

# Github 기초 활용하기

# 형상관리를 위한 소프트웨어



소스 코드의 변경 이력을 관리

- **GITHUB 란?**

GIT으로 만든 **원격저장소 서비스**

- **GIT이란?**



리눅스 창시자 리누스 토발즈가 만든 분산 버전 관리 시스템 (DVCS)

- **분산 버전 관리 시스템**

Distributed Version Control System (DVCS)

여러 사람이 협동 작업하는 환경에서 문서변경사항을 관리하는 시스템

- **특징**

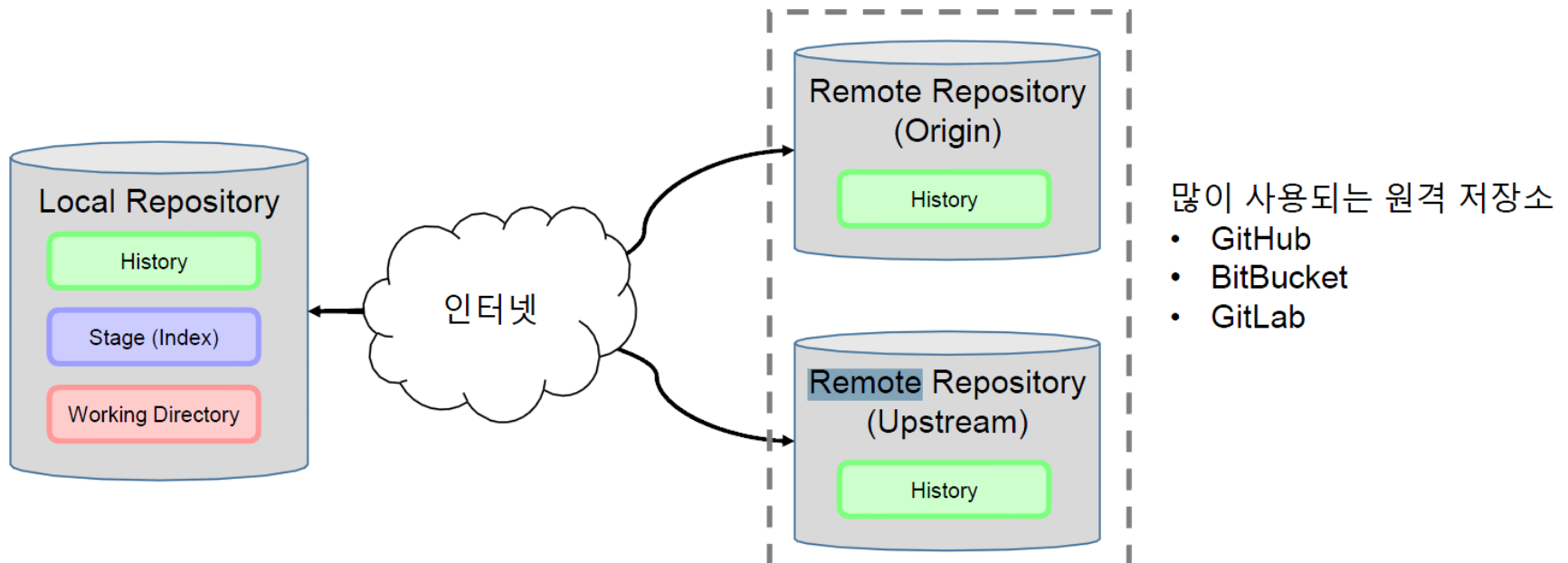
- 변경사항을 적절히 저장했다가 필요한 시점으로 돌릴 수 있다.
- 서로 다른 변경사항들을 쉽게 합칠 수 있는 기능을 제공한다.
- 저장소가 로컬(내 컴퓨터)에 있어 네트워크가 끊어져도 작업 가능하다.
- 다른 버전관리 시스템보다 빠르다.
- 원격저장소를 연결해 협동작업이 가능하다.
- 변경사항을 관리할 대상을 스테이지(Stage)를 이용해 관리 가능하다.

# Git의 3가지 영역

- **작업 폴더(Working Directory)**
  - 사용자가 변경하는 실제 파일이 들어가는 폴더
- **스테이지(Stage, Index)**
  - 변경사항을 관리할 파일들의 리스트
- **변경이력(History)**
  - 커밋(Commit)이라 불리는 변경사항 묶음과 커밋 들의 연결관계

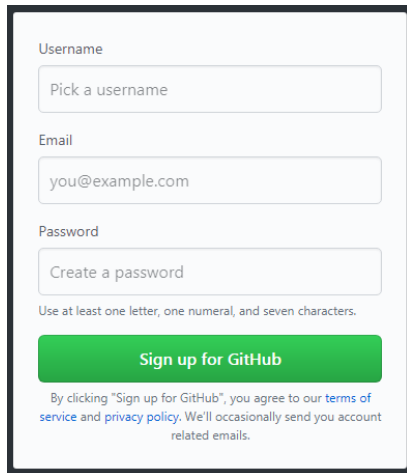
# 로컬저장소와 원격저장소

- 협업을 위해서는 원격저장소가 필수적
- 로컬저장소와 원격저장소 간에 이력을 주고받을 수 있음
- 원격저장소가 여러 개 일 수 있음



# Github 접속과 가입

- <https://github.com> 에 접속한다.
- 계정이 있는 사람은 [Sign in]으로 로그인
- 계정이 없는 사람은 [Sign up]으로 가입
- 가입 시 username과 email이 기존계정과 겹치면 안된다.
- 비용 플랜은 Free를 선택하면 공개 저장소만 만들 수 있다.
- 가입 완료 후 꼭! 이메일 인증을 받아야 한다.



Username

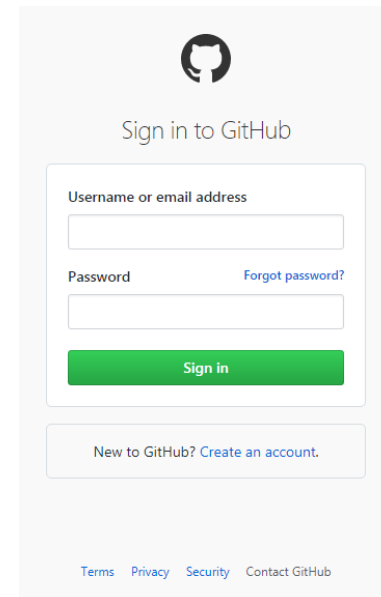
Email


Password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.





Sign in to GitHub

Username or email address

Password [Forgot password?](#)

[Sign in](#)

New to GitHub? [Create an account.](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

# 새 github 프로젝트 만들기

1. 로그인 한 첫 화면에서 [Start Project]를 누른다.
2. Repository Name에 'git-workshop' 이라 입력한다.
3. Description에 'git 사용 실습' 이라 입력한다.
4. Initialize this repository with a README 옵션에 체크한다.
5. [Create repository] 를 누른다.



Owner: [Avatar] / Repository name: **2** git-workshop ✓

Great repository names are short and memorable. Need inspiration?

Description (optional): **3** git 사용 실습

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**4** ☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip th

Add .gitignore: None ▾ | Add a license: None ▾ ⓘ

**5** [Create repository]

# Git과 Github

- Git

Git은 분산 버전 관리 시스템이다.

Git으로 프로젝트를 개발하는 사람은 모두 현재 상태의 파일뿐만 아니라 그 프로젝트의 전체 이력을 가지고 있게 된다는 뜻이다.

버전관리 시스템 : 파일의 변경 내역을 계속 추적하도록 개발된 소프트웨어

- GitHub

GitHub는 Git 저장소(repository)를 업로드 할 수 있는 웹사이트를 말한다.

GitHub는 다른 사람들과의 협업을 매우 용이하게 해준다.

리포지토리를 공유할 수 있는 중앙저장소

- 웹 기반 인터페이스
  - Forking
  - pull requests
  - Issues
  - Wikis
- Github는 위와 같은 기능을 제공하여 팀원들과 보다 효율적으로 변경안을 구체화하고 토론하며 검토할 수 있게 해준다.



