


- 배포 자동화 준비 : CodeDeploy 생성 및 연동하기 (Travis-ci, S3)
- 1. EC2에 IAM 역할 추가하기 (IAM : role-ec2-codedeploy)
- 2. EC2 환경에 AWS CodeDeploy 에이전트 설치
- 3. CodeDeploy IAM 역할 생성 (IAM : role-codedeploy)
- 4. CodeDeploy 생성
- 5. Travis-ci, S3, CodeDeploy 연동
 - 1) EC2 디렉토리 생성
 - 2. 스프링 프로젝트 appspec.yml 파일 추가
 - 3. .travis.yml 파일에 CodeDeploy 설정 추가
- 6. Git Push 테스트
 - 파일 및 인스턴스 정리



2) Travis-ci, S3, CodeDeploy 연동

1. EC2에 IAM 역할 추가하기 (ec2 → CodeDeploy 접근 권한)

2. EC2 환경에 AWS CodeDeploy 에이전트 설치

3. CodeDeploy IAM 역할 생성 (CodeDeploy → ec2 접근 권한)

4. CodeDeploy 생성 (3번에서 생성한 IAM 역할 사용)

5. Travis-ci, S3, CodeDeploy 연동하기

6. Git push 테스트

1. AWS CodeDeploy 배포 내역 생성

2. ~/app/step2/zip/ 에 프로젝트 파일 잘 도착하면 연동 성공

→

2-5) Travis-ci, S3, CodeDeploy 연동

1. EC2 디렉토리 생성 (~/app/step2 && ~/app/step2/zip)

2. 스프링 프로젝트에 appspec.yml 파일 추가 (AWS CodeDeploy 설정 파일)

3. .travis.yml 파일에 CodeDeploy 설정 추가

배포 자동화 준비 : CodeDeploy 생성 및 연동하기 (Travis-ci, S3)

배포 자동화 준비 : CodeDeploy 생성 및 연동하기 (Travis-ci, S3)

이제 AWS 배포 시스템 CodeDeploy를 생성해준다. 해당 과정에서는 총 2개의 IAM(ec2에 필요한 IAM, codeDeploy에 필요한 IAM)을 생성한다. 이제 CodeDeploy를 이에 연동시켜주면 거의 모든게 끝난다. EC2 환경에 파일을 받을 디렉토리 생성과 몇 가지 설정을 추가해주면 된다.

IAM 역할, 사용자

이전에 #2-1에서는 Travis CI를 위한 IAM 사용자를 만들어줬다면 이번에는 IAM 역할을 생성해 줄 것이다. IAM 역할과 사용자의 차이점은 다음과 같다.

- 역할
 - AWS 서비스에만 할당할 수 있는 권한
 - EC2, CodeDeploy, SQS 등
- 사용자
 - AWS 서비스 외에 사용할 수 있는 권한
 - 로컬 PC, IDC 서버 등


Travis-CI, S3, CodeDeploy를 연동하기에 앞서 일단 **AWS CodeDeploy**부터 생성해보자.

1. EC2에 IAM 역할 추가하기 (IAM : role-ec2-codedeploy)

1) IAM > 역할 > 역할 만들기 > AWS 서비스 선택 > EC2 선택

지금 만들 권한은 EC2에서 사용할 것이기 때문에 사용자가 아닌 역할로 처리한다. 서비스 선택에서는 EC2를 선택해주면 된다.

신뢰할 수 있는 유형의 개체 선택

 AWS 서비스 EC2, Lambda 및 기타	 다른 AWS 계정 귀하 또는 타사 소유	 웹 ID Cognito 또는 OpenID 공급
--	--	---

AWS 서비스가 사용자를 대신하여 작업을 수행하도록 허용합니다. [자세히 알아보기](#)

사용 사례 선택

일반 사용 사례

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



Lambda
Allows Lambda functions to call AWS services on your behalf.

