SOFTWARE MAESTRO

2022년도 「SW마에스트로」 과정 제13기 프로젝트 최종점검 보고서

1. 프로젝트 개요서

프로젝트명	RainMaker		
프로젝트 요약	Github의 데이터를 분석해서 Four Key Metrics를 보여주고, 위험 예측 알람을 보내주는 프로젝트		
기술 키워드	웹서비스, 빅데이터 처리, 데이터 분석, BI, DevOps		
ICT연구개발 기술 분류	융합SW – 응용고도화 SW		
팀 명	알락꼬리여우원숭이	**팀 원**	류동인, 백종현, 조인혁
목적 및 필요성	**개발조직의 수요가 증가하고, 개발자의 몸값이 치솟는 상황에서 개발조직의 퍼포먼스를 향상하는 방식의 HR전략이 대두되고 있음. 이를 위해서 개발조직의 생산성을 측정하는 것이 필수적인데, 한국에서 이를위한 서비스가 명확하지 않음. 추가적으로 BI시장의 성장세가 가파르며, 이는 BI서비스에 대한 수요가 증가하고 사용자들의 접근성이 나아졌다고 판단됨.**		
프로젝트 개요	**Google의 DORA팀에서 발표한 Four Key Metrics를 통해 개발조직의 생산성을 측정・분석해주는 서비스.**		
수행 방법 및 프로젝트 관리	**Cloud(Azure)상에서 데이터 파이프라인을 구축하고, 이 파이프라인으로 사용자의 Github 활동정보를 수집하고, 분석한다. 분석한 결과를 기반으로 대시보드를 구성해서 웹으로 서빙한다.** **JIRA, Slack, Notion, Conflunce, Github를 이용해서 프로젝트관리, 메시징, 문서관리, 소스코드관리 를 수행한다.**		
개발 결과물	**Service URL : www.rainmaker.cool** **Github URL : https://github.com/Ring-tail-lemur/RainMaker**		
기대 효과 및 활용 방안	**개발조직의 생산성 향상을 위한 인사이트를 제공하여, 더 나은 개발조직의 HR전략 수립에 도움을 준다.** **기업의 내부 전략 수립을 위한 도구로서의 활동과 외부 투자자들이 기업의 내실을 확인할 수 있는 지표가 될 수 있다.** **구독서비스를 통해 수익을 창출 할 계획이며, 개인 또는 더 큰 조직이 대시보드를 활용할 수 있도록 하는 수직적 확장, 더 많은 지표를 추가하는 수평적 확장을 계획하고있다.**		

SOFTWARE MAESTRO

【 목 차 】

2. 프로젝트 최종보고서

□ 프로젝트명

ㅇ RainMaker

□ 목적 및 필요성

ㅇ 문제인식

- 개발조직의 생산성을 개선하는 것은 IT시대의 큰 도전과제다. 대부분의 기업들(제조기업)은 생산성을 정량적으로 측정하고 이에 근거한 개선 전략을 수립해서 적은 비용으로 더 큰 가치를 창출해 낼 수 있다. 이는 작업관리, 생산관리, 품질관리와 같은 다양한 분야에 걸쳐서 수행되고 있으며, 과거에 이런 학문이 산업혁명을 주도하기도 했다. 하지만 전통적인 생산조직과는 다르게 개발조직의 생산성을 정량적으로 측정할 수 있는 방법은 최근까지 정립되지 않았다. 예컨데, 각 조직에서 주관적으로 만든 메트릭이나 KPI와 같은 방식으로만 생산성을 측정하고 있는 것이다. 하지만 2014년 Google의 DORA(Devops Reserch and Assessment) 연구팀에서 Four Key Metrics라는 생산성 측정 지표를 발표한 이후로 개발조직에서도 생산성을 측정할 수 있는 발판을 마련하게 된다. DORA 연구팀에서 후속연구를 거듭한 결과 Four Key Metrics는 2022년 3월에 Technology Radar Vol. 26에서 Techniques분야의 가장 기대되는 기술(Adopt, 1번)로 뽑히기도 했다. 이로써 Four Key Metrics는 개발조직의 생산성 측정의 사실상 표준이 되어 활용되고 있다.

  Four Key Metrics는 이미 해외의 빅 IT기업에서 적극적으로 활용하고 있으며, Four Key Metrics를 측정해주는 솔루션이 존재하고 있다. 하지만 이러한 세계적인 흐름에 비해 한국 개발 시장에서는 Four Key Metrics가 적극적으로 사용되지 않고 있다. 우리는 그 이유를 아래와 같이 확인했다.

- 한국 개발시장의 프로세스에 맞지않는 측정방식

  한국의 개발 조직은 아직 Review가 적극적으로 활용되지 않는 경우도 많다. 또한 생산성 측정이 더 나은 조직이 되기위한 FeedBack으로 활용되는 것이 아니라, 하위조직을 평가하는 용도로 사용될 위험성도 있다.

  몇몇 서비스들은 사용자 데이터를 받아오기위해 연동하는 서비스들이 외국에서 주로 사용되는 툴들 이다. 예컨데 ShortCut/CircleCI/Slack와 같은 툴을 주로 지원하는 반면, Kakao/Travis/GitHub Action와 같은 툴은 지원하고 있지 않다.

- 인지도 부족

Four Key Metrics는 2014년에 처음으로 소개되었으며, 매년 후속 연구를 진행한 끝에 2022년에 되어서야 기술트렌드 중 하나로 자리잡게 되었다. 하지만 아직 한국에서 적극적으로 소개된 적이 없어서 국내에서는 낮은 인지도를 가진 기술이다.

- 비싼 비용

외국의 Four Key Metrics 솔루션은 평균적으로 1인당 한달에 $20~$40의 비용을 지불해야 한다. 특히 몇몇 업체를 제외하면, 무료버전 가입 절차가 매우 까다롭다.

- 너무 많은 초기설정 소요

완전한 지표를 얻어내기 위해서 GitHub, Jira, Jenkins, Slack등 수 많은 외부 서비스와 연동해야 하며, 연동되지 않은 외부 서비스가 있는 경우에는 특정 지표는 표시되지 않는다.

- 그래프로부터 해석의 난해함

Four Key Metrics는 실험을 통해 얻어진 지표이다. 따라서 각 지표가 개발 조직의 퍼포먼스와 인과관계가 있다는 사실은 알 수 있지만, 세부적으로 지표를 분석해서 원인을 파악하고 전략을 수립하기에는 직관적이지 않은 수치들이다.

○ 기획 의도(문제해결)

- 외부 서비스와의 연동을 최소화하고, 더 좋은 지표를 원하는 사용자가 외부 서비스 추가 연동

우리 서비스는 사용자 조직의 퍼포먼스를 측정하기 위해서 3가지 종류의 이벤트를 수신해야한다. [1]Git Repository , Issue Management System, CI/CD Tool에서 발생하는 이벤트들이다. 이때 우리는 필수적으로 연동해야하는 외부 서비스를 Git Repository로 한정하고, Git Repository에서 발생하는 이벤트만으로 나머지 이벤트들의 발생을 추론해서 지표를 보여 준다. 여기서 더 정확한 지표를 얻고 싶다면 추가적인 연동을 할 수 있도록 해준다.

1) GitHub, GitLab, BitBuket 등이 있다.

- 한국에서 주로 사용하는 외부 서비스와의 연동

Kakao, 잔디, Slack, E-Mail을 메시지 툴로 사용하고, GitHub, GitLab, BitBuket을 VCS 툴로, Jira, flow를 Issue Management System으로, GitHub Action, Jenkins를 CI/CD툴로 연동할 수 있도록 한다.

- 팀 단위의 분석만 제공

개인을 평가하거나 비난하는 지표로 사용되지 않도록 팀의 모든 이벤트를 대상으로 분석한다.

- 기술 소개 활동

기술 블로그를 운영하고, 사용 안내, 컨퍼런스 발표 등을 통해서 DORA Metrics과 DevOps 측정의 필요성을 소개한다.

- 친절한 분석 결과 제공

Four Key Metrics이 익숙하지 않은 한국의 개발조직은 4가지의 지표만으로 어떤 개선사항을 진단하거나 전략을 수립하기 힘들다. 또한 Four Key Metrics 자체가 인사이트를 얻기에 어려운 지표이기도 하다. 따라서 각 지표를 더 쪼개서 자세하게 표현해줄 필요가 있다. Four Key Metrics의 변경 리드타임과 평균 회복 시간은 Break Down를 분석하거나, Event 기반으로 데이터를 정렬해서 제시하고, 배포 빈도와 변경 실패율을 시간의 흐름에 따라 배포를 나열한다. 이로서 어떤 PR이, 어떤 배포가, 어떤 실패가 유독 시간을 많이 소모하게 되는지 확인 할 수 있으며, 각 지표를 구성하고 있는 세부적인 내용을 파악할 수 있게 된다. 이는 사용자가 개발조직의 생산성을 재고하기 위한 전략을 세우기 용이하도록 도와준다.

- 높은 신뢰성

개발조직은 다양한 방법으로 개발업무를 수행하고 있다. 또한 Git에 포함되어있는 다양한 기능들을 활용하는 프로젝트는 높은 복잡성 때문에 데이터를 수집하고, 사용자의 활동을 추적하며, 데이터를 분석하기 어려운 환경이 된다. 우리는 이런 복잡한 상황에서도 정확한 지표를 뽑아낼 수 있도록 데이터 구조를 설계했다.

○ 시장/소비자 동향/분석, 경쟁 제품 비교/차별화 요소

- 시장/소비자 동향/분석

우리의 서비스는 개발팀이 생성하는 수많은 데이터를 기반으로 인사이트를 제공하는 이벤트 기반 빅데이터 분석 및 [2]BI 플렛폼이다. IDC의 2020년 데이터에 따르면 전 세계 전체 비즈니스 인텔리전스 및 분석 시장은 192억 달러에 달했으며, 팬데믹 관련 경제 충격에도 불구하고 5.2%나 성장했다. 앞으로 기업들이 미래의 비즈니스 발전을 위해 데이터를 사용하는 디지털 전환과 더 스마트한 방법에 집중하면서 BI 성장이 가속화될 것으로 예상된다. 또한 수많은 기업이 IT열풍에 힘입어 디지털 트렌스포메이션을 감행하면서 좋은 개발 인력 확보를 위한 연봉 인상 경쟁이 본격화 된지 1년이 지났다. 그 사이 개발자 몸값은 천정부지로 뛰었는데, 이는 기업 입장에서 인건비 부담이 증가하고 수익성 악화까지 걱정해야하는 상황이 되었다. 기업은 이를 해소하기 위한 전략이 필수적으로 필요해 진 것이다. 지난 2월 중소벤처기업부는 2025년까지 부족한 개발 인력이 최소 4만 명에 이를 것이라는 전망을 내놨다. 하지만 개발자 공급치는 이에 따라오지 못하고 있고, 이제 기업들은 개발자들의 업무 효율과 생산성을 높일 방법을 고민하는 HR 전략을 수립하기 시작했다. 개발 조직의 생산성을 제고하고, 위험을 관리하는 저희 서비스는 이러한 흐름의 정 중앙을 겨냥하고 있다.

