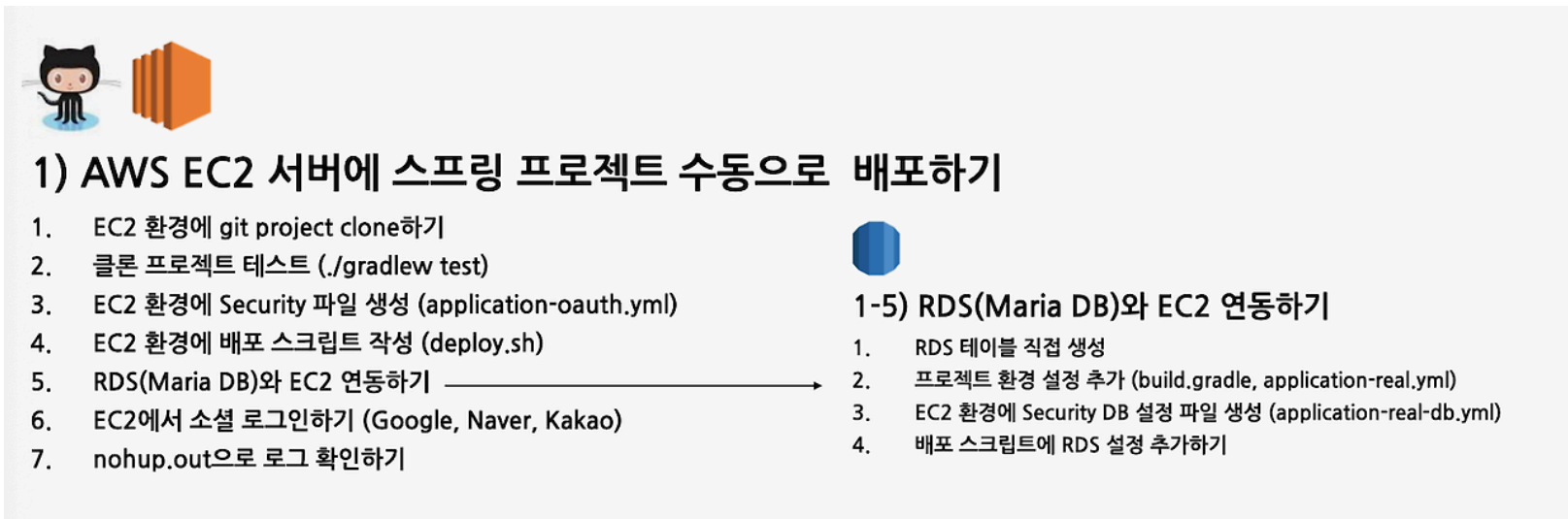


- AWS EC2 서버에 스프링 프로젝트 수동으로 배포하기
- 1. EC2 환경에 Git Project Clone하기
- 2. 클론 프로젝트 테스트하기
- 3. EC2 환경에 Security 파일 생성하기
- 4. EC2 환경에 배포 스크립트 작성하기
- 5. RDS와 EC2 연동하기
- 6. EC2에서 소셜 로그인하기
- 7. nohup.out으로 로그 확인하기
- 파일 및 인스턴스



└─AWS EC2 서버에 스프링 수동 배포하기

AWS EC2 서버에 스프링 프로젝트 수동으로 배포하기

스프링 프로젝트를 개발했고 EC2, RDS 등 배포 환경을 구성이 끝났으면 이젠 실제로 서비스를 한 번 배포할 차례이다. 배포 환경 구성은 다음과 같다.

- 스프링 프로젝트
- AWS EC2 (Linux)
- AWS RDS (MariaDB)

1. EC2 환경에 Git Project Clone하기

1) EC2에 git 설치

먼저 깃 허브에 코드를 받아 올 수 있게 EC2에 깃을 설치한다.

- `sudo yum install git`

```
[ec2-user@spring-deploy-test ~]$ sudo yum install git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                               | 3.7 kB      00:00
amzn2extra-docker                         | 3.0 kB      00:00
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.32.0-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.32.0-1.amzn2.0.1 for package: git-2
-1.amzn2.0.1.x86_64
Processing Dependency: perl-Git = 2.32.0-1.amzn2.0.1 for package: git-2
```

git 설치

2) git 설치 확인

설치가 완료되면 다음 명령어로 설치 상태를 확인한다.

- **git --version**

```
[ec2-user@spring-deploy-test ~]$ git --version
git version 2.32.0
```

git 버전확인

3) 프로젝트를 저장할 디렉토리 생성

깃이 성공적으로 설치되면 git clone으로 프로젝트를 저장할 디렉토리를 생성한다.

- **mkdir ~/app && mkdir ~/app/step1**

```
[ec2-user@spring-deploy-test ~]$ mkdir ~/app && mkdir ~/app/step1
[ec2-user@spring-deploy-test ~]$
```

디렉토리 생성

4) 디렉토리로 이동

생성된 디렉토리로 이동한다.

