오픈소스소프트웨어

Open-Source Software

ICT융합학부 조용우

Git

What is Git?

- Git is an open source, distributed version control system designed for speed and efficiency
- Linus Torvalds, 2005

Git

- https://git-scm.com/
- 분산 버전관리 시스템
- 작업 파일을 로컬 저장소 뿐만 아니라 원격 저장소에도 배포하여 저장
- 작업 이력을 관리하고 분기하여 원하는 시점으로 파일을 복원 혹은 통합할 수 있음
- 원격 저장소를 활용하여 팀 단위 협업 가능

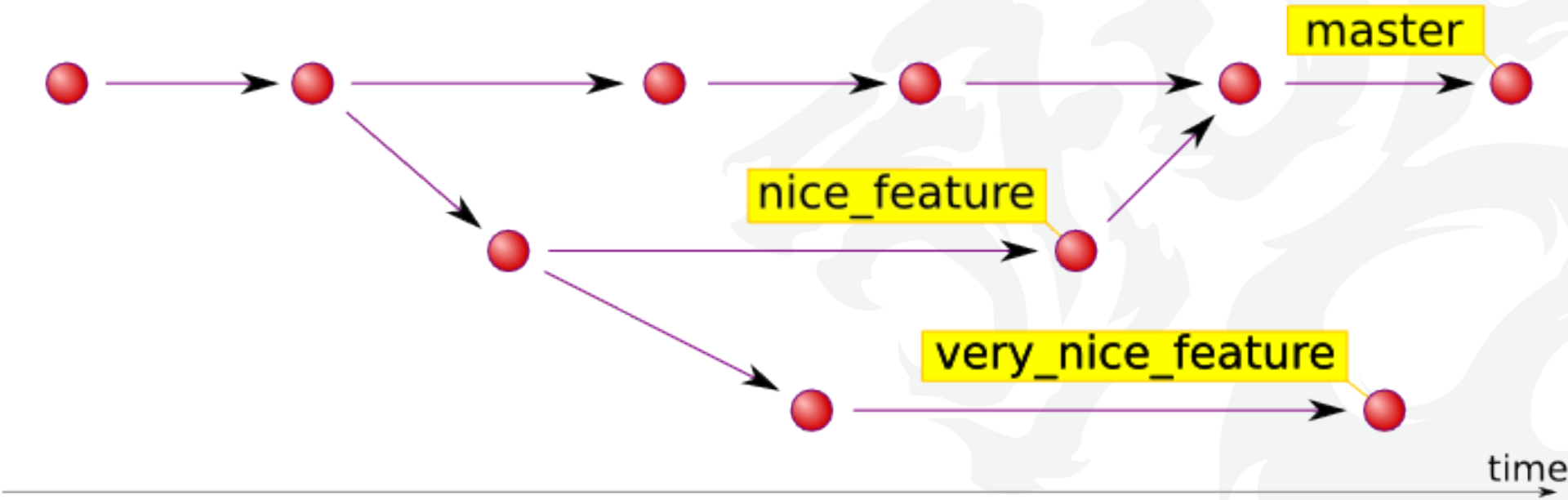
Version control system generals

- Version control
- Branching and merging

Git의 특징#1 - 브랜치와 머지

- 멀티플 로컬 브랜치를 지원

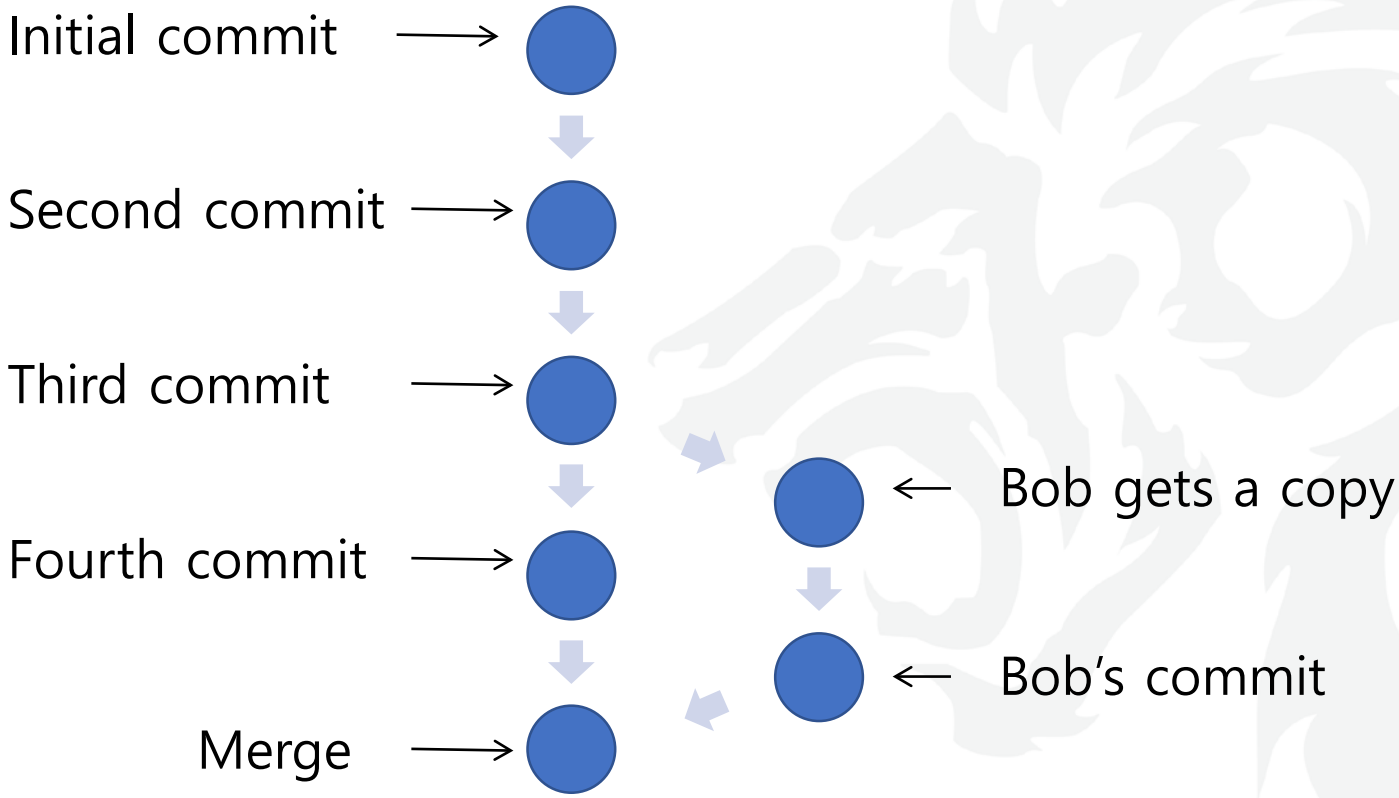
Branching



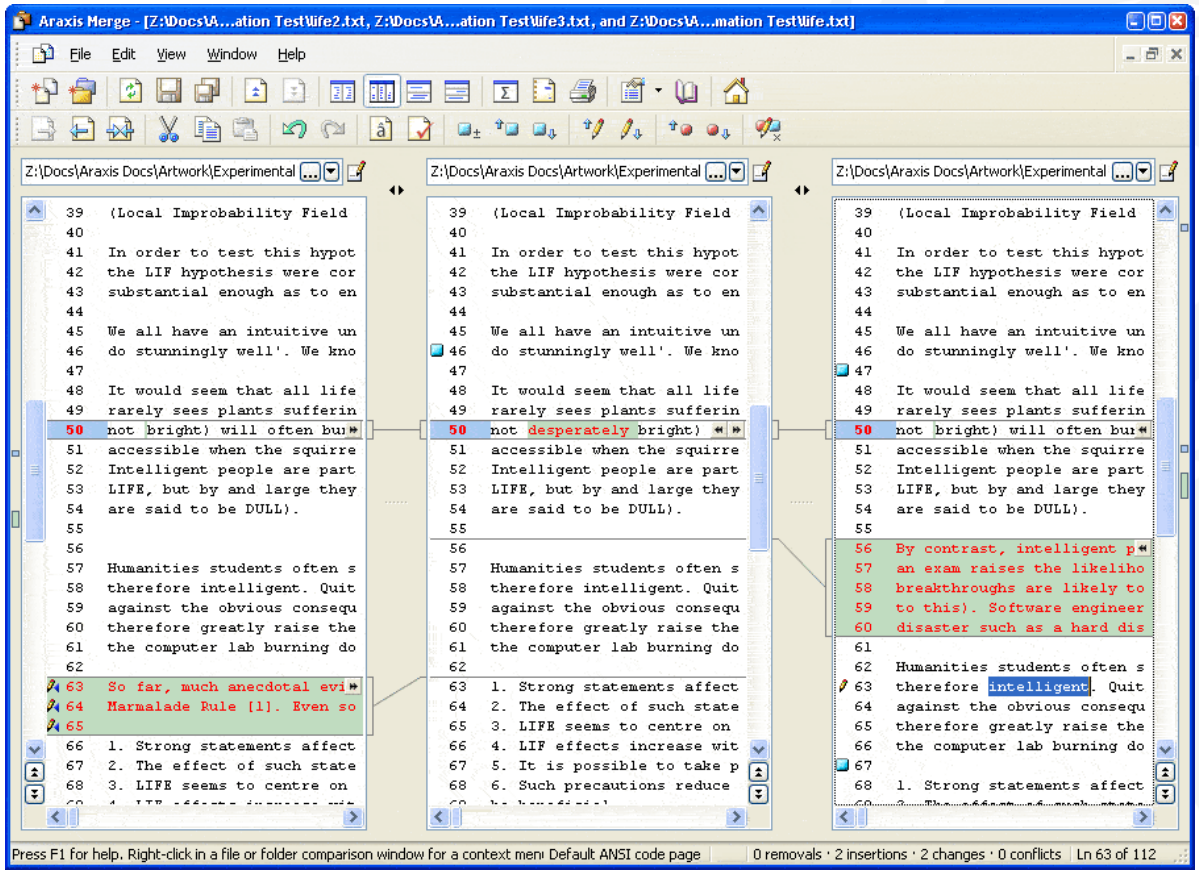
Merging

- There are occasions when multiple versions of a file need to be collapsed into a single version.
- E.g. A feature from one branch is required in another
- This process is known as a merge.
- Difficult and dangerous to do in CVS
- Easy and cheap to do it git