- 경쟁 제품 비교/차별화 요소

- Atlassian(Jira)
    - Jira의 보고서항목이 팀의 성과를 분석해서 제공한다. 하지만 백로그, 스프린트와 같이 issue management system의 데이터를 기준으로 분석하는 방식으로, VCS, issue management system, CI/CD tool에서 발생하는 이벤트를 기반으로 분석하는 DORA Metrics보다 세부적이고 명확한 인사이트를 제공하지 못한다.
- Azure DevOps Service
    - Jira와 같은프로젝트 관리 서비스를 제공하지만, 분석보다는 다양한 툴(Cloud, Git 등)의 통합성에 초점을 맞추고 있다.
- GoogleCloud DevOps Solutions
    - Fourkeys : DORA Metrics를 측정해서 시각화해주는 DashBoard (결과를 분석해서 인사이트를 제공하지 않음)
    - 팀이 담당하는 서비스의 자동화, CD/CI 등 기술적인 문제에 대한 솔루션을 제공
- LinearB
    - 각 지표를 측정하는 방식이 연구에서 주장하는 내용이나 일반적인 상식과 같이 작동하지 않는다. 예컨데, GitHub에서 여러 pull

2) Business Intelligence 기업에서 데이터를 수집, 정리, 분석하고 활용하여 효율적인 의사결정을 할 수 있는 방법

request를 거쳐서 main에 병합된 경우에는 Lead Time For Change를 측정하지 않는 식이다.
    - 번아웃이나, 한 번에 너무 많은 코드를 커밋하는 것 등 팀의 안정성에 대한분석은없다.
    - 특정 외부 솔루션을 연동하지 않으면 표지되지 않는 지표나 그래프가 많이 존재한다.
    - 연동과정이 번거럽고 많은 시간을 소모시킨다.
- HayStack
    - 인사 관리를목적으로 하는 전체 솔루션의 일부로 Dora Metrics를 차용하는 형태이다. 따라서 단지 팀의 상태를 시각화 하는 것을 목적으로 한다.
    - 무료버전을 사용할 수가 없다. 따라서 진입장벽이 높다.
- 차별점
    - 외부 서비스와의 연동을 최소화하고, 더 좋은 지표를 원하는 사용자는 외부 서비스 추가 연동
    - 한국에서 주로 사용하는 외부 서비스와의 연동
    - 팀 단위의 분석만 제공
    - 기술 소개 활동
    - 필수적인 기능만 함축적으로 만들고, 낮은 구독 비용 유지

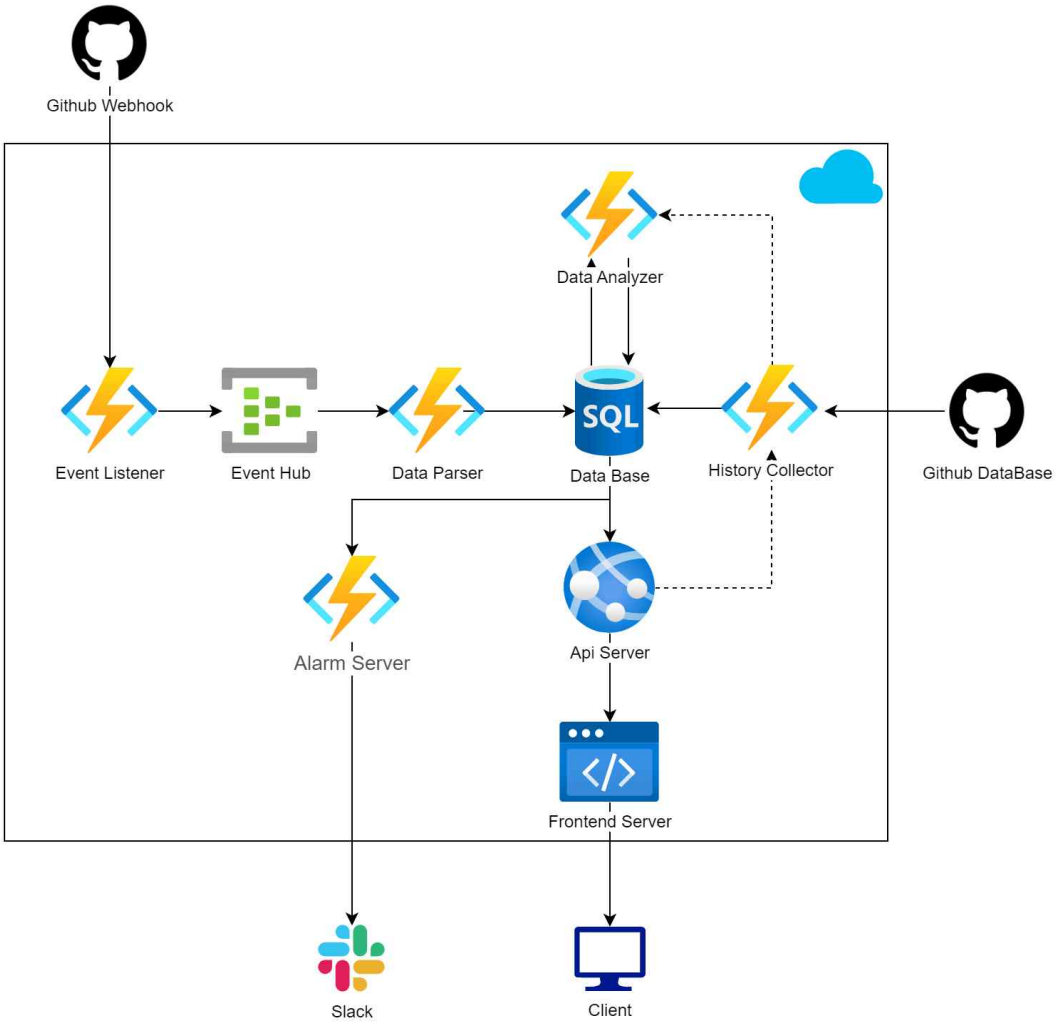
□ 프로젝트 개요

ㅇ 프로젝트 소개

- Git Repository의 데이터와, Git Repository에서 발생하는 이벤트 이용해서 Four Key Metrics를 계산하고, 분석해서 사용자에게 보여주는 프로젝트이다. 또한 Git Repository의 이벤트를 분석해서 다양한 위험 감지 알람을 보내주기도 한다.

ㅇ 시스템 구성도

* 실선 : 데이터의 이동
* 점선 : 요청을 보냄(Http Request를 통한 Triggering)



- Event Listener(Azure Function - Javascript: Azure의 Serverless Funtion)

Github에서 사용자의 활동을 추적하기 위해서 Github Webhook을 받아서 Event Hub로 넘겨주는 역할을 한다. 이때 필요하지 않은 정보를 탈락시키고, 모든 이벤트를 같은 포멧으로 변환하는 일을 수행한다.

- Event Hub(Azure EventHub: Azure의 이벤트 처리용 Kafka)

Kafa기반의 Event Hub로 이벤트를 큐잉해서 고가용성과 안정성을 보장한다. Event Hub에 큐잉된 데이터들은 정해진 시간이 지나거나, 들어온 데이터의 개수가 정해진 양을 넘기면 Data Parser로 전달된다. 이때 들어온 데이터를 Batch로 처리하기 때문에 성능 향상을 기대할 수 있다.

- Data Parser(Azure Function - Javascript: Azure의 Serverless Funtion)

Event Hub를 통해 전달받은 데이터를 DB에 정의해 둔 Table의 형태로 변

환해서 DB에 삽입한다. 이때 DB에 삽입하는 로직은 Batch Insert로 수행한다.

- Data Base(Azure SQL: Azure용 SQL-Server)

SQL-Server는 OLAP와 OLTP를 동시에 제공한다. 우리는 이러한 특징을 이용해서 하나의 Azure SQL의 일반 테이블에 데이터를 Insert하고, 이를 Column Store방식의 OLAP모드 테이블로 옮겨서 분석 쿼리를 수행한다.

Data Base에 데이터를 삽입하는 모듈은 Parser와 History Collector, Data Analyzer가 있고, 데이터를 분석하는 모듈은 Data Analyzer, 데이터를 읽어가는 모듈은 Api Server, Alarm Server가 있다.

- Data Analyzer(Azure Function – Java: Azure의 Serverless Function)

Data Analyzer는 3분마다 한번 씩 데이터 분석을 처리하는 모듈이다. 이때 DB에 있는 데이터와 함께 Github Rest Api를 통해서 추가적인 정보를 받아와서 데이터 분석을 수행한다. 받아온 데이터는 Data Base에 저장된다.

- Api Server(Azure App Service – Spring Boot)

Data Base에 저장되어있는 분석이 완료된 데이터를 Frontend Server로 전달해주는 API서버이다.

- Frontend Server(Azure Static Web App – Vue.js)

사용자 화면을 그려주는 Frontend Server이다.

- Histroy Collector(Azure Function – Java: Azure의 Serverless Function)

RainMaker서비스는 최초 회원가입 시점에서 사용자에 대한 데이터가 전혀 없어서 분석을 수행할 수 없다. 따라서 회원가입한 사용자의 과거 이력을 읽어 오는 작업이 필수적인데, 이를 수행하기 위한 모듈이다. History Collector는 성능과 확장성에 중심을 두고 구현했다. 사용자의 Github 과거 이력을 가져오기까지 사용자는 대기해야 하므로, 병렬 프로그래밍과 Bulk연산을 활용해서 성능을 극대화 시켰고, 서비스가 확장될 때 가장 빈번하게 변경이 일어나는 부분으로, 코드의 수정 없이 새로운 작업을 추가하고 수정할 수 있도록 했다.

History Collector는 Github DataBase에서 GraphQL, RestApi등을 이용해서 데이터를 받아오고, 이를 DB에 삽입한 후 Data Analyzer를 호출함으로서 데

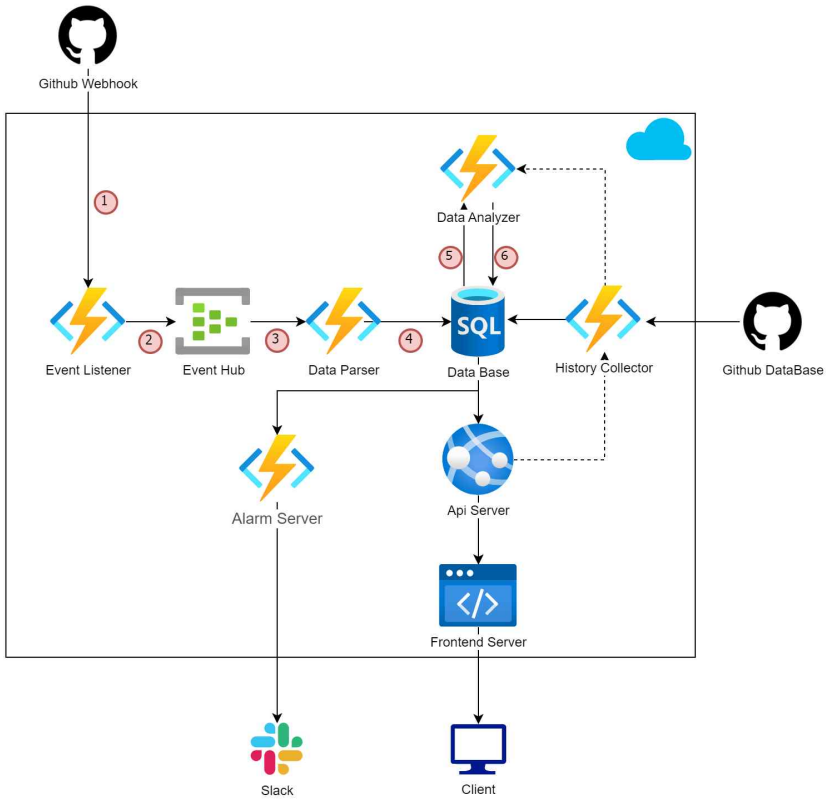
이터 분석까지 수행되도록 한다.