- **cd ~/app/step1**

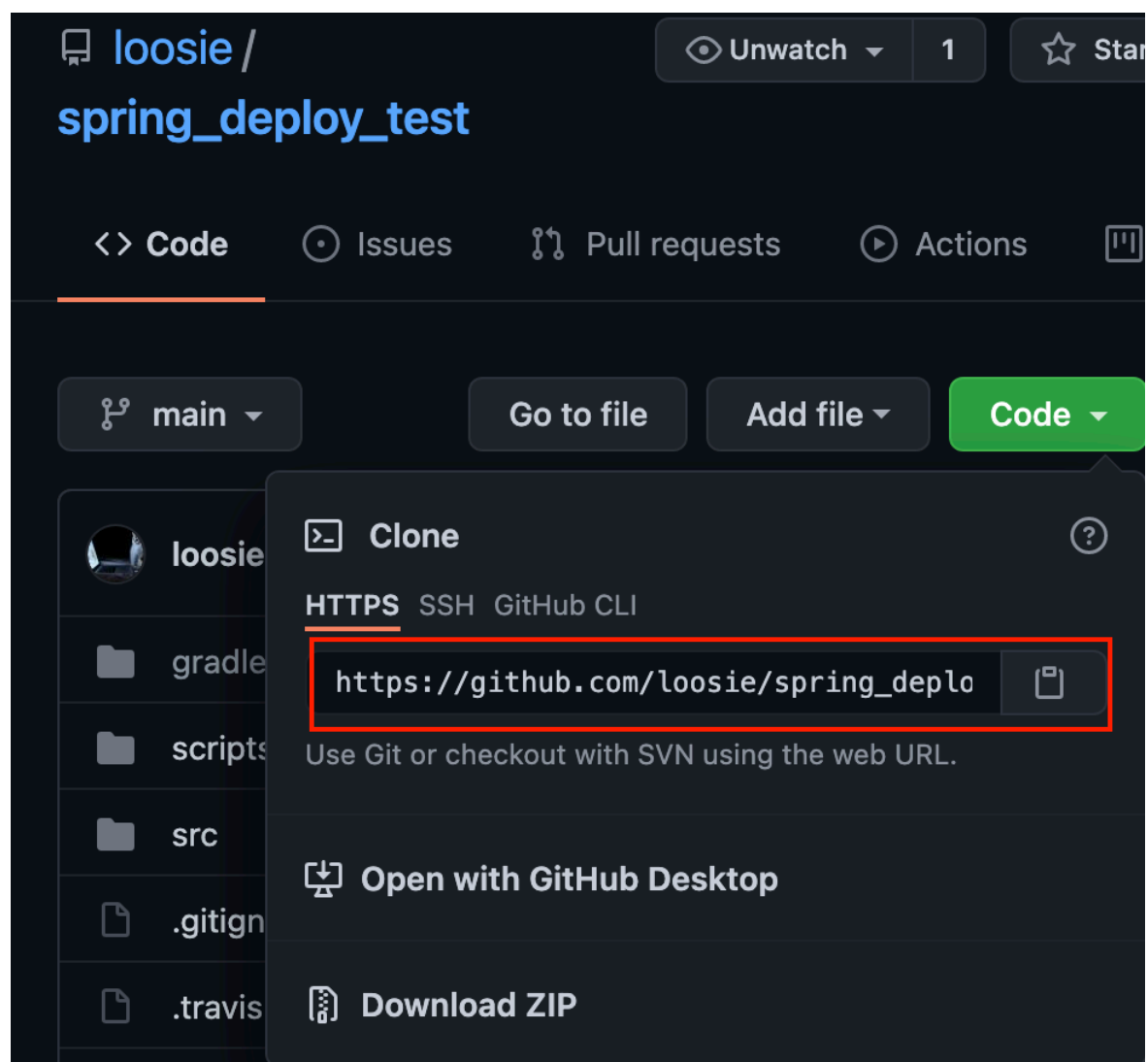
```
[ec2-user@spring-deploy-test ~]$ cd ~/app/step1
[ec2-user@spring-deploy-test step1]$
```

디렉토리 이동

5) git 프로젝트 주소 복사 > ec2에 git clone하기

본인 깃허브 웹페이지에서 https 주소를 복사한다.

- **git clone 복사한 주소**



git 프로젝트 주소 복사

```
[ec2-user@spring-deploy-test step1]$ git clone https://github.com/loosie/spring_deploy_test.git
'spring_deploy_test'에 복제합니다...
remote: Enumerating objects: 133, done.
remote: Counting objects: 100% (133/133), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 133 (delta 34), reused 115 (delta 18), pack-reused 0
오브젝트를 받는 중: 100% (133/133), 80.53 KiB | 13.42 MiB/s, 완료.
델타를 알아내는 중: 100% (34/34), 완료.
```

ec2에 git 프로젝트 clone

git clone이 끝났으면 잘 복사되었는지 확인해준다.

- **cd 프로젝트명**
- **ll**

```
[ec2-user@spring-deploy-test step1]$ cd spring_deploy_test
[ec2-user@spring-deploy-test spring_deploy_test]$ ll
합계 28
-rw-rw-r-- 1 ec2-user ec2-user 21 8월 26 00:15 README.md
-rw-rw-r-- 1 ec2-user ec2-user 509 8월 26 00:15 appspec.yml
-rw-rw-r-- 1 ec2-user ec2-user 1224 8월 26 00:15 build.gradle
drwxrwxr-x 3 ec2-user ec2-user 21 8월 26 00:15 gradle
-rwxrwxr-x 1 ec2-user ec2-user 5764 8월 26 00:15 gradlew
-rw-rw-r-- 1 ec2-user ec2-user 2942 8월 26 00:15 gradlew.bat
drwxrwxr-x 2 ec2-user ec2-user 23 8월 26 00:15 scripts
-rw-rw-r-- 1 ec2-user ec2-user 53 8월 26 00:15 settings.gradle
drwxrwxr-x 4 ec2-user ec2-user 30 8월 26 00:15 src
```

clone 확인

2. 클론 프로젝트 테스트하기

./gradlew test

테스트 코드를 정상적으로 설계하였다면 통과할 것이다.

```
[ec2-user@spring-deploy-test spring_deploy_test]$ ./gradlew test
Downloading https://services.gradle.org/distributions/gradle-6.1.1-all.zip
.....10%.....20%.....30%.....40%.....
%.....60%.....70%.....80%.....90%.....
00%
```

./gradlew test

```
Use --warning-mode all to show the individual deprecation warnings.
See https://docs.gradle.org/6.1.1/userguide/command_line_interface_and_line_warnings

BUILD SUCCESSFUL in 2m 50s
4 actionable tasks: 4 executed
```

테스트 성공

만약 테스트가 실패해서 수정하고 깃허브에 push를 했다면 프로젝트 폴더 안에서 다음 명령어를 사용하면 된다.

