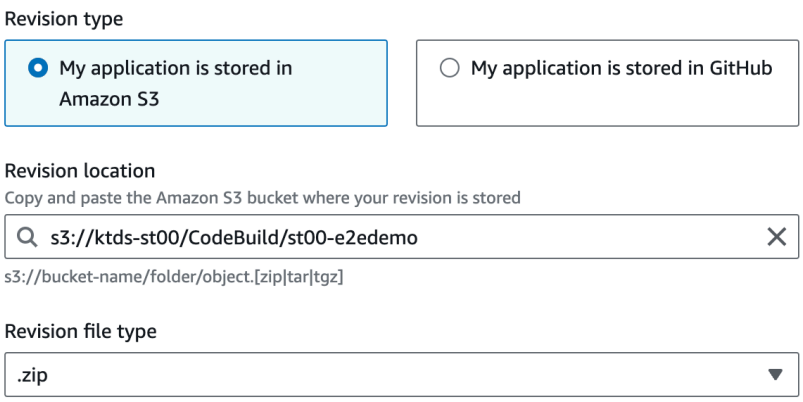
3.2.1 배포 그룹에는 이전 단계에서 생성한 배포 그룹이 이미 표시되어야 합니다.

Revision type – 이전 실습에서는 AWS CodeBuild 를 사용하여 애플리케이션을 구축하고 S3 에 저장했습니다. 해당 빌드를 사용하여 애플리케이션을 배포하겠습니다. [내 애플리케이션이 Amazon S3 에 저장되어 있음]을 선택 상태로 유지하세요.

3.2.2 Revision location – AWS CodeBuild 가 이전 실습에서 빌드 아티팩트를 저장한 s3// 위치를 가져와야 합니다. S3 로 이동하여 이 정보를 얻고 여기에 복사하세요. 우리의 경우에는 "s3://ktds-st##/CodeBuild/st##-e2edemo"여야 합니다.



3.2.3 Once you enter the S3 bucket/folder information, you will have to choose the [Revision file type]. Select **.zip**



3.3 Deployment description

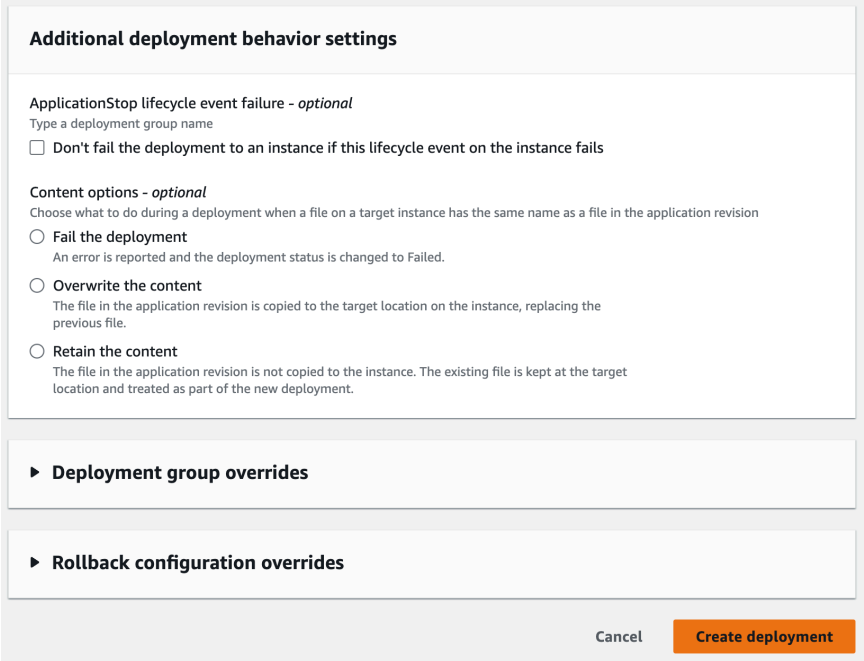
3.3.1 Enter a description of your choice

3.4 Additional deployment behavior settings

대상 인스턴스가 애플리케이션 revision 의 파일과 동일한 이름을 갖는 경우 배포 실패와 같은 몇 가지 추가 동작을 선택할 수 있습니다. 여기서는 아무것도 선택하지 않겠습니다.

