


캡스톤 디자인 I 종합설계 프로젝트

프로젝트 명	GIT WATCHER
팀 명	GET
문서 제목	결과보고서

Version	1.2
Date	2018-MAY-29

팀원	전 호 현(조장)
	유 문 상
	최 원 대
	이 근 하

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 "**Git Watcher**"를 수행하는 팀 "**GET**"의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 "**GET**"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	결과보고서-Git Watcher.doc
원안작성자	전호현
수정작성자	전호현

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2018-05-27	전호현	1.0	최초 작성	
2018-05-28	전호현	1.1	내용 수정	
2018-05-29	전호현	1.2	내용 수정	

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

목 차

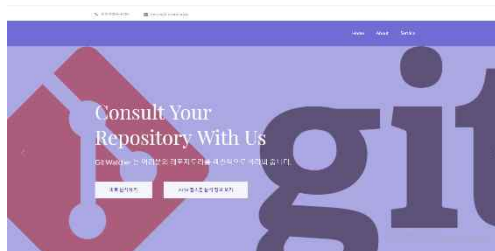
1	개요	4
1.1	프로젝트 개요	4
1.2	추진 배경 및 필요성	4
1.2.1	추진 배경	4
1.2.2	Git Inspector의 장점	5
1.2.3	Git Inspector의 문제점	5
1.2.4	Git Watcher의 필요성	6
2	개발 내용 및 결과물	7
2.1	목표	7
2.2	연구/개발 내용 및 결과물	7
2.2.1	연구/개발 내용	7
2.2.1.1	Github repository의 변동사항을 알려줄 API 탐색	7
2.2.1.2	AWS Backend Architecture 구축	7
2.2.1.3	Git inspector 기능 파악	9
2.2.1.4	GitHub repository의 push 인식하여 Amazon S3에 코드 저장, 유지	9
2.2.1.5	AWS EC2에서 Git inspector 분석 수행	10
2.2.1.6	Git push 시 Git inspector 분석 html파일 저장	11
2.2.1.7	Git inspector 기능 추가	12
2.2.1.8	웹 페이지 구축	14
2.2.2	시스템 기능 요구사항	15
2.2.3	시스템 비기능(품질) 요구사항	16
2.2.4	시스템 구조 및 설계도	17
2.2.5	활용/개발된 기술	18
2.2.6	현실적 제한 요소 및 그 해결 방안	20
2.2.7	결과물 목록	21
2.3	기대효과 및 활용방안	22
2.3.1	기대효과	22
2.3.2	활용방안	22
3	자기평가	23
4	참고 문헌	23
5	부록	24
5.1	사용자 매뉴얼	24
5.2	테스트 케이스	30

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

1 개요

1.1 프로젝트 개요

본 프로젝트(이하 Git Watcher)는 Git repository에 대한 분석을 진행하여 그 결과를 사용자에게 제공해 주는 웹 서비스입니다.




Git Watcher에서는 크게 Git repository에 기여한 Contributor 각각의 커밋 횟수, 삽입 및 삭제 라인 수, 이슈 생성 개수 그리고 자체적으로 계산한 Github 활동점수를 제공합니다. 이외에도 너무 적은 라인 변화가 일어난 커밋만 뽑아서 보여주는 기능, 사용자 별 활동 점수를 기간별로 나누어 시각화하여 보여주는 기능 등을 제공합니다. 사용자는 이를 통해 스스로의 Github repository 사용에 대한 정보를 얻을 수 있습니다.

1.2 추진 배경 및 필요성

1.2.1 추진 배경



그림 1. Github

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

Github는 가장 보편적으로 사용되는 버전 관리 시스템 중 하나이다. 수많은 오픈소스 프로젝트들이 Github의 저장소에 존재한다. 그리고 각각의 오픈소스 프로젝트들은 다수의 Contributor(기여자)들로부터 만들어진다. 우리 팀은 Github 저장소에 존재하는 프로젝트의 분석을 Contributor들에게 제공하여 그들이 실제 프로젝트 코드에 얼마만큼 기여했으며, Github를 얼마나 잘 사용했는지를 알 수 있도록 하는 서비스를 만들고자 했다. 그러던 중, 각 Contributor들의 깃허브 활용 정도를 보여주는 오픈소스 프로젝트, Git Inspector를 찾게 되었고, 이 시스템의 문제점을 보완하고 필요한 기능을 추가하여 평가에 적합한 서비스를 제공하고자 한다.

1.2.2 Git Inspector의 장점

```

+--[1mAuthor          Commits    Insertions    Deletions    % of changes+--[0;0m
DESKTOP-8DP01N2#one1    10         3239         195         36.28
DESKTOP-G7MQ1D1#jhh5     2          122          3          1.32
HoHyun Jun             10        2788        1171        41.82
SANG YUN LEE           2           43          43          0.91
SangYunLEE             1          258          72          3.49
Unknown                2        1522          10         16.18

```

Below are the number of rows from each author that have survived and are still intact in the current revision:

```

+--[1mAuthor          Rows      Stability    Age    % in comments+--[0;0m
DESKTOP-8DP01N2#one1  2895      89.4         0.5     5.08
HoHyun Jun           1615      57.9         0.0     4.33
SangYunLEE           992       384.5        0.0     0.91

```

위 그림은 Git inspector를 실행한 결과화면이다. 그림에서 볼 수 있듯이, Git inspector는 Github repository에 있는 오픈소스 프로젝트 기여자들의 Github 활동에 대한 다양한 정보를 제공한다. 각 기여자들의 Commit 수, 코드 삽입 라인 수, 코드 삭제 라인 수, 전체 코드 중에서 몇 퍼센트의 변화를 주었는지 등의 정보를 제공하는 것을 위 그림에서 볼 수 있다.


1.2.3 Git Inspector의 문제점

```

C:\Users\HohyunJun\Desktop>git inspector master
$ ./gitinspector.py -F html https://github.com/hohyunjun/DiningCode
Opening into 'C:\Users\HohyunJun\AppData\Local\Temp\tmp55ng2xt\gitinspector'...
remote: Counting objects: 1005, done.
Receiving objects: 29% (292/1005), 1.19 MiB | 1.09 MiB/s  s

```

위 그림은 실제 Git inspector를 실행시키는 과정이다. 특정 repository에 대한 분석을 하려면 위와 같은 코드를 분석을 진행하고 싶을 때마다 터미널에 명령어를 입력해야 한다. 사용자가 사용하기에 불편하고, 코드 오타로 인해 오류가 생길 가능성도 크다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

```
jhh51@DESKTOP-G7M01D1 MINGWB4 ~/Desktop/호현/gitinspector-master
$ ./gitinspector.py -F http://github.com/kebyunjun/DiningCode
Cloning into 'C:\Users\jhh51\AppData\Local\Temp\tmpg5wg2pxt.gitinspector'...
remote: Counting objects: 1005, done.
Receiving objects: 29% (292/1005), 1.19 MiB | 1.09 MiB/s  s
```

뿐만 아니라, Git inspector를 실행하면 실행할 때마다 사용자가 원하는 Github repository 로 부터 Cloning을 받아와야 한다. Repository를 Cloning을 하는 시간은 Repository에 있는 코드 정보 전체를 받아와야 하므로, 시간이 매우 오래 걸린다. 따라서 사용자는 분석을 위해 오랫동안 기다려야 한다.

1.2.4 Git Watcher의 필요성

우리 팀은 위에서 본 것과 같은 Git inspector의 장점은 살리고, 단점을 보완하여 사용자 친화적인 웹 서비스를 만들었다. Git inspector의 사용성을 개선하여, 사용자가 사용하기 쉬운 User Interface를 제공한다. 쉽게 레포지토리에 대한 분석을 할 수 있도록, 기존에 저장된 레포지토리 목록을 보여주고, 간단한 클릭 한 번으로 저장된 분석자료를 볼 수 있다.

또한, 매 실행 시마다 Cloning이 이루어져서 생기는 성능적인 문제를 해결하기 위해, Amazon S3에 Github 저장소의 코드를 저장하고 유지하고 있다. 이에 따라 사용자의 분석 요청이 들어올 때, 새로 코드를 Cloning 해올 필요성이 사라지고, 저장된 코드를 분석만 하면 되기 때문에 기존의 Git inspector 보다 시간이 크게 단축되었다. 그러므로 사용자는 보다 쉽고 빠르게 Github 활용도에 대한 정보를 얻을 수 있다.

그리고 Git inspector에는 없는 분석 항목을 추가하여 사용자에게 프로젝트에 대해 더 자세한 정보를 제공한다. 레포지토리에 올려진 Issue에 대한 정보나 branch에 대한 정보, 자체적으로 계산한 개인 별 활동 점수, 그리고 이에 대한 시각화 그래프 등을 제공한다. 이러한 분석 항목을 통해 사용자는 GitHub에 있는 더 많은 기능들을 알게 되며, 이를 더 효율적으로 사용할 수 있을 것이다.