역할 만들기

2) 정책 선택 > EC2RoleForA.. 검색 후 선택> 태그 (키, 값) 입력 > 최종 확인 후 생성

정책에선 EC2RoleForA를 검색하여 AmazonEC2RoleforAWS-CodeDeploy를 선택한다.

정책 필터

	정책 이름
<input checked="" type="checkbox"/>	 AmazonEC2RoleforAWSCodeDeploy
<input type="checkbox"/>	 AmazonEC2RoleforAWSCodeDeployLimited

정책 추가

태그는 본인이 원하는 이름으로 작성한다.

IAM 태그는 사용자 역할에 추가할 수 있는 키-값 페어입니다. 태그는 이메일 주소와 같은 사용자 정보를 포함하거나 직책과 ; 대한 액세스를 구성, 추적 또는 제어할 수 있습니다. [자세히 알아보기](#)

키	값(선택 사항)
<input type="text" value="Name"/>	<input type="text" value="role-ec2-codedeploy"/>
<input type="text" value="새 키 추가"/>	<input type="text"/>

49 태그를 더 추가할 수 있습니다.

태그 추가

마지막으로 역할의 이름을 등록하고 생성을 완료한다.

요한 정보를 입력하고 이 역할을 검토하십시오.

역할 이름*

role-ec2-codedeploy

영숫자 및 '+=,.-_' 문자를 사용합니다. 최대 64자입니다.

역할 설명

Allows EC2 instances to call AWS services on your behalf.

최대 1000자입니다. 영숫자 및 '+=,.-_' 문자를 사용합니다.

신뢰할 수 있는 개체 AWS 서비스: ec2.amazonaws.com

정책



AmazonEC2RoleforAWSCodeDeploy [↗](#)

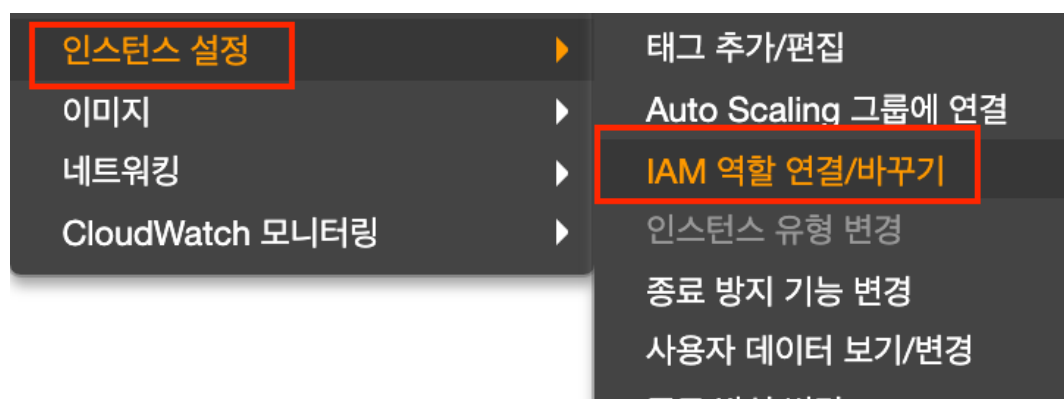
권한 경계

권한 경계가 설정되지 않았습니다

최종 확인

3) EC2 인스턴스로 이동 > 인스턴스 설정 > IAM 역할 연결/바꾸기 > IAM 역할(role-ec2-codedeploy) 선택 > 재부팅

역할 생성이 끝났으면 EC2 서비스로 이동하여 EC2 인스턴스 목록으로 이동한 뒤 본인의 인스턴스를 마우스 오른쪽 버튼으로 눌러 IAM 역할 연결/바꾸기를 선택한다.



역할 바꾸기

방금 생성한 IAM 역할을 선택 후 적용한다. 그리고 역할 선택이 완료되었으면 EC2 인스턴스를 재부팅해준다. 재부팅을 해야만 역할이 정상적으로 적용된다.