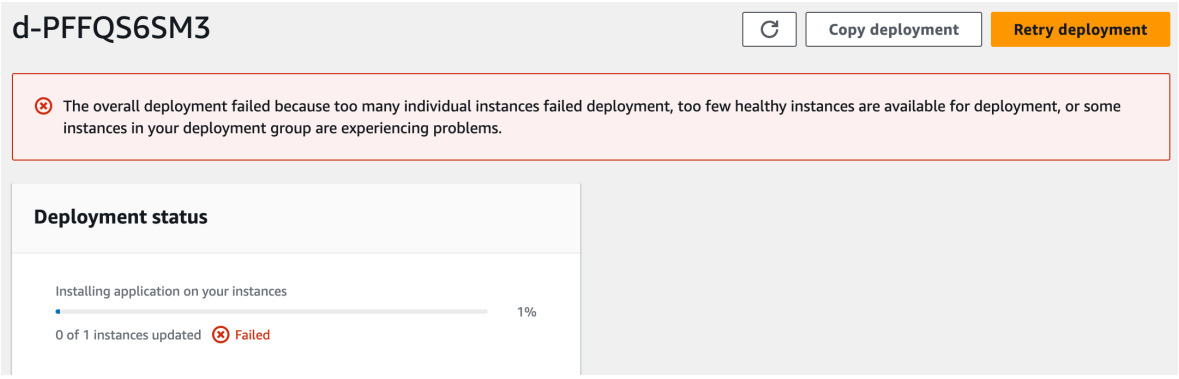
3.5 Deployment group overrides – this section allows us to override any settings that we specified when creating the deployment group. We will not use this.

3.6 Rollback configuration overrides – this section allows us to override the rollback configurations we have previously set. We will not use this.



3.7 Click on [Create deployment]

3.8 [Create deployment]을 클릭하면 CodeDeploy 가 자동으로 실행됩니다. 그러나 이 배포는 실패할 것으로 예상됩니다. 이벤트 링크를 따라가서 실패한 이유를 확인하세요.



3.9 CodeDeploy 가 appspec.yml 파일을 찾을 수 없음을 알 수 있습니다. 여러 단계에서 수행할 작업을 CodeDeploy 에 지시하기 위해 이 파일을 생성해야 합니다.

4. Update Source Code

여러 단계에서 실행할 CodeDeploy 용 bash 셸 스크립트 4 개가 포함된 스크립트 폴더와 함께 appspec.yml 을 생성합니다.

4.1 Create appspec.yml

4.1.1 On the root folder, create a new file and name it appspec.yml

4.1.2 Add the following code to appspec.yml.

Note: <stu##>은 본인 id 으로 수정

```
version: 0.0

os: linux

files:
  - source: target/<st##>-e2edemo.war
    destination: /home/ec2-user/apache-tomcat-9.0.90/webapps/
hooks:
  BeforeInstall:
    - location: scripts/before_install.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/after_install.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
```

```
      runas: root
```

appspec.yml 파일은 CodeDeploy 단계에서 발생해야 하는 작업을 지정합니다. 설치 전, 설치 후, 애플리케이션이 중지되고 애플리케이션이 시작될 때 실행될 쉘 스크립트를 지정했습니다. 쉘 스크립트의 세부사항은 다음 단계에서 제공됩니다.

4.2 Create bash scripts for phases

4.2.1 프로젝트의 루트디렉토리에 scripts 폴더를 생성

4.2.2 다음 코드 사용해서 after_install.sh 를 scripts 디렉토리에 작성

```
#!/bin/bash
#
```

4.2.3 다음 코드 사용해서 before_install.sh 를 scripts 디렉토리에 작성

```
#!/bin/bash
rm -rf /home/ec2-user/apache-tomcat-9.0.90/webapps/st##-e2edemo*
```

4.2.4 다음 코드 사용해서 start_server.sh 를 scripts 디렉토리에 작성

```
#!/bin/bash
/home/ec2-user/apache-tomcat-9.0.90/bin/startup.sh
```

4.2.5 다음 코드 사용해서 stop_server.sh 를 scripts 디렉토리에 작성

```
#!/bin/bash
isExistApp=`pgrep java`
if [[ -n  $isExistApp ]]; then
   /home/ec2-user/apache-tomcat-9.0.90/bin/shutdown.sh
fi
```

