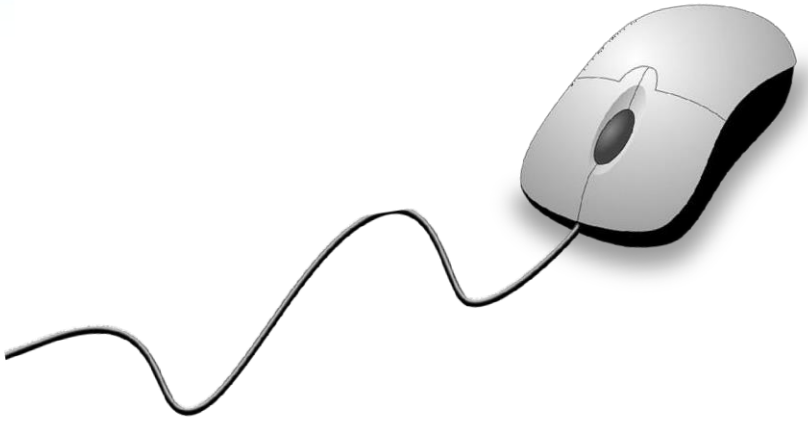
공개SW 솔루션 설치 & 활용 가이드

응용SW > 협업관리



git

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

소개	• 분산형 버전 컨트롤 시스템 (Distributed Version Control System) • Linux 커널 소스 코드의 버전 컨트롤 위해 리누스 토발즈에 의해 개발		
주요기능	• 텍스트 형태에 특화된, 그러나 거의 모든 파일의 버전 컨트롤 • 분산된 개발자들간의 협업 위한 구조와 프로토콜 제공 • 빠른 속도와 성능		
대분류	• 응용 SW	**소분류**	• 협업관리
라이선스 형태	• GPL v2	**사전설치 솔루션**	• N/A
실행 하드웨어	• 일반 데스크탑 사양의 하드웨어	**버전**	• 2.19.1 (2018년 10월 기준)
특징	• 분산된 버전 컨트롤과 협업 지원 • 기존 분산시스템 대비 빠른 속도 제공 • 리눅스, 윈도우, MacOS 등 대다수 운영체제 지원 • Github, gitlab, sourceforge, bitbucket 등 다수의 프로젝트 관리 서비스에서 git 지원		
보안취약점	• 취약점 ID : CVE-2017-1000117 • 심각도 : 8.8 HIGH(V3) • 취약점 설명 : Git 'ssh : // URL 처리 결함으로 원격 사용자가 대상 시스템에서 임의의 명령 실행 가능 • 대응방안 :2.14.1 이상으로 업데이트 • 참고 경로 : https://www.securityfocus.com/bid/100283		
개발회사/커뮤니티	• Git 개발 커뮤니티 / git@vger.kernel.org		
공식 홈페이지	• https://git-scm.com		

2. 기능요약

- 로컬 시스템 내에서의 버전 관리 (init, add, commit)
- 버전별 변경 내용, 시간, 변경 내용 요약 메세지 관리 (log, show, status, diff)
- 개발 분기화와 분기된 개발 버전 간 통합 (branch, merge)
- 특정 버전으로의 회귀 (checkout)
- 특정 변경 내용의 취사적 통합 (cherry-pick)
- 최신 원격 저장소로의 변경사항 재적용 (rebase)
- 특정 변경 유발한 버전 찾는 바이너리 탐색 (bisect)
- 분산된 원격 저장소 관리 (clone, remote)

3. 실행 환경

- 대다수의 리눅스 배포판과 Mac OS에는 git이 설치되어 있음
- Git 공식 사이트 (https://git-scm.com/downloads) 에서 운영체제 별 설치파일 다운로드
- 받을 수 있음
- 커맨드 프롬프트에서 'git –v' 명령 통해 현재 설치된 git 버전 확인 가능
- 다양한 GUI 클라이언트가 존재하며 (https://git-scm.com/downloads), 다수의 IDE에서도 git 지원
- 본 문서는 Ubuntu 16.04 에서 터미널 통해 git 사용하는 경우 설명

