


캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	Git watcher
팀 명	GET
문서 제목	중간보고서

Version	1.3
Date	2018-04-12

팀원	전호현(조장)
	유문상
	최원대
	이근하
지도교수	이경용 교수

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12


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	중간보고서-Git Watcher.doc
원안작성자	전호현
수정작업자	유문상

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2018-04-08	전호현	1.0	최초 작성	
2018-04-09	유문상	1.1	내용 수정	Git push 시 Git inspector분석 내용추가
2018-04-11	전호현	1.2	내용 추가	BootStrap 사용 웹 페이지 꾸미기 추가 향후 추진계획 추가
2018-04-12	전호현	1.3	내용 수정	프로젝트 목표 수정

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

목 차

1	프로젝트 목표	4
2	수행 내용 및 중간결과	5
2.1	계획서 상의 연구내용	5
2.1.1	GitHub 프로젝트의 변동사항을 알려줄 API 탐색	5
2.1.2	AWS Backend Architecture 구축	5
2.1.3	Git Inspector	6
2.1.4	Amazon S3에 코드 저장, 유지	7
2.1.5	웹 페이지 구축	7
2.1.6	Git inspector의 기능 추가	7
2.2	수행내용	8
2.2.1	GitHub repository의 push 인식하여 Amazon S3에 코드 저장하기	8
2.2.2	Git inspector Customizing	9
2.2.3	Lambda Git inspector packaging	10
2.2.4	AWS EC2에서 Git inspector 분석 수행	10
2.2.5	Git push 시 Git inspector 분석파일 저장	11
3	수정된 연구내용 및 추진 방향	13
3.1	수정사항	13
3.1.1	AWS Lambda를 EC2로 대체	13
3.1.2	EC2 서버에서 Git push 시마다 분석, 결과값 저장	14
4	향후 추진계획	15
4.1	향후 계획의 세부 내용	15
4.1.1	결과값 점수화	15
4.1.2	분석 기능 추가	15
5	고충 및 건의사항	16


1 프로젝트 목표

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 있는지의 기준을 만들어 그 내용을 사용자에게 수치화하여 보여준다. 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, 데이터베이스에 코드 정보를 미리 저장하여 더 빠른 분석결과를 제공한다. 더 나아가 Git Inspector에서는 제공하지 않는 기능들도 추가하여 제공할 것이다.

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.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

2 수행 내용 및 중간결과

2.1 계획서 상의 연구내용

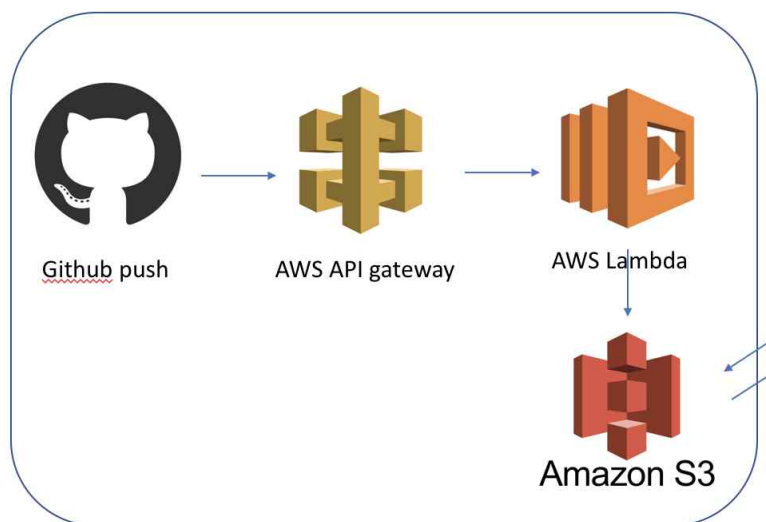
2.1.1 GitHub 프로젝트의 변동사항을 알려줄 API 탐색

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 한다.


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 경우부터 시작하여, 이것을 API gateway와 연결하고, AWS Lambda와 연동시켜 Amazon S3에 업데이트된 코드를 저장시킬 수 있어야 한다.

두 번째 아키텍처는 사용자의 요청이 들어올 때마다 분석을 수행하는 아키텍처이다. AWS Lambda에 Git inspector를 올리고, 웹페이지로부터 사용자 요청을 받고, 그 결과를 보내줄 수 있어야 한다. 또한, 최신의 레포지토리 상태를 유지하고 있는 첫번째 아키텍처의 Amazon S3와도 연결되어야 한다. 저장되어 있지 않은 repository를 사용자가 요청할 경우에 대비하여, Git inspector가 바로 Github repository에서 Cloning 하여 분석을 수행할 수 있도록 해야한다.