4.3 From HelloWorldController.java, change the greetings to reflect that we are now distributing for CodeDeploy.

```java
@GetMapping("/hello")
public String sayHello() {
    return "Hi. Welcome to MSA cloud apps - CodeDeploy-v1";
}
```

4.4 Save and commit the changes to local and AWS CodeCommit

5. Execute AWS CodeBuild and check

5.1 Use CodeBuild to create a new build

5.1.1 Navigate to AWS CodeBuild console

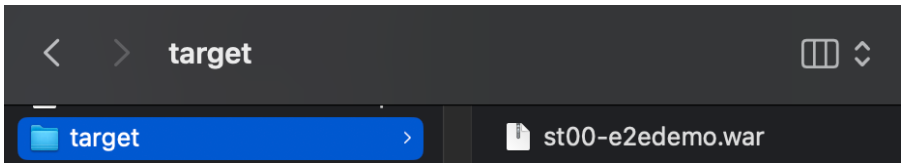
5.1.2 Navigate to your [Build project] – for our lab, this should be <st##>-e2edemo

5.1.3 Run the build

5.1.4 Navigate to S3 bucket where the artifacts have been created – for our lab, this should be <st##>-main-bucket/CodeBuild

5.1.5 새로 생성된 아티팩트를 다운로드합니다. 최신 빌드인지 확인하려면 생성 날짜를 확인하세요. 필요에 따라 다른 버전을 볼 수도 있습니다.

5.1.6 From the downloads directory, unzip the artifact and observe the contents of the artifact.



5.1.7 다운로드한 파일에는 "target" 하위 디렉터리와 빌드에서 생성된 <st##>-e2edemo.war 파일만 포함되어 있습니다. appspec.yml 과 스크립트가 누락되었습니다. 무엇이 잘못되었나요?

5.2 Modify buildspec.yml to fix the problem

5.2.1 VS Code 에서 buildspec.yml 파일을 살펴보고, 다음 코드가 포함된 아티팩트 섹션이 있는지 확인하세요. 이는 AWS CodeBuild 가 아티팩트를 생성하는 데 사용하는 지침입니다. 검토해 보면, appspec.yml 과 CodeDeploy 에 필요한 모든 스크립트를 생성하기 위한 지침이 누락되었습니다.

```
artifacts:
  files:
    - target/st##-e2edemo.war
```

5.2.2 Modify buildspec.yml as shown below to add the necessary output artifacts.

** **주의: <st##>은 본인 id 를 변경**

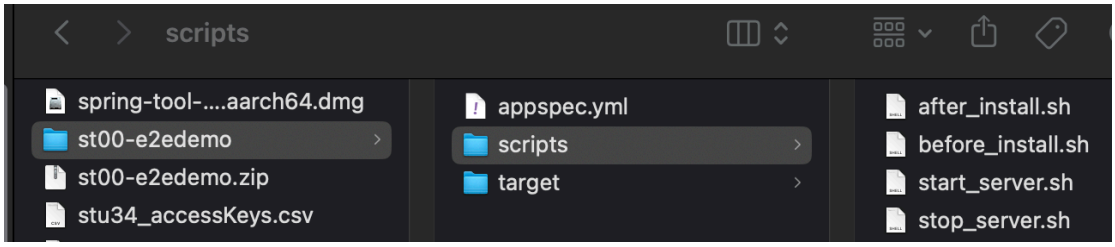
```
artifacts:
  files:
    - target/<st##>-e2edemo.war
    - appspec.yml
    - scripts/before_install.sh
    - scripts/after_install.sh
    - scripts/start_server.sh
    - scripts/stop_server.sh
```