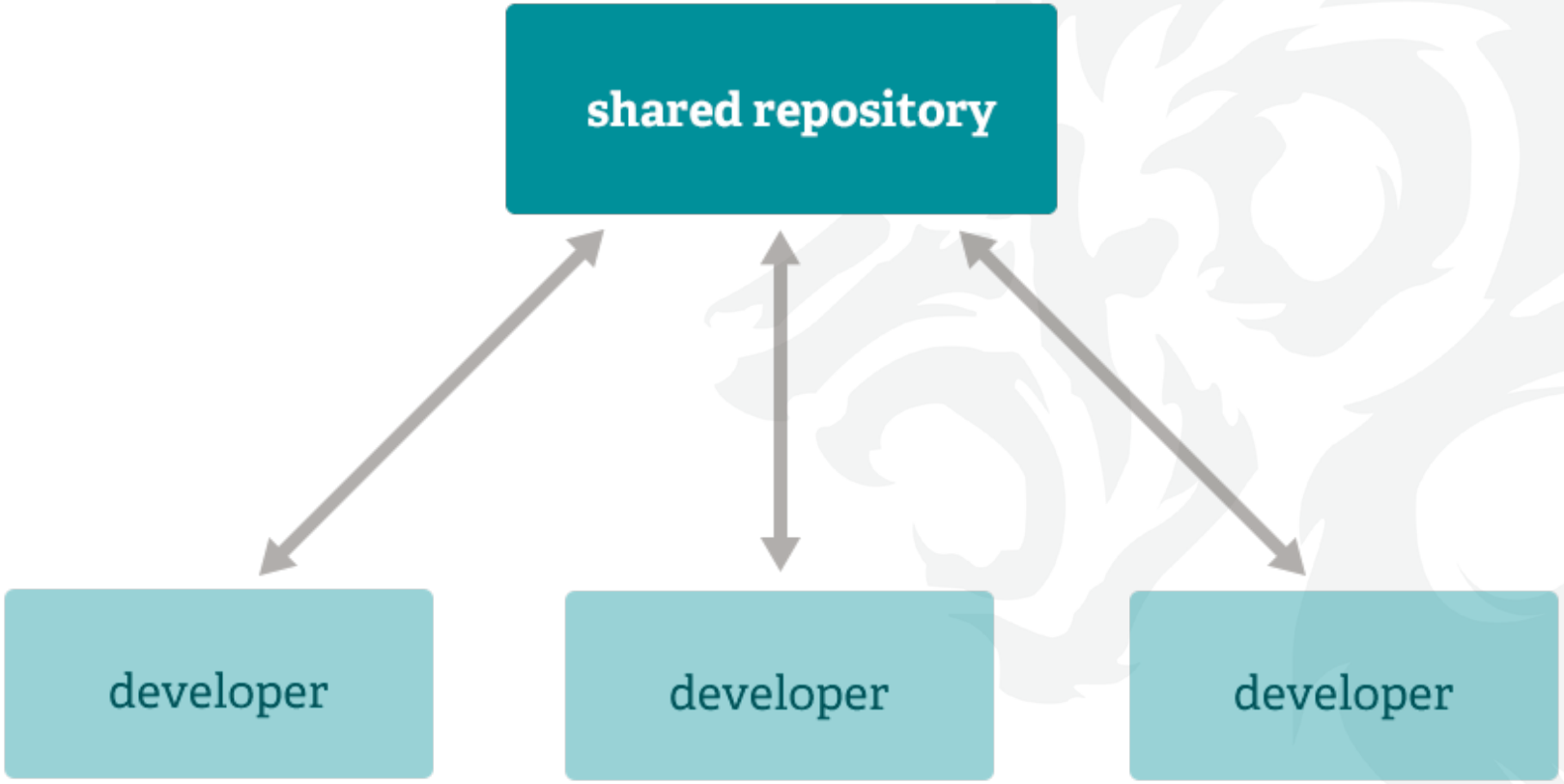
Multiple versions



Merging


Araxis Merge - [Z:\Docs\A...ation Test\life2.txt, Z:\Docs\A...ation Test\life3.txt, and Z:\Docs\A...mation Test\life.txt]
File Edit View Window Help
Z:\Docs\Araxis Docs\Artwork\Experimental
Z:\Docs\Araxis Docs\Artwork\Experimental
Z:\Docs\Araxis Docs\Artwork\Experimental
(Local Improbability Field
In order to test this hypot
the LIF hypothesis were cor
substantial enough as to en
We all have an intuitive un
do stunningly well'. We kno
It would seem that all life
rarely sees plants sufferin
not bright) will often bu
accessible when the squirre
Intelligent people are part
LIFE, but by and large they
are said to be DULL).
Humanities students often s
therefore intelligent. Quit
against the obvious consequ
therefore greatly raise the
the computer lab burning do
So far, much anecdotal evi
Marmalade Rule [1]. Even so
1. Strong statements affect
2. The effect of such state
3. LIFE seems to centre on
not desperately bright)
4. LIF effects increase wit
5. It is possible to take p
6. Such precautions reduce
By contrast, intelligent p
