

국민대 빅데이터경영MBA 파이썬 프로그래밍을 통한 데이터 분석

2016년 1학기 강의
정광윤 anappleofmyeyes@gmail.com



국민대학교
KOOKMIN UNIVERSITY

경영대학원
GRADUATE SCHOOL OF BUSINESS ADMINISTRATION

Previously on Python Programming

이번주에 학습했던 것을 잠시 보도록 하겠습니다.

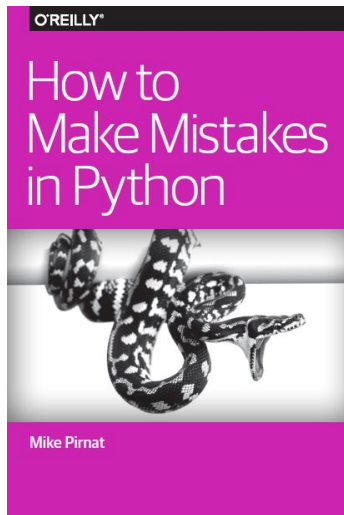
1. Setting Development Environment
2. Python Primer

Kookmin University
Graduate School of Business Administration
MBA in Big Data Analytics

Setting Development Environment

1. Python 설치
2. Sublime Text 3 설치
3. Programming Font 설치
4. Package Manager - pip 설치
5. 가상 환경 - virtualenv, virtualenvwrapper 설치
6. iPython, jupyter notebook 설치

Setting Development Environment



System을 지저분하게 하지 말자!
- Re: 그래서 **virtualenv**를 설치했습니다.

Polluting the System Python

Once you start developing or working with multiple projects on that computer, you're going to eventually have **conflicts over package dependencies**. ... The solution for this is to use so-called virtual environments. (virtualenv, pyvenv, etc.)

기본 **Shell**을 사용하지 말자!
- Re: 그래서 **iPython**을 설치했습니다.

Using the Default REPL(Read Evaluate Print Loop)

When I started with Python, one of the first features I fell in love with was the interactive shell, or REPL. Nearly 16 years later, I still reach for that same Python shell when I want to try something out...**which is a shame**, because there are far better alternatives that I should be using instead. ... The most notable of these are IPython and the browser-based Jupyter Notebook.

IPython Alternatives

- Sublime Text 3에 설치한 SublimeREPL
- <http://repl.it/> 등도 있음

Python Primer

1. Indentation
2. Code Block은 Colon(:)으로 시작
3. Comment는 #, ““ ... ””, ““““ ... ””””
4. 0부터 시작
5. Package(Directory) > Module(File)
6. 문서 내 한글 사용하기 위해 맨 앞에 # coding: UTF-8
7. Python의 모든 것은 객체다!

01

Python 101

Python의 기본적인 문법을 보도록 하겠습니다. 여기서는 Jupyter Notebook으로 보도록 하겠습니다.

Kookmin University
Graduate School of Business Administration
MBA in Big Data Analytics

02

Crash Course in Git

Kookmin University
Graduate School of Business Administration
MBA in Big Data Analytics

현업에서는 혼자서 개발하는 경우보다 함께 개발하는 경우가 더 많습니다. 함께 개발할 때 소스 이력을 위해 사용하는 **Git**이라는 도구를 알아보고, **Github**를 이용해 소통하는 방법을 배워보도록 하겠습니다.

버전 관리 시스템

Version Control System(VCS)

