FirmBanking API Run & Deploy

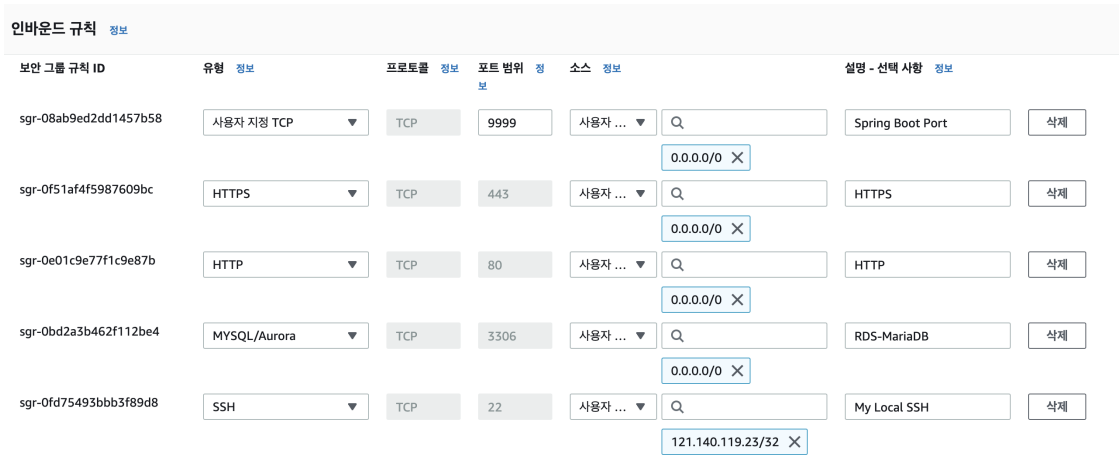
📝작성자 : 김상윤, 최지희

JAVA version : 11

Gradle based Spring Boot Project

🔶 배포 방법 (AWS)

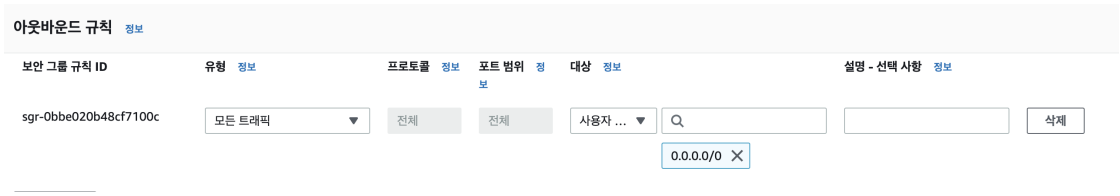
- **보안 그룹 :**
  - **인바운드 규칙**



인바운드 규칙 정보

보안 그룹 규칙 ID	유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명 - 선택 사항 정보	
sgr-08ab9ed2dd1457b58	사용자 지정 TCP	TCP	9999	사용자 ... 0.0.0.0/0	Spring Boot Port	삭제
sgr-0f51af4f5987609bc	HTTPS	TCP	443	사용자 ... 0.0.0.0/0	HTTPS	삭제
sgr-0e01c9e77f1c9e87b	HTTP	TCP	80	사용자 ... 0.0.0.0/0	HTTP	삭제
sgr-0bd2a3b462f112be4	MYSQL/Aurora	TCP	3306	사용자 ... 0.0.0.0/0	RDS-MariaDB	삭제
sgr-0fd75493bbb3f89d8	SSH	TCP	22	사용자 ... 121.140.119.23/32	My Local SSH	삭제

  - **아웃바운드 규칙 :**



아웃바운드 규칙 정보

보안 그룹 규칙 ID	유형 정보	프로토콜 정보	포트 범위 정보	대상 정보	설명 - 선택 사항 정보	
sgr-0bbe020b48cf7100c	모든 트래픽	전체	전체	사용자 ... 0.0.0.0/0		삭제

