S.

API Rail

대용량 트래픽 병렬처리 서비스 연구

국민대학교 소프트웨어학부
캡스톤디자인 2022-31조

S. Contents

S. About the Team

강동호

스토어링크 개발3팀 R&D Engineer

강동호, 어떤 사람?

좋은 기술을 계속해서 탐구해나가는 개발자를 추구합니다.

최근에는 주로 Kotlin 언어를 활용한 SpringBoot에 WebFlux를 적용하여 개발하고 있습니다.

강동호, 잘하는 건?

- Spring/Webflux
- Node/TypeScript
- React.js
- Kotlin/Java

강동호, 맡은 역할은?

Back-end Engineering

강동호, 어떻게 연락해?

E-mail : dongho@kookmin.ac.kr

Github : https://github.com/gongdongho12

S. About the Team

최주원

스토어링크 개발3팀 R&D Engineer

최주원, 어떤 사람?

지난 2012년부터 임베디드 시스템에 흥미를 가지게 되어 대학교까지 소프트웨어학부로 진학했습니다.

현재는 Go/Nest.js/Next.js 를 중점으로 개발하고 있습니다.

최주원, 잘하는 건?

- Embedded
- Vue.js/React.js
- Nest.js/Adonis.js
- Go

최주원, 맡은 역할은?

Front-end Engineering

최주원, 어떻게 연락해?