- **Alarm Server(Azure Function – Python: Azure의 Serverless Function)**

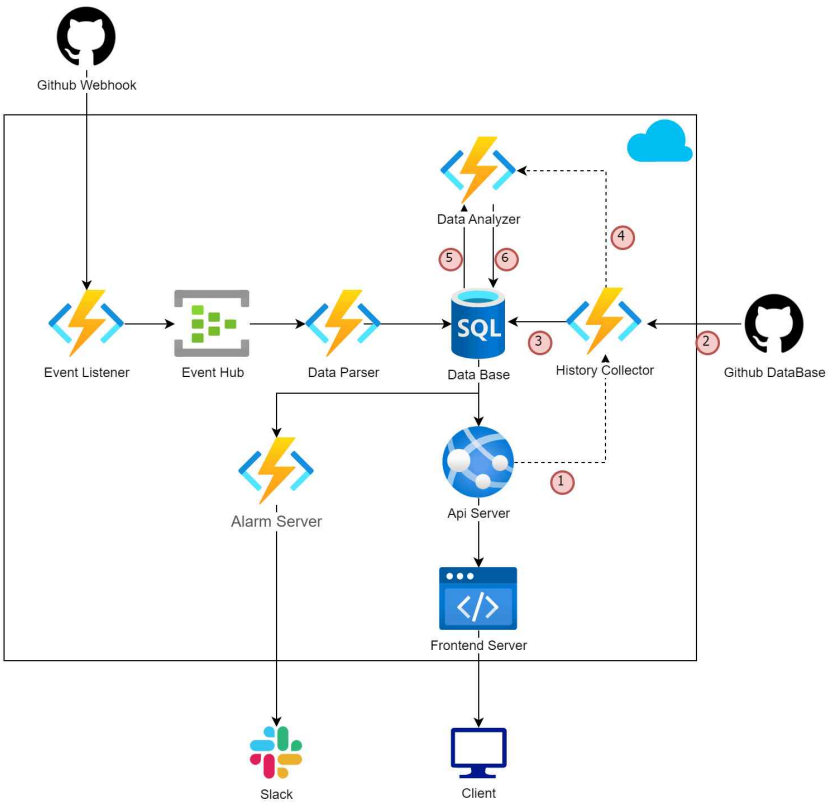
Alarm Server는 1분마다 DB를 확인해서 사용자의 위험을 감지하고, 감지된 위험에 따라서 Slack으로 사용자에게 알림을 보낸다.

ㅇ 주요 기능

- Github에서 발생한 이벤트를 분석해서 Four Key Metrics와 세부사항 DB에 적재할 수 있다.



- 사용자의 과거 Github 활동 이력을 최근 100개의 PR을 기준으로 받아와서 분석 후 DB에 저장할 수 있다.



- 사용자는 대시보드에서 Four Key metrics와 세부사항을 조회할 수 있다.
- 사용자의 활동 이력 중에 위험이 감지되는 경우(Burn Out 위험 등) 알람을 수 있다.

○ 개발 환경

구분		상세내용
S/W 개발환경	OS	Ubuntu, Window, MacOS
	개발환경(IDE)	Intellij, WebStorm, Visual Studio Code, DataGrip, PyCharm
	개발도구	Azure(Cloud)
	개발언어	Python(Azure Function), Java(Azure Function), Javascript(Azure Function), Java(Spring Boot), Javascript(Vue.js, char.js), T-SQL
	기타사항	.
H/W 구성장비	디바이스	.
	센서	.
	통신	.
	언어	.
	기타사항	.
프로젝트 관리환경	형상관리	Git
	의사소통관리	Notion, Slack, Kakao
	기타사항	Jira, Confluence

□ 수행 방법 및 프로젝트 관리

○ 주요 기능별 수행방법

지표 추출	- Github Webhook, Github DataBase를 활용해서 사용자 데이터를 수집하고, Azure Function을 통해서 데이터를 분석하므로서 Four Key Metrics를 데이터를 만듬
알림 전송	- Azure Function을 이용해서 1분마다 DB를 조회하고 조회한 결과에서 사용자의 위험을 감지해서 알림 전송
UX/UI 디자인	- 템플릿을 이용해서 기본적인 프론트 화면을 구성, Char.js를 통해서 사용성 향상 - 디자인 외주를 통해 가시성 향상

○ 수행방법 확보방안(아웃소싱 등)

- 지표 추출 : 협업 시 발생하는 이벤트들을 모아주기 위해 Big Data Pipelining을 통하여 DB에 이벤트들을 확보한다. DB에서 실시간으로 끊임없이 발생하는 다양한 데이터를 모으기 위해, Azure에서 제공하는 서비스들을 통해 Big Data에 대응할 수 있도록 아키텍쳐를 설계한다. 또한 데이터를 모아줄 DB 또한 Azure에서 제공하는 Data Warehouse 서비스인 Azure MS Sql을 활용하여 무결성과 정합성을 확보할 수 있도록 설계하여 단단한 시스템

을 통해 지표를 추출할 수 있도록 구성하였다.

- 알림 전송 : Serverless Function을 통하여, 위험 상황 발생을 지속적으로 추적하고 이를 통하여 사용자들에게 알람을 보내줄 수 있도록 구성하였다.

- UX/UI 디자인 : 좋은 지표를 보여주기 위해, 오픈소스 라이브러리 Chart.js를 통하여 그래프를 통하여 사용자들에게 정보를 제공하도록 하였다. 또한 완성도 높은 디자인을 위해서, 외부 템플릿을 사용하여 서비스들을 사용할 수 있도록 만들 예정이다.

○ 추진 일정

기능명	구분	추진 내용	진척도	추진 일정					
				6월	7월	8월	9월	10월	11월
지표 추출	계획	지표추출을 위한 계획	100%	■					
	분석	웹훅 분석	100%		■	■			
	설계	추출을 위한 아키텍쳐 및 DB 설계	100%	■	■	■			
	개발	지표 추출을 위한 로직 개발	100%			■	■	■	
	테스트	실제로 들어오는 웹훅을 통한 테스트	100%			■	■	■	
	종료	사용자가 사용할 지표 추출 완료	100%			■	■	■	
알림 전송	계획	알림 전송을 위한 계획	100%				■		
	분석	알림을 보내기 위한 정보 분석	80%				■	■	
	설계	알림을 위한 아키텍쳐 및 DB설계	100%				■	■	
	개발	알림 전송을 위한 로직 개발	60%					■	■
	테스트	실제로 들어오는 웹훅을 통한 테스트	80%					■	■
	종료	사용자에게 의미있는 알림을 전달	40%						■
UI/UX 디자인	계획	편리한 디자인을 위한 계획	100%	■					
	분석	UI/UX를 어떻게 구성할지 분석	100%		■	■	■		
	설계	디자인 설계	100%			■	■		
	개발	FrontEnd의 로직 개발	100%				■	■	
	테스트	저장된 데이터를 통해 테스트	100%				■	■	
	종료	사용자에게 의미있는 UI를 전달	80%					■	■

○ 역할 분담

담 당		역할 및 상세활동
연수생	류동인	Data Listener 개발 // C10K를 고려한 빅 데이터 수집 Pipeline 구축 // CI/CD pipeline 구축
	조인혁	OLAP 데이터베이스 구축 및 관리 // DB설계 및 구축 // OLAP특성을 이용한 데이터 분석 모듈 개발 // History Collector 모듈 개발 // BackEnd API Server 최적화 및 개발 리딩 // FrontEnd Server 개발
	백종현	BigData Pipeline에서 발생할 수 있는 Data Duplication 해결 // Data Parer 개발 // BackEnd API Server 구축 및 개발 // FrontEnd Server 개발 리딩
멘토	이다니엘	Project 개발 방법론, Lean Product Development, Product 방향에 대한 조언
	최정현	Agile 방법론, 프로젝트 계획 및 아이템 전반에 대한 조언
	유저스틴	Azure 기반 Architecture 관련 지식 전수 및 조언

○ 기술 습득 노력

담 당		기술 습득 노력
연수생	류동인	10K 문제 해결을 위한 CloudEvents(OpenSource) + Kafka 습득 // Serveless Archtecture에 관한 지식 습득 // Github Action 사용법 습득
	백종현	Mssql 사용법 및 최적화 공부, Data ETL 방식 및 Big traffic에 대응할 수 있는 방법들을 멘토링과 서칭을 통해 공부, 더 좋은 코드 개발을 위해 강의와 책을 통해 지속적으로 공부
	조인혁	mssql 사용법 및 최적화 공부, Event Driven Architecture / Big Data Analysis Architecture공부, 타 연수생들과 스프링 스터디 진행, Vue.js 강의 수강

