

# Daily Mission (daily-mission.com)

대국민 미션 인증 프로젝트 ☺

1일 1알고리즘 온라인 모임을 운영하며, 체계적인 관리의 필요성을 느껴 프로젝트를 진행했습니다.

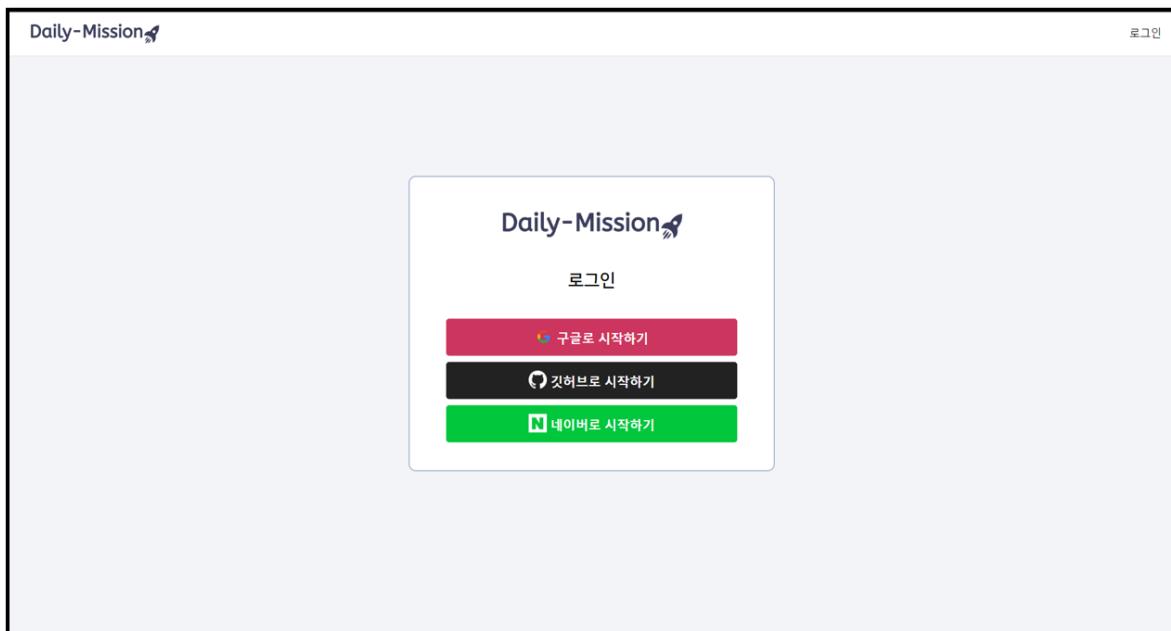
- 누구나 본인들만의 미션을 생성하고 참여자들을 모집해 각자의 미션을 진행할 수 있습니다.
- 미션에 참여하는 사용자는 인증 제출 요일에 반드시 인증 포스트를 작성해야 합니다.
- 제출 요일에 포스트를 작성하지 않은 참여자는 자동으로 해당 미션에서 강퇴됩니다.

## 사용 방법

현재 미션 생성은 관리자만 할 수 있으며, 조만간 일반 사용자들에게 open 할 예정입니다.

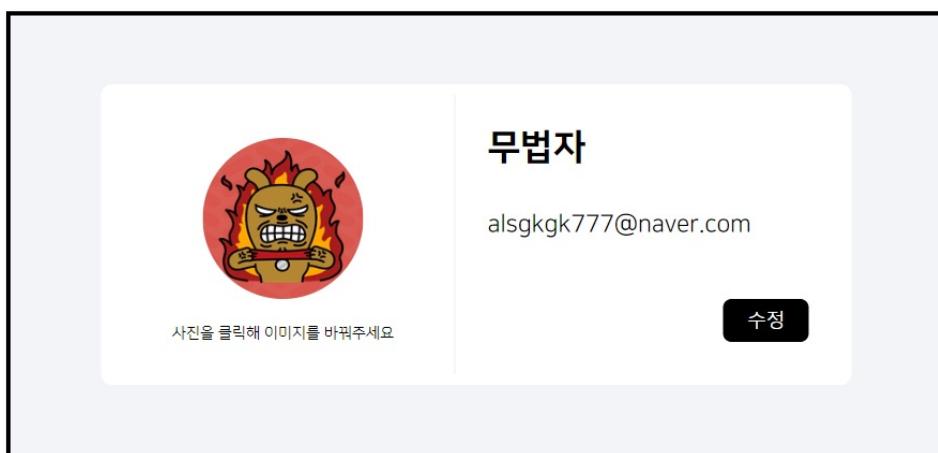
### 1. 로그인

구글/깃허브/네이버로 로그인을 할 수 있습니다.