항상 최신의 레포지토리 상태 유지

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

2.1.3 Git Inspector

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

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

2.1.4 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.


2.1.5 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. 분석 결과를 원하는 파일로 만들어 주는 format 이나 특정 확장자 파일만 검색하도록 도와주는 옵션에 대한 User Interface 또한 존재해야만 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.

2.1.6 Git inspector의 기능 추가

기존 Git inspector의 기능에 더하여, 사용자의 GitHub 활용성에 도움을 줄 수 있을 만한 분석 요소를 찾아서 분석기능을 추가하도록 한다.

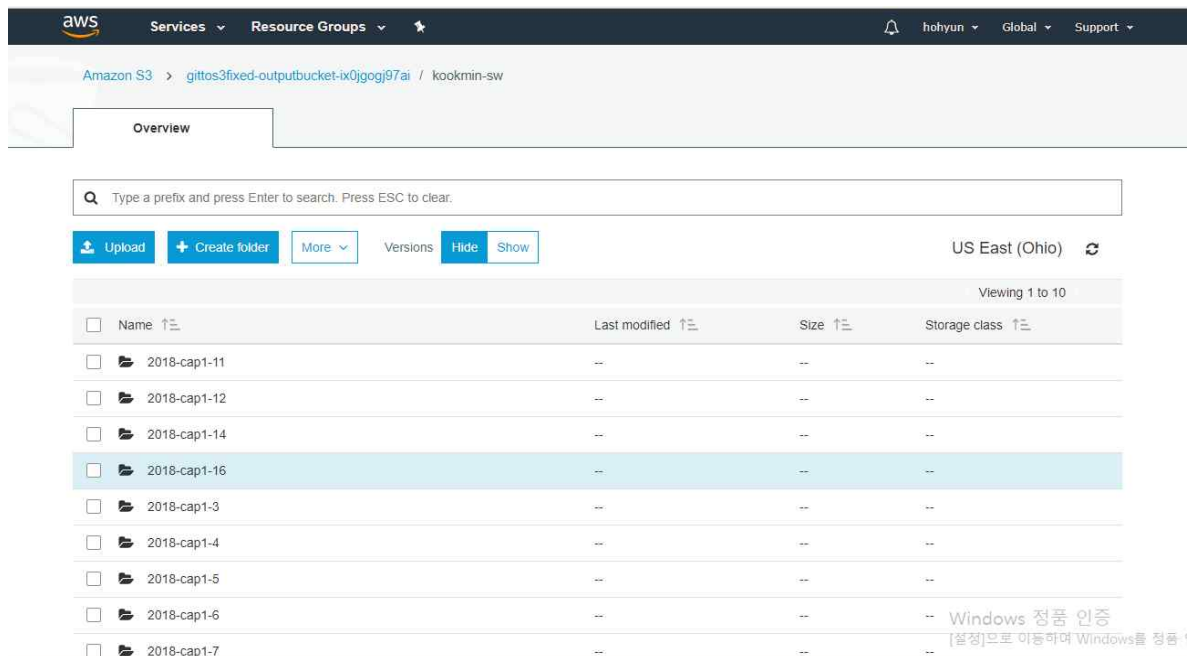
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12


2.2 수행내용

2.2.1 GitHub repository의 push 인식하여 Amazon S3에 코드 저장하기

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.



 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

2.2.2 Git inspector Customizing

Git inspector를 프로젝트에서 사용할 수 있도록 커스터마이징 하는 작업이다. 원래 Git inspector는 Github repository 주소를 가지고 cloning 후에 분석을 진행한다. 우리 팀에서 진행하는 프로젝트의 경우, Amazon S3에 저장된 코드(zip파일)을 가져와서 분석을 진행해야 한다. 그러므로, Git inspector에 필요한 커스터마이징은 크게 두가지이다.