Version Control Systems



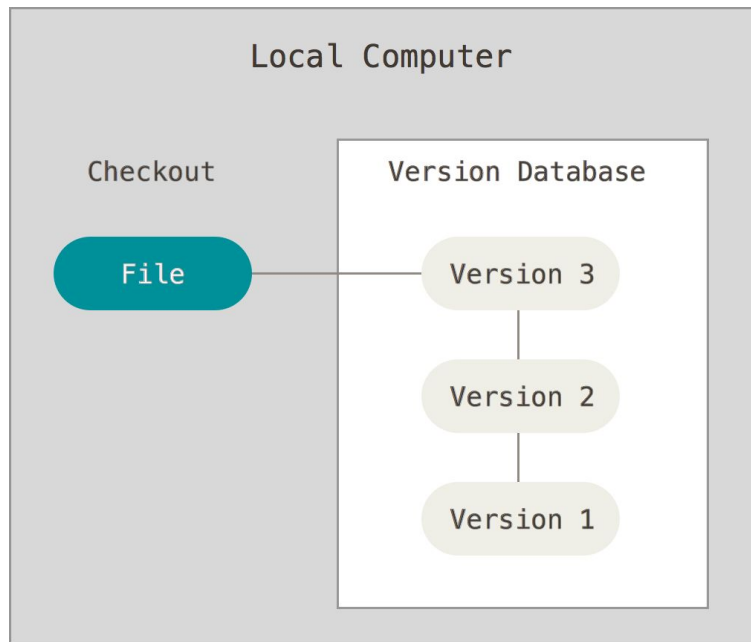
<u>Version</u>	<u>Control</u>	<u>Systems</u>	<u>or</u>
<u>Revision</u>	<u>Control</u>	<u>Systems</u>	<u>or</u>
<u>Source</u>	<u>Control</u>	<u>Management</u>	

대개 특정 시스템 or 서비스를 만들 때는 다수가 모여 개발을 합니다. 이 때 소스를 같이 사용하기 때문에 소스 버전, 누가 무엇을 변경했는지 등을 관리할 필요가 있습니다.

이렇게 VCS를 사용하면










- 각 파일을 이전 상태로 되돌릴 수 있습니다.
- 시간에 따라 수정 내용을 비교해 볼 수 있습니다.
- 누가 문제를 일으켰는지도 추적할 수 있습니다.
- 누가 언제 만들어낸 이슈인지 알 수 있습니다.
- 파일을 잃어버리거나 잘 못 고쳤을 때 쉽게 복구할 수 있습니다.

Types of Version Control Systems

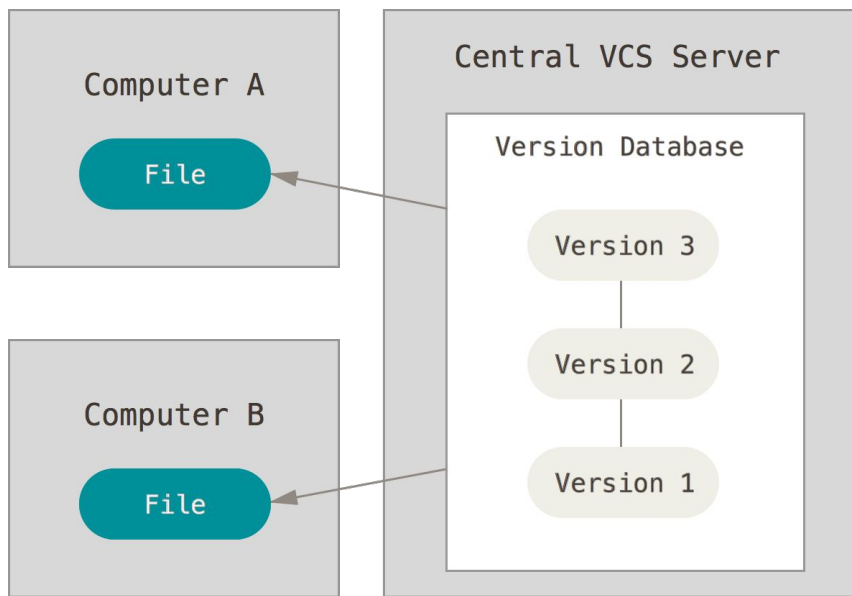


Local Version Control Systems

- 사람들이 흔히 사용하는 방법입니다.
- 아래처럼 문서를 관리하는 경우가 많습니다. 그러나 우리가 개발하게 될 소스는 어떨까요?