5.2.3 Save, commit and push to local and AWS Commit

5.2.4 Navigate to AWS CodeCommit to verify the changes have been added

5.2.5 Re-execute the AWS CodeBuild

5.2.6 Download the artifact again and check that this time, we have all the necessary files

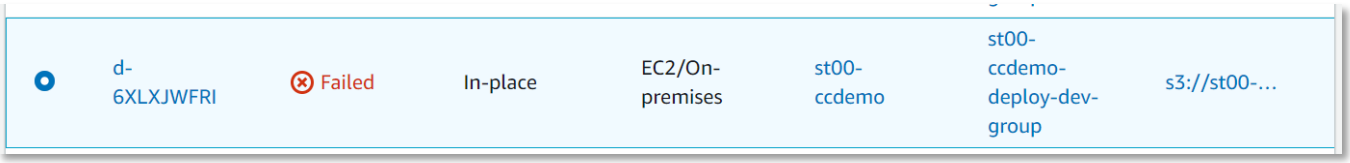


5.3 Re-execute your AWS CodeDeploy

5.3.1 Navigate to AWS CodeDeploy console

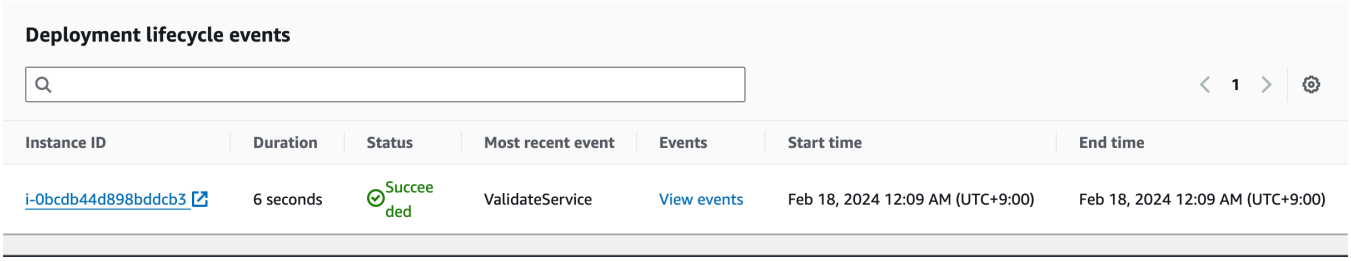
5.3.2 Select [Deployments] from the left-side menu

5.3.3 Select the last failed deployment



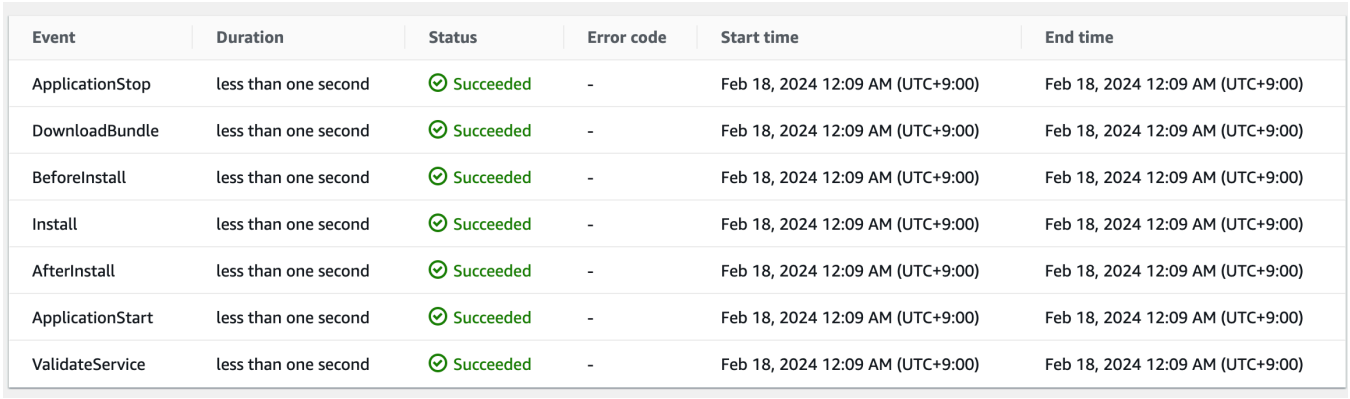
5.3.4 [Retry deployment] will now be enabled on the top menu. Select and click it to restart your deployment

5.3.5 This time, CodeDeploy should be successful. Select the successful [Deployment Id] and scroll down to [Deployment lifecycle events]. Click on the [View events] and verify that all the phases have completed successfully.