- **git pull**

만약 다음과 같이 gradlew 실행 권한이 없다는 메시지가 뜬다면 실행 권한 추가 후 다시 테스트를 수행하면 된다.

- **chmod +x ./gradlew**

```
-bash: ./gradlew: Permission denied
```

3. EC2 환경에 Security 파일 생성하기

배포를 하기 전 .gitignore로 git 제외 대상이라는 보안 파일들을 EC2서버에 생성해줘야 한다. 애플리케이션을 실행하기 위해 공개된 저장소에 ClientId와 ClientSecret을 올릴 수 없으니 서버에서 직접 이 설정들을 가지고 있게 해준다.

먼저 step1이 아닌 app 디렉토리에 yml 파일을 생성한다.

- **vim /home/ec2-user/app/application-oauth.yml**

```
[ec2-user@spring-deploy-test step1]$ vim /home/ec2-user/app/application-oauth.yml
```

파일 생성

그리고 로컬에 있는 application-oauth.yml 파일 내용을 그대로 붙여넣는다. 그리고 방금 생성한 파일은 deploy.sh라는 배포 스크립트에서 읽고 실행할 수 있도록 설정해주면 된다. 후에 RDS관련 Security파일은 5번 과정에서 추가로 진행된다.

4. EC2 환경에 배포 스크립트 작성하기

작성한 코드를 실제 서버에 반영하는 것을 배포라고 한다. 이 책에서 배포라 하면 다음의 과정을 모두 포괄하는 의미라고 보면 된다.

- git clone 혹은 git pull을 통해 새 버전의 프로젝트 받음
- Gradle이나 Maven을 통해 프로젝트 테스트와 빌드
- EC2 서버에서 해당 프로젝트 실행 및 재실행

앞선 과정을 배포할 때마다 개발자가 하나하나 명령어를 실행하는 것은 불편함이 많다. 그래서 이를 쉘 스크립트로 작성해 스크립트만 실행하면 앞의 과정이 차례로 진행되도록 할 것이다. 참고로 쉘 스크립트와 빔(vim)은 서로 다른 역할을 한다.

- 쉘 스크립트는 .sh라는 파일 확장자가 가진 파일이다. 노드 JS가 .js라는 파일을 통해 서버에서 작동하는 것처럼 쉘 스크립트 역시 리눅스에서 기본적으로 사용할 수 있는 스크립트 파일의 한종류이다.
- 빔은 리눅스 환경과 같이 GUI가 아닌 환경에서 사용할 수 있는 편집 도구이다.

1) deploy.sh 파일 생성

~/app/step1/에 deploy.sh 파일을 하나 생성한다.

- **vim ~/app/step1/deploy.sh**

```
[ec2-user@spring-deploy-test spring_deploy_test]$ vim ~/app/step1/deploy.sh
```

파일 생성

2) vim으로 쉘 스크립트 작성하기

- i : 입력
- :wq : 저장후 종료

```
#!/bin/bash

REPOSITORY=/home/ec2-user/app/step1
PROJECT_NAME=spring_deploy_test
EC2_HOST_NAME=spring-deploy-test

cd $REPOSITORY/$PROJECT_NAME/

echo "> Git Pull"

git pull

echo "> 프로젝트 Build 시작"

./gradlew build
```

```
echo "> step1 디렉토리로 이동"

cd $REPOSITORY

echo "> Build 파일 복사"

cp $REPOSITORY/$PROJECT_NAME/build/libs/*.jar $REPOSITORY/

echo "> 현재 구동중인 애플리케이션 pid 확인"

CURRENT_PID=$(pgrep -f ${EC2_HOST_NAME})

echo "현재 구동 중인 애플리케이션 pid : $CURRENT_PID"

if [ -z "$CURRENT_PID" ]; then
    echo "> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다."
else
    echo "> kill -15 $CURRENT_PID"
    sudo kill -15 $CURRENT_PID
    sleep 5
fi

echo "> 새 애플리케이션 배포"

JAR_NAME=$(ls -tr $REPOSITORY/ | grep jar | tail -n 1)

echo "> JAR Name: $JAR_NAME"

nohup java -jar W
    -Dspring.config.location=classpath:/application.yml,/home/ec2-user/app/applic
ation-oauth.yml W
    $REPOSITORY/$JAR_NAME 2>&1 &
```

스크립트 내용 설명

REPOSITORY=/home/ec2-user/app/step1

- 프로젝트 디렉토리 주소는 스크립트 내에서 자주 사용하는 값이기 때문에 이를 변수로 저장한다.
- 마찬가지로 PROJECT_NAME=spring_deploy_test도 동일하게 변수로 저장한다.
- 셸에서는 타입없이 선언하여 저장한다.
- 셸에서는 \$ 변수명으로 변수를 사용할 수 있다.

cd \$REPOSITORY/\$PROJECT_NAME/

- 제일 처음 git clone 받았던 디렉토리로 이동한다.
- 바로 위의 셸 변수 설명을 따라 /home/ec2-user/app/step1/spring_deploy_test 주소로 이동한다.

git pull

- 디렉토리 이동 후, master 브랜치의 최신 내용을 받는다.