Name	
	120525_문서_업데이트.txt
	120604_문서.txt
	120605_문서_수정판.txt
	120605_문서_수정판2.txt
	120605_문서_최신 복사.txt
	120605_문서_최신.txt
	120605_문서.txt
	1200602_문서.txt
	문서_회의용.txt

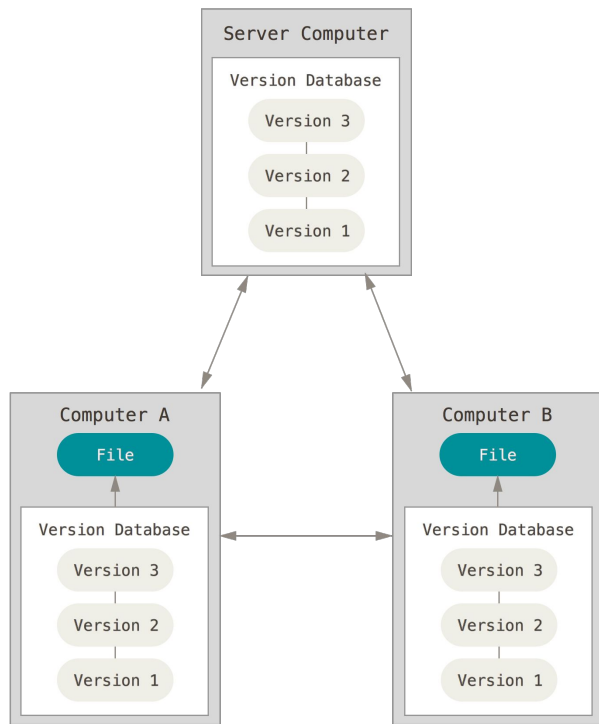
Types of Version Control Systems



Centralized Version Control Systems

- 중앙집중식 버전 관리 시스템
- 파일을 관리하는 서버가 별도로 있고 클라이언트가 중앙 서버에서 파일을 받아서 사용(Checkout)하는 방식입니다.
- CVS, Subversion, Perforce 등
- 장점
 1. 누가 무엇을 했는지 알 수 있음
 2. Local 보다 중앙 서버에서 관리하는 것이 쉬움
- 단점
 1. 중앙 서버에 문제가 발생하면 모든 사용자가 개발을 진행하지 못함

