

Grunt & Git 활용

작성자: 조성민

소속팀 : UIT 개발실

작성년월일: 2014. 8. 8

대외비

목차

2.1 버전 관리 시스템

2.1.1 버전 관리란?

2.1.2 중앙집중식 버전 관리 시스템(CVCS)

2.1.3 분산 버전 관리 시스템(DVCS)

2.2 Git 기초

2.2.1 Git

2.2.2 설치

2.2.3 설정

2.3 저장소 만들기

2.3.1 git init

2.3.2 저장소 구조

--- 2.1 버전 관리 시스템

VCS

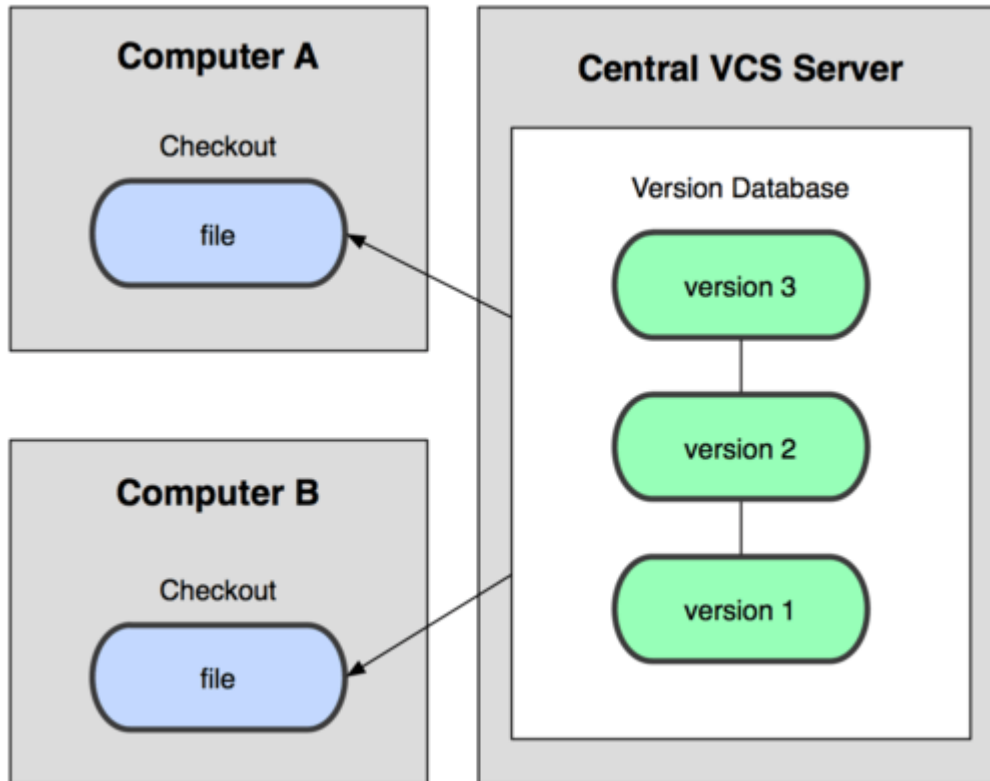
버전 관리 시스템은 파일 변화를 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 꺼내올 수 있는 시스템을 말한다.

그래픽 디자이너나 웹 디자이너도 버전 관리 시스템(VCS - Version Control System)을 사용할 수 있으며, VCS로 이미지나 레이아웃의 버전(변경 이력 혹은 수정 내용)을 관리할 수도 있다.

VCS 장점

1. 각 파일을 이전 상태로 되돌릴 수 있다.
2. 프로젝트를 통째로 이전 상태로 되돌릴 수 있다,
3. 시간에 따라 수정 내용을 비교해 볼 수 있다,
4. 누가 문제를 일으켰는지도 추적할 수 있다,
5. 누가 언제 만들어낸 이슈인지도 알 수 있다.
6. 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구할 수 있다.