Git Watcher는 Back-end 부분이 AWS Lambda를 활용한 Serverless Architecture로써, 서비스의 관리 측면에서 기존 아키텍처보다 간편하다. “서버의 존재”에 대해 신경 쓰지 않아도 된다. 즉, 서버가 어떤 사양으로 돌아가고 있는지, 서버의 개수를 늘려야 할지, 네트워크를 어떤걸 사용할지 등을 설정할 필요가 없다. 뿐만 아니라, 특정 작업을 하기 위하여 서버를 준비하고 하루 종일 켜놓는 것이 아니라, 필요할 때만 함수가 호출되어 처리되며 함수가 호출된 만큼만 비용이 들기 때문에, 비용이 많이 절약된다. 또한, 트래픽이 늘어남에 따른 확장성도 매우 뛰어나다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2 개발 내용 및 결과물

2.1 목표

사용자가 Git repository를 더 효율적으로 사용할 수 있도록 돕는 분석 도구를 개발한다.

본 프로젝트의 목표는 기존에 존재하는 오픈소스 프로젝트인 Git Inspector를 보완하여, GitHub를 활용해 프로젝트를 진행한 참여자가 GitHub를 더 효율적으로 사용할 수 있도록 Git repository의 분석 정보를 제공하는 것이다. Git Inspector는 git repository에 대한 통계적인 분석 도구이다. Git inspector에서 제공해주는 정량적인 분석 결과를 활용하여, 사용자가 얼마나 Github를 효율적으로 사용하고 있는지를 보여주는 활동 점수를 만들어 제공한다. 이 점수를 높이려고 노력하는 과정에서 사용자는 Github를 더 효율적으로 사용할 수 있다.

Git Inspector의 보완의 경우, 명령어 Command를 사용해야 하는 Git Inspector를 UI를 제공하여 편리하게 사용할 수 있으며, Amazon S3에 코드 정보를 미리 저장하여 더 빠른 분석결과를 제공한다. 더 나아가 Git Inspector에서는 제공하지 않는 기능들도 제공된다.

이를 통해 사용자는 보다 쉽고 빠르며, 편하게 자신이 GitHub를 얼마나 잘 활용하고 있는지에 대한 정보를 제공받을 수 있다. 프로젝트에 대한 커밋 횟수, 변화 수 등의 상세 분석 결과를 바탕으로 더 효율적으로 GitHub를 사용할 수 있다.

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

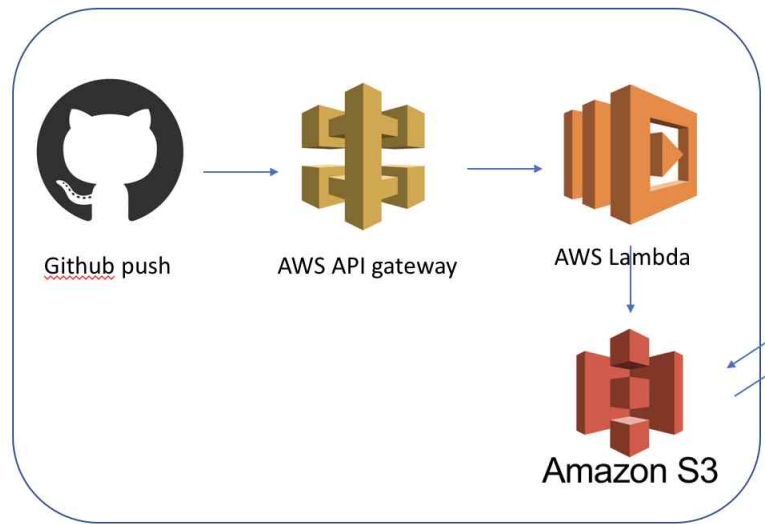
2.2.1.1 Github repository의 변동사항을 알려줄 API 탐색

지정된 Github 레포지토리의 활동을 계속 주시하여 새로운 push가 일어날 경우에, 데이터베이스에 있는 레포지토리 코드를 업데이트 할 수 있어야 한다. 이를 위해서는 Github 프로젝트의 변동사항을 알려줄 API가 필요하다. Github API 중에서, webhook 이라는 API를 활용하여, 특정 저장소에서 push가 일어나는지를 감시하도록 하였다.

2.2.1.2 AWS Backend Architecture 구축

회의를 통해 구상한 Pipeline 아키텍처를 구축한다. 우리 팀이 만들고자 하는 프로젝트의 아키텍처는 크게 두 가지로 나뉜다. 첫번째는 항상 최신의 레포지토리 상태를 유지시킬 수 있는 아키텍처이다. 사전에 주어진 GitHub repository에서 push가 일어났을 경우 부터 시작하여, 이것을 API gateway와 연결하고, AWS Lambda와 연동시켜 Amazon S3에 업데이트된 코드를 저장시킬 수 있도록 구축하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29



항상 최신의 레포지토리 상태 유지

두 번째 아키텍처는 사용자의 요청이 들어올 때마다 분석을 수행하는 아키텍처이다. 원래는 AWS Lambda에 Git inspector를 올려서 사용자의 요청 시에만 Lambda가 작동하여 분석작업을 하게 함으로써, 완벽한 Serverless Architecture를 만들고자 했다. 그런데, Git inspector의 요구사항인 Git이 Lambda에 깔려있지 않아서 packaging 작업에 문제가 생겼다. AWS Lambda의 경우, 기본 리눅스 환경에서 제공하지 않는 라이브러리가 필요한 경우에는 적합하지 않다는 사실을 알게 되었고, 결국 AWS EC2에 Git inspector를 올려서 분석 작업을 진행하는 것으로 아키텍처를 변경하였다.



사용자 요청이 들어올 때 마다 분석
그림1. 기존의 아키텍처



사용자 요청이 들어올 때 마다 분석
그림2. 변경된 아키텍처

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2.2.1.3 Git inspector 기능 파악

Git Inspector를 활용하여 코드를 분석할 것이므로, 기존의 Git Inspector가 어떤 기능을 제공하는지를 완벽하게 파악한다. Git Inspector 저장소에 있는 Documentation을 활용하여 그 기능을 직접 수행하면서 기능을 파악한다.

기본 정보

- 커밋 횟수
- 삽입 라인
- 삭제 라인
- 변화 수
- 나이 = 유저가 적은 모든 줄의 평균 지난 일수,
- 안정성 = 유저가 적은 라인이 지금까지 살아남아 있는 비율,
- 코멘트 비율 = 전체 줄 수 대비 주석 비율.

옵션


- `--format`: 분석 결과를 내가 원하는 파일로 만들어 줌 = html,json 등등.
- `--grading`: 더 자세한 결과를 보여줌.
날짜별 소스코드 생산 라인 수(timeline),
어떤 소스코드에 가장 많이 커밋 했는가.
- `-f`: 특정 확장자 파일만 검색.
- `-H`: 중복을 더 엄격하게 찾음?
- `-m`: 순환 복잡도 체크

이하 4개를 조합해서 날짜 조건 주게 할 수 있을듯 함.

- `-T`: 월별 커밋 횟수
- `-w`와 같이 쓰면 주별로 변경 가능.
- `--since=날짜`: 특정 날짜 이후 커밋만 검색
- `--until` = 특정 날짜 이전

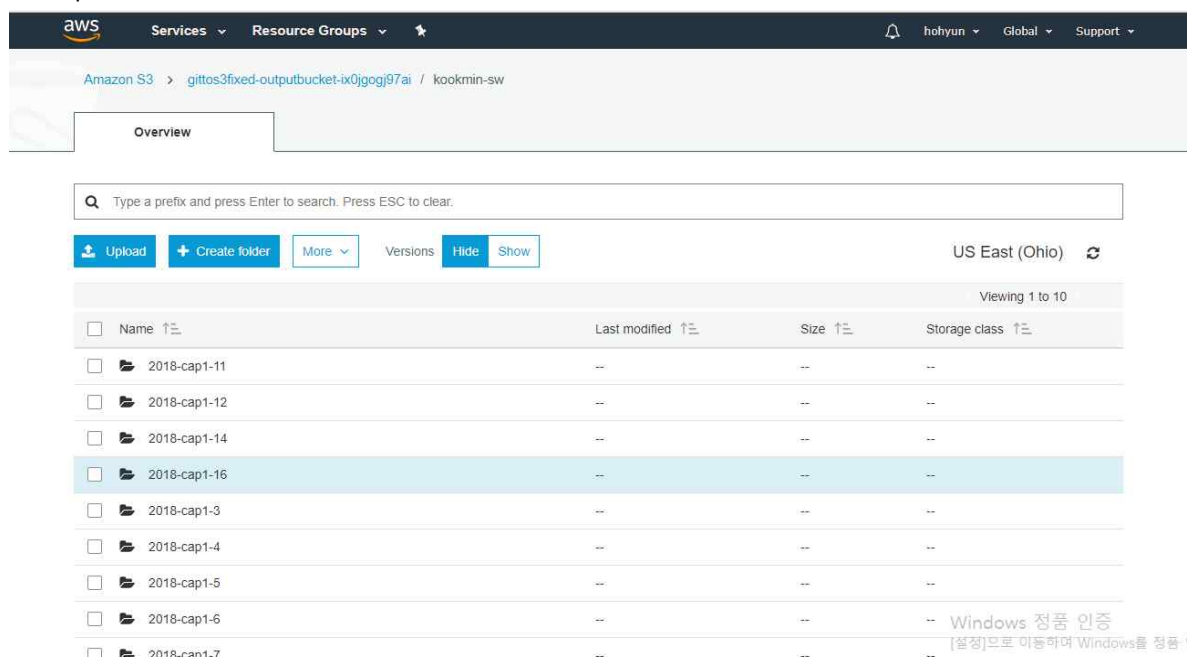
2.2.1.4 GitHub repository의 push 인식하여 Amazon S3에 코드 저장, 유지

실행 시마다 Github repository의 Cloning이 일어나는 Git inspector의 문제점을 해결하기 위해, 특정 Github repository의 코드를 미리 데이터베이스에 저장하고 유지할 수 있도록 한다. 위에서 찾은 API를 AWS Lambda에 올려놓고, API로부터 신호가 올 때마다 AWS S3 데이터베이스에 있는 코드를 GitHub repository의 코드로 업데이트한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

Github repository의 push를 인식할 API인 webhook으로부터 POST요청을 받았을 때, 레포지토리의 코드를 클론해 올 수 있도록 AWS 구조를 구축했다. 먼저, 사용자의 Github push를 인식하는 webhook이 POST요청을 보낼 수 있는 주소를 API Gateway를 통해 만들었고, 이 API Gateway로 POST요청이 왔을 때, AWS Lambda를 수행할 수 있도록 구성했다. AWS Lambda에서는 webhook을 보낸 repository의 코드를 git clone하여 Amazon S3의 버킷에 저장하는 역할을 한다.