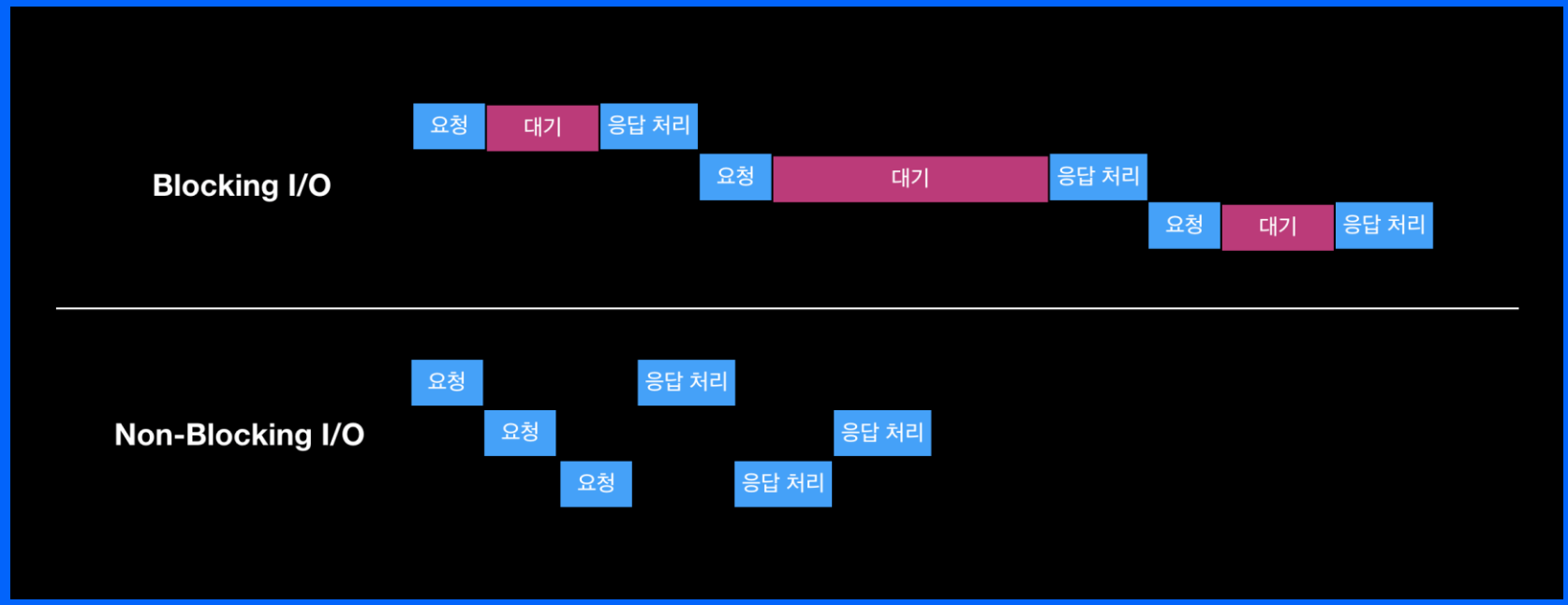
E-mail : cjw980221@kookmin.ac.kr

Github : https://github.com/sch2307

S. Present a Problem

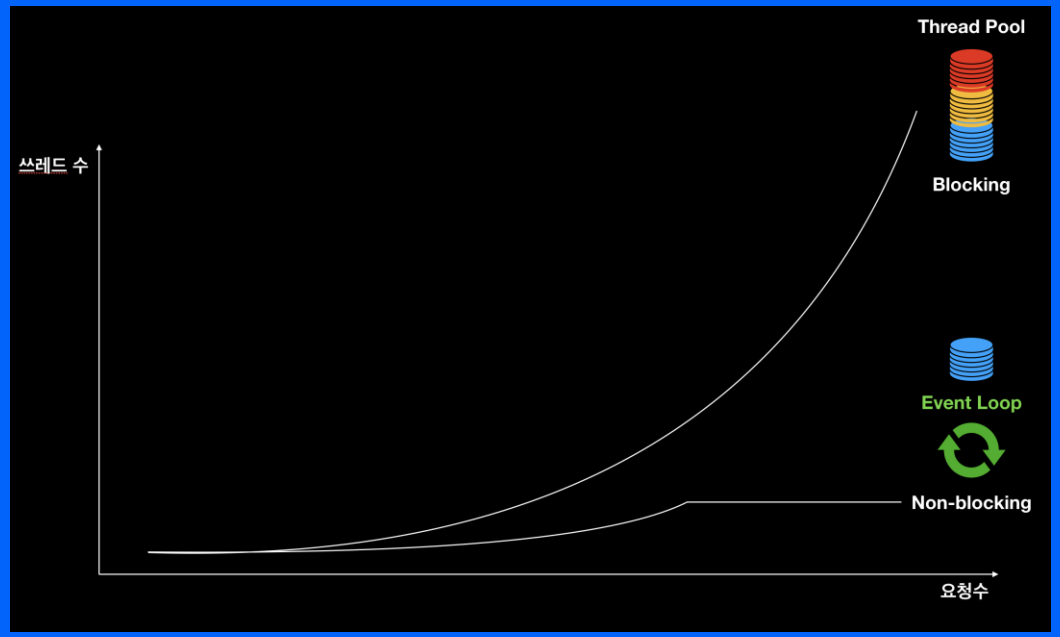
- 스토어링크는 **쇼핑몰 별로 키워드로 검색할 때 나오는 순위**를 관리해주는 스타트업
- 어떤 키워드로 마케팅을 진행해야 하는지 컨설팅 및 **순위 상승을 위한 솔루션** 제공
- 이때 활용 되는 데이터는 **쇼핑몰 별 순위나 리뷰 그리고 매출액**을 **수집**하여 진행
- 수집되는 키워드는 사용자들이 자주 입력하는 상위 10퍼센트 키워드이며 이는 **지속적으로 증가**
- 지속적으로 키워드가 증가하는 **서비스의 수집 안정성 개선 및 러닝 타임 단축**을 위한 방안 연구

