Pro Git(https://git-scm.com/book/ko/v2/)를 바탕으로 작성하였습니다.

YoonJoon Lee
SoC KAIST

Fall 2018

What we will take a look in this series

1. Getting Started
2. Git Basics
3. Git Branch
4. Git Server - GitLab

What we will take a look today

1.Getting Started

2. Git Basics

3. Git Branch

4. Git Server - GitLab

What I will talk about in this section

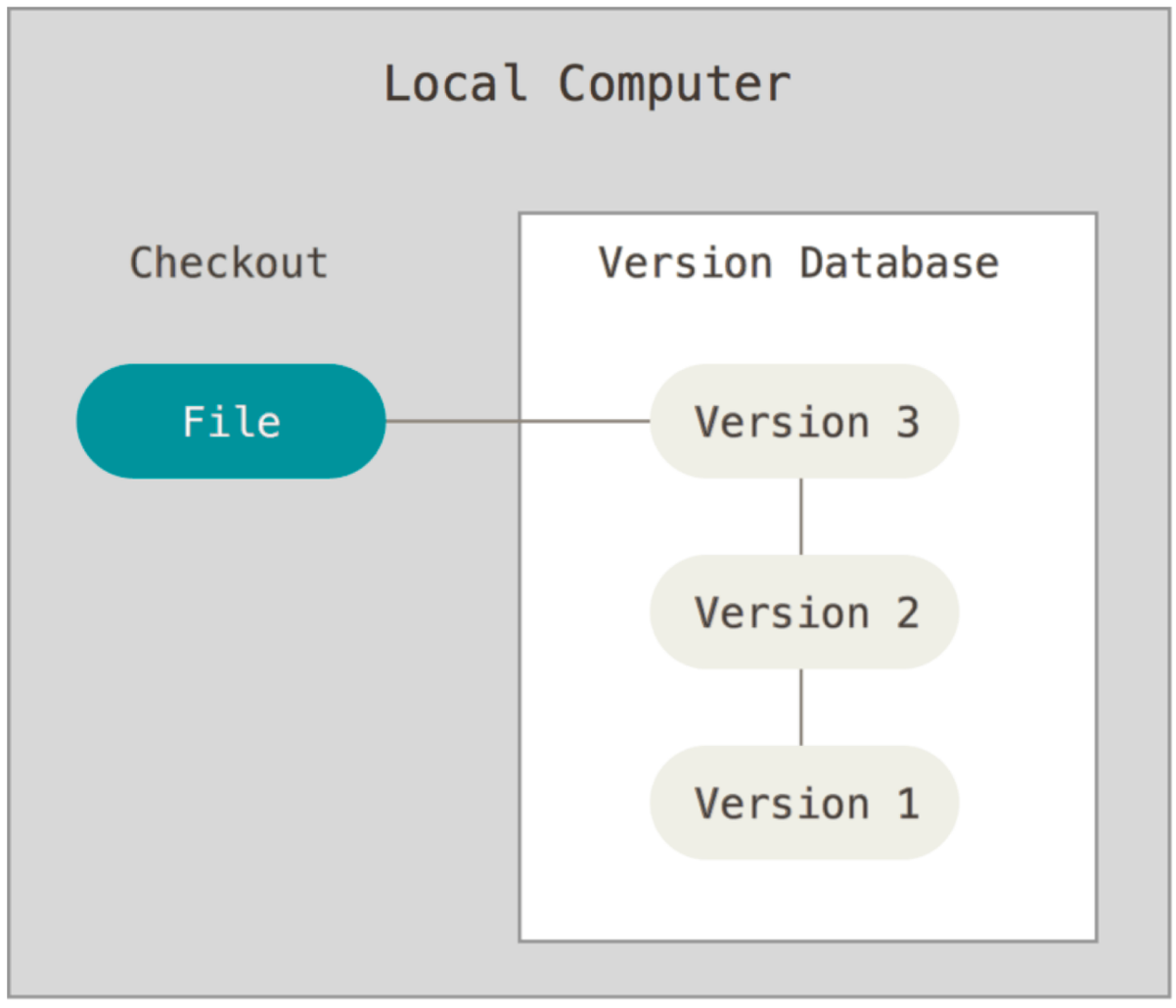
1. About Version Control?
2. A Short History
3. Git Basics
4. The Command Line
5. Installing Git
6. First-Time Git Setup

