

교육 및 스터디 계획

교육 및 스터디 주제



Javascript Runtime



GIT

Version Control



GITHUB

Designer / Publisher 공통사항

Designer – Node JS & npm 기초, GIT
사용법

Publisher – Node JS & npm 기초 및 기
본 예제, GIT 사용법



Grunt



Gulp

Task Runner



Bower

Package
Manager



SASS

CSS
Preprocessor

Publisher

학습 계획 - 교육 단계

01

Git, Github 기초

Git, Github 기본개념 및 기본 사용법

03

Node js 기초 예제

Node js , Express 웹서버 기초예제,
Git 기초 사용법

05

Work Flow 향상

Grunt, Gulp, Bower, Sass

02

Node js 기초 교육

Node js 기본개념 및 기초

04

Node js 활용 예제

Node js, Express, Mongodb 예제

06

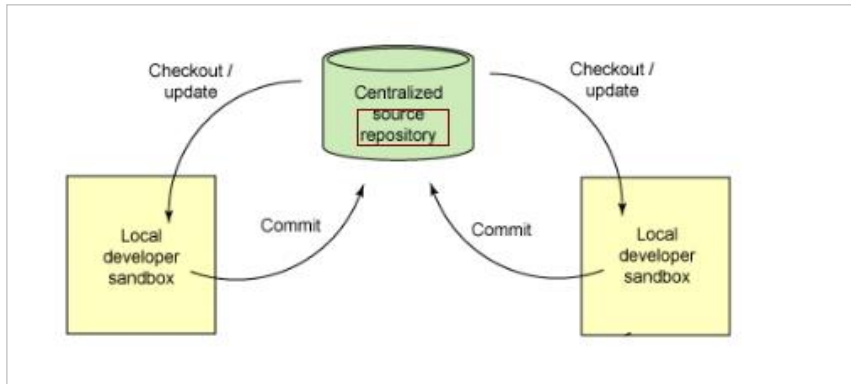
Node Web App 예제

Node Web App 구축 실습

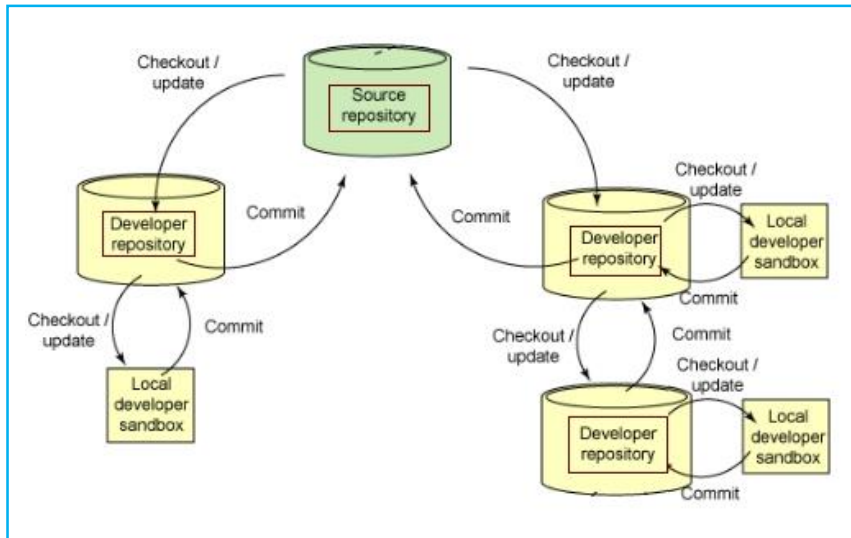
Git, Github 소개

Git, Github 익히기

Git 간단 소개



중앙집중형 버전관리 모델 (대표사례: SVN)



분산형 버전관리 모델 (대표사례: GIT)



git는 2006년경 BitKeeper라는 리눅스 커널 개발에 사용하던 분산형 패치 도구에 대한 대안으로 리누스 토발즈가 직접 개발한 분산형 소스 컨트롤(Source Control Management) 이다.

주요 특징

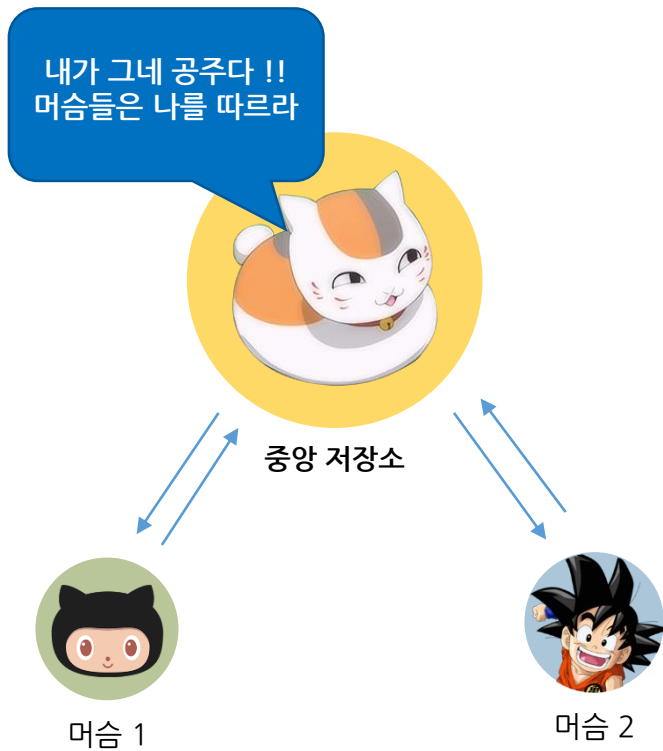
- 분산된 개인 저장소 사용 (로컬컴퓨터의 프로젝트 디렉토리안의 .git/디렉토리)
- 분산된 개인 저장소를 사용하여 거의 모든 명령을 로컬에서 실행하며 데이터의 안전성이 높다.
- 중앙 프로젝트 저장소에 Pushing하여 중앙저장소에 반영하는 형태

Git, Github 익히기

Git 간단 소개

중앙집중형(SVN) 과 분산형 (GIT)

SVN



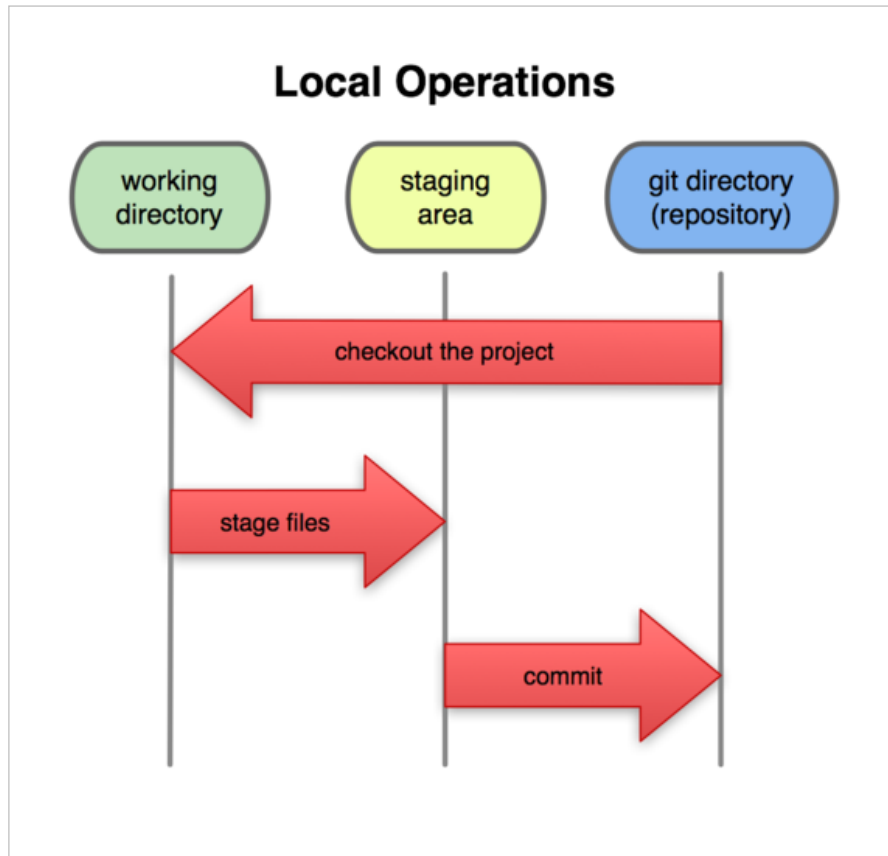
GIT



Git, Github 익히기

Git 간단 소개

기본적인 작업 흐름 - 로컬환경



Git은 파일을 Committed, Modified, Staged 이렇게 세가지 상태로 관리한다.

- **Committed**

데이터가 로컬 저장소에 안전하게 저장됐다는 것을 의미한다.

- **Modified**

수정한 파일을 아직 로컬 저장소에 커밋하지 않은 것을 말한다.