Central VCS



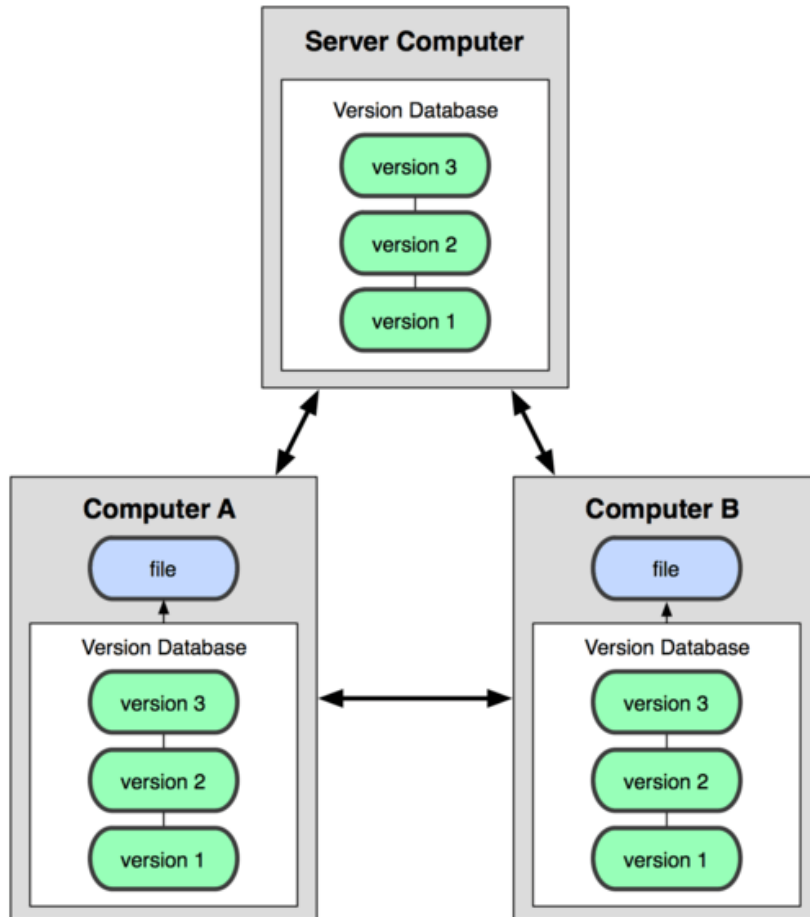
장점

1. 프로젝트에 참여한 사람이면 누가 무엇을 하고 있는지 알 수 있다.
2. 관리자는 누가 무엇을 할 수 있는지 꼼꼼하게 관리할 수 있다.
3. 모든 클라이언트의 로컬 데이터베이스를 관리하는 것보다 VCS 하나를 관리하기가 훨씬 쉽다.

단점

1. 중앙 서버에 문제가 발생하면 서버가 복구될 동안 아무도 다른 사람과 협업 할 수 없고 자료를 받거나 백업할 방법도 없다.
2. 중앙 서버의 하드 디스크에 문제가 발생하면 모든 자료를 잃을 수 있다.

Distributed VCS



Git, Mercurial, Bazaar, Darcs

장점

1. 서버에 문제가 생기면 이 복제물로 다시 작업을 시작할 수 있다.
2. 클라이언트 중에서 아무거나 골라도 서버를 복원할 수 있다.
3. 리모트 저장소가 많을 수도 있다.
4. 사람들은 동시에 다양한 그룹과 다양한 방법으로 협업할 수 있다.
5. 계층 모델 같은 중앙집중식 시스템으로는 할 수 없는 몇 가지 워크플로우를 사용할 수 있다.

2.2 Git

Git

만든 사람

리누스 토발즈



만든 배경

리눅스 커널 소스코드를 관리하기 위해 사용 하던 BitKeeper가 상용화되어 버리는 바람에 리눅스 개발 커뮤니티(특히 리누스 토발즈)가 자체 도구를 만드는 계기가 되었으며, 2005년 만들어지게 되었다.