○ 문제점 및 해결방안

구 분	문제점 및 해결방안
관리 측면	- 동떨어진 파트를 담당하는 팀원끼리 의사불통 프로젝트 초기에 우리는 3명이 각각 다른 파트의 모듈을 개발했다. 데이터 파이프라인 처리르는 특성상 데이터가 직렬적으로 흘러가는 형태의 프로그램이 개발되고 있었고, Event Listener – Event Hub – Event Parser – Event Analizer의 양 끝단에 있는 Event Listener를 개발하는 인원과 Event Analizer를 개발하는 인원이 의사소통이 적절히 되지 않은 것이다. 언듯 보기에 의사소통이 조금 적더라도 문제가 없을 것으로 보였지만, 둘 다 DB의 변경과 데이터 처리 포멧의 영향을 함께 받았고 이는 우리의 개발 어렵게 만들었다. 이를 해결하기 위해서 JIRA를 활용한 데일리 스크럼, Slack활용 방안을 적극적으로 변경, Github에서 진행되는 코드 리뷰 이외에도 2주에 한번 서로의 코드를 같이 읽는 방식으로 팀의 유대감을 증진하고 협업 능률을 개선할 수 있었다.
개발 측면	- History Collector의 성능문제 History Collector는 사용자가 회원가입 하는 시점에 Github에서 사용자의 과거 활동이력을 받아오는 모듈이다. 사용자는 이 모듈의 작업이 완료될 때 까지 기다려야하기 때문에 모듈을 실행하는 오랜 시간이 걸려서는 안된다. 최초에 우리는 Github Rest Api를 통해 데이터를 받아오도록 설계했는데, RestApi 특성상 Github이 제공하는 데이터를 모두 받아와야 하므로 통신해야하는 데이터의 사이즈가 커서 성능이 나쁘기도 했고, 무엇보다 원하는 정보를 받아오기위해 RestApi를 여러번 호출해야하는 문제가 있었다. 이를 해결하기 위해서 GrphQL를 이용해서 데이터의 크기와 호출 횟수를 획기적으로 줄일 수 있었고, 병렬 프로그래밍을 도입해서 큰 성능 향상을 이룰 수 있었다. - History Collector의 확장성있는 설계 History Collector는 데이터 파이프라인의 가장 유명한 디자인 패턴인 ETL형태를 취하고 있다. 한 모듈이 이 모든 파트를 수행해야 했기 때문에 복잡하고 수정이 어려운 문제가 있었다. 그런데 프로젝트의 수행이 기본적이 기능을 구현하고 병렬적으로 기능을 추가해가는 방식으로 진행되었기 때문에 수정과 확장이 계속해서 발생할 것이 불보듯 뻔했다. 따라서 우리는 새로운 데이터 파이프라인을 추가하는 것을 코드의 수정 없이 할 수 있도록 했다. Config 파일을 외부에서 주입해주면 Config파일을 따라서 데이터를 가져오고, 변환하고, 적재하는 것 까지 모두 통제 할 수 있도록 한 것이다. 이후로 필요한 데이터가 최초 설계보다 2배 많은 가짓수로 늘어났을 때 도 코드를 한줄도 수정하지 않고 확장할 수 있었다.

□ 개발 결과물

ㅇ 결과물 형태 및 서비스 방식

- 결과물 형태
  - Web Application

- 서비스 방식
  - 기술 블로그 운영을 통해 서비스에 쉽게 접근할 수 있도록 구성
  - 서비스를 처음에는 무료로 사용할 수 있도록 구성하고, 더 많은 지표 제공이나 팀 선택을 필요로 한다면 매월 돈을 납부하는 구독 방식을 채용
  - 사용자가 서비스에 Github 저장소에 접근 가능한 권한이 부여된 Github 토큰을 제공하고, 이를 통해 사용자의 데이터를 불러옴
  - 사용자의 데이터를 서빙/분석하여 지표를 추출하고, 추출한 데이터를 Web Application의 다양한 그래프와 표를 통해 제공
  - 사용자가 Slack WebHook URL을 등록하면, 등록한 채팅서버에 실시간으로 지표와 알림을 제공 받을 수 있다.

○ 프로젝트 주요 결과물

- 팀의 생산성 지표 Dora Metrics를 도출
- 개발 진행 상황을 분석하여 Dora Metrics의 세부적인 지표를 추출
- 회사/팀의 일원들의 Burnout과 같은 위험성을 감지하여, Slack 메시지를 통해 위험 요소 탐지 알림을 전달

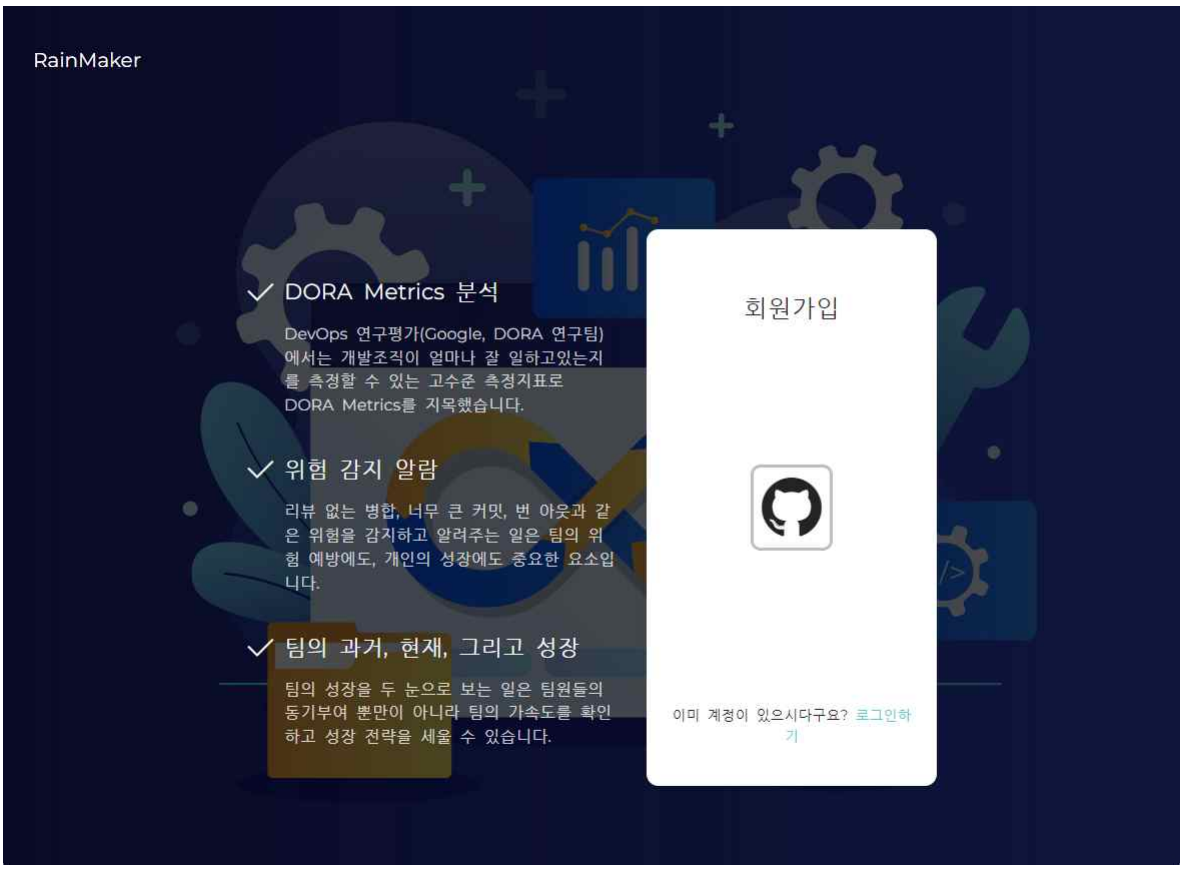
○ 프로젝트 세부 산출물(개발물, 제작물)

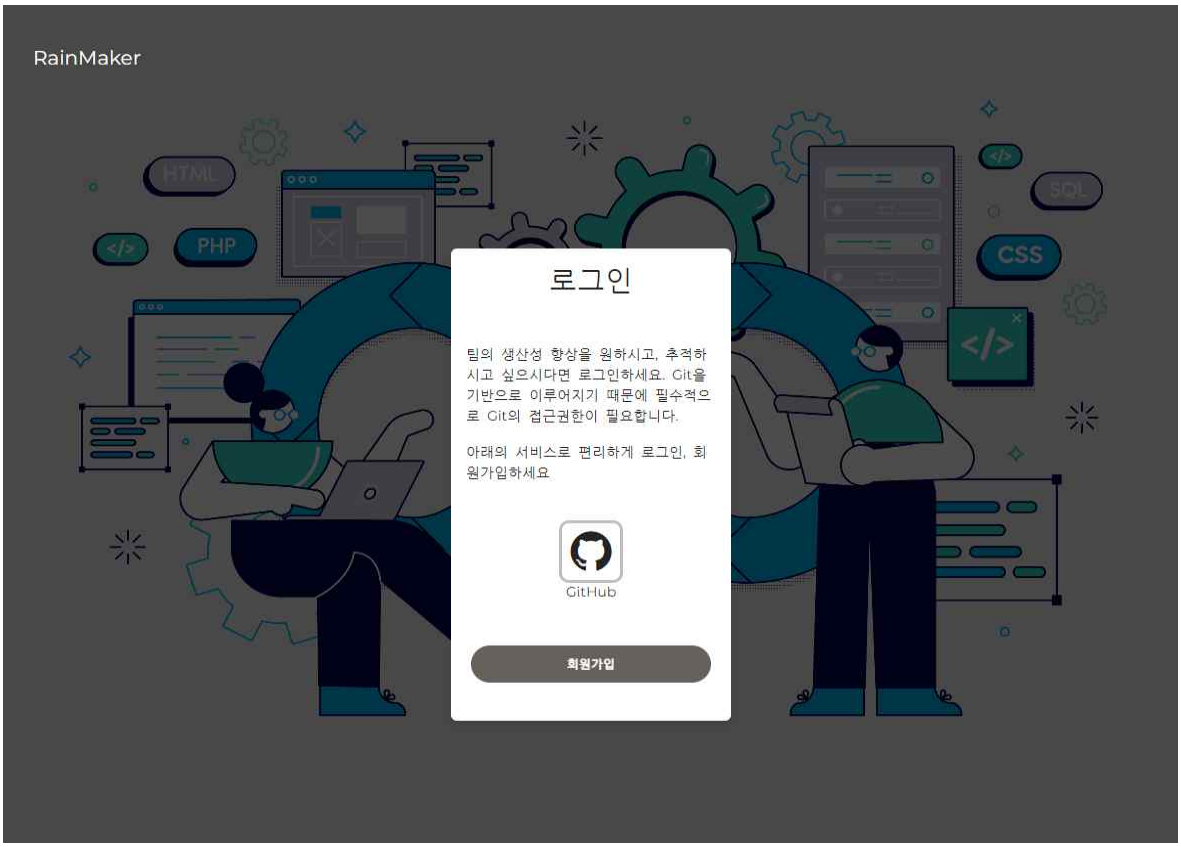
- 메인페이지

메인페이지에서 제품에 관련된 개괄적인 설명을 전달받을 수 있으며, 관련 블로그를 통해 서비스에서 제공해주는 지표들의 자세한 설명과 활용 방안에

대해 알 수 있다.