an exam raises the likeliho
breakthroughs are likely to
to this). Software engineer
disaster such as a hard dis
therefore intelligent. Quit
Press F1 for help. Right-click in a file or folder comparison window for a context menu Default ANSI code page
0 removals · 2 insertions · 2 changes · 0 conflicts
Ln 63 of 112

Git의 특징#2 - 분산형



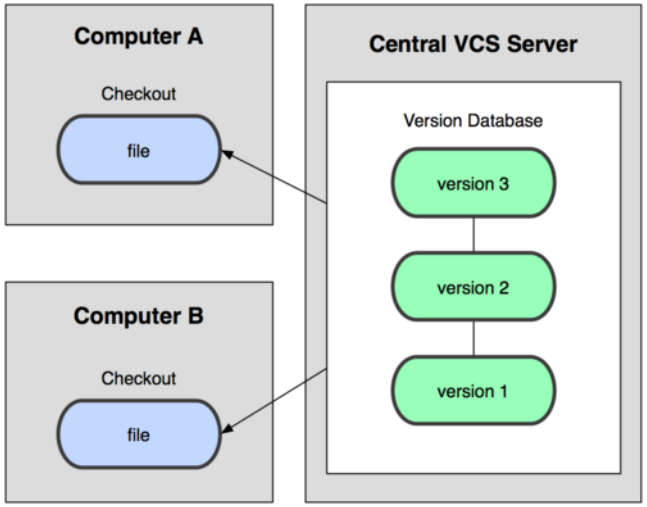
CVS vs. DVS
: Centralized version control generals

- A single server holds the code base
- Clients access the server by means of check-in/check-outs
- Examples include CVS, Subversion, Visual Source Safe.
- Advantages: Easier to maintain a single server.
- Disadvantages: Single point of failure.

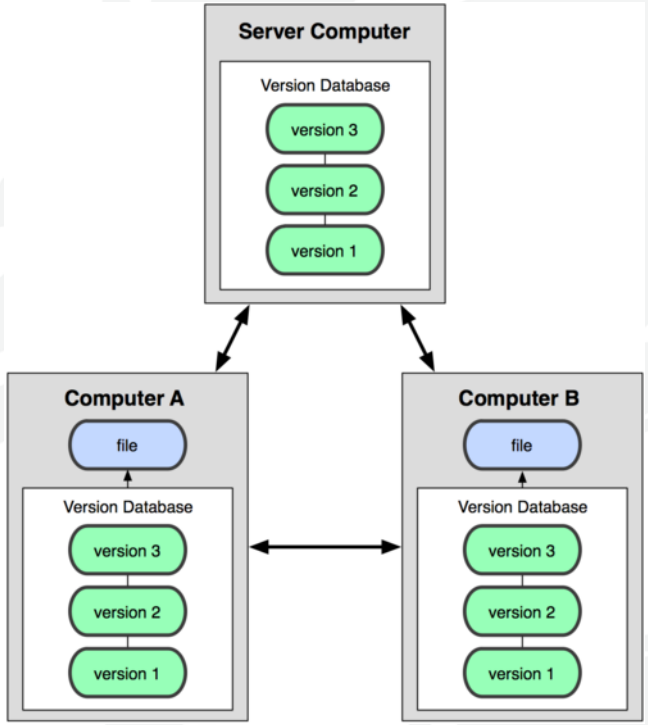
CVS vs. DVS
: Distributed version control generals

- Each client (essentially) holds a complete copy of the code base.
- Code is shared between clients by push/pulls
- Advantages: Many operations cheaper. No single point of failure
- Disadvantages: A bit more complicated!

Git uses a distributed model



Centralized model



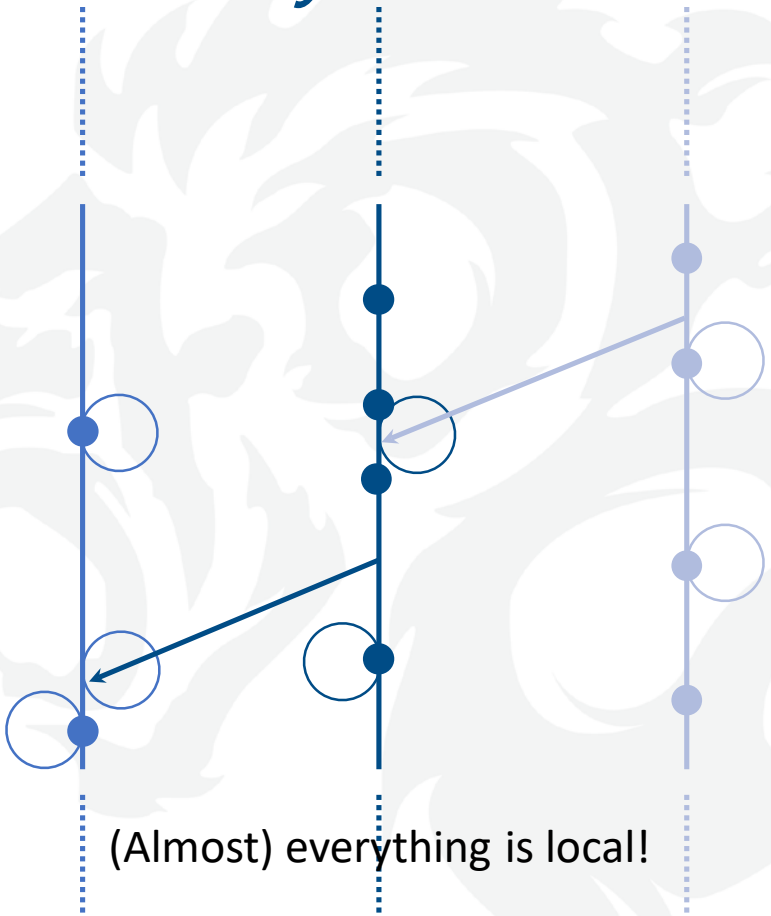
Distributed model
: Many operations are local