### 2. 사용자 정보 변경

로그인 후 사용자 이름 / 이미지를 변경할 수 있습니다.



### 3. 흡

참여자가 많은 미션과 신규 생성된 미션을 조회 할 수 있습니다.

Daily-Mission

Hot한 미션

- 1일 1알고리즘 3기  
우법자  
미후 한 문제 알고리즘 풀고 인증하기 😊  
3명 미션 참여중
- 1일 1알고리즘  
우법자  
미후 한 문제 알고리즘 풀고 인증하기 😊  
1명 미션 참여중
- 1일 1유니온  
수락  
미후 한 문제 알고리즘 풀고 인증하기 😊  
1명 미션 참여중
- 1일 1음악듣기  
JOKER  
미후 아침 9시에 유니온에 도착하는 미션입니다.  
1명 미션 참여중
- 저녁 먹기 미션  
수락  
저녁 먹는 미션입니다.  
2명 미션 참여중
- 저녁 먹기 미션  
수락  
저녁 먹는 미션입니다.  
1명 미션 참여중

신규 미션

- 1일 1알고리즘 3기  
우법자  
미후 한 문제 알고리즘 풀고 인증하기 😊  
3명 미션 참여중
- 저녁 먹기 미션  
수락  
저녁 먹는 미션입니다.  
1명 미션 참여중
- 1일 1음악듣기  
JOKER  
1일 1음악듣기  
1명 미션 참여중
- 1일 1소시  
JOKER  
미후에 한번의 소시 먹기..  
1명 미션 참여중
- 1일 1알고리즘  
우법자  
미후 한 문제 알고리즘 풀고 인증하기 😊  
1명 미션 참여중

### 4. 전체 미션

종료된 미션을 포함해 전체 미션을 조회 할 수 있습니다.

Daily-Mission

전체 미션

- 1일 1알고리즘 3기  
우법자  
미후 한 문제 알고리즘 풀고 인증하기 😊  
3명 미션 참여중
- 저녁 먹기 미션  
수락  
저녁 먹는 미션입니다.  
1명 미션 참여중
- 1일 1턱걸이  
JOKER  
미달아일 헉皈이 하는 미션  
0명 미션 참여중
- 1일 1음악듣기  
JOKER  
1일 1음악듣기  
1명 미션 참여중
- 1일 1소시  
JOKER  
미후에 한번의 소시 먹기..  
1명 미션 참여중
- 종료된 미션입니다  
수락  
미달아일 헉皈이 하는 미션  
0명 미션 참여중
- 종료된 미션입니다  
수락  
영어 공부하는 미션입니다  
0명 미션 참여중
- 종료된 미션입니다  
수락  
꽃구경 가자  
꽃구경 가는 미션입니다.  
0명 미션 참여중
- 종료된 미션입니다  
수락  
1DAY 1COMMIT  
소스형상온라인(Github, Giteao)에 commit하고 인증하는 미션!  
1명 미션 참여중
- 종료된 미션입니다  
수락  
주말마다 영화보기  
영화 이후지게 볼실까요?  
2명 미션 참여중

## 5. 미션 디테일

미션 디테일 정보를 확인할 수 있습니다. 또한, 현재 참여중인 사용자와 해당 미션에 제출된 포스트 목록을 조회 할 수 있습니다.

The screenshot shows the 'Daily-Mission' application interface. At the top, there's a navigation bar with icons for Home, 전체 미션 (All Missions), Posts, and MY. A user profile icon is also present. The main area features a large central box with the title 'ALGORITHMS' and a flow diagram of four colored shapes (orange, green, yellow, orange) connected by arrows. To the right, a sidebar displays '무법자's 1일 1알고리즘' (Mu法자's 1 Day 1 Algorithm) with a progress bar indicating '19일 중 1일 남음' (19 days left). Below the sidebar, a section titled '참여자' (Participants) lists three users: 무법자 (Mu法자), 수박 (Pineapple), and NEVER. The bottom half of the screen shows a grid of algorithm challenges with their titles, descriptions, and statistics. For example, one challenge is 'Generate a String With Characters That Have Odd Co' (한글 설명: 주어진 숫자만큼의 알파벳을 표출한다(각 알파벳의 문자는 뽑아야 한다)) with 1 submission, 0 accepted, 0 ms runtime, and 1.6 MB memory usage. Another challenge is '[200329] Increasing Decreasing String' (설명: 주어진 문자를 일정한 순서대로 오름차순 내림차순으로 정복해서 표출하는 문자) with 1 submission, 0 accepted, 0 ms runtime, and 7.8 MB memory usage.

