

초급 백엔드 스터디

8주차 – 프로젝트 배포, API 문서화

지난 주에는...

- 유효성 검증
- Global Exception Handler (AOP)
- 예외 메시지 리팩토링

이번 주에는...

- API 문서화
- 프로젝트 배포

API 문서화

- 백엔드가 만든 API에 대한 사용법을 문서로 공유하는 것
- 프론트엔드와 협업할 때 API 문서를 공유한다.

API 문서화 과정

1. **spring doc**을 이용해서 OpenAPI 규격으로 API 문서 생성
(OpenAPI : 표준 API 문서 규격)
2. **Swagger-ui**를 사용하여 spring doc이 생성한 API 문서에 swagger 디자인 적용
(swagger 이외에도 다양한 API 문서화 도구가 있습니다.)

API 문서화

- Spring doc 의존성을 추가한다.

implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.6.0'

```
dependencies {  
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.6.0'  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.boot:spring-boot-starter-validation'  
    compileOnly 'org.projectlombok:lombok'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'  
}
```

API 문서화

- spring doc은 swagger-ui를 적용한 문서를 만들어준다.
<http://localhost:8080/swagger-ui/index.html> 접속

todo-controller	
POST	/todo
DELETE	/todo/{todoId}
PATCH	/todo/{todoId}
GET	/todo/list

API 문서화

- API 호출 시 필요한 정보를 알 수 있고, 요청을 보내볼 수도 있다.


The image shows a user interface for an API documentation tool. At the top, there is a header bar with a green tab labeled 'POST' and a text input field containing '/todo'. To the right of the input field are two icons: a clipboard and an upward-pointing arrow. Below the header bar, there is a section titled 'Parameters' with a 'Try it out' button to its right. Underneath the 'Parameters' section, it says 'No parameters'. Below that, there is a section for the 'Request body' with the word 'required' in red text. To the right of this section is a dropdown menu showing 'application/json'. At the bottom, there are two tabs: 'Example Value' and 'Schema'. The 'Example Value' tab is active, showing a JSON object in a dark box:

```
{  "content": "string",  "userId": 0}
```


API 문서화

- API 호출 시 필요한 정보를 알 수 있고, 요청을 보내볼 수도 있다.

Server response

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "message": "존재하지 않는 user id 입니다." }</pre> <p> Download</p> <p>Response headers</p> <pre>connection: close content-type: application/json date: Mon, 02 Sep 2024 01:09:38 GMT transfer-encoding: chunked</pre>

API 문서화

- @ApiResponse를 사용하여 status code마다 설명을 적을 수 있다.

```
@PostMapping  
@ApiResponse(responseCode = "404", description = "요청에 들어온 user id 가 존재하지 않음")  
public ResponseEntity<Void> createTodo(@RequestBody @Valid TodoCreateRequest request) throws Exception {  
    Long todoId = todoService.createTodo(request.getContent(), request.getUserId());  
    return ResponseEntity.created(URI.create("/todo/" + todoId)).build();  
}
```

Code	Description
404	요청에 들어온 user id 가 존재하지 않음

프로젝트 배포

- 누구나 인터넷을 통해 요청을 보낼 수 있도록
우리가 만든 서버를 인터넷에 연결된 서버 컴퓨터에서 24시간
실행시켜 두는 것

프로젝트 배포

- 노트북도 설정만 잘 해주면 서버 컴퓨터로 쓸 수 있다.
하지만 서버 컴퓨터를 설정하고 관리하는 것은 매우 복잡하다.
(네트워크 설정, 방화벽 설정, 컴퓨터 온도 관리, 모니터링 등)
- 그래서 보통 잘 관리되는 **서버 컴퓨터를 빌려서 배포**한다.

프로젝트 배포

- 서버 컴퓨터에서 우리가 만든 어플리케이션을 어떻게 실행할 수 있을까?
- 소스코드 옮기기 → 인텔리제이 설치하기 → 실행하기?

- 인텔리제이가 실행하는 방법

- [illegible]

프로젝트 배포

- 매번 변경된 소스 코드를 git으로 받아온 뒤 명령어로 실행하는 것은 어렵다.
- 따라서 어플리케이션을 하나의 파일로 실행할 수 있도록, 여러 소스코드를 합치는 **빌드** 과정을 거쳐 실행한다.

프로젝트 배포

- 자바 어플리케이션 빌드
- ./gradlew build

```
→ gdsc-2024-2-backend-study git:(week7-real) ✕ ./gradlew build
```

```
Welcome to Gradle 8.8!
```

```
Here are the highlights of this release:
```


프로젝트 배포

- 자바 어플리케이션 빌드
- 어플리케이션을 빌드할 때는 모든 테스트가 통과해야 한다.

```
TodoRepositoryTest > todoUpdateTest() FAILED
    java.lang.IllegalStateException at DefaultCacheAwareContextLoaderDelegate.java:145

TodoRepositoryTest > todoFindOneByIdTest() FAILED
    java.lang.IllegalStateException at DefaultCacheAwareContextLoaderDelegate.java:145

TodoRepositoryTest > todoFindAllByMemberTest() FAILED
    java.lang.IllegalStateException at DefaultCacheAwareContextLoaderDelegate.java:145
```