# Git의 장점

- **변경 취소 가능** : 실수 했을 때 구 버전의 작업 파일로 돌아갈 수 있음.
- **모든 변경에 대한 완벽한 이력 History** : 짧게는 하루, 길게는 1년 전에 프로젝트가 어떤 형태였는지도 알 수 있다.
- **변경한 이유를 기록** : 협업을 하다보면(때로는 내가 작성한 코드라할지라도) 왜 변경했는지 모르겠을 때가 많다. 이때 Git의 커밋 메시지(commit message)를 이용하면 변경한 이유를 쉽게 기록할 수 있으며, 추후에 참조할 수 있다.
- **변경에 대한 확신** : 이전 버전으로의 복귀가 쉽기 때문에 자신을 원하는 대로 다 변경가능하다. 잘 안되면 언제든지 이전 버전으로 복귀하면 된다.
- **여러 갈래의 히스토리(History)** : 콘텐츠의 변경 내용을 테스트해보거나 기능을 독립적으로 실험해보기 위해 별도의 브랜치(branch)를 생성할 수 있다. 완료되면 변경 내용을 마스터 브랜치(Master branch)로 병합할 수 있고, 잘 작동하지 않을 경우 삭제 가능하다.

# Git의 장점

- **충돌 해결 능력** : Git을 이용하면 여러사람이 동시에 같은 파일을 작업할 수 있다. Git은 대개 자동으로 변경사항을 병합할 수 있다. 그렇지 못할 경우에 충돌이 무엇인지 알려주고 이를 해결하기 쉽게 해준다.
- **독립된 히스토리(History)** : 여러 사람들이 다른 브랜치(branch)에서 작업이 가능하다. 기능을 독립적으로 개발하고, 완료되었을 때 그 기능을 병합할 수 있다.

# 왜 GitHub를 사용하는가?

- GitHub는 단순히 Git 저장소를 제공하는 것 이상의 가치를 제공한다.
- **기록요구** : Issue(이슈)를 사용해 버그를 기록하거나 개발하고 싶은 새로운 기능을 구체화할 수 있다.
- **독립된 히스토리(History)에 대한 협력** : branch와 pull requests를 이용해 다른 브랜치 또는 기능에 협력할 수 있다.
- **진행 중인 작업 검토** : pull requests 목록을 통해 현재 무슨 작업이 진행되고 있는지 모두 볼 수 있다. 그리고 특정 pull request를 클릭하여 최근의 변경 내용과 변경에 관한 모든 논의 내용을 볼 수 있다.
- **팀의 작업 진척 상황 확인** : 펄스(pulse)를 훑어보거나 commit history를 살펴보면 팀의 작업 진척 상황을 알 수 있다.

# Git & Github

- **Commit**

: 하나 또는 다수의 파일에 변경 내용을 저장할 때마다 새로운 commit 생성한다.

ex) "이 변경 내용을 commit하고 이를 GitHub로 push 합시다."

- **commit message**

: commit을 생서할 때 마다 왜 변경을 했는지에 대한 이유를 설명하는 메시지를 제공해야한다. 이 commit message는 변경을 한 이유를 추후에 이해할 때 매우 유용하다.

ex) "새로운 SEC 가이드라인에 대한 수잔의 의견을 commit 메시지에 꼭 넣으세요."

- **Branch**

: 테스트를 해보거나 새로운 기능을 개발하기 위해 사용할 수 있는 따로 떨어진 독립적인 commit들을 말한다.

ex) "새로운 검색기능을 구현하기 위해 branch를 생성합시다."

- **master branch**

: 새로운 Git 프로젝트를 만들 때마다 'master'라고 불리는 기본 branch가 생성된다. 배포할 준비가 되면 작업이 최종적으로 마무리되는 branch이다.

ex) "master로 바로 commit 하면 안 된다는 것을 기억하세요"

# Git & Github

- **feature or topic branch**

: 새로운기능을 개발할 때마다 작업할 브랜치를 만드는데, 이를 feature branch라고 한다.

ex) "feature branch가 너무 많습니다. 이들 중 하나나 두개를 완료해서 출시하는데 주력합니다."

- **release branch**

: 직접 QA(품질보증) 작업을 하거나 고객의 요구로 구 버전의 소프트웨어를 지원해야 한다면 모든 수정이나 업데이트를 위한 장소로 release branch가 필요하다. feature branch와 release branch는 기능적 차이는 없지만, 팀월들과 프로젝트에 대해 이야기할 때 확연히 구별할 수 있어 유용하다.

ex) "release branch 전체에 대한 보안 버그를 수정해야 합니다."

- **Merge**

: 병합(merge)은 한 branch에서 완성된 작업을 가져와 다른 branch에 포함하는 방법이다. 흔히 feature branch를 master branch로 merge한다.

ex) "'내 계정'기능이 훌륭합니다. 배포할 수 있게 master로 merge 해줄 수 있습니까?"

# Git & Github

- **Tag**

: 특정 이력이 있는 commit에 대한 참조. 어떤 버전의 코드가 언제 배포(release)되었는지 정확히 알 수 있도록 제품 배포 기록에 가장 자주 사용된다.

ex) "이번 배포판에 tag를 달고 출시합시다."

- **Check out**

: 프로젝트 history의 다른 버전으로 이동해 해당 시점의 파일을 보기 위해 'checkout' 한다. 가장 일반적으로 branch에서 완료된 작업을 모두 보기 위해 branch를 checkout하지만, commit도 checkout할 수 있다.

ex) "최종 릴리즈 태그(release tag)를 checkout 해주시겠어요? 제품에 버그가 있어서 검증하고 수정해야 합니다."

- **Pull request**

: 원래 pull request는 branch에서 완료된 작업을 다른 사람이 리뷰하고 master로 merge요청을 하기 위해 사용되었다. 요즘은 개발 가능한 기능에 대한 논의를 시작하는 초기 단계에서 자주 사용한다.

ex) "다른 팀원들이 어떻게 생각하는지 알 수 있게 새로운 투표 기능에 대한 pull request를 만드십시오."

# Git & Github

- **Issue**

: GitHub는 기능에 대해 논의하거나 버그를 추적하거나 혹은 두 가지 경우 모두 사용될 수 있는 Issue라는 기능을 갖고 있다.

ex) "당신이 옳아요, 아이폰에서는 로그인도 되지 않네요. 버그를 검증하기 위한 단계를 기록하면서 GitHub에 issue를 생성해 주시겠습니까?"

- **Wiki**

: Ward Cunningham이 최초로 개발하였다. wiki는 링크들 간을 연결해 간단하게 웹페이지를 만드는 방법이다. GitHub 프로젝트는 문서 작성에 자주 Wiki를 사용한다.

ex) "복수의 서버에서 작동하게끔 프로젝트 환경 설정 방법을 설명하는 페이지를 추가해 주시겠습니까?"

- **Clone**

: 종종 로컬로 작업하기 위해 프로젝트 복사본을 GitHub에서 다운로드 한다. repository를 사용자의 컴퓨터로 복사하는 과정을 복제(cloning)라고 한다.

ex) "repository를 clone하고, 버그를 수정한 다음 오늘 밤 수정본을 다시 GitHub로 push 해주시겠습니까?"

# Git & Github

- **Fork**

: 때로는 프로젝트를 직접 변경하는 데 필요한 권한을 보유하지 못할 때가 있다. 잘 알지 못하는 사람이 작성한 오픈소스 프로젝트이거나, 회사에서 작업을 같이 많이 하지 않는 다른 그룹이 작성한 프로젝트일 수도 있다. 그러한 프로젝트를 변경하고 싶다면 먼저 GitHub의 사용자 계정에 프로젝트 복사본을 만들어야한다. 그 과정을 repository를 fork한다고 한다. 그런다음 복제(clone)하고, 변경하고, pull request를 이용해 원본 프로젝트에 변경 내용을 반영할 수 있다.

ex) "홈페이지 마케팅 원고를 당신이 어떻게 재작성 했는지 보고 싶습니다. repository를 fork하고 수정안과 함께 pull request를 제출해주세요."



# 프로젝트 보기


- 이 장에서는 프로젝트의 상태를 확인할 수 있는 방법을 살펴볼 것이다. 잘 알려진 부트스트랩(Bootstrap)의 오픈 소스 프로젝트를 예제로 사용하겠다.

# 프로젝트 페이지 소개




- 부트스트랩은 개발자들이 웹앱을 빠르게 개발할 수 있게 해주는 프로젝트이다. GitHub의 프로젝트 페이지로 이동한다. 홈페이지에 아주 많은 정보가 있지만, 먼저 가장 중요한 요소들을 살펴보자.

# 프로젝트 페이지 소개




- 위 사진 왼쪽 위를 보면 twbs/bootstrap라고 적혀있는데, 이는 twbs라는 사용자가 bootstrap이라는 프로젝트를 소유하고있다는 것을 뜻한다.