URL : https://www.rainmaker.cool/main



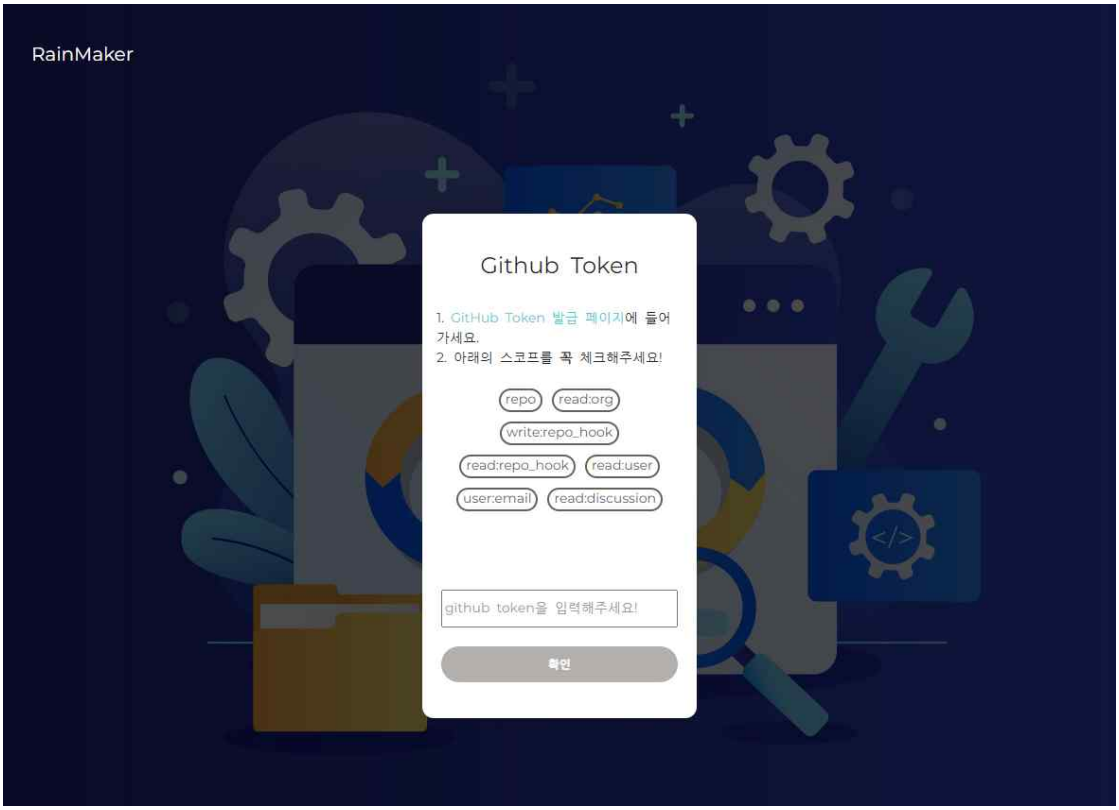


- 로그인, 회원가입 페이지

로그인, 회원가입 페이지에서 Github OAuth를 이용해 간편하게 서비스를 사용할 수 있다.

회원 가입 URL : https://www.rainmaker.cool/register

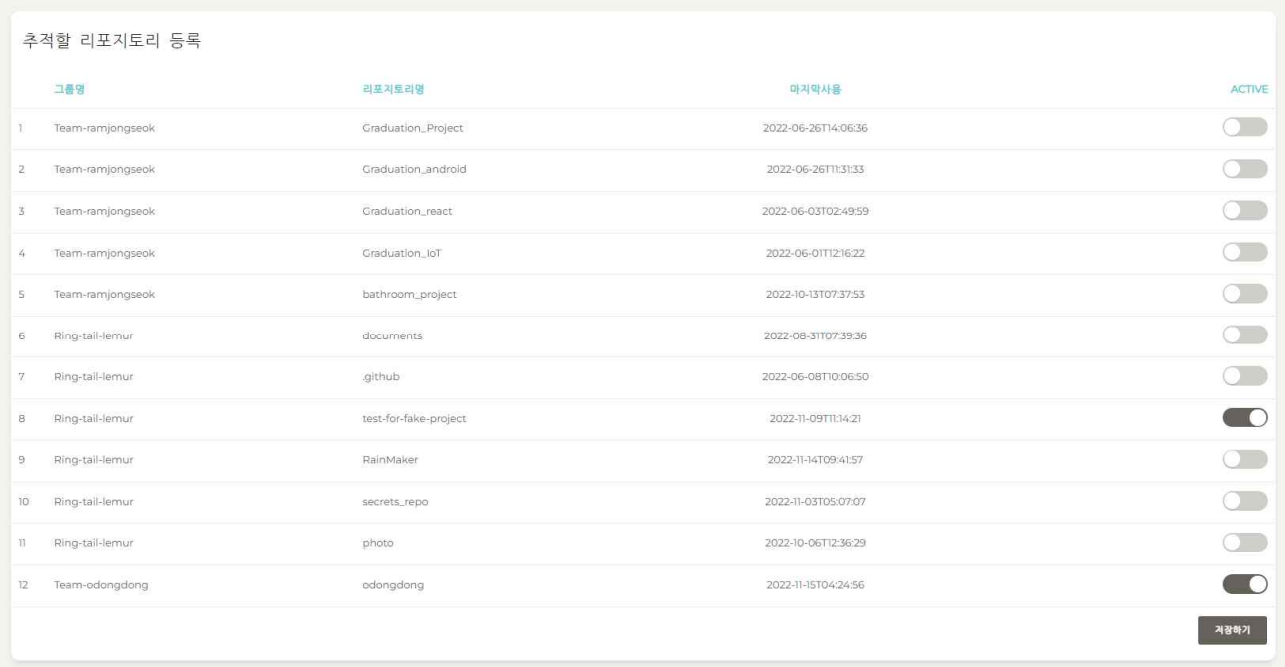
로그인 URL : https://www.rainmaker.cool/login



- Token 등록 페이지

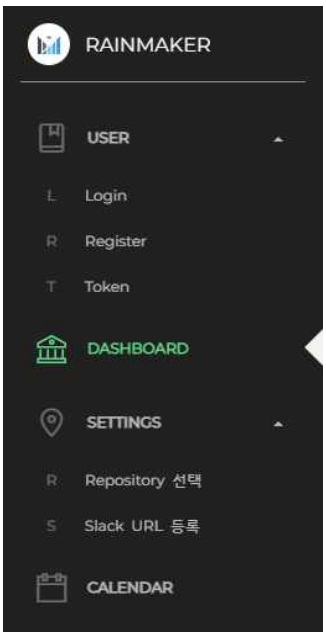
서비스를 사용하기 위해서 Github의 데이터가 필요하며, 추적하고 싶은 데이터를 위해서는 반드시 Github Token 등록을 해야한다. 등록할 Token의 접근권한에 대해서 페이지의 스코프를 통해 확인할 수 있다.

URL : https://www.rainmaker.cool/register/token

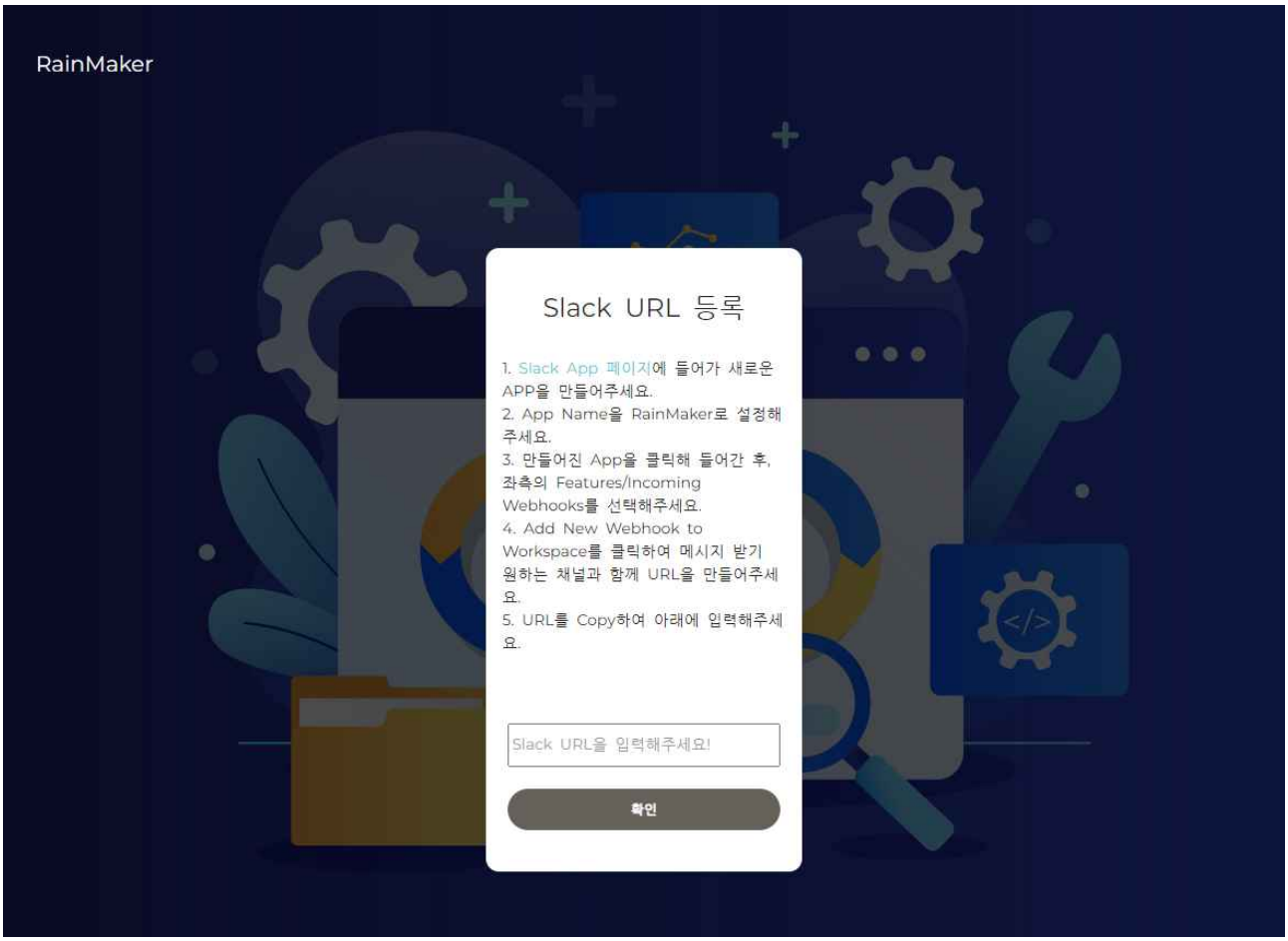


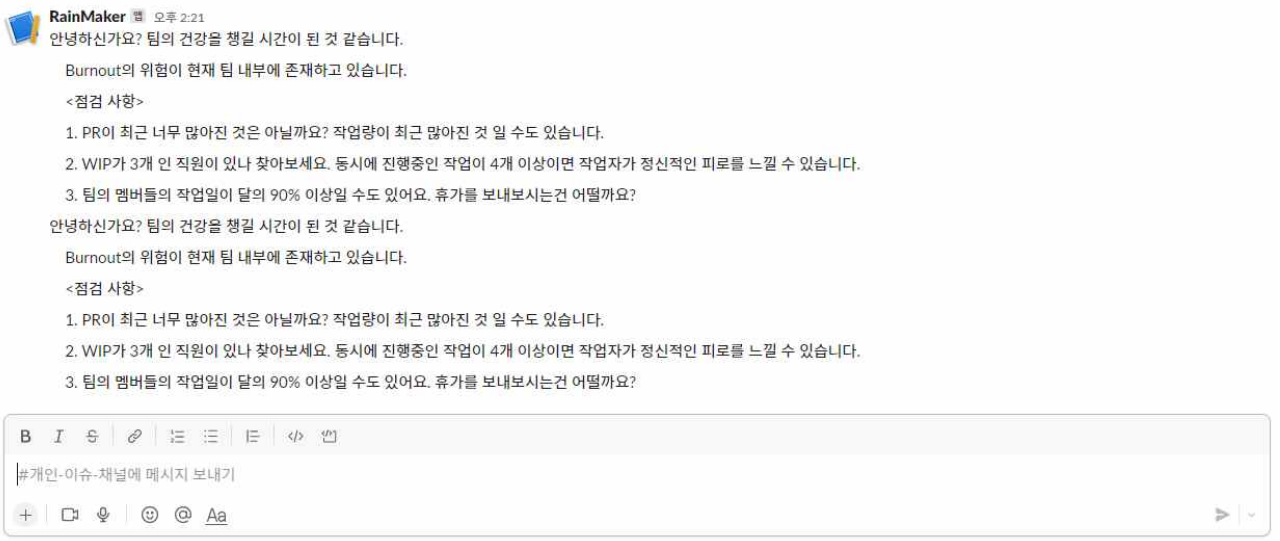
- Github 리포지토리 등록 페이지

RainMaker 서비스에서 추적할 리포지토리를 등록할 수 있다. 원하는 리포지토리를 선택하고 저장하게 되면, 관련 데이터를 수집과 Github Webhook 등록을 자동으로 수행하게 된다.