IAM 역할 연결/바꾸기

인스턴스에 연결할 IAM 역할을 선택합니다. IAM 역할이 없는 경우 [새 IAM 역할 생성]을 선택하여 IAM 콘솔에서 역할을 생성합니다. IAM 역할이 이미 인스턴스에 연결되어 있으면 선택한 IAM 역할이 기존 역할을 대체합니다.

인스턴스 ID i-02010aaaa30a909db (spring-deploy-test) ⓘ

IAM 역할

role-ec2-codedeploy



새 IAM 역할 생성 ⓘ

역할 선택

2. EC2 환경에 AWS CodeDeploy 에이전트 설치

EC2에 접속해서 다음 명령어를 입력한다.

- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/latest/install . --region ap-northeast-2`

내려받기가 성공했다면 다음과 같은 메시지가 콘솔에 출력된다.

```
[ec2-user@spring-deploy-test ~]$ aws s3 cp s3://aws-codedeploy-ap-northeast-2/latest/install . --region ap-northeast-2
download: s3://aws-codedeploy-ap-northeast-2/latest/install to ./install
```

내려받기 성공

install 파일에 실행 권한이 없으니 실행 권한을 추가한다.

- `chmod +x ./install`

install 파일로 설치를 진행한다.

- `sudo ./install auto`
- 만약 `/usr/bin/env: ruby: No such file or directory` 에러가 발생한다면 `yum install`로 루비를 설치하면 된다.
 - `sudo yum install ruby`

설치가 끝났으면 Agent가 정상적으로 실행되고 있는지 상태 검사를 합니다.

- `sudo service codedeploy-agent status`

다음과 같이 running 메시지가 출력되면 정상이다.

```
[ec2-user@spring-deploy-test ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 3333
```

aws codedeploy agent running

3. CodeDeploy IAM 역할 생성 (IAM : role-codedeploy)

CodeDeploy에 접근하려면 EC2와 마찬가지로 권한이 필요하다. AWS의 서비스이니 IAM역할을 생성한다.

1) IAM > 역할 > 역할 만들기 > CodeDeploy 선택

또는 서비스를 선택하여 해당 서비스의 사용 사례 확인

API Gateway	CodeBuild	EMR Containers	IoT SiteWise	RDS
AWS Backup	CodeDeploy	ElastiCache	IoT Things Graph	Redshift
AWS Chatbot	CodeGuru	Elastic Beanstalk	KMS	Rekognition
AWS Marketplace	CodeStar Notifications	Elastic Container Registry	Kinesis	RoboMaker
AWS Support	Comprehend	Elastic Container Service	Lake Formation	S3
Amplify	Config	Elastic Transcoder	Lambda	SMS
AppStream 2.0	Connect	ElasticLoadBalancing	Lex	SNS
AppSync	DMS	EventBridge	License Manager	SWF
Application Auto Scaling	Data Lifecycle Manager	Forecast	MQ	SageMaker
Application Discovery Service	Data Pipeline	GameLift	Machine Learning	Security Hub
Batch	DataBrew	Global Accelerator	Macie	Service Catalog
Braket	DataSync	Glue	Managed Blockchain	Step Functions
Budgets	DeepLens	Greengrass	MediaConvert	Storage Gateway
Certificate Manager	Directory Service	GuardDuty	Migration Hub	Systems Manager
Chime	DynamoDB	Health Organizational View	Network Firewall	Textract
CloudFormation	EC2	Honeycode	OpsWorks	Transfer
CloudHSM	EC2 - Fleet	IAM Access Analyzer	Personalize	Trusted Advisor
CloudTrail	EC2 Auto Scaling	Incident Manager	Purchase Orders	VPC
CloudWatch Alarms	EC2 Image Builder	Inspector	QLDB	WorkLink
CloudWatch Application Insights	EKS	IoT	RAM	WorkMail
CloudWatch Events	EMR			

사용 사례 선택

CodeDeploy
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

CodeDeploy - ECS
Allows CodeDeploy to read EC2 objects, launch Lambda functions, publish to SNS topics, and update ECS services on your behalf.