## 6. 미션 참여

미션 생성후 전달받은 참여코드를 입력해 미션에 참여할 수 있습니다.

This screenshot shows the participation details for the '무법자's 1일 1알고리즘 3기' mission. On the left, there's a cartoon illustration of a brown bear with a fiery halo. The right side contains mission details: '2020-04-01 ~ 2020-06-30' and a weekly schedule from Monday to Sunday. A prominent green button at the bottom right says '미션 참여하기' (Participate in the Mission). Below this, a modal window asks for a participation code ('참여하기 위해서는 참여코드가 필요합니다.') and has a '참여 코드' input field and a '확인' (Check) button. At the bottom, it says '미션 포스트가 없습니다' (No mission posts available).

## 7. 포스팅 목록

전체 인증 포스트 목록을 조회할 수 있습니다.

The screenshot shows a post on the Daily-Mission platform. The post title is "Generate a String With Characters That Have Odd Co". It has a runtime of 4 ms, faster than 17.18% of C++ online submissions, and memory usage of 6.6 MB, less than 100.00% of C++ online submissions. The post was submitted a few seconds ago and accepted. It uses C++ and is in English. The post content describes generating a string with odd counts of characters. Below the post is a photo of a window with a view of a building and some plants on a windowsill.

## 8. 내 미션 목록

내가 참여중인 미션 목록과 제출한 포스트 목록을 조회할 수 있습니다. 강퇴당한 미션에는 입장할 수 없습니다.

The screenshot shows the user's mission list on the Daily-Mission platform. The user has three active missions: "1일 1유니온", "1일 1알고리즘", and "1일 1알고리즘 3기". Each mission has a yellow button labeled "강퇴당한 미션입니다". Below the missions is a section titled "내가 쓴 글" (My Posts) which lists several posts with titles like "Generate Parentheses", "3Sum", "3Sum Closet", "유니온 특별일!", "아직도 수요일이오?", etc., each with a thumbnail image and a brief description.

## 9. 인증 포스트 History

참여중인 미션의 Weekly 포스트 History를 조회할 수 있습니다.

The screenshot shows the 'History' section of the Daily-Mission app. At the top left, there are navigation icons: '홈' (Home), '전체 미션' (All Missions), '포스팅' (Posting), and 'MY'. The main title is '1일 1알고리즘 3기'. On the right, there is a '제출 보드' (Submission Board) with a placeholder '미션 사진을 DRAG 해주세요!' (Drag your mission photo here). Below it is a '제출 정보' (Submission Information) section with four icons: '수박' (Watermelon), 'JOKER', 'NEVER', and '무법자' (Lawbreaker). To the right is a weekly calendar grid:

SUN	MON	TUE	WED	THU	FRI	SAT
X	X	X	X	X	휴무	휴무
X	X	X	X	X	휴무	휴무
X	X	X	X	X	휴무	휴무
X	X	X	X	X	휴무	휴무

제출 완료 후에는 아래와 같이 화면이 변경됩니다.

The screenshot shows the same 'History' section after a submission. The weekly calendar grid now has a green checkmark in the 'TUE' column under 'WEEK 3' (the third week of the mission). A green button at the bottom of the board says '오늘 하루도 고마웠습니다!' (Thank you for today!).

## 10. 포스트 제출

제목/내용/사진을 입력해 인증 포스트를 작성합니다.

The screenshot shows a modal dialog box titled '[200331] Generate a String With Characters'. Inside, there is a description: '설명: 주어진 숫자만큼의 알파벳을 표출한다(각 알파벳의 수자는 출수여야 한다)' (Description: Outputs the specified number of characters (each character's count must be odd)). Below it are fields: '난이도: ★☆☆☆☆' (Difficulty: ★☆☆☆☆) and '사용 언어: C++'. At the bottom of the dialog is a '제출' (Submit) button with a green checkmark icon. The background of the app shows the same mission history as the previous screenshots.

# Project Structure

React(SPA) + Spring Boot(API Server) 구조로 개발했으며, 저는 API Server 를 담당했습니다.

프로젝트에 사용할 기술 목록을 사전에 정의하고, 약 5개월간 해당 기술들을 학습 후 [기술 블로그](#)에 정리했습니다.

사용한 기술스택은 다음과 같습니다.

- Spring Boot (API Server)
- Spring Security (Security)
- Spring Batch (Batch)
- MariaDB (RDB)
- JPA & QueryDSL (ORM)
- OAuth2.0 + JWT (Login)
- Redis (Cache)
- JUnit (Test)
- AWS (Infra)
- Nginx (Reverse Proxy Server)
- Rabbit MQ (Message Broker)
- Jenkins & Codedeploy (CI/CD)

ERD & API는 다음과 같이 정리했습니다.

