

Team-E0F

Final Project

SaaS로 제공하는 성능 측정 시뮬레이터

➤ ↓

⇐

❯

∩ # ▽

01

Overview

- 팀 소개 및 구성
- 프로젝트 배경 및 필요성
- WBS (Work Breakdown Structure)

02

Tech Stack

- 분야별 사용 기술
- 기술 사용 방식

03

Composition

- 인프라 구성도
- 서비스 구성도
- CI/CD Demo

04

Project Process

- 주요 기능
- 웹 개발

05

Result Demo

- 프로젝트 데모 영상

06

Project Review

- 회고
- 부록



Project Overview



Introduce Team-EOF
Objectives of the project

01



Project Background

— □ ×



주제

Public Cloud AWS를 활용하여 쉽고 간편한 성능 측정 서비스를 제공한다.

project period: 2024.04.12-2024.06.05



배경 및 필요성

클라우드 인프라는 점점 더 복잡하고 다양화되고 있습니다. 성능 측정은 시스템의 안정성과 효율성을 유지하기 위해 필수적인 과정이지만, 대부분의 성능 측정 서비스는 전문적인 스크립트 작성이 필요합니다. 이는 많은 사용자들에게 큰 장벽을 느끼게 합니다.

EOF(Engineer of Future) 팀은 이러한 문제를 해결하기 위해 서버 성능 측정 서비스를 누구나 쉽고 간편하게 사용할 수 있도록 SaaS(Software as a Service) 형태로 제공하는 프로젝트를 기획하였습니다. 이를 통해 사용자들은 복잡한 스크립트 작성 없이도 손쉽게 서버 성능을 모니터링하고, 분석할 수 있게 됩니다.



프로젝트 목표

- 사용자 경험:** 직관적인 UI/UX 설계, 서비스 제공 편의성
- 비용 절약:** 최소 비용, 고효율 인프라 구성, 고가용 서비스를 제공
- CI/CD:** 효율적 팀 업무처리를 위한 자동화 배포 시스템 구축
- 데이터 관리:** AWS RDS를 활용한 멀티 AZ구성, S3 백업을 통한 안정성 제공
- 운영 효율성:** AWS EKS를 활용하여 유연한 서비스 스케일링 제공

Team Member



김정우

leader

contact

010-7237-4110

jwook_7@naver.com



백승현

Engineer

contact

010-7123-1387

tmdgjis1387@gmail.com



이원우

Developer

contact

010-2232-2114

ppuanam123456789@gmail.com



윤하경

Developer

contact

010-6652-9450

imvoonhk@gmail.com

팀 소개

팀명: EOF(Engineer of Future)

슬로건: “클라우드 엔지니어가 미래다!”

클라우드 기술이 미래의 엔지니어링의 핵심이라는 믿음으로, 구성된 팀입니다.

EOF팀은 미래의 엔지니어들이 보다 나은 성능과 안정성을 제공하기 위해 클라우드 기반의 서비스로 누구나 쉽게 접근하고 사용할 수 있는 성능 측정 솔루션을 제공하는 데 최선을 다하겠습니다.

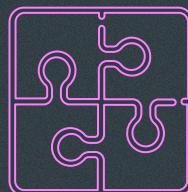


팀 구성

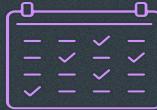
역할

이름

수행 내역



팀장 (PM)	김정우	프로젝트 계획 및 관리, 인프라 설계 및 구축
팀원 (PL)	백승현	인프라 설계 및 구축
팀원	이원우	성능 측정 서비스 APP 개발
팀원	윤하경	성능 측정 서비스 APP 개발



4월
3주차
04/15 ~
04/19

4월
4주차
04/22 ~
04/26

5월
1주차
04/29 ~
05/03

5월
2주차
05/06 ~
05/10

5월
3주차
05/13 ~
05/17

5월
4주차
05/20 ~
05/24

5월
5주차
05/27 ~
05/31

6월
1주차
06/03 ~
06/07

6월
2주차
06/10 ~
06/12

프로젝트 주제 선정

APP 개발 주제선정

자료 조사

성능측정 도구 분석
성능 지표 분석

산출물 작성

유저시나리오, 시스템 구성도,
API 설계서, DB 스키마 설계서

APP 개발

성능 측정 서비스 Front-End & Back-End 개발
Front & Back 연동 및 RDS 전환

인프라 구성

개발 환경 구성
Dockerfile, Terraform, CI/CD, AWS Cloud

프로젝트 관리

Slack, Notion, Google WorkSpace, Jira, Git 을 활용한 업무 협업 관리
WBS(Work Breakdown Structure) 관리, 프로젝트 결과서 작성



02

Tech Stack



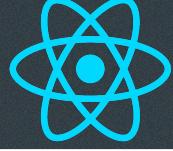
Technologies used
for the project



Skill <Infra>



Infra					
	AWS	Kubernetes	Docker	Terraform	Jenkins

Service			
	React	Fast API	MySQL

Cooperation					Google Workspace
	Jira	Slack	Notion	Git	Google WorkSpace

Skill <Service>



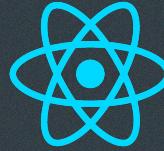
Infra					
	AWS	Kubernetes	Docker	Terraform	Jenkins

Service			
	React	Fast API	MySQL

Cooperation					Google Workspace
	Jira	Slack	Notion	Git	Google WorkSpace

Skill <Cooperation>



Infra					
	AWS	Kubernetes	Docker	Terraform	Jenkins
Service					
	React	Fast API	MySQL		
Cooperation					
	Jira	Slack	Notion	Git	Google WorkSpace

프로젝트 / TeamEof
타임라인

스print 보드

- TEAM-1 기존 앱을 테스트 도구를 바꿈...
- TEAM-2 어플리케이션 설정한다.
- TEAM-3 Backend 구현**
- TEAM-4 Frontend 구현
- TEAM-5 Indx 구현
- TEAM-6 세대 후보코드
- TEAM-7 개발환경을 구현한다.
- 만들기 대체

스프린트 시작

10개의 이슈를 이 스프린트에 포함합니다.
필수 필드는 빨간색 표시되어 있습니다

스프린트 이름 *

7주차 스프린트

기간 *

사용자 지정

시작 날짜 *

2024. 6. 2. 오후 10:56

종료 날짜 *

2024. 6. 13. 오후 10:56

스프린트 목표

<api> 기능 구현
프론트-전원적 기능 디테일 수정
백엔드-프론트와 연결 및 추가 기능 구현
- 인프라- AWS 환경 위에 CI/CD 등 서비스 구축하기

프로젝트 / TeamEof
2024-3

검색

일정 중 5

미지 일정

DB 구축 (DB 스키마 정의가 진행 작업)
개발환경을 구성한다.

TEAM-31

Back - Dockerfile 작성
개발환경을 구성한다.

TEAM-42

Front - Dockerfile 작성
개발환경을 구성한다.

TEAM-53

test Create API 구현
BACKEND 구현

TEAM-43

Back - 스크래쳐 코드 작성
BACKEND 구현

TEAM-27

Back - 스파클론 코드 작성
BACKEND 구현

TEAM-22

How to use Jira

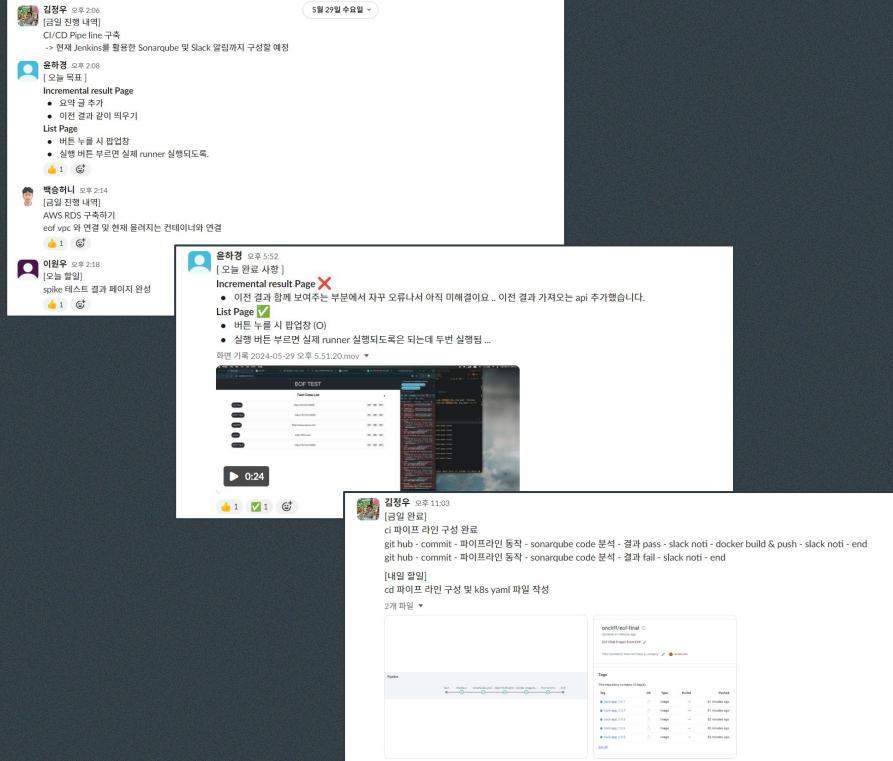


매 주 백로그를 작성하고, 팀원 별로

티켓을 분배합니다. 스프린트를 진행하기

위해 기간을 정해 업무를 수행합니다.