Deployment lifecycle events

Instance ID	Duration	Status	Most recent event	Events	Start time	End time
i-0bcdb44d898bddcb3	6 seconds	Succeeded	ValidateService	View events	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)



Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
DownloadBundle	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
BeforeInstall	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
Install	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
AfterInstall	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
ApplicationStart	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)
ValidateService	less than one second	Succeeded	-	Feb 18, 2024 12:09 AM (UTC+9:00)	Feb 18, 2024 12:09 AM (UTC+9:00)

6. Verify proper deployment from EC2 instance

6.1 Check Tomcat9 deployment

6.1.1 Navigate to the SSH session of your EC2 instance

6.1.2 Change to root user with the following code

```
sudo su -
```

6.1.3 Change to the Tomcat9 folder and check that the new artifacts have been uploaded

```
cd /home/ec2-user/apache-tomcat-9.0.90/webapps
ls
```

```
[root@ip-172-31-36-105 ~]# cd /home/ec2-user/apache-tomcat-9.0.85/webapps/
[root@ip-172-31-36-105 webapps]# ls
docs  examples  host-manager  manager  ROOT  st00-e2edemo  st00-e2edemo.war
[root@ip-172-31-36-105 webapps]#
```

6.2 Check AWS-CodeDeploy-Agent log

codedeploy-agent 는 모든 아티팩트의 압축을 풀고 /opt/code-deploy-agent/deployment-root 에서 작동합니다. 해당 디렉터리로 이동하여 로그 디렉터리를 찾으세요. 로그 디렉터리로 이동하여 배포 로그 정보를 추적합니다. 아래와 비슷한 내용이 표시됩니다.

```
[root@ip-172-31-36-105 webapps]# cd /opt/codedeploy-agent/deployment-root/
[root@ip-172-31-36-105 deployment-root]# ls
9eae919a-65e3-456b-8ba7-36feb1478d4d  deployment-instructions  deployment-logs  ongoing-deployment
[root@ip-172-31-36-105 deployment-root]# cd deployment-logs/
[root@ip-172-31-36-105 deployment-logs]# tail codedeploy-agent-deployments.log
[2024-02-17 15:09:17.740] [d-NUDK6HVM3]LifecycleEvent - ApplicationStop
[2024-02-17 15:09:17.740] [d-NUDK6HVM3]Script - scripts/stop_server.sh
[2024-02-17 15:09:17.768] [d-NUDK6HVM3][stderr]NOTE: Picked up JDK_JAVA_OPTIONS:  --add-opens=java.base/java.lang=ALL-UNNAM
ED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.co
ncurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
[2024-02-17 15:09:19.825] [d-NUDK6HVM3]LifecycleEvent - BeforeInstall
[2024-02-17 15:09:19.826] [d-NUDK6HVM3]Script - scripts/before_install.sh
[2024-02-17 15:09:21.930] [d-NUDK6HVM3]LifecycleEvent - AfterInstall
[2024-02-17 15:09:21.930] [d-NUDK6HVM3]Script - scripts/after_install.sh
[2024-02-17 15:09:23.004] [d-NUDK6HVM3]LifecycleEvent - ApplicationStart
[2024-02-17 15:09:23.004] [d-NUDK6HVM3]Script - scripts/start_server.sh
[2024-02-17 15:09:23.031] [d-NUDK6HVM3][stdout]Tomcat started.
[root@ip-172-31-36-105 deployment-logs]#
```

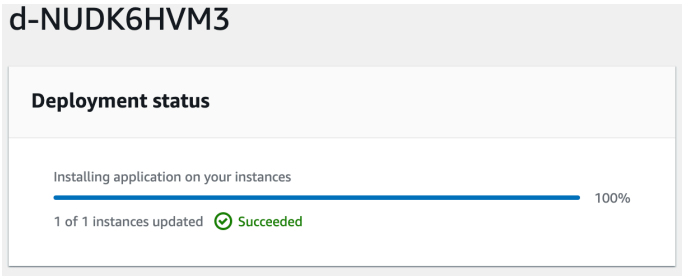
각 수명 주기 이벤트에는 scripts 디렉터리 내부의 bash 셸 스크립트가 실행된 것으로 표시됩니다.

6.3 Check the AWS CodeDeploy Agent activity

6.3.1 /opt/codedeploy-agent/deployment-root 로 다시 이동하여 마지막 배포 디렉터리(생성된 영숫자 디렉터리)를 찾습니다. 그것으로 이동하십시오.