Git의 특징#3 - 크기와 속도

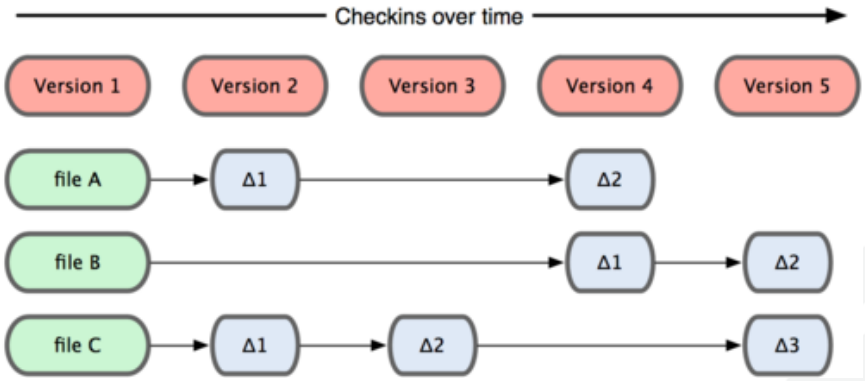
- C언어로 작성되어 작고 빠르다

Git is designed for *speed* and *efficiency*

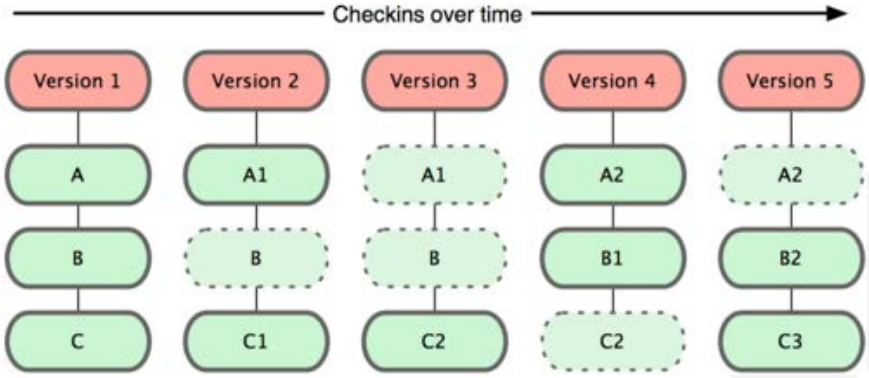
- No network needed for
  - → Performing a diff
  - → Viewing file history
  - → Committing changes
  - → Merging branches
  - → Obtaining any other revision of a file
  - → Switching branches



Git takes snapshots



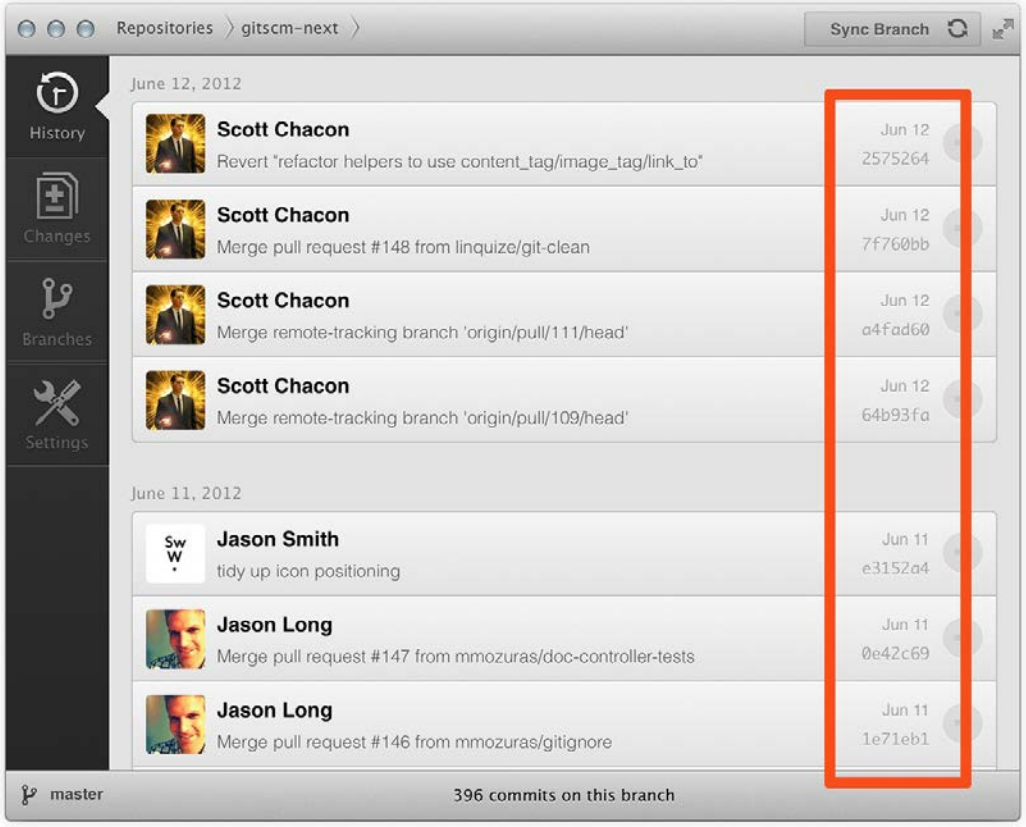
*Storing data as **changes** to a base version of each file*



*Storing data as **snapshots** of the project over time*

Git의 특징#4 - 데이터 일관성

- 체크썸 확인



Git uses checksums

- In Subversion each modification to the central repository incremented the version # of the overall repo.
- How will this numbering scheme work when each user has their own copy of the repository, and commits changes to their local copy of the repo before pushing to the central server?
- Instead, Git generates a unique SHA-1 hash – 40 character string of hex digits, for every commit. Refer to commits by this ID rather than a version number. Often we only see the first 7 characters:

```
1677b2d Edited first line of readme
258efa7 Added line to readme
0e52da7 Initial commit
```

Modification
: Check-out generals

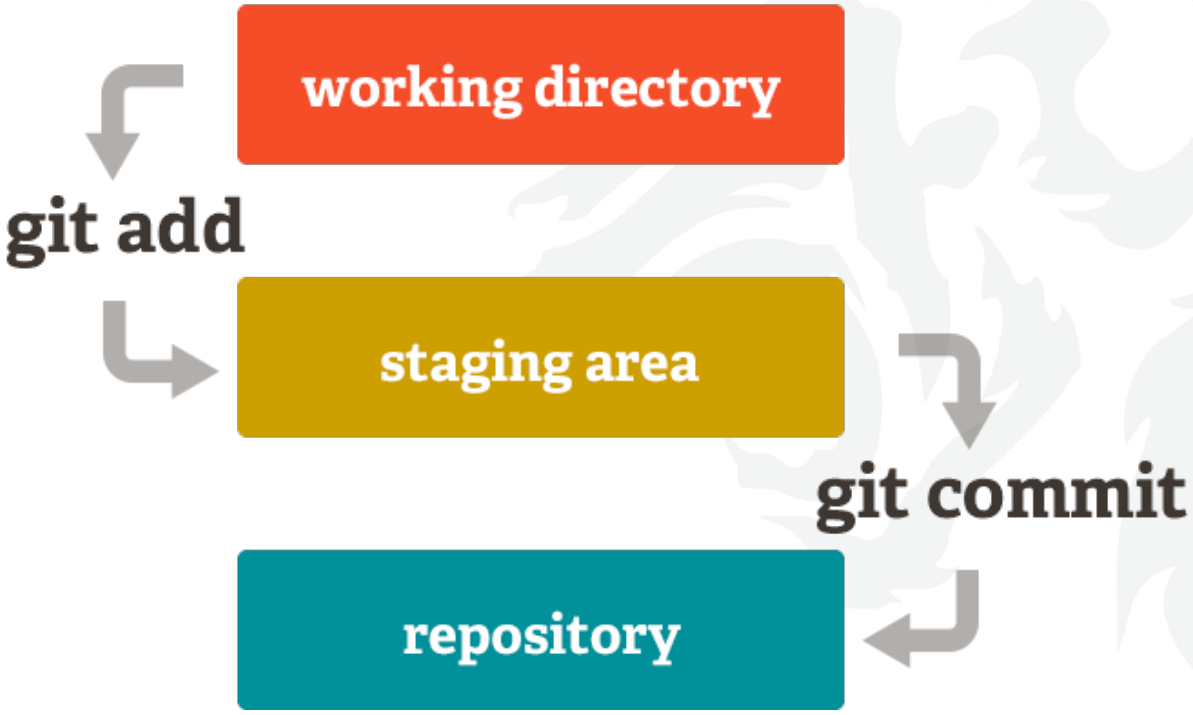
- If you want to make a change the file needs to be checked out from the repository
- Usually done a file at a time.
- Some VCSs will lock checked out files so only one person may edit at a time.