 This repository Search


Pull requestsIssuesGist


  


twbs / bootstrap


 6,046  93,452  40,027

<> Code


 Issues 382


 Pull requests 136


 Pulse

 Graphs

The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web.  
<http://getbootstrap.com>

 11,700 commits

 26 branches

 36 releases

670 contributors

Branch: master

New pull request


New file


Upload files

Find file


HTTPS

<https://github.com/twbs>





Download ZIP

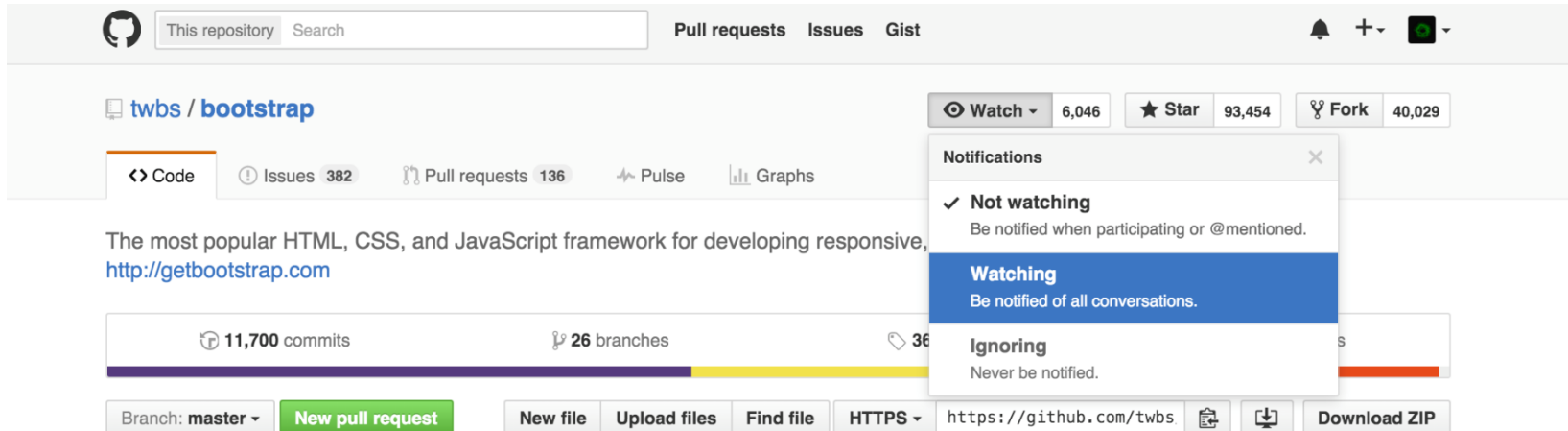
 cvrebert committed with cvrebert Port ada8f21 to v3

Latest commit 0f1d09b 9 hours ago

dist	Run grunt.	11 days ago
docs	Port ada8f21 to v3	9 hours ago
fonts	Add glyphicons fonts in woff2	a year ago
grunt	Use iOS 9.2 in Sauce tests	16 days ago
js	Update jQuery to 1.12.1	16 days ago
less	variables.less: Fix typo in comment for @caret-width-base	2 months ago
nuget	Update copyright years to 2016	2 months ago

# 프로젝트 페이지 소개

- GitHub를 그동안 오래 보아왔지만, 눈, 별, 나무줄기(?)형태의 아이콘이 정확히 어떤의미인지는 모르고있었다.
- '눈'은 이 프로젝트에 새로운 변화가 생길 때 마다 알림을 받겠다는 뜻이다. '구독' 정도의 의미로 이해하면 좋겠다.
- '별'은 즐겨찾기 또는 별점을 부여했다는 것이고,
- '나무줄기'는 아이콘에도 나와있지만, 이 프로젝트를 fork한 수를 의미한다.



# README.md파일 보기

- 프로젝트의 루트에 README.md라는 파일이 있으면 그 파일의 콘텐츠가 프로젝트 홈페이지의 폴더와 파일 목록 바로 아래에 표시된다.
- 이 파일은 프로젝트에 대한 소개와 협력자들에게 유용한 추가 정보(소프트웨어를 어떻게 설치하는지, 자동화된 테스트를 어떻게 실행하는지, 코드를 어떻게 사용하는지, 프로젝트에 어떻게 기여할 수 있는지 등)를 제공한다.
- 요즘들어 README.md 파일은 badge도 자주 포함하는데, badge는 자동화된 test suite같이 프로젝트의 현 상태를 알려주기 위해 사용되는 이미지다.

# README.md파일 보기

- [그림]에서는 부트스트랩이 동작하는 두 개의 다른 프로젝트의 버전을 표시하고 있다. 또한 자동화된 테스트를 통과했고, 종속관계(dependencies)가 업데이트 되었다는 것과, 브라우저와 운영체제의 버전을 보여주고 있다.

composer.json

version bump

a year ago

package.js

bump version

4 months ago

package.json

Update dependencies.

3 months ago

README.md

# Bootstrap

slack251/6740

bowerv3.3.6

npmv3.3.6

buildpassing

devDependenciesout-of-date

nugetinaccessible

Firefox

4410.0.1

Chrome

4810.0.1

IE

118.1

iPhone

9.210.1

Edge

2010

Safari

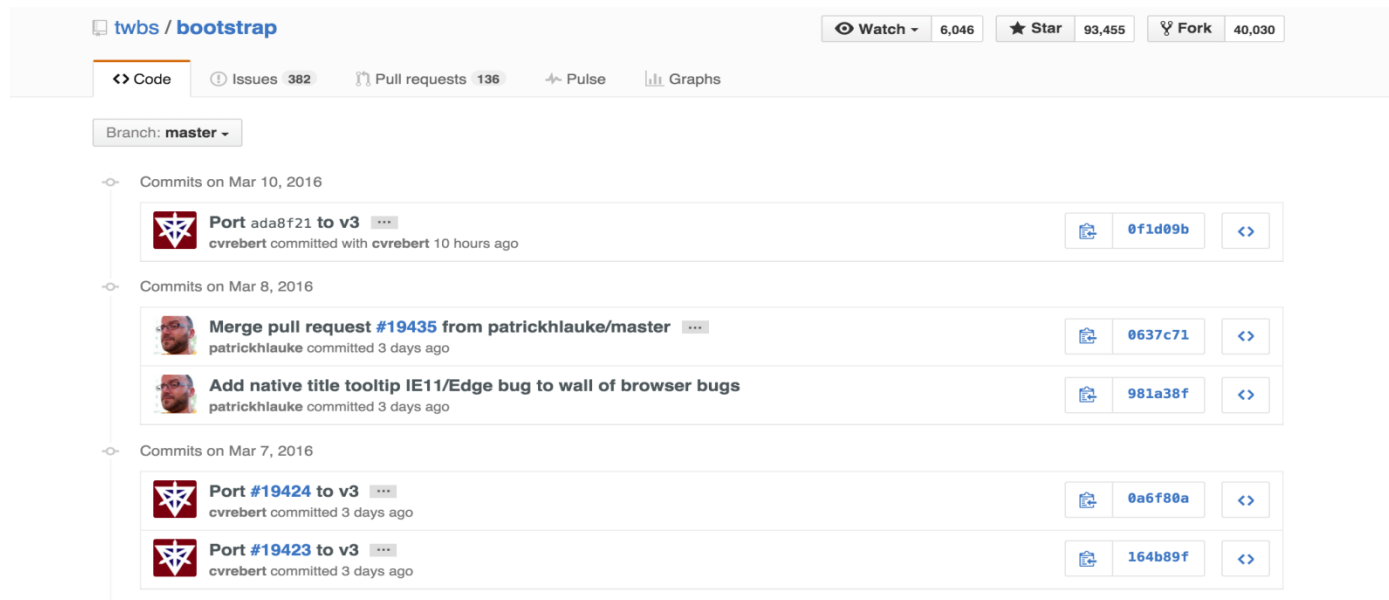
910.1

Bootstrap is a sleek, intuitive, and powerful front-end framework for faster and easier web development, created by [Mark Otto](#) and [Jacob Thornton](#), and maintained by the [core team](#) with the massive support and involvement of the community.

To get started, check out <http://getbootstrap.com>!

# Commit history 보기


- 다음으로 살펴 볼 commit history는 어떤 특정 branch에서 작업이 완료되었을 때 가장 최근 작업이 무엇인지 알아보는 좋은 방법이다. 이를 확인하기 위해 GitHub의 부트스트랩페이지로 이동해 "18,705 commits" 링크를 클릭한다











- 가장 최근의 작업이 목록 위쪽으로 오는 순서로 commit 목록이 표시된다.
- 각 commit을 클릭하면, 변경이 된 이유를 설명하는 commit message가 표시된다.