Git 목표(장점)

1. 빠른 속도
2. 단순한 구조
3. 비선형적인 개발(수천 개의 동시 다발적인 브랜치)
4. 완벽한 분산
5. 리눅스 커널 같은 대형 프로젝트에서도 유용할 것(속도나 데이터 크기 면에서)

Git 단점

1. 기존 VCS 보다 어렵다.

Git 설치

공식 홈페이지에서 git을 다운로드 받아서 설치

<http://git-scm.com>

지원 OS

1. Windows
2. Mac
3. Linux



The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--everything-is-local". A search bar is in the top right. The main text describes Git as a "free and open source distributed version control system" and "easy to learn" with "lightning fast performance". It lists features like "cheap local branching" and "multiple workflows". A "Try Git" button is present. Below this, there are sections for "About", "Documentation", "Downloads", and "Community". The "Downloads" section lists "Windows GUIs", "Mac Build", "Tarballs", and "Source Code". The "About" section mentions "Pro Git" by Scott Chacon. The bottom of the page features logos of companies and projects using Git, including Google, Facebook, Microsoft, Twitter, LinkedIn, Netflix, PostgreSQL, Android, Rails, Qt, GNOME, Eclipse, and others. A footer note states "This open sourced site is hosted on GitHub." and "Git is a member of Software Freedom Conservancy".

Mac

Xcode 등이 설치되어 있다면 기본적으로 설치되어 있음.

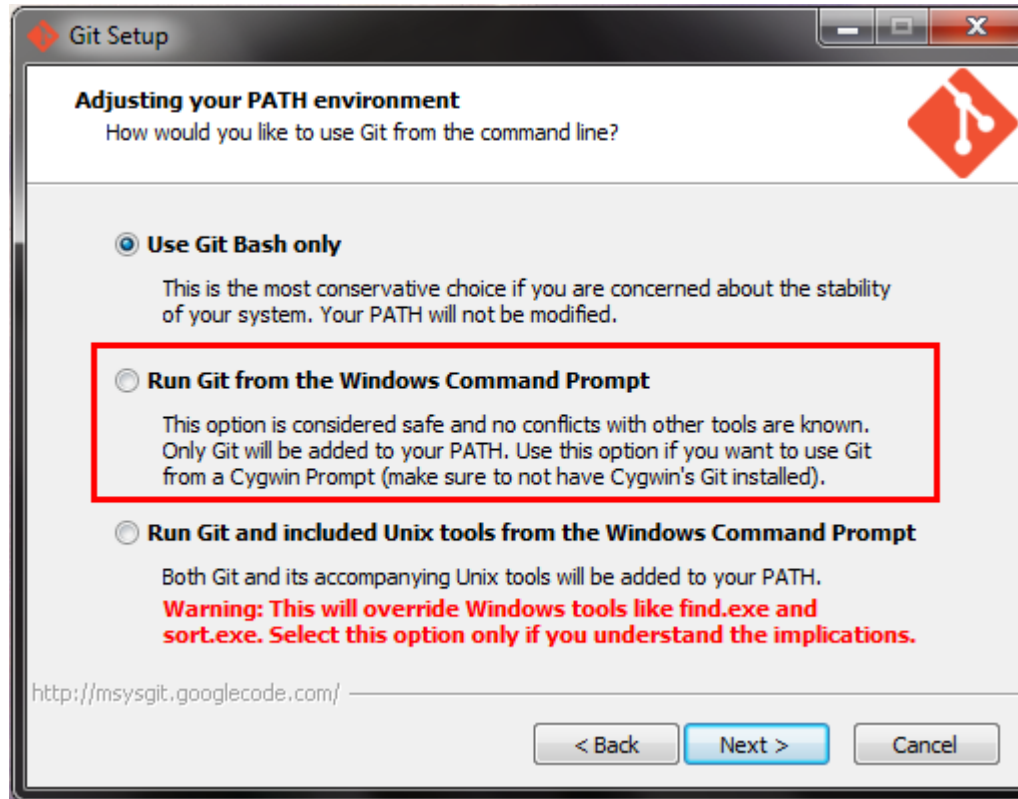
만약 설치되어 있지 않다면 공식 홈페이지에서 다운받아 설치하거나 homebrew 등을 통해 손쉽게 설치 및 업그레이드 가능.