위와 같이 AWS Architecture를 구현한 이후, 생성된 AWS API Gateway를 Github repository settings의 webhook에 등록했다. 이제 repository에 push가 일어날 때마다 Amazon S3에 zip파일의 코드가 업데이트되어 저장된다. 아래 그림은 현재 캡스톤 디자인을 진행하는 팀들에게 webhook 등록을 공지하여 코드파일이 Amazon S3에 잘 저장된 모습이다.



2.2.1.5 AWS EC2에서 Git inspector 분석 수행

Git inspector를 EC2에서 실행시키기 위해, AWS EC2 서버를 만들고, Git inspector를 모듈화시키고, 수정했다. 그리고 S3에 저장된 코드 zip 파일을 불러올 수 있도록 python 코드를 작성했다.

결과적으로 사용자에게 보여줄 것은 html파일이므로, Git inspector의 디폴트 출력값을 html로 변경했다. 그리고 Git inspector를 간단히 불러 사용할 수 있도록 모듈화했다. S3에 저장된 코드 zip 파일을 가져오기 위해서, python에 있는 boto 라이브러리를 사용하여 S3 버킷에 연결하고 가져오는 코드를 작성했다. 이 과정을 다 수행하고 나서, 결과적으로 Django 프레임워크를 활용하여 웹 페이지를 시연해 본 결과, 생각했던 것보다 분


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

석 결과값이 나오는 속도가 느렸다. 이에 따라 기존에 Git push가 일어났을 때 코드를 저장하는 것에서 더 나아가, EC2 서버에 있는 Git inspector에서 분석작업까지 마친 후 결과 HTML 파일을 EC2에 미리 저장해 놓는 방식으로 구조를 변경하기로 하였다.

2.2.1.6 Git push 시 Git inspector 분석 html파일 저장

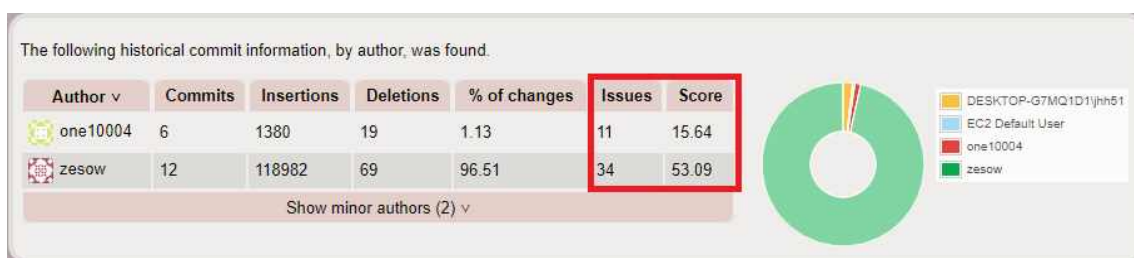
Git push -> S3에 코드 zip 파일로 저장 -> EC2에 있는 Git inspector가 S3에 새로운 zip 파일이 올라왔음을 인식 -> 새로운 zip 파일 가져와서 분석 -> EC2 내부에 결과파일 저장의 플로우를 구현했다. AWS S3에 repository가 업데이트되면 aws sns 서비스가 ec2에 알람을 보내준다. Ec2에는 알람을 받아서 s3에서 repository를 다운로드, 분석하여 html로 뽑아주는 서버가 구동 중이다. 분석 중에 다른 알람이 들어와서 알람을 놓치는 상황을 방지하기 위해 멀티 쓰레딩으로 구현하였다. 최종적으로 사용자는 웹 페이지에서 미리 저장되어 있는 HTML 파일의 리스트를 조회할 수 있고, 리스트 중 하나를 선택하면 최소한의 대기시간으로 최신의 분석결과를 볼 수 있다. 아래의 사진은 저장된 HTML 파일들을 보여준다.

```
[ec2-user@ip-172-31-2-76 html]$ dir
kookmin-sw_2018-cap1-12_branch_master.html
kookmin-sw_2018-cap1-14_branch_master.html
kookmin-sw_2018-cap1-16_branch_db.html
kookmin-sw_2018-cap1-16_branch_web.html
kookmin-sw_2018-cap1-4_branch_dPremodel.html
kookmin-sw_2018-cap1-4_branch_dPreturn.html
kookmin-sw_2018-cap1-4_branch_Function_receptionPage_patientInfo.html
kookmin-sw_2018-cap1-4_branch_generalSurvey.html
kookmin-sw_2018-cap1-4_branch_master.html
kookmin-sw_2018-cap1-4_branch_setting_drools.html
kookmin-sw_2018-cap1-4_branch_UI_hospitalSurvey.html
kookmin-sw_2018-cap1-6_branch_frontend2.html
kookmin-sw_2018-cap1-6_branch_gi_lambda.html
kookmin-sw_2018-cap1-6_branch_gitInspector_Server.html
kookmin-sw_2018-cap1-6_branch_master.html
kookmin-sw_2018-cap1-8_branch_master.html
[ec2-user@ip-172-31-2-76 html]$
```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2.2.1.7 Git inspector 기능 추가


기존의 Git inspector에는 없던 각 Contributor 별로 Issue를 뽑아내서 보여주는 기능과, 개인 별 활동점수를 각각 계산하여 보여주는 기능을 추가하였다. Issue의 경우, 개인 별로 개설한 Issue + Comment 횟수를 합산한 값을 보여주도록 하였고, github API를 사용하여 구현하였다. 개인 별 점수는 (개인별 커밋 + 이슈 + 이슈 Comment / 전체 커밋 + 이슈 + 이슈 Comment) *100 공식으로 기존에 있는 데이터를 활용하여 도출하였다. 아래 그림은 기존의 화면에 추가한 기능을 보여준다.



사용자가 Score를 높이기 위해 의미가 없는 커밋을 하는 Abusing을 어떻게 Score를 뽑아내는 과정에서 제외시킬 수 있을지 고민하다가, 코드 라인의 변화 수가 -10~+10 인 커밋을 따로 뽑아 그 변화 내용을 볼 수 있도록 구현하였다. 코드 라인의 변화 수가 적다고 해서 무조건 의미가 없는 커밋은 아니라고 생각하였기 때문에, 직접적으로 Score에 영향을 주진 않았다. 아래는 추가한 기능의 실행화면이다. Commit 메시지와 변화된 라인 수, 그리고 그 변화된 라인이 의미가 있는지 없는지를 판단하기 위한 링크가 제공된다.

Commit that has low line changes.

Commit hash	Commit Msg	Change	Link
6c3f07d	Set theme jekyll-theme-cayman	2	Click
4edaa67	영상 썸네일 이미지 추가	2	Click
d3df90a	Update index.md	4	Click
63488a9	Add interim ppt, pdf files	2	Click
aaa598d	중간보고서 내용 추가 및 아키텍처 수정	2	Click
8415114	ms	2	Click
19d364c	중간평가 보고서 초안 업로드, 계획서 Usecase Diagram 수정	4	Click
3a92209	cleaning	3	Click
1109de1	thread added	4	Click
afee941	Create ttest9	1	Click
ebdc58e	Create ttest8	1	Click
1141b2a	Create ttest6	1	Click
0182b51	Create ttest5	1	Click
2f78722	Create ttest4	1	Click
18a8e2b	Create okokok	1	Click
e10fc2c	Create testqs6	1	Click
faa1f85	Create testqs5	1	Click

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

사용자 별 활동점수를 시각적으로 한 눈에 볼 수 있도록 표현하기 위해 그래프를 통해 나타내는 기능을 추가하였다. 원래는 일별, 월별, 주별로 각각 사용자의 점수를 비교하는 그래프를 보여주는 기능을 추가하려고 했었다. 이에 대한 데이터를 추출해 내는 것까지는 성공하였으나, 그래프를 활용하여 보여주는 부분은 월별 개인 점수를 보여주는 것 밖에 구현하지 못했다. 아래 그래프의 x축은 월, y축은 점수, 그리고 각각의 그래프 색깔이 user 개개인을 뜻한다. 즉, 2월부터 5월까지 개인별 점수를 한 눈에 비교할 수 있도록 보여주는 그래프이다.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

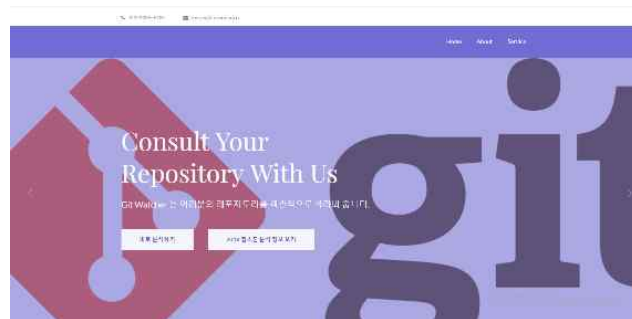
2.2.1.8 웹 페이지 구축


Git Watcher는 사용자에게 사용하기 편리한 User Interface를 제공할 수 있어야 한다. 사용자에게 Amazon S3에 저장된 저장소의 목록을 보여줄 수 있도록 User Interface를 구성한다. 또한, 분석 결과로 보여지는 기본 정보인 커밋 횟수, 삭제 라인, 변화 수 등을 기준으로 Contributor들을 정렬하여 보여줄 수 있어야 한다.