프로젝트 배포

- 자바 어플리케이션 빌드
- 레포지토리 테스트는 제거하고, 서비스 테스트는 통과하도록 수정

프로젝트 배포

- 자바 어플리케이션 빌드
- 빌드 성공시 하나의 실행가능한 **.jar** 파일이 만들어진다.

```
BUILD SUCCESSFUL in 5s  
7 actionable tasks: 4 executed, 3 up-to-date
```

✓ build

> classes

> generated

✓ libs

todo-api-0.0.1-SNAPSHOT.jar

todo-api-0.0.1-SNAPSHOT-plain.jar

프로젝트 배포

- 자바 어플리케이션 실행
- `java -jar "jar 파일"`

```
→ gdsc-2024-2-backend-study git:(week8-practice) × java -jar build/libs/todo-api-0.0.1-SNAPSHOT.jar
```

```
2024-11-20T11:32:50.713+09:00 INFO 79131 --- [todo-api] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-11-20T11:32:50.815+09:00 WARN 79131 --- [todo-api] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-11-20T11:32:50.982+09:00 INFO 79131 --- [todo-api] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2024-11-20T11:32:50.994+09:00 INFO 79131 --- [todo-api] [main] com.example.todoapi.TODOApiApplication : Started TODOApiApplication in 2.23 seconds (process running for 2.47)
```

프로젝트 배포 과정

1. 로컬에서 어플리케이션을 **빌드**하여 **jar 파일**을 만든다.
2. Java가 설치된 서버 컴퓨터에 jar 파일을 옮겨 놓는다.
3. 서버 컴퓨터에서 **jar 파일**을 **실행**한다.

프로젝트 배포

- **AWS, GCP** 와 같은 서비스를 통해 컴퓨터를 대여 받을 수 있다.
대여받은 컴퓨터에 java를 설치하고 jar 파일을 옮겨서 실행하면
서버 컴퓨터에 스프링 서버를 실행할 수 있다.
(멀리 떨어진 서버에 java를 어떻게 설치하고, jar 파일을 어떻게 옮길까요?)

프로젝트 배포

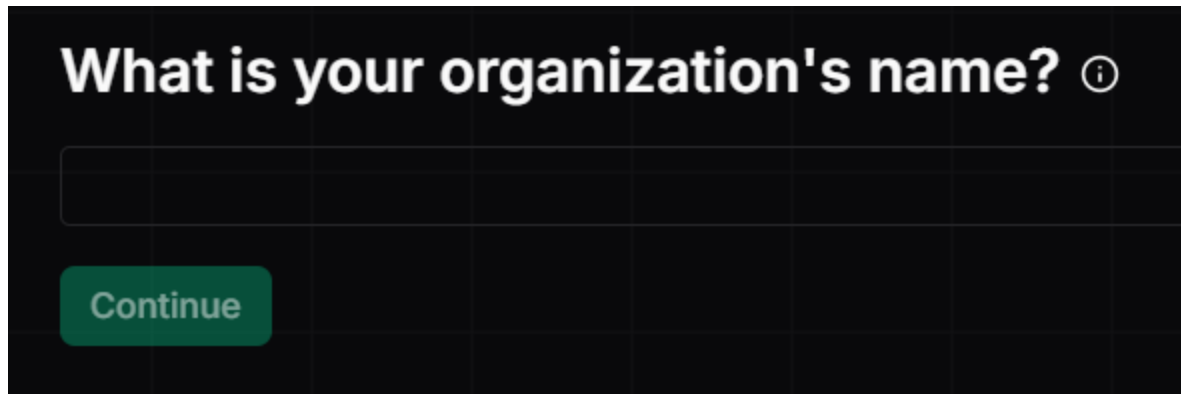
- **Koyeb, Qoddi** 와 같은 서비스는 컴퓨터를 빌려줄 뿐만 아니라, 깃허브와 연동해서 빌드, 실행까지 자동화 해준다.
(실제로는 aws 인스턴스를 빌려서 배포하는 게 일반적입니다.)

프로젝트 배포

- <https://app.koyeb.com/>
- 깃허브 계정 또는 이메일로 회원가입하자.
(ppt와 동일하게 진행하고 싶다면 이메일로 가입하자)

프로젝트 배포

- 조직 이름을 입력한다.

A screenshot of a dark-themed web form. At the top, the text "What is your organization's name?" is displayed in white, followed by a small circular information icon. Below this is a rectangular text input field. At the bottom left of the form area is a green button with the word "Continue" in white text.

What is your organization's name? ⓘ

Continue

프로젝트 배포

Myself

School

Work

What is your role?

Hobbyist

What do you plan to build on Koyeb?

Personal project

Which language do you use for the project you intend to deploy?