일의 진행 과정은 칸반보드 위에 표시됩니다.



How to use Slack



슬랙을 통해 업무 내용을 소통합니다.

업무 시작 전 백로그를 기준으로 오늘
과업에 관해서 공유합니다.

업무 종료 전 금일 진행 상황을 공유하며,
완료된 티켓과 미흡한 부분에 대해 의견을
주고받습니다.

ARAOLGA commented 5 minutes ago

팀장님 코드 확인 후 머지 부탁해요....
진짜 끝났습니다. 진짜로 아니 진짜로..

[Front]

1. Loading 기능 구현 및 테스트 완료
2. Front 및 Back Time Sync issue 수정 및 테스트 완료

[Back]

1. Spike Test 결
2. Spike Test 진

JIRA 티켓 완료.
<https://teamof.a>

oncliff-climbing commented now

고생 많았습니다.

[Front]

1. Loading 기능 구현 및 테스트 완료 → 수정 확인
2. Front 및 Back Time Sync issue 수정 및 테스트 완료 → 수정 확인

[Back]

3. Spike Test 결과 비교 기능 구현 완료 → 수정 확인
4. Spike Test 진행 시 DB 저장 오류 해결 → 수정 확인

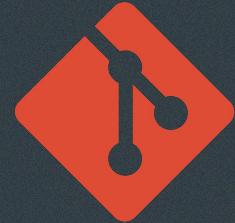
[JIRA]

TEAM-13 Issue 상태 완료 처리

근데 이 땐 게 개발 ..?

1

How to use Git



코드 형상관리를 위해 Git Repository를 활용합니다. 코드 변경 사항이 발생하면 Pull Request를 요청하고, 진행 내역에 관해서 간략한 설명을 작성합니다. 전달된 요청은 팀원 간 코드 리뷰 후

Code Merge를 수행합니다.

Pipeline

Definition

Pipeline script

```

1 pipeline {
2     agent any
3     environment {
4         K8S_NODE_IP = '43.206.109.37'
5         K8S_NODE_PORT = '30000'
6         K8S_NODE_NAME = 'node-01'
7         DOCKER_TAG = '1.0'
8         K8S_DEPLOYMENT_APP = 'app-deploy'
9         K8S_DEPLOYMENT_BACK = 'back-app'
10        K8S_YAML = 'app-deploy.yaml'
11        K8S_NAMESPACE = 'k8s-namespace'
12        K8S_DEPLOYMENT_FRONT = 'front-app'
13        K8S_DEPLOYMENT_SA = 'sa-back-app'
14        K8S_SA_NAME = 'sa-back-app'
15        K8S_SA_SECRET = 'sa-back-app'
16        K8S_SELECTOR = 'app=app-northeast-1'
17    }
18
19    stages {
20        stage('Checkout') {
21            steps {
22                checkout([$class: 'GitSCM',
23                    branches: [[name: '*/main']],
24                    doGenerateTag: false,
25                    extensions: [],
26                    submoduleCfg: [],
27                    userRemoteConfigs: [[
28                        credentialsId: 'K8S_DEPLOYMENT_CREDENTIALS_ID',
29                        ref: '+refs/heads/main',
30                        url: 'https://github.com/oncliff/eof-final.git'
31                    ]],
32                    poll: true
33                }
34            }
35        }
36        stage('Docker Image Build and Push') {
37            steps {
38                script {
39                    def env = [
40                        K8S_NODE_IP: K8S_NODE_IP,
41                        K8S_NODE_NAME: K8S_NODE_NAME,
42                        DOCKER_TAG: DOCKER_TAG,
43                        K8S_DEPLOYMENT_APP: K8S_DEPLOYMENT_APP,
44                        K8S_DEPLOYMENT_BACK: K8S_DEPLOYMENT_BACK,
45                        K8S_YAML: K8S_YAML,
46                        K8S_NAMESPACE: K8S_NAMESPACE,
47                        K8S_DEPLOYMENT_FRONT: K8S_DEPLOYMENT_FRONT,
48                        K8S_SA_NAME: K8S_SA_NAME,
49                        K8S_SA_SECRET: K8S_SA_SECRET,
50                        K8S_SELECTOR: K8S_SELECTOR
51                    ]
52                    env['K8S_NODE_IP'] = K8S_NODE_IP
53                    env['K8S_NODE_NAME'] = K8S_NODE_NAME
54                    env['DOCKER_TAG'] = DOCKER_TAG
55                    env['K8S_DEPLOYMENT_APP'] = K8S_DEPLOYMENT_APP
56                    env['K8S_DEPLOYMENT_BACK'] = K8S_DEPLOYMENT_BACK
57                    env['K8S_YAML'] = K8S_YAML
58                    env['K8S_NAMESPACE'] = K8S_NAMESPACE
59                    env['K8S_DEPLOYMENT_FRONT'] = K8S_DEPLOYMENT_FRONT
60                    env['K8S_SA_NAME'] = K8S_SA_NAME
61                    env['K8S_SA_SECRET'] = K8S_SA_SECRET
62                    env['K8S_SELECTOR'] = K8S_SELECTOR
63
64                    def pipelineScript = """
65                        pipeline {
66                            agent any
67                            environment {
68                                K8S_NODE_IP: ${K8S_NODE_IP},
69                                K8S_NODE_NAME: ${K8S_NODE_NAME},
70                                DOCKER_TAG: ${DOCKER_TAG},
71                                K8S_DEPLOYMENT_APP: ${K8S_DEPLOYMENT_APP},
72                                K8S_DEPLOYMENT_BACK: ${K8S_DEPLOYMENT_BACK},
73                                K8S_YAML: ${K8S_YAML},
74                                K8S_NAMESPACE: ${K8S_NAMESPACE},
75                                K8S_DEPLOYMENT_FRONT: ${K8S_DEPLOYMENT_FRONT},
76                                K8S_SA_NAME: ${K8S_SA_NAME},
77                                K8S_SA_SECRET: ${K8S_SA_SECRET},
78                                K8S_SELECTOR: ${K8S_SELECTOR}
79                            }
80                            stages {
81                                stage('Docker Image Build and Push') {
82                                    steps {
83                                        sh "cd ${K8S_YAML} && kubectl apply -f ."
84                                    }
85                                }
86                                stage('Deploy to Kubernetes') {
87                                    steps {
88                                        sh "cd ${K8S_YAML} && kubectl get pods --selector ${K8S_SELECTOR} --output yaml > ${K8S_YAML}.yaml && kubectl apply -f ${K8S_YAML}.yaml"
89                                    }
90                                }
91                            }
92                        }
93                    """
94                    echo pipelineScript
95                    sh pipelineScript
96                }
97            }
98        }
99    }
100
101   post {
102       success {
103           script {
104               def pipelineScript = """
105                   pipeline {
106                       agent any
107                       environment {
108                           K8S_NODE_IP: ${K8S_NODE_IP},
109                           K8S_NODE_NAME: ${K8S_NODE_NAME},
110                           DOCKER_TAG: ${DOCKER_TAG},
111                           K8S_DEPLOYMENT_APP: ${K8S_DEPLOYMENT_APP},
112                           K8S_DEPLOYMENT_BACK: ${K8S_DEPLOYMENT_BACK},
113                           K8S_YAML: ${K8S_YAML},
114                           K8S_NAMESPACE: ${K8S_NAMESPACE},
115                           K8S_DEPLOYMENT_FRONT: ${K8S_DEPLOYMENT_FRONT},
116                           K8S_SA_NAME: ${K8S_SA_NAME},
117                           K8S_SA_SECRET: ${K8S_SA_SECRET},
118                           K8S_SELECTOR: ${K8S_SELECTOR}
119                       }
120                       stages {
121                           stage('Docker Image Build and Push') {
122                               steps {
123                                   sh "cd ${K8S_YAML} && kubectl apply -f ."
124                               }
125                           }
126                           stage('Deploy to Kubernetes') {
127                               steps {
128                                   sh "cd ${K8S_YAML} && kubectl get pods --selector ${K8S_SELECTOR} --output yaml > ${K8S_YAML}.yaml && kubectl apply -f ${K8S_YAML}.yaml"
129                               }
130                           }
131                       }
132                   }
133               """
134               echo pipelineScript
135               sh pipelineScript
136           }
137       }
138   }
139 }