- ErdCloud : <https://www.erdcloud.com/d/HcjcicwpDs8zmGdCqL>
- GitBook : <https://minholee93.gitbook.io/daily-mission/~settings/share>

■ 프로젝트에서는 API Server / Infra / 기획 / 설계 / 일정관리 등을 담당했습니다.

## Spring Boot (API Server)

React(SPA)에서 요청한 데이터를 JSON으로 response 한다.

구조는 다음과 같습니다.

- config : project configuration을 관리한다.
- exception : custom exception message를 관리한다.
- security : security, oauth, jwt 관련 기능들을 관리한다.
- util : util 기능들을 관리한다.
- web
  - controller : API를 관리한다.
  - dto : request/response dto를 관리한다.
  - repository : domain + JPA/QueryDSL를 관리한다.
  - service : domain에 정의한 business logic 호출 순서를 관리한다.

■ 비즈니스 로직은 service가 아닌 반드시 domain에 작성했습니다.

```
mission.setCredential()
mission.matchCredential()
mission.isPossibleToParticipate()
mission.updateImage()
mission.isDeletable()
mission.isEndable()
mission.getHistories()
mission.getAllParticipantUser()
mission.getParticipantCountNotBanned()
mission.isValidFileExtension()
mission.isValidStartDate()
mission.isValidMission()
...
```

# Spring Security (Security)

Security 설정을 추가해 인가된 사용자만 특정 API에 접근할 수 있도록 제한한다. 또한, CORS 설정을 통해 허용된 도메인에서만 API를 호출할 수 있다.

구조는 다음과 같습니다.

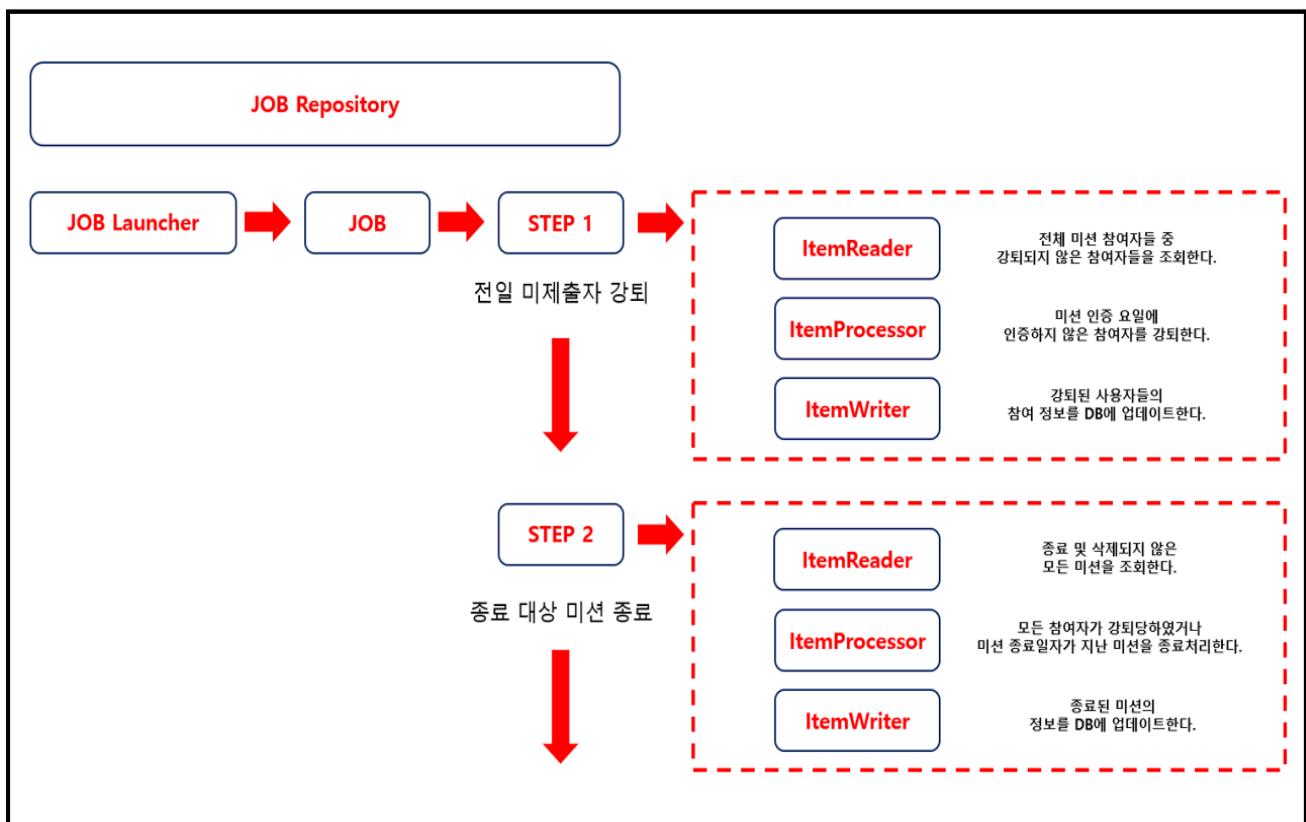
- Allowed Origins : <https://daily-mission.com>
- Session Creation Policy : STATELESS
- CSRF : disable
- Form Login : disable
- Authentication Entry Point : RestAuthenticationEntryPoint.class
- Token Authentication Filter : UsernamePasswordAuthenticationFilter.class