Types of Version Control Systems



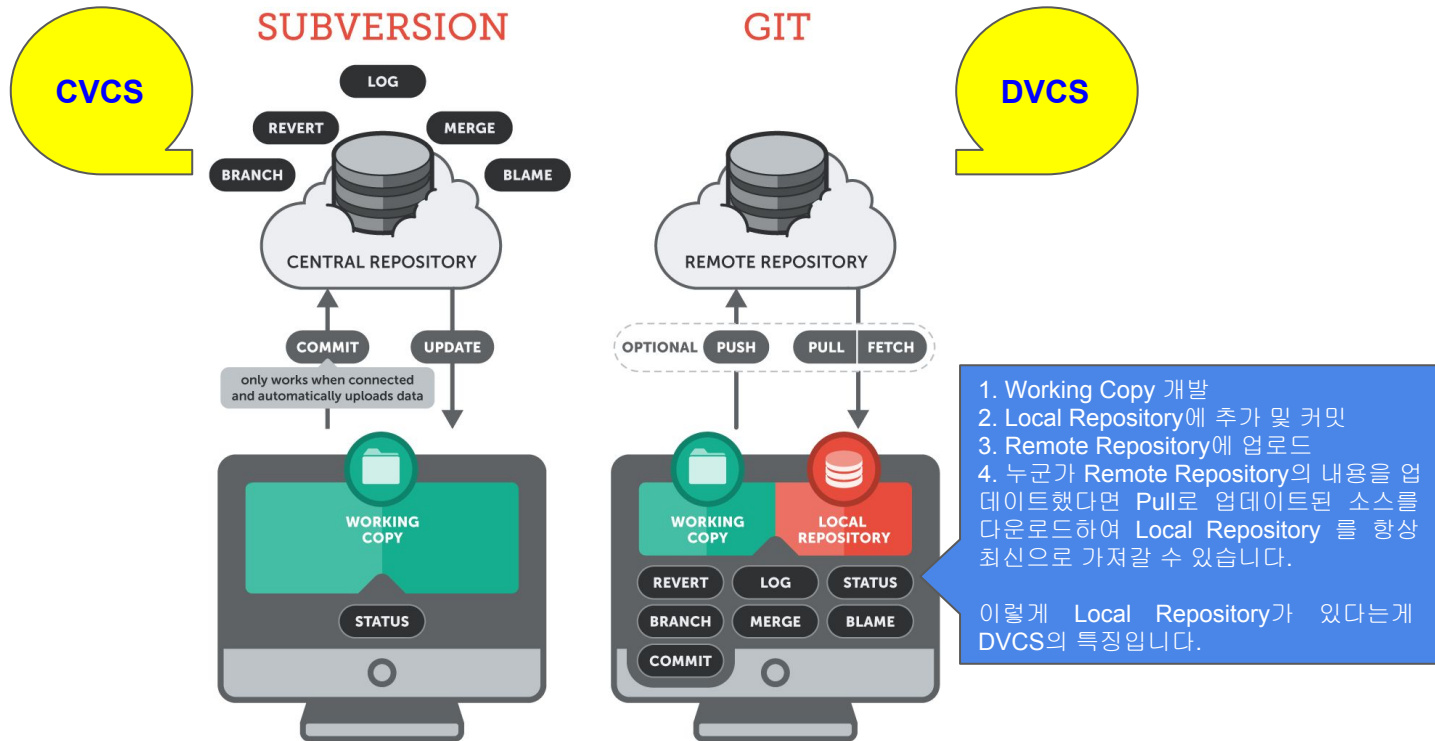
Distributed Version Control Systems

- 분산 버전 관리 시스템
- CVS의 단점을 보완하기 위해 만들어졌습니다. 각기 Local System에 Server의 모든 파일을 복제해두어 분산시키는 방식입니다.
- Git, Mercurial, Bazaar, Darcs 등
- SVN에 비해 학습하기가 어렵다는 단점 외에 버전 관리를 위한 여러기능을 제공하며 속도가 빠르고, 특히 여럿이 분산 시스템 환경에서 개발할 때 용이합니다.