4. 설치 및 실행

세부 목차

1. 저장소 만들기 (git init)
2. 파일들의 버전 상태 확인하기 (git status)
3. 변경된 내용 새로운 버전으로 만들기 (git add, git commit)
4. 현재 변경한 내용 보기 (git diff)
5. 버전 기록 보기 (git log)
6. 특정 파일 제거한 버전 만들기 (git rm)
7. 이전 버전으로 되돌리기 (git revert)

4. 설치 및 실행

4.1 저장소 만들기 (git init)(1/2)

- 저장소는 git 으로 버전 관리하는 대상 파일들과 버전 관리 정보 담는 디렉토리
- Repository 또는 repo 라고도 줄여서 부름
- 특정 디렉토리에서 'git init' 명령어 입력하면 해당 디렉토리가 git 저장소로 초기화
- 쓰기/읽기 권한만 있다면 어떤 계정으로 어떤 디렉토리에서 작업하는지는 상관없음
- 저장소가 정상적으로 초기화 되면 메시지가 뜸

저장소로 사용할 디렉토리 생성하고 그 안으로 이동

```
$ mkdir the_repo; cd the_repo
$ git init
Initialized empty Git repository in the_repo/.git/
$
```

4. 설치 및 실행

4.1 저장소 만들기 (git init)(2/2)

- Git 저장소는 버전 컨트롤 대상이 되는 파일들이 존재하는 공간 (워킹 스페이스) 과 버전 컨트롤 위한 관리 정보 (metadata) 담는 공간으로 나뉨
- 워킹 스페이스의 파일들은 git 저장소 디렉토리에 존재
- 후자는 '.git' 라는 이름의 디렉토리로 git 저장소 디렉토리 아래 존재하며, 리눅스에서 '.' 으로 시작하는 이름의 파일 / 디렉토리는 숨김 설정 되므로 평범하게는 보이지 않음
- 'ls –a' 명령어 사용하면 숨김 파일도 볼 수 있으므로, '.git' 디렉토리 확인

```
$ ls -a
.   ..  .git
$
```

4. 설치 및 실행

4.2 파일들의 버전 상태 확인하기 (git status)

- 저장소 내에서 'git status' 명령 통해 현재 저장소 내의 파일들의 상태 알 수 있음
- 저장소 디렉토리 내에 존재하지만 아직 버전 정보가 관리되지 않고 있는 파일, 마지막 버전 이후 변경 내용이 존재하는 파일, 삭제된 파일 등 표시

```
$ echo hello > hello
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hello

nothing added to commit but untracked files present (use "git add"
to track)
```

hello 라는 텍스트 담고 있는, hello 라는 이름의 파일 생성

저장소 디렉토리에 존재하지만 이 저장소에서 버전이 관리되지 않고 있는 파일

4. 설치 및 실행

4.3 변경된 내용 새로운 버전으로 만들기 (git add, git commit)

- 'git add' 명령 통해 특정 파일들의 현재 내용이 다음 버전이 될 것 git 에 알릴 수 있음
- 'git commit' 명령은 'git add' 통해 등록된 내용 실제 버전으로 만듬, '-m' 옵션통해 해당 버전에 대한 설명 하는 메시지 등록

```
$ git add hello
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hello

$ git commit -m "Add hello file"
[master (root-commit) 1b8505e] initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello
$
```

Hello 파일이 다음 버전으로 변경될 것임의미

4. 설치 및 실행

4.4 현재 변경한 내용 보기 (git diff)

- 'git diff' 명령 통해 현재 버전 관리 중인 파일의 변경된 내용 볼 수 있음

```
$ echo hello world > hello
$ git diff
diff --git a/hello b/hello  i
ndex ce01362..3b18e51 100644
--- a/hello
+++ b/hello
@@ -1 +1 @@
-hello
+hello world
$
```

'hello' 파일의 내용이 'hello' 에서 'hello world' 로 변경되었음 의미

4. 설치 및 실행

4.4 버전 기록 보기 (git log)(1/2)