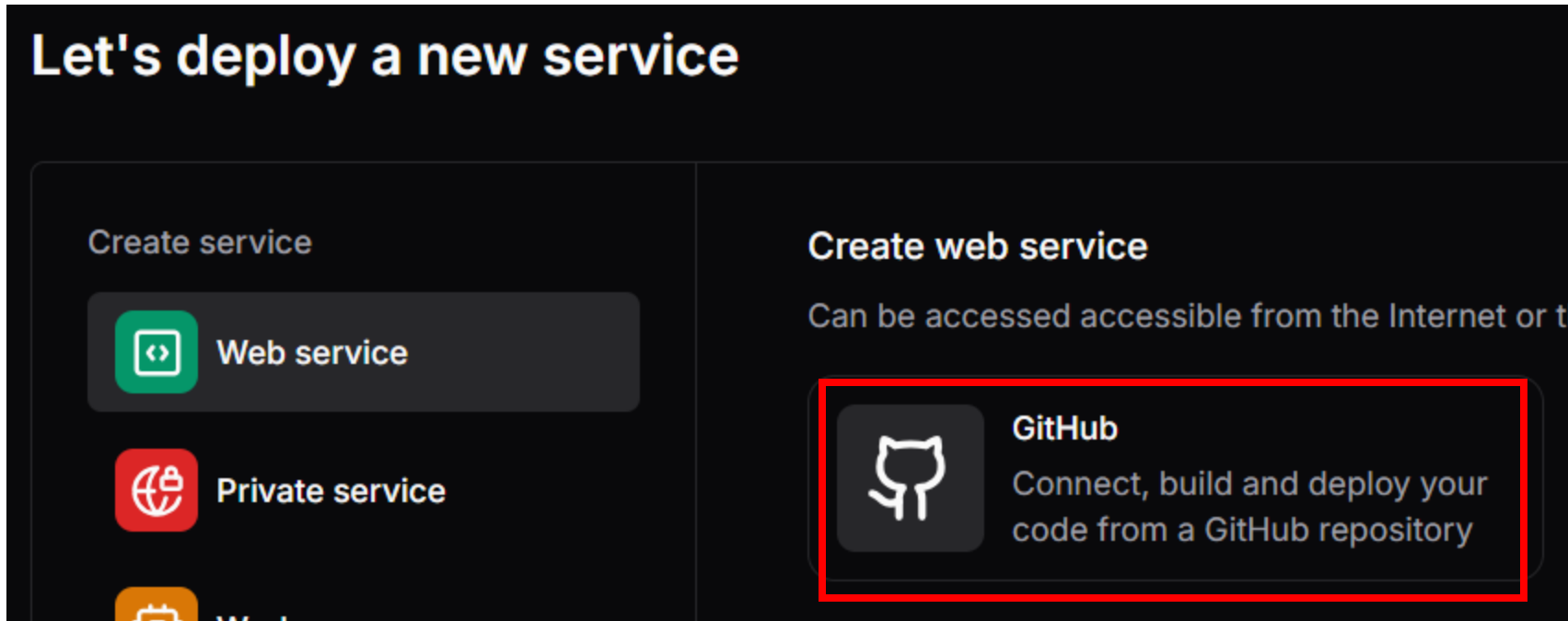
Java

How did you hear about Koyeb?

Search engine

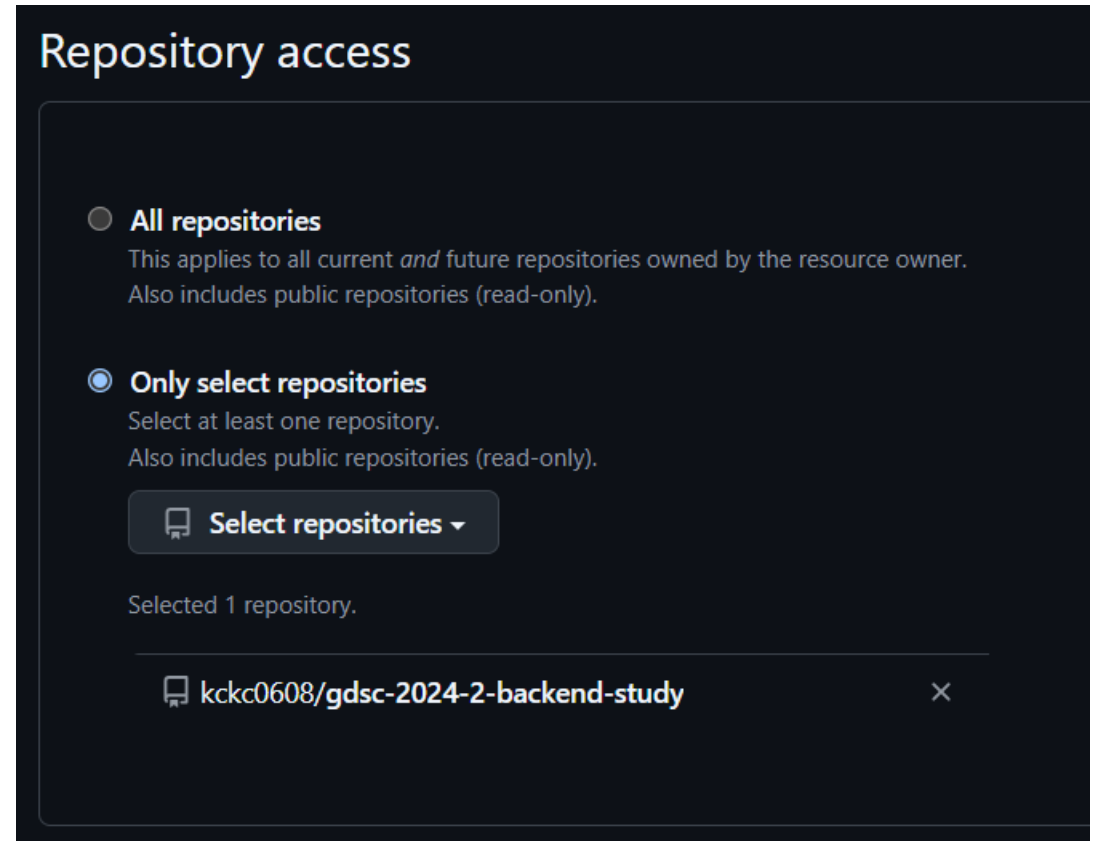
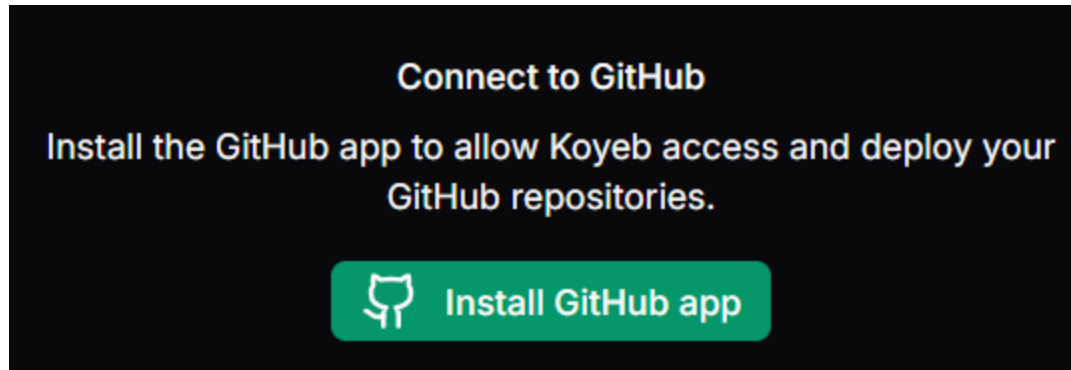
프로젝트 배포