./gradlew build

- 프로젝트 내부의 gradlew로 build를 수행한다.

```
cp ./build/libs/*.jar $REPOSITORY/
```

- build의 결과물인 jar 파일을 복사해 jar 파일을 모아둔 위치로 복사한다.

```
CURRENT_PID=$(pgrep -f spring_depoly_test)
```

- 기존에 수행 중이던 스프링 부트 애플리케이션을 종료한다.
- pgrep은 process id만 추출하는 명령어이다.
- -f 옵션은 프로세스 이름으로 찾는다.

```
if ~ else ~ fi
```

- 현재 구동 중인 프로세스가 있는지 없는지를 판단해서 기능을 수행한다.

```
JAR_NAME=$(ls -tr $REPOSITORY/ | grep jar | tail -n 1)
```

- 새로 실행할 jar 파일명을 찾는다.
- 여러 jar 파일이 생기기 때문에 tail -n로 가장 나중의 jar 파일(최신 파일)을 변수에 저장한다.

```
-Dspring.config.location
```

- 스프링 설정 파일 위치를 지정한다.
- 기본 옵션들을 담고 있는 application.yml과 OAuth 설정들을 담고 applcation-oauth.yml의 위치를 지정한다.
- application-oauth.yml은 절대경로를 사용한다. 외부에 파일이 있기 때문이다.

이렇게 생성한 스크립트에 실행 권한을 추가한 다음 실행해준다.

- **chmod +x ./deploy.sh**
- **./deploy.sh**

```
[ec2-user@spring-deploy-test step1]$ ./deploy.sh
> Git Pull
이미 업데이트 상태입니다 .
> 프로젝트 Build 시작

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 4s
5 actionable tasks: 1 executed, 4 up-to-date
> step1 디렉토리로 이동
> Build 파일 복사
> 현재 구동 중인 애플리케이션 pid 확인
현재 구동 중인 애플리케이션 pid :
> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다 .
> 새 애플리케이션 배포
> JAR Name: freelec-springboot2-webervice-1.0.1-SNAPSHOT.jar
[ec2-user@spring-deploy-test step1]$ nohup: appending output to `nohup.out'
```

스크립트 실행

5. RDS와 EC2 연동하기

RDS는 MariaDB를 사용 중이다. 이 MariaDB에서 스프링부트 프로젝트를 실행하기 위해선 몇 가지 작업이 필요하다. 진행할 작업은 다음과 같다.

- 테이블 생성 : H2에서 자동 생성해주던 테이블들을 MariaDB에선 직접 쿼리를 이용해 생성한다.
- 프로젝트 설정 : 자바 프로젝트가 MariaDB에 접근하려면 데이터베이스 드라이버가 필요하다. MariaDB에서 사용 가능한 드라이버를 프로젝트에 추가한다.
- EC2(Linux) 설정 : 데이터베이스의 접속 정보는 중요하게 보호해야 할 정보이다. 공개되면 외부에서 데이터를 모두 가져갈 수 있기 때문이다. 따라서 보안을 위해 EC2서버 내부에서 접속 정보를 관리하도록 설정한다. (application-real-db.yml)

1) RDS 테이블 직접 생성

JPA가 사용될 **엔티티 테이블**과 **스프링 세션이 사용될 테이블** 2가지 종류를 생성한다. JPA가 사용할 테이블은 **테스트 코드 수행 시 로그로 생성되는 쿼리**를 사용하면 된다.

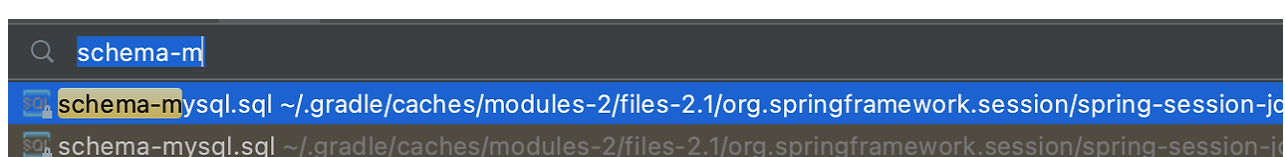
```
Hibernate: create table posts (id bigint not null auto_increment, created_date
    datetime, modified_date datetime, author varchar(255), content TEXT not
    null, title varchar(500) not null, primary key (id)) engine=InnoDB
Hibernate: create table user (id bigint not null auto_increment, created_date
    datetime, modified_date datetime, email varchar(255) not null, name
    varchar(255) not null, picture varchar(255), role varchar(255) not null,
    primary key (id)) engine=InnoDB
```

엔티티 테이블 생성 쿼리

```
create table posts (
    id bigint not null auto_increment,
    created_date datetime, modified_date datetime,
    author varchar(255), content TEXT not null,
    title varchar(500) not null, primary key (id)
) engine=InnoDB
```

```
create table user (
    id bigint not null auto_increment,
    created_date datetime,
    modified_date datetime,
    email varchar(255) not null,
    name varchar(255) not null,
    picture varchar(255),
    role varchar(255) not null,
    primary key (id)
) engine=InnoDB
```

스프링 세션 테이블은 **schema-mysql.sql** 파일에서 확인할 수 있다. 해당 파일은 'File 검색(cmd+shift+O, ctrl+shift_N)'으로 찾을 수 있다.