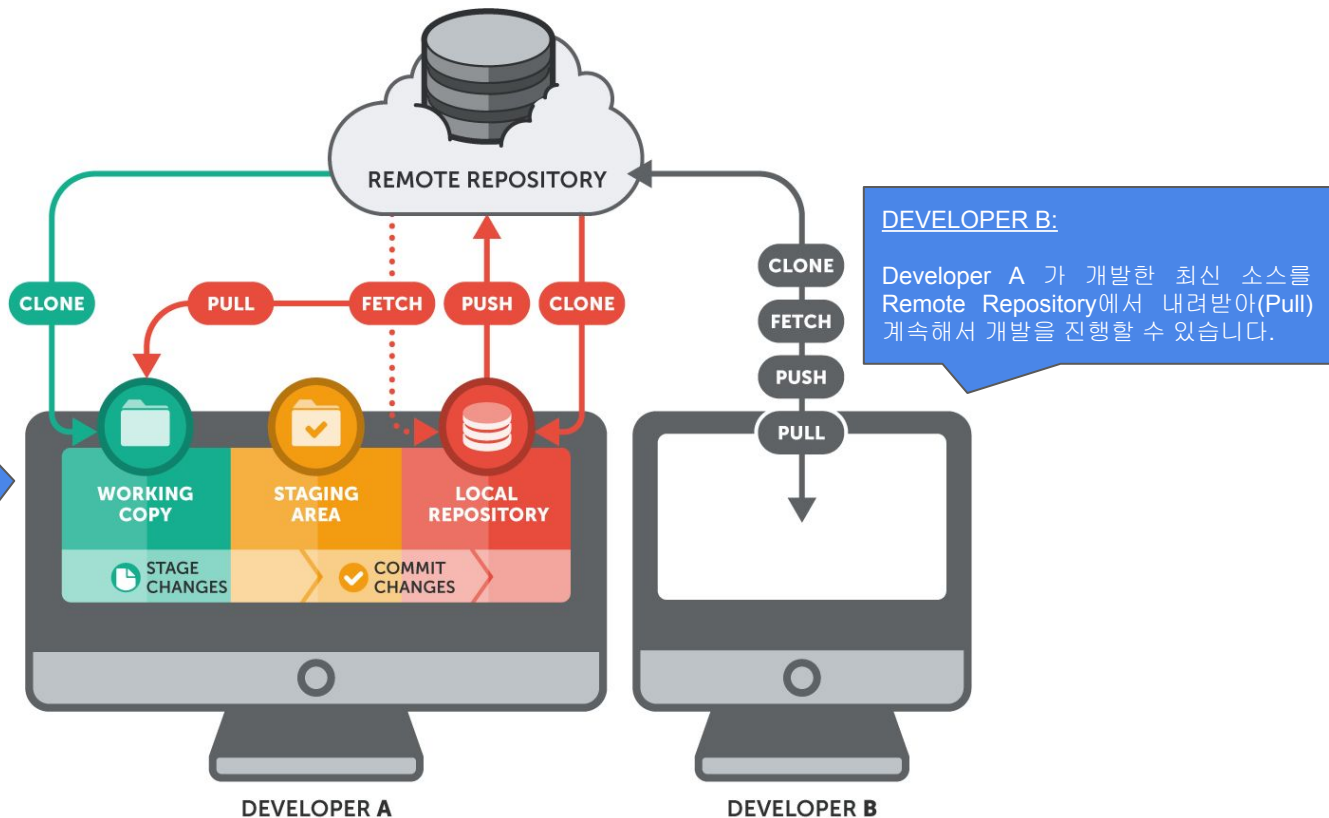
GIT

Global Information Tracker

Distributed VCS



Version Control Flow Using Git



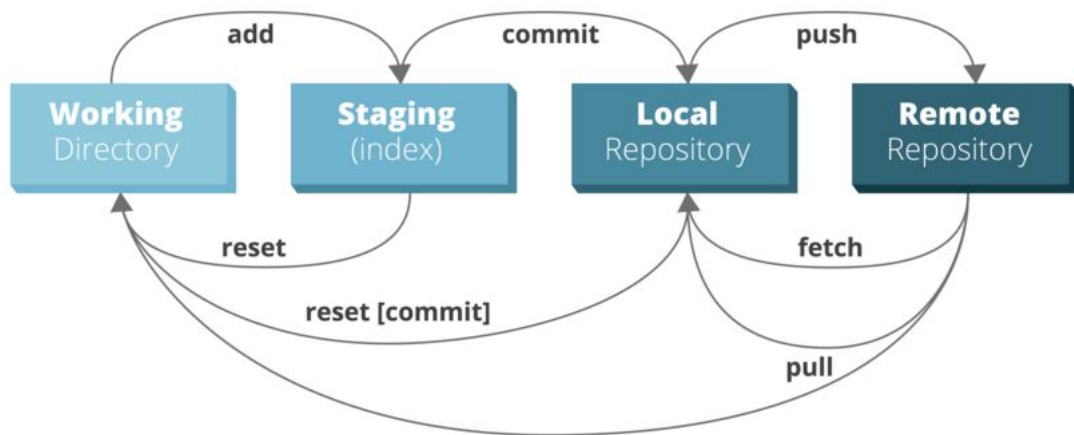
DEVELOPER A:

버전 관리의 작업 흐름을 먼저 이해해야 합니다. 개발 중인 소스(Working Copy)를 Staging Area에 추가(Add)하고 이것을 Local Repository에 커밋(Commit)을 합니다. 그러면 본인의 컴퓨터에서 소스 버전 관리를 할 수 있습니다. Developer B와 같이 개발하기 위해서는 Remote Repository에 Push를 해야합니다.

DEVELOPER B:

Developer A가 개발한 최신 소스를 Remote Repository에서 내려받아(Pull) 계속해서 개발을 진행할 수 있습니다.

Git Command



위 흐름을 잘 기억해두어야 합니다.
흐름에 맞는 Git Command로 버전 관리를 할 수 있습니다.
혹시 위 Command가 기억이 안난다면?

GIT 작업 흐름과 명령어

도움말: git '명령어' --help

Git의 글로벌 설정은 \$HOME/.gitconfig에 저장 (git config --help)

생성

새 저장소 생성하기
cd ~/projects/myproject
git init
git add .