- 배포 서비스는 **Github** 선택



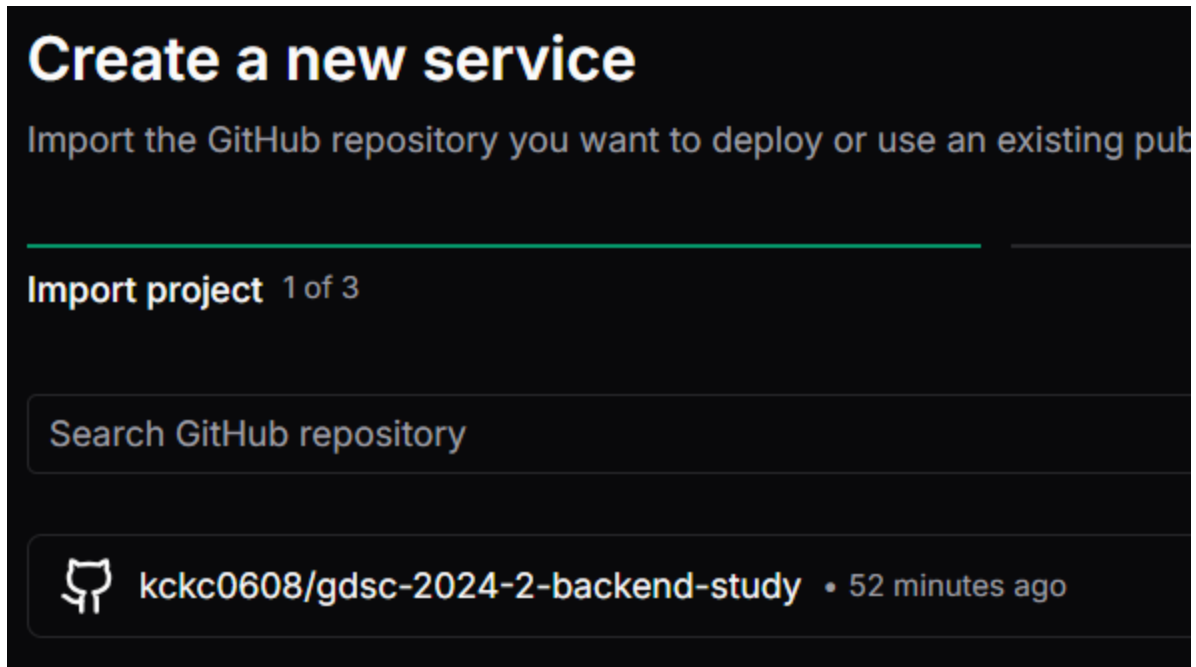
프로젝트 배포

- Github 레포지토리를 연동한다.