CodeDeploy 역할 생성

CodeDeploy는 권한이 하나뿐이라서 선택 없이 바로 다음으로 넘어가면 된다.

정책 필터 ▾ <input type="text" value="검색"/>		
정책 이름 ▾	사용 용도	설명
▶ AWSCodeDeployRole	없음	Provides CodeDep

권한

2) 태그 (키, 값) 입력 > 역할 이름 입력 > 최종 확인 후 생성

태그 추가(선택 사항)

IAM 태그는 사용자 역할에 추가할 수 있는 키-값 페어입니다. 태그는 이메일 주소와 같은 사용자 정보를 포함하거나 정책과 같은 내용일 수 대한 액세스를 구성, 추적 또는 제어할 수 있습니다. [자세히 알아보기](#)

키	값(선택 사항)
<input type="text" value="Name"/>	<input type="text" value="role-codedeploy"/>
<input type="text" value="새 키 추가"/>	<input type="text"/>

태그 추가

에 필요한 정보를 입력하고 이 역할을 검토하십시오

역할 이름*

영숫자 및 '+=, @-_' 문자를 사용합니다. 최대 64자입니다.

역할 설명

Allows CodeDeploy to call AWS services such as Auto Scaling o

최대 1000자입니다. 영숫자 및 '+=, @-_' 문자를 사용합니다.

신뢰할 수 있는 개체

AWS 서비스: codedeploy.amazonaws.com

정책

 [AWSCodeDeployRole](#) 

권한 경계

권한 경계가 설정되지 않았습니다

다음 태그가 제공됩니다

최종 확인

4. CodeDeploy 생성

CodeDeploy는 AWS의 배포 서비스이다. CodeDeploy는 대체제가 없기 때문에 대체로 해당 서비스를 사용한다. 오토 스케일링 그룹 배포, 블루 그린 배포, 롤링 배포, EC2 단독 배포 등 많은 기능을 지원한다.

1) AWS CodeDeploy 이동 > 애플리케이션 생성 > **애플리케이션 이름(spring-deploy-test)** 입력 > **컴퓨팅 플랫폼(EC2/온프레미스)** 선택

애플리케이션 구성

애플리케이션 이름
애플리케이션 이름을 입력합니다

100자 이내

컴퓨팅 플랫폼
컴퓨팅 플랫폼 선택

EC2/온프레미스

▼

취소

애플리케이션 생성

애플리케이션 생성

2) 배포 그룹 생성 > 배포 그룹 이름(**spring-deploy-test-group**) 입력 > 서비스 역할(**role-codedeploy**) 입력 > 현재 위치 선택

만약 본인이 배포할 서비스가 2대 이상이라면 블루/그린을 선택하면 된다.

배포 그룹 이름

배포 그룹 이름 입력

spring-deploy-test-group

100자 제한

서비스 역할

서비스 역할 입력

AWS CodeDeploy가 대상 인스턴스에 액세스하도록 허용하는 CodeDeploy 권한이 있는 서비스 역할을 입력합니다.

arn:aws:iam::910939935434:role/role-codedeploy

배포 유형

애플리케이션 배포 방법 선택

☒ 현재 위치

배포 그룹의 인스턴스를 최신 애플리케이션 개정으로 업데이트합니다. 배포 중에 각 인스턴스가 업데이트를 위해 잠시 오프라인 상태로 전환됩니다.

☐ 블루/그린

배포 그룹의 인스턴스를 새 인스턴스로 교체하고 최신 애플리케이션 개정을 해당 인스턴스에 배포합니다. 대체 환경의 인스턴스가 로드 밸런서에 등록된 후 원본 환경의 인스턴스는 등록 취소되고 종료할 수 있습니다.