- 'git log' 명령 통해 그동안의 버전 정보 볼 수 있음
- 이 정보에는 버전의 ID, 해당 버전 만든 사람, 해당 버전이 만들어진 시간, 해당 버전에 남긴 메시지가 있음

```
$ git log
commit 71ab1baad6e376fce4ee240b0ffa438a2f11ac1
Author: SeongJae Park <sj38.park@gmail.com
Date:   Mon Oct 29 07:29:31 2018 +0900

    hello: Update to hello world

commit b209ff04de1f1f20778ddd764055252c7691eb05
Author: SeongJae Park <sj38.park@gmail.com>  D
ate:    Mon Oct 29 07:29:07 2018 +0900

    Add hello file
$
```

버전의 ID

이 버전 만든 사람

이 버전이 만들어진 시간

4. 설치 및 실행

4.4 버전 기록 보기 (git log)(2/2)

- 'git show' 명령 통해 특정 버전의 전 버전 대비 변경 내용 볼 수 있음, 인자로 버전 ID 줘서 어떤 버전의 전 버전 대비 변경 내용 볼지 지정

```
$ git show 71ab1
commit 71ab1baad6e376fce4ee240b0ffa438a2f11ac1
Author: SeongJae Park
<sj38.park@gmail.com>
Date:   Mon Oct 29 07:29:31 2018 +0900

    hello: Update to hello world

diff --git a/hello b/hello
index ce01362..3b18e51 100644
--- a/hello
+++ b/hello
@@ -1 +1 @@
-hello
+hello world
```

71ab1 으로 시작하는 버전 ID 의 버전의 상세 정보 요청

'hello' 였던 내용 'hello world' 로 변경했음.

4. 설치 및 실행

4.5 특정 파일 제거한 버전 만들기 (git rm)

- 'git rm' 명령은 다음 버전부터 더 이상 버전 관리 하지 않 파일 지정
- 'git commit' 명령 통해 다음 버전 만들어 줘야 실제로 해당 파일이 버전 관리 대상에서 사라짐

```
$ git rm hello
rm 'hello'
sjpark@kabylab:~/git_solution$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)



sjpark@kabylab:~/git_solution$ git commit -m "hello: Stop version control"
[master ed5b186] hello: Stop version control
 1 file changed, 1 deletion(-)
 delete mode 100644 hello
sjpark@kabylab:~/git_solution$ git status
On branch master
nothing to commit, working tree clean
```

4. 설치 및 실행

4.6 이전 버전으로 되돌리기 (git revert)

- 'git revert' 명령은 특정 버전에서 변경한 내용 무효화 하는 버전 만듦
- 명령 수행 시 무효화 버전의 메시지 입력하는 텍스트 에디터가 뜸, 메시지 저장하고 에디터 종료하면 파일 내용이 이전 버전으로 되돌려진 걸 볼 수 있음

```
$ git revert ed5b1
[master afc4a5d] Revert "hello: Stop version control
"
 1 file changed, 1 insertion(+)
 create mode 100644 hello
sjpark@kabylab:~/git_solution$ git log
commit afc4a5d677df21e51d92edad587fcac3ec833bc5
Author: SeongJae Park <sj38.park@gmail.com>  D
ate:   Mon Oct 29 07:45:04 2018 +0900

    Revert "hello: Stop version control"

    This reverts commit ed5b1866f08603444364b56fd04b523aa2de58e6
    .

commit ed5b1866f08603444364b56fd04b523aa2de58e6
Author: SeongJae Park <sj38.park@gmail.com>
Date:   Mon Oct 29 07:39:42 2018 +0900

    hello: Stop version control
```

'ed5b1' 으로 시작하는 id 의 버전의 변경내용 되돌린다.

