프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서

외부 서비스 정보 정리 문서

본 문서는 프로젝트에서 사용된 외부 서비스 정보를 정리한 문서입니다. 코드 컴파일 및 서비스 운영 시 필요한 외부 서비스 가입 및 활용 정보를 제공합니다.

사용된 외부 서비스

1. AWS DynamoDB

- **설명**: NoSQL 데이터베이스 서비스로, 빠른 읽기/쓰기 성능을 제공하며 확장성이 뛰어남.
- **활용 목적**: 프로젝트에서 `curriculumId`를 파티션 키로, `time`을 정렬 키로 사용하여 데이터를 저장하고 조회하는 용도로 사용됨.
- **스프링 설정**:

```
cloud:
  aws:
    dynamodb:
      region: ${AWS_REGION}
      endpoint: http://localhost:8000
```

- **Java 설정**:

```
@Configuration
public class DynamoDBConfig {
    @Value("${AWS_ACCESS_KEY}")
    private String accessKey;

    @Value("${AWS_SECRET_KEY}")
    private String secretKey;

    @Value("${AWS_REGION}")
    private String region;
```

```
    @Bean
    public DynamoDbClient dynamoDbClient(){
        return DynamoDbClient.builder()
            .region(Region.of(region))
            .credentialsProvider(StaticCredentialsProvider.create(AwsBasic
Credentials.create(accessKey, secretKey)))
            .build();
    }
}
```

2. AWS S3

- **설명**: 객체 스토리지 서비스로, 다양한 데이터 유형을 저장하고 관리할 수 있음.
- **활용 목적**: 프로젝트에서 파일 및 데이터 백업을 위한 저장소로 활용
- **스프링 설정**:

```
cloud:
  aws:
    credentials:
      access-key: ${AWS_ACCESS_KEY}
      secret-key: ${AWS_SECRET_KEY}
    region:
      static: ${AWS_REGION}
    s3:
      bucket: ${AWS_BUCKET_NAME}
```

- **Java 설정**:

```
@Configuration
public class S3Config {
    @Value("${cloud.aws.credentials.access-key}")
    private String accessKey;

    @Value("${cloud.aws.credentials.secret-key}")
    private String secretKey;

    @Value("${cloud.aws.region.static}")
```

```
    private String region;

    @Bean
    public S3Client s3Client() {
        AwsBasicCredentials credentials = AwsBasicCredentials.create(ac
cessKey, secretKey);

        return S3Client.builder()
            .region(Region.of(region))
            .credentialsProvider(StaticCredentialsProvider.create(credential
s))
            .build();
    }
}
```

3. AWS RDS (MySQL)

- **설명**: AWS에서 제공하는 관리형 관계형 데이터베이스 서비스
- **활용 목적**: MySQL 기반의 데이터 저장 및 관리
- **스프링 설정**:

```
spring:
  datasource:
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      maximum-pool-size: 20
      minimum-idle: 5
      idle-timeout: 30000
      max-lifetime: 1800000
      connection-timeout: 30000
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
    properties:
```

```
    hibernate:
      dialect: org.hibernate.dialect.MySQL8Dialect
```

관련 프로젝트 의존성

```
dependencies {
    // Spring Boot 기본 의존성
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    runtimeOnly 'com.mysql:mysql-connector-j'

    // AWS SDK v2: S3 및 DynamoDB
    implementation platform('software.amazon.awssdk:bom:2.24.0')
    implementation 'software.amazon.awssdk:s3'
    implementation 'software.amazon.awssdk:dynamodb'
    implementation 'software.amazon.awssdk:dynamodb-enhanced'
}
```

기타 참고 사항

- 모든 서비스는 사용량에 따라 비용이 발생할 수 있으므로 과금 정책을 확인하고 관리 필요
- 보안 및 접근 제어를 위해 IAM 역할 및 권한을 적절히 설정해야 함
- 각 서비스의 API 문서를 참고하여 연동 및 활용 방법을 숙지할 것

4. openvidu

준비물 : 도메인, EC2인스턴스, 고정 Public IP

1. 쉘 명령어 수행

```
sh <(curl -fsSL http://get.openvidu.io/community/singlenode/latest/install.sh)
```

2. 본인이 준비한 도메인과, IP, 이메일 등 개인 정보 입력
3. https:// 본인도메인/home으로 접속해서 openviducall 페이지가 잘 열리는지 확인

4. 화상채팅 라이브 포트는 7880으로 고정되어 있기 때문에 nginx https location설정 후 http://본인도메인:7880으로 프록시 설정
5. 프론트에서 https://본인도메인:{nginx에서 설정한 https 포트} 로 연결