기본적으로 설정한 내용들을 좌측 사이드바를 통해 변경이 가능하다.

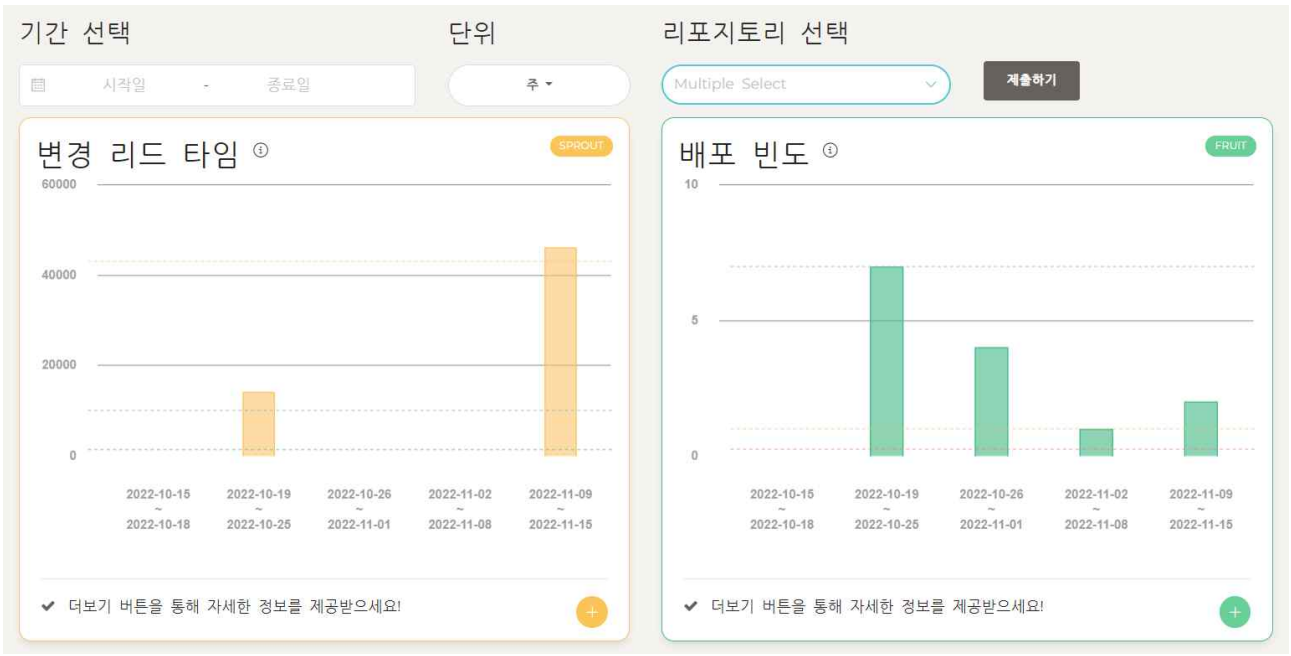


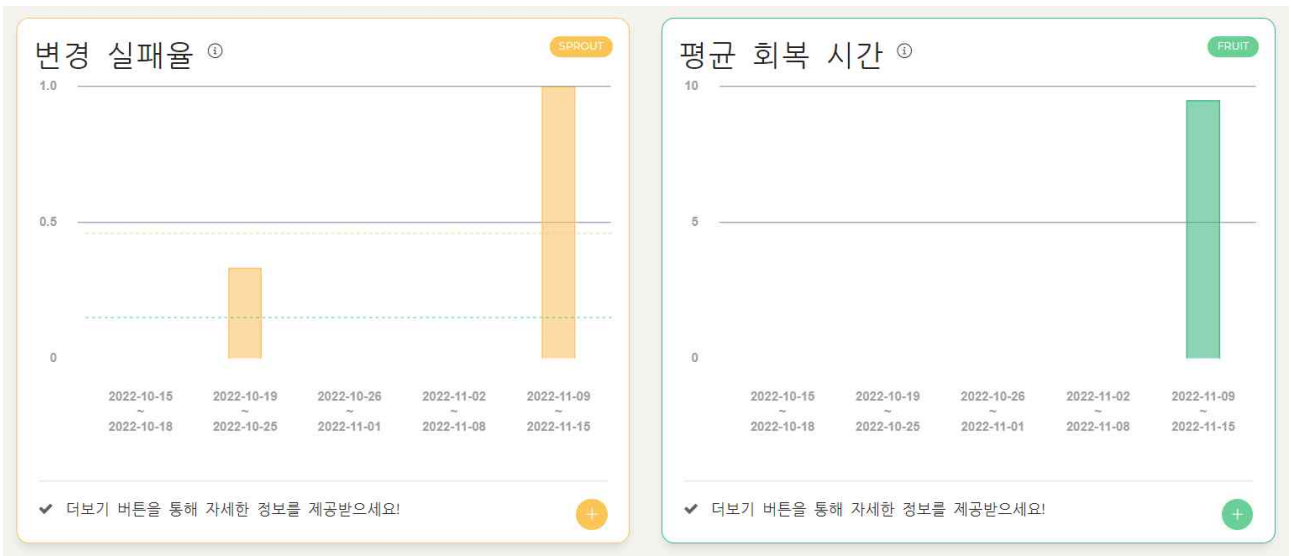


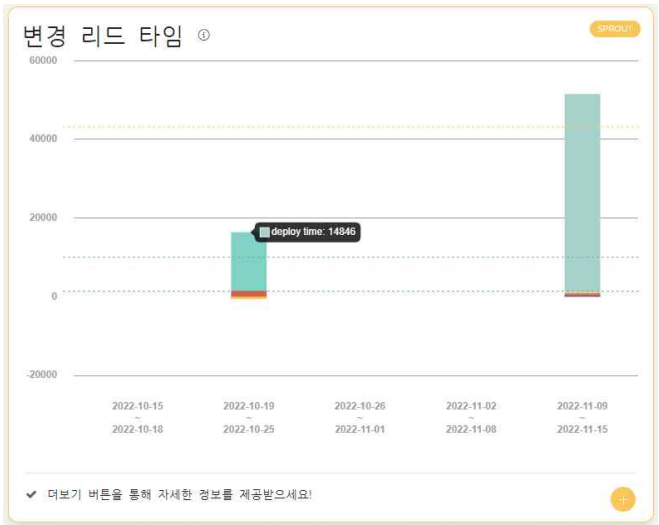
- Slack URL 등록 페이지 및 슬랙 알람

팀원들의 Burnout 관련 알람을 받고 싶으면, Slack URL을 등록하여 위험 감지 알림을 받을 수 있다. 서비스에서 위험을 감지하게 되면 Slack으로 메시지를 보낸다.

URL : https://www.rainmaker.cool/user/SlackWebhookRegister


기간 선택
시작일 - 종료일
단위
주
리포지토리 선택
Multiple Select
제출하기
변경 리드 타임
SPROUT
60000
40000
20000
0
2022-10-15 ~ 2022-10-18
2022-10-19 ~ 2022-10-25
2022-10-26 ~ 2022-11-01
2022-11-02 ~ 2022-11-08
2022-11-09 ~ 2022-11-15
더보기 버튼을 통해 자세한 정보를 제공받으세요!
배포 빈도
FRUIT
10
5
0
2022-10-15 ~ 2022-10-18
2022-10-19 ~ 2022-10-25
2022-10-26 ~ 2022-11-01
2022-11-02 ~ 2022-11-08
2022-11-09 ~ 2022-11-15
더보기 버튼을 통해 자세한 정보를 제공받으세요!

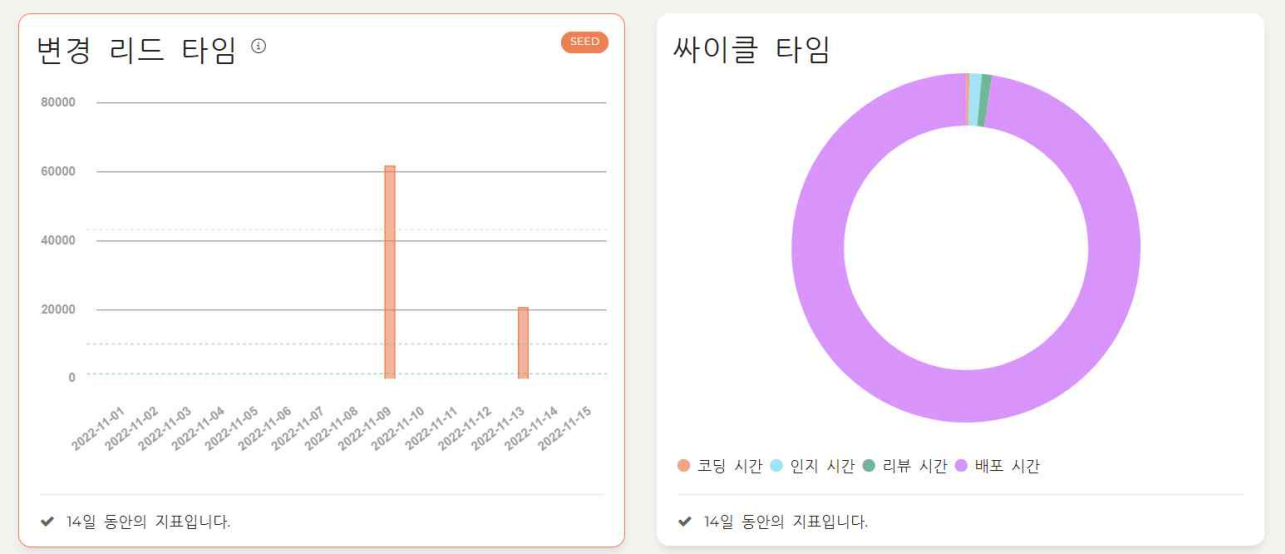

변경 실패율
SPROUT
1.0
0.5
0
2022-10-15 ~ 2022-10-18
2022-10-19 ~ 2022-10-25
2022-10-26 ~ 2022-11-01
2022-11-02 ~ 2022-11-08
2022-11-09 ~ 2022-11-15
더보기 버튼을 통해 자세한 정보를 제공받으세요!
평균 회복 시간
FRUIT
10
5
0
2022-10-15 ~ 2022-10-18
2022-10-19 ~ 2022-10-25
2022-10-26 ~ 2022-11-01
2022-11-02 ~ 2022-11-08
2022-11-09 ~ 2022-11-15
더보기 버튼을 통해 자세한 정보를 제공받으세요!