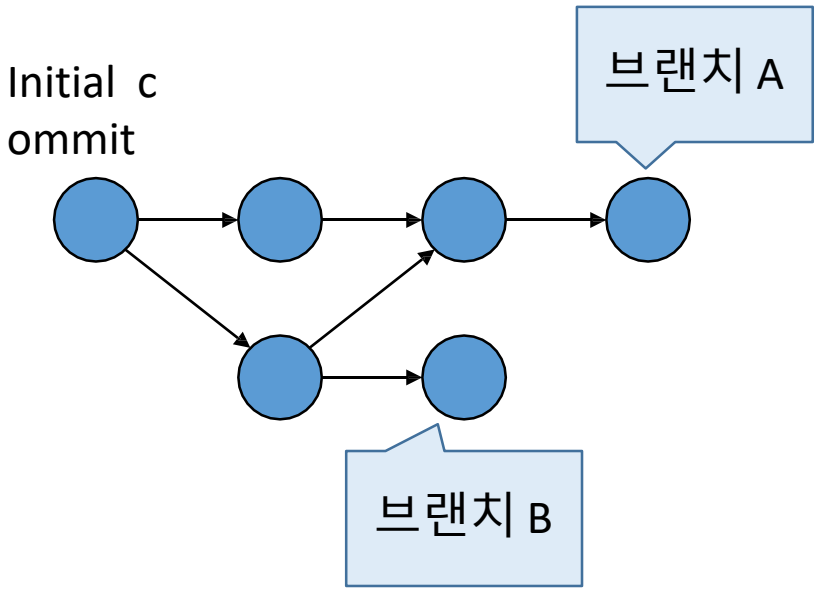
5. 기능소개

세부 목차

5. 기능소개

5.1 브랜치 개념 (1/2)

- 두개의 버전 사이에 전 / 후 (부모, 자식 이라고도 함) 관계가 존재
- 하나의 버전이 두개 이상의 자식 버전 가질 수도 있으며, 저장소의 버전들은 하나의 뿌리를 공유하지만 여러 개의 가지 (branch) 갖는 나무처럼 보여질 수 있음
- 각각의 가지 브랜치라 하며, 기존과 다른 개발 과정으로 분기 하는데 사용
- 버전은 두개 이상의 부모 버전 가질 수도 있음, 두개의 브랜치의 최신 버전 부모로 갖는 버전 만드는 것 두개의 브랜치 병합 (merge) 한다고 표현

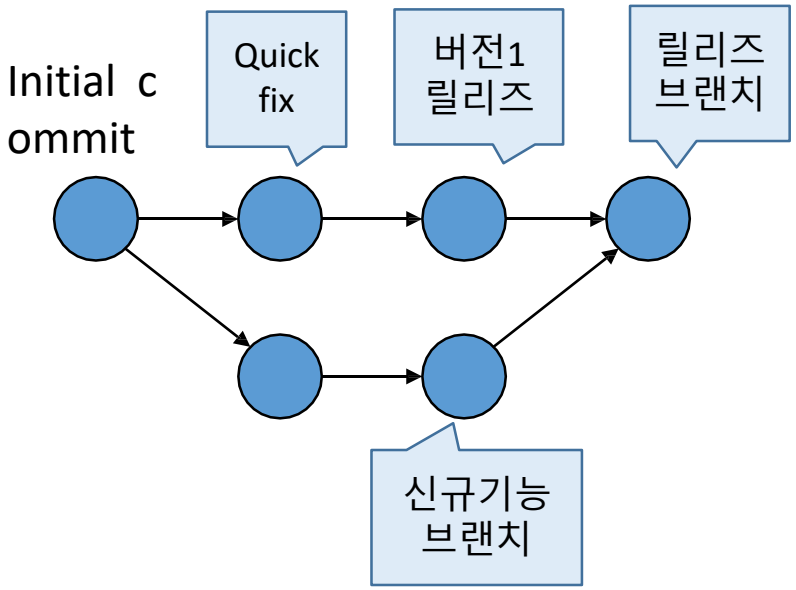


5. 기능소개

5.1 브랜치 개념 (2/2)

- 브랜치는 다양한 목적으로 사용될 수 있으며, git 에서 매우 많이 사용되는 기능
- 빠른 시일 내에 릴리즈가 예정되어 있는데 그때까지 마무리 하기 어려운 신규 기능 구현하는 경우라면 별도 브랜치에서 신규 기능 구현하고, 릴리즈 후에 기존 브랜치에 해당 신규 기능 병합시킬 수 있음

  신규 기능이 별도 브랜치에서 개발되는 동안, 빠른 시일 내에 수정 가능한 간단한 버그는 기존 브랜치에서 재빨리 수정하고 릴리즈 할 수 있음



5. 기능소개

5.2 브랜치 목록 보기

- 'git branch' 명령은 현재 저장소에 존재하는 브랜치 목록 보여줌
- 목록 중 현재 저장소가 따라가고 있는 브랜치에는 '* ' 표시, 새로 'git commit' 통해 버전 만들면 해당 브랜치는 이 '* '표시가 된 브랜치의 최신 버전 됨
- 'master' 브랜치는 'git init' 시에 기본적으로 만들어 주는 브랜치

```
$ git branch
* master
$
```