```

Use Groovy Sandbox

Pipeline Syntax

Changes

Summary

- 1. Update App.js (commit: f5ce08e) (details)

Commit f5ce08e65de65c1a4bf3858e6c919219
Update App.js
(commit: f5ce08e)

front-app/src/App.js (diff)

How to use Jenkins



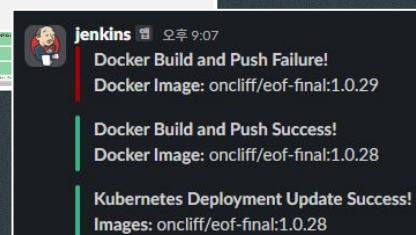
Jenkins 를 활용하여 ci/cd 자동화

배포라인을 구축합니다. Git Commit

발생 시 Webhook을 통해서

Pipeline 이 동작됩니다. Pipeline 실행

결과는 슬랙을 통해 전달됩니다.



— □ ×

Project Composition



Diagram for
Infra and Service

03

— □ ×

÷ ▷ ⇕

Diagram <Infra>

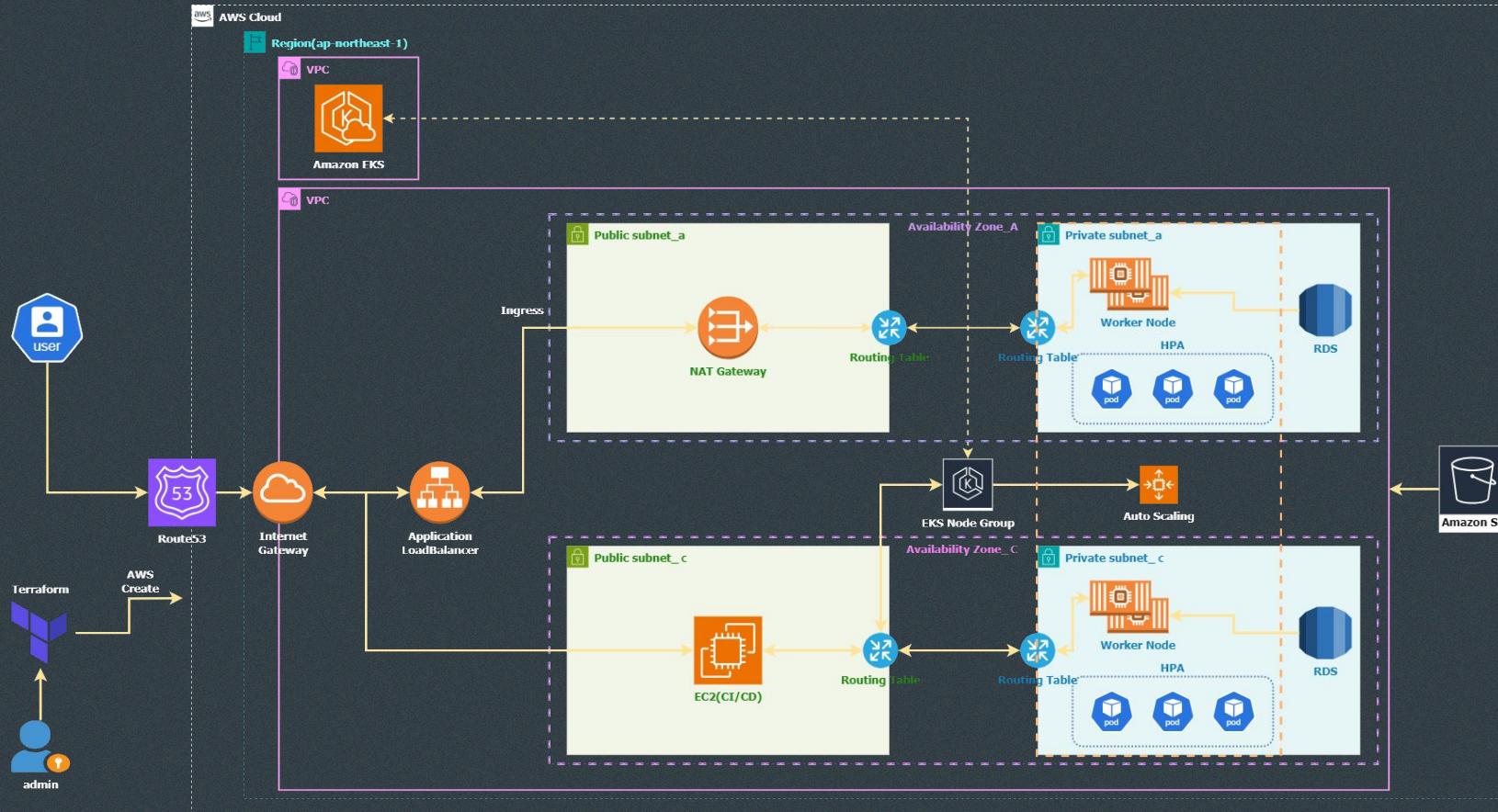
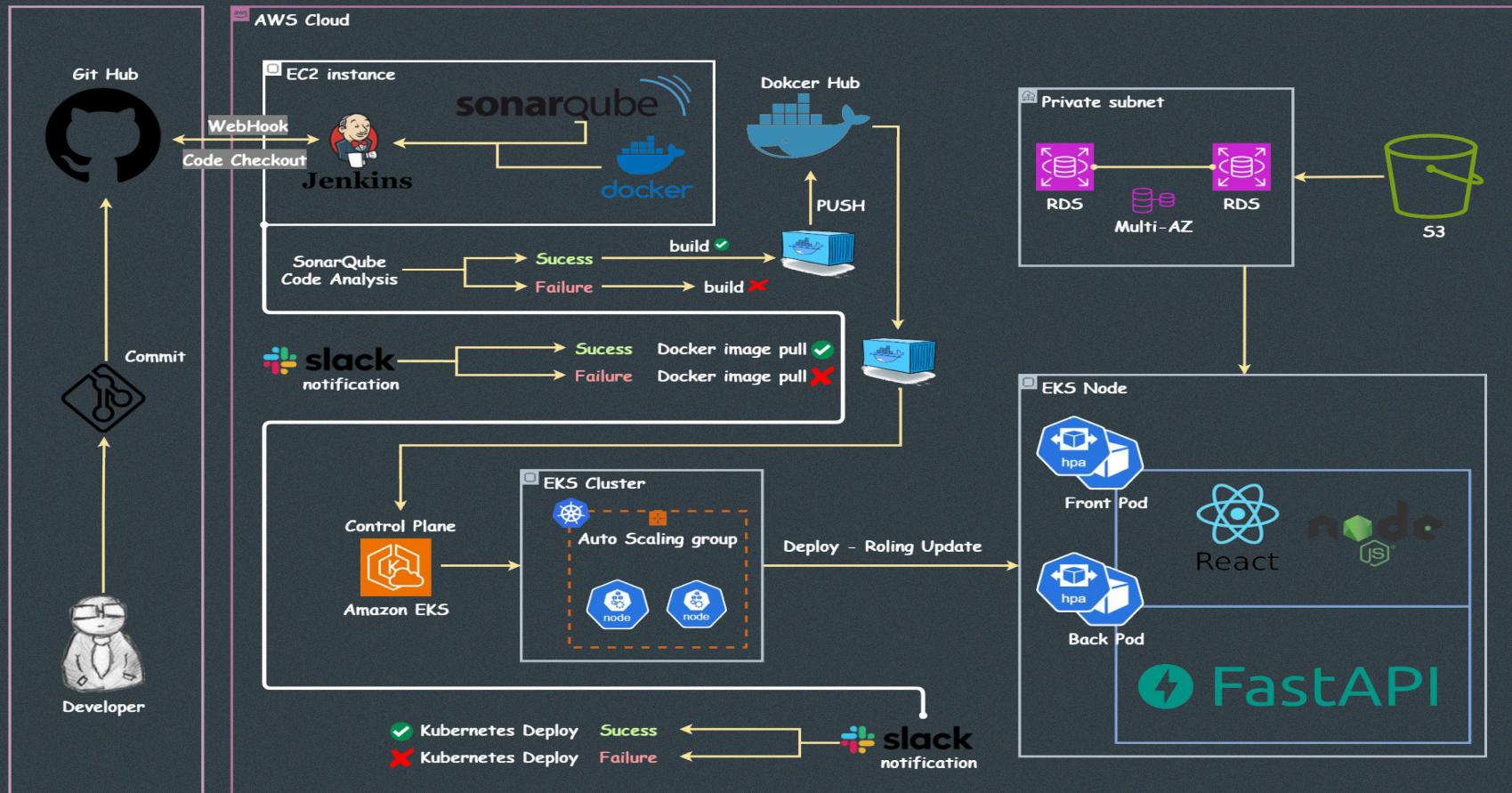