6.3.2 최신 하위 디렉토리를 찾으십시오. 이름은 성공한 배포 ID 와 일치해야 합니다. 해당 디렉터리로 이동합니다.

```
[root@ip-172-31-36-105 deployment-root]# ls
9eae919a-65e3-456b-8ba7-36feb1478d4d  deployment-instructions  deployment-logs  ongoing-deployment
[root@ip-172-31-36-105 deployment-root]# ls -ltr
total 0
drwxr-xr-x 2 root root  46 Feb 17 14:10 deployment-logs
drwxr-xr-x 7 root root 101 Feb 17 15:09 9eae919a-65e3-456b-8ba7-36feb1478d4d
drwxr-xr-x 2 root root 247 Feb 17 15:09 deployment-instructions
drwxr-xr-x 2 root root   6 Feb 17 15:09 ongoing-deployment
[root@ip-172-31-36-105 deployment-root]# cd 9eae919a-65e3-456b-8ba7-36feb1478d4d/
[root@ip-172-31-36-105 9eae919a-65e3-456b-8ba7-36feb1478d4d]# ls -ltr
total 0
drwxr-xr-x 4 root root 62 Feb 17 14:10 d-UVCPBOUM3
drwxr-xr-x 4 root root 62 Feb 17 14:22 d-JSSCQDUM3
drwxr-xr-x 4 root root 62 Feb 17 14:47 d-Q5W02JMAG
drwxr-xr-x 4 root root 62 Feb 17 14:57 d-UCOT85UM3
drwxr-xr-x 4 root root 62 Feb 17 15:09 d-NUDK6HVM3
```



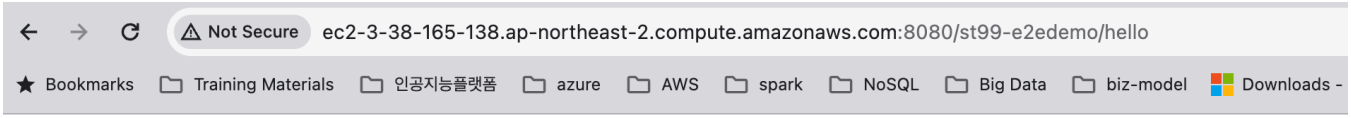
6.3.3 일치하는 하위 디렉터리로 이동합니다. 여기서는 로그와 아티팩트 파일을 찾을 수 있습니다. EC2 인스턴스 내의 codedeploy-agent 는 이 모든 작업을 수행하기 위해 AWS CodeDeploy 와 성공적으로 통신했습니다. 핵심 요소 중 하나는 EC2 인스턴스를 생성할 때 연결한 IAM 인스턴스 프로필입니다. 이는 EC2 에 이러한 배포 활동을 수행하는 데 필요한 모든 IAM 권한을 부여했습니다.

```
[root@ip-172-31-36-105 deployment-archive]# pwd
/opt/codedeploy-agent/deployment-root/9eae919a-65e3-456b-8ba7-36feb1478d4d/d-NUDK6HVM3/deployment-archive
[root@ip-172-31-36-105 deployment-archive]# ls -l
total 4
-rw-r--r-- 1 root root 519 Feb 17 14:51 appspec.yml
drwxr-xr-x 2 root root 100 Feb 17 15:09 scripts
drwxr-xr-x 2 root root  30 Feb 17 15:09 target
[root@ip-172-31-36-105 deployment-archive]# ls -l target
total 19248
-rw-r--r-- 1 root root 19707225 Feb 17 14:52 st00-e2edemo.war
[root@ip-172-31-36-105 deployment-archive]# ls -l scripts/
total 16
-rwxr-xr-x 1 root root  15 Feb 17 14:51 after_install.sh
-rwxr-xr-x 1 root root  77 Feb 17 14:51 before_install.sh
-rwxr-xr-x 1 root root  64 Feb 17 14:51 start_server.sh
-rwxr-xr-x 1 root root 126 Feb 17 14:51 stop_server.sh
```

6.4 Check service deployment on the browser

6.4.1 On your browser, navigate to the EC2 instance IPv4 DNS address at port 8080

6.4.2 Tomcat 이 실행 중인지 확인해야 합니다. 거기에서 <st##>-e2edemo/hello 로 이동하여 새 배포를 확인합니다.



Thank you for working so hard. Welcome to CNA - CodeDeploy-v5.0