■ 전체 User가 접근할 수 있어야 하는 API는 permitAll()을 선언했습니다. 반대로 인가된 사용자만 접근할 수 있어야 하는 API에는 @PreAuthorize를 선언해 접근을 제한했습니다.

# Spring Batch (Batch)

매일 새벽 3시에 jenkins job이 미인증 사용자들을 강퇴하고, 종료일자가 지난 미션을 종료한다.

구조는 다음과 같습니다.



■ Batch Job의 중복 처리방지를 위해 job parameter를 전달받아 batch job의 면등성을 유지했습니다.

```
@Value ("#{jobParameters[requestDate]}")
```

# JPA & QueryDSL (ORM)

객체 중심 domain 설계 및 반복적인 CRUD 작업을 대체해 비즈니스 로직에 집중한다.

- JPA : 반복적인 CRUD 작업을 대체해 간단히 DB에서 데이터를 조회한다.
- QueryDSL : Join & Projections 등 JPA로 해결할 수 없는 SQL은 QueryDSL로 작성한다.

구조는 다음과 같습니다.

- Post (Domain Class)
- PostRepository (JPA Interface)
- PostRepostioryCustom (QueryDSL Interface)
- PostRepositoryCustomImpl (QueryDSL Implements Class)

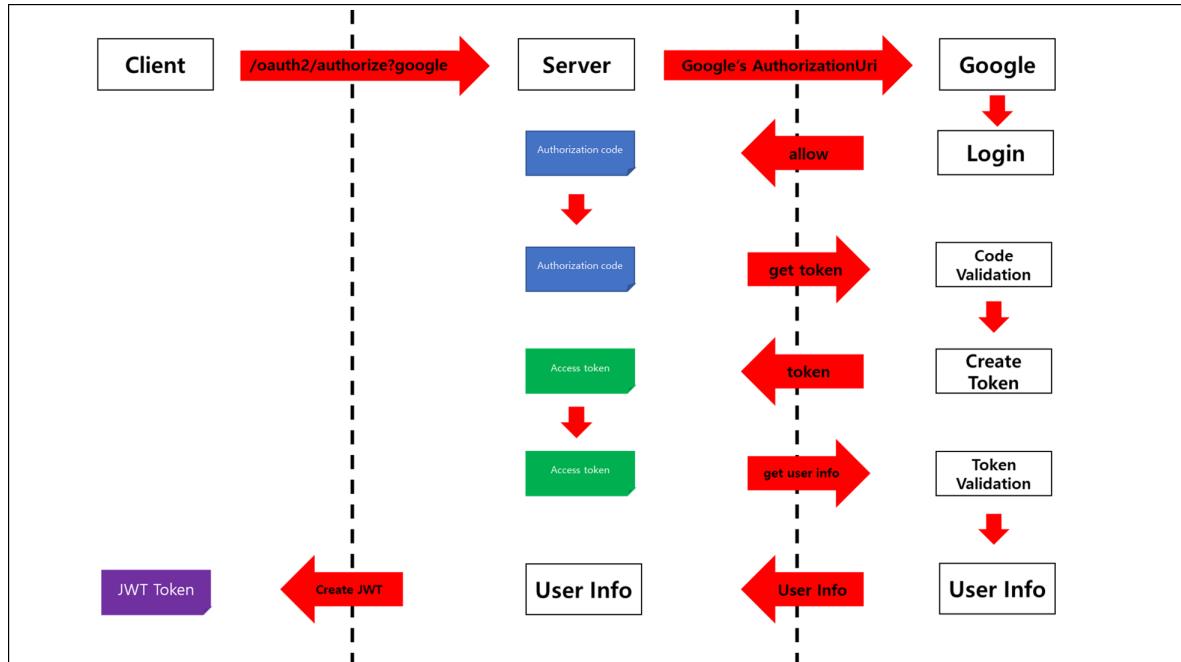
■ JPA와 QueryDSL로 구현한 CRUD는 JUnit Test로 반드시 실행되는 SQL을 직접 확인했습니다.

```
select *
from mission mission0_
where mission0_.deleted=?
order by (
    select count(participant1_.mission_id)
    from participant participant1_
    where mission0_.id=participant1_.mission_id
) desc,
mission0_.created_date desc
```