Diagram <Service>

— □ ×



CI/CD

<Demo>



ec2-43-206-109-37.ap-northeast-1.compute.amazonaws.com

Session Servers Tools Games Sessions View Tunneling Packages Settings Help Split MultiExec

X server Exit

Quick connect...

2.ec2-43-206-109-37.ap-northeast-1.compute.amazonaws.com

```
ubuntu@ip-10-0-1-17:~$ k get all -n eof -o wide
NAME          READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
pod/back-app-7594fcdf-29f15  1/1    Running   0          8m47s  10.0.2.91  ip-10-0-2-54.ap-northeast-1.compute.internal  <none>        <none>
pod/back-app-7594fcdf-5gtsk  1/1    Running   0          8m46s  10.0.3.59  ip-10-0-3-169.ap-northeast-1.compute.internal  <none>        <none>
pod/front-app-58557fc9df-j95n7 1/1    Running   0          8m46s  10.0.3.97  ip-10-0-3-169.ap-northeast-1.compute.internal  <none>        <none>
pod/front-app-58557fc9df-t4ndk 1/1    Running   0          8m46s  10.0.2.164 ip-10-0-2-54.ap-northeast-1.compute.internal  <none>        <none>

NAME          TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE     SELECTOR
service/back-service ClusterIP  172.20.178.89 <none>       8080/TCP  5h18m  app=back
service/front-service  ClusterIP  172.20.172.2  <none>       80/TCP    5h18m  app=front

NAME          READY   UP-TO-DATE  AVAILABLE   AGE     CONTAINERS   IMAGES          I   SELECTOR
deployment.apps/back-app  2/2    2          2          5h18m  back        oncliff/eof-final:back-app_1.0.116  app=back
deployment.apps/front-app 2/2    2          2          5h18m  front       oncliff/eof-final:front-app_1.0.116 app=front

NAME          DESIRED  CURRENT  READY   AGE     CONTAINERS   IMAGES          I   SELECTOR
replicaset.apps/back-app-7594fcdf  2       2       2       8m47s  back        oncliff/eof-final:back-app_1.0.116  app=back,pod-template-hash=7594fcdf
replicaset.apps/front-app-58557fc9df 2       2       2       8m47s  front       oncliff/eof-final:front-app_1.0.116 app=front,pod-template-hash=58557fc9df

NAME          REFERENCE  TARGETS  MINPODS  MAXPODS  REPLICAS   AGE
horizontalpodautoscaler.autoscaling/eof-back-app-hpa  Deployment/back-app  <unknown>/50%  2         10        2         5h18m
horizontalpodautoscaler.autoscaling/eof-front-app-hpa  Deployment/front-app <unknown>/50%  2         10        2         5h18m

ubuntu@ip-10-0-1-17:~$
```



— □ ×
04

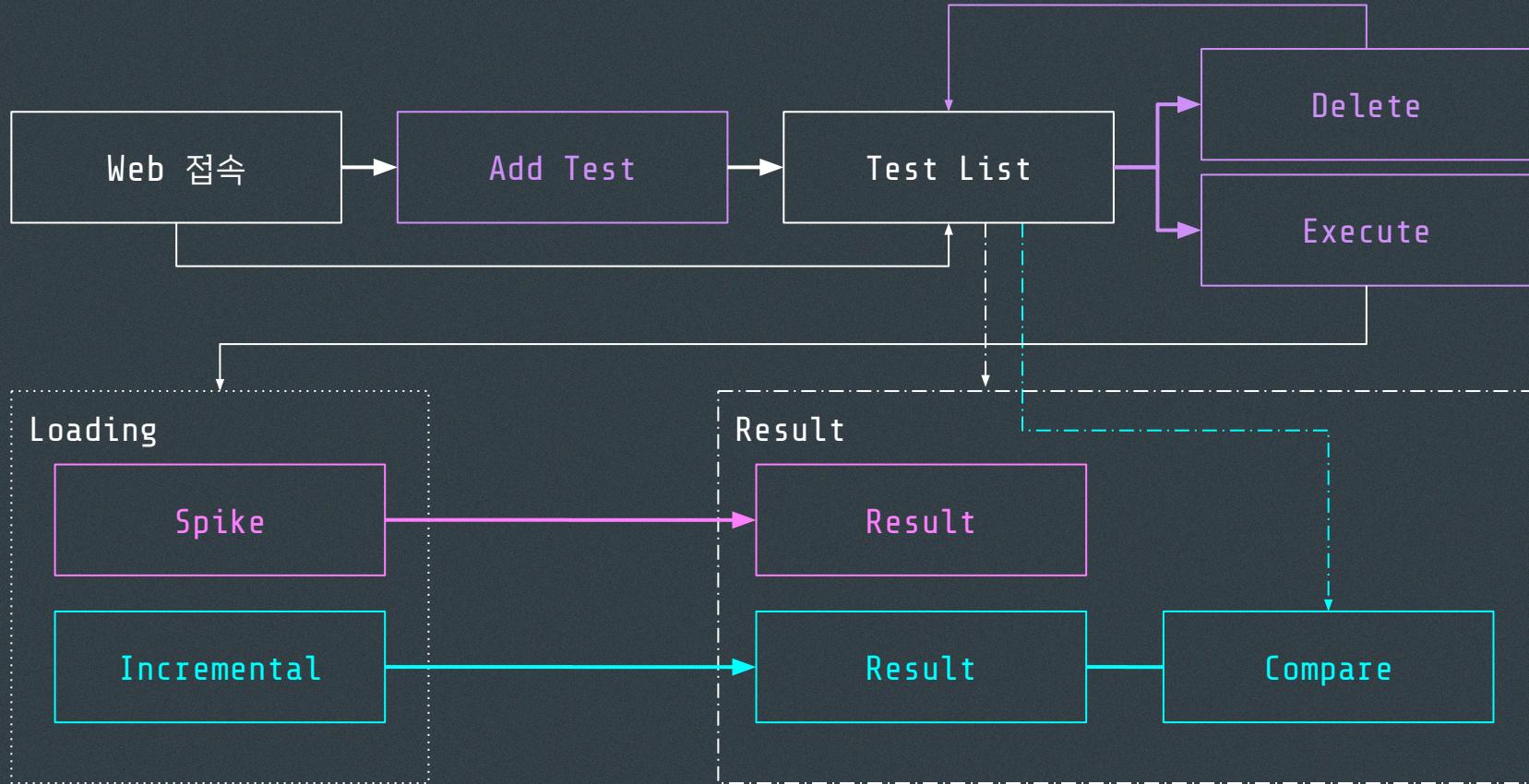
Project Process

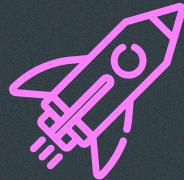
Main function
Composition of Service



Functional Flow Chart

— □ ×





Spike Test

- 트래픽의 갑작스러운 증가 후 감소하는 패턴.
- 시스템이 급격한 트래픽 증가에 얼마나 잘 대처하는지 평가.
- 초기 유저 수를 일괄적으로 생성한 후 부하를 주입.