# Commit history 보기

- commit message 아래로 commit의 일부분으로 추가되거나 삭제 또는 수정된 각각의 파일을 볼 수 있다. 삭제된 콘텐츠는 빨간색으로, 추가된 콘텐츠는 녹색으로 표시된다.

 twbs / **bootstrap**

 Watch ▾ 6,046  Star 93,455  Fork 40,030

 Code  Issues 382  Pull requests 136  Pulse  Graphs


### Port ada8f21 to v3


Update Wall of Browser Bugs entry for #15990


<http://bugzil.la/1139853> was resolved as a duplicate of <http://bugzil.la/924068>


Refs #15990

[ci skip]

 master

 **cvrebert** committed with **cvrebert** 10 hours ago 1 parent 0637c71 commit 0f1d09b86cb75b3be84435f24a4c130d7f9b47c1

 Showing 1 changed file with 1 addition and 1 deletion. Unified Split

2  docs/\_data/browser-bugs.yml View

	@@ -104,7 +104,7 @@
104	104 summary: >
105	105 Right border of ` <select>` menu is sometimes missing when screen is set to uncommon resolution</select>
106	106 upstream_bug: >
107	- Mozilla#1139853
	+ Mozilla#924068
108	108 origin: >
109	109 Bootstrap#15990
110	110



# Pull Request 보기

- pull request는 현재 진행 중인 작업이 무엇인지 알 수 있게 해준다. 프로젝트 페이지 화면에서 오른쪽위의 pull request를 클릭하면 공개된 pull request 목록이 표시된다. 이것은 사람들이 현재 작업하고 있는 기능이나 수정사항을 나타낸다.
- pull request를 살펴보면 사람들이 현재 무슨 작업을 하고 있으며, 버그 수정을 하든 기능 개발을 하든 각가의 변경 사항에 대해 어떤 역할을 하고 있는지 알 수 있다.

The screenshot shows the GitHub interface for the `twbs / bootstrap` repository. At the top, there are statistics for Watch (6,046), Star (93,455), and Fork (40,031). Below this, navigation tabs include Code, Issues (382), Pull requests (136), Pulse, and Graphs. The 'Pull requests' tab is active, showing a list of 136 open pull requests. The list includes filters for 'is:pr is:open', 'Labels', and 'Milestones', along with a 'New pull request' button. The pull request list shows details for four items, including their titles, status (checked or not), labels (e.g., browser bug, docs, v4), and the author.

Author	Labels	Milestones	Assignee	Sort
cvrebert	browser bug, docs, v4	v4.0.0-alpha.3		
andreiduca	css, feature, v3			
cvrebert	js, meta, v4	v4.0.0-alpha.3		
cvrebert	browser bug, docs, v4	v4.0.0-alpha.3		

# Issue 보기


- pull request로 현재 진행되고 있는 버그 수정과 기능을 알 수 있다면, issue를 통해서 프로젝트에 필요한 작업을 넓은 관점에서 볼 수 있다.
- pull request는 주로 issue에 링크되어 있지만, 아무도 작업을 시작하지 않아 pull request가 없는 issue 또한 존재한다. issue 목록을 보기 위해 링크를 클릭하면 모든 공개 issue 목록이 자동으로 나타난다.

The screenshot shows the GitHub interface for the `twbs / bootstrap` repository. At the top, there are buttons for `Watch` (6,046), `Star` (93,455), and `Fork` (40,032). Below these are tabs for `Code`, `Issues` (382), `Pull requests` (136), `Pulse`, and `Graphs`. The `Issues` tab is selected, and a search filter `is:issue is:open` is applied. There are also buttons for `Filters`, `Labels`, `Milestones`, and a green `New issue` button. The main content area displays a list of 382 open issues. The first four issues are visible, each with a title, a status label (e.g., `awaiting reply`), and a comment count. The issues are:

- `a class="btn" in div class="well" - background disappears by hover event in IE` (awaiting reply, 11 comments)
- `.table-hover on table with nested tables also adds hover style to nested tables` (awaiting reply, CSS, 1 comment)
- `Docs aren't clear about whether .card-columns orders cards vertically or horizontally` (docs, v4, 1 comment)
- `text-emphasis-variant mixin links don't darken` (CSS, v4, 0 comments)

# Issue 보기

- 누군가 Bootstrap 사용 중 겪은 어려움을 적어 공유하는 모습이다.

 twbs / bootstrap


[Watch](#) 6,046 [Star](#) 93,455 [Fork](#) 40,032

[Code](#) [Issues 382](#) [Pull requests 136](#) [Pulse](#) [Graphs](#)

## .table-hover on table with nested tables also adds hover style to nested tables #19456

[New issue](#)

[Open](#) Kuzmin opened this issue 5 hours ago · 1 comment



Kuzmin commented 5 hours ago

So, I found myself in a situation where I have some nested tables. Feels like I'm back in the 90s again. The code is something like this;

```
<table class="table table-hover">
  <tbody>
    <tr>
      <td>
        <table class="table">
          <tr>...</tr>
          <tr>...</tr>
          <tr>...</tr>
        </table>
      </td>
    </tr>
  </tbody>
</table>
```

And this adds the hover styles to the nested tables. I felt like this might not be the intended behaviour, so I thought I'd file a bug!

Labels

awaiting reply

CSS

Milestone

No milestone

Assignee



No one assigned

Notifications

[Subscribe](#)

You're not receiving notifications from this thread.

2 participants

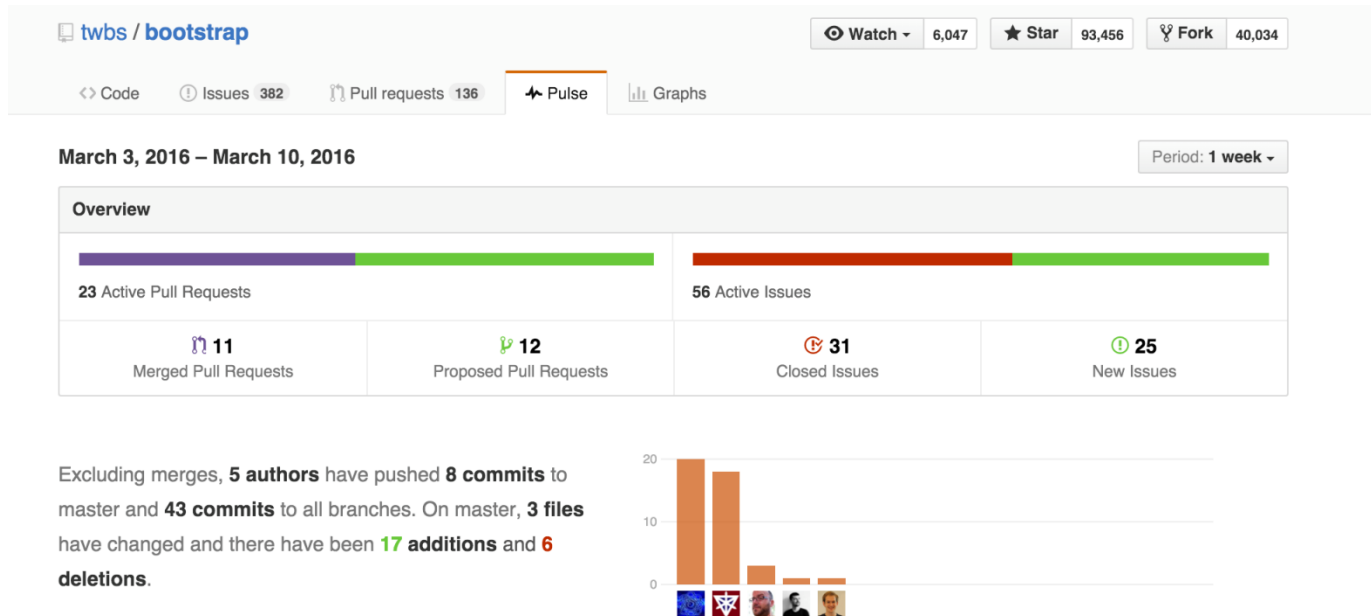
# Project 보기

- 프로젝트개설 및 프로젝트 진행을 돕는 도구 제공

The screenshot displays the GitHub project page for `twbs / bootstrap`. At the top, it shows the repository name and statistics: 248,529 users, 7,310 watches, 133,561 stars, and 65,435 forks. Below this are tabs for Code, Issues (334), Pull requests (46), Projects (3), Security, and Insights. The main section is a Kanban board titled "v4.3.2" with a progress bar and "Updated 9 hours ago". The board has three columns: "To do" (2 items), "Inbox" (12 items), and "Shipped" (37 items). Each column contains cards representing issues or pull requests, with details like issue numbers, authors, and labels (e.g., accessibility, backport-to-v4, v4, v5, css, docs, build). The "Inbox" column also shows "Changes approved" for several items. The "Shipped" column shows items that have been successfully deployed.