- **Staged**

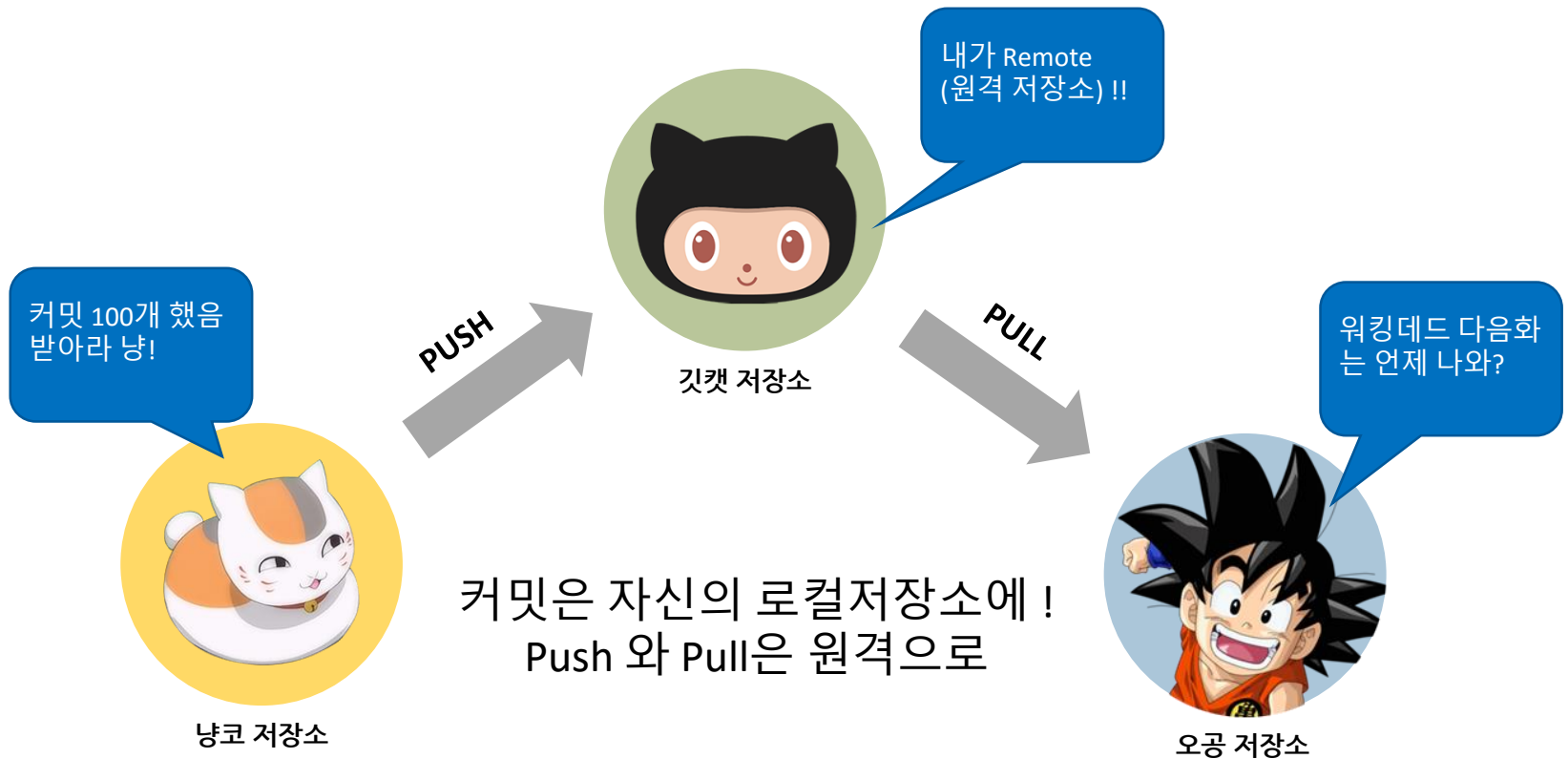
현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태를 의미한다.

Git 디렉토리에 있는 파일들은 Committed 상태이다. 파일을 수정하고 Staging Area에 추가했다면 Staged이다. 그리고 Checkout하고 나서 수정했지만, 아직 Staging Area에 추가하지 않았다면 Modified이다.

Git, Github 익히기

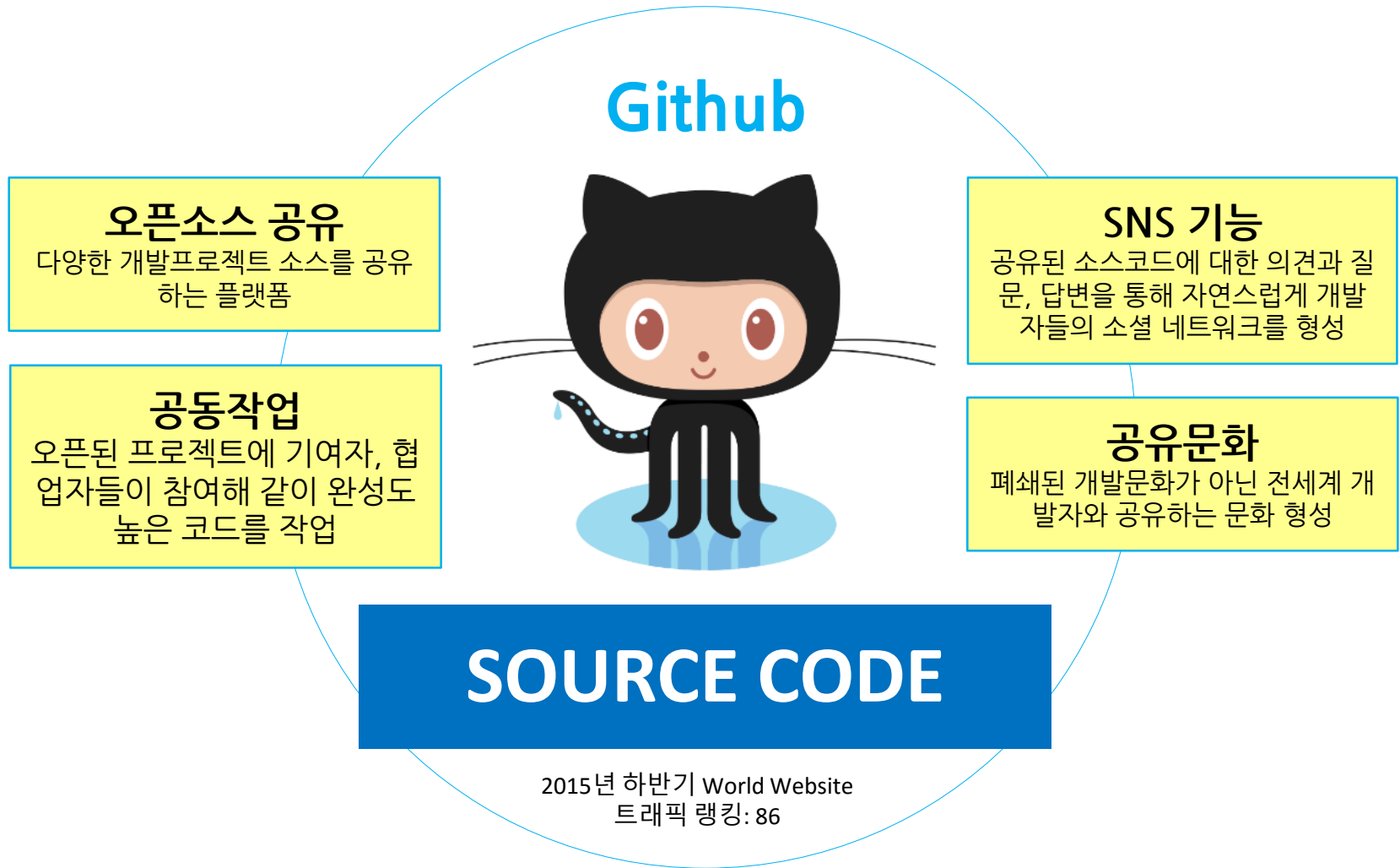
Git 간단 소개

기본적인 작업 흐름 - 네트워크 환경



Git, Github 익히기

Github 간단 소개



Git 실무활용을 위한 필수 학습내용

Git 저장소 만들기

git init, git clone

수정하고 저장소에 저장하기

git add, git status, git commit, git rm

커밋 히스토리 조회

git log

되돌리기

git commit --amend, git checkout

리모트 저장소

git remote, git fetch, git pull, git push

브랜치 이해

git branch, git checkout

브랜치와 Merge 기초

git checkout, git merge, git branch -d

브랜치 관리

git branch (-v, --merged, -d)

Remote 브랜치

git remote, git push

Git 서버 설치 및 설정

[Git 공식 사용설명 Book](#)

Git 에서 제작한 공식 사용설명문서를 온라인이나 PDF파일로 받아볼 수 있다.
특히 한글번역이 잘 되어있어 좋은 학습자료역할을 한다.

Git, Github 설치하기

Git, Github 익히기

Git을 사용하는 두가지 방법

Command Line Interface

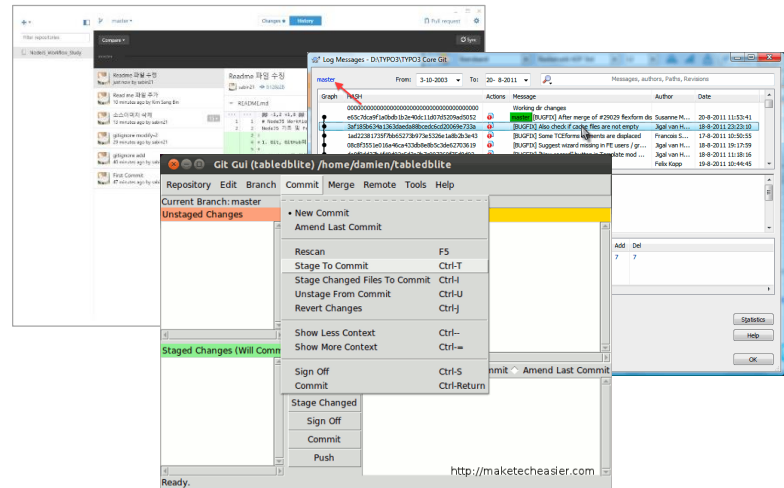
```
MINGW64~/c:/Users/sabin/Desktop/test_directory
sabin@sabin-win10 MINGW64 ~/Desktop/test_directory (master)
$ git add *
sabin@sabin-win10 MINGW64 ~/Desktop/test_directory (master)
$ git commit -m "index.html 파일 추가"
[master f6cd5e0] index.html 파일 추가
2 files changed, 11 insertions(+)
create mode 100644 Thumbs.db
create mode 100644 index.html
sabin@sabin-win10 MINGW64 ~/Desktop/test_directory (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
sabin@sabin-win10 MINGW64 ~/Desktop/test_directory (master)
$
```

Git Bash 화면

Graphic User Interface



시스템 터미널이나 Shell 명령창에 명령어를 입력하는 인터페이스 방식은 프로그래머가 아닌 사용자들에게는 생소하고 거부감이 들 수 있다.