Incremental Test

- 트래픽이 점진적으로 증가하는 패턴.
- 시스템이 점진적으로 증가하는 부하를 얼마나 잘 처리하는지 평가.
- 초기 입력값만큼 가상 유저 생성,
일정 간격으로 유저수를 증가시켜 부하를 주입.

User Input

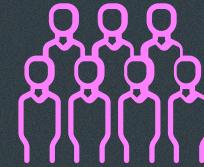
— □ ×



대상 URL



테스트 명



초기 유저 수 (명)



증가 유저 수 (명)



증가 간격 (초)



증가 횟수 (번)

spike
incremental

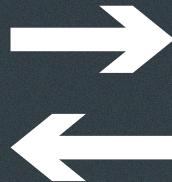
How to Load Test

— □ ×



USER

HTTP GET



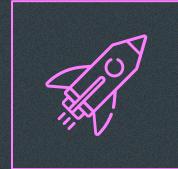
Target URL

- ✓ 초당 요청 수
- ✓ 응답 시간
- ✓ 실패율

Spike Test Result

— □ ×

IF 증가 유저 수 or 증가 간격 or 증가 횟수 == 0 :



실패율



ex) 0%

유저 수



ex) 3000명

평균 응답 속도



ex) 5초

Incremental Test Result

— □ ×

ELSE :



초당 실패율



ex) 0%

유저 수



ex) 100명

평균 응답 시간



ex) 1초

초당 요청 수

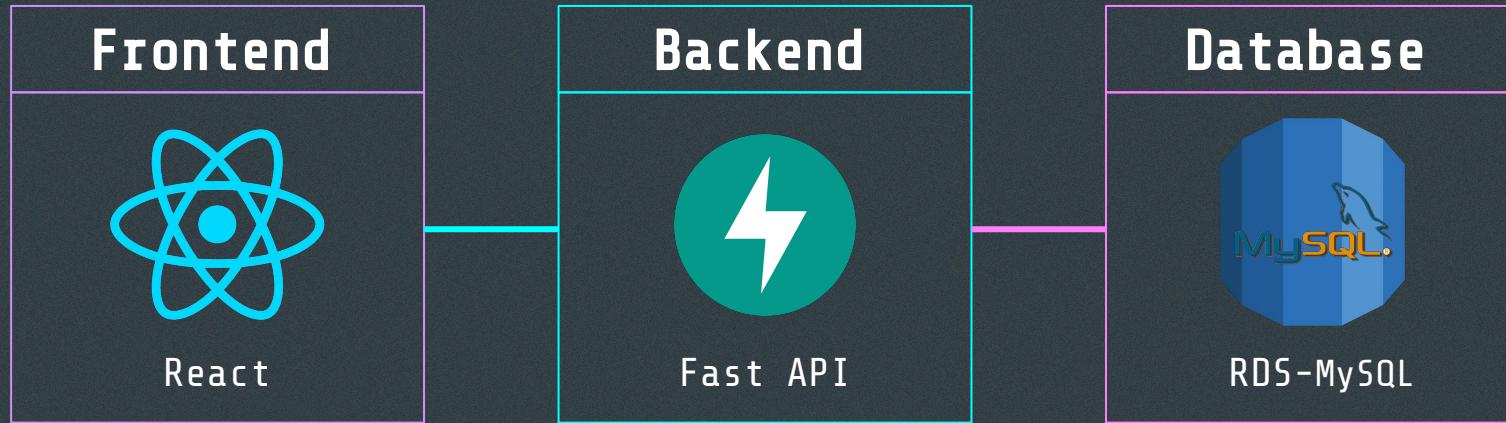


ex) 82

현재 시각



ex) 16:02:34



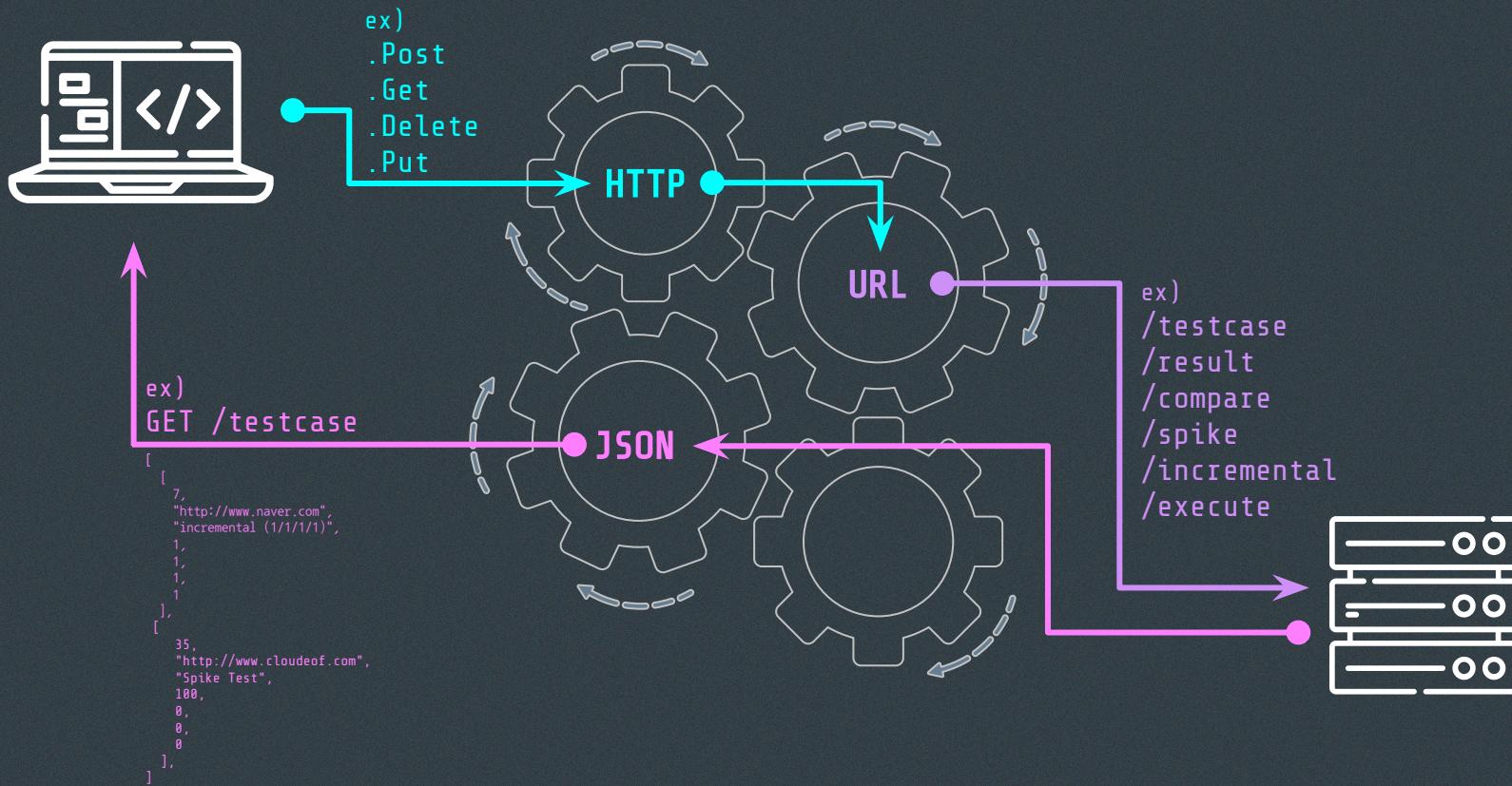
기존의 유기적으로 결합된 MPA(Mulitple Page Application) 방식보다 현재 개발 트렌드인 모듈화 방식에 맞춰서 대표적인 SPA(Single Page Application) 프레임워크를 조사하던 도중 우리 시스템에 적합한 React를 선정하게 되었다.

성능측정 서비스에 걸맞는 비동기 방식인 Fast API를 선정하게 되었다. 테스트 결과 비동기(Fast API) 방식이 동기(Django, Flask) 보다 약 15초가량 빠른 것을 확인할 수 있었습니다. (USER=1000 기준)

성능 테스트 결과를 저장하고 관리하기 위해 자주 사용되는 MySQL과 PostgreSQL 중 익숙한 MySQL을 선정하게 되었다. 실시간 성능 데이터를 수집 및 시각화에 적합한 InfluxDB가 있지만 시간이 부족해 적용해 보지는 못했다.

What is a Rest API?

— □ ×



Post

/testcase

테스트케이스 정보를 저장한다.

Get

/testcase

테스트케이스 목록을 불러온다.