또한, 사용자의 요청에 따라 새롭게 분석을 진행하여 사용자에게 보여줄 수 있어야 한다. 기존 Amazon S3에 저장되어 있지 않은 repository에 대해서도 분석이 가능하도록, 사용자로부터 repository의 URL을 받을 수 있도록 하는 User Interface가 만들어져야 한다. 사용자의 실시간 분석 요청에 대해서도 반응할 수 있도록 하는 User Interface가 필요하다.

사용자의 요청을 받을 웹 페이지를 Twitter Bootstrap을 이용해 스타일링하여 제작하였다. 웹 페이지의 첫 화면에서 사용자는 github URL을 입력하여 직접 clone해서 분석하거나, Webhook을 등록한 사용자의 경우에는 저장된 분석 결과 확인 버튼을 눌러 결과 HTML을 바로 확인할 수 있다. 저장된 분석 결과 확인의 경우 Clone 해서 분석하는 것보다 훨씬 빠르게 사용자에게 분석 결과를 제공한다.

이렇게 만들어진 웹 페이지는 Django framework를 활용하여 EC2에서 서비스되도록 하였다.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

7. Github Repository 분석(완료)

- 6에서 Cloning해온 Github Repository를 분석하여 그 분석결과를 html형태로 사용자에게 보여줄 수 있다.

8. Github URL 목록에 있는 Repository code & 분석 HTML update(완료)

- Webhook이 등록된 repository의 사용자가 해당 repository에 push를 할 경우, AWS Lambda에서 이를 인식하여 해당 코드를 zip파일 형태로 Amazon S3에 저장한다. 그리고 Amazon S3로부터 EC2가 신호를 받아서, 이 zip파일에 대한 분석결과를 미리 HTML 파일 형태로 저장해 놓는다.

2.2.3 시스템 비기능(품질) 요구사항


1. 신뢰성과 이용가능성 (달성)

특정 Github Project 에 기여한 사람들의 정보를 가지고 그 기여도를 평가하는 것이 목적인 만큼, 정보를 정확하게 분석하는 것이 중요하다. 서비스의 분석 결과는 Github repository 에 기반하여 100%의 정확도를 보장할 수 있으며, 사용자가 원할 때 언제든지 사용 가능하다.

2. 성능/처리량 (미달성)

기존의 Git inspector 어플리케이션이 분석을 할 때마다 저장소를 cloning 하여 분석을 하였기 때문에, 분석 속도가 매우 느렸었다. 분석 속도는 프로젝트의 크기가 크면 클수록 더 늘어났었다. 따라서 이 시스템의 성능은 최소한 기존의 Git inspector 보다 빨라야 하며, GitHub Repository 를 유지하고 있는 경우에 프로젝트의 크기에 따른 분석 시간의 차이가 최소화되어야 한다. GitHub Repository 를 유지하고 있는 경우 응답시간은 최대 3 초를 넘지 않으며, GitHub 에서 바로 Cloning 하여 분석하는 경우의 응답시간은 최대 30 초를 넘지 않는다.

애초에 목표했던 GitHub Repository 를 유지하고 있는 경우의 응답시간은 3 초를 넘지 않는다. 하지만 GitHub 에서 바로 Cloning 하여 분석하는 경우의 응답시간은 최대 30 초 이상이 걸린다. 이 부분의 경우, Repository 크기가 커짐에 따라 Git inspector 의 분석 시간이 길어지는 것이 원인이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

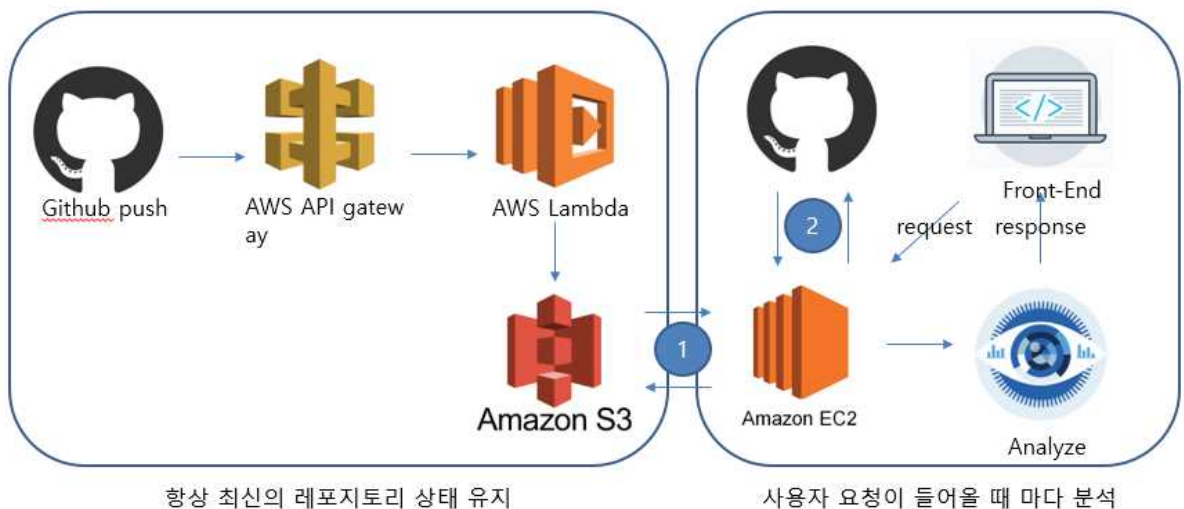
3. 사용성(달성)

인터페이스 구성에 있어 사용자가 손쉽게 메뉴를 찾을 수 있도록 구성하여야 한다. 사용자는 구성된 Interface만을 보고도 이것이 어떠한 기능을 수행할 것인지를 인지할 수 있어야 하며, 수행하고자 하는 기능을 불편함 없이 사용할 수 있어야 한다.

4. 적응성(Adaptability)(달성)

Git Watcher 서비스는 네트워크가 연결된 어느 하드웨어에서나 사용이 가능하다.


2.2.4 시스템 구조 및 설계도



위 그림이 Git Watcher 아키텍처의 최종 버전이다. 처음에 계획하였던 아키텍처는 항상 최신의 레포지토리 상태를 유지하는 부분 뿐만 아니라 사용자 요청이 들어올 때 마다 분석하는 부분도 Lambda 를 활용한 완벽한 Serverless Architecture 였었다. 하지만, AWS Lambda 에 Git inspector 를 패키징 하는 과정에서 문제가 발생했다.

AWS Lambda 의 경우, 기본 리눅스 환경에서 제공하지 않는 라이브러리가 필요한 경우에는 패키징에 적합하지 않았다. 이 말인 즉슨, Git 을 사용하는 Git inspector 를 Lambda 에 패키징하기 적합하지 않다는 것이다.

그래서 우리 팀에서는 상의 끝에 AWS EC2 에 수정한 Git inspector 를 올려서 레포지토리에 대한 분석작업을 하는 것으로 아키텍처를 변경하였다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2.2.5 활용/개발된 기술

1. Amazon Web Service

a. API Gateway

AWS API Gateway는 어떤 규모에서든 개발자가 API를 생성, 게시, 유지 관리, 모니터링 및 보호할 수 있게 해주는 AWS 서비스이다. AWS 또는 다른 웹 서비스, 그리고 AWS 클라우드에 저장된 데이터에 액세스하는 API를 생성할 수 있다.

Git Watcher에서는 이 API Gateway를 활용하여, 사용자의 Git repository에 Webhook을 걸어 repository에 변경이 일어났음을 인지할 수 있다. 그리고 이 신호를 AWS Lambda에게 보내주는 역할을 한다.

b. Lambda

AWS Lambda는 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해 주는 컴퓨팅 서비스이다. AWS Lambda는 필요 시에만 코드를 실행하며, 하루에 몇 개의 요청에서 초당 수천 개의 요청까지 자동으로 확장이 가능하다. 사용한 컴퓨팅 시간에 대해서만 요금을 지불하면 되고 코드가 실행되지 않을 때는 요금이 부과되지 않는다.