배포 그룹 생성

3) 환경 구성 설정 > **Amazon EC2 인스턴스** 선택 > **키, 값** 입력

환경 구성

이 배포에 추가할 Amazon EC2 Auto Scaling 그룹, Amazon EC2 인스턴스 및 온프레미스 인스턴스의 조합 선택

☐ Amazon EC2 Auto Scaling 그룹

☒ Amazon EC2 인스턴스

1개의 일치하는 고유한 인스턴스. 자세한 내용을 보려면 여기를 클릭하십시오.

EC2 인스턴스의 태그 그룹을 세 개까지 이 배포 그룹에 추가할 수 있습니다.

단일 태그 그룹: 해당 태그 그룹이 식별한 인스턴스가 배포됩니다.

다중 태그 그룹: 모든 태그 그룹이 식별한 인스턴스만 배포됩니다.

태그 그룹 1

키

값 - 선택 사항

Name

spring-deploy-test

태그 제거

태그 추가

환경 구성 설정

4) 배포 설정 > **로드 밸런싱 활성화 체크 해제** > 배포 그룹 생성

배포 구성이란 한번 배포할 때 몇 대의 서버에 배포할지를 결정한다. 2대 이상이라면 1대씩 배포할지, 30% 혹은 50% 나눠서 배포할지 등등 여러 옵션을 선택하겠지만, 1대 서버다 보니 전체 배포하는 옵션으로 선택하면 된다.

배포 설정

배포 구성

기본 및 사용자 지정 배포 구성 목록에서 선택합니다. 배포 구성은 애플리케이션이 배포되는 속도와 배포 성공 또는 실패 조건을 결정하는 규칙 세트입니다.

CodeDeployDefault.AllAtOnce

▼

또는

배포 구성 만들기

로드 밸런서

배포 프로세스 중에 수신 트래픽을 관리할 로드 밸런서를 선택합니다. 로드 밸런서는 배포 중인 각 인스턴스에서 트래픽을 차단하고 배포 성공 후 인스턴스에 대한 트래픽을 다시 허용합니다.

☐ 로드 밸런싱 활성화

▶ 고급 - 선택 사항

취소

배포 그룹 생성

배포 설정

5. Travis-ci, S3, CodeDeploy 연동

배포 그룹 생성까지 완료되었다면 CodeDeploy 설정은 끝이다. 이제 Travis CI와 CodeDeploy를 연동해주면 된다. 먼저 S3를 넘겨줄 zip 파일을 저장할 디렉토리를 하나 생성해보자.

1) EC2 디렉토리 생성

EC2 서버에 접속해서 다음가 같이 디렉토리를 생성한다.

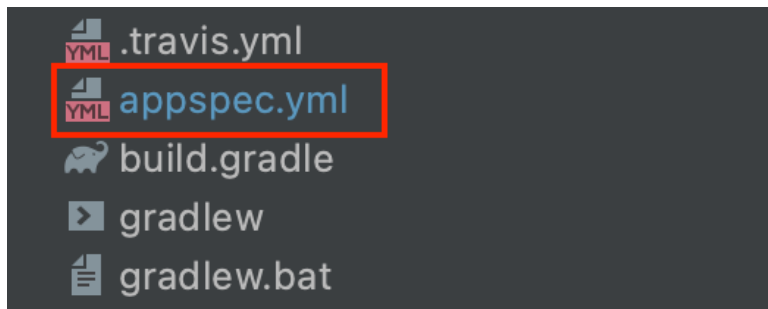
- `mkdir ~/app/step2 && mkdir ~/app/step2/zip`

```
[ec2-user@spring-deploy-test ~]$ mkdir ~/app/step2 && mkdir ~/app/step2/zip
```