- **EC2 배포 :**
  - AWS EC2 Spec :
    - 프리티어
    - OS : Ubuntu 20.04
  - deploy :
    1. 생성한 EC2 인스턴스 연결
    2. git 설치

```
sudo yum install -y git
git --version
```

3. java 설치

```
sudo yum -y install java-11-openjdk java-11-openjdk-devel
java --version
```

4. git clone

```
git clone {원격 저장소 주소}
```

5. 실행

```
cd {서비스 폴더명}
sudo chmod +x ./gradlew
./gradlew bootRun
```

- **DataBase(AWS RDS사용)**
  - **AWS RDS Spec :**
    - 프리티어
    - Maria DB version : 10.6.7
    - 포트 : 3306
    - 엔진 유형 : Maria DB
    - 인스턴스 클래스 : db.t3.micro
    - 스토리지 유형 : 범용 SSD(gp2)
    - 퍼블릭 액세스 가능 : Y
  - **TABLE 생성 쿼리문 :**
    1. **CustMst**

```
CREATE TABLE `CustMst` (
  `CustId` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `CustNm` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_general_ci',
  `OrgCd` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `CallbackURL` VARCHAR(200) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `ApiKey` VARCHAR(64) NOT NULL COLLATE 'utf8mb4_general_ci',
  `PriContactNm` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `PriContactTel` VARCHAR(20) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `PriContactEmail` VARCHAR(100) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `SecContactNm` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `SecContactTel` VARCHAR(20) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `SecContactEmail` VARCHAR(100) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `TxSequence` VARCHAR(200) NOT NULL COLLATE 'utf8mb4_general_ci',
  `InUse` CHAR(1) NOT NULL COLLATE 'utf8mb4_general_ci',
  `CreatedAt` DATETIME NULL DEFAULT NULL,
  `UpdatedAt` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`CustId`) USING BTREE
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB;
```

    2. **BankMst**

```
CREATE TABLE `BankMst` (
  `BankId` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `BankCd` VARCHAR(3) NOT NULL COLLATE 'utf8mb4_general_ci',
  `BankNm` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_general_ci',
  `SwiftCd` VARCHAR(11) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `CreatedAt` DATETIME NULL DEFAULT NULL,
  `UpdatedAt` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`BankId`) USING BTREE
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB;
```

3. **TxLog**

```
CREATE TABLE `TxLog` (
  `TxIdx` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_general_ci',
  `CustId` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxDate` DATE NOT NULL,
  `TelegramNo` VARCHAR(6) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `TxType` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `BankCd` VARCHAR(3) NOT NULL COLLATE 'utf8mb4_general_ci',
  `Size` MEDIUMINT(8) NULL DEFAULT NULL,
  `RoundTrip` DECIMAL(5,3) NULL DEFAULT NULL,
  `StmtCnt` SMALLINT(6) NULL DEFAULT NULL,
  `Status` VARCHAR(3) NOT NULL COLLATE 'utf8mb4_general_ci',
  `StartDT` DATETIME NULL DEFAULT NULL,
  `EndDT` DATETIME NULL DEFAULT NULL,
  `EncData` MEDIUMBLOB NULL DEFAULT NULL,
  `NatvTrNo` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `ErrCode` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `ErrMsg` VARCHAR(200) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `MsgId` VARCHAR(40) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `OrgCd` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`TxIdx`) USING BTREE
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB;
```

4. **TxStat**

```
CREATE TABLE `TxStat` (
  `CustId` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxDate` DATE NOT NULL,
  `BankCd` VARCHAR(3) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxType` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxCnt` MEDIUMINT(8) NOT NULL,
  `TxSize` INT(11) NOT NULL,
  `OrgCd` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`CustId`, `TxDate`, `BankCd`, `TxType`) USING BTREE
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB;
```