우리 팀에서는 이 AWS Lambda를 Serverless Architecture의 핵심적인 기능을 수행하는 데 사용했다. API Gateway로부터 사용자의 repository에 변경이 일어났다는 신호를 받으면, Lambda는 이 응답으로 해당 사용자의 repository 코드를 AWS S3로 zip파일 형태로 다운로드 하는 역할을 한다. 즉, 사용자의 push 이벤트에 의해 Lambda가 트리거되고, 이 때만 코드가 실행된다.

c. S3

Amazon S3는 Simple Storage Service의 약자로, 인터넷용 스토리지 서비스이다. 개발자가 보다 쉽게 웹 규모 컴퓨팅 작업을 할 수 있도록 설계되었다. Amazon S3에서 제공하는 단순한 웹 서비스 인터페이스를 사용하여 웹에서 언제 어디서나 원하는 양의 데이터를 저장하고 검색할 수 있다. 높은 확장성과 신뢰성을 갖추고 있다.

우리 팀에서는 Amazon S3를 사용자의 코드를 저장하는 저장소로써 사용했다. 그리고 이 S3에 유지되던 코드가 업데이트 되었을 때, AWS SNS 서비스를 이용하여 EC2에 신호를 보내 EC2에 있는 Git inspector가 자동으로 분석 자료를 저장하도록 구현했다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

d. SNS

Amazon SNS 는 Amazon Simple Notification Service의 약자이다. 이는 구독 중인 엔드포인트 또는 클라이언트에 메시지 전달 또는 전송을 조정 및 관리하는 웹 서비스이다.

Git Watcher는 S3에 코드가 업데이트 되었음을 이 SNS 서비스를 통해 인식하고, Git watcher가 구동중인 EC2에 신호를 보내 준다.

e. EC2

Amazon EC2는 Amazon Elastic Compute Cloud의 약자로, AWS 클라우드에서 확장식 컴퓨팅을 제공한다. Amazon EC2를 사용함으로써 하드웨어에 선투자할 필요가 없어 더 빠르게 애플리케이션을 개발하고 배포할 수 있다. Amazon EC2를 통해 원하는 만큼 가상 서버를 구축하고 보안 및 네트워크 구성과 스토리지 관리가 가능하다. Amazon EC2는 요건이나 갑작스러운 인기 증대 등 변동사항에 따라 확장하거나 축소할 수 있어 트래픽 예측 필요성이 줄어든다.

Git Watcher에서 EC2는 Git Watcher를 구동시키는 서버 역할을 한다. EC2에서는 Amazon S3로부터 알림을 받아서 S3로부터 repository를 다운로드, 후 Git inspector로 분석하여 이 파일을 HTML 형태로 저장해 놓는 역할을 한다.

2. Django


Django는 파이썬으로 만들어진 무료 오픈소스 웹 어플리케이션 프레임워크이다. 쉽고 빠르게 웹사이트를 개발할 수 있도록 돕는 구성요소로 이루어진 웹 프레임워크라고 볼 수 있다. MVC 패턴을 따르고, DRY 원리를 따른다.

Git Watcher는 Django를 통해 서비스된다. EC2 위에 Django 서버를 올려놓았고, 이 Django server에 들어오는 요청들을 Django에서 관리한다. Django는 서버에 들어오는 요청에 따라 적절한 HTML 페이지를 사용자의 웹 브라우저로 보내주는 역할을 수행한다.

3. Javascript d3

D3 또는 Data-Driven Documents는 웹 브라우저 상에서 동적이고 인터랙티브한 정보 시각화를 구현하기 위한 자바스크립트 라이브러리이다. D3는 SVG 와 HTML5, CSS 등 웹 표준에 기반해 구현되어 있다.

우리 팀에서는 이 D3 라이브러리를 활용하여 사용자에게 각 사용자별 점수를 시각화하여 보여줄 수 있는 그래프를 만들었다. 이를 통해 사용자는 한 눈에 자신의 점수를 같은 repository의 다른 Contributor들과 비교할 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

4. Twitter Bootstrap

Bootstrap은 웹사이트를 쉽게 만들 수 있게 도와주는 HTML, CSS, JS 프레임워크이다. 하나의 CSS로 휴대폰, 태블릿, 데스크탑까지 다양한 기기에서 작동한다. 다양한 기능을 제공하여 사용자가 쉽게 웹사이트를 제작, 유지, 보수할 수 있도록 도와준다.

우리 팀에서는 네트워크가 연결된 어떤 기기에서나 손쉽게 웹 서비스에 접근할 수 있으며, 생산성 높은 웹 페이지 개발 방법을 찾다가 이 Twitter Bootstrap을 사용하게 되었다. 이 프레임워크를 사용하여 웹 서비스의 페이지들을 디자인하고 배치하였다.

2.2.6 현실적 제한 요소 및 그 해결 방안

1. 직접 Repository URL을 입력해 실시간으로 분석하는 경우의 Scalability 문제


직접 Repository URL을 입력해 실시간으로 해당 저장소를 분석하는 경우, 바로 해당 Repository를 Cloning하고 분석까지 해야하므로, Repository의 크기가 커짐에 따라 그 실행 속도가 매우 느려질 수 밖에 없다.

이에 대한 해결 방법으로는, Git inspector 전체 코드에 대해 기존 알고리즘보다 더 빠른 속도를 낼 수 있는 알고리즘을 사용하여 수정하는 것이다. 이를 통해 최대 분석 시간이 30초 이내가 될 수 있도록 한다.

2. Webhook이 걸려있는 Repository로부터 동시다발적으로 push가 일어날 경우 업데이트 누락


AWS Lambda로 여러 개의 알람이 동시다발적으로 들어왔을 때, S3에 코드의 저장이 완벽하게 이루어지는지가 불투명하다. 또한, S3에 코드가 저장되었음을 알려주는 SNS 신호를 받는 EC2에서 이 알람을 동시다발적으로 받을 수 있을지도 불투명한 상황이다.

이를 해결하기 위해, Multi Threading을 활용한 구현이 필요하다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2.2.7 결과물 목록

대분류	소분류	기능	기술문서 유무
Back-end Architecture	API Gateway	Webhook 을 등록한 Git repository 의 push 를 인식.	무
	AWS Lambda Amazon S3	API Gateway 의 신호를 받아 S3 에 해당 git repository 의 업데이트된 코드 저장	무
	Amazon SNS	Amazon S3 에 코드가 업데이트 되었음을 EC2 에 알려준다	무
	AWS EC2	Amazon SNS 로부터 신호를 받으면 S3 에 저장된 코드에 대한 분석을 자동으로 수행하여 저장한다.	무
	Git inspector	Issue 값을 계산해서 보여준다	무
		Score 값을 계산해서 보여준다	무
		Commit with low line change 리스트를 보여준다.	무
		사용자들 간의 점수를 비교하는 그래프를 보여준다.	무
Front-end Architecture	메인 화면	팀원 및 서비스를 소개하며 사용자가 쉽게 이해할 수 있는 화면을 보여준다.	무
		바로 분석하기를 누르면 분석 주소 입력창으로 이동한다.	무
		2018 캡스톤 분석 정보 보기를 누르면 분석된 자료의 리스트를 제공한다.	무
	분석하기 화면	사용자로부터 repository URL 을 받고, 사용자가 Submit 버튼을 누르면 분석을 수행하여 그 결과를 웹 페이지에 띄워준다.	무
	분석정보 보기 화면	기존에 미리 분석해놓은 파일들의 리스트를 제공한다.	무
		파일들의 리스트에서 실행을 누르면 분석결과 HTML 파일 화면을 보여준다.	무

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

2.3 기대효과 및 활용방안


2.3.1 기대효과

GitHub는 대규모의 소프트웨어 프로젝트를 진행하는 산업체부터, 대학교의 팀 프로젝트, 개인 프로젝트 사용자까지 보편적으로 사용하는 버전 관리 시스템이다. 이렇게 GitHub 저장소에 올려진 프로젝트들을 분석하여 자신이 어떤 기간에 얼마나 프로젝트를 진행했으며, 얼마나 많은 코드들이 삭제되었고 추가되었는지 등의 정보를 제공하면, 사용자는 자신이 GitHub를 얼마나 효율적으로 사용했는지에 대한 정보를 얻을 수 있다.

또한, GitHub 프로젝트에 참여한 기여자들 간의 기여도를 평가할 수 있다. 사용자가 GitHub 프로젝트에 대해 커밋을 한 횟수, 살아남은 코드 줄 수, 작성한 코드가 차지하는 비율 등을 제공 받음으로써 프로젝트에 얼마나 기여했는지를 평가할 수 있다. 또한, Git Watcher에서 제공하는 분석 결과 자료를 보면서, 원래 알지 못했던 GitHub repository의 기능들을 인지할 수 있다. 예를 들어, Git Watcher의 분석 결과를 통해 원래는 몰랐던 Issue 기능에 대해 인지할 수 있다.

2.3.2 활용방안

- 개인 프로젝트를 진행하는 경우, 자신이 기간별로 얼마만큼의 작업을 했으며, 코드의 삽입과 삭제가 얼마나 많이 이루어졌는지 등을 확인하는 데 활용되어, 더 효율적으로 GitHub를 사용할 수 있다.
- 대규모 소프트웨어 프로젝트를 진행하는 대학 수업에서, 프로젝트에 기여한 학생들의 기여도를 평가할 수 있다. 누가 작성한 코드가 얼마나 코드에 실제로 반영되었는지, 기간별로 얼마만큼의 작업이 이루어졌는지 등을 알 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

3 자기평가

최종 결과물 – Git Watcher Web Application

주요 평가 기준

1. Git Watcher 웹 서비스를 처음 이용하는 사람도 쉽게 이용할 수 있는가?
2. 충분히 큰 크기의 Git repository 에 대해서도 분석이 잘 이루어지는가?
3. 한 번에 여러 개의 Git push 가 일어나도 코드의 업데이트와 분석이 이루어지는가?
4. Git push 부터 시작해서 분석결과 파일의 생성이 예외없이 잘 이루어지는가?