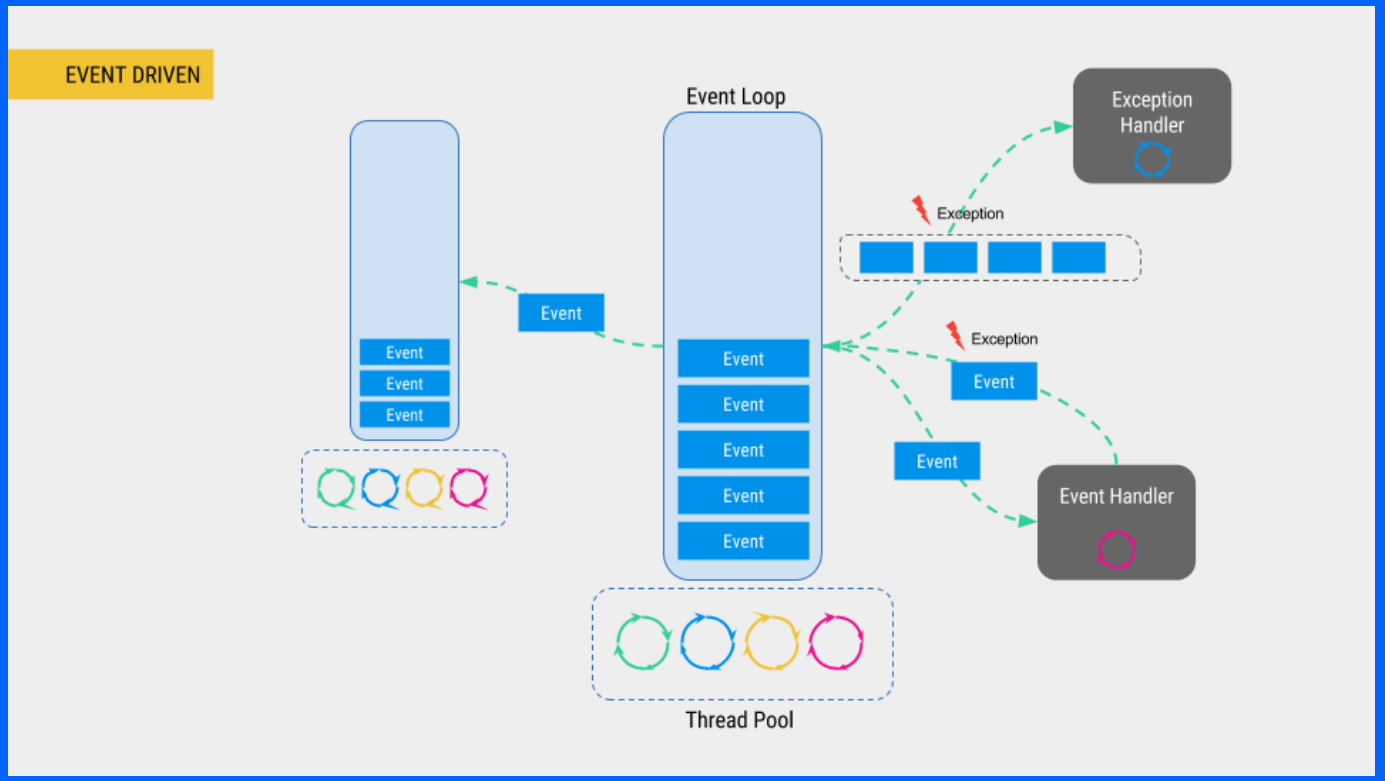
S. Solution 1 - WebFlux



- 기존의 쓰레드 풀 처리 방식의 Blocking I/O에서는 **요청이 끝날 때까지 반드시 대기**
- 네트워크 호출을 미리 해두고 처리를 기다리는 WebFlux에서는 **처리속도가 대폭 감소**
- 만약 요청의 수가 셀 수 없이 많아질 때는 **Back Pressure 기능으로 받아들일 수 있는 양을 조절**