※ homebrew 참고사항

1. homebrew 공식 사이트 : <http://brew.sh/>
2. 설치 방법 : `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/homebrew/go/install)"` 터미널에 입력
3. `brew install git`
4. `git --version` 을 입력할때 “git version 1.7.10.2 (Apple Git-33)” 과 같이 나온다면 최신 git 버전이 정상적으로 수행 될 수 있도록 추가 작업을 해줘야 한다.

수정 방법 참고 : <http://apple.stackexchange.com/questions/93002/how-to-properly-update-git-on-mac>

Windows



Windows에서 Git을 설치할 때는 왼쪽에 보이는 옵션 설정 부분만 유의하자.

첫번째 옵션을 선택했을 때는 오로지 Git bash에서만 Git 명령어를 사용할 수 있으므로 두번째 옵션이나 세번째 옵션을 선택하자.

단, 세번째 옵션은 위험 요소가 있으니 잘 생각해 본 후 선택하자.

2.2.2 설치

설치 완료

The first screenshot shows a Git Bash terminal window titled 'MINGW32:/C/Users/chosungmin/Desktop'. It displays the Git version '1.9.4-preview20140611' and lists common commands like 'add', 'bisect', 'branch', etc.

The second screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. It displays the Git version '2013 Microsoft Corporation. All rights reserved.' and lists common commands like 'add', 'bisect', 'branch', etc.

뒤에 있는 창은 Git Bash 창이고,
앞쪽에 있는 창은 Command 창이다.

이전 장에서 설명한 설치 옵션 중 첫번째를
선택했을 경우 Command 창에서 Git
명령어가 작동되지 않는다.

※ Tip

Git Bash 는 Context Menu에 자동 등록
되니 작업 폴더 어디에서나 마우스 오
른쪽 버튼을 눌러 Git Bash 창을 띄울
수 있다.

Command 를 이용할 경우는 **Shift+마우스
오른쪽 버튼**을 눌러 “여기서 명령창 열
기” 메뉴를 이용하면 된다.

Console Emulator (Windows 사용자)

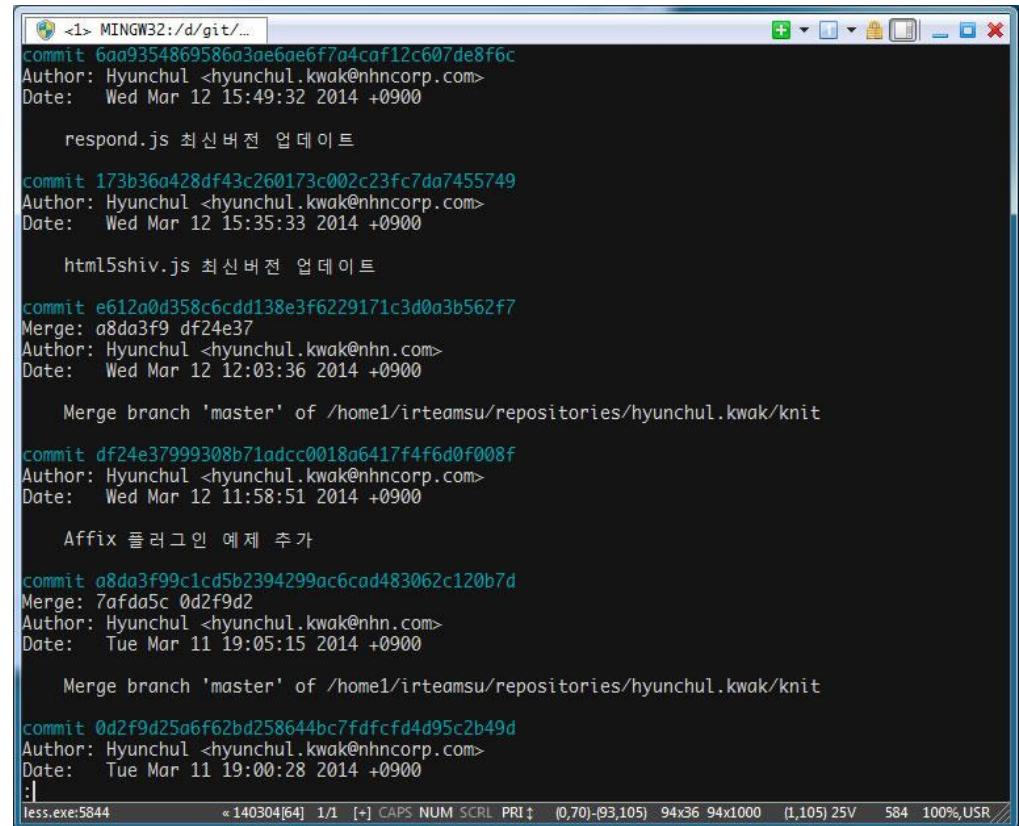
Git Bash나 Command 가 맘에 들지 않는다면 Console Emulator를 사용해보기 바란다.