## OAuth2.0 + JWT (Login)

구글/깃허브/네이버 oauth provider를 사용해 불필요한 회원가입 프로세스를 제거한다. 또한, JWT Token을 사용해 Authorization Header 기반 인증 시스템을 구현한다.

구조는 다음과 같습니다.



■ 이름/이메일/사진 3가지 정보만 oauth provider에 요청해, token 유출에 따른 보안 문제를 최소화했습니다.

# Redis (Cache)

Global Cache Server를 사용해 반복적인 메서드의 호출을 차단, API 응답 성능을 높인다.

구조는 다음과 같습니다.

- @CachePut : key 값의 Cache를 갱신한다.
- @Cacheable : key가 존재할 경우 Cache 된 결과값을 Return 한다. 존재하지 않을 경우 메서드를 실행 후 결과값을 Cache 한다.
- @CacheEvict : key 값의 Cache를 제거한다.
- TTL : Time-To-Live 를 설정해 Cache가 Alive 할 수 있는 최대 시간을 지정한다.

■ JUnit Test에서 Cache가 활성화 될 경우, 정상적으로 Test를 수행할 수 없습니다. 따라서 test 환경의 application.yml은 cache를 disable 했습니다.

```
spring.cache.type : none
```

# JUnit (Test)

Layer 별로 Bean을 최소한으로 등록시켜 테스트 하고자 하는 로직에 집중해 테스트를 수행한다.

구조는 다음과 같습니다.

- Domain 테스트 : domain 객체들은 가장 핵심이며, 이 객체를 사용하는 계층들이 프로젝트에 다양하게 분포되기 때문에 반드시 테스트 코드를 작성한다.

```
public class MissionTest {  
    ...  
}
```

- Repository 테스트 : @DataJpaTest 어노테이션을 통해서 Repository에 대한 Bean만 등록한다. 커스텀하게 작성한 쿼리 메서드, QueryDSL 등의 메서드를 테스트한다. ORM은 SQL을 직접 작성하지 않으니 반드시 실제 쿼리가 어떻게 출력되는지 확인한다.

```
@RunWith(SpringRunner.class)  
@DataJpaTest(includeFilters = @ComponentScan.Filter(  
    type = FilterType.ASSIGNABLE_TYPE,  
    classes = {JpaConfig.class, QueryDslConfig.class}  
)  
public class MissionRepositoryTest {  
    ...  
}
```

- Service 테스트 : 테스트 진행시 중요 관점이 아닌 것들은 Mocking 처리해서 외부 의존성을 줄인다.

```
@RunWith(MockitoJUnitRunner.class)  
public class MissionServiceTest {  
    ...  
}
```

- Controller 테스트 : 모든 Bean을 올리고 테스트를 진행한다. @Transactional 어노테이션을 추가해 테스트 후 DB를 자동으로 RollBack 한다.