/testcase/{id}/spike

spike 테스트 결과를 가져온다.

/testcase/{Id}/execute

부하 테스트를 진행한다.

/testcase/{Id}/stats

부하테스트 결과를 1초마다 가져와 그래프를 그린다.

/testcase/{Id}/results

incremental 테스트의 최신 결과를 가져온다.

/testcase/{Id}/stats/{selectedResult}

incremental 테스트 중 선택한 번호의 결과를 가져온다.

Delete

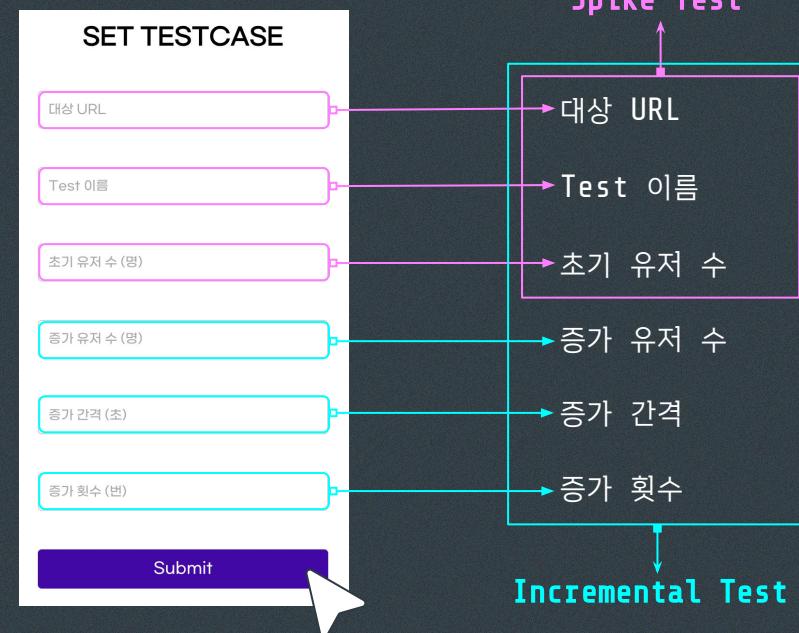
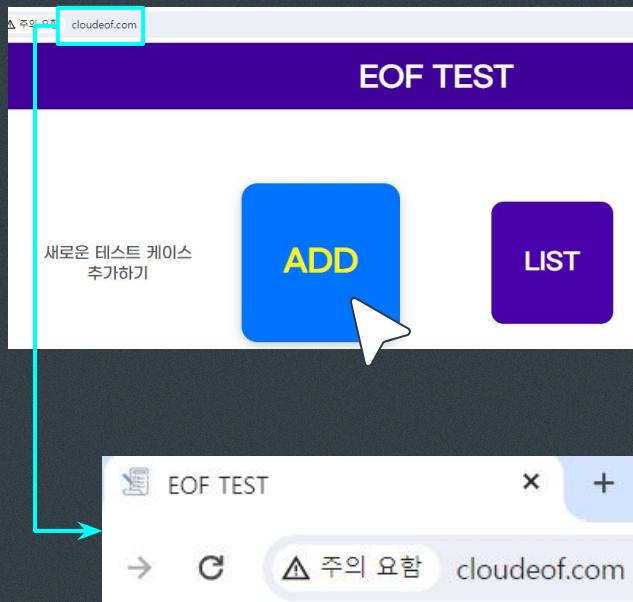
/testcase/{Id}

테스트케이스를 삭제한다.

Function </addTestcase>

— □ ×

Client



Function </addTestcase>

— □ ×

Serve

I
Client

SET TESTCASE

https://cloudcrew.site/

Incremental Test

1000

100

10

3

Submit



Post (/testcase)

테스트케이스를 추가한다

api.test: 2 행 (총) (exact)

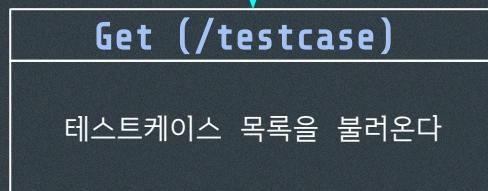
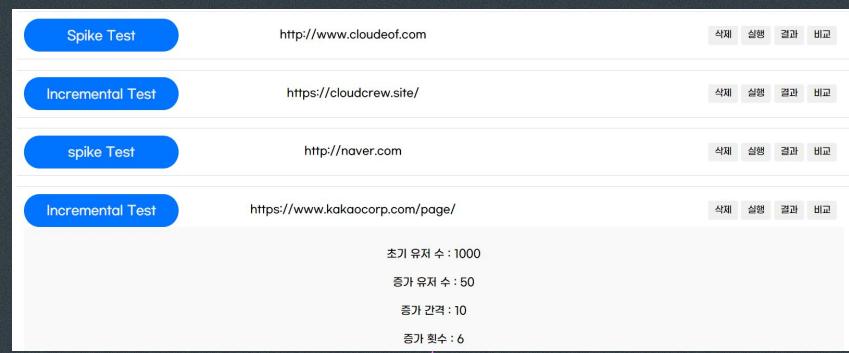
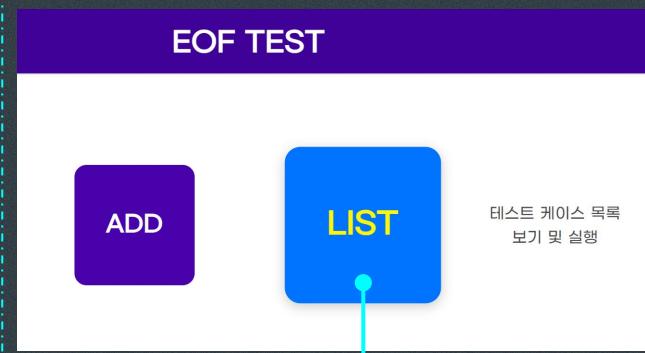
#	test_id	target_url	test_name	user_num	user_plus_num	interval_time	plus_count
1	46	http://www.cloudeof.com	Spike Test	1,000	0	0	0
2	47	https://cloudcrew.site/	Incremental Test	1,000	100	10	3

Function </testList>

— □ ×

Serve

I
Client



api.test: 4 행 (총) (exact)							
#	test_id	target_url	test_name	user_num	user_plus_num	interval_time	plus_count
1	46	http://www.cloudeof.com	Spike Test	1,000	0	0	0
2	47	https://cloudcrew.site/	Incremental Test	1,000	100	10	3
3	48	http://naver.com	spike Test	5,000	0	0	0
4	49	https://www.kakaocorp.com/page/	Incremental Test	1,000	50	10	6

EOF TEST

Test Case List

Spike Test	http://www.cloudeof.com
Incremental Test	https://clouddcrew.site/
spike Test	http://naver.com
Incremental Test	https://www.kakaocorp.com/page/

삭제 실행 결과 비교

삭제 실행 결과 비교

Delete
해당 테스트 케이스를 삭제합니다.

Get
테스트를 실행합니다.

Get
테스트 실행 결과를 보여줍니다.

Get
해당 테스트의 과거 결과와
최신 테스트의 결과를 보여줍니다.

* spike test는 비교 기능을 제공하지 않습니다.

Function </delete>

— □ ×

Serve

I Client

www.cloudeof.com 내용:
Spike Test(를) 정말 삭제하시겠습니까?