# Pulse 보기

- pulse는 프로젝트에 대한 최근의 활동 내용을 엿볼 수 있는 좋은 방법이다. 오른 쪽 위에서 기간을 최종일, 3일, 1주 또는 한 달 등의 기간으로 지정할 수 있다.
- pulse는 merge되거나 추가된 pull request의 수를 먼저 표시한다.또한 얼마나 많은 issue가 마감되고 개설되었는지 표시한다.
- pulse가 활성화된 pull request와 issue의 수를 언급할 때 이것이 단지 각가의 개수를 말하는 것이 아니라, 사용자가 지정한 시기에 개설되고 마감된 request와 issue의 수를 말하는 것임을 이해하는 것이 중요하다.

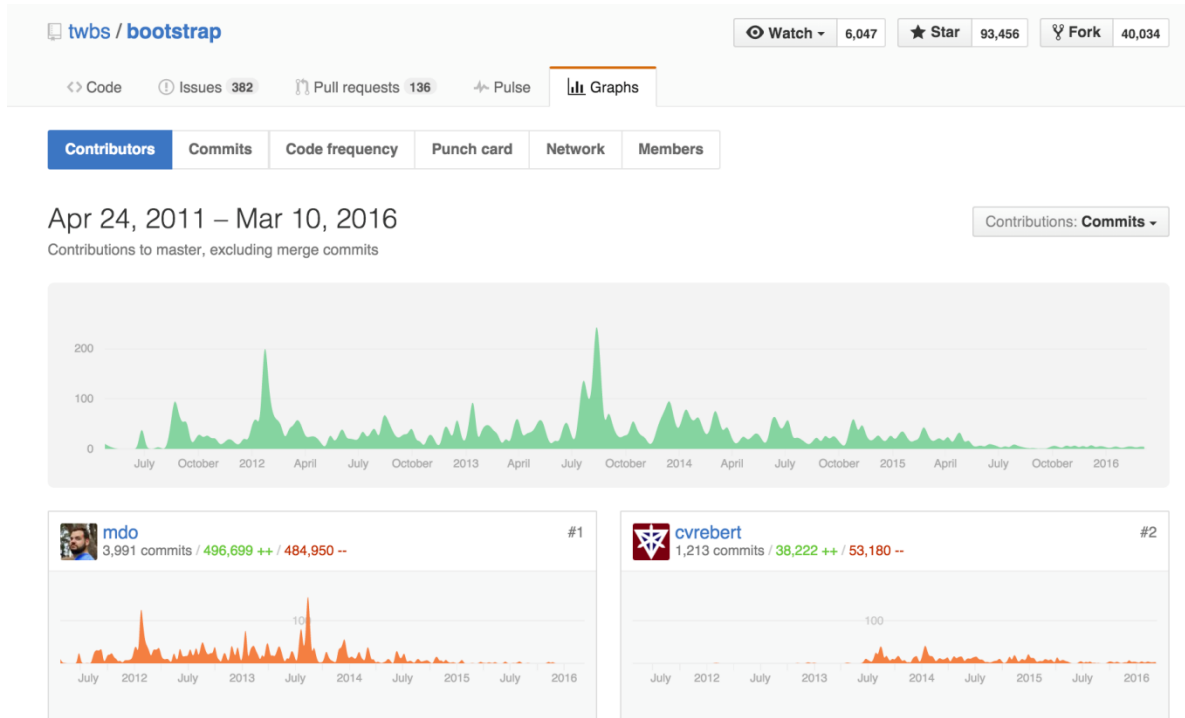


# Pulse 보기

- 예를들어, 이 책을 작성할 시기에, 부트스트랩은 지나주에 총 23개의 “활성화된” pull request가 있었는데 그중에 15개가 merge되고, 8개가 아직 논의가 되고 있다. 하지만 총 28개의 공개 pull request를 갖고 있다. 그 다음으로 나오고 있는 단락은 최근의 변경 내용을 간략히 요약한 것으로 개발자의 수, 마스터의 commit수, 전체 branch의 총 commit수, master branch에서 추가되거나 삭제 또는 수정된 파일의 수를 나열하고 있다.
- 그리고 추가되거나 삭제된 콘텐츠의 라인 수를 표시하는데, 파일에서 텍스트 라인 이나 하나 수정되면 Git은 한 라인이 삭제되고 그 자리에 다른 라인이 추가 된 것으로 인식한다는 것을 알아야한다.
- 오른쪽 막대그래프는 지정된 시기에 commit을 가장 많이 생성한 기여자 (contributor)를 막대그래프로 나타낸다. 그 아래에는 병합되고 논의되고 있는 pull request의 제목과 마감되고 개설 된 issue를 나타낸다.
- pulse는 “Unresolved conversation”로 표시되는 목록으로 끝나는데 이는 새롭게 추가된 댓글이 있으나 아직 마감되지 않는 모든 issue와 pul request의 목록을 말한다.

# GitHub Graphs 보기

- Graphs 화면은 좀 더 긴 기간 동안 프로젝트에 행한 작업을 알 수 있게 해준다.