먼저, 4 번의 경우 Webhook 이 등록되어 있는 Git repository 에서 push 가 일어났을 경우 EC2 에 분석결과 파일의 생성이 예외없이 잘 이루어 지는 것을 확인했다. 또한, Webhook 이 등록되어 있는 Git repository 의 경우 repository 의 크기가 커도 분석결과 파일이 잘 저장되는 것을 확인했다. 하지만, 문제는 사용자가 직접 URL 을 입력하는 경우 해당 repository 가 너무 클 때 분석 시간이 너무 오래 걸린다는 것이다. URL 에 해당하는 repository 를 직접 Cloning 한 후 분석을 하기 때문이다. 이 부분의 경우, 팀원들과 논의해 본 결과 Cloning 이 필수적인 작업이라 시간을 줄이기는 힘들 것이라고 결론지었다.


Git Watcher 에 대해 전혀 모르는 학과 동기에게 Git Watcher 웹 페이지를 알려준 뒤 어떤 설명도 없이 한번 사용해 보라고 했더니, 아무런 문제없이 잘 사용하는 것을 확인했다. 웹 서비스 자체가 간단하고 UI 를 직관적으로 구성되어 있어 사용하기 어렵지 않았다는 반응이었다.

한 번에 여러 개의 Git push 가 일어나도 코드의 업데이트와 분석이 잘 이루어지는지에 대한 부분은 일단은 멀티 쓰레딩을 통해 구현을 하여 4~5 개가 동시에 push 가 일어났을 경우에는 코드가 잘 저장되었다. 그러나 그 이상의 경우에는 올바르게 작동하지 않는 경우가 발생했다.

결론적으로, Git Watcher 는 Git push 가 동시다발적으로 일어나는 경우와, 미리 Webhook 을 등록하지 않고 직접 URL 을 입력해 즉석에서 분석을 할 때 저장소의 크기가 너무 큰 경우 시간이 오래 걸리는 것을 제외하고는 사용자의 큰 불편함 없이 잘 서비스 될 수 있을 것으로 평가된다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	웹 페이지	Github Webhook	https://developer.github.com/webhooks/			
2	웹 페이지	Git Inspector	https://github.com/ejwa/gitinspector			
3	웹 페이지	Amazon Lambda	https://aws.amazon.com/ko/lambda/features/			
4	웹 페이지	Amazon S3	https://aws.amazon.com/ko/s3/			
5	웹 페이지	Javascript d3	https://d3js.org/			

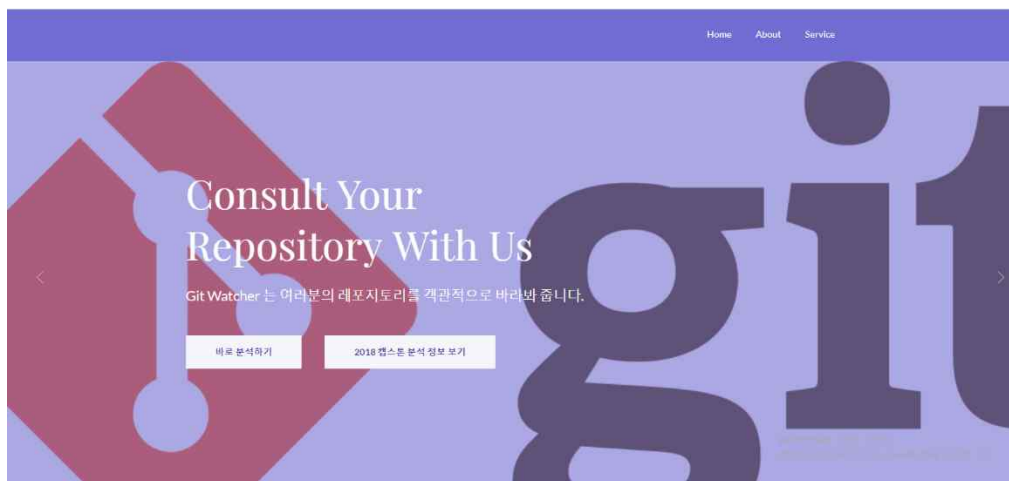
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

5 부록

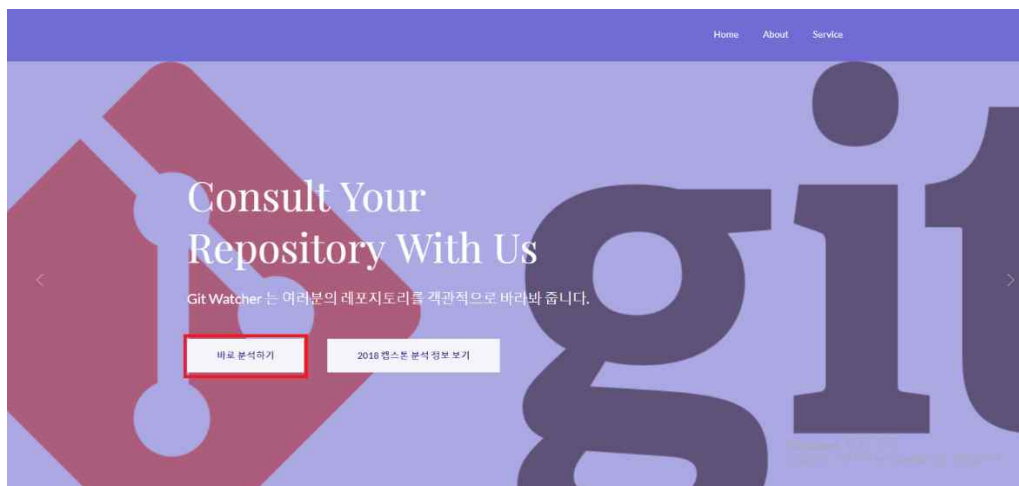
5.1 사용자 매뉴얼

1. Github repository 주소로 바로 분석하기

- A. <http://ec2-52-15-82-149.us-east-2.compute.amazonaws.com:8080/> 주소로 접속한다.

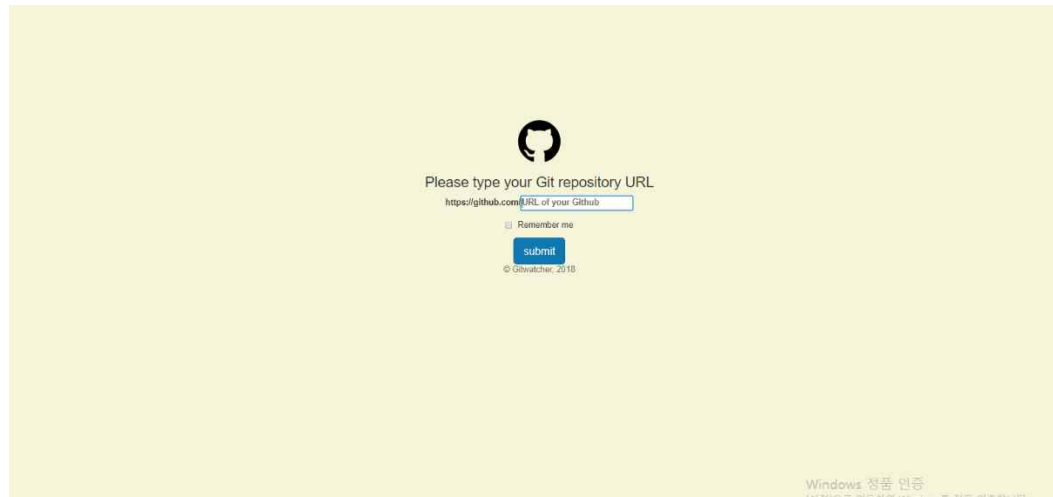


- B. ‘바로 분석하기’ 버튼을 누른다.