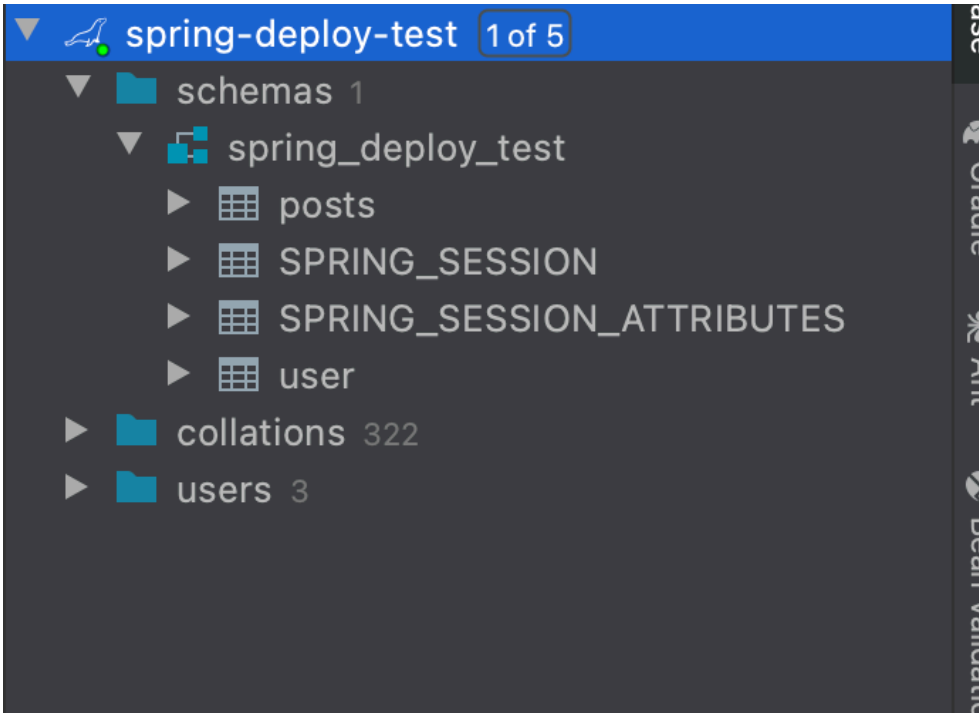



```
CREATE TABLE SPRING_SESSION (
    PRIMARY_ID CHAR(36) NOT NULL,
    SESSION_ID CHAR(36) NOT NULL,
    CREATION_TIME BIGINT NOT NULL,
    LAST_ACCESS_TIME BIGINT NOT NULL,
    MAX_INACTIVE_INTERVAL INT NOT NULL,
    EXPIRY_TIME BIGINT NOT NULL,
    PRINCIPAL_NAME VARCHAR(100),
    CONSTRAINT SPRING_SESSION_PK PRIMARY KEY (PRIMARY_ID)
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;

CREATE UNIQUE INDEX SPRING_SESSION_IX1 ON SPRING_SESSION (SESSION_ID);
CREATE INDEX SPRING_SESSION_IX2 ON SPRING_SESSION (EXPIRY_TIME);
CREATE INDEX SPRING_SESSION_IX3 ON SPRING_SESSION (PRINCIPAL_NAME);

CREATE TABLE SPRING_SESSION_ATTRIBUTES (
    SESSION_PRIMARY_ID CHAR(36) NOT NULL,
    ATTRIBUTE_NAME VARCHAR(200) NOT NULL,
    ATTRIBUTE_BYTES BLOB NOT NULL,
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_PK PRIMARY KEY (SESSION_PRIMARY_ID, ATTRIBUTE_NAME),
    CONSTRAINT SPRING_SESSION_ATTRIBUTES_FK FOREIGN KEY (SESSION_PRIMARY_ID) REFERENCES SPRING_SESSION(PRIMARY_ID) ON DELETE CASCADE
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;
```

IntelliJ 오른쪽 탭 > Database에서 RDS와 연결한 다음 쿼리문을 열어 해당 쿼리들을 실행해주면 간단히 생성된다.



테이블 생성 완료

2) 프로젝트 환경 설정 추가

먼저 MariaDB 드라이버를 build.gradle에 등록한다.

build.gradle

```
compile('org.mariadb.jdbc:mariadb-java-client')
```

그리고 서버에서 구동될 환경을 하나 구성한다. src/main/resources/에 application-real.yml파일을 추가한다. application-real.yml로 파일을 만들면 profile=real인 환경이 구성된다고 보면 된다. 실제 운영될 환경이기 때문에 보안/로그상 이슈가 될 만한 설정들을 모두 제거하며 RDS 환경 profile 설정이 추가된다.

application-real.yml

```
spring:
  profiles:
    include: oauth, real-db
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
  session:
    store-type: jdbc
```

3) EC2 환경에 Security DB 설정 파일 생성

OAuth와 마찬가지로 RDS 접속 정보도 보호해야 할 정보이니 EC2 서버에 직접 설정 파일을 둔다. app 디렉토리에 application-real-db.yml 파일을 생성한다.

- **vim ~/app/application-real-db.yml**

```
spring:
  datasource:
    url: jdbc:mariadb://rds주소:포트명(기본은 3306)/database 이름
    driverClassName: org.mariadb.jdbc.Driver
    username: db계정
    password: db계정 비밀번호
  jpa:
    hibernate:
      ddl-auto: none
```

4) 배포 스크립트에 RDS 설정 추가하기

```
...

nohup java -jar W
-Dspring.config.location=classpath:/application.yml,/home/ec2-user/app/application-oauth.yml,/home/ec2-user/app/applic
ation-real-db.yml,classpath:/application-real.yml W
-Dspring.profiles.active=real W
$REPOSITORY/$JAR_NAME 2>&1 &
```

/app/step1/nohup.out 파일을 열어 맨 밑에 다음과 같은 로그가 보인다면 성공적으로 수행된 것이다.

```
g
2021-08-26 20:52:11.437 INFO 3861 --- [          main] o.s.b.w.embedded.tomcat.TomcatW
ebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-08-26 20:52:11.442 INFO 3861 --- [          main] com.loosie.book.springboot.Appl
ication : Started Application in 13.204 seconds (JVM running for 14.37)
~
```

nohup.out 로그 확인

추가로 curl 명령어로 html 코드가 정상적으로 보이는 것을 확인해주면 된다.