다운로드

<http://www.fosshub.com/ConEmu.html>

장점

1. 복사/붙여넣기가 편하다. (Ctrl+C, V 가능)
2. 탭기능을 지원해준다.
3. 컬러설정을 지원
4. 원하는 폰트 설정 가능



```
<1> MINGW32:/d/git/...
commit 6aa9354869586a3ae6ae6f7a4caf12c607de8f6c
Author: Hyunchul <hyunchul.kwak@nhncorp.com>
Date:   Wed Mar 12 15:49:32 2014 +0900

    respond.js 최신 버전 업데이트

commit 173b36a428df43c260173c002c23fc7da7455749
Author: Hyunchul <hyunchul.kwak@nhncorp.com>
Date:   Wed Mar 12 15:35:33 2014 +0900

    html5shiv.js 최신 버전 업데이트

commit e612a0d358c6cdd138e3f6229171c3d0a3b562f7
Merge: a8da3f9 df24e37
Author: Hyunchul <hyunchul.kwak@nhn.com>
Date:   Wed Mar 12 12:03:36 2014 +0900

    Merge branch 'master' of /home1/irteamsu/repositories/hyunchul.kwak/knit

commit df24e37999308b71adcc0018a6417f4f6d0f008f
Author: Hyunchul <hyunchul.kwak@nhncorp.com>
Date:   Wed Mar 12 11:58:51 2014 +0900

    Affix 플러그인 예제 추가

commit a8da3f99c1cd5b2394299ac6cad483062c120b7d
Merge: 7afda5c 0d2f9d2
Author: Hyunchul <hyunchul.kwak@nhn.com>
Date:   Tue Mar 11 19:05:15 2014 +0900

    Merge branch 'master' of /home1/irteamsu/repositories/hyunchul.kwak/knit

commit 0d2f9d25a6f62bd258644bc7fdcf4d95c2b49d
Author: Hyunchul <hyunchul.kwak@nhncorp.com>
Date:   Tue Mar 11 19:00:28 2014 +0900

    :|
less.exe:5844 140304[64] 1/1 [~] CAPS NUM SCRL PR1↑ (0,70)-(93,105) 94x36 94x1000 (1,105) 25V 584 100%,USR
```

사용자 설정

Git을 처음 사용하는 사용자는 반드시 사용자 설정을 해줘야 한다.

```
$ git config --global user.name chosungmin  
$ git config --global user.email chosungmin@nhn.com
```

옵션 설명

--global 옵션을 사용할 경우 딱 한번만 설정하면 된다.

만약 프로젝트마다 다른 이름과 이메일 등을 설정하고 싶을 때는 --global 옵션을 빼고 명령어를 실행시켜 주면 된다.