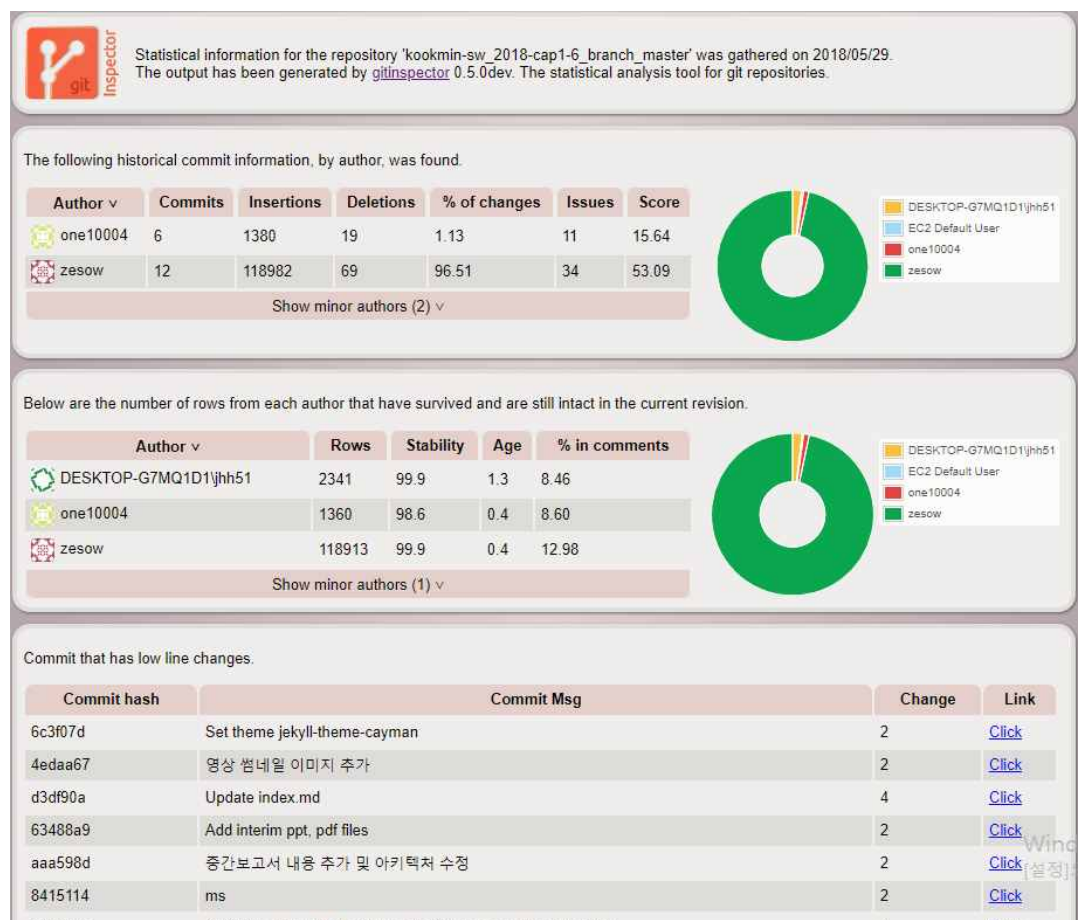



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

C. 분석을 원하는 Git repository 주소를 양식에 맞추어 적는다.



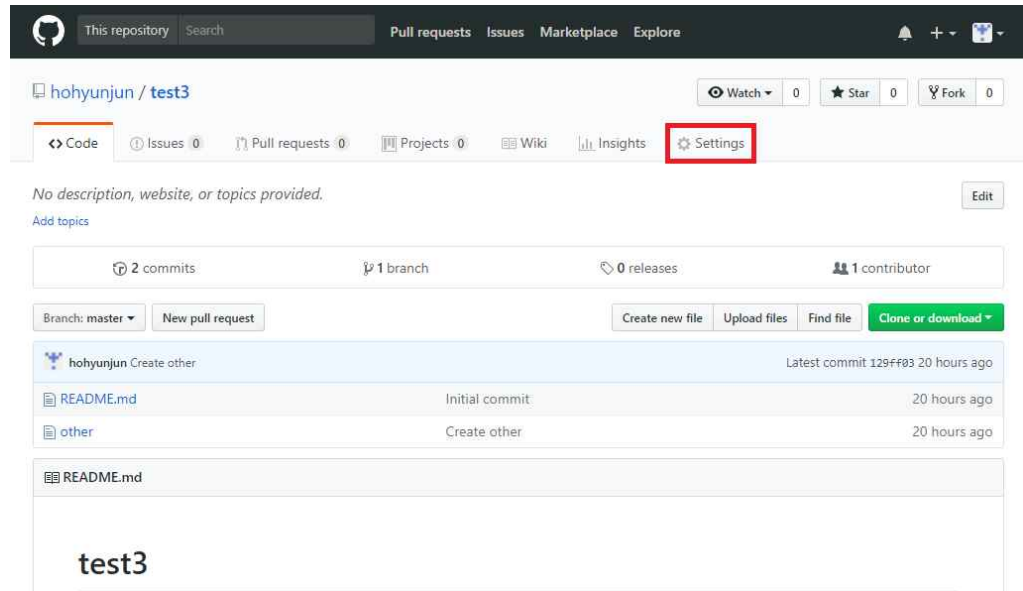
D. 분석 시간동안 대기하면 분석이 완료된다.



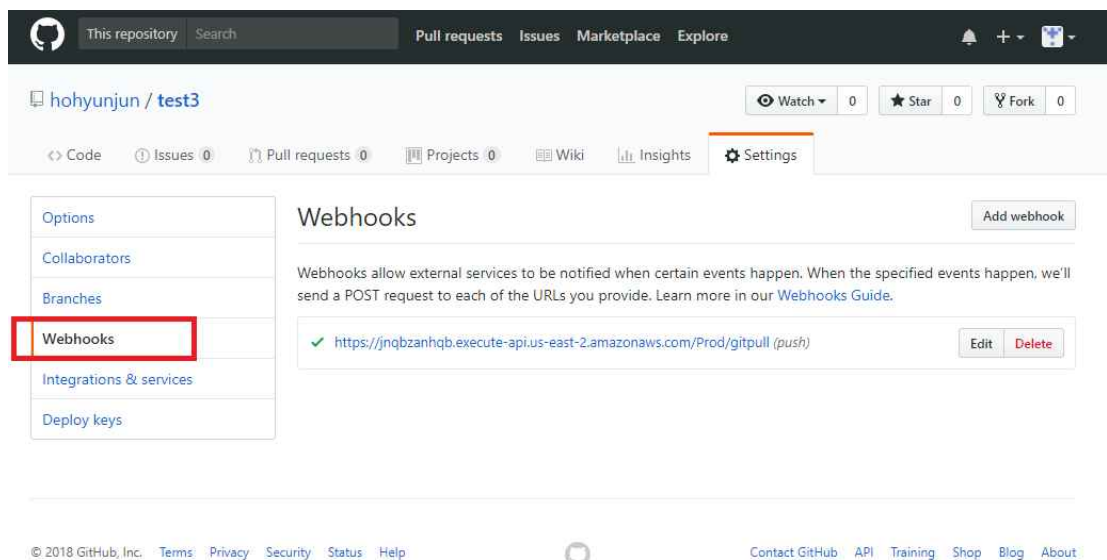
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29