이를 극복하고 CLI 환경에서 Git을 익혀야 하는 이유는 CLI 환경에서 Git을 사용할 줄 알게 되면 GUI 툴을 쉽게 다루게 되지만 그 반대는 안되기 때문이다. 또한 Text로 된 명령어를 입력하며 Git을 사용하면 Git의 핵심적인 개념과 기본 구조를 자연스럽게 이해하게 되기 때문이기도 하다.

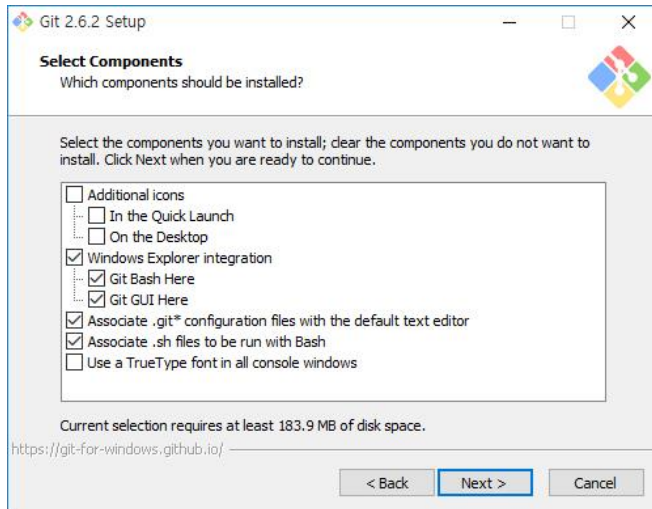
Git, Github 익히기

Git 사용 준비

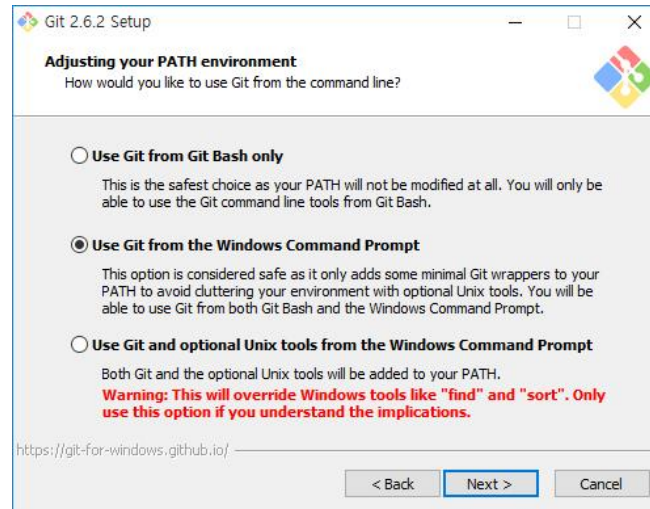
Git 설치하기

1. 아래 링크에서 자신의 시스템과 맞는 버전의 installer를 다운로드 한다.

<https://git-scm.com/downloads>



2. 설치화면의 기본 체크된 상태로 다음으로 넘어간다.



3. Use Git from the Windows Command Prompt를 선택하고 설치를 마무리한다.

Git, Github 익히기

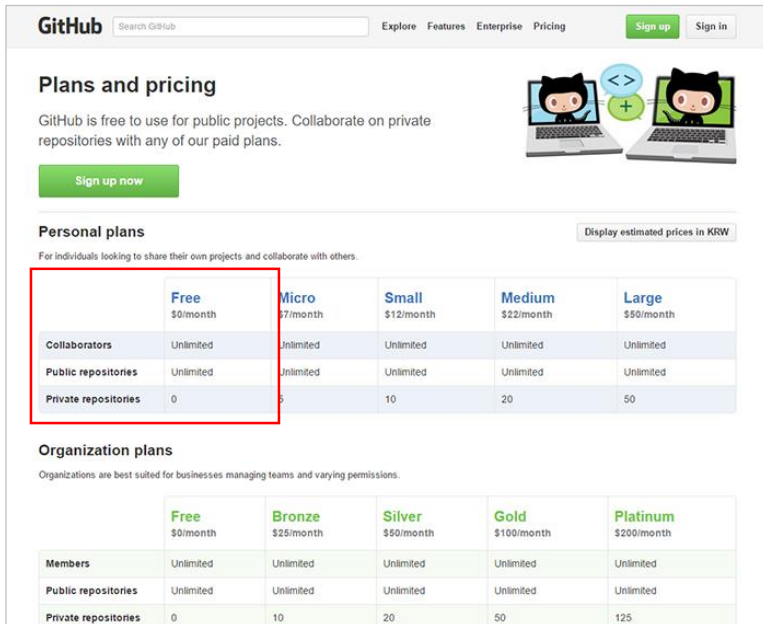
Github 사용 준비

Github Desktop 설치 및 Github 계정 등록

1. 아래 링크에서 윈도우용 Github Desktop Installer를 다운받아 설치한다.

<https://desktop.github.com/>

Github Desktop를 설치완료하고 실행하면 **Github의 계정 정보**를 요구한다.



Plans and pricing

GitHub is free to use for public projects. Collaborate on private repositories with any of our paid plans.

[Sign up now](#)

Personal plans

For individuals looking to share their own projects and collaborate with others.

	Free	Micro	Small	Medium	Large
	\$0/month	\$7/month	\$12/month	\$22/month	\$50/month
Collaborators	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Public repositories	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Private repositories	0	5	10	20	50

Organization plans

Organizations are best suited for businesses managing teams and varying permissions.

	Free	Bronze	Silver	Gold	Platinum
	\$0/month	\$25/month	\$50/month	\$100/month	\$200/month
Members	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Public repositories	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Private repositories	0	10	20	50	125

2. <https://github.com/pricing> 에서 계정을 등록을 시작한다. 필요한 정보를 입력하고 Free Plan으로 계정등록을 완료한다.

3. 등록된 계정정보를 **Github Desktop에 입력**하여 Github Desktop의 사용준비를 완료한다.

Git, Github 익히기

Git, Github - 기초 사용법

Git CLI 맛보기

git 을 설치하고 가장 먼저 해야 할 일은 git의 사용환경을 설정하는 것이다.

```
git config --global user.name "John Doe"  
git config --global user.email johnoe@example.com
```

1. 사용자 정보 설정

github 계정을 만들었으면 그때 사용한 User name과 이메일 주소를 입력한다.

```
git config --global core.editor emacs
```

2. 기본 편집기 설정

git에서 사용할 텍스트 편집기를 고른다. 따로 설정을 하지 않으면 기본적으로 vi, vim을 사용한다.

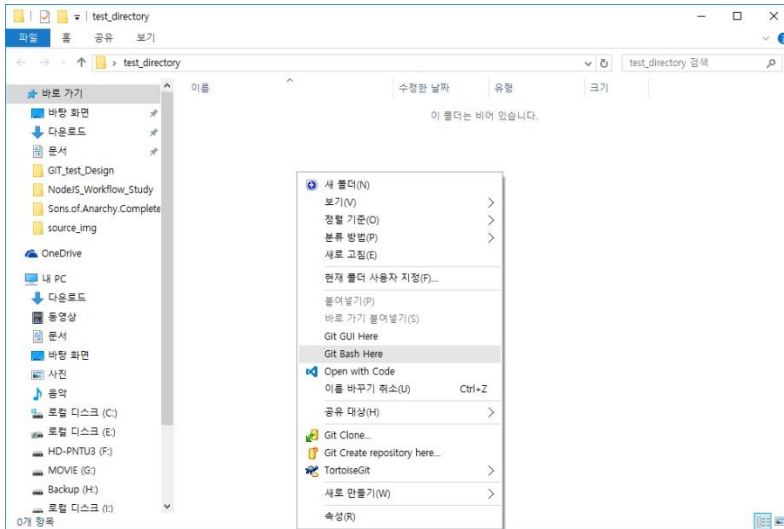
설정파일 위치 (Windows)

git의 사용환경 설정 파일은 .gitconfig 파일이며 일반적으로 C:\사용자\사용자명\ 위치에 저장된다.

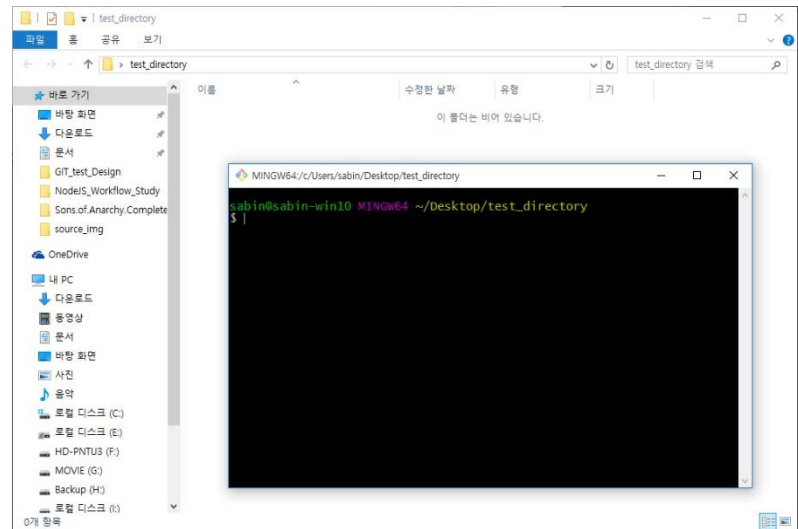
Git Bash 맛보기

Git, Github 익히기

Git Bash - 맛보기



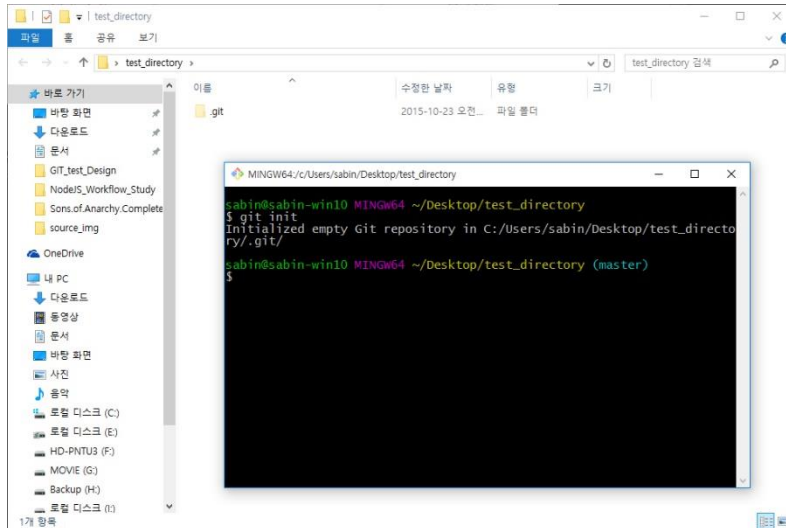
1. 새로운 디렉토리를 생성하고 마우스 오른쪽 클릭으로 메뉴를 열어 "Git Bash Here" 를 선택한다.