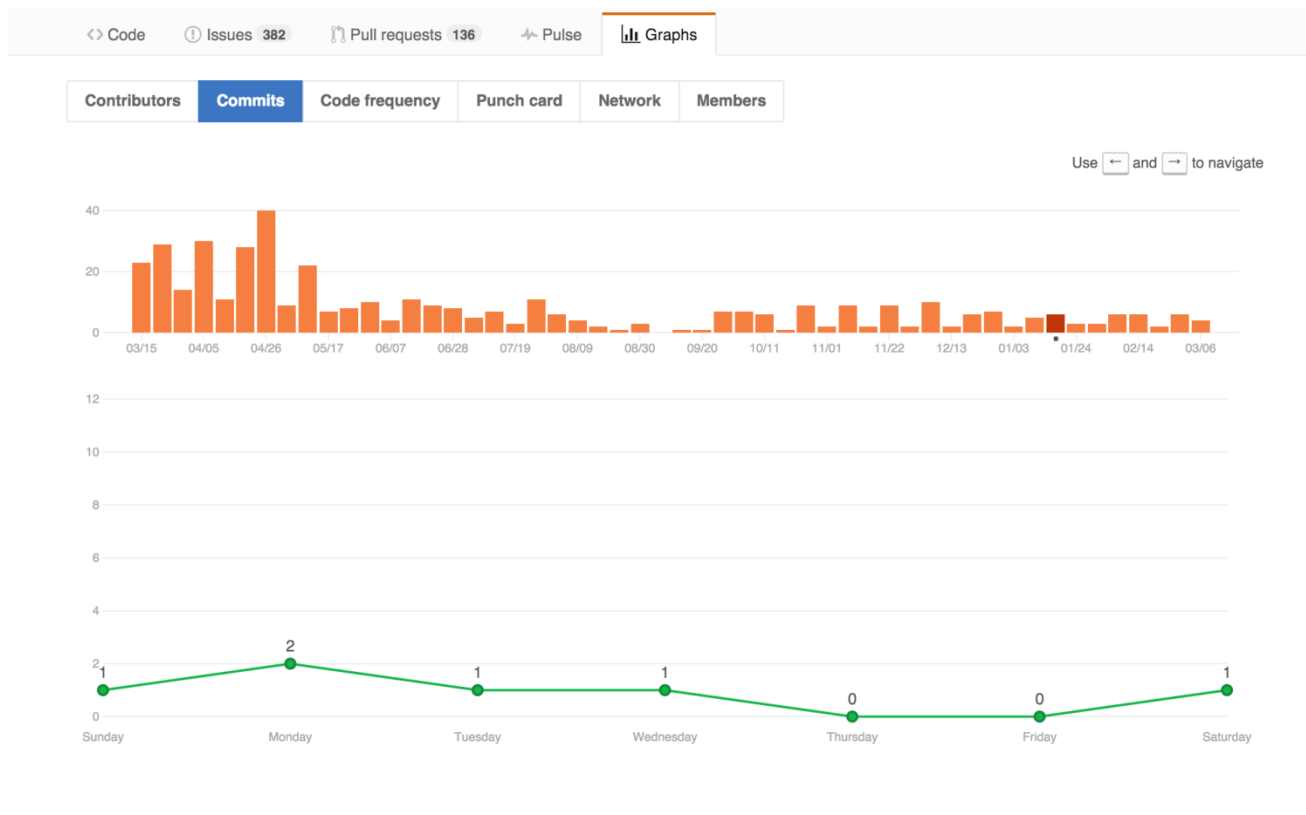
# 기여자(Contributor) Graph

- 기여자 그래프는 일정 기간 동안 commit수, 추가 또는 삭제 수에 근거해 기여자들을 부여준다. 전체 공헌 활동에 대한 그래프가 먼저 나오고, 뒤이어 각 개발자들의 공헌을 나타내는 작은 그래프가 가장 많이 공헌한 개발자 순으로 나온다.
- 기본 commit그래프는 일정 기간 동안 master branch에 만들어진 commit수를 나타낸다. master branch에 merge된 commit만을 나타낸다는 것을 반드시 알아두자. feature branch에 일주일 내내 작업했는데 아직 그 작업이 병합되지 않았다면, 배포 준비가 되어 master branch로 merge되어서야 비로소 공헌내용을 볼 수 있었을 것이다.
- commit에 정해진 기준은 없다. 개발자들이 오류를 조사하거나 무언가를 테스트하지 않고 코드를 작성한다면 대략 5-10분 마다 commit을 생성할 것이다. 그러나 팀에 따라서 어떤 개발자들은 비슷한 시간을 작업해도 다른 사람보다 훨씬 적은 수의 commit을 생성한다.



# commit Graph

- [그림] commit그래프는 프로젝트 전 기간에 걸쳐 얼마나 많은 commit이 발생했는지 나타내는데 활동이 얼마나 활발했는지 그리고 어떻게 달라졌는지 대략적인 추측을 해볼 수 있다.



# commit Graph

- commit Graph를 살펴보는 첫 번째 이유는 매주 얼마나 많은 commit이 발생 했는지 알 수 있기 때문이다. 일주일 간 발생한 commit 수를 하나의 막대로 나타낸 막대그래프를 먼저 표시한다. 주기적인 또는 장기간의 동향을 알아보는 좋은 방법이다.
- 프로젝트의 commit수가 서서히 감소하는가? 개발자 수가 많으면 commit수가 지속적으로 증가하는가? 대부분의 commit이 매달 마지막 주에 발생하는가? 아니면 계절적 추세인가? 이 그래프는 commit 수가 일정 기간 동안 어떻게 변하고 있는지 잘 보여주어 생산성을 대략 추측 해볼 수 있다.

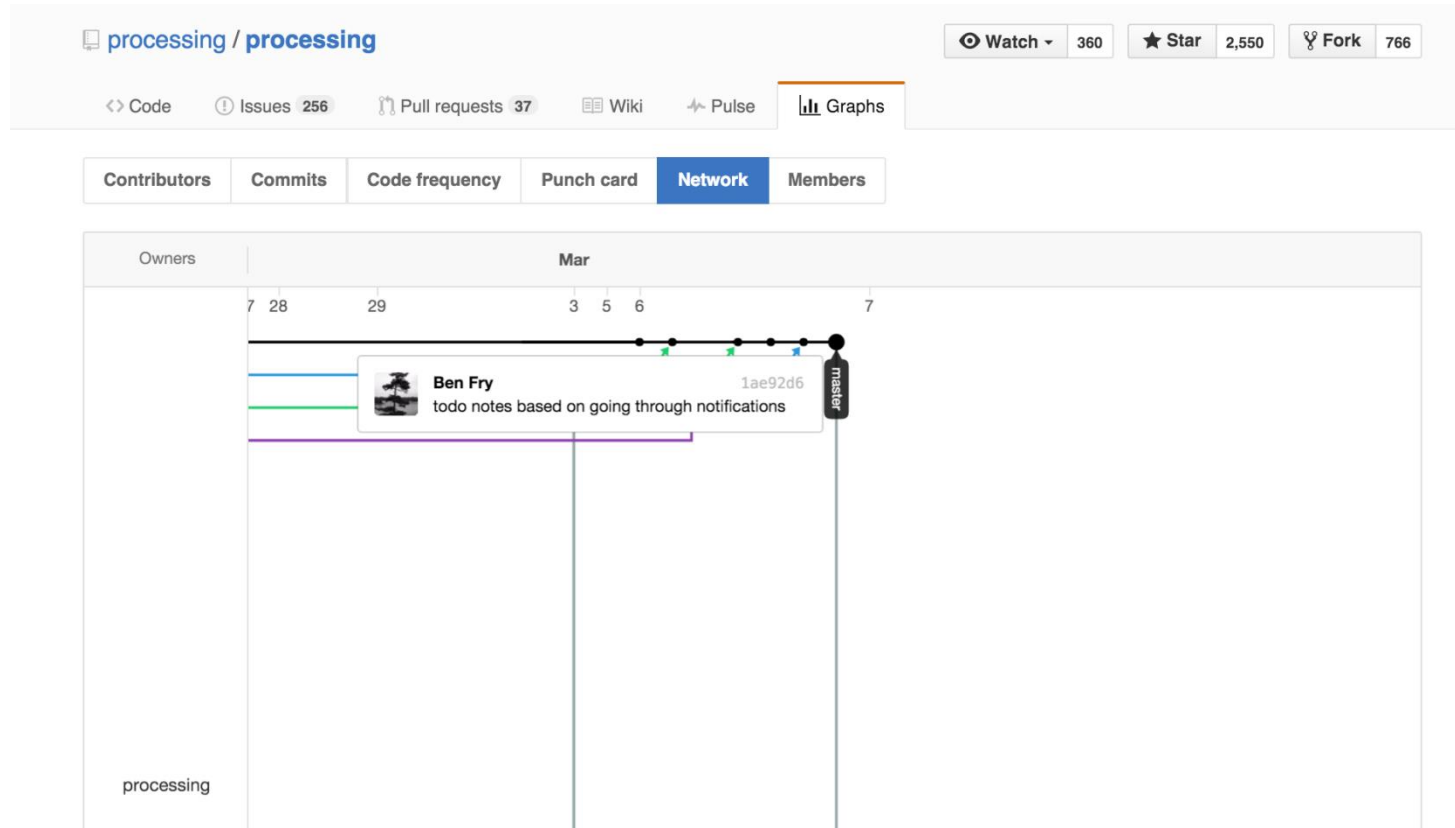
# Code Frequency Graph

- code frequency 그래프는 일정 기간에 프로젝트에 추가되고 삭제된 코드줄수를 보여주며, 특히 코드에 큰 변화가 있을 때 이를 식별하는 데 매우 유용하다. 개발자들이 대규모 리팩토링을 할 때는 commit마다 몇 백 또는 몇 천의 코드줄을 추가하고 삭제한다. 반면 일반적인 비즈니스에서는 commit이 몇 줄만 추가되거나, 수정 또는 삭제된 코드를 포함한다. 이러한 대규모 리팩터링의 경우 commit 수는 많이 변경되지 않을 수도 있지만 추가되고 삭제되는 줄 수는 급증한다. 따라서 코드에 큰 변화가 생겼는지 알고 싶다면 code frequency 그래프를 살펴보면 된다.
- [그림]에서 대규모의 리팩토링이 실행된 것은 2013년 2월과 3월 임을 알 수 있다.



# Network Graph

- 네트워크 그래프는 모든 branch의 수와 해당 branch의 commit 수를 나타낸다. 또한, 기여자(Contributors)가 생성한 fork도 모두 보여준다.