1. Zip 파일 압축 해제 후 분석
2. Github repository의 Cloning 없이 로컬 폴더 분석

이러한 커스터마이징 작업을 위해 Git inspector의 코드를 분석하고 수정하였다. 실행 시 로컬 폴더를 분석할지, 로컬에 있는 zip 파일을 분석할지, Github repository의 주소를 입력하여 분석을 진행할지 선택할 수 있다.

```
PS C:\Users\Wone10\Desktop\gitinspector-master - 복사본> python gitinspector.py
1. Local Folder 2. Local Zip File 3. Git Address : 1
Path of Local Folder : C:\Users\Wone10\Desktop\캡스톤테스트\creativepoo
```

```

Statistical information for the repository 'creativepoo' was gathered on 2018/03/21.
The following historical commit information, by author, was found:

+-----+-----+-----+-----+-----+
| [1mAuthor          | Commits | Insertions | Deletions | % of changes|[0:0m
+-----+-----+-----+-----+-----+
| DESKTOP-G7M01D1W\jhh5 | 1       | 41         | 0         | 1.23
| LeeGeunHAHAHA      | 16      | 766        | 68        | 24.97
| SungJun Lee        | 19      | 878        | 44        | 27.60
| occidere            | 4       | 296        | 0         | 8.86
| one10004            | 4       | 482        | 0         | 14.43
| shinjong93         | 1       | 104        | 0         | 3.11
| zesow               | 3       | 361        | 0         | 10.81
| 신종혁              | 3       | 300        | 0         | 8.98
+-----+-----+-----+-----+-----+

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

```


```
PS C:\Users\Wone10\Desktop\gitinspector-master - 복사본> python gitinspector.py
1. Local Folder 2. Local Zip File 3. Git Address : 2
Path of Local Zip File : C:\Users\Wone10\Desktop\캡스톤테스트
Name of Zip File : creativepoo
```

```

Statistical information for the repository 'creativepooUnzip' was gathered on 2018/03/21.
The following historical commit information, by author, was found:

+-----+-----+-----+-----+-----+
| [1mAuthor          | Commits | Insertions | Deletions | % of changes|[0:0m
+-----+-----+-----+-----+-----+
| DESKTOP-G7M01D1W\jhh5 | 1       | 41         | 0         | 1.23
| LeeGeunHAHAHA      | 16      | 766        | 68        | 24.97
| SungJun Lee        | 19      | 878        | 44        | 27.60
| occidere            | 4       | 296        | 0         | 8.86
| one10004            | 4       | 482        | 0         | 14.43
| shinjong93         | 1       | 104        | 0         | 3.11
| zesow               | 3       | 361        | 0         | 10.81
| 신종혁              | 3       | 300        | 0         | 8.98
+-----+-----+-----+-----+-----+

```

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

위 쪽에 위치하는 두 개의 사진은 로컬 폴더의 분석을 하는 것을, 아래의 두 개 사진은 zip파일을 압축해제하여 분석하는 것을 보여준다. 두 과정 모두 Git repository의 Cloning 없이 분석작업만 이루어진다. Zip 파일의 경우에는 파일의 압축을 풀어, 해당 Local 폴더로 이동하여 git inspector 분석을 수행한다.

2.2.3 Lambda Git inspector packaging

Python Code로 짜여진 Git inspector를 AWS Lambda에 Packaging하여 사용자가 요청을 할 때만 Lambda가 작동하여 분석작업을 하도록 하는 작업이다. 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 진행하기로 하였다.

2.2.4 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 분석 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.