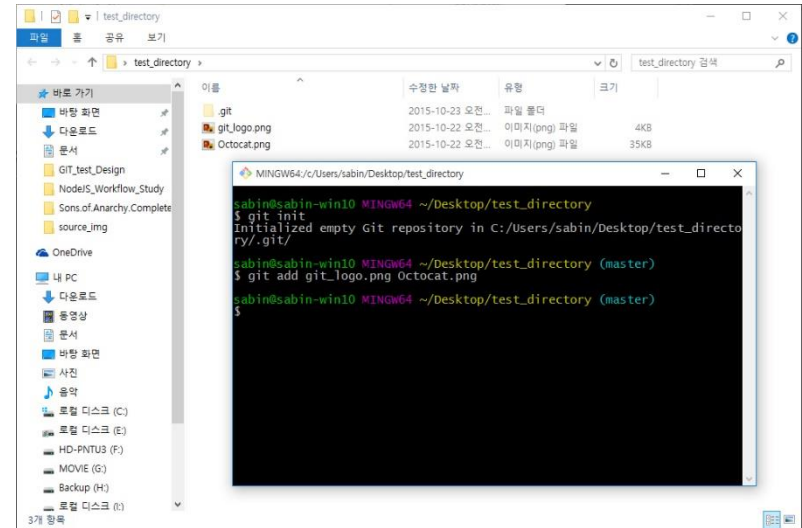
2. Git CLI을 실행할 수 있는 Git Bash 명령어 창이 뜬다.

Git, Github 익히기

Git Bash - 맛보기



3. **git init** 명령어를 입력하고 엔터키를 누른다.

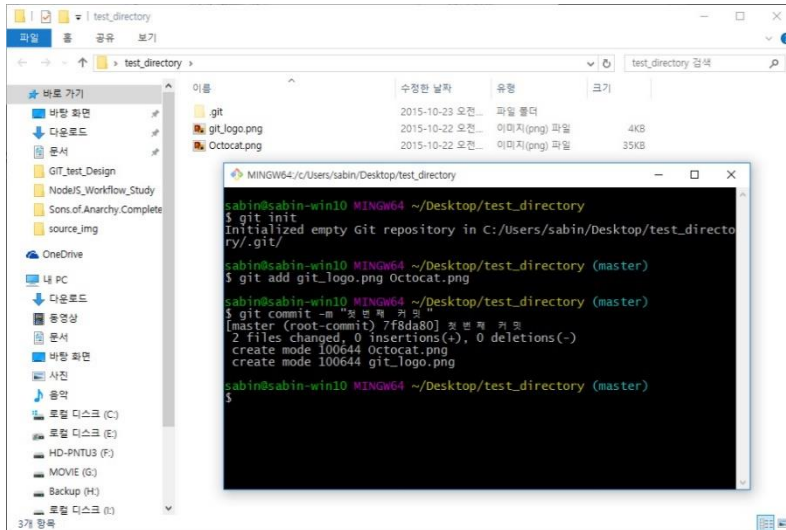


4. 간단한 이미지 파일을 두개 복사하여 작업중인 디렉토리에 붙여넣는다. 그리고 Git Bash 창에 **git add <file name> <file name>**을 입력하고 엔터키를 누른다.

git add <file name> 명령어는 해당 파일을 Git에서 관리대상으로 만든다는 의미이다.

Git, Github 익히기

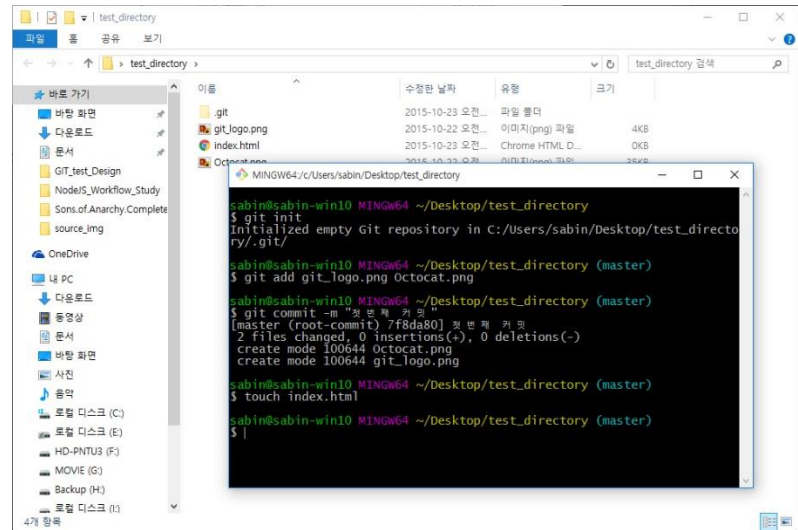
Git Bash - 맛보기



5. `git commit -m "첫번째 커밋"` 명령어를 입력하고 엔터키를 누른다.

`git commit` 명령어는 Stage 상태에 있던 파일들을 git 로컬저장소에 "확정하여 저장" 한다는 의미로 이해하면 된다.

`git commit` 다음에 입력하는 `-m "message"` 는 커밋의 Summary 정보를 입력하는 것으로 이 커밋이 어떤 커밋인지 이해를 돕는 제목을 입력하면 된다. 이 Summary 정보를 입력하지 않으면 Commit명령이 취소된다.



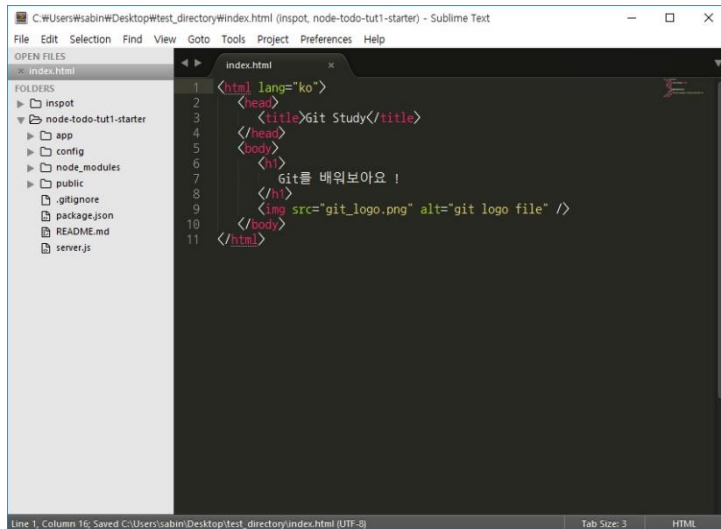
6. `touch index.html` 을 입력하고 엔터키를 누른다. 엔터키를 누르고 나면 작업중인 디렉토리에 index.html 파일이 생성된다. `git add index.html`을 입력하여 수정된 index.html 파일을 stage 상태에 올린다.

그리고 다시 `git commit -m "두번째 커밋"` 명령어를 입력하여 Git저장소에 저장을 완료한다.

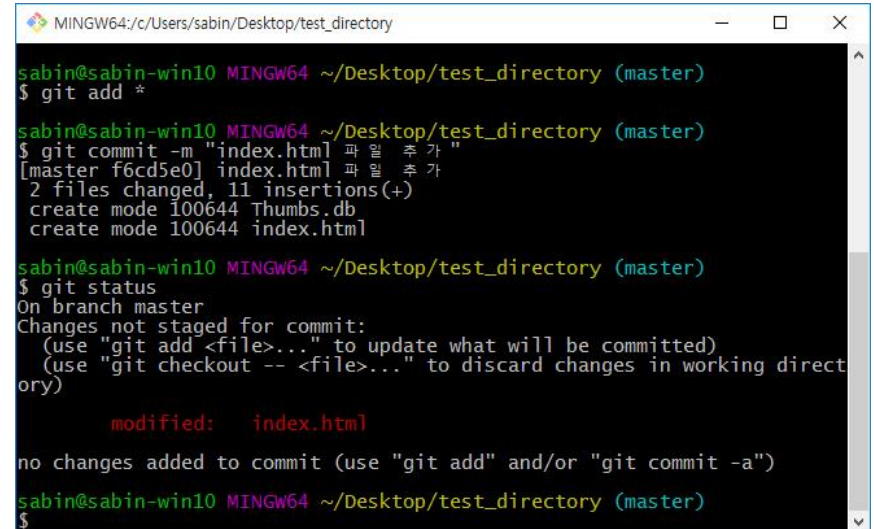
`touch filename.확장자` 명령은 Bash shell에게 비어있는 파일을 생성하라고 지시한것이다.

Git, Github 익히기

Git Bash - 맛보기



7. 에디터 프로그램으로 생성된 index.html파일을 열어 간단한 내용을 입력하고 저장한다.