기존 저장소 Clone하기
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone you@host.org/project.git

보기

워킹 디렉터리의 파일 상태 보기
git status

파일의 변경사항 보기
git diff

\$ID1과 \$ID2 사이의 변경사항 보기
git diff \$Id1 \$Id2

커밋 히스토리의 보기
git log

특정 파일의 커밋 히스토리별 변경사항 보기
git log -p \$file \$dir/ectory/

특정 파일을 누가 언제 고쳤는지 보기
git blame \$file

\$ID 커밋 보기
git show id

\$ID 버전의 파일 보기
git show \$id:\$file

로컬 브랜치들 보기
git branch
(* 표시는 현재 브랜치를 나타냄)

범례

\$id - 커밋 ID, 브랜치 이름, 태그 이름을 나타냄
\$file - 파일 이름
\$branch - 브랜치 이름

개념

Git 기본

master : 기본 브랜치
origin : 기본 리모트 저장소
HEAD : 현재 브랜치
HEAD^ : HEAD의 부모
HEAD~4 : HEAD의 부모의 부모의 부모

되돌림

마지막 커밋 시점으로 되돌리기
git reset --hard
▲ Hard Reset은 되돌릴 수 있음

마지막 커밋 내용을 되돌리고 커밋하기
git revert HEAD
새로운 커밋 생성

특정 커밋 내용을 되돌리고 커밋하기
git revert \$id
새로운 커밋 생성

마지막 커밋 수정하기
git commit -a --amend (앞쪽 커밋에서 수정하고 싶을 때)

\$id 시점의 파일을 꺼내기
git checkout \$id \$file

브랜치

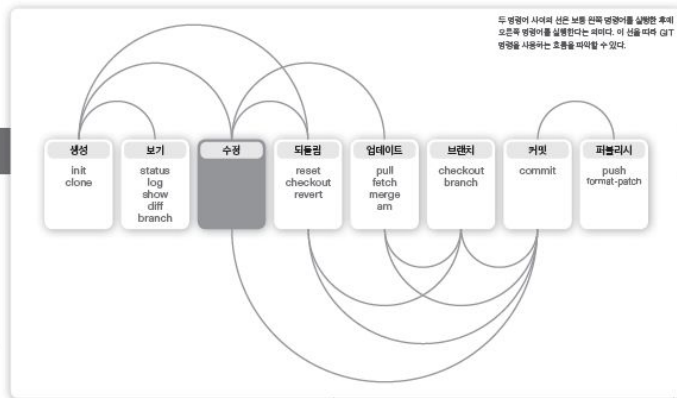
브랜치를 Checkout하기
git checkout \$id

\$branch1을 \$branch2에 Merge하기
git checkout \$branch2
git merge \$branch1

새 브랜치 만들기
git branch \$branch

\$other와 같은 커밋을 가리키는 브랜치를 새로 만들고 바로 Checkout하기
git checkout -b \$new_branch \$other

\$branch를 삭제하기
git branch -d \$branch



업데이트

origin에서 최신 데이터를 가져오기
git fetch
(Merge하지는 않음)

origin에서 최신 데이터를 가져와 Merge하기
git pull
(Fetch하고 Merge까지 함)

누군가 보낸 패치를 Merge하기
git am -3 patch-mbox
(충돌이 발생하면 해결 후 git am --resolved)

퍼블리시

현재 모든 수정사항을 커밋하기
git commit -a

다른 개발자에게 보낼 패치를 작성하기
git format-patch origin

origin으로 업데이트를 Push하기
git push

버전이나 마일스톤을 생성하기
git tag v1.0

유용한 명령어

문제가 발생한 커밋 이전 탐색하기

git bisect start (이전 탐색 시작)
git bisect good \$id (\$id를 문제 없는 상태로 표시)
git bisect bad \$id (\$id를 문제 있는 상태로 표시)

git bisect bad/good (현재 상태가 문제가 있는지 없는지 분별)
git bisect visualize (log를 실행하여 확인함)
git bisect reset (시작했던 상태로 Checkout 함)

저장소의 무결성 검사 및 저장소 청소하기
git fsck
git gc --prune

워킹 디렉터리에서 'foo()'라는 문자열 검색하기
git grep "foo()"