2.2.5 Git push 시 Git inspector 분석파일 저장

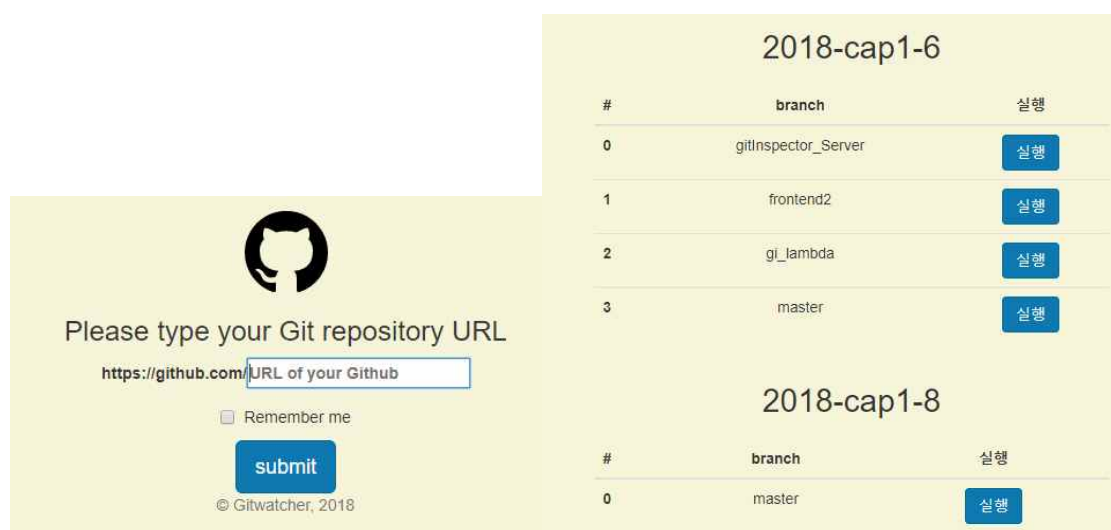
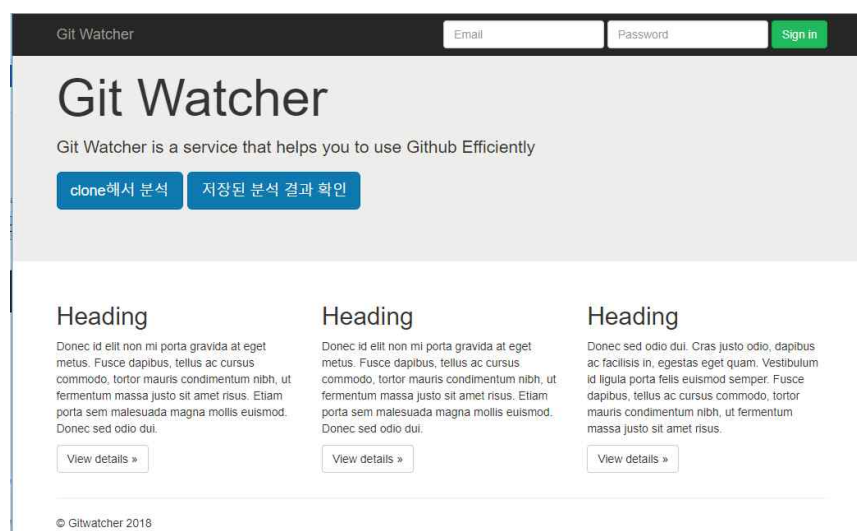
Git push -> S3에 코드 zip 파일로 저장 -> EC2에 있는 Git inspector가 S3에 새로운 zip 파일이 올라왔음을 인식 -> 새로운 zip 파일 가져와서 분석 -> EC2 내부에 결과파일 저장의 플로우를 구현했다. AWS S3에 repository가 업데이트되면 aws sns 서비스가 ec2에 알람을 보내준다. Ec2에는 알람을 받아서 s3에서 repository를 다운로드, 분석하여 html로 뽑아주는 서버가 구동 중이다. 분석 중에 다른 알람이 들어와서 알람을 놓치는 상황을 방지하기 위해 멀티 쓰레딩으로 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 HTML파일들을 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	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

2.2.6 Twitter Bootstrap을 사용하여 웹 페이지 제작

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.



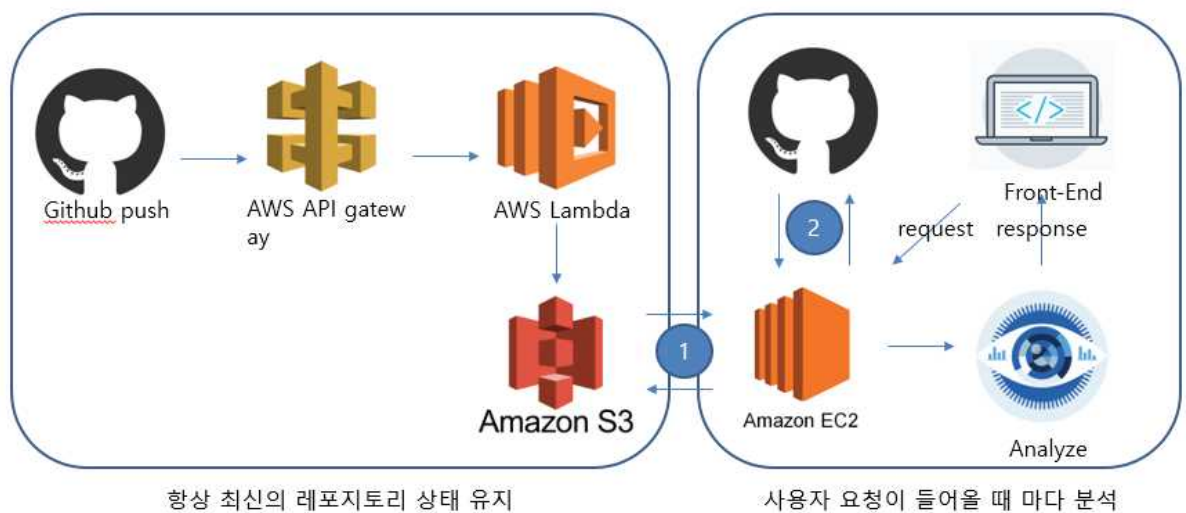
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12