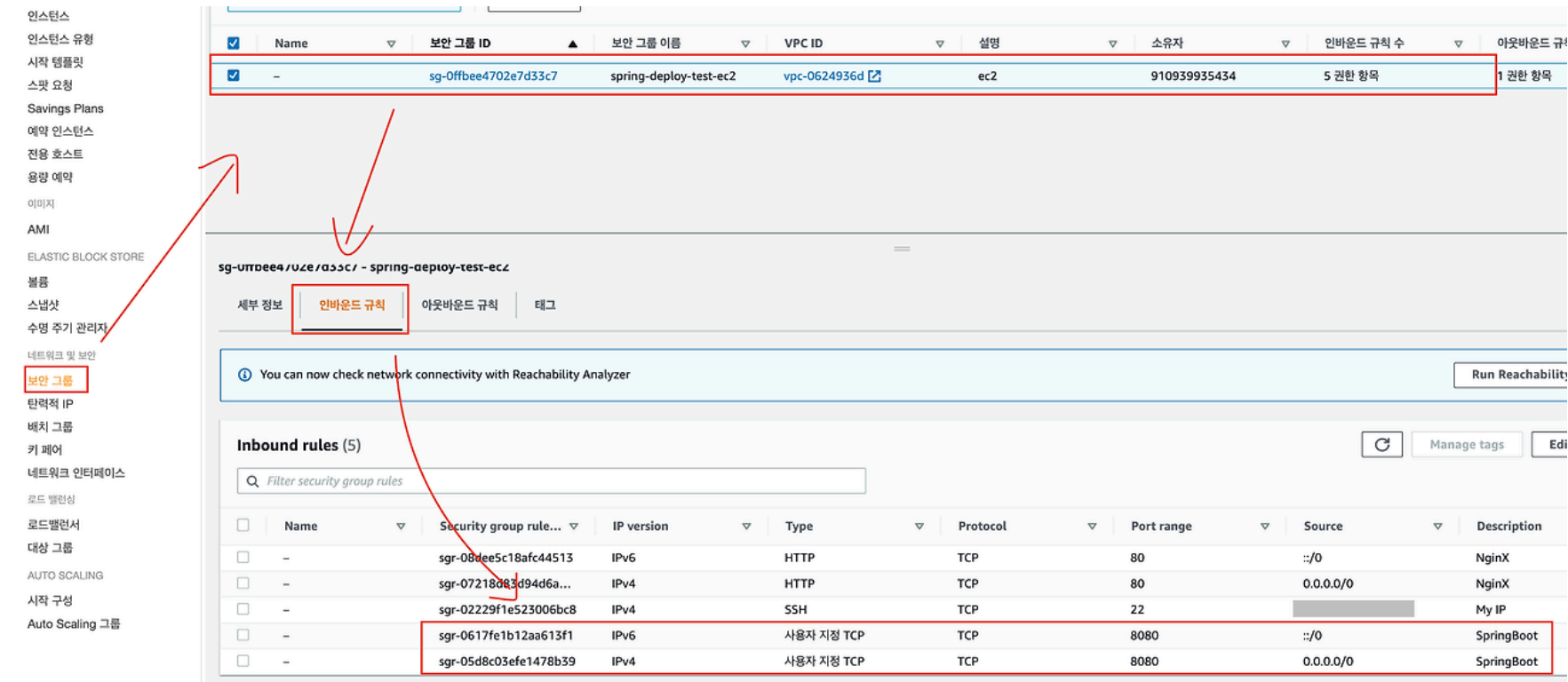
- curl localhost:8080

6. EC2에서 소셜 로그인하기

curl 명령어를 통해 EC2에 서비스가 잘 배포된 것은 확인하였다. 이제 브라우저에서 확인해 볼 차례이다. 그 전에 다음과 같은 몇 가지 작업을 해야한다.

1) AWS 보안 그룹 변경

먼저 EC2에 스프링 부트 프로젝트가 8080 포트로 배포되었으니, 8080 포트가 보안 그룹에 열려 있는지 확인한다.

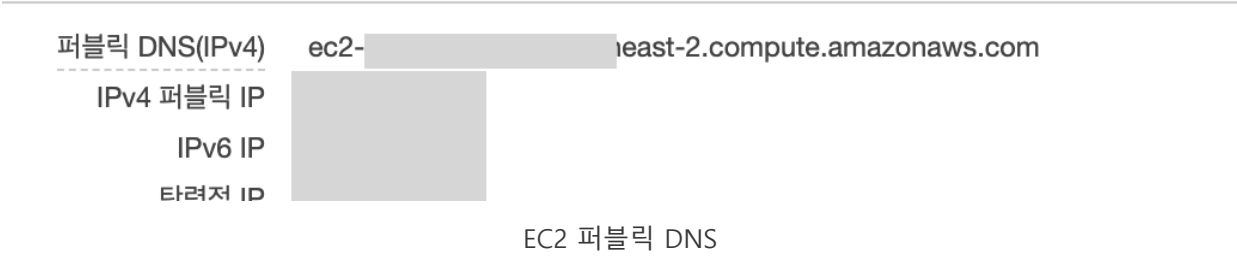


EC2 보안그룹 확인하기

8080이 열려있다면 OK, 안되어있다면 [편집] 버튼을 눌러 추가해 준다.

2) AWS EC2 도메인으로 접속

왼쪽 사이드바의 [인스턴스] 메뉴를 클릭한다. 본인이 생성한 EC2 인스턴스를 선택하면 다음과 같이 상세 정보에서 퍼블릭 DNS를 확인할 수 있다.