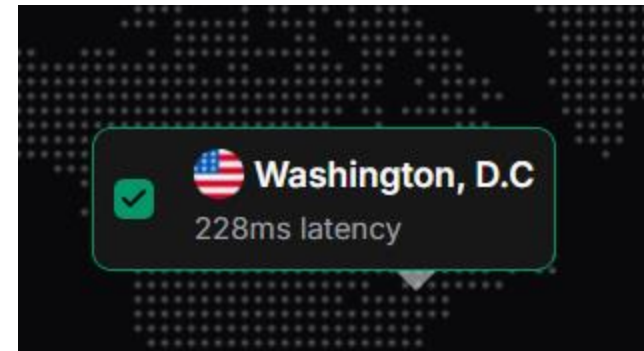
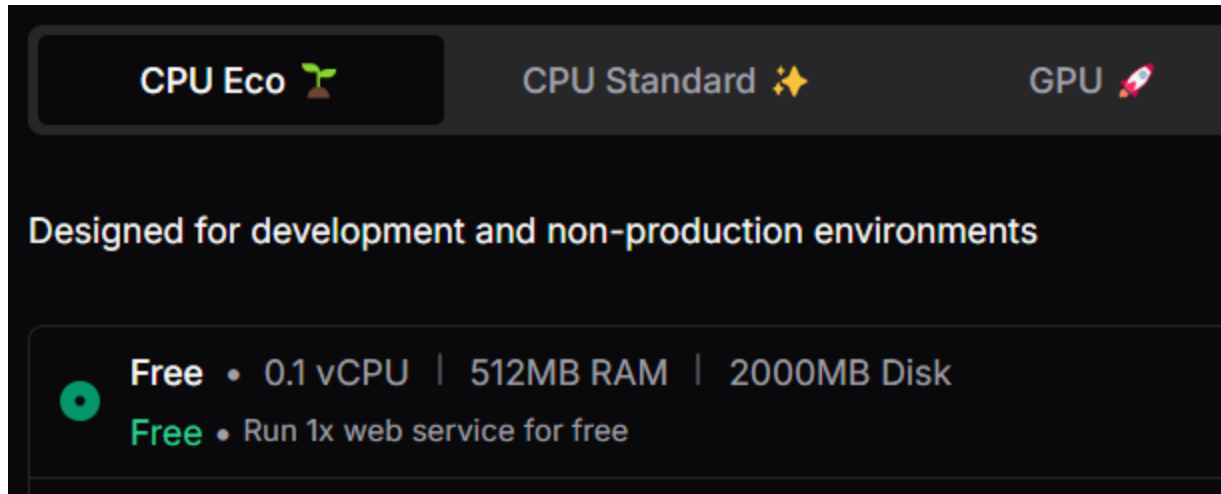
프로젝트 배포

- 프로젝트 레포지토리를 선택한다.



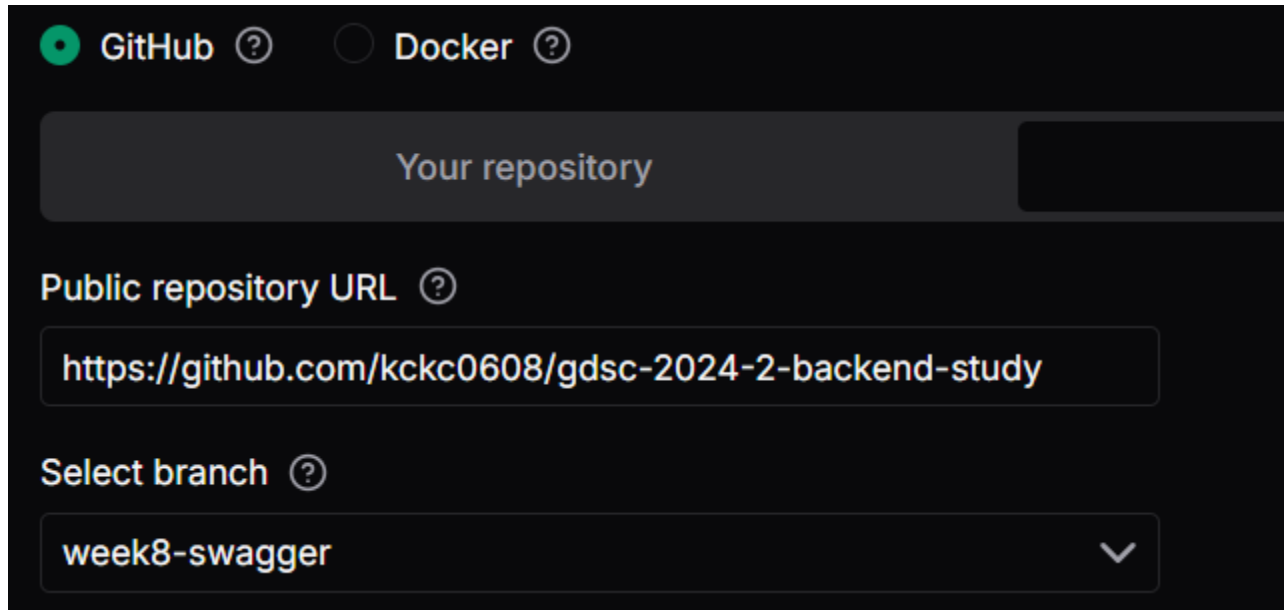
프로젝트 배포

- CPU는 무료 CPU, 배포 지역은 지연율이 제일 낮은 곳을 선택한다.



프로젝트 배포

- 배포하려는 코드가 있는 브랜치를 선택한다.