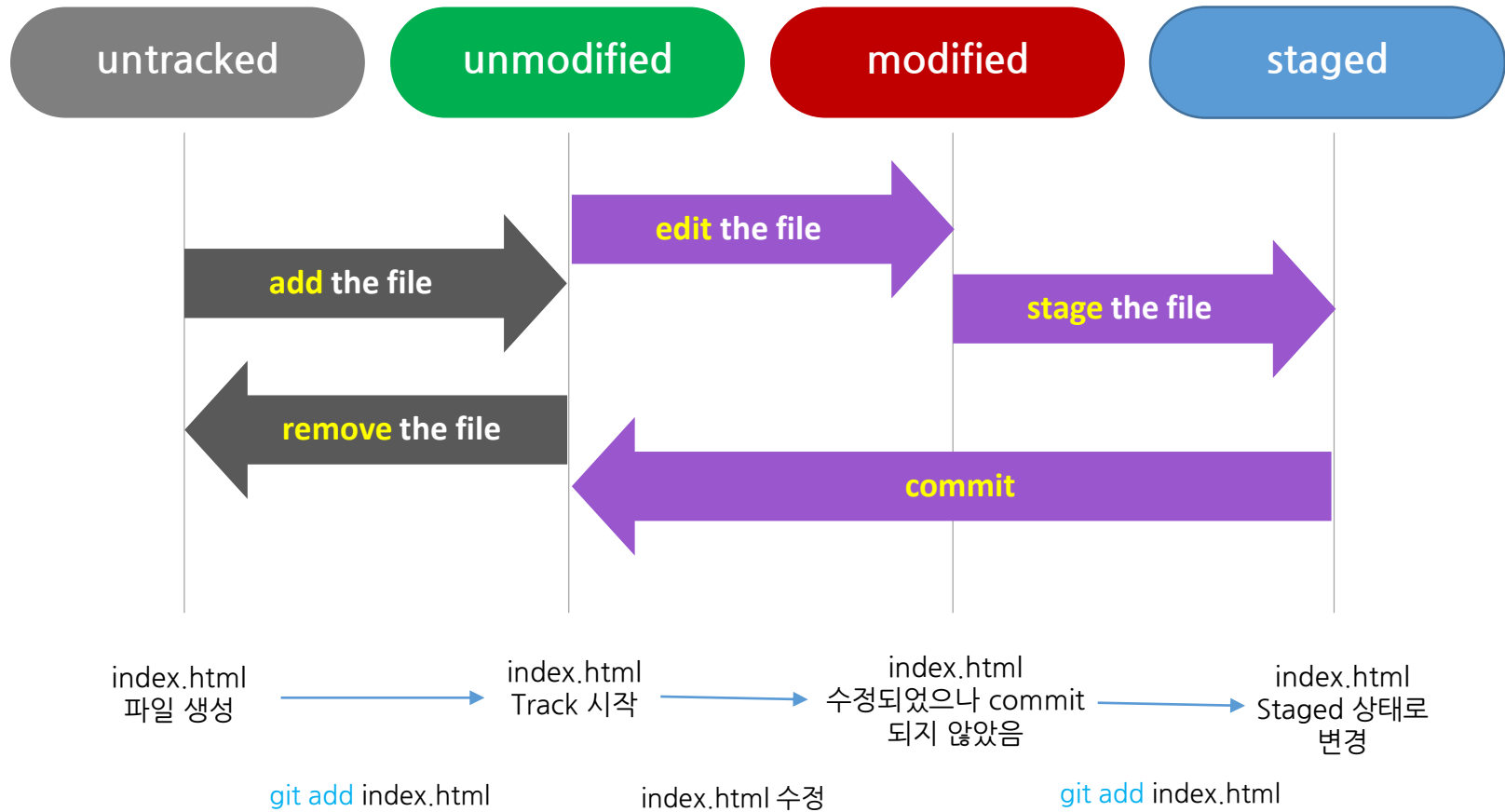
8. `git status` 을 입력하고 엔터키를 누르면 현재 git 상태를 보여준다. index.html파일이 modified 된 상태라는 것을 알려준다. `git add index.html`을 입력하면 수정된 index.html이 다시 stage 상태에 올라간다. 그리고 다시 `git commit -m "index.html 수정"`을 입력하고 실행하여 커밋을 완료한다.

일단 커밋을 하여 관리대상이 된 파일이라도 수정을 하면 다시 `git add` 명령으로 Stage에 올리고 Commit를 해야 수정된 부분이 반영되어 Commit이 완료된다. 중간의 add 과정을 생략하려면 `git commit` 명령에 `-a` 옵션을 추가하면된다.

Git, Github 익히기

Git Bash - 맛보기

Git 파일의 Lifecycle



Github Desktop

기초사용법

Git, Github 익히기

Github Desktop - 기초사용법

저장소 만들기

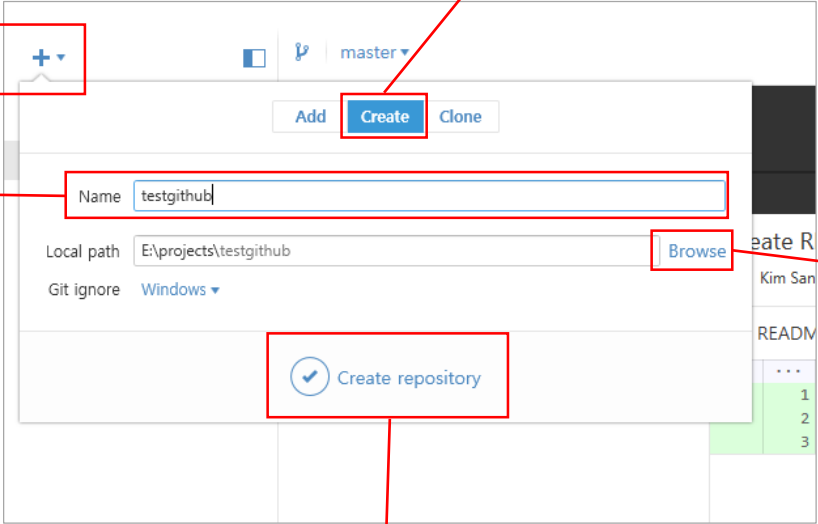
1. 클릭하여 새로운 저장소를 생성한다.

2. 기본적으로 Create 탭이 활성화 되어 있다.

3. 저장소 이름을 입력

4. 저장소의 로컬 위치를 지정한다

5. 저장소 생성 완료



The screenshot shows the 'Create repository' dialog in Github Desktop. The dialog has a title bar with a close button and a 'master' branch selector. Below the title bar are three buttons: 'Add', 'Create' (highlighted with a red box), and 'Clone'. The 'Create' button is also highlighted by a red arrow from annotation 2. Below the buttons is a form with the following fields: 'Name' (containing 'testgithub'), 'Local path' (containing 'E:\projects\testgithub'), and 'Git ignore' (set to 'Windows'). A 'Browse' button is next to the 'Local path' field, highlighted by a red box and a red arrow from annotation 4. At the bottom of the dialog is a large button with a checkmark icon and the text 'Create repository', highlighted by a red box and a red arrow from annotation 5. Red arrows from annotations 1 and 3 point to the '+' icon in the top left and the 'Name' input field, respectively.

Git, Github 익히기

Github Desktop - 기초사용법

커밋하기

Change 탭 (변경사항 확인)

변경된 파일들 목록을
확인하고 최종 Commit
할 파일들 선택

변경사항 요약 제목 및
자세한 설명 문구 삽입

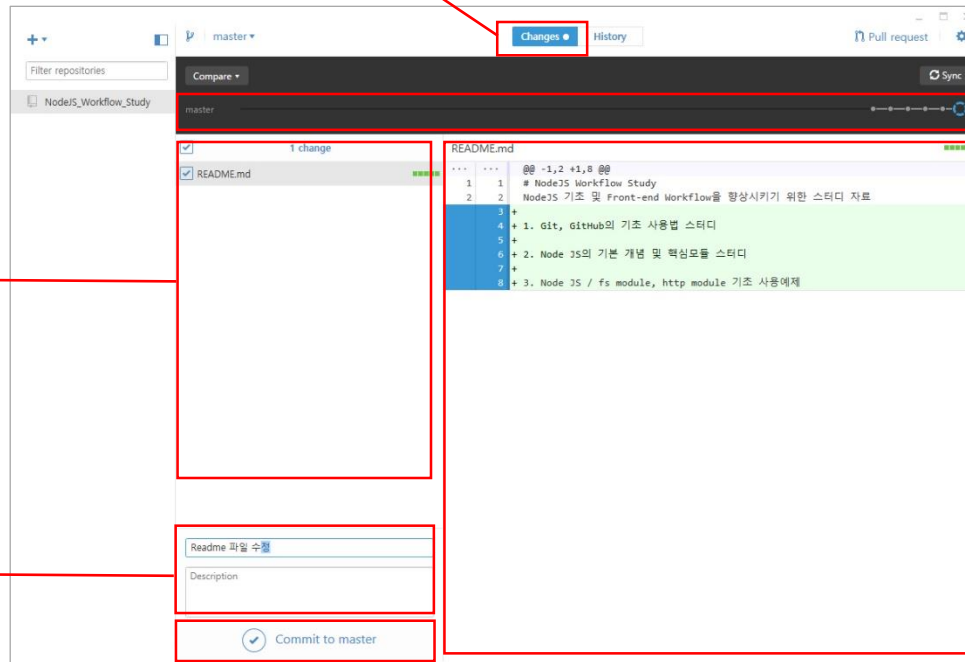
최종확인 및 COMMIT

해당 저장소의 파일들이
Commit 되었던 이력을 볼수
있다.

변경된 내용 보기

- CSS,HTML, Javascript,
C,C#,Java 등의 소스파일안에
서 수정,삭제,추가된 내용을
확인할 수 있다.

-PPT,DOC, AI 등 특정 프로그
램 포맷을 사용하는 바이너리
파일의 내용과 변경사항은 볼
수 없다.



Git, Github 익히기

Github Desktop - 기초사용법

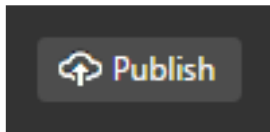
히스토리 및 변경된 파일 내용 확인

History 탭 (변경이력 확인)

Commit 내역 리스트

해당 저장소의 파일들이 Commit되었던 이력을 볼수 있다.

Revert 버튼을 클릭하면 이전 상태로 되돌린다.



로컬 저장소에서 작업이 완료되어 원격저장소(Github site)로 올릴 준비가 되면 Publish 버튼을 클릭한다.