이 주소가 EC2에 자동으로 할당된 **도메인**이다. 인터넷이 되는 장소 어디나 이 주소를 입력하면 우리의 EC2서버에 접속할 수 있다. 그럼 이제 해당 도메인 주소에 8080 포트를 붙여 브라우저에 입력하면 다음과 같이 잘 접속되는 것을 확인할 수 있다.

스프링 부트로 시작하는 웹 서비스 ver.1.0.0

[글 등록](#)[Google Login](#)[Naver Login](#)[Kakao Login](#)

게시글번호	제목	작성자	내용	최종수정일
-------	----	-----	----	-------

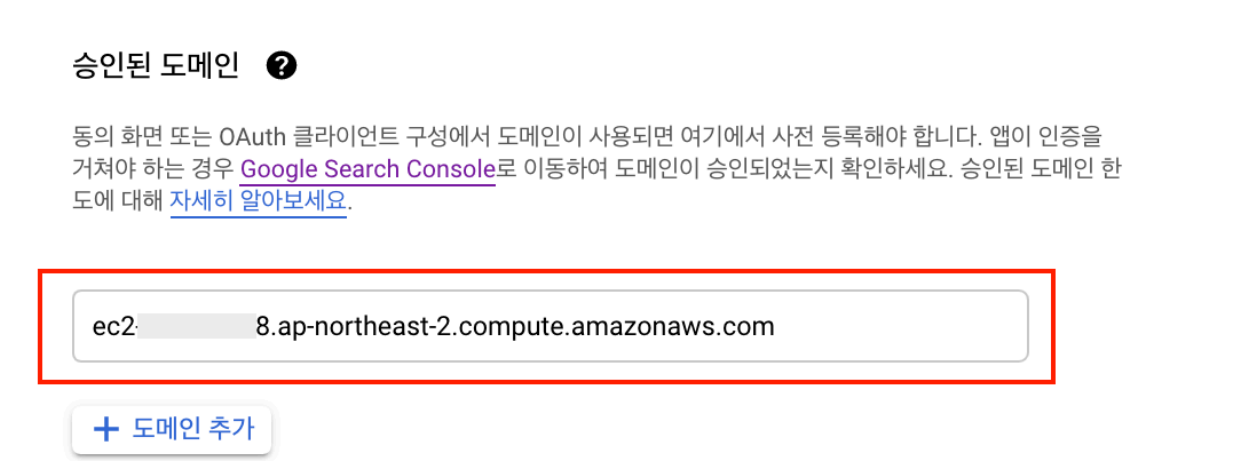
배포 성공

3) 소셜 로그인에 EC2 도메인 등록하기

이제 마지막으로 프로젝트에서 사용했던 소셜 로그인 기능을 해결해주면 된다. 각 서비스에 접속하여 EC2 도메인을 등록해주면 된다.

1. 구글

1) <https://console.cloud.google.com/> 접속 > API 탐색 및 사용 설정 > OAuth 동의 화면 > 앱 수정 > 승인된 도메인 **EC2 도메인** 입력 (http제외)



승인된 도메인 입력

2) 사용자 인증 정보 > OAuth 2.0 클라이언트 ID 해당 프로젝트 클릭 > 승인된 리디렉션 URI에 **EC2 도메인:8080/login/oauth2/code/google** 등록

승인된 리디렉션 URI

웹 서버의 요청에 사용

URI *

http://localhost:8080/login/oauth2/code/google

http://ec2-ap-northeast-2.compute.amazonaws.com:8080/login/o

승인된 리디렉션 URI 입력

3) 구글 로그인 성공

스프링 부트로 시작하는 웹 서비스 ver.1.0.0

글 등록

Logged in as: JW

Logout

게시글번호

제목

작성자

내용


구글 로그인 성공

2. 네이버

1) <https://developers.naver.com/main/> 접속 > 내 애플리케이션 > API 설정 > PC 웹 > 서비스 URL에 **EC2 도메인** 입력 & Callback URL (**EC2 도메인:8080/login/oauth2/code/naver**)입력

PC 웹

서비스 URL


http://ec2-3.ap-northeast-2.compute.amazonaws.com/

서비스 URL예시: (O) http://naver.com (X) http://www.naver.com
서비스 URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.
서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 아이디로 로그인** 배지가 노출됩니다.

네이버아이디로그인
Callback URL (최대 5개)

http://localhost:8080/login/oauth2/code/naver

—

http://ec28.ap-northeast-2.compute.amazonaws.com:8080,

+

텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.
Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

네이버 URL 입력

네이버는 아직 지원되지 않아 서비스 URL은 하나만 입력이 가능하다. 즉, EC2 주소를 입력하면 localhost는 적용이 안된다. 그러므로 개발 단계에서는 등록하지 않는 것을 추천한다. 만약 localhost도 동시에 테스트하고 싶다면 네이버 서비스를 하나 더 생성해서 키를 발급받으면 된다.

2) 네이버 로그인 성공

스프링 부트로 시작하는 웹 서비스 ver.1.0.0

글 등록

Logged in as:

Logout

게시글번호	제목	작성자	내용
-------	----	-----	----

네이버 로그인 성공

3. 카카오

1) <https://developers.kakao.com/> 접속 > 내 애플리케이션 선택 > 플랫폼 선택 > Web 사이트 도메인 수정 버튼 > **EC2 도메인** 추가

Web

삭제

사이트 도메인	<div><div>http://localhost:8080</div><div>http://ec2-<div></div>.ap-northeast-2.compute.amazonaws.com:8080</div></div>
---------	--

· 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

사이트 도메인 추가

2) 카카오 로그인 > Redirect URL에 **EC2 도메인:8080/login/oauth2/code/kakao** 추가

Redirect URI

삭제

수정

Redirect URI	<div><div>http://localhost:8080/login/oauth2/code/kakao</div><div>http://ec2-<div></div>.ap-northeast-2.compute.amazonaws.com:8080/login/oauth2/code/kakao</div></div>
--------------	--

· 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)

· REST API로 개발하는 경우 필수로 설정해야 합니다.