The screenshot shows a dark-themed deployment interface. At the top, there are two radio buttons: 'GitHub' (selected) and 'Docker'. Below this is a section labeled 'Your repository' with a text input field containing the URL 'https://github.com/kckc0608/gdsc-2024-2-backend-study'. Underneath, there is a 'Select branch' dropdown menu with 'week8-swagger' selected and a downward arrow icon.

GitHub ? Docker ?

Your repository

Public repository URL ?

`https://github.com/kckc0608/gdsc-2024-2-backend-study`

Select branch ?

`week8-swagger` ▼

프로젝트 배포

- Koyeb은 깃허브에서 빌드할 어플리케이션 소스코드를 가져오고, Koyeb이 미리 준비한 자바 환경에서 소스코드를 빌드한다.

1. 어플리케이션을 **빌드**하여 **jar 파일**을 만든다.
2. Java가 설치된 서버 컴퓨터에 jar 파일을 옮겨 넣는다.
3. 서버 컴퓨터에서 **jar 파일을 실행**한다.

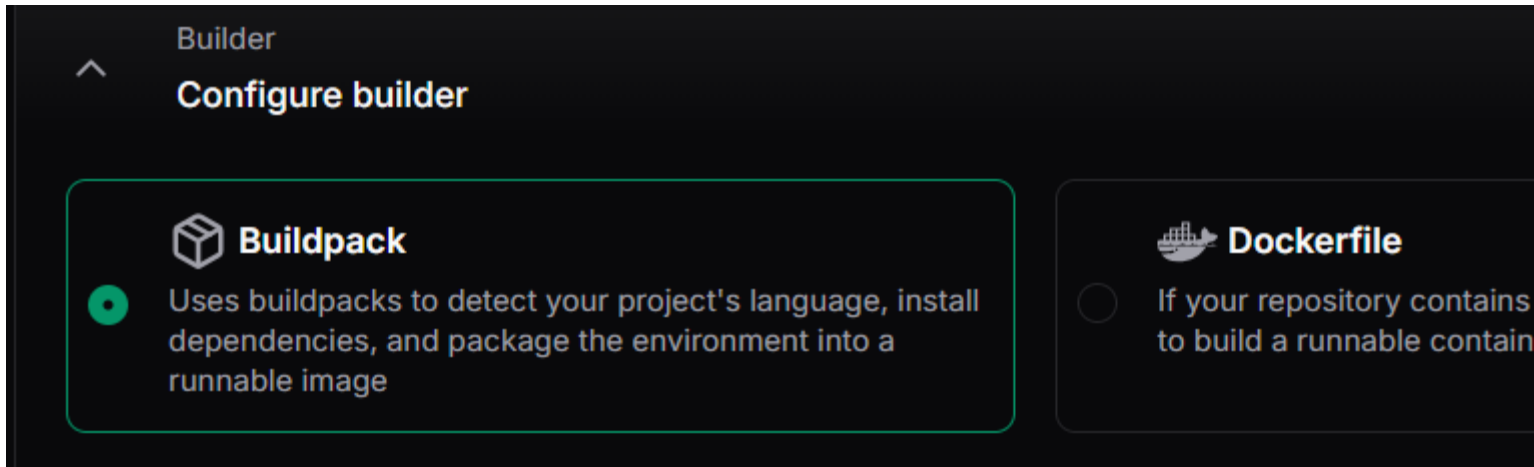
프로젝트 배포

- Koyeb이 제공하는 컴퓨터에 빌드한 jar 파일이 생긴다.

1. 어플리케이션을 **빌드**하여 **jar 파일**을 만든다.
2. Java가 설치된 서버 컴퓨터에 jar 파일을 옮겨 넣는다.
3. 서버 컴퓨터에서 **jar 파일을 실행**한다.

프로젝트 배포


- 빌더는 **Buildpack**을 선택한다.





프로젝트 배포

- Buildpack 설정에서 Run command를 다음과 같이 설정한다.


Configure Buildpack
Customize Buildpack configuration to meet the needs of your repository