5. 기능소개

5.3 브랜치 생성 및 이동하기

- 'git branch' 명령에 새 브랜치 이름 인자로 주면 해당 이름의 브랜치 생성
- 'git checkout' 명령에 브랜치 이름 인자로 주면 해당 브랜치로 이동
- 브랜치 이동하면 워킹 스페이스의 파일들이 해당 브랜치 최신 버전의 내용으로 바뀜
- 브랜치 이동 후 버전 만들면 이동된 브랜치의 최신 버전 됨

```
$ git branch
* master
$ git branch new_branch
$ git branch
* master  ne
  w_branch
$ git checkout new_branch  Swit
ched to branch 'new_branch'
$ git branch
  master
* new_branch
```

5. 기능소개

5.4 다른 브랜치와 병합하기

- 'git merge' 결과 타겟 브랜치 최신 버전과 현재 브랜치의 최신 버전의 내용이 병합된새로운 버전 생성
- 'git merge' 로 생성된 버전은 현재 브랜치의 최신 버전이 되며, 현재 브랜치의 기존 최신 버전과 타겟 브랜치의 최신 버전 부모 브랜치로 갖게 됨

```
$ git merge new_branch
Merge made by the 'recursive' strategy.
 bye | 0
 1 file changed, 0 insertions(+), 0 deletions(-)

 create mode 100644 bye
$ git log --graph --oneline
*   0fb9c35 Merge branch 'new_branch'
|\
| * 41f0c59 Add bye file
* | ed5b186 hello: Stop version control
* | 71ab1ba hello: Update to hello world
|/
* b209ff0 Add hello file
$
```

'master' 브랜치의 최신 버전

'new_branch' 브랜치의 최신 버전

'master' 브랜치의 버전

'master' 브랜치의 버전

6. 활용예제

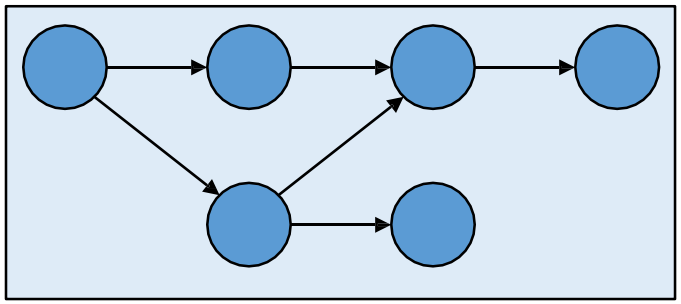
세부 목차

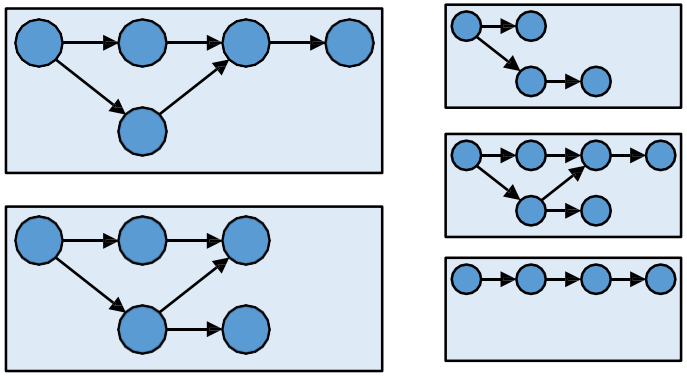
1. 원격 저장소 개념
2. 원격 저장소 복사하기
3. 원격 저장소 추가하기
4. 원격 저장소의 버전 가져오기
5. 원격 저장소로 버전 보내기
6. 패치 만들기
7. 패치 보내고 받기
8. 패치 주고받는 다른 방법들