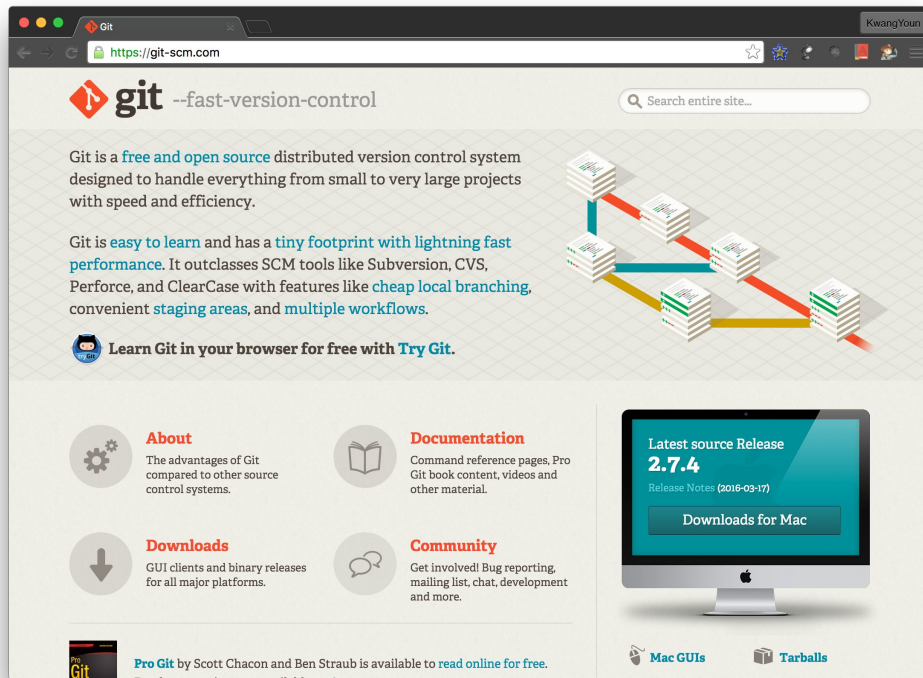
Merge 충돌 해결

Merge 충돌 내용 보기
git diff (충돌 내용 전체 보기)
git diff --base \$file (Merge Base를 기준으로)
git diff --ours \$file (본 브랜치를 기준으로)
git diff --theirs \$file (Merge를 브랜치를 기준으로)

충돌이 생기는 패치 버리기
git reset --hard
git rebase --skip

충돌 해결 후 Merge 진행하기
git add \$conflicting_file (충돌 해결한 파일)
git rebase --continue

Git 설치



<http://www.git-scm.com> 에서 본인의 운영체제에 맞게 다운로드 합니다.
모두 최초 세팅된 대로 Next 버튼을 클릭하여 설치합니다.

Github

가장 대표적인 Git Hosting Service인 Github을 통해 실습을 하겠습니다.

Remote Repository(Git Hosting Services)



Features	Github	Bitbucket	Sourceforge	Gitlab	Kiln	Codeplane	Code Plex	Beanstalk
Pricing*	Free	Free	Free	Free	\$18/mo	\$9/mo	Free	\$15/mo
Private Repo	Paid	Unlimited, Free	Yes	Unlimited, Free	Paid	Unlimited, Paid	Unlimited, Upto 30 Days	10
Public Repo	Unlimited, Free	Unlimited, Free	Yes	Unlimited, Free	Paid	Unlimited, Paid	Unlimited	10
Storage Limit	1GB per repo	2GB	None	None	None	2GB	None	3GB
Users	Unlimited	5 & Unlimited if public	Collaboration not possible	Unlimited	5	Unlimited	Unlimited	5
VCS	Git, SVN	Git, Hg	Git, SVN, Hg	Git	Git, Hg	Git	Git, SVN, TFS, Hg	Git, SVN
Graphs	Yes	No	No	Yes	No	No	No	No
Web Hosting	Static sites. Page generator	Static sites	Dynamic Sites, CMS	Static	Yes	No	No	No
Code Review	Yes	Yes	Yes	Yes	No	No	No	Yes
Wiki	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Bug Tracking	Yes (Login Required)	Yes	Yes	Yes	Yes	Yes	Yes	No
Discussion Forum	No	No	Yes	No	No	No	Yes	No

<http://beebom.com/2015/03/github-alternatives>

실습

1. <http://github.com>에 회원 가입합니다.
2. 내 컴퓨터에서 Git 설치한 최초에 다음을 실행합니다.
 - a. `git config --global user.name "이름"`
 - b. `git config --global user.email "이메일"`
3. 수업 자료가 있는 Repository를 내 컴퓨터로 복사해옵니다: `git clone https://github.com/initialkommit/kookmin.git`
4. 잘 복사해왔는지 상태를 확인해봅니다: `git status`
5. 숨겨져 있는 디렉토리(.git) 확인해봅니다.
 - a. `ls`
 - b. `ls -a`
 - c. `ls -la`
6. 내용을 수정하고 자료를 올리는 실습은 다음 강의 때 해보겠습니다.