Build command 

 Override


Run command 


java -jar build/libs/todo-api-0.0.1-SNAPSHOT.jar

 Override

Work directory 

./api

 Override

☐ Privileged 

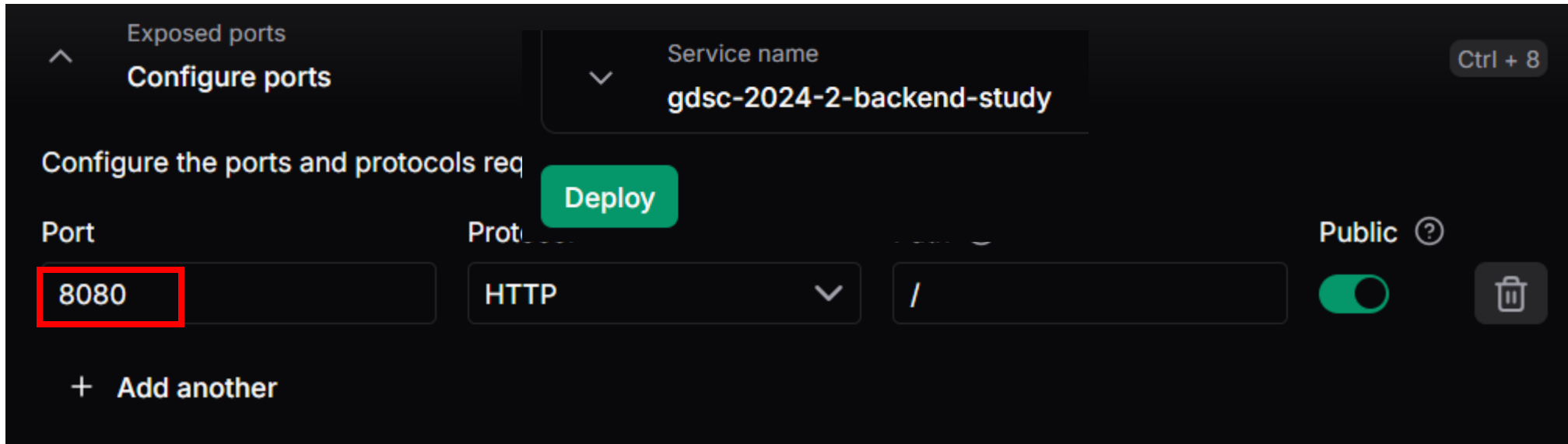
프로젝트 배포

- 이 명령어는 배포 과정에서 3번 과정에 해당하는 명령어이다.

1. 어플리케이션을 **빌드**하여 **jar 파일**을 만든다.
2. Java가 설치된 서버 컴퓨터에 jar 파일을 옮겨 넣는다.
3. 서버 컴퓨터에서 **jar 파일을 실행**한다.

프로젝트 배포

- 어플리케이션을 실행할 포트를 8080으로 지정한다.



Exposed ports

Configure ports


Service name

gdsc-2024-2-backend-study

Ctrl + 8

Configure the ports and protocols required for the application

Deploy

Port	Protocol	Path	Public	
8080	HTTP	/	<input checked="" type="checkbox"/>	

+ Add another

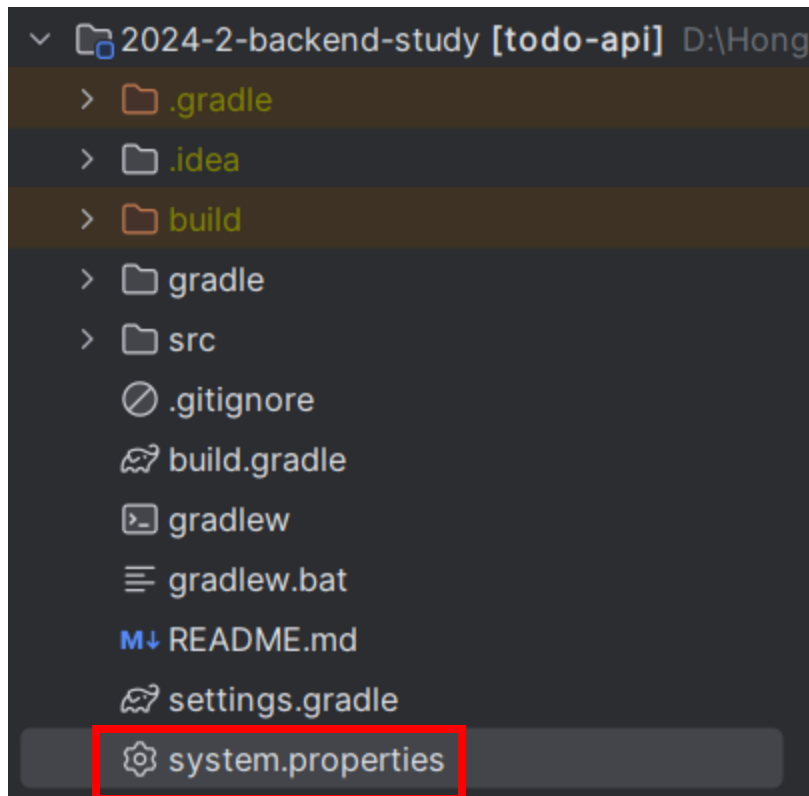
프로젝트 배포

- Koyeb은 기본적으로 빌려줄 컴퓨터에 java 8을 설치한다.
- 우리는 java 17을 사용하므로 Koyeb이 java 17을 설치하도록 세팅 해주자.

프로젝트 배포

- 프로젝트 루트에 **system.properties** 파일을 만든다.

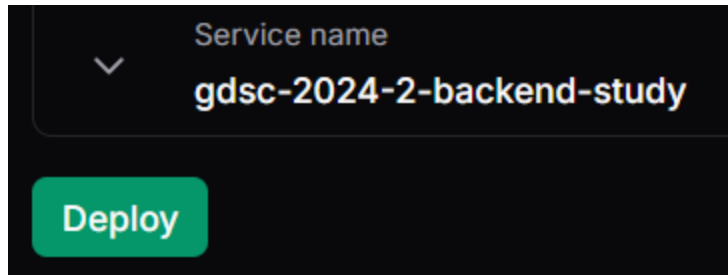
(Koyeb 공식문서 참고 : <https://www.koyeb.com/docs>)



```
1 java.runtime.version=17
```