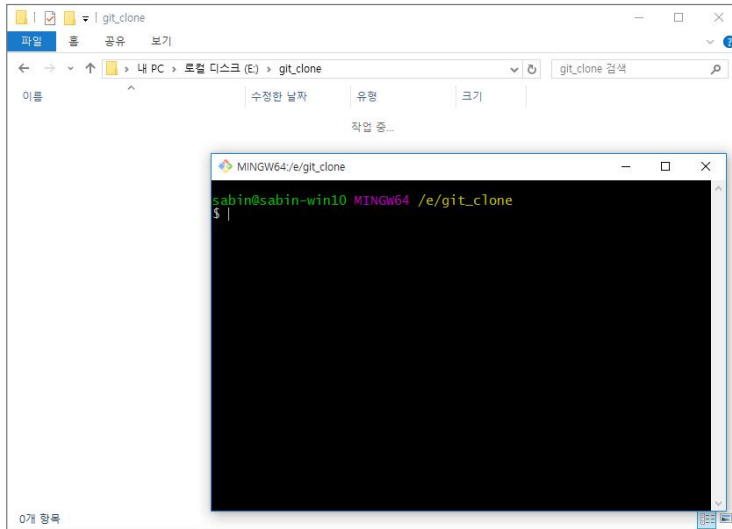
원격저장소가 생성되면 Publish 버튼은 Sync 버튼으로 변경된다. 이후 로컬 저장소에서 파일을 변경하여 커밋되었으면 Sync 버튼을 클릭하여 원격저장소와 동기화 시킬 수 있다.

Git

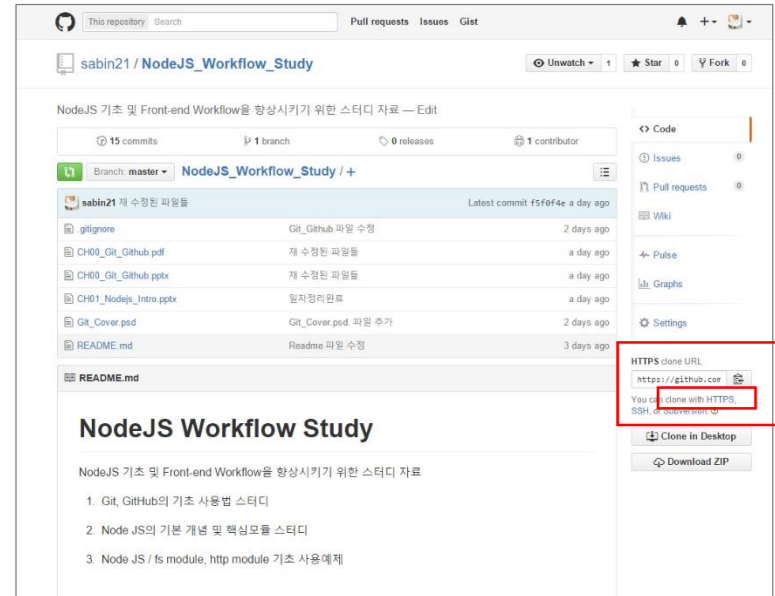
저장소 클론하기

Git, Github 익히기

Git - 저장소 클론하기



1. Git 저장소를 클론하고 싶은 위치에서 git bash창을 연다.

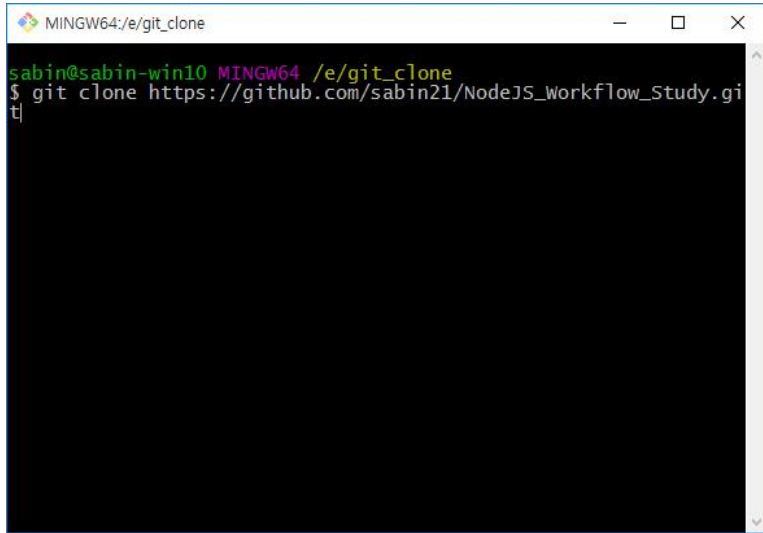


2. 클론하고자 하는 github 프로젝트를 찾아가서 우측 하단의 URL 복사창의 밑에 있는 HTTPS를 클릭하여 나타나는 HTTPS 주소를 카피한다.

Git 이 원격저장소와 통신하는 방법 중 대표적인 두가지는 SSH와 HTTPS 방식이다. SSH와 HTTPS방식을 주로 사용하는 이유는 보안성 때문이다. SSH 방식은 인증과 데이터 암호화를 모두 지원하고 읽기와 쓰기가 가능하기 때문에 Git 기본 통신 프로토콜로 사용된다. HTTPS 방식은 설정하기가 상대적으로 쉽지만 속도가 느린편이고 쓰기 기능을 설정하기가 복잡하다.

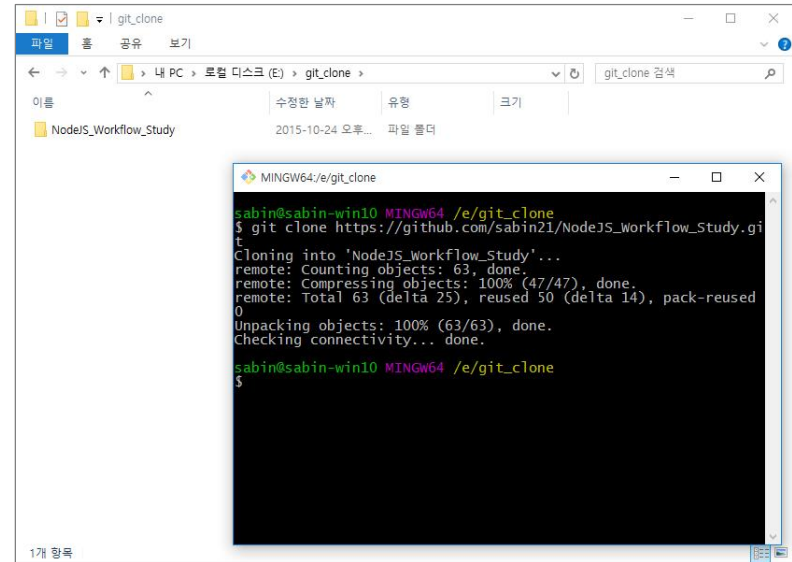
Git, Github 익히기

Git - 저장소 클론하기



```
MINGW64:/e/git_clone
sabin@sabin-win10 MINGW64 /e/git_clone
$ git clone https://github.com/sabin21/NodeJS_workflow_Study.git
```

3. git clone 를 입력하고 복사한 https주소를 오른쪽 클릭으로 붙여넣기 한다.

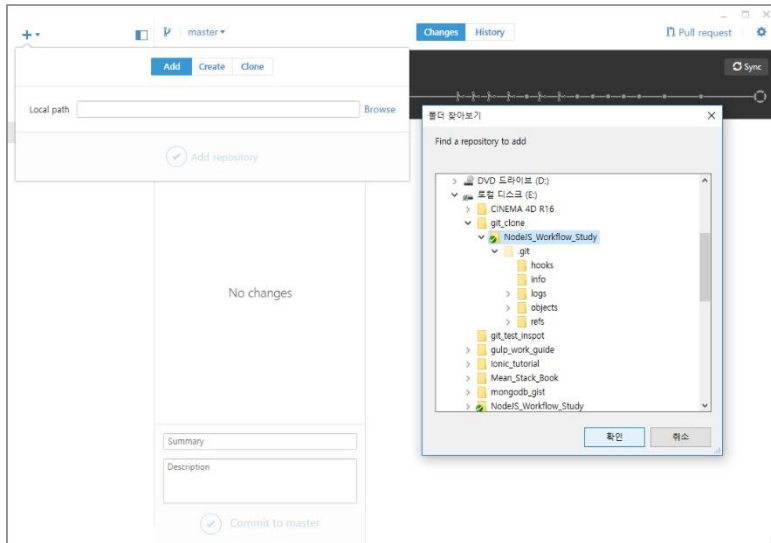


```
MINGW64:/e/git_clone
sabin@sabin-win10 MINGW64 /e/git_clone
$ git clone https://github.com/sabin21/NodeJS_workflow_Study.git
Cloning into 'NodeJS_workflow_Study'...
remote: Counting objects: 63, done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 63 (delta 25), reused 50 (delta 14), pack-reused 0
Unpacking objects: 100% (63/63), done.
Checking connectivity... done.
sabin@sabin-win10 MINGW64 /e/git_clone
$
```

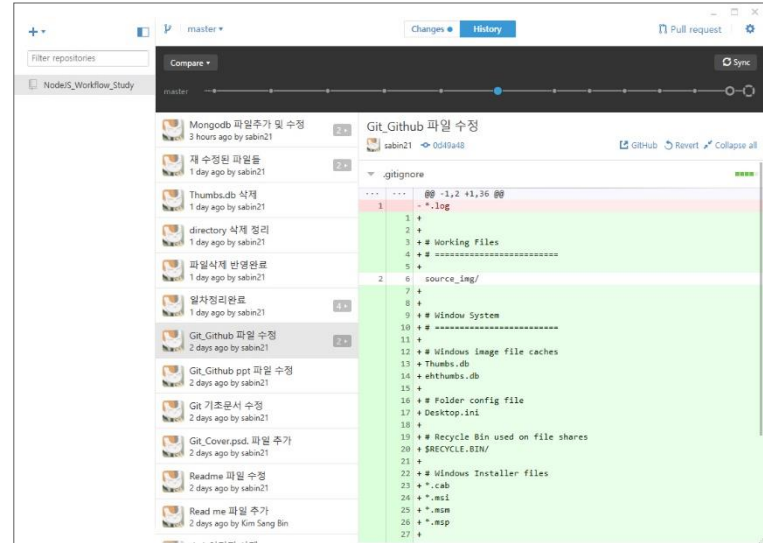
4. 명령을 실행하면 클론과정을 진행하는 메시지가 나오고 클론이 완료된다.