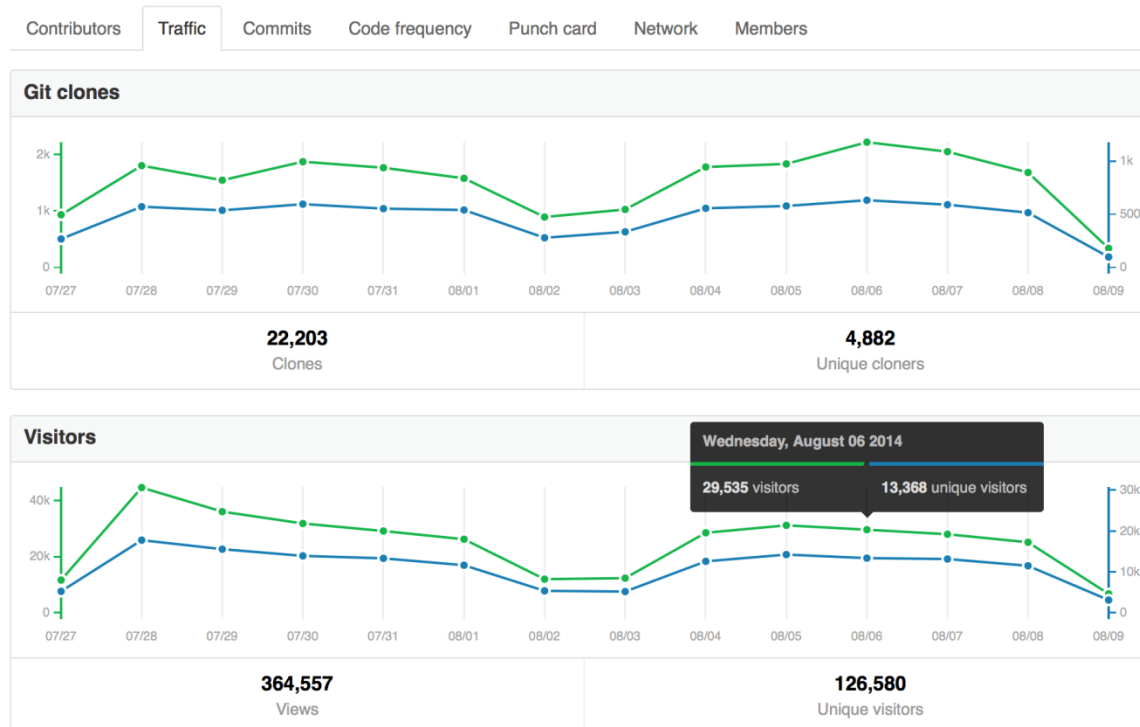
# Members List

- 접근 권한에 상관없이 볼 수 있는 마지막 그래프. 만약 fork의 수가 평소와 다르다면 메시지가 나타나고 전체 멤버 중 일부의 멤버 만 화면에 표시된다.
- 멤버 목록은 누가 repository를 fork 했는지 보여준다. 이들은 초기 parent repository의 협력자가 아니기 때문에 pull request를 이용해 기여하려면 repository 복사본이 필요하다.

The screenshot shows the GitHub interface for the repository `processing / processing`. At the top, there are buttons for `Watch` (360), `Star` (2,550), and `Fork` (766). Below these are tabs for `Code`, `Issues` (256), `Pull requests` (37), `Wiki`, `Pulse`, and `Graphs`. The `Members` tab is selected, showing a list of contributors. The list includes the repository owner `processing / processing` and several forks by other users: `000fan000 / processing`, `3nrique / processing`, `5atk6 / processing`, `5y / processing`, `861186267 / processing`, `9Oeufs / processing`, `a-whitehead / processing`, `abauerenator / processing`, `AbdulazizAlaa / processing`, `abhijit8234 / processing`, and `abigpotostew / processing`.

# traffic Graph

- 이 traffic Graph는 프로젝트의 소유자와 협력자만 볼 수 있다. 트래픽(traffic)그래프는 일정 기간 조회 수와 순방문자 수를 표시하며, 링크된 참조 사이트 목록과 GitHub 프로젝트 사이트에서 가장 인기 있는 콘텐츠를 보여준다. 이는 오픈 소스 프로젝트가 인기가 있는지 알아볼 수 있는 방법이기도 하다. 지금까지의 "README.md 파일, commit, pull request, issue, pulse, GitHub 그래프"를 살펴봄으로써 프로젝트를 더 잘 파악할 수 있을 것이다.



# 깃허브를 이용한 협업

# 브랜치로 커밋하기

- README.md 파일 확장
- 브랜치 생성
- 브랜치에서 Pull request 하기
  1. 프로젝트 페이지 우측 Pull Request 탭의 New pull request를 클릭한다. 이때 바로 생성이 되지 않는 것은, Github가 어느 브랜치에 생성해야 할지를 몰라서이다.
  2. 왼쪽에 base:master가 있으면 올바른 것. pull request를 생성하고 이것이 받아들여지면 마스터 브랜치로 병합된다.
  3. 이어 compare:master 버튼을 클릭해 어떤 브랜치에 pull request를 생성해야 하는지 지정해주어야 한다. compare:브랜치는 마스터 브랜치로 병합되었으면 하는 브랜치이다.



# 브랜치 병합하기

- 브랜치 병합은 누가 하는가? pull request를 생성한 사람이 병합할지, 아니면 생성자가 아닌 다른 사람이 할지에 대한 질문이 많다. 정답은 없지만 다음 사항들을 고려해보자.
- 보통 pull request는 생성한 사람이 병합하도록 조언한다. 대부분의 경우 pull request를 생성한 사람이 해당 코드에 대해 가장 잘 알고 있는 사람이기 때문이다. 단, 반드시 다른 사람들로부터 `:+:[^1]`를 최소한 둘 이상 받았을 때 한다.
- 많은 회사에서는 pull request를 생성한 사람은 병합 할 수 없다는 원칙을 세운다. 그 이유는 아무 피드백 없이 pull request를 생성하고 병합하지 않도록 하기 위함이다. 의도는 좋지만 그다지 이상적인 권고사항은 아니다.

# 브랜치 병합하기



**This branch has no conflicts with the base branch**

Merging can be performed automatically.



**Merge pull request**

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



**Merge pull request #2 from tadm/samle**

Create blockquote.html



**Confirm merge**

Cancel



**Pull request successfully merged and closed**

You're all set—the `sample` branch can be safely deleted.



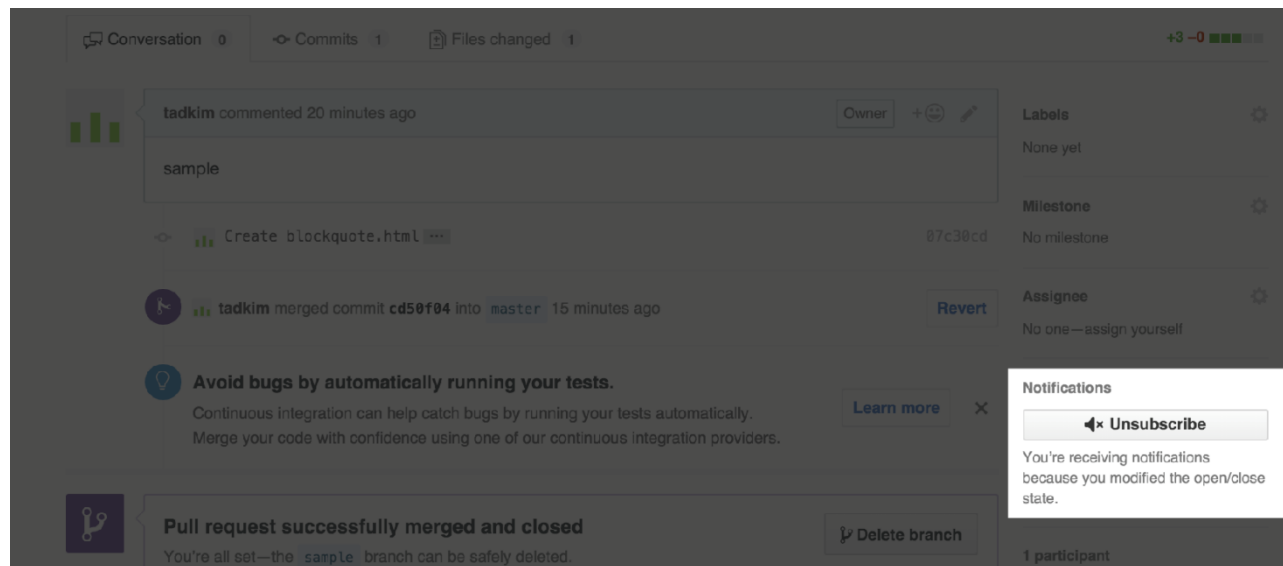
**Delete branch**

# Pull request 알림 기능(Notification)

- 다음 사항에 해당되는 사람은 기본적으로 pull request에 가입된다.
- Pull request 알림대상
  - pull request를 작성한 사람
  - pull request에 댓글을 단 사람
  - (해당사항에 대해)commit을 생성한 사람
  - @mention이 된 사람
  - Pull request 알림 기능
- pull request에 새로운 댓글 등록시
- pull request에 대한 commit 생성시
- pull request가 병합 또는 닫힐 때

# Pull request 알림 기능(Notification)

- pull request가입 확인 및 알림기능 해제방법 페이지의 오른쪽을 보면 pull request에 가입되어 있음을 알 수 있으며, 만약 가입되어 있는 pull request에 더이상 관심이 없다면 Unsubscribe버튼을 클릭해 해당 pull request에 대한 알림을 끄면 된다.
- pull request가입 방법 반대로 눈여겨보고 싶은 pull request인데 가입되어 있지 않을 경우 Subscribe 버튼을 클릭하여 해당 pull request에 대한 모든 활동 내용에 대한 알림을 받으면 된다.