S. Solution 2 - Event Loop

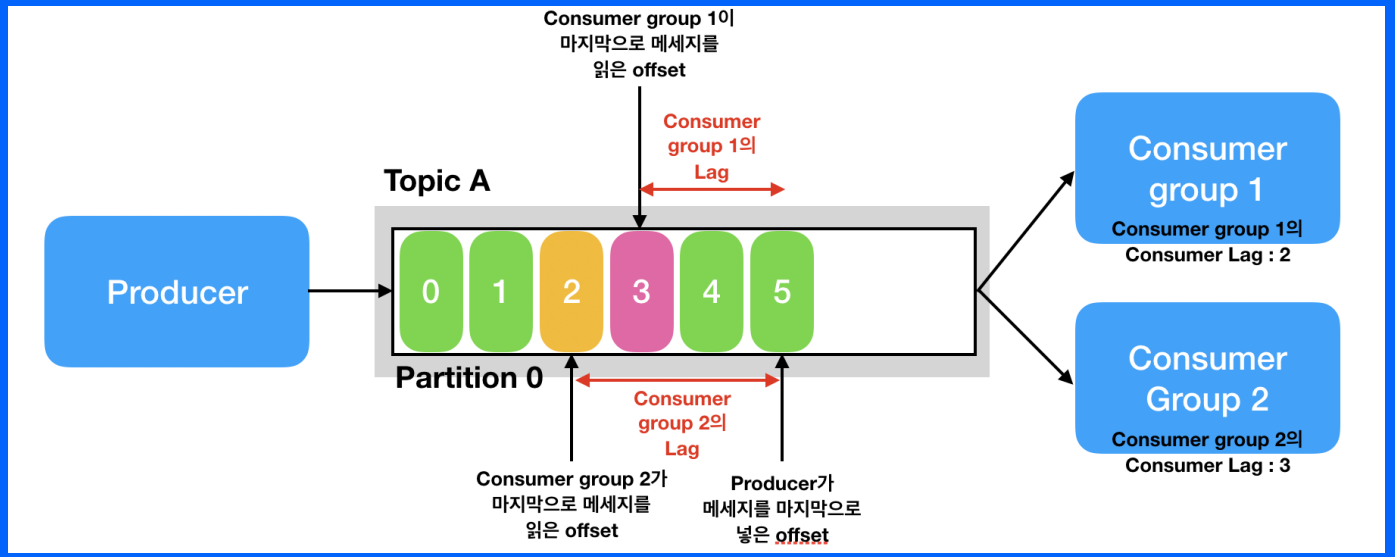




- 스프링에서 멀티 쓰레드를 처리하는 방법에는 **쓰레드 풀 블로킹 방식**과 **이벤트 루프 방식**이 존재
- 각 처리가 끝날때 까지 기다리지 않고 지속적으로 **핸들러를 처리**하여 결과를 리턴
- 처리가 도중에 멈추지 않고 **Netty 를 활용하여 Callback 이 진행되어 병목현상이 일어나지 않음**

S. Solution 3 - Kafka

kafka



- Kafka는 **메시지 브로커로 메시지가 전달되는 서버가 구독하는 방식**의 오픈소스 미들웨어
- **Zoo Keeper에 저장된 데이터를 순차적으로 읽어**와 메시지의 크기가 크더라도 대응할 수 있는 파일형식의 데이터 구조이고 에러가 발생했을 때에도 저장된 파일을 불러와 문제상황에 대응 가능
- Kafka에서 읽어 소비한 부분을 **Offset으로 관리**하여 API의 처리 여부를 구분

S. ObjectMapper Serializer

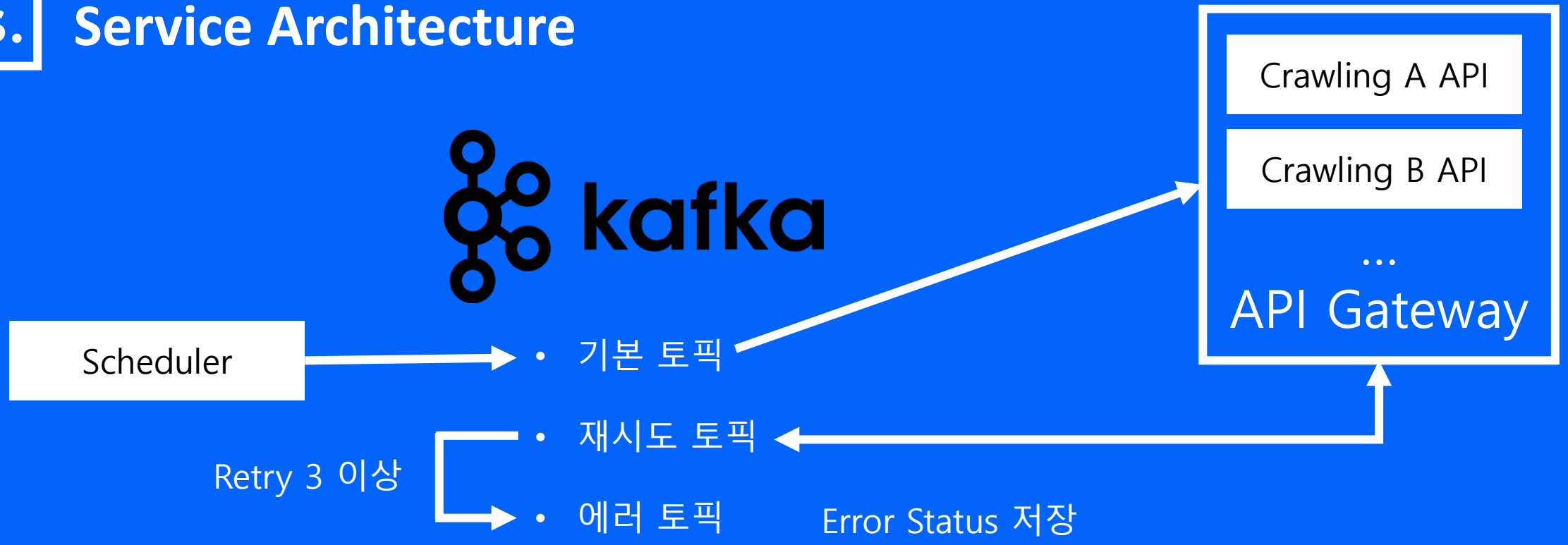
```kotlin
fun objectToString(value: Any) = objectMapper.writeValueAsString(value)
fun stringToObjectByKey(key: String, value: String): Any {
    return when (key) {
        "alert" -> objectMapper.readValue(value, AlertMessage::class.java)
        else -> value
    }
}
```

```kotlin
private fun processReceivedData(): Consumer<ReceiverRecord<String, Any>> {
    return Consumer<ReceiverRecord<String, Any>> { r ->
        val receivedData: Any = r.value()
        if (receivedData != null) {
            val key = r.key()
            val topic = r.topic()
            val value: Any = stringToObjectByKey(key, receivedData as String)
            log.info { "Kafka topic: ${topic} data: ${receivedData}" }

            // data를 consuming할때마다 sink로 전송
            when (topic) {
                "error" -> {
                    consumeErrorAlert(value)
                    sinksErrorMany.emitNext(value, Sinks.EmitFailureHandler.FAIL_FAST)
                }
                else -> {
                    consumeAlert(value)
                    sinksMany.emitNext(value, Sinks.EmitFailureHandler.FAIL_FAST)
                }
            }
        }
        r.receiverOffset().acknowledge()
    }
}
```