Modification : Check-in generals

- When changes are completed the new code is checked-in.
- A commit consists of a set of checked in files and the diff between the new and parent versions of each file.
- Each check-in is accompanied by a user name and other meta data.
- Check-ins can be exported from the Version Control system the form of a patch.

Git의 특징#5 - "Staging Area"

- "Staging Area"라고 하는 중간영역을 사용



Basic Git workflow

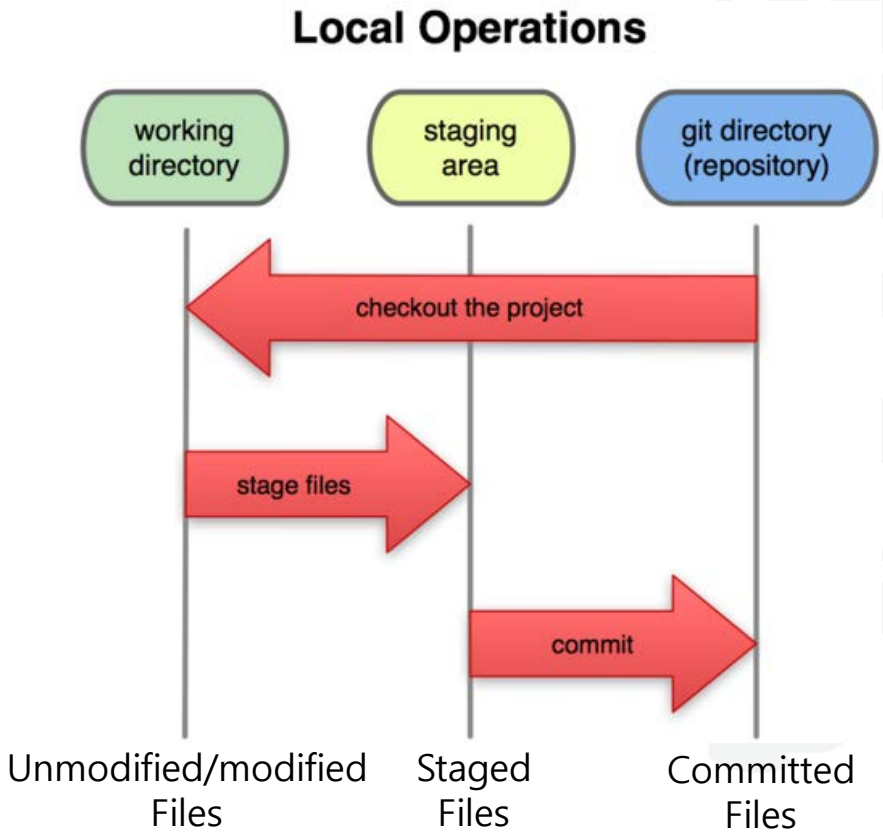
1. Modify files in your working directory.
2. Stage files, adding snapshots of them to your staging area.
3. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

- Notes:
  - → If a particular version of a file is in the git directory, it's considered committed.
  - → If it's modified but has been added to the staging area, it is staged.
  - → If it was changed since it was checked out but has not been staged, it is modified.

작업흐름

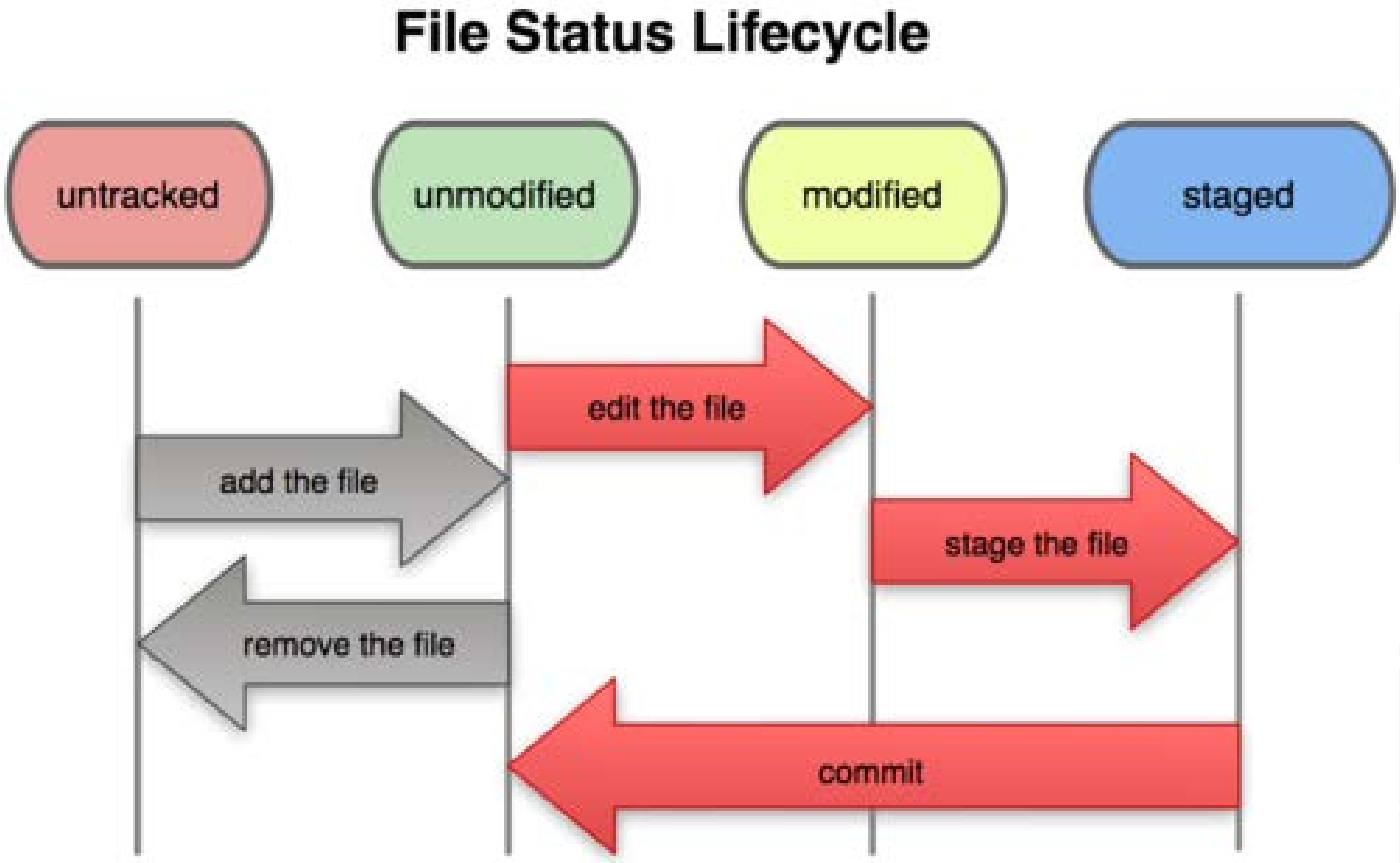
- Git은 파일을 committed, modified, stage의 3가지 상태로 관리
  - committed: 데이터가 로컬 저장소에 안전하게 저장됨
  - modified: 수정한 파일을 아직 로컬 저장소에 커밋하지 않음
  - staged: 현재 수정된 파일을 곧 커밋할 예정이라 표시한 상태
- Git 디렉토리에 있는 파일들은 committed 상태이다. 파일을 수정하고 Staging Area에 추가했다면 staged이다. 그리고 Checkout하고 나서 수정했지만, 아직 Staging Area에 추가하지 않았다면 modified이다.