About Version Control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

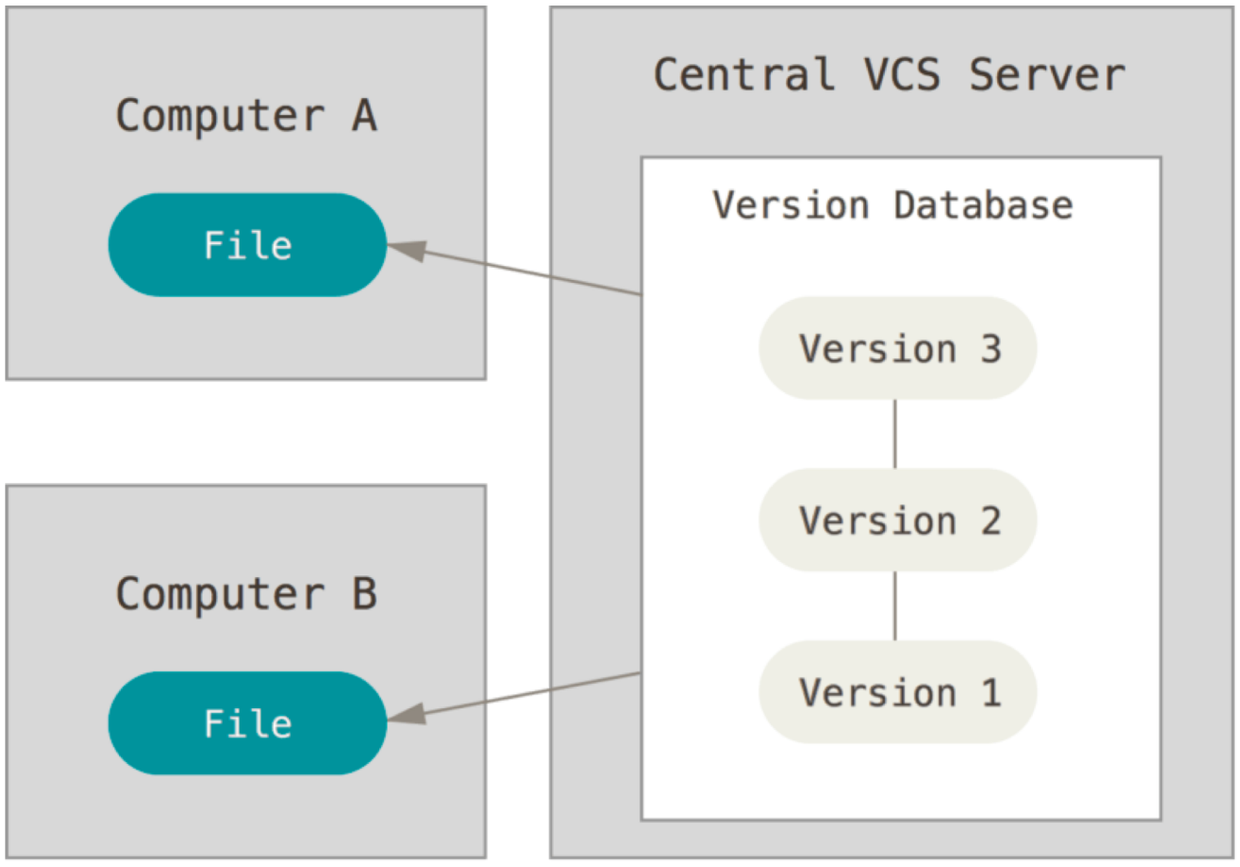
Local Version Control Systems

Local VCSs had a simple database that kept all the changes to files under revision control.



Centralized Version Control Systems

People need to collaborate with developers on other systems, Centralized Version Control Systems (CVCSs) such as CVS, Subversion, and Perforce.