# Pull Request의 좋은 연습사례

- 개발환경에 익숙해지는데 도움이 될 만한 연습 방법을 정리&공유 한다.
- 모든 것에 **pull request** 만들기
  - : 버그를 수정하거나 새 기능을 추가할 때마다 반드시 브랜치에서 작업하고 마스터로 병합하기 전에 다른 사람들에 의견을 받기 위한 pull request를 만든다.
- 알아보기 쉬운 제목 짓기
  - : 작업이 어떻게 진행되는지 알고자 다른 팀원들이 pull request 페이지를 들여다볼 것이다. 제목을 보고 무슨 작업인지 알아챌 수 있어야 한다.
- 댓글에 공들이기
  - : @mention되지 않았더라도 공들여 댓글을 작성한다. 프로젝트의 진행상황을 잘 알 수 있을 뿐만 아니라 작업의 전체적인 질을 향상시킬 것이다.
- 주요인물에 대한 @mention
  - : 마케팅, 법무, 운영팀의 피드백을 받고 싶으면 필요한 사용자를 @mention해서 그들이 pull request를 보도록 하고, 피드백 할 수 있는 환경을 만들어준다.

# Pull Request의 좋은 연습사례

- **테스트 시행**

: 최소한 한 명의 개발자라도 pull request에서 최근 변경을 다운로드하고, 적합한 브랜치인지 확인하며, 자동화된 테스트를 반드시 실행하도록 한다. 중대한 변경 내용을 그저 눈으로 코드를 보서는 충분히 알 수 없다.

- **pull request를 승인하는 명확한 정책 갖기**

: 대부분의 회사에서는 pull request가 병합되기 전에 pull request의 원 작성자 외에 한 두명의 사람들이 검토를 하고 :+1:을 제공하도록 하고 있다.

- **pull request와 issue의 차이**

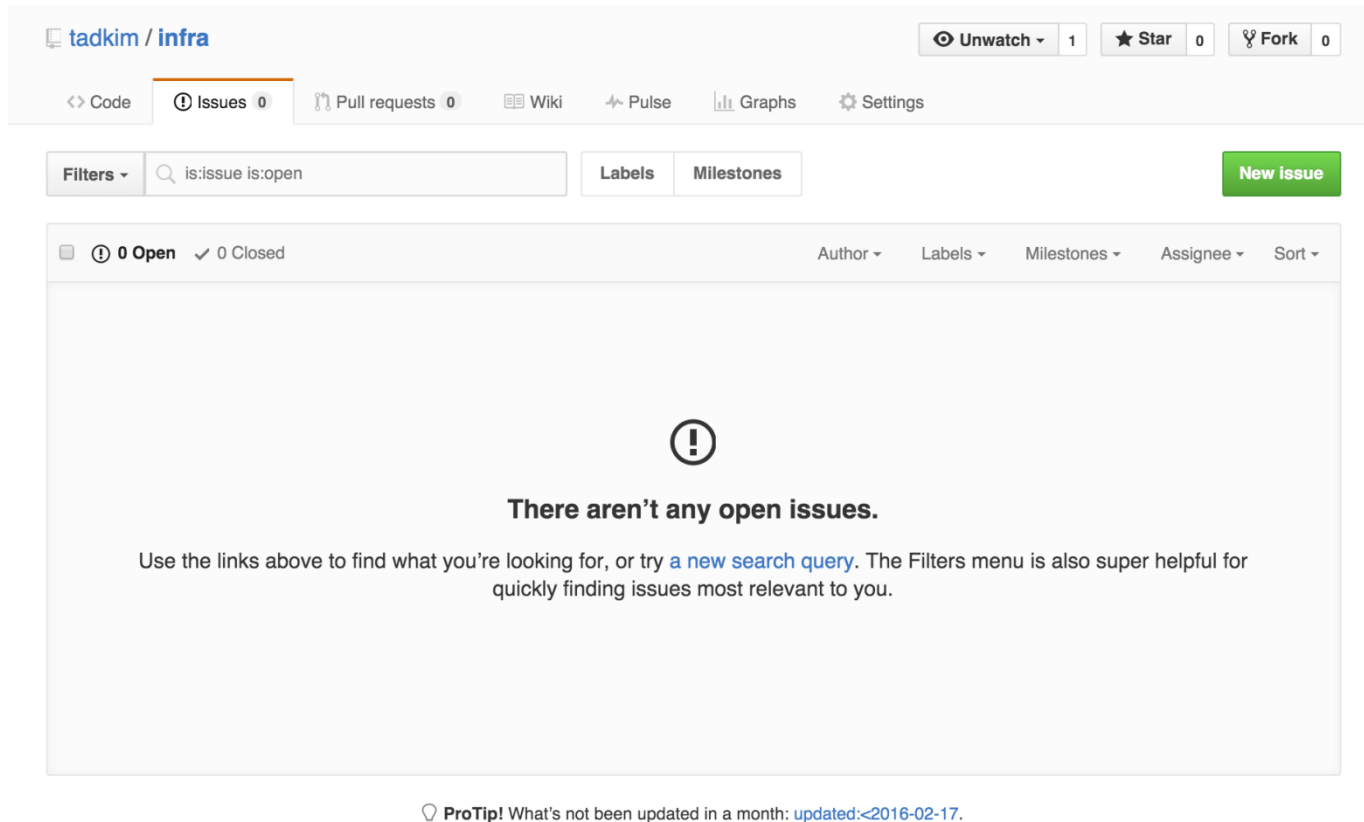
: pull request가 완료도니 작업의 통합에 이용되는 반면, Issue는 버그 또는 논의하거나 개발해야하는 새로운 기능 요구사항을 설명하는 데 이용한다.

# Github 협업 #2 Issue

- Issue는 수정해야 하는 버그나 새로 개발해야 하는 기능을 관리하는 가볍고 사용하기 편리한 도구이다.
- 보통 새로운 프로젝트를 시작할 때 GitHub Issue를 사용해 버그나 기능을 관리한다. 그리고 Issue가 제공하지 않는 기능이 있을 때만 Pivotal tracker, JIRA, Light-House, Trello, Asana 등과 같은 툴을 이용한다.

# Issue 생성하기

- 새로운 이슈를 생성하기위해서 Github의 프로젝트 페이지에서 Issue 탭을 클릭 한다.






# Issue 생성하기

## New Colony Project #3

**Open** tadkim opened this issue just now · 0 comments

Edit New issue



tadkim commented just now

Owner + 🗨️ ✎️

새롭게 시작한 Colony.  
첫 번째 프로젝트는 꼭, 반드시, Git&Github를 활용 해볼 예정입니다.  
관련하여 의견있으시면 남겨주세요.

Labels  
None yet


Milestone  
No milestone

Assignee

## New Colony Project #3

**Open** tadkim opened this issue a minute ago · 0 comments

Edit New issue



Write Preview

AA<sup>~</sup> B  *i “ < > ↻ ⋮ ≡ ☑️ @ 📌*

#새롭게 시작한 Colony.  
첫 번째 프로젝트는 꼭, 반드시, Git&Github를 활용 해볼 예정입니다.  
관련하여 의견있으시면 남겨주세요.  
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

📖 Styling with Markdown is supported

Cancel Update


Labels  
None yet

Milestone  
No milestone


Assignee

Assign someone to this issue

Filter people

 tadkim

because you authored the thread.



Write Preview

AA<sup>~</sup> B  *i “ < > ↻ ⋮ ≡ ☑️ @ 📌*

# Github 협업 #3 Wiki

- Wiki기능은 협력자들이 다량의 연결된 페이지를 개발하기 쉽게 해주는 간단한 콘텐츠관리 시스템이다.
- 보통 Github wiki는 사용자 문서 개발자 문서 또는 모두를 캡처해 프로젝트와 관련된 모든 정보가 GitHub를 통해 접근 가능하도록 하는 데 사용된다.