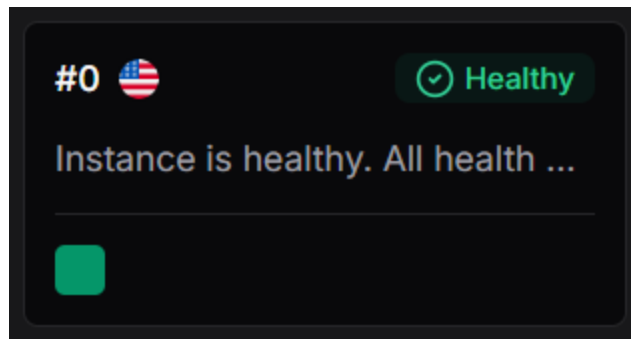
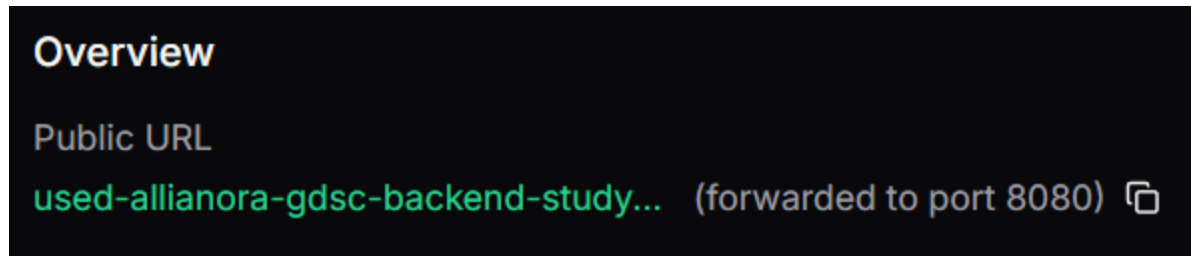
프로젝트 배포

- 설정이 끝났다면 Deploy 버튼을 클릭한다.



프로젝트 배포

- Public URL이 생성되고, 빌드와 배포가 진행된다. (몇 분 소요)




프로젝트 배포

- Public URL 경로 뒤에 `/swagger-ui/index.html` 을 붙여 접속해보자.

Overview

Public URL

`used-allianora-gdsc-backend-study...` (forwarded to port 8080) 

- 위 경로의 경우

`https://used-allianora-gdsc-backend-study-b53b3dee.koyeb.app`**`/swagger-ui/index.html`**

프로젝트 배포

- API 문서가 나오면 배포가 잘 된 것이다.

Servers

<http://used-allianora-gdsc-backend-study-b53b3dee.koyeb.app> - Generated server url

todo-controller

POST /todo

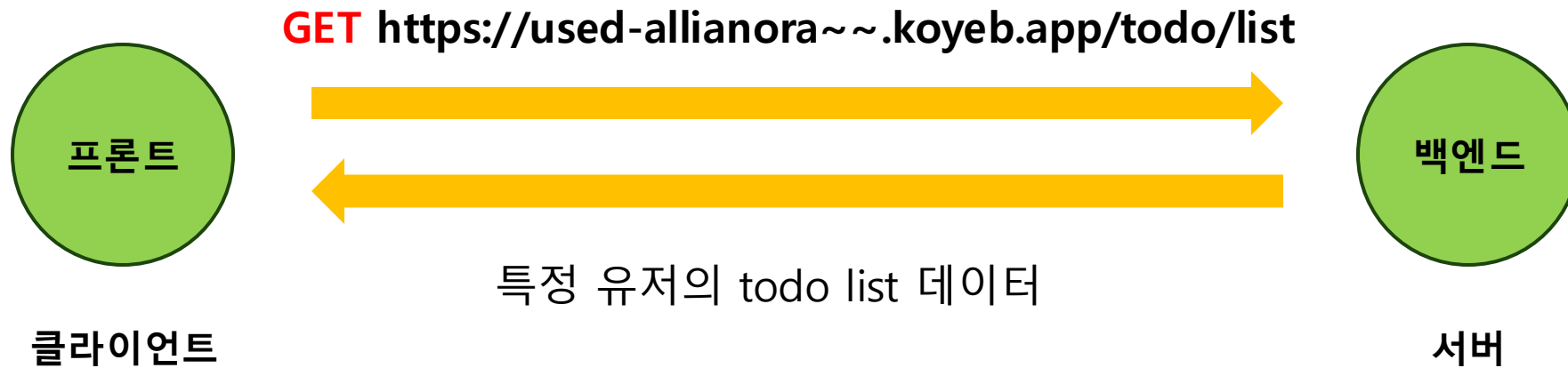
DELETE /todo/{todoId}

PATCH /todo/{todoId}

GET /todo/list

프로젝트 배포

- 클라이언트는 public URL로 백엔드와 데이터를 주고 받는다.



프로젝트 – 과제

- swagger-ui API 문서를 만들어보자.
- Koyeb을 사용해서 프로젝트를 배포해보자.

다음으로

- AWS, Docker 를 공부하고 직접 서버를 설정해서 배포해보기
- Java가 어떻게 빌드되는지, **gradle, jvm** 관련 내용을 더 찾아보기
- Spring Security를 공부하고 로그인, 세션, jwt를 공부해보기

(개발자 유미 유튜브 강의 영상 추천 : <https://www.youtube.com/@xxxjjhhh>)