5. **TxTrace**

```
CREATE TABLE `TxTrace` (
  `CustId` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxDate` DATE NOT NULL,
  `TxSequence` VARCHAR(200) NOT NULL COLLATE 'utf8mb4_general_ci',
  `TxStarted` CHAR(1) NOT NULL COLLATE 'utf8mb4_general_ci',
  `OrgCd` VARCHAR(10) NOT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`TxDate`, `CustId`) USING BTREE
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB;
```

- **데이터베이스 설정 후 Spring 연동**

  config/application.yaml파일에 데이터베이스 설정 내용 작성

```yaml
datasource:
  url: jdbc:mariadb://firmbankingapi.cwtbzbyluijp.ap-northeast-2.rds.amazonaws.com/firmbanking
  driver-class-name: org.mariadb.jdbc.Driver
  username: inspien_test
  password: 12345678
```

  위와 같이 application.yaml파일 내 Spring/datasouce에 연동 할 database의 host명(IP), username, password 작성

🔸 실행 방법

- **Git Clone**

```
mkdir {디렉토리 명}
cd {디렉토리 명}
git clone https://github.com/inspien-cloud/FirmBankingAPI.git
```

- **서비스 실행 명세**
  1. Eureka Server 실행 (port번호 : 6901)
  2. API Gateway 실행 (port번호 : 8080)
  3. Transfer, BankStatement, TransferCheck, crud 서비스 모두 실행
     (각각 port번호 : 포트명세 참조)
  4. Eureka 웹 콘솔에 접속하여, 정상적으로 모든 서비스들이 동작되는지 확인 (http://localhost:6901)
  5. API-Gateway를 통하여 **API 명세**에 기재된 API호출
     a. 만일 호출 시, **PKIX path building failed Error 발생하였다면, 하단의 인증서등록 참조**
  6. 리액트 웹 콘솔 실행 (🔸 **Admin 페이지 구동** 참조)
  7. Prometheus & Grafana 실행 (🔸 **Prometheus / Grafana 구동** 참조)

- **인증서 등록**
  1. **Installcert.java 설치 후 실행**

```
curl -O https://gist.githubusercontent.com/lesstif/cd26f57b7cfd2cd55241b20e05b5cd93/raw/InstallCert.java
javac InstallCert.java

java InstallCert '[연결하고자 하는 주소]'
=> ex) java InstallCert 127.0.0.1:9000
```

  2. **'jssecacerts' 파일 java 인증서로 변환**

```
Added certificate to keystore 'jssecacerts' using alias '127.0.0.1-1'
```

설치 후 위와 같은 문구가 출력됨과 동시에 ‘jssecacerts’ 파일이 추가됨. (이때 alias는 127.0.0.1-1이다.) 해당 파일을 자바 인증서로 변환시키기 위해 아래 코드 실행

```
## alias 옵션뒤에 위의 alias명 입력
keytool -exportcert -keystore jssecacerts -storepass changeit -file output.cert -alias '[위에서 설정한 alias]'
=> ex) keytool -exportcert -keystore jssecacerts -storepass changeit -file output.cert -alias 127.0.0.1-1
```

위 과정이 정상적으로 진행되었다면 아래와 같은 문구와 함께 output.cert라는 java 인증서 추출 완료

3. **java 인증서 JVM에 등록**

   접속하려는 서버의 JVM 경로를 확인 후 해당 경로에 java 인증서 등록을 위해 아래 코드 실행

```
sudo  keytool -importcert -keystore '{cacerts의 경로}' -storepass changeit -file output.cert -alias '{alias}'
=> ex) sudo keytool -importcert
-keystore /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/lib/security/cacerts
-storepass changeit -file output.cert -alias transfer
```

4. **인증서 등록 확인**

   인증서가 제대로 등록되었는지 확인하기 위해 위에서 실행한 InstallCert 명령어로 다시 연결시도

```
java InstallCert '[연결하고자 하는 주소]'
=> ex) java InstallCert 127.0.0.1:9000
```

   이때 에러 없이 잘 동작할 시, 정상적으로 인증서 설치가 완료된것.