- Serialize(직렬화)란, 전송을 목적으로 **객체를 바이트 스트림으로 변환하는 프로세스**
- Kafka 는 기본적으로 **String 데이터**를 저장하고 있음
- 이를 **Jackson ObjectMapper 를 통해 Object로 전환하여 토픽**으로 전송함

S. Service Architecture



- Scheduler 가 기본 토픽으로 API 메시지를 호출하도록 **전달**
- 이후 처리가 정상적으로 넘어가면 끝 그렇지 않다면 **Retry topic 으로 전달하여 재시도**
- 만약 전달한 값이 3번 이상이 된다면 API는 **Error topic 으로 전달**하여 에러를 DB에 기록

S. Progress

- Crawling Running Time

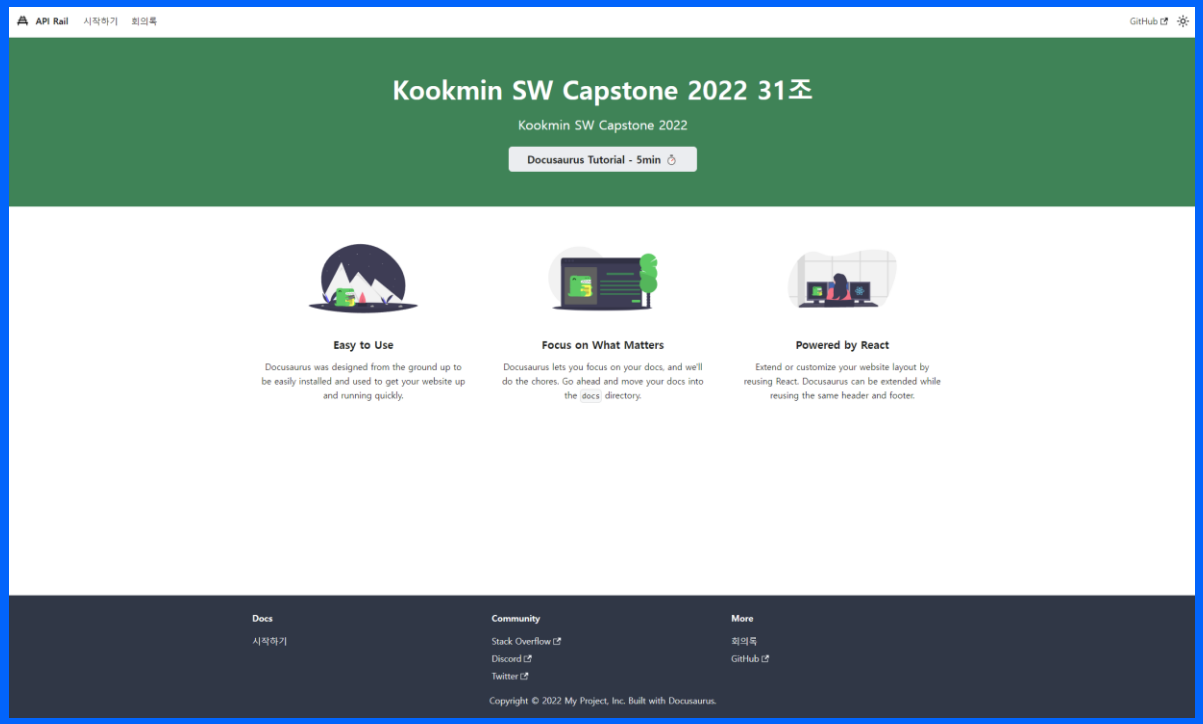
8 Hours

1 Hours

약 84% 개선

S. Progress

- Docusaurus Deployment



https://kookmin-sw.github.io/capstone-2022-31