A Local Git project has 3 areas

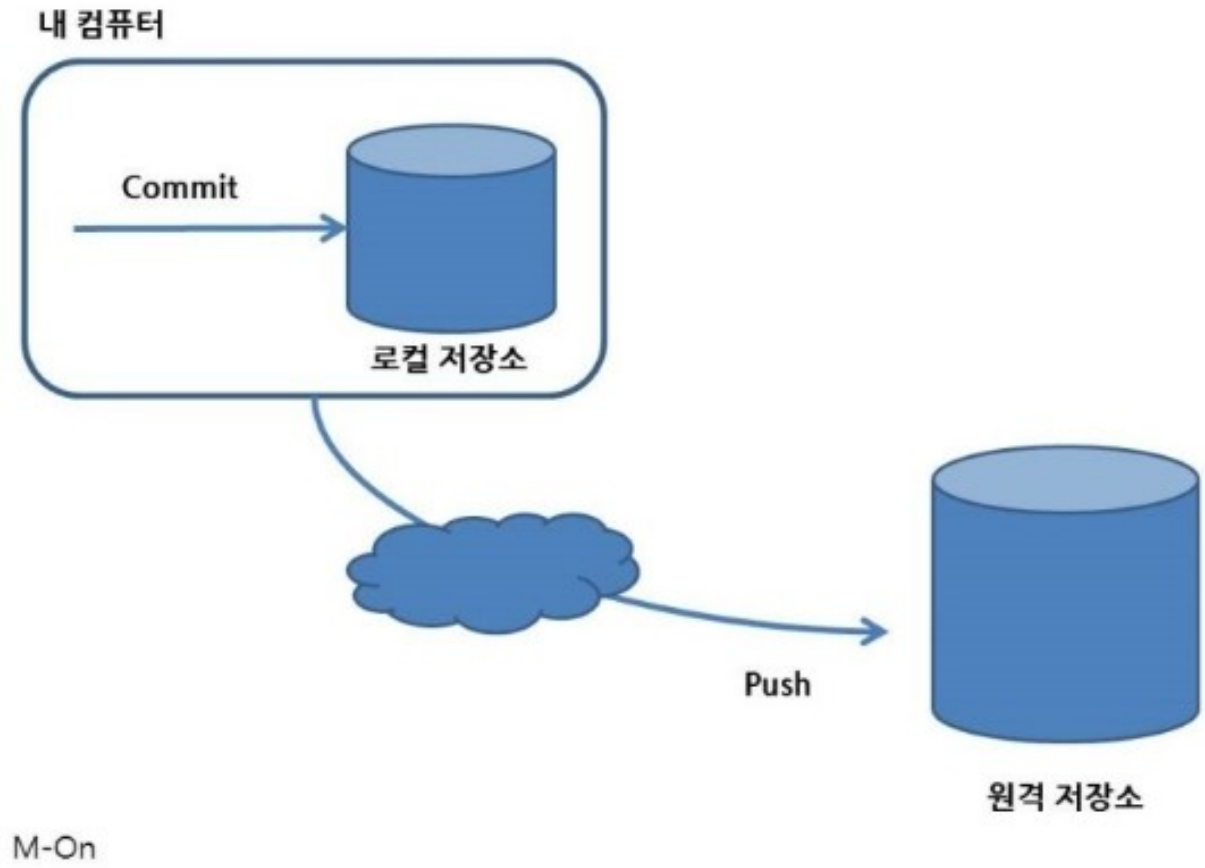


Note: working directory sometimes called the “working tree”, staging area sometimes called the “index”.

Git file lifecycle



작업흐름



작업흐름

- 저장소 생성 or 복제로 작업 디렉토리를 만들기
- 작업할 파일을 생성
- 파일을 스테이징 영역에 추가/제거
- 커밋
- 리모트 저장소에 push

Working with others

- Here's what you normally do:
  - Download the current HEAD from the central repository
  - Make your changes
  - Commit your changes to your local repository
  - Check to make sure someone else on your team hasn't updated the central repository since you got it
  - Upload your changes to the central repository
- If the central repository has changed since you got it:
  - It is your responsibility to merge your two versions
    - This is a strong incentive to commit and upload often!
  - Git can often do this for you, if there aren't incompatible changes

Advice: Keeping it simple

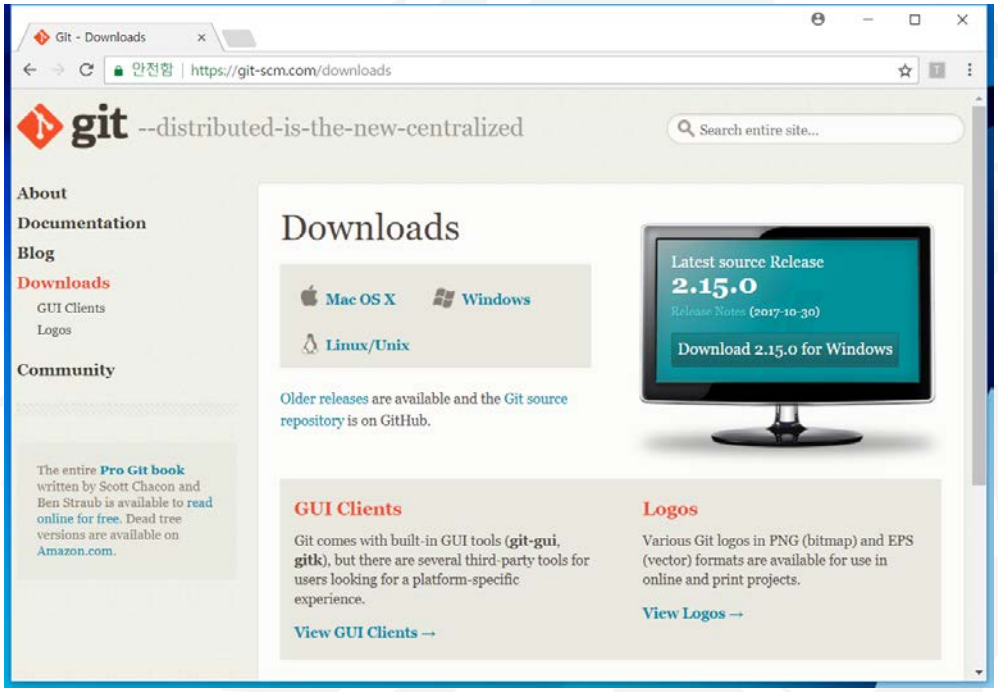
- If you:
  - → Make sure you are current with the central repository
  - → Make some improvements to your code
  - → Update the central repository before anyone else does
- Then you don't have to worry about resolving conflicts or working with multiple branches
  - → All the complexity in git comes from dealing with these
- Therefore:
  - → Make sure you are up-to-date before starting to work
  - → Commit and update the central repository frequently