확인 취소

Spike Test

http://www.cloudeof.com

삭제

삭제 실행 결과 비교

Incremental Test

https://cloudcrew.site/

삭제 실행 결과 비교

EOF TEST

Test Case List

Incremental Test

https://cloudcrew.site/

삭제 실행 결과 비교

Delete (/ testcase/{Id})

테스트 케이스를 삭제한다

api:test: 3 행 (중) (exact)		test_name	user_num	user_plus_num	interval_time	plus_count
#	test_id	target_url				
1	47	https://cloudcrew.site/	Incremental Test	1,000	100	10

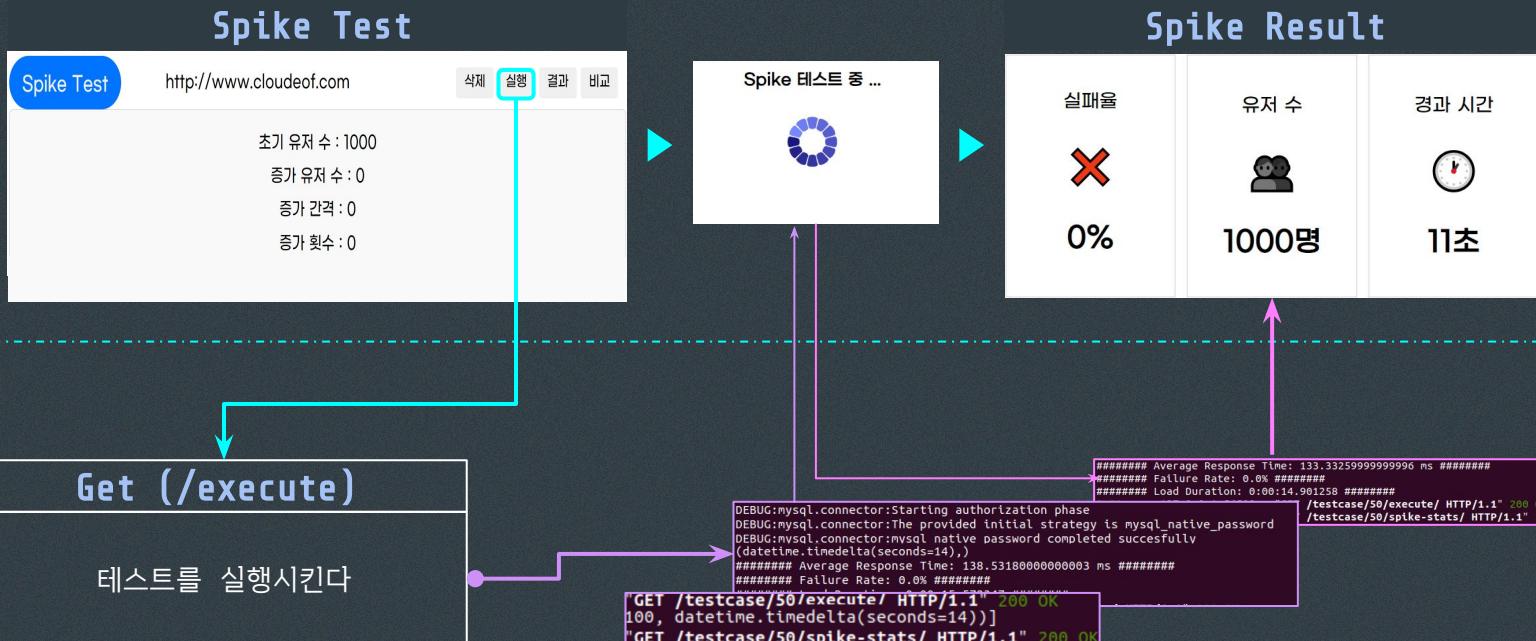
api:test: 4 행 (중) (exact)		test_name	user_num	user_plus_num	interval_time	plus_count
#	test_id	target_url				
1	46	http://www.cloudeof.com	Spike Test	1,000	0	0
2	47	https://cloudcrew.site/	Incremental Test	1,000	100	10

Function </execute/Spike>

— □ ×

Serve

I
Client



Function </ testcase/Incremental >

— □ ×

Serve

I
Client

Incremental Test

Incremental Test

http://www.naver.com

실행

결과

비교

초기 유저 수 : 100

증가 유저 수 : 10

증가 간격 : 3

증가 횟수 : 5

incremental Loading



Result

테스트가 완료되었습니다!

Test 이름: Incremental Test
대상 URL: http://www.naver.com
초기 유저 수: 100 명
증가 유저 수: 10 명
증가 간격: 3 초
증가 횟수: 5 번

결과 확인하기

Get (/execute)

테스트를 실행시킨다

```
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): www.naver.com:80
DEBUG:urllib3.connectionpool:http://www.naver.com:80 "GET / HTTP/1.1" 302 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.naver.com:443
DEBUG:urllib3.connectionpool:https://www.naver.com:443 "GET / HTTP/1.1" 206 36649
DEBUG:root:Request to http://www.naver.com successful. Response time: 158.71699999999998 ms, Content length: 232180 bytes
130 3.462573
130 1.081816
130 1.08678
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): www.naver.com:80
DEBUG:urllib3.connectionpool:http://www.naver.com:80 "GET / HTTP/1.1" 302 138
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.naver.com:443
DEBUG:urllib3.connectionpool:https://www.naver.com:443 "GET / HTTP/1.1" 206 41284
DEBUG:root:Request to http://www.naver.com successful. Response time: 416.765 ms, Content length: 258509 bytes
```

Function </ testcase / Incremental >

— □ ×

Serve

I
Client

Incremental Test

Incremental Test

http://www.naver.com

삭제 실행 결과 비교

초기 유저 수 : 100

증가 유저 수 : 10

증가 간격 : 3 초

증가 횟수 : 5 번

테스트가 완료되었습니다!

Test 이름: Incremental Test
대상 URL: http://www.naver.com
초기 유저 수: 100 명
증가 유저 수: 10 명
증가 간격: 3 초
증가 횟수: 5 번

결과 확인하기

Result



Get (/results)

테스트 결과를 가져온다

```
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.naver.com:443
DEBUG:urllib3.connectionpool:https://www.naver.com:443 "GET / HTTP/1.1" 200 31251
DEBUG:root:Request to http://www.naver.com successful. Response time: 337.675 ms, Content length: 210318 bytes
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): www.naver.com:80
DEBUG:urllib3.connectionpool:http://www.naver.com:80 "GET / HTTP/1.1" 302 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.naver.com:443
DEBUG:urllib3.connectionpool:https://www.naver.com:443 "GET / HTTP/1.1" 200 44761
DEBUG:root:Request to http://www.naver.com successful. Response time: 337.3689999999999 ms, Content length: 262841 bytes
##### Average Response Time: 368.3482625 ms #####
##### Failure Rate: 0.0% #####
##### Load Duration: 0:02:04.343408 #####
```

```
"GET / testcase / 54 / execute / HTTP / 1.1" 200 OK
"GET / testcase / 54 / stats / HTTP / 1.1" 200 OK
```

Function </ testcase/Incremental >

— □ ×

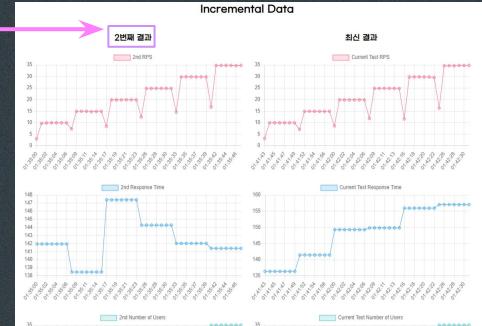
Serve

I Client



결과 비교:
○ test1 ● test2 ○ test3
○ test4 ○ test5

실행



Get
(stats/{selectedResult})

이전 테스트 결과 가져오기
최신 테스트 결과 가져오기

```
[{2, 3214, 54, Decimal('4.44'), Decimal('0.00'), Decimal('137.25'), 100, datetime.timedelta(seconds=7732)}, {2, 3215, 54, Decimal('93.38'), Decimal('0.00'), Decimal('137.25'), 100, datetime.timedelta(seconds=7733)}, {2, 3216, 54, Decimal('0.00'), Decimal('137.25'), 100, datetime.timedelta(seconds=7733)}, {2, 3217, 54, Decimal('32.98'), Decimal('0.00'), Decimal('138.26'), 110, datetime.timedelta(seconds=7733)}, {2, 3218, 54, Decimal('0.00'), Decimal('138.26'), 110, datetime.timedelta(seconds=7733)}, {2, 3219, 54, Decimal('84.07'), Decimal('0.00'), Decimal('138.26'), 110, datetime.timedelta(seconds=7733)}, {2, 3220, 54, Decimal('0.00'), Decimal('137.54'), 120, datetime.timedelta(seconds=7733)}, {2, 3221, 54, Decimal('100.90'), Decimal('0.00'), Decimal('137.54'), 120, datetime.timedelta(seconds=7733)}, {2, 3222, 54, Decimal('77.44'), Decimal('0.00'), Decimal('137.54'), 120, datetime.timedelta(seconds=7733)}, {2, 3223, 54, Decimal('17.95'), Decimal('0.00'), Decimal('137.71'), 130, datetime.timedelta(seconds=7749)}, {2, 3224, 54, Decimal('17.95'), Decimal('0.00'), Decimal('137.71'), 130, datetime.timedelta(seconds=7749)}, {2, 3225, 54, Decimal('0.00'), Decimal('137.71'), 130, datetime.timedelta(seconds=7750)}, {2, 3226, 54, Decimal('44.1'), Decimal('0.00'), Decimal('137.71'), 130, datetime.timedelta(seconds=7751)}, {2, 3227, 54, Decimal('115.89'), Decimal('0.00'), Decimal('137.71'), 130, datetime.timedelta(seconds=7752)}, {2, 3228, 54, Decimal('0.00'), Decimal('137.26'), 140, datetime.timedelta(seconds=7755)}, {2, 3229, 54, Decimal('0.00'), Decimal('137.26'), 140, datetime.timedelta(seconds=7755)}, {2, 3230, 54, Decimal('129.64'), Decimal('0.00'), Decimal('137.26'), 140, datetime.timedelta(seconds=7755)}, {2, 3231, 54, Decimal('0.00'), Decimal('137.71'), 150, datetime.timedelta(seconds=7760)}, {2, 3232, 54, Decimal('156.51'), Decimal('0.00'), Decimal('137.71'), 150, datetime.timedelta(seconds=7760)}, {2, 3233, 54, Decimal('139.52'), Decimal('0.00'), Decimal('137.71'), 150, datetime.timedelta(seconds=7761)}, {2, 3234, 54, Decimal('0.00'), Decimal('137.71'), 150, datetime.timedelta(seconds=7761)}]
```