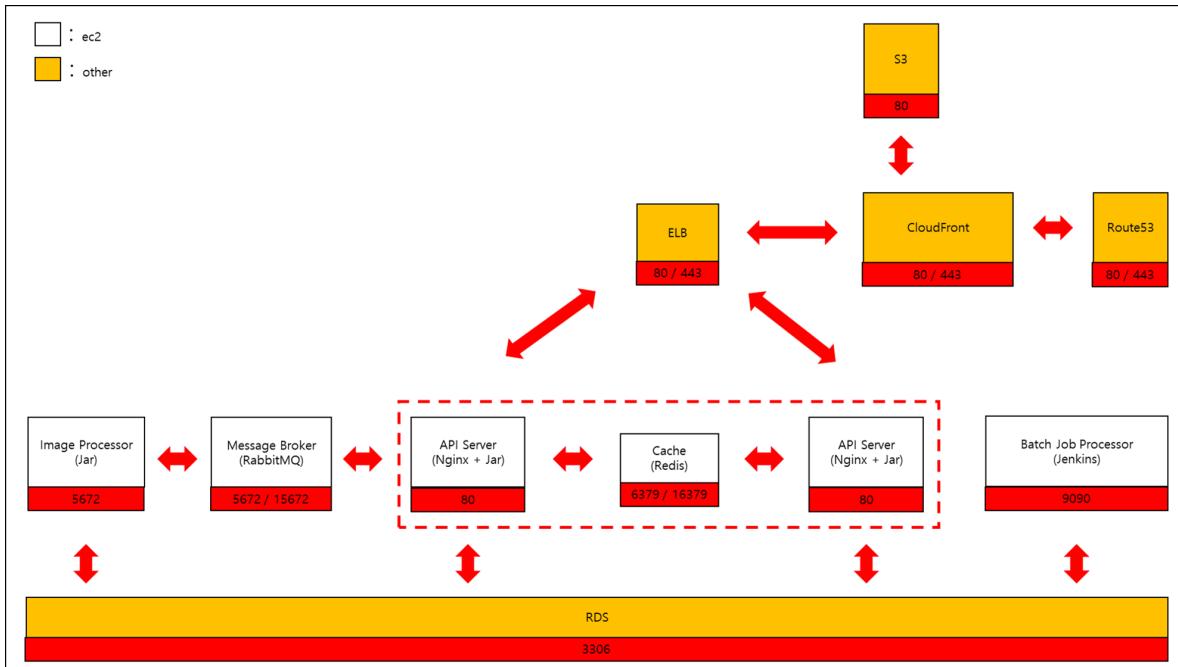
```
@RunWith(SpringRunner.class)  
@SpringBootTest(properties = "spring.config.location=classpath:/application.yml")  
@Transactional  
public class MissionControllerTest {  
    ...  
}
```

■ 총 96개의 Test Case를 작성했습니다. (mission : 45 / Participant : 8 / Post : 26 / User : 17)

# AWS (Infra)

전체 프로젝트 인프라 구성 및 계정 별 권한을 관리한다.

구조는 다음과 같습니다.



■ EC2의 ssh 접근권한은 반드시 본인의 IP 만 허용했으며, 사용자/서버 별 IAM 계정 및 권한을 부여해 보안성을 강화했습니다.

## Nginx (Reverse Proxy Server)

클라이언트로부터 전달받은 요청을 어플리케이션 서버에 전달한 뒤, 어플리케이션 서버가 반환한 결과값을 다시 클라이언트에게 전달한다.

구조는 다음과 같습니다.

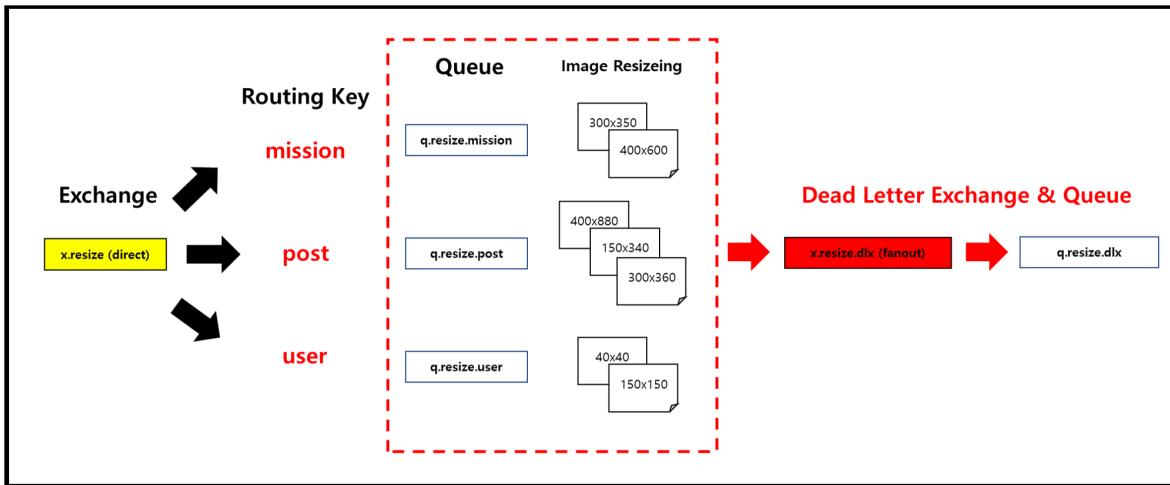
- worker\_process **auto** : server의 core 갯수만큼 worker process를 생성한다.
- worker\_connections **1024** : core별로 동일한 file을 한번에 open 할 수 있는 number 값으로 limit을 설정한다.
- client\_body\_buffer\_size **10K** : post submission의 buffer size를 설정한다.
- client\_max\_body\_size **8m** : post submission의 form data size를 설정한다.
- client\_header\_buffer\_size **1K** : cleint header의 buffer size를 지정한다.
- client\_body\_timeout **12** : client body를 receive 할 수 있는 max time 을 설정한다.
- client\_header\_timeout **12** : client header를 receive 할 수 있는 max time 을 설정한다.
- keepalive\_timeout **15** : 다음 data 를 받기 위해 connection을 열어 놓는 최대 시간을 정의한다.
- send\_timeout **10** : client가 response 된 data 중 아무것도 받지 않는 최대 시간을 정의한다.
- add\_header X-Frame-Options “**SAMEORIGIN**” : 클릭재킹 공격을 방어 한다.
- add\_header X-XSS-Protection “**1; mode=block**” : XSS 공격을 방어 한다.
- limit\_req\_zone \$binary\_remote\_addr zone=**MYZONE**:10m rate=**1r/s** : USER 별로 incoming connection rate를 제한한다.
- limit\_req zone=**MYZONE** burst=**5** : rate limiting 을 초과하는 connection을 즉시 reject 하지 않고 wait 한다.