디렉토리 생성

2. 스프링 프로젝트 appspec.yml 파일 추가

Travis CI의 Build가 끝나면 S3에 zip 파일이 전송되고, 이 zip파일은 /home/ec2-user/app/step2/zip로 복사되어 압축을 풀 예정이다. Travis CI의 설정은 .travis.yml로 진행했다면 AWS CodeDeploy 설정은 **appspec.yml**로 진행한다.



appspec.yml

```
version: 0.0
os: linux
files:
  - source: / # CodeDeploy에서 전달해 준 파일 중 destination으로 이동시킬 대상 지정 ('/' 루트 파일은 전체파일을 의미)
    destination: /home/ec2-user/app/step2/zip/ # source에서 지정된 파일을 받을 위치
    overwrite: yes # 기존 파일 덮어쓰기 허용
```

version: 0.0

- CodeDeploy 버전을 이야기한다.
- 프로젝트 버전이 양니므로 0.0외에 다른 버전을 사용하면 오류가 발생한다.

source

- CodeDeploy에서 전달해 준 파일 중 destination으로 이동시킬 대상을 지정한다.
- 루트 경로(/)를 지정하면 전체 파일을 이야기한다.

destination

- source에서 지정된 파일을 받을 위치이다.
- 이후 Jar를 실행하는 등은 destination에서 옮긴 파일들로 진행된다.

overwrite

- 기존에 파일들이 있으면 덮어쓸지를 결정한다.
- 현재 yes라고 했으니 파일들을 덮어쓰게 된다.

3. .travis.yml 파일에 CodeDeploy 설정 추가

.travis.yml에도 CodeDeploy 내용을 추가한다. deploy 항목에 다음 코드를 추가한다.

```
language: java
jdk:
  - openjdk11

branches:
  only:
    - main

# Travis CI 서버의 Home
cache:
  directories:
    - '$HOME/.m2/repository'
    - '$HOME/.gradle'
```

```
script: "./gradlew clean build"

# CI 실행 완료 시 메일로 알림
notifications:
  email:
    recipients:
      - jong9712@naver.com

before_deploy:
  - zip -r spring_deploy_test *
  - mkdir -p deploy
  - mv spring_deploy_test.zip deploy/spring_deploy_test.zip

## (S3로 파일업로드 | CodeDeploy 배포) 외부 서비스와 연동될 행위들을 선언
deploy:
  # S3로 파일업로드
  - provider: s3
    access_key_id: $AWS_ACCESS_KEY # Travis repo settings에 설정된 값
    secret_access_key: $AWS_SECRET_KEY # Travis repos settings에 설정된 값
    bucket: spring-deploy-test-build # s3 버킷
    region: ap-northeast-2
    skip_cleanup: true
    acl: private # zip 파일 접근을 private으로
    local_dir: deploy # before_deploy에서 생성한 디렉토리
    wait-until-deployed: true
    on:
      all_branches: true # master 말고 다른 모든 브랜치 허용
##### 새롭게 추가한 코드 #####

# CodeDeploy 배포
- provider: codedeploy
  access_key_id: $AWS_ACCESS_KEY # Travis repo settings에 설정된 값
  secret_access_key: $AWS_SECRET_KEY # Travis repos settings에 설정된 값
  bucket: spring-deploy-test-build # s3 버킷
  key: spring_deploy_test.zip # 빌드 파일 압축해서 전달
  bundle_type: zip # 압축 확장자
  application: spring-deploy-test # 웹 콘솔에서 등록한 CodeDeploy 애플리케이션 이름
  deployment_group: spring-deploy-test-group # 웹 콘솔에서 등록한 CodeDeploy 배포 그룹 이름
  region: ap-northeast-2
  wait-until-deployed: true
  on:
    all_branches: true # master 말고 다른 모든 브랜치 허용

#####
```

6. Git Push 테스트

모든 내용을 작성했다면 프로젝트를 commit하고 push한다. 깃헙으로 push가 되면 Travis CI가 자동으로 시작된다.