6. 활용예제

6.1 원격 저장소 개념

- 과거의 중앙형 버전 관리 시스템의 저장소는 중앙 서버와 클라이언트로 구성되어, 모든 버전 데이터는 서버에 존재하고, 개발자들은 클라이언트 통해 여기 접속하는 구조였음
- Git 은 서버 / 클라이언트 없이 모든 개발자가 자신의 저장소 가지며 저장소 간에 버전 정보를 주고 받으며 협업할 수 있게 하는 분산 버전 관리 시스템의 형태임
- 따라서 별도로 중앙 저장소가 존재하지 않고, 남들의 저장소 중 공신력 있는 개발자의 저장소에 개발한 내용이 모이게 되고, 그렇게 모인 최신 버전 다른 개발자들이 자신의 저장소에 가져가게 되는, 유연하고 안정적인 협업 모델 구축
- 남의 저장소 원격 저장소 (remote repository)라고 함





6. 활용예제

6.2 원격 저장소 복사하기

- '.git' 디렉토리로 접근 가능한 프로토콜 제공하는 저장소는 해당 프로토콜 통해 복사가 가능하며, 주로 사용되는 프로토콜은 http, https, ssh 등이고, git 독자 프로토콜 존재
- 'git clone' 명령 통해 원격 저장소 자신의 컴퓨터 위로 복사 가능
- 복사된 저장소는 명령 내린 디렉토리 아래 생성됨
- 저장소의 원격 저장소 정보는 'git remote' 명령 통해 확인 가능

```
$ git clone https://github.com/git/git
Cloning into 'git'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 257917 (delta 0), reused 0 (delta 0), pack-reused 257916
Receiving objects: 100% (257917/257917), 102.83 MiB | 11.28 MiB/s, done.
Resolving deltas: 100% (190967/190967), done.
$ ls
git
$ cd git; git remote -v
origin  https://github.com/git/git (fetch)
origin  https://github.com/git/git (push)
$
```

6. 활용예제

6.3 원격 저장소 추가하기

- 프로젝트의 기능별로 다른 관리자가 존재하는 경우 (Linux 커널이 이에 해당) 특정 기능 관리자의 저장소가 해당 기능 위한 최신 버전 가지고 있으므로 여러 원격 저장소 추가하고 필요에 따라 각 저장소의 파일 가져와 작업해야 함
- 'git remote add' 명령 통해 직접 원격 저장소 추가할 수 있음

```
$ git remote add korg git://git.kernel.org/pub/scm/git/git.git
$ git remote -v
korg    git://git.kernel.org/pub/scm/git/git.git (fetch)
korg    git://git.kernel.org/pub/scm/git/git.git (push)
origin  https://github.com/git/git (fetch)
origin  https://github.com/git/git (push)
$
```

6. 활용예제

6.4 원격 저장소의 버전 가져오기

- 'git remote update' 명령은 원격 저장소의 버전들 가져옴
- 'git fetch <원격 저장소 이름>' 명령 통해 특정 원격 저장소의 버전들만 가져올 수 있음
- 'git checkout' 명령에 '<원격 저장소 이름>/<원격 저장소의 브랜치 이름>' 인자로 줘서 특정 원격 저장소의 특정 브랜치 파일 워킹 스페이스로 가져올 수 있음

```
$ git remote update
Fetching origin  Fe
tching korg
$ git fetch origin
$ git checkout origin/master
Note: checking out 'origin/master'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
…
```

6. 활용예제

6.5 원격 저장소로 버전 보내기

- 'git push <원격 저장소 이름> <브랜치 이름>' 명령은 특정 저장소로 특정 브랜치의 버전들 보냄
- 해당 원격 저장소로 쓰기 권한이 있어야만 가능

```
$ git push self master
Username for 'https://github.com': sjp38
Password for 'https://sjp38@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 358 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/sjp38/git.git
   c670b1f..1597845  master -> master
$
```

6. 활용예제

6.6 패치 만들기

- 협업 하는 경우라면 자신이 만든 버전의 파일 상대방과 공유해야 함
- 간단한 내용은 변경 사항 담은 패치 만들어 프로젝트 관리자에게 보내고, 이걸 프로젝트 관리자가 자신의 저장소에 반영하는게 일반적임
- 'git format-patch <버전>' 명령 이용해 대상 버전으로부터 현재 워킹 스페이스 버전 사이의 버전들 패치 파일들로 만들 수 있음