Git, Github 익히기

Git - 저장소 클론하기



5. github desktop 에서 저장소 Add 를 선택하고 클론한 위치를 확인해 준다.



6. 클론된 저장소를 확인해 보면 단순히 파일만 클론된 것이 아니라 커밋했던 **모든 Commit History** 까지 클론됐다는 것을 확인할 수 있다.

- 이 부분이 SVN과 GIT의 결정적인 차이이다. Git을 '분산버전관리 시스템'이라는 이유는 위의 예처럼 원격저장소나 다른 저장소를 클론하면 파일 뿐만 아니라 모든 커밋 히스토리 까지 복사하여 동일한 저장소를 로컬에 생성시키기 때문이다.

이런 특성 때문에 원격저장소의 하드디스크에 문제가 생기더라도 이를 클론한 컴퓨터가 있으면 다시 저장소를 복원할 수 있다.

Git

기초사용법

Git, Github 익히기

Git - 기초사용법

커밋 후 수정 내용 보기

git diff

커밋한 이후로 어떤 부분이 달라졌는지 확인하기 위해 diff 명령을 사용한다. add 명령어로 Staged 상태로 변경한 후에는 사용할 수 없다.

git diff --staged

diff 명령으로는 이미 staged 상태로 변경된 파일의 수정사항을 확인 할 수 없지만 --staged 옵션을 같이 사용하면 수정사항을 확인할 수 있다.

```
sabin@sabin-win10 MINGW64 /e/projects/inspot_gittest (master)
$ git diff
diff --git a/views/index.ejs b/views/index.ejs
index 039a2fc..213ea2d 100644
--- a/views/index.ejs
+++ b/views/index.ejs
@@ -27,7 +27,7 @@
     <div class="container-3">
       <div class="wrap-inner">
         <h1>Git Test Page</h1>
-        <p>Git의 기초 사용법을 테스트하기 위한 샘플 화면 </p>
+        <p>Git와 Github 기초 사용법을 익혀보자 !</p>
       </div>
     </div>
@@ -52,6 +52,6 @@
   });
   </script>
+
</body>
</html>
warning: LF will be replaced by CRLF in views/index.ejs.
The file will have its original line endings in your working directory.
```

수정된 파일 이름

수정되기 이전 내용

수정된 이후 내용

Git, Github 익히기

Git - 기초사용법

커밋 히스토리 확인

git log

지금까지 저장소에 저장해온 커밋 히스토리를 조회하기 위해 사용한다. 아무런 옵션을 주지 않고 사용하면 가장 최근에 커밋한 내용부터 차례로 보여준다.

git log -3

가장 최근에 커밋한 결과 3개만 표시한다.

git log --stat -2

가장 최근에 커밋한 결과 2개와 수정된 파일의 통계정보를 표시한다..

git log --graph

브랜치와 머지 정보까지 아스키 그래프 형태로 표시한다.

```
sabin@sabin-win10 MINGW64 /e/projects/inspot_gittest (master)
$ git log
commit 81a68eb4932fa0f8c80fa05818722d3485719708
Author: sabin21 <bloodyrosa@naver.com>
Date: Sun Oct 25 19:53:00 2015 +0900

    First commit

commit 5d560ac36a6687c410cb3400d417f1b471b95484
Author: sabin21 <bloodyrosa@naver.com>
Date: Sat Oct 24 16:45:28 2015 +0900

    gitignore 테스트

commit 0c303fbe3236d6f6ef44dc4c6c64472c486cec09
Author: sabin21 <bloodyrosa@naver.com>
Date: Sat Oct 24 16:44:40 2015 +0900

    README 파일 추가
```


Git, Github 익히기

Git - 기초사용법

Commit을 덮어쓰기

```
git commit --amend
```

종종 완료한 커밋을 수정해야 할 때가 있다. 너무 일찍 커밋했거나 어떤 파일을 빼먹었을 때 그리고 커밋 메시지를 잘못 적었을 때 한다. 다시 커밋하고 싶으면 --amend 옵션을 사용한다. 이 옵션을 실행하면 기본 텍스트 에디터가 실행되고 최근 커밋에서 입력한 메시지 등을 수정할 수 있도록 된다.

이 옵션을 사용하고 나서 커밋을 하면 나중에 한 커밋이 이전 커밋을 덮어쓴다.

Commit 취소하기

```
git reset HEAD^
```

--amend는 마지막 커밋한 내용을 최신 커밋으로 덮어쓰지만 git reset HEAD^ 은 커밋한 것 자체를 취소하기만 한다. 그리고 수정되었던 파일은 Staged 단계가 아닌 워킹디렉토리 단계로 돌아간다.

Modified 파일 되돌리기

```
git checkout -- README.txt
```

한번 커밋했던 파일을 수정했는데 이 수정한 파일을 커밋했던 상태로 되돌리고자 할때 사용한다.

이 명령을 쓸 때는 조심해야 한다. 커밋된 이후에 작업한 내용이 완전히 사라지기 때문에 꼭 되돌려야하는지 한번 더 생각해 보아야 한다.

파일상태를 Unstage로 변경하기

```
git reset HEAD README.txt
```

아직 커밋할 생각이 없는 파일을 실수로 add 명령으로 Stage 상태로 만들었을 때, 이 를 다시 Unstage 상태로 변경하기 위해 사용한다.

Git, Github 익히기

Git - 기초사용법

파일 삭제하기

```
git rm README.txt
```

Git에서 파일을 제거하려면 git rm 명령으로 Tracked 상태의 파일을 삭제한 후에 커밋해야 한다. 이 명령은 워킹디렉토리에 있는 파일도 삭제하기 때문에 실제로 지워진다.

만약 지우려는 파일을 이미 add 하여 Staging Area에 올렸었다면 삭제되지 않기 때문에 -f 옵션을 주어 강제로 삭제해야 한다.

```
git rm -r doc/
```

doc 디렉토리와 그 아래 있는 파일들을 삭제한다.

```
git rm --cached README.txt
```

만약 파일을 실제 하드디스크에서는 지우지 않고 Git에서 추적하지 않게만 하고 싶다면 --cached 옵션을 사용하여 명령을 실행한다.

파일 이름 변경하기

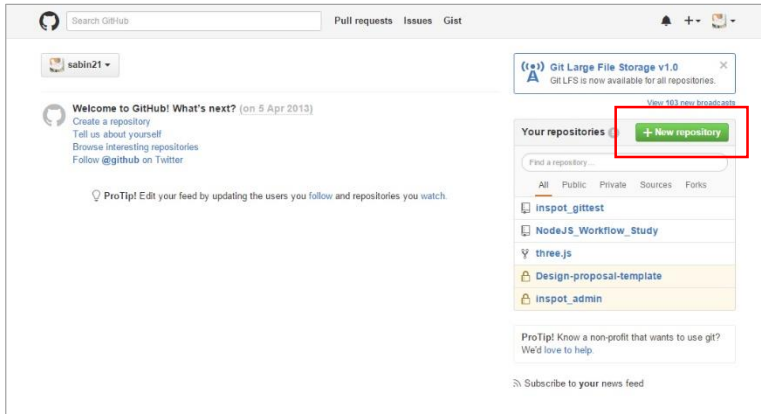
```
git mv README.txt Readme.md
```

파일이름을 변경하고자 하면 git mv 다음에 '현재파일명' '바뀐파일명' 을 기재하고 실행한다. 만약 Git상에서 이름을 바꾼 게 아니라 다른곳에서 파일 이름을 바꾸었으면 git rm README.txt 하고 git add Readme.md 를 실행한다.

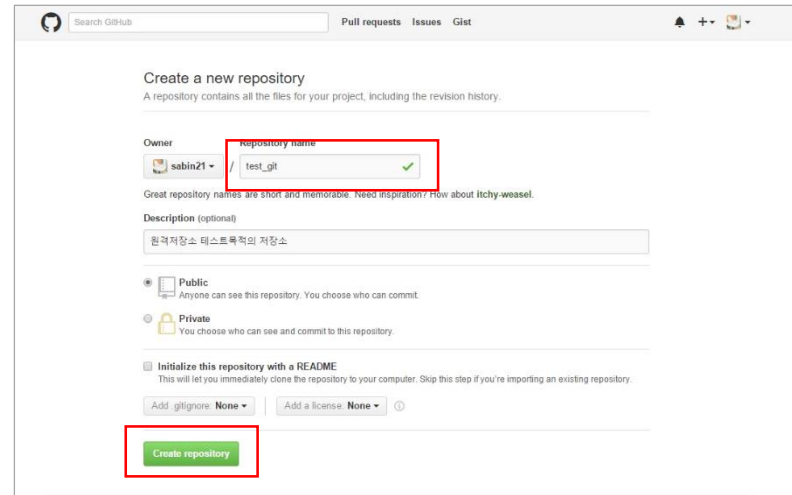
Git, Github 익히기

Git - 기초사용법

원격저장소로 Push하기



1. 먼저 원격저장소로 사용할 github 사이트로 이동하여 New repository 를 클릭한다.