loosie / spring_deploy_test

build passing

Current

Branches

Build History

Pull Requests

More options

✓ main # travis-ci, s3, codedeploy 연동

~ #16 passed

Restart build

Commit 5e1b0cb

Ran for 1 min 50 sec

Compare 661c874..5e1b0cb

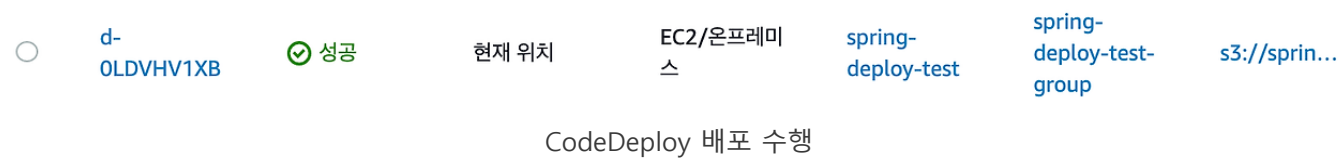
about a minute ago

Branch main

Travis CI 빌드 성공

AWS CodeDeploy 배포 내역 생성

Travis CI가 끝나면 CodeDeploy 화면 아래에서 배포가 수행되는 것을 확인할 수 있다.



~/app/step2/zip/에 프로젝트 파일 전달

배포가 끝났다면 다음 명령어로 파일들이 잘 도착했는지 확인해보자.

- `cd /home/ec2-user/app/step2/zip`
- `ll`

다음과 같이 프로젝트 파일이 압축해제되어 잘 도착한 것을 볼 수 있다.

```
[ec2-user@spring-deploy-test zip]$ ll
합 계 28
-rw-rw-r-- 1 root root 21  8월 27 01:16 README.md
-rw-rw-r-- 1 root root 521 8월 27 01:16 appspec.yml
drwxr-xr-x 9 root root 113 8월 27 01:18 build
-rw-rw-r-- 1 root root 1224 8월 27 01:16 build.gradle
drwxr-xr-x 3 root root 21  8월 27 01:18 gradle
-rwxrwxr-x 1 root root 5764 8월 27 01:16 gradlew
-rw-rw-r-- 1 root root 2942 8월 27 01:16 gradlew.bat
drwxr-xr-x 2 root root 23  8월 27 01:18 scripts
-rw-rw-r-- 1 root root 53  8월 27 01:16 settings.gradle
drwxr-xr-x 4 root root 30  8월 27 01:18 src
```

step2/zip

파일 및 인스턴스 정리

현재까지 AWS 환경과 EC2(Linux), Spring 환경에 생성된 인스턴스 및 파일 구조는 다음과 같다.

AWS

EC2 인스턴스

- spring-deploy-test

RDS 인스턴스

- spring-deploy-test

S3 버킷

- 사용자: spring-deploy-test-build

IAM

- 사용자: spring-travis-deploy (Travis-ci → S3, CodeDeploy 접근 권한)
- 역할: role-ec2-codedeploy (ec2 → CodeDeploy 접근 권한)
- 역할: role-codedeploy (CodeDeploy → ec2 접근 권한)

CodeDeploy 그룹

- Spring-deploy-test

Spring

Travis-ci 설정파일

- .travis.yml

Codedeploy 설정파일

- appspec.yml

EC2

/app

application-oauth.yml

application-db.yml

/step1

/spring_deploy_test

deploy.sh

/step2

/zip

스프링 프로젝트

step1: 수동배포

/step1/deploy.sh: 수동배포 스크립트

step2: 자동배포

/step2/zip: codedeploy에서 전송한 스프링 프로젝트

파일 및 인스턴스

Travis CI와 S3, CodeDeploy 연동이 완료되었다. 연동 테스트는 완료되었으니 이제는 실제 실행 파일들을 배포해보도록 하자. 설정만 바꾸면 되므로 다음 작업은 무척 간단하다.