'origin' 원격 저장소의 'master' 브랜치 최신 버전으로부터 현재 최신 버전 사이의 버전들 패치로 만들어냄

```
$ git format-patch origin/master
0001-README-Meaninglessly-for-example.patch
$ cat 0001-README-Meaninglessly-for-example.patch
From dbeaa067e34816c3bc638679342186e5b66abae7 Mon Sep 17 00:00:00 2001
From: SeongJae Park <sj38.park@gmail.com>
Date: Mon, 29 Oct 2018 09:40:09 +0900
Subject: [PATCH] README: Meaninglessly for example
…
```

6. 활용예제

6.7 패치 보내고 받기

- 'git send-email' 명령 통해 특정 패치 파일 특정 사용자에게 이메일로 보낼 수 있음
- 메일 전송 위해서 메일 서버 설정은 별도로 해줘야 함
- 'git am' 명령은 남들로부터 받은 패치 자신의 저장소 적용

```
$ git send-email --to sj38.park@gmail.com 0001-README-Meaninglessly-for-
example.patch
0001-README-Meaninglessly-for-example.patch
…
From: SeongJae Park <sj38.park@gmail.com>
To: sj38.park@gmail.com
Subject: [PATCH] README: Meaninglessly for example
Date: Mon, 29 Oct 2018 09:47:31 +0900
…
$ git am 0001-README-Meaninglessly-for-example.patch
Applying: README: Meaninglessly for example
$ git log --oneline
1597845 README: Meaninglessly for example
c670b1f Sixth batch for 2.20
16ce0b9 Merge branch 'js/mingw-default-ident'
```

6. 활용예제

6.8 패치 주고받는 다른 방법들

- 협업자들 간에 수정한 내역 주고 받는 데는 이외에도 다양한 방법 있음
- 작은 프로젝트에서는 하나의 원격 저장소에 협업자 모두가 'git push' 하기도 함
- 타인이 접근할 수 있는 자신의 저장소에 최신 버전 올려두고 타인에게 자신의 저장소 주소와 최신 버전 올려둔 브랜치 이름 알려줘서 타인이 직접 해당 버전들 가져갈 수 있게 하기도 하며, 이런 이야기 알리는 메시지 풀리퀘스트 (pull request)라고 함
- Git 저장소 호스팅 및 협업 서비스 제공하는 github에서는 web interface 통해 pull request 주고 받 수 있는 기능 제공

7. FAQ

Q **'git revert' 는 버전 되돌린다는 변경 사항이 남는데 어떤 기록도 없이 예전 버전으로 되돌릴 수는 없나요?**

A 해당하는 예전 버전으로 브랜치 새로 만들고 거기서부터 재시작하는 방법도 있습니다. 그 외에도 'git reset –hard' 등의 명령 이용하는 방법이 있습니다만, 일반적으로 권장되는 방법이 아니라 이 문서에서는 설명하지않습니다.

Q **꼭 CLI 로 git 다뤄야만 하나요?**

A 문서에서도 설명했듯 git 위한 많은 GUI 도구들이 나와 있습니다. 하지만 모든 GUI 도구들이 모든 git 기능 지원하는 건 아니라서 일부 심화 기능은 사용이 어려울 수도 있습니다.

8. 용어정리

용어	설 명
저장소 (Repository)	Git 이용해 관리되는 파일들과 그 관리 정보 담는 디렉토리
커밋 (Commit)	버전 만드는 행위, 또는 특정 버전의 스냅샷
브랜치 (Branch)	부모-자식 관계로 엮인 버전들의 연결
원격 저장소 (Remote repository)	https, ssh, git 등의 프로토콜 통해 접근 가능한 타인의 git 저장소
푸시 (Push)	로컬 저장소의 변경 사항 원격 저장소로 업데이트 하는 행위
패치 (Patch)	소스코드의 변경사항 설명하는 파일
풀리퀘스트 (Pullrequest)	자신의 버전이 어느 주소의 원격 저장소에 있으니 자신의 저장소로 가져가 달라는 요청 메시지

Open Source Software Installation & Application Guide