설정 확인

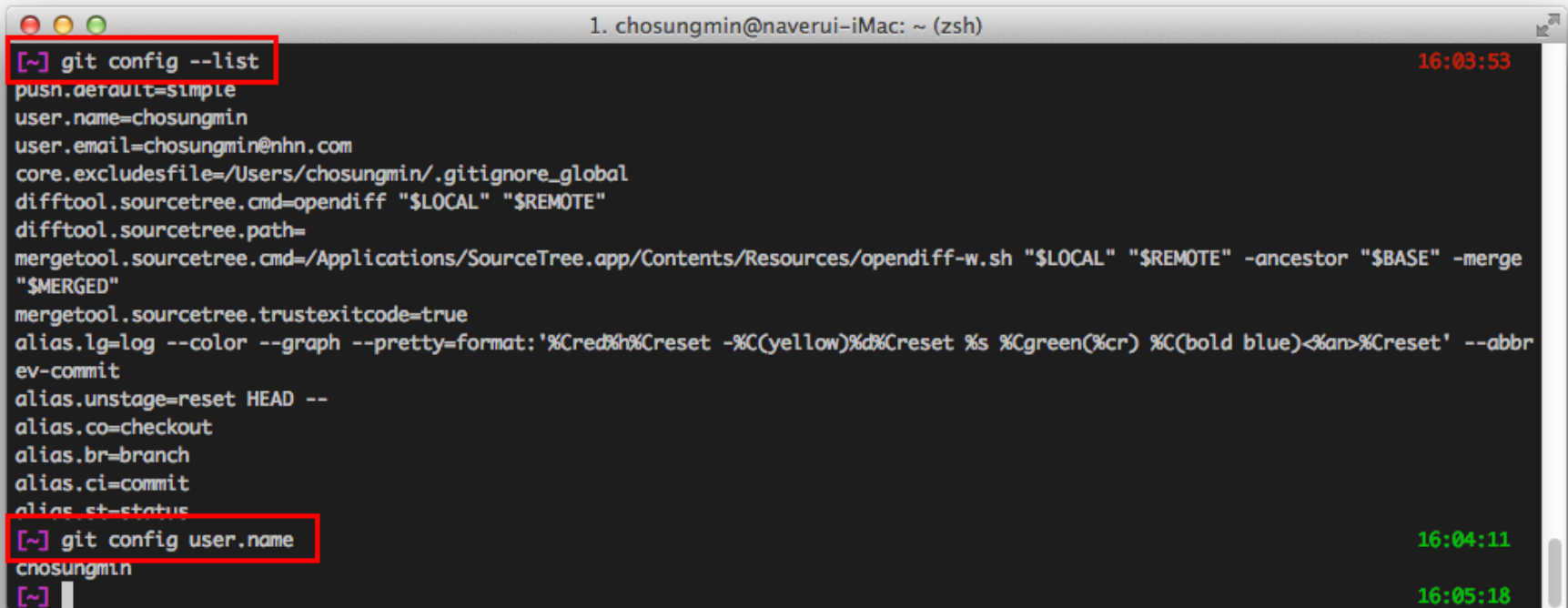
내가 설정한 Git 설정을 확인해 보자

```
$ git config --list  
또는 $ git config {key}
```

※참고

Git은 같은 키를 여러 파일(/etc/gitconfig와 ~/.gitconfig 같은)에서 읽기 때문에 같은 키가 여러개 있을 수도 있다.

이 경우에는 Git은 나중 값이 사용된다.



```
1. chosungmin@naverui-iMac: ~ (zsh)  
[~] git config --list  
push.default=simple  
user.name=chosungmin  
user.email=chosungmin@nhn.com  
core.excludesfile=/Users/chosungmin/.gitignore_global  
difftool.sourcetree.cmd=opendiff "$LOCAL" "$REMOTE"  
difftool.sourcetree.path=  
mergetool.sourcetree.cmd=/Applications/SourceTree.app/Contents/Resources/opendiff-w.sh "$LOCAL" "$REMOTE" -ancestor "$BASE" -merge "$MERGED"  
mergetool.sourcetree.trustexitcode=true  
alias.lg=log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<an>%Creset' --abbrev-commit  
alias.unstage=reset HEAD --  
alias.co=checkout  
alias.br=branch  
alias.ci=commit  
alias.st=status  
[~] git config user.name  
chosungmin  
[~]
```