"GET /testcase/54/stats/2 HTTP/1.1" 200 OK

— □ ×

↑

Result Demo

— □ ×

05

— □ ×

÷ ▶ ↓ ↑





Project Review



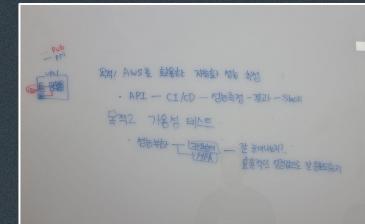
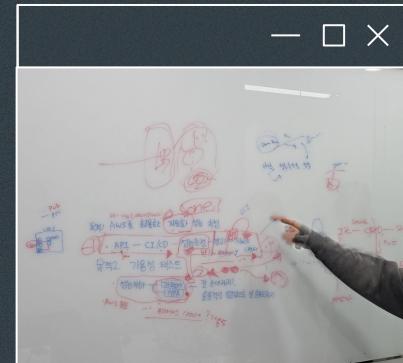
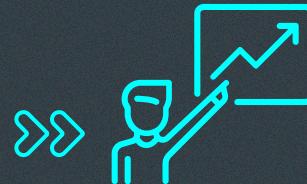
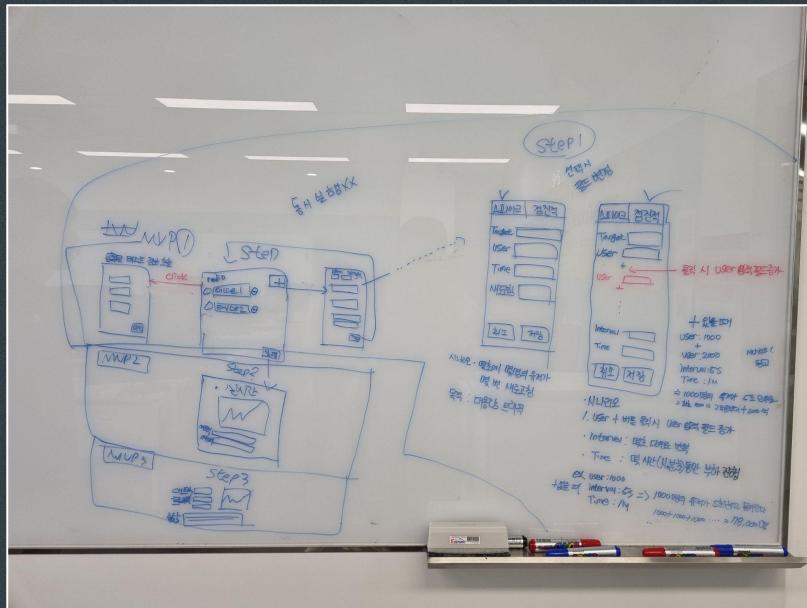
Review

— □ ×

	프로젝트 기간 동안 느낀점	프로젝트 기간 동안 아쉬운점
 김정우	<p>프로젝트를 통해 새로운 기술과 도구를 배우고 적용할 수 있는 기회였습니다. 테라폼을 활용한 AWS 자동화 환경 구축과 CI/CD 파이프라인 구성 등을 통해 DevOps 업무를 접할 수 있었고, 이를 통해 배포 자동화와 코드 품질 관리에 대한 중요성을 체감할 수 있었습니다.</p>	<p>목표를 설정하고 완료하는 과정에서 여러 가지 어려움이 있었습니다. 정해진 기간 내에 주어진 티켓을 해결해야 했지만, 예상보다 다양한 문제가 발생하였습니다. 이를 해결하기 위해 많은 트러블슈팅 과정을 거치면서 일정이 지연되었습니다. 여러 목표를 완료하지 못한 점은 아쉽지만, 이 과정을 통해 많은 성장을 이룬 것 같습니다.</p>
 백승현	<p>업무 분포를 하기 때문에 인프라 설계 구성에 힘을 많이 쓸을 것 같았지만, 예상외로 개발 참여도가 있었고 흥미가 생겼습니다. 개발에 흥미를 일깨워준 원우님 하경님 감사합니다. P.S. 팀원들이 서로 응원해준 덕분에 즐기며 작업할 수 있었습니다!</p>	<p>프로젝트를 시작하며 추구했던 목표에 100% 달성할 수 없었던 부분이 아쉬웠지만, 과거에 진행했던 프로젝트보다 큰 규모의 작업을 진행해서 다양한 부분에서 많은 것들을 배울 수 있었다. 예를 들어 체계적 팀 협업을 위한 지라, 슬랙 등에 사용 및 짓을 통한 공유 레파지토리에 분야를 나누어 코드를 풀 리퀘스트 해보는 것들이 있었던 것 같다.</p>
 이원우	<p>백엔드 API를 만들어보면서 API의 동작 원리에 대해 배울 수 있었습니다. 또한 리액트를 처음 사용해보며 컴포넌트 단위로 나누어 재사용에 용이하다는 것을 알게 되었고, 컴포넌트 간의 유연한 연결이 UI를 통해 잘 나타나는 것을 보며 재미있게 작업할 수 있었습니다.</p>	<p>프론트에서 백엔드로의 API 요청을 보내 처리하는 것은 원하는 대로 처리가 가능했지만, 백엔드에 DB의 수정이 이뤄진 후 해당 내용을 프론트에서 처리하는 데 어려움이 있었습니다. 웹 소켓과 같은 양방향 통신에 대해 공부해서 개선하고 싶습니다.</p>
 윤하경	<p>비동기 방식으로 <code>gevent</code> 라이브러리를 활용하여 많은 수의 동시 요청을 처리하는 테스트를 구현하는 목적을 달성할 수 있었다.</p>	<p>네트워크 지연, 타임아웃 등 세부적인 오류 원인을 구분하여 로깅하고 대응하는 구조를 통해 더욱 효과적인 테스트를 개선하고 싶다.</p>

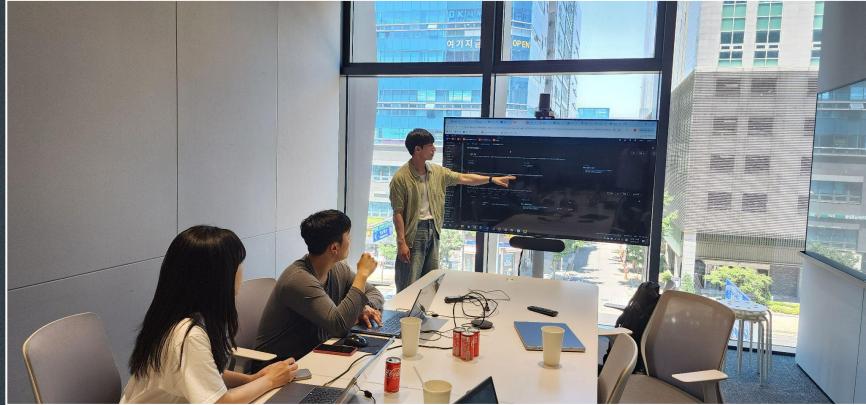
Epilogue

— □ ×



Epilogue

— □ ×



Thank you

⌘ ⌂ # ▽

◀

÷ ⌂ ⌄

— □ ×

— □ ×