Redirect URL 추가

3) 카카오 로그인 성공

카카오 로그인 성공

7. nohup.out으로 로그 확인하기

nohup은 no hang up의 약자이다. 해석 그대로 끊지말라는 뜻이다. 세션과 연결을 종료해도 지금 실행시킨 프로그램을 종료하지 말라는 뜻이다. 따라서 중단없이 실행하고자 하는 프로그램 명령어 앞에 nohup만 붙여주면 된다.

- **nohup java -jar ₩ ...**
- &를 프로그램 끝에 붙여주면 해당 프로그램이 백그라운드로 실행된다.
 - nohup은 프로그램을 데몬의 형태로 실행시키는 것이기 때문에 로그아웃으로 세션이 종료되어도 프로그램이 종료되지 않는다. 그러나 &(백그라운드) 실행은 단지 프로그램을 사용자 눈에 보이지 않는 형태로만 돌리는 것이기 때문에 세션과 연결이 끊어지면 프로그램도 함께 종료된다.

이렇게 실행시키면 "nohup: appending output to `nohup.out` " 메시지와 함께 해당 프로그램의 표준출력이, nohup 을 실행시킨 경로에 nohup.out으로 출력된다.

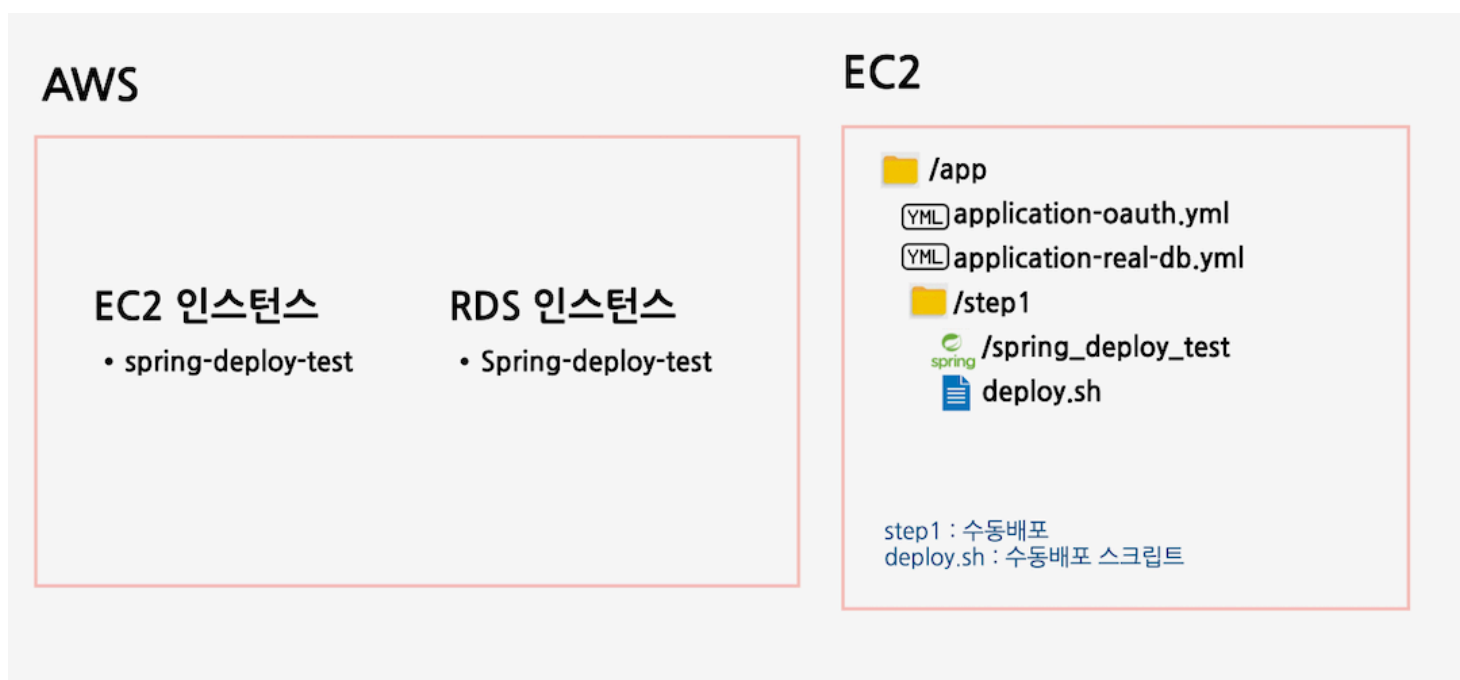
./deploy.sh에 가 실행된 공간에 다음과 같은 nohup.out의 파일도 생성되는 것을 볼 수 있다. 해당 프로그램이 잘 돌아갔는지 확인하려면 **vim nohup.out** 의 명령어를 통해 해당 로그를 확인할 수 있다.

```
bash: !-: command not found
[ec2-user@spring-deploy-test step1]$ ls
deploy.sh  freelec-springboot2-webservice-1.0.0-SNAPSHOT.jar  nohup.out  spring_deploy_test
```

step1 디렉토리

파일 및 인스턴스

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



인스턴스 및 파일 구조

수동 배포 작업이 끝났으니 이젠 Travis CI, S3, CodeDeploy를 사용하여 배포 자동화를 진행해보자.