2. Webhook 에 미리 등록 후 분석결과 바로 조회하기

A. 자신의 Git repository 접속 후 상단의 Settings 메뉴 클릭



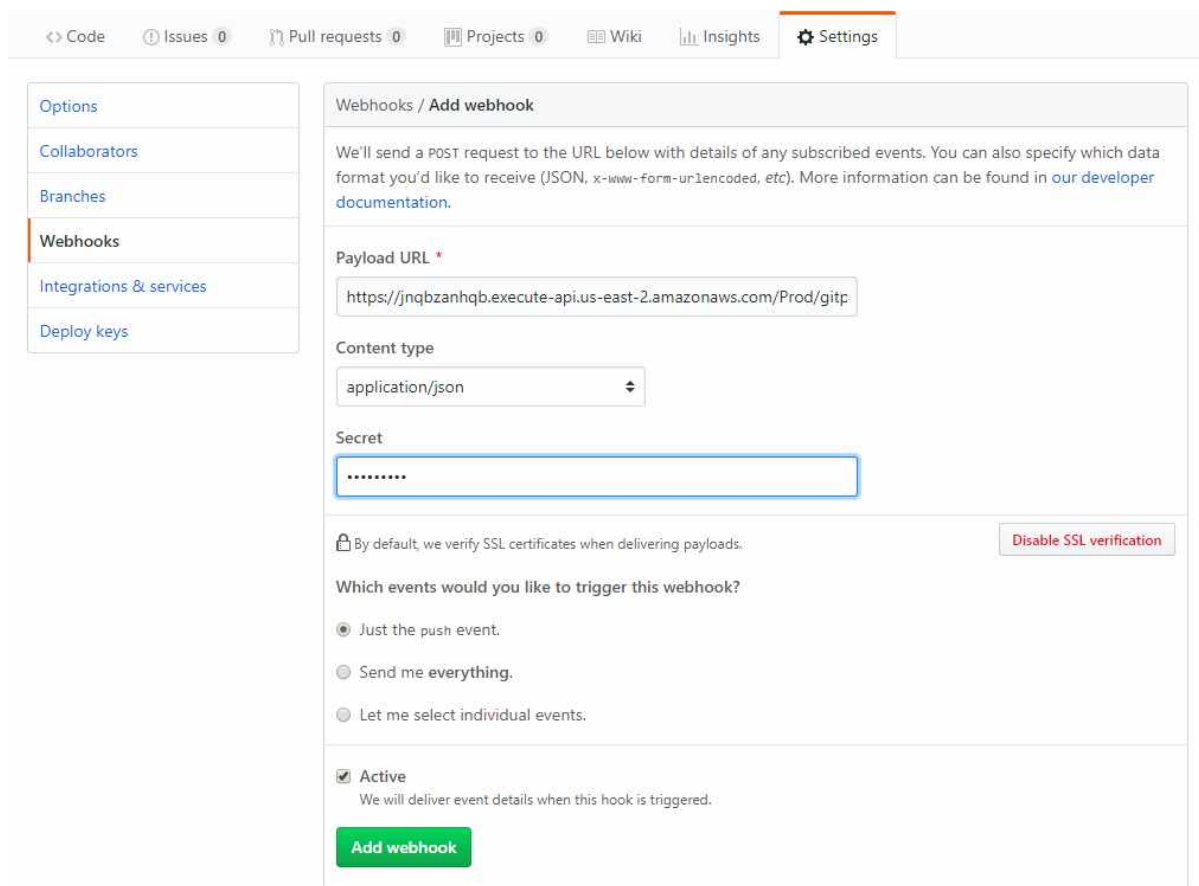
B. Settings 메뉴에서 좌측 Webhook 메뉴 클릭



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

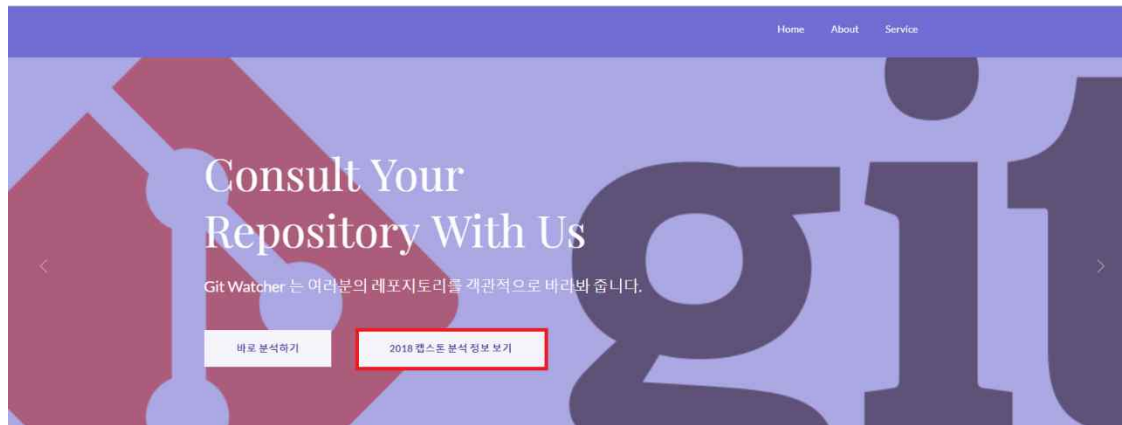
C. Add webhook 누른 후,

- i. Payload URL 에
<https://jnqbzanhqb.execute-api.us-east-2.amazonaws.com/Prod/gitpull> 입력.
- ii. Content Type 은 application/json 선택, Secret은 kookminsw 입력
- iii. 아래 선택은 Just the push event 체크, Active 체크
- iv. 설정사항 확인 후 Add webhook을 누른다.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29


- D. <http://ec2-52-15-82-149.us-east-2.compute.amazonaws.com:8080/> 주소로 접속 후, 오른쪽 버튼 클릭



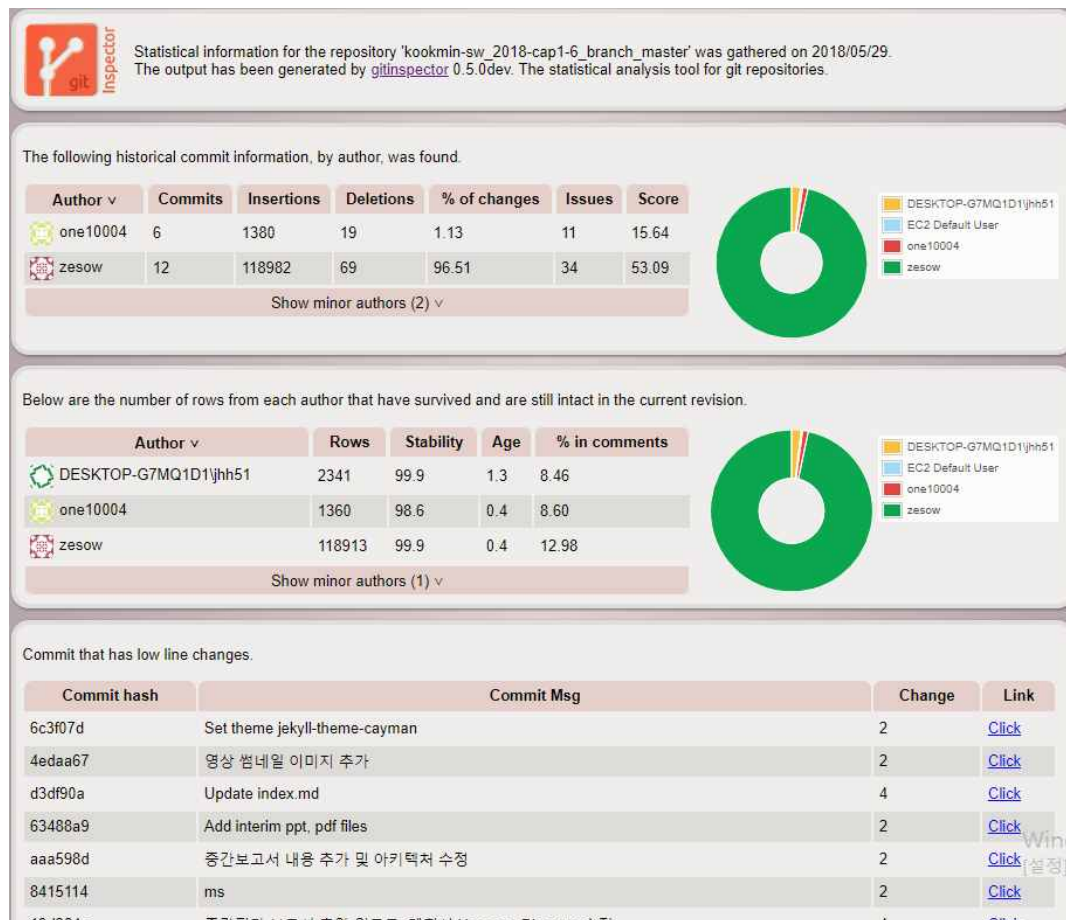
- E. 목록에서 자신의 branch 를 선택하고 실행을 누른다.


2018-cap1-12		
#	branch	실행
0	master	<button>실행</button>

2018-cap1-16		
#	branch	실행
0	web_refactoring	<button>실행</button>
1	ida_hotfix	<button>실행</button>
2	db	<button>실행</button>
3	ai_classification	<button>실행</button>
4	web_peviewer	<button>실행</button>
5	web_visualize	<button>실행</button>
6	web	<button>실행</button>
7	ann_by_static	<button>실행</button>
8	dynamic_analysis	<button>실행</button>
9	web_tensorflow_connect	<button>실행</button>

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29


F. 실행 결과를 바로 확인할 수 있다.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

5.2 테스트 케이스

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
Back-end Architecture	API Gateway	Webhook 을 등록한 Git repository 의 push 를 인식.	Webhook 을 등록한 Git repository 에서 push 를 하면, S3 에 업데이트된 코드가 저장된다.	새로운 코드 Zip 파일 S3 에 저장	성공
	AWS Lambda Amazon S3	API Gateway 의 신호를 받아 S3 에 해당 git repository 의 업데이트된 코드 저장			성공
	Amazon SNS	Amazon S3 에 코드가 업데이트 되었음을 EC2 에 알려준다	Webhook 을 등록한 Git repository 의 push 이후 EC2 에 자동으로 분석결과 HTML 파일이 생성된다.	EC2 에 자동 생성된 HTML	성공
	AWS EC2	Amazon SNS 로부터 신호를 받으면 S3 에 저장된 코드에 대한 분석을 자동으로 수행하여 저장한다.			성공
	Git inspector	Issue 값을 계산해서 보여준다	분석결과 HTML 파일에 Issue, Score, Commit with low line change 리스트, 사용자들 간의 점수 비교 그래프가 올바르게 출력된다.	Issue 출력	성공
		Score 값을 계산해서 보여준다		Score 출력	성공
		Commit with low line Change 리스트를 보여준다.		리스트 출력	성공
		사용자들 간의 점수를 비교하는 그래프를 보여준다.		그래프 출력	성공

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.2	2018-MAY-29

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
Front-end Architecture	메인 화면	팀원 및 서비스를 소개하며 사용자가 쉽게 이해할 수 있는 화면을 보여준다.	메인화면 주소 접속	메인 화면	성공
		바로 분석하기를 누르면 분석 주소 입력창으로 이동한다.	메인화면에서 바로 분석하기 클릭	분석하기 웹페이지	성공
		2018 캡스톤 분석 정보 보기를 누르면 분석된 자료의 리스트를 제공한다.	메인화면에서 2018 캡스톤 분석 정보 보기 클릭	분석정보 웹페이지	성공
	분석하기 화면	사용자로부터 repository URL 을 받고, 사용자가 Submit 버튼을 누르면 분석을 수행하여 그 결과를 웹 페이지에 띄워준다.	Repository URL 을 넣고 Submit 버튼을 누른다	분석완료 페이지	성공
	분석정보 보기 화면	기존에 미리 분석해놓은 파일들의 리스트를 제공한다.	분석정보 보기 화면 접속	분석된 파일의 리스트	성공
		파일들의 리스트에서 실행을 누르면 분석결과 HTML 파일 화면을 보여준다.	분석정보 보기에서 실행 클릭	분석완료 페이지	성공