■ Application Server와의 중복된 CORS 설정을 방지하기위해 Nginx에는 CORS 설정을 하지 않았습니다. 또한, proxy\_set\_header를 통해 client에서 전달된 header를 Application Server로 전달했습니다.

## Rabbit MQ (Message Broker)

화면 별 규격에 맞게 image를 resize 한다. direct exchange와 routing key를 사용해 queue와 연결된 consumer에게 resizing job을 분배한다.

구조는 다음과 같습니다.



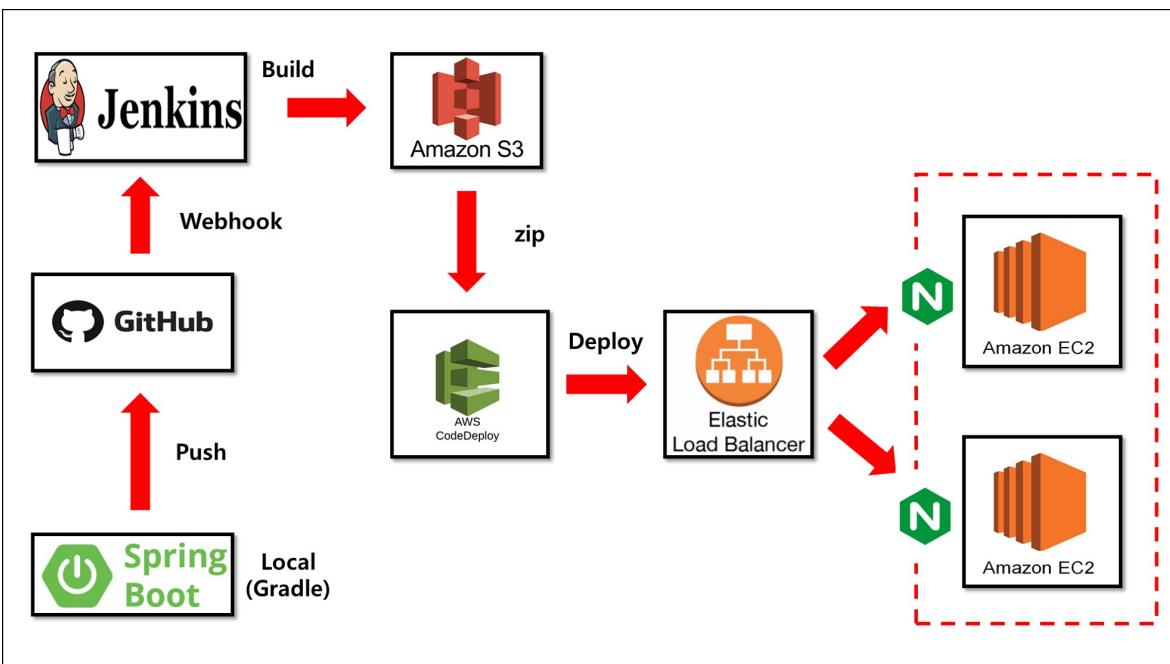
■ spring amqp를 사용해 손쉽게 retry mechanism 을 구현했습니다.

- initial-interval : 3s
- max-interval: 10s
- max-attempts: 5
- multiplier: 2

## Jenkins & CodeDeploy (CI/CD)

Jenkins와 AWS의 CodeDeploy를 사용해 CI/CD를 구현한다.

구조는 다음과 같습니다.



■ AWS의 ELB를 사용해 무중단 배포를 구축했습니다.