사용자 설정 리스트

git config 옵션 리스트

<http://git-scm.com/book/ko/Git맞춤-Git-설정하기>

1. **core.editor** : Git은 편집기를 설정
2. **commit.template** : 커밋할 때 Git이 보여주는 커밋 메시지는 이 옵션에 설정한 템플릿 파일
3. **core.pager** : Git은 log나 diff같은 명령의 메시지를 출력할 때 페이지로 나누어 보여준다.
4. **user.signingkey**
5. **core.excludesfile** : .gitignore 파일을 저장소 밖에 두고 관리하고 싶으면 core.excludesfile에 해당 파일의 경로를 설정
6. **help.autocorrect** : Git 명령어를 잘못 입력 했을때 힌트를 주는 보여준다.
7. **color.ui** : true로 설정하면 Git이 알아서 컬러풀한 결과값을 보여준다.
8. **merge.tool** : merge, diff 툴 설정
9. **core.autocrlf**
10. **core.whitespace** : Git에는 공백 문자를 다루는 방법으로 네 가지가 미리 정의돼 있다.
11. **receive.fsckObjects** : Push할 때마다 각 개체가 SHA-1 체크섬에 맞는지, 잘못된 개체가 가리키고 있는지 검사 여부 설정
12. **receive.denyNonFastForwards** : 이미 Push한 커밋을 Rebase해서 다시 Push하지 못하게 설정
13. **receive.denyDeletes** : 브랜치를 삭제하는 Push를 거절

--- 2.3 저장소 만들기

git init

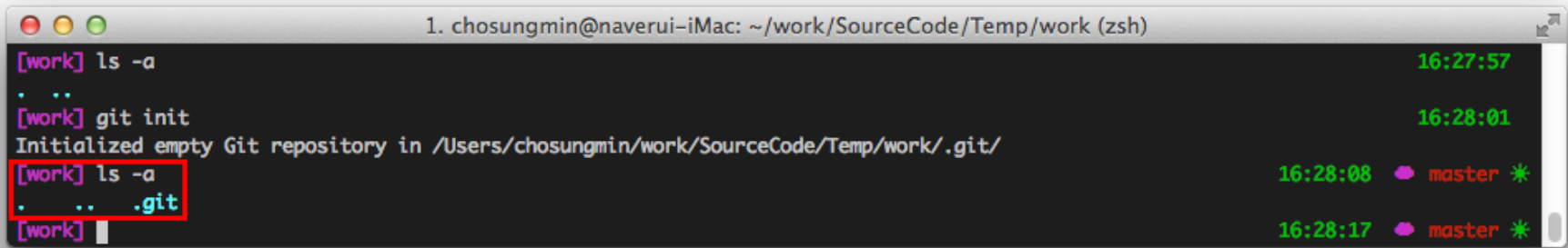
작업 디렉토리를 git으로 관리하기 위해서는 “git init” 명령어를 통해 git 저장소를 초기화 해줘야 한다.

```
$ git init  
또는  
$ git init [디렉토리명]
```

“git init” 명령어 입력시 폴더명을 넣지 않을 때는 현재 디렉토리가 git 저장소로 설정이 되며,
디렉토리를 넣을 때는 해당 디렉토리가 git 저장소로 설정된다.