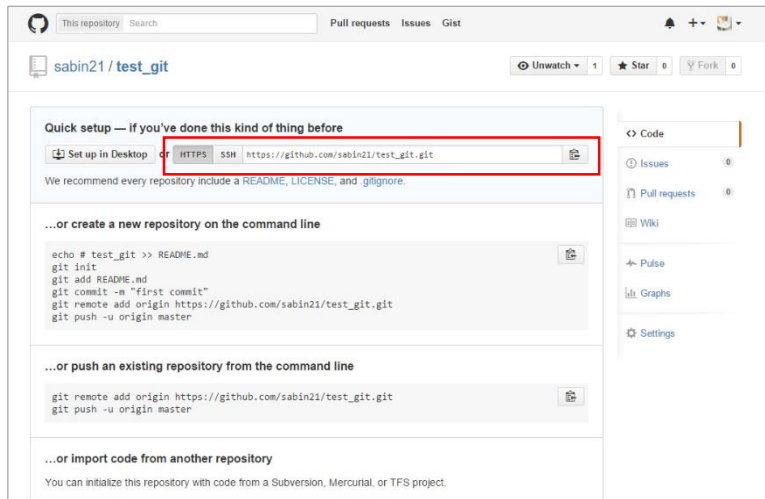


2. 적당한 저장소의 이름을 입력하고 저장소의 설명을 입력한다. 기타 필요한 옵션을 확인하고 원격저장소 생성을 완료한다.

Git, Github 익히기

Git - 기초사용법

원격저장소로 Push하기



3. 생성된 원격저장소의 HTTPS 주소를 확인하고 클립보드로 카피한다.

```
sabin@sabin-win10 MINGW64 /e/projects/remote_test (master)
$ git remote add test_git https://github.com/sabin21/test_git.git
```

git remote add [저장소명] [저장소 Https 주소]

4. git bash 창에 git remote add 를 입력하고 띄어쓰기 다음에 github에 만든 저장소명을 입력하고 저장소의 https 주소를 붙여넣기 한다.

```
sabin@sabin-win10 MINGW64 /e/projects/remote_test (master)
$ git remote -v
test_git      https://github.com/sabin21/test_git.git (fetch)
test_git      https://github.com/sabin21/test_git.git (push)
```

git remote -v

5. git remote -v를 입력하여 원격저장소의 정보를 다시 확인한다. fetch와 push를 위한 주소 정보가 출력된다.

Git, Github 익히기

Git - 기초사용법

원격저장소로 Push하기

```
sabin@sabin-win10 MINGW64 /e/projects/remote_test (master)
$ git push test_git master
Username for 'https://github.com': bloodyrosa@naver.com
Password for 'https://bloodyrosa@naver.com@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 227 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/sabin21/test_git.git
 * [new branch]      master -> master
```

git push [저장소명] [브랜치명]

6. git push 다음에 push할 저장소명을 입력하고 다음에 브랜치명을 입력한다.
기본으로 생성되는 브랜치명은 master이다.

Github의 저장소로 연결할 주소로 HTTPS 주소를 사용하였기 때문에 Push를 실행하면 사용자 이메일정보와 비밀번호를 입력해야 만 한다.

Git, Github 익히기

Git - 기초사용법

Git 저장소에서 불필요한 파일,디렉토리를 제외시키기

```
# Working Files
# =====

source_img/

# Window System
# =====

# Windows image file caches
Thumbs.db
ehthumbs.db

# Folder config file
Desktop.ini

# Recycle Bin used on file shares
$RECYCLE.BIN/

# Windows Installer files
*.cab
*.msi
*.msm
*.msp

# Windows shortcuts
*.lnk

# OSX
# =====

.DS_Store
.AppleDouble
.LSOverride
```

.gitignore

.gitignore 파일을 생성하여 Git 버전관리 대상에서 제외시키고
자 하는 파일이나 디렉토리의 목록을 기재한다.

윈도우나 OSX 등 OS가 자동생성하는 파일들은 일차적으로 제외시켜야 한다.
저장소안에서 버전관리가 필요없는 파일이나 디렉토리 명을 작성하여 버전관리 대상
에서 제외시킬 수 있다.

은 주석처리

directory name/ 디렉토리를 통채로 제외시키려면 디렉토리명 뒤에 "/"을 붙인다.

*.doc doc확장자를 가진 모든 파일들을 제외시킨다.

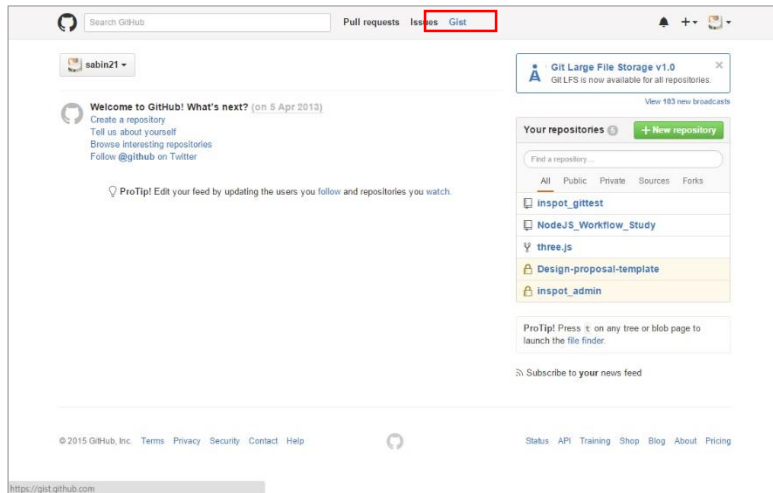
!sabin.doc 이 파일은 제외시키지 않는다.

doc/**/*.txt doc디렉토리 아래에 있는 모든 txt파일을 무시한다.

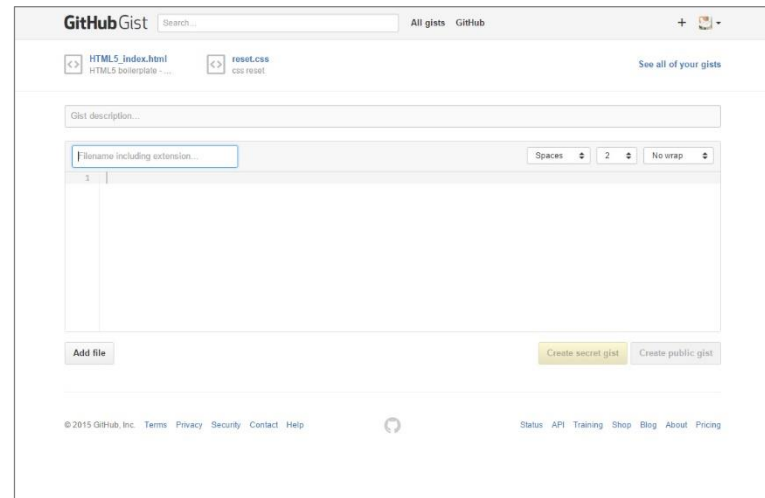
Git, Github 익히기

Github - Gist 사용하기

Github 사이트에서 Gist를 사용하면 자주 반복해서 쓰는 코드의 조각인 스니펫 (Snippets) 또한 버전관리하여 보관할 수 있으며 실무작업 시 편리하게 이용할 수 있다.
- Gist 기능은 Sublime Text , Eclipse, Visual Studio 들과 연동하여 사용 할 수 있다.



1. github 상단 메뉴에서 Gist를 클릭한다.



2. Gist 페이지 처음에 새로운 Snippet을 편집할 수 있는 화면이 보인다. 제일 위 입력필드에는 Snippet의 간단한 설명을 입력하고 아래 필드에서 파일명과 확장자를 입력한다. 하단의 코드입력 필드에 자주 쓰는 코드를 입력하면 된다. 하단의 Add File을 클릭하면 코드 입력필드가 또 하나 생성된다.

Git

개인학습 과제

Git 실무활용을 위한 필수 학습내용

Git 저장소 만들기

git init, git clone

수정하고 저장소에 저장하기

git add, git status, git commit, git rm

커밋 히스토리 조회

git log

되돌리기

git commit --amend, git checkout

리모트 저장소

git remote, git fetch, git pull, git push

브랜치 이해

git branch, git checkout

브랜치와 Merge 기초

git checkout, git merge, git branch -d

브랜치 관리

git branch (-v, --merged, -d)

Remote 브랜치

git remote, git push

Git 서버 설치 및 설정

Web Project 실무에서 Git을 원활하게 사용하기 위해서는 브랜치를 생성, 관리하고 Merge하는 부분을 필수적으로 습득해야 한다.

PSD, AI 파일을 Git으로 버전관리 할 경우 예측



Git, Github 익히기

Bash Shell 기초 명령어

<code>mkdir css js images fonts</code>	디렉토리 생성 / 여러이름을 띄어쓰기로 연달아 쓰면 많은 디렉토리를 한번에 생성할 수 있다.	<code>touch style.css app.js README.md</code>	비어 있는 3개의 파일을 한꺼번에 생성한다.
<code>ls</code>	디렉토리안에 있는 파일과 디렉토리 정보를 출력한다.	<code>ls -l</code>	사용권한, 소유자,그룹, 크기,날짜 상세정보를 출력한다.
<code>mv style.css css/style.css</code> <code>mv app.js js/app.js</code>	style.css 파일을 css 디렉토리 안으로 이동한다. app.js 파일을 js 디렉토리안으로 이동한다.	<code>ls css/</code> <code>ls js/</code>	css, js 디렉토리안의 내용을 출력한다.
<code>mv images img</code> <code>mv index.html home.html</code>	images 디렉토리명을 img로 변경한다. index.html 파일명을 home.html로 변경한다.	<code>vim README.md</code>	vim 에디터로 README.md파일을 연다. i키를 누르면 문서편집 모드로 변환된다. esc키를 누르고 :wq를 입력하면 저장하고 vim 에디터가 종료된다.
<code>cat README.md</code>	README.md 파일의 내용을 확인한다.		
<code>cp css/style.css base.css</code>	css디렉토리안의 style.css 파일을 base.css 이름으로 현재 위치에 복사한다.	<code>cp -R css/ sass/</code>	css 디렉토리를 sass 이름의 디렉토리로 복사한다.
<code>rm base.css</code>	base.css 파일을 삭제한다.	<code>rm -rf js</code>	js 디렉토리를 무조건(안에 있는 파일까지 모두) 삭제한다.

Git, Github 익히기

05- Git 기본개념 및 이해

Git 기본

<https://mylko72.gitbooks.io/git/content/index.html>

Git 공식 Book

<https://git-scm.com/book/ko/v2>