Git의 특징#6 - 오픈소스

- GNU General Public License version 2.0 (GPLv2)

Git commands

Download and install



- http://git-scm.com/downloads
- https://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide#323764

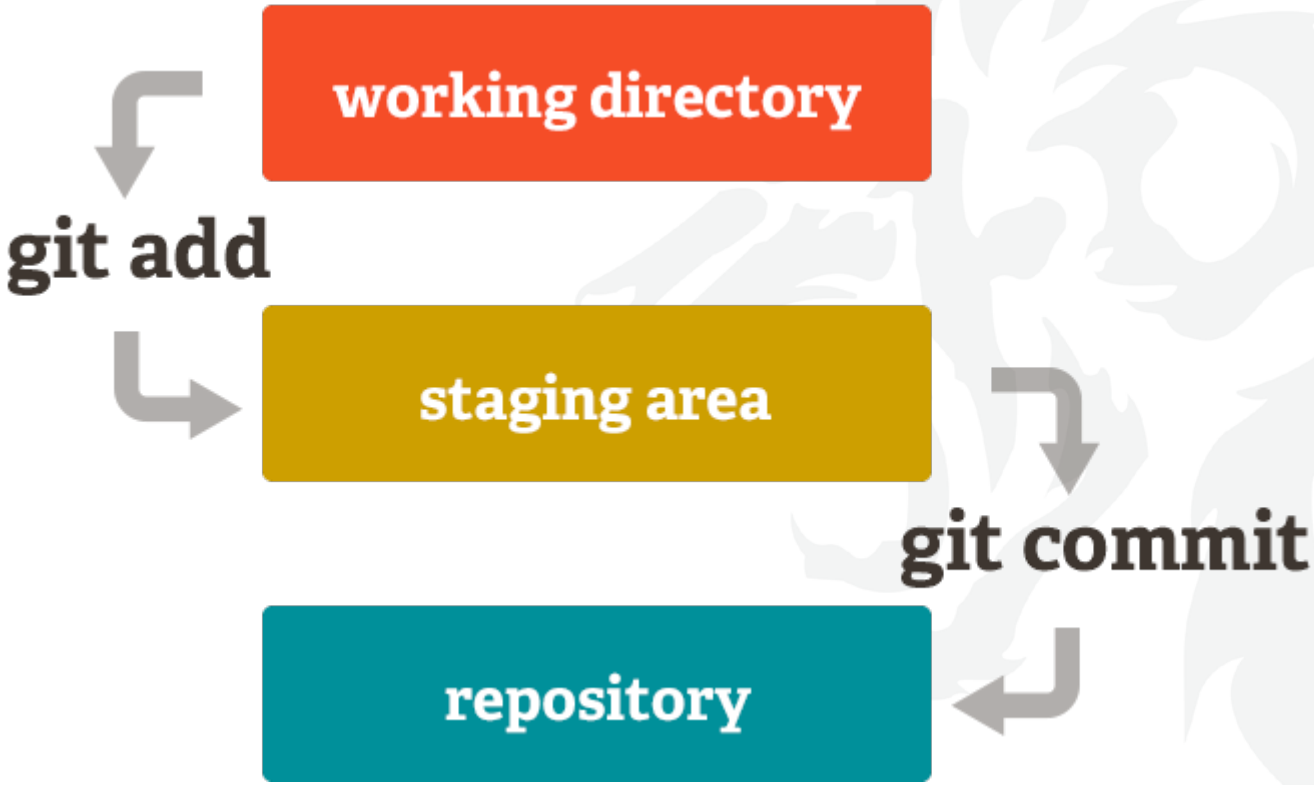
Git commands for initial setup

```
$git config --global user.name "<YOUR_NAME>"
```

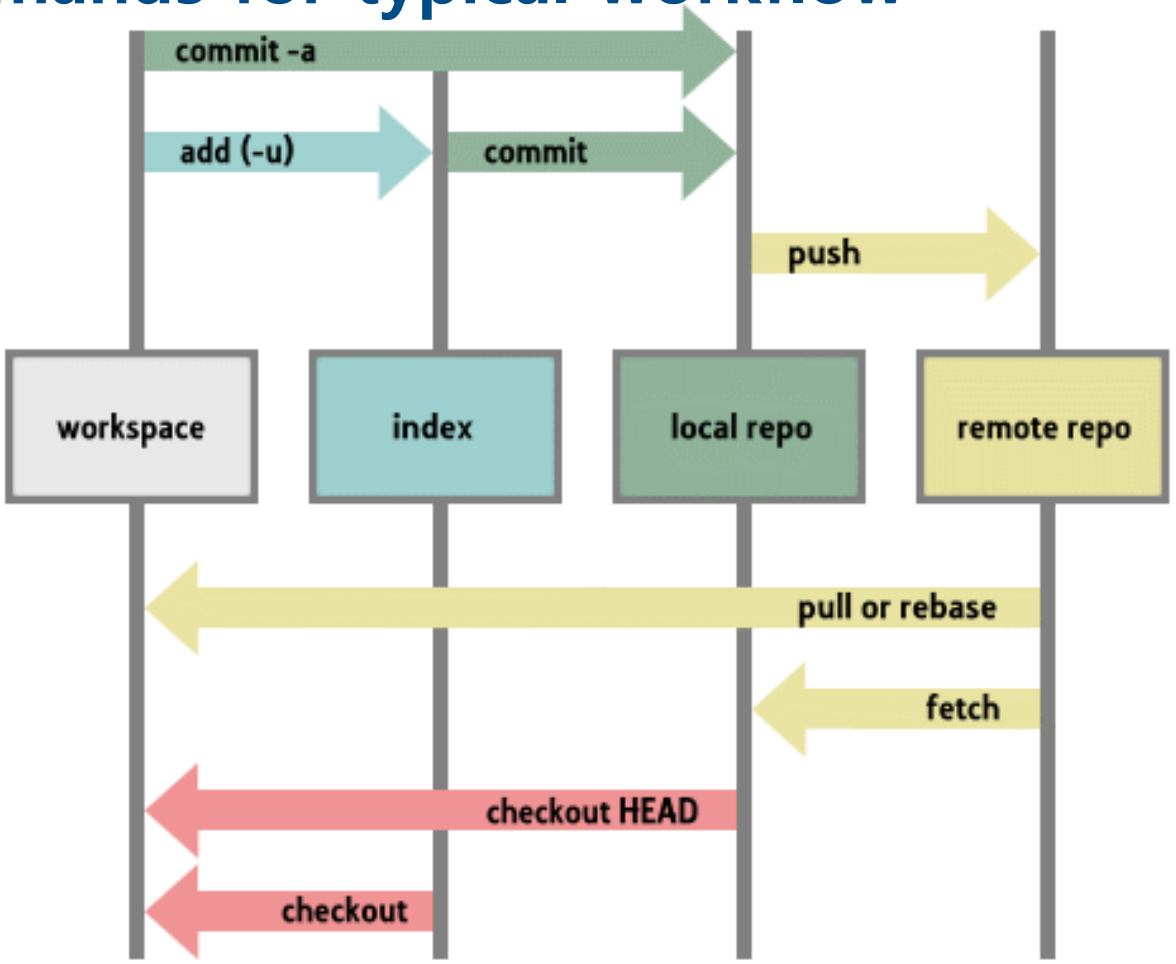
```
$git config --global user.email "<YOUR_EMAIL>"
```

- or edit them in ~/.gitconfig

Git commands: add / commit



Git commands for typical workflow



Git commands for typical workflow

`git pull remote_repository`

- → Get changes from a remote repository and merge them into your own repository

`git status`

- → See what Git thinks is going on
- → Use this frequently!