3 수정된 연구내용 및 추진 방향

3.1 수정사항

3.1.1 AWS Lambda를 EC2로 대체

기존 연구사항에서는, Git inspector를 Lambda에 Packaging 하여 올려서 분석 요청이 들어올 때만 Lambda가 동작하여 분석결과를 보여주는 형태였다. 그런데 Git inspector가 Git이 깔려 있어야만 실행이 되고, Lambda는 기본 리눅스 환경에서 제공하지 않는 라이브러리의 경우 사용이 힘들다는 사실을 알게 되었다. 이에 따라, 기존 Lambda 부분을 AWS EC2로 대체하기로 하였다. 이제 EC2 서버에서 Git inspector가 사용자의 요청을 받아 분석을 수행한다.




 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

3.1.2 EC2 서버에서 Git push 시마다 분석, 결과값 저장

기존의 설계 의도에 따라, EC2 서버에 사용자 요청이 주어질 경우 수집된 코드의 zip 파일을 가져와 분석하여 결과값을 제공하는 방식을 수행해 보았다. 분석 결과값이 제대로 나오긴 하지만, 사용자가 요청을 보내고 응답 받는 시간이 10초 이상으로, 매우 오래 걸렸다. 코드의 zip파일 크기가 커지면 커질수록 시간이 더 오래 걸릴 것으로 예상되었다. 그래서, Git push가 일어났을 때 코드만 저장하는 것이 아니라, 이 저장된 값을 분석한 결과를 EC2에 저장하는 방식으로 추진 방향을 수정하였다.

결과적으로, Webhook을 등록한 Git repository의 사용자가 Git push를 하면, S3에 코드의 zip 파일이 저장되고, 이를 Git inspector가 올려진 EC2가 인식하여 분석작업을 한 뒤 HTML 형태로 결과값을 저장해 놓는다. 사용자가 요청을 보냈을 경우, 이미 분석되어 있는 HTML 파일만 보내주면 되므로, 응답시간이 크게 단축된다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

4 향후 추진계획

4.1 향후 계획의 세부 내용

4.1.1 결과값 점수화

"Github repository 분석을 통해 사용자의 효율적인 Github 사용을 돕는 서비스"를 만드는 것이 우리 팀의 최종적인 목표이다. 프로그램을 돌려서 나온 분석결과를 바탕으로, 사용자가 얼마나 Github를 효율적으로 사용하고 있는지의 기준을 만들어 그 내용을 수치화하여 사용자에게 보여줄 수 있어야 한다. 어떻게 하는 것이 효율적인 Github 사용인지에 대해 생각해 본다.

4.1.2 분석 기능 추가

기존에 Git inspector에서 제공하는 커밋 횟수, 삽입 라인, 삭제 라인, 변화 수, 안정성 등의 항목에 더해, 사용자의 효율적인 Github 사용을 위해 제공할 수 있는 기능에 대해 생각해 본다. 그리고 Git inspector의 코드를 수정하여 기능을 추가한다.

4.1.3 분석 결과 Graph화

현재 Git inspector의 분석 결과 화면은 기여자 별로 몇 퍼센트의 변화에 기여했는지, 얼마만큼의 code row가 현재까지 살아 남았는지에 대한 정보만 원형 그래프로 제공한다. 이외의 정보들, 기간별로 얼마나 많은 code를 삽입했는지에 대한 정보 분석 결과도 Graph를 통해 사용자가 더 보기 쉽도록 만든다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명	Git Watcher	
	팀 명	GET	
	Confidential Restricted	Version 1.3	2018-APR-12

5 고충 및 건의사항