※ 참고

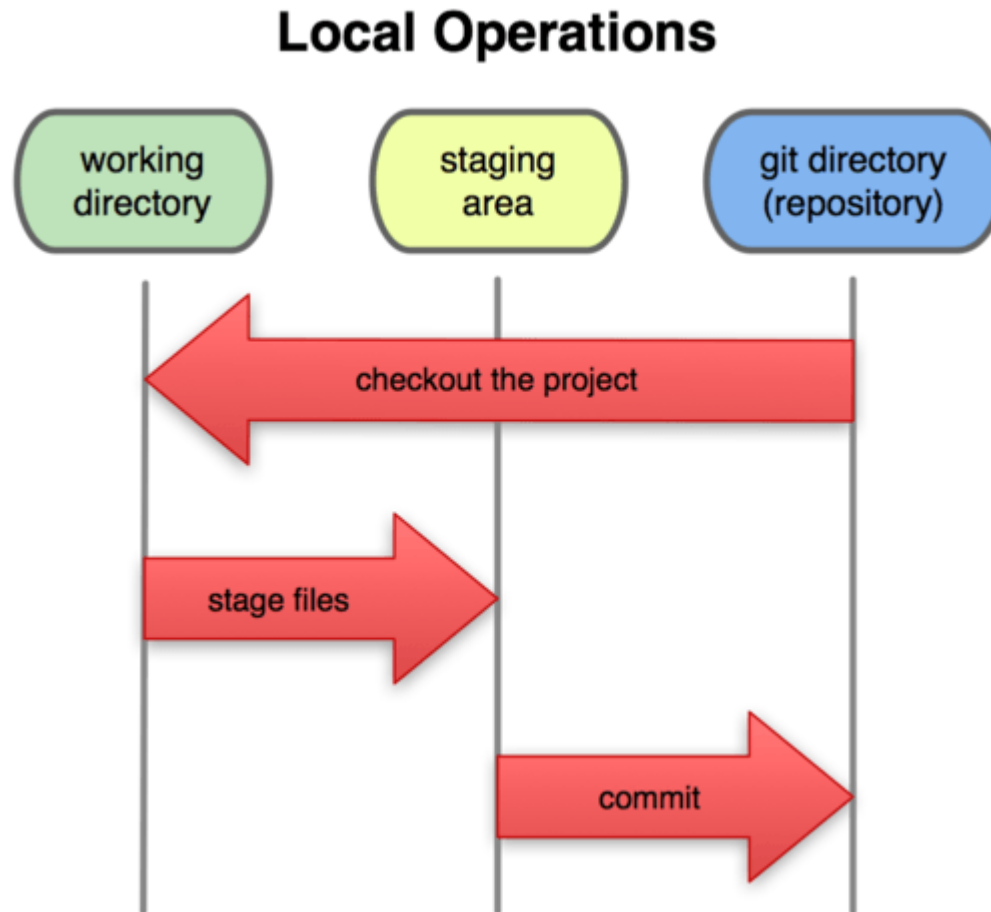
작업 디렉토리가 정상적으로 git 저장소로 설정되었다면 해당 디렉토리에 “.git” 디렉토리가 생성된걸 확인할 수 있다.



```
1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)  
[work] ls -a  
.  
..  
[work] git init  
Initialized empty Git repository in /Users/chosungmin/work/SourceCode/Temp/work/.git/  
[work] ls -a  
.  
.. .git  
[work]
```

The terminal screenshot shows the execution of the 'git init' command in a directory named 'work'. The output indicates that an empty Git repository has been initialized in the '.git' subdirectory. The final 'ls -a' command confirms the presence of the '.git' directory alongside the standard '.' and '..' entries. The terminal window title is '1. chosungmin@naverui-iMac: ~/work/SourceCode/Temp/work (zsh)'.

로컬 저장소 구조



로컬 저장소 구조 설명

1. Working Directory

프로젝트의 특정 버전을 Checkout한 것이다.

Repository는 지금 작업하는 디스크에 있고 그 디렉토리에 압축된 데이터베이스에서 파일을 가져와서 Working Directory를 만든다.

2. Staging Area

Staging Area는 Git 디렉토리에 있다.

단순한 파일이고 곧 커밋할 파일에 대한 정보를 저장한다.

종종 인덱스라고 불리기도 하지만, Staging Area라는 명칭이 표준이 되어가고 있다.

3. Repository

Repository는 Git이 프로젝트의 메타데이터와 객체 데이터베이스를 저장하는 곳을 말한다.

Repository가 Git의 핵심이다.

다른 컴퓨터에 있는 저장소를 Clone 할 때 Repository가 만들어진다.

Git으로 하는 일은 기본적으로 아래와 같다.

1. Working Directory에서 파일을 수정한다.
2. Staging Area에 파일을 Stage해서 커밋할 스냅샷을 만든다.
3. Staging Area에 있는 파일들을 커밋해서 Repository에 영구적인 스냅샷으로 저장한다.

Thank you.