🔸 PORT 명세 (default : localhost)

서비스명	Port
Eureka Server	6901
API Gateway	8080
FirmBankingAPI_transfer	9000
FirmBankingAPI_Bankstatement	9002
FirmBankingAPI_TransferCheck	9004
FirmBankingAPI_crud	9006
FirmBankingAPI_admin	4000
prometheus	9090
grafana	3000

🔸 API 명세

모든 API 호출은 API-Gateway를 통하는 것이 원칙

(ex. 이체의 경우 : https://localhost:8080/api/rt/v1/transfer post 요청)

API 명	method	URI
지급이체(송금)	POST	api/rt/v1/transfer
이체처리결과 조회	POST	api/rt/v1/transfer/check
거래명세	POST	api/rt/v1/bankstatement

🔸 Request Body 명세

- **지급이체(송금)**

```
{
    "api_key" : "7242191d-865c-48df-aa02-e3cf10bffd6d",
    "org_code" : "10000262",
    "drw_bank_code" : "088",
    "drw_account" : "832210312031",
    "drw_account_cntn" : "문세인",
    "rv_bank_code" : "081",
    "rv_account" : "46291012501007",
    "rv_account_cntn" : "핍랄샐",
    "amount" : 1000,
    "tr_dt" : "20220817",
    "tr_tm" : "155011",
    "msg_id" : "testRequest"
}
```

- **이체처리결과 조회**

```
{
    "api_key" : "7242191d-865c-48df-aa02-e3cf10bffd6d",
    "org_code" : "10000262",
    "tr_dt" : "20220817",
    "drw_bank_code" : "088",
    "msg_id" : "testRequest"
}
```

- **거래명세**

```
{
"org_code":"10000262","bank_code":"088",
"natv_tr_no":"090001","send_dt":"20220817","send_tm":"133243",
"account":"52212601","scf_dt":"20201222","scf_tm":"150550",
"rnd_gb_code":"21","scf_code":"30","amount":"5000","obc_amount":"0",
"after_sign":"-","after_amount":"10000","cms_code":"","can_tr_no":"000000","can_ogn_dt":"00000000",
"account_cntn":"XXXXXX","mid":null,
"cpt_rtu_org_code":"100000XX","cpt_rtu_ogn_tr_dt":"20220124",
"cpt_rtu_ogn_tr_no:":"000024","cpt_rtu_ogn_orr_no":"ORRNO20220124142358912582",
"rv_nm":"XXXXX","cpt_rtu_yn":"Y","cpt_rq_dt":"20201223","cpt_rq_no": "FR5829495665"
}
```

🔸 Response 명세

- **지급이체(송금)**

Status	Response Body

Status	Response Body
200(OK)	{"status":200,"natv_tr_no":"20220817002089000025","request_at":"20220817134022","amount":1000}

- **이체처리결과 조회**

Status	Response Body
200(OK)	{"status":200,"natv_tr_no":"20220817002089000025","transfer_at":"20220817134018","amount":1000}

- **거래명세**

Status	Response Body
200(OK)	{"status":200,"request_at":"202208174220"}

🔸 Admin 페이지 구동

▪Installation

```
cd FirmBankingAPI_admin
npm install
```

▪Run

```
npm start
```

위 코드 실행 후 http://localhost:4000에 접속하여 확인

🔸 Prometheus / Grafana 구동

▪Prometheus

- **Run**

  ```
  cd Prometheus
  prometheus --config.file=prometheus.yml->
  ```

  위 코드 실행 후 http://localhost:9090 에 접속하여 확인

▪Grafana

- **Run**

  ```
  cd Grafana
  ./bin/grafana-server web
  ```

  위 코드 실행 후 http://localhost:3000에 접속하여 확인