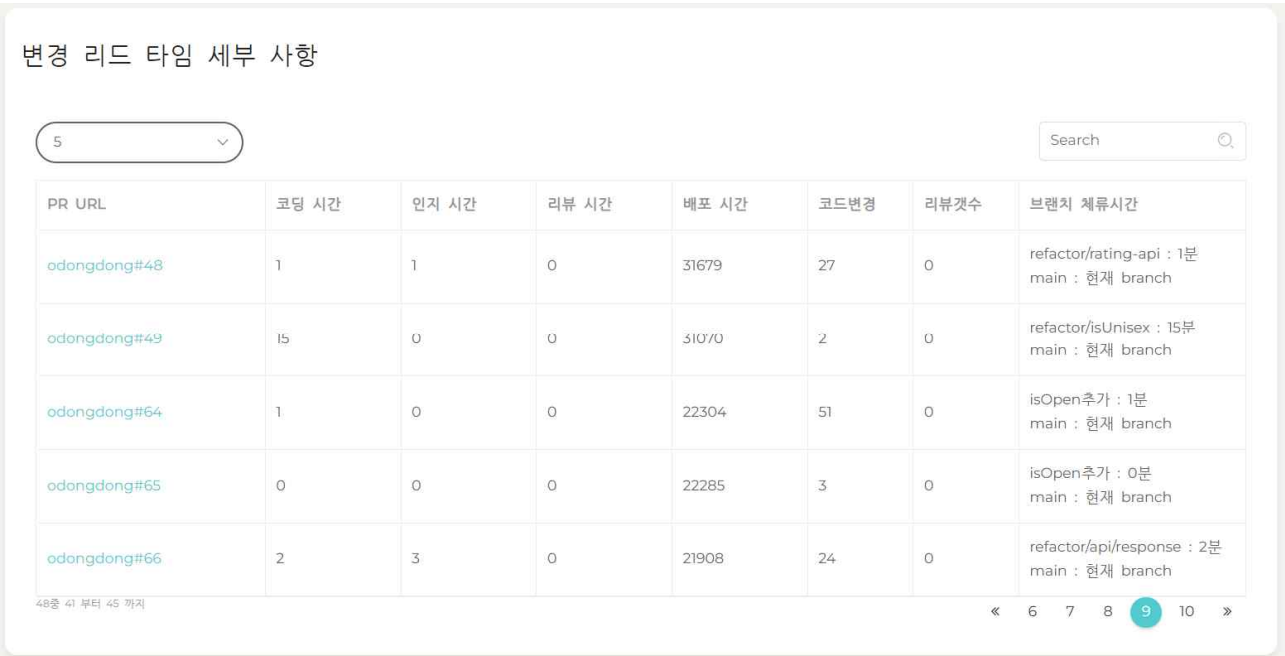

변경 리드 타임
SPROUT
60000
40000
20000
0
-20000
deploy time: 14846
2022-10-15 ~ 2022-10-18
2022-10-19 ~ 2022-10-25
2022-10-26 ~ 2022-11-01
2022-11-02 ~ 2022-11-08
2022-11-09 ~ 2022-11-15
더보기 버튼을 통해 자세한 정보를 제공받으세요!

- 메인 대시보드

메인 대시보드에서 Four key Metrics를 제공한다. 기간 선택 및 주/일별 데이터 선택, 리포지토리 선택이 가능하다. 기본적으로 1달의 정보를 모든 리포지토리에 대해서 주 단위로 보여준다. 지표에 마우스를 올려 자세한 정보 확인이 가능하다. 또한 우측 상단의 뱃지를 통해 팀의 생산성 수준과 성과를 파악할 수 있다.

URL : https://www.rainmaker.cool/dashboard



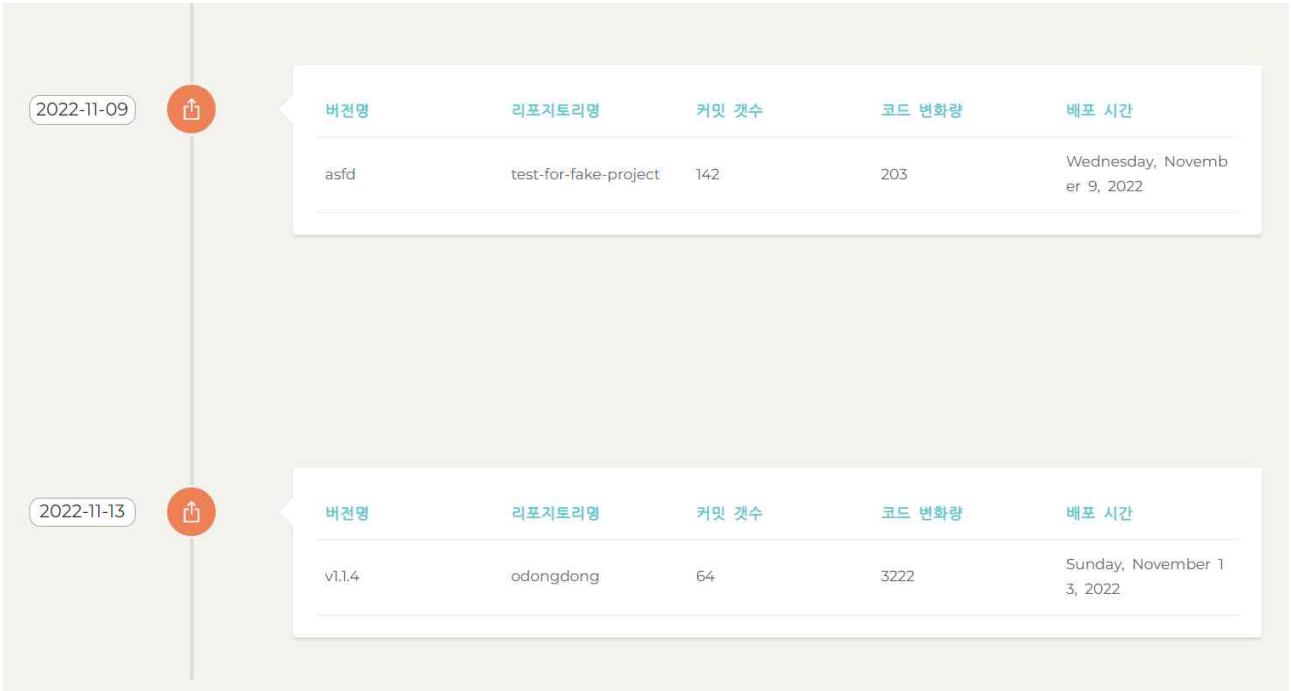


- 변경 리드 타임 상세 페이지

리드 타임에 관련한 14일 동안의 상세한 정보를 확인할 수 있다. 리드 타임의 구성 요소인 코딩 시간, 인지 시간, 리뷰 시간, 배포 시간을 상세하게 볼 수 있으며, 각 요소에 대한 평가를 확인할 수 있다. 또한 변경 리드 타임의

세부 사항을 통해 진행된 Pull Request의 정보를 확인할 수 있다.

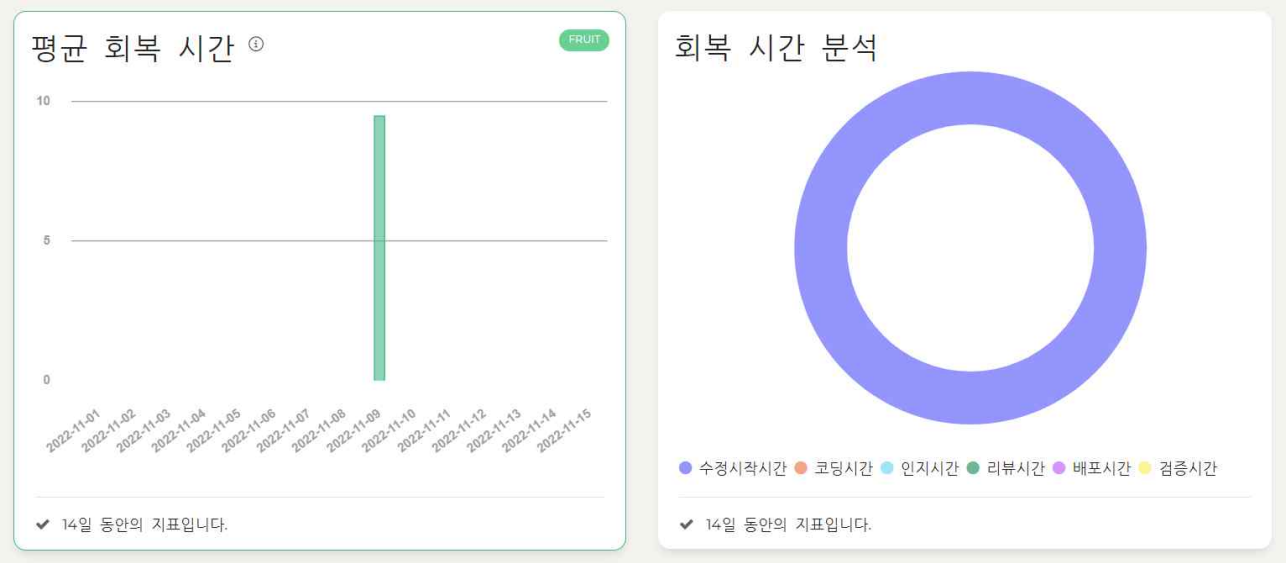
URL : https://www.rainmaker.cool/dashboard/lead-time-for-change

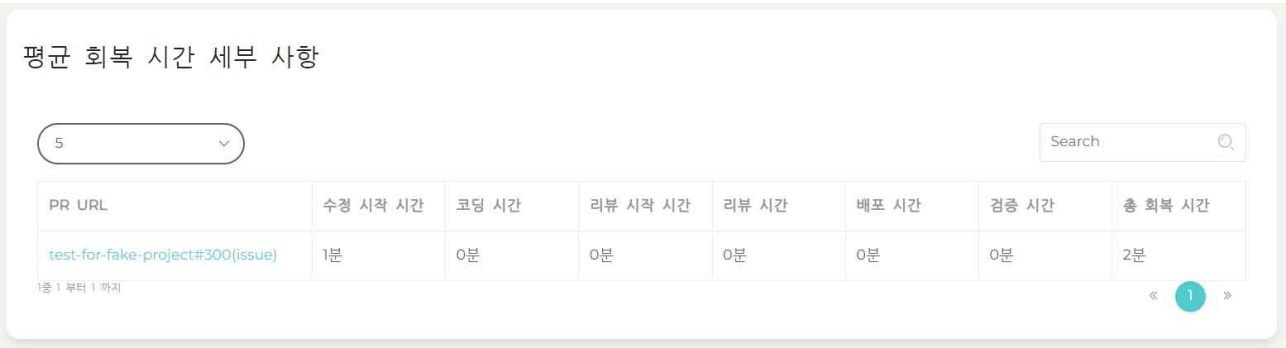


- 배포 빈도 상세 페이지

배포 빈도와 관련하여 14일 동안의 상세한 정보를 확인할 수 있다. 배포가 일어난 시점과 배포와 관련된 자세한 정보들을 제공받을 수 있다.