Git Client

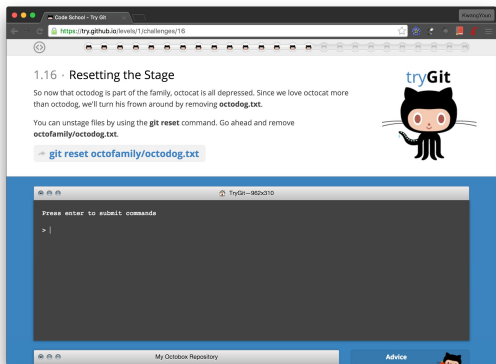


GitHub Desktop

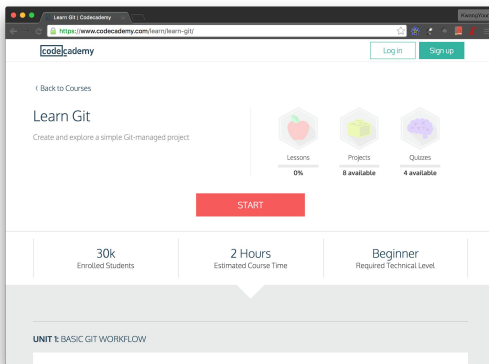


검은 바탕 화면의 Git Bash가 익숙치 않은 분들은 GUI 방식의 위 Git Client로 버전 관리하는 것도 좋을 것 같습니다.

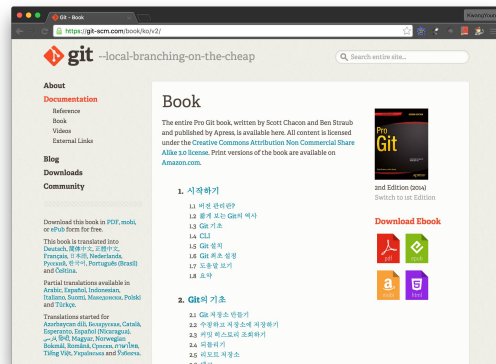
If you want to learn more



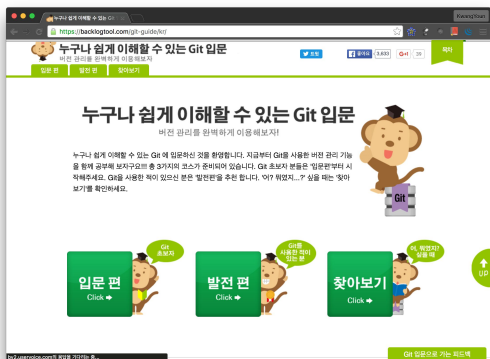
www.codeschool.com/courses/try-git/



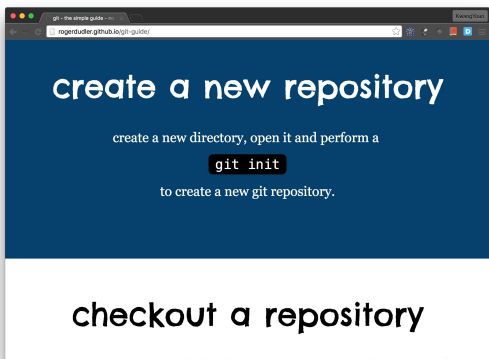
try.github.io/levels/1/challenges/16/



git-scm.com/book/ko/v2/



backlogtool.com/git-guide/kr/



rogerdudler.github.io/git-guide/

위에서 학습한 Git 외에 알아두어야 할 것이 생각보다 많습니다. Git을 통해 팀원들과 어떻게 하면 효과적으로 프로젝트를 진행할지 전략을 세울 수 있습니다. 이 학습 사이트를 통해 Git Branch 등을 더 연습해보시기 바랍니다.

국민대
파이썬
데이터