- Work on your files (remember to add any new ones)

`git commit –m "What I did"`

`git push`

clone
: Git command to create a local copy of a repo

- To clone an already existing repo to your current directory:

```
$ git clone <url> [local dir name]
```

  - → This will create a directory named *local dir name*, containing a working copy of the files from the repo, and a .git directory (used to hold the staging area and your actual repo)

init
: Git command to create a local copy of a repo

- To create a new Git repo in your current directory:

```
$ git init
```

- This will create a .git directory in your current directory.
- Then you can commit files in that directory into the repo:

```
$ git add filename
$ git commit –m "initial project version"
```

Git commands for committing files

- The first time we ask a file to be tracked, and every time before we commit a file we must add it to the staging area:

```
$ git add README.txt hello.c
```

  - → This takes a snapshot of these files at this point in time and adds it to the staging area.

- To move staged changes into the repo we commit:

```
$ git commit –m "Fixing bug #22"
```

- Note: To unstage a change on a file before you have committed it:

```
$ git reset HEAD  --  filename
```

- Note: To unmodify a modified file:

```
$ git checkout  --  filename
```

- Note: These commands are just acting on your local version of repo.

Git commands: status / diff

- To view the status of your files in the working directory and staging area:

```
$ git status
```

or

```
$ git status -s
```

(-s shows a short one line version similar to svn)

- To see what is modified but unstaged:

```
$ git diff
```

- To see staged changes:

```
$ git diff --cached
```

Main Git commands

command	description
`git clone url [dir]`	copy a git repository so you can add to it
`git add files`	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help [command]`	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository

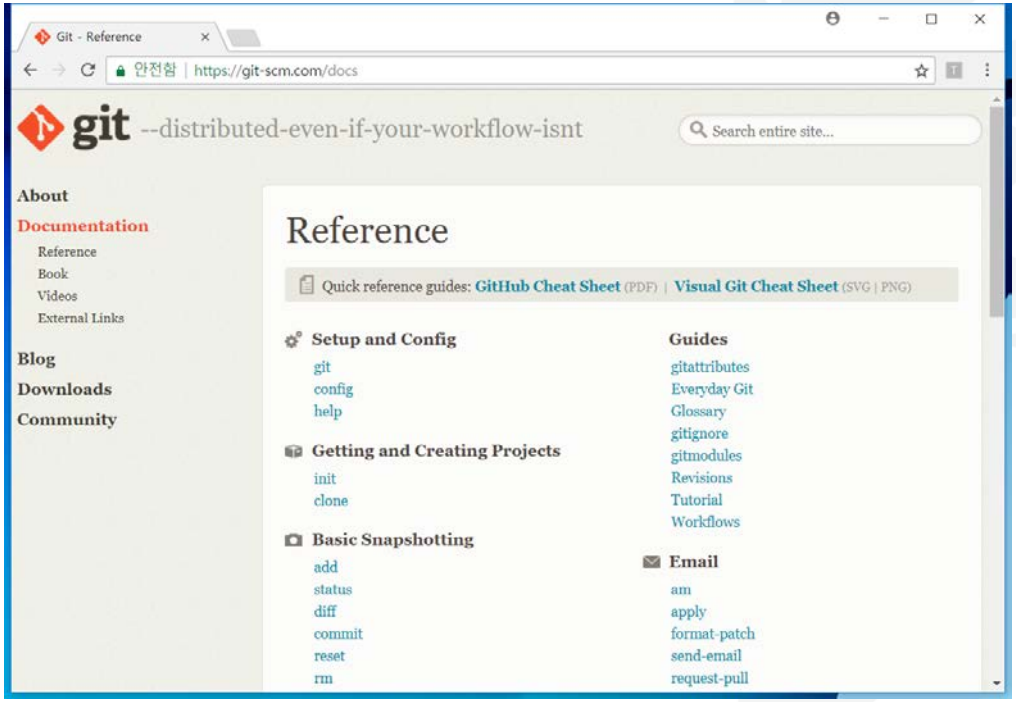
* others: **`init, reset, branch, checkout, merge, log, tag`**

Git commands for help

- At the command line:

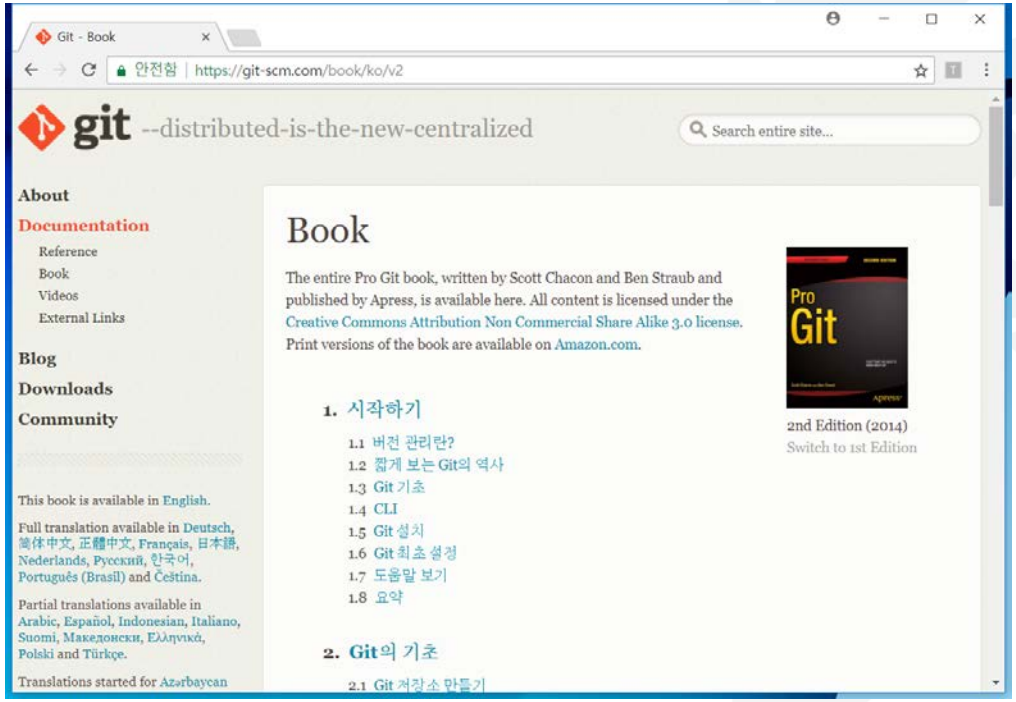
```
$git help <keyword>
$git <keyword> --help
$man git-<keyword>
```

References



- https://git-scm.com/docs

References



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