URL : https://www.rainmaker.cool/dashboard/deployment-frequency





- 평균 회복 기간 상세 페이지

평균 회복 기간과 관련하여 14일 동안의 상세한 정보를 확인할 수 있다. 평균 회복 기간의 구성 요소를 관련 Pull Request와 연관지어 상세하게 볼 수 있으며, 각 요소에 대한 평가를 확인할 수 있다. 또한 평균 회복 기간의 세부 사항을 통해 진행된 관련 이슈의 정보를 상세하게 표시해준다.

□ 기대효과 및 활용분야

ㅇ 기대효과

사용자 측면	프로젝트 및 팀을 이끌어 나가는 매니저의 입장에서, 우리 프로젝트를 통해 프로젝트의 진행을 막는 병목 지점을 식별하고 이를 능동적으로 해결해 나아갈 수 있다. 프로젝트 진행 상황에 있어 방해가 되는 Rework Rate(코드를 다시 작성하는 행위) // Change Failure Rate(배포 실패율 -> 전반적인 개발 과정에 있어 코딩/리뷰 실패) // Code Review Delay(성숙하지 않은 코드 리뷰 문화) // MTTR(런타임 중 에러의 복구율) 등의 프로젝트의 병목 지점이 될 수 있는 요소들과 관련된 지표를 확인하고 이를 효과적으로 해결 할 수 있다. 프로젝트 및 팀 개발 상황 측정을 자동화하여 시간적, 비용적 이득을 얻을 수 있다. 또한 팀에 속한 개발자 개인의 입장에서, 우리 프로젝트는 개발 중에 생길

	수 있는 많은 위험 요소(e.g. Big Diff(한 commit에 있어 너무 많은 코드 변경 // Change Failure rate(자신의 코드 실패율) 등)들을 파악하고 개발자에게 알려준다. 이를 받아봄으로써, 개발자 개인은 팀 협업에 있어서 자신의 어떤 행동이 팀에 위험을 주는지를 자동화하여 파악할 수 있고, 이를 통해 더 성숙한 협업 문화를 위해 노력할 수 있다.
비즈니스 측면	현재 최소한의 데이터 파이프라인을 유지하고 있는 고정 비용은 하루 평균 15,000₩으로 예측된다. 또한, 고정비용을 구성하는 인프라 scale을 넘어서는 한 팀이 하루 500개의 Event를 발생시켰을 때, 데이터 파이프라인에서 추가되는 비용은 약 2,000₩ 정도로 추산된다. 초기 사용자가 30팀이라고 예측했을 때, 우리가 하루 15,000₩을 받게 된다면, 우리측은 한 팀 당 8,000₩의 마진을 남기면서도, 다른 업체보다 낮은 가격으로 구독 서비스를 제공해줄 수 있다. 국내 SW 관련 기업의 수는 2019년 기준 25,188개가 있다. 우리의 목표는 한국 개발 시장에서의 시장 선점을 통한 독점이므로, 그 수 중 1/10의 시장 점유율을 먼저 가져갈 수 있을 것이라고 판단. 하루 2천만원의 마진을 남길 수 있다 라는 목표를 가지고 있다.
개발자 측면	Event 기반 Bigdata Pipeline을 개발하는 과정에서 발생하는 여러 문제들과 개념들을 해결하고 학습해 나가는 과정에서 데이터 처리 방식 및 Infra structure에 대한 이해를 높일 수 있음. 또한, 개발자를 대상으로 Product를 개발함으로써 개발자 문화를 탐구해 나가는 과정에서 질적 향상을 이뤄낼 수 있을 것으로 기대됨

○ 결과물 활용 방안

- 상표 등록
- 예비 창업 패키지
- 프로그램 저작권 출원
- 잔디/플로우(국내 IT 협업 툴)과 기술협약 및 내부 서비스로 온프라미스 서비스 제공

□ 중간점검 개선 요구사항 반영 결과(주요)

개선 요구사항	반영 결과
최종적인 결과물에 대한 것을 예상하기 어려워 최종 결과 보고서는 가시적인 결과를 보여주는 것이 필요함(예로 프로젝트 단위 밑에 많은 서비스 팀으로 나누어 개별 DORA 지표를 설명하여 이에 관련된 인사이트를 보여주는 것이 필요함)	지표 추적을 원하는 프로젝트를 각 프로젝트 뿐만이 아니라 팀 단위로도 모두 선택할 수 있도록 하여, 작은 프로젝트 단위 뿐만이 아닌 서비스 팀 단위로의 지표 또한 추출해주고 있음.
기획 의도는 좋으나 많은 이론적 배경이 필요하고 개발에 있어서 데이터 소스의 부정적 생성에 유의해야 함	잘못된 데이터 소스의 생성에 대해서는 기준을 세워 부정적인 데이터를 최대한 삭제하려고 하였음.
DORA 지수에 스토리텔링에 집중해서 설명을	유저가 가입한 Github의 종속된 팀의 정보라면

개선 요구사항	반영 결과
하는 것이 좋음. 최종 발표 때는 프로젝트 단위 밑에 많은 서비스 팀으로 나누어 개별 DORA 지표를 설명해야 하고 이에 관련된 인사이트가 보여질 수 있도록 하는 것을 권장함	프로젝트의 단위 뿐 아니라 서비스 팀 단위로의 지표 또한 추출이 가능함. 따라서 모든 단위로의 지표를 설명하는 것이 가능하나, 현재 팀 단위로의 비교를 통한 지표 추출을 이루어지고 있지 않음. 추후 개발을 통해 발전시킬 예정임.
새로운 이론에 대하여 우리나라 환경과 문화를 고려한 접근은 우수하나, 수집된 정보에 대한 분석 및 Insight 정보 제공이 필요함	기술 블로그를 만들어 수집된 정보에 관련된 정보를 제공하였고, 활용 방안 또한 제공하려고 노력하였음.
중간 보고까지 구현하여 검증은 좋았으나, 시장 진입을 위하여 차별화 기능 개발이 필요함 : 지표 도출 뿐만 아니라, 개발건전성 파 악 정보 제공 등	생산성 파악을 위한 지표 도출은 기본적으로 제공하고 있으며, 추가적으로 개발건전성 파악 정보 제공을 위하여 데이터를 분석하여 알람으로 개발자의 번아웃 위험 감지 알림을 제공하고 있음.
프로젝트 관리 시스템과의 연관되어 정보가 제공되어야 서비스 사용성 및 활용성을 증대가 예상됨	현재 프로젝트 관리 시스템인 Github의 토큰을 통해 데이터를 수집하여 서비스의 지표를 제공하고 있기 때문에, 사용자가 느낄 서비스 사용성과 활용성이 부정적이지 않을 것으로 예상됨.
새로운 지표에 대한 학습 측면에서는 유의미한 프로젝트라고 판단되나 DORA Metrics의 지표로 생산성에 대해 소통할 수 있는 예시 를 최종적으로 보여주는 것을 권장함	생산성에 대해 소통할 수 있는 예시를 보여주기 위해 예시 데이터를 포함한 페이지를 제작하여 사용자들에게 사용방안을 제시하고 소통할 수 있도록 개선시킬 예정임.
차별화로 제시된 부분이 해당 프로그램 기능에 포함 되었는지 향후 체크하는 것을 권장함 (특히, 외부 서비스와의 연동에 대해서)	외부 서비스와의 연동을 최소화하기 위해 Github Token만 제공해준다면, 이후의 데이터 수집 부분은 사용자가 신경 쓸 필요 없도록 구성하였음. 또한 더 많은 정보를 원하는 경우는 Slack URL을 등록하여 처리할 수 있도록 서비스를 제공함.
비즈니스를 위해 기존 툴과 비교하였을 때 반드시 본 프로그램을 써야 하는 킬러 콘텐츠가 필요한 측면으로 검토하는 것을 권장함	현존하는 경쟁사 제품과는 다르게 지표들의 자세한 정보들을 제공해주고 있으며, 지표를 추출해주는 과정이 더 명확하고 오차없이 계산하기 때문에 경쟁력을 가질 수 있을 것으로 보임.
오너 입장에서는 꼭 필요한 솔루션이 될 수도 있을 것 같고 하지만 아직 시장에 적용 사례가 많지 않다는 점에서 초기 고객사 확보 에 어려움을 겪을 수도 있다고 생각함	시장 적용 사례를 파악해보고, 이와 관련된 솔루션을 기술 블로그에 적용하여 제공해주기 위해 노력하고 있음
솔루션의 신뢰성과 정확성을 증명할 수 있는 사례 데이터 축적이 필요하다고 생각함	서비스의 활용 방안을 보여주기 위해서 사례 데이터를 지속적으로 만들어내었으며, 이를 서비스 예시 사용 방법으로 보여주어 사용자에게 신뢰성과 정확성을 보여주었음.
모니터링 대상에 대한 척도를 재정비 할 필요가 있음 (프로젝트에 각 배포 모듈별로 비교하는 도표 제공 등)	현재 각 배포 모듈별 디테일한 시각적인 정보를 제공하고 있으나, 프로젝트 배포 모듈별의 비교 정보는 추후 개선을 통해 추가할 예정임.
결과물의 대시보드가 사용자에게 어떤 정보를 제공해 주는 것인지 가이드 정보 제공이 필요	가이드라인을 제공하기 위해 예시 데이터를 포함한 페이지를 만들 예정이며, 또한 기술 블로

개선 요구사항	반영 결과
함(도표 보는 방법 등)	그에서 자세한 정보 제공을 해주고 있음.
팀의 병목 지점을 탐지해서 시각화하고 개발자의 위험을 탐지하기 위해 신뢰할만한 정략적인 평가 방안이 필요함 (개발자 역량을 어떻게 평가하고, 평가된 결과를 얼마나 신뢰할만 한지, 생산성 향상과의 상관관계는 어떻게 증명할 것인지 등)	팀의 병목 지점을 탐지해서 시각화하는 부분에서의 평가는 Google의 Four key 보고서를 바탕으로 진행하여 신뢰할만한 지표를 제공해주었다고 판단함. 그리고 팀의 기획 변경으로 인해 개발자의 역량에는 집중하지 않고 개발하였음. 팀의 역량을 평가하고 개선 방향을 정하는데 있어 개인에 대한 Blame이 들어가게 된다면 오히려 팀에 악영향을 미칠 수 있다고 판단.