Centralized Version Control Systems

Advantages, especially over local VCSs:

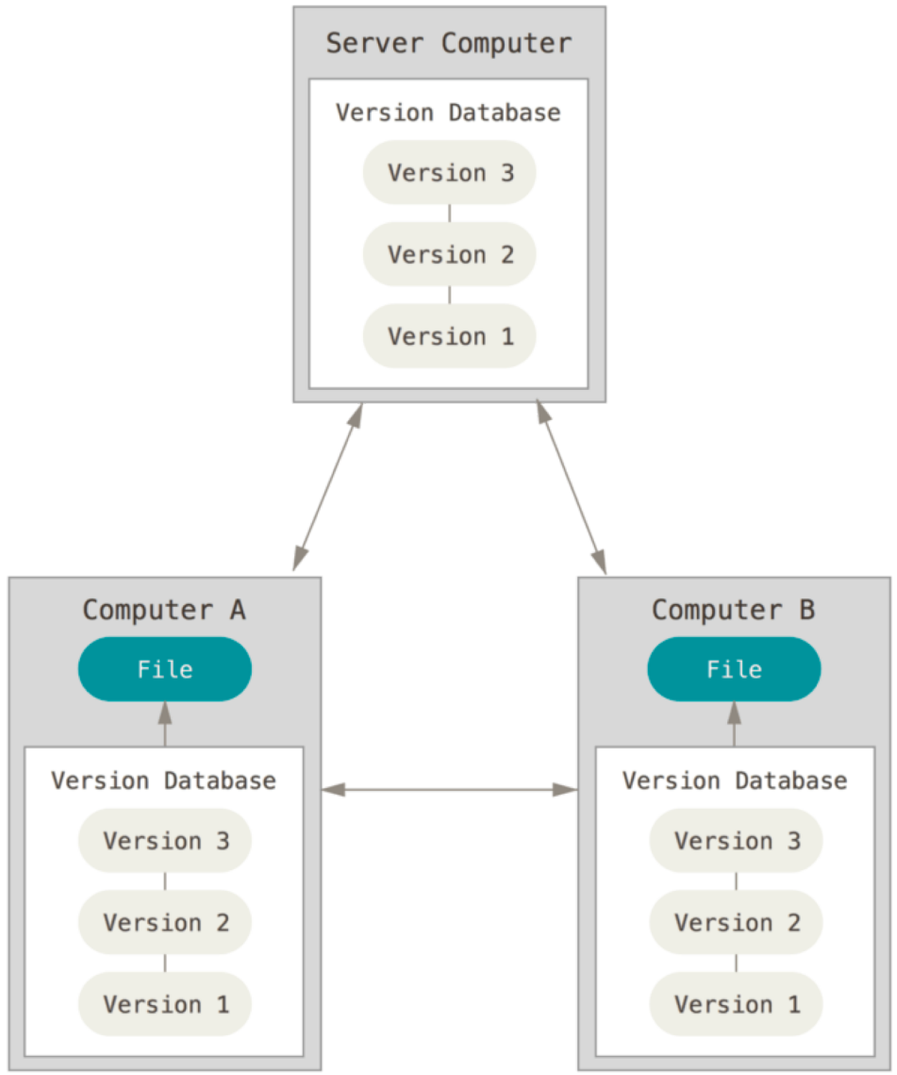
- What everyone else on the project is doing.
- Administrators have fine-grained control over who can do what.
- It's far easier to administer a CVCS than it is to deal with local databases on every client.

Some serious downsides:

- The single point of failure.

Distributed Version Control Systems

In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.



A Short History of Git

- In 2002, the Linux kernel project began using a proprietary DVCS called BitKeeper.
- In 2005, the tool's free-of-charge status was revoked.
- This prompted the Linux development community to develop their own tool based on some of the lessons they learned while using BitKeeper.
- Since its birth in 2005, Git has evolved and matured to be easy to use and yet retain these initial qualities.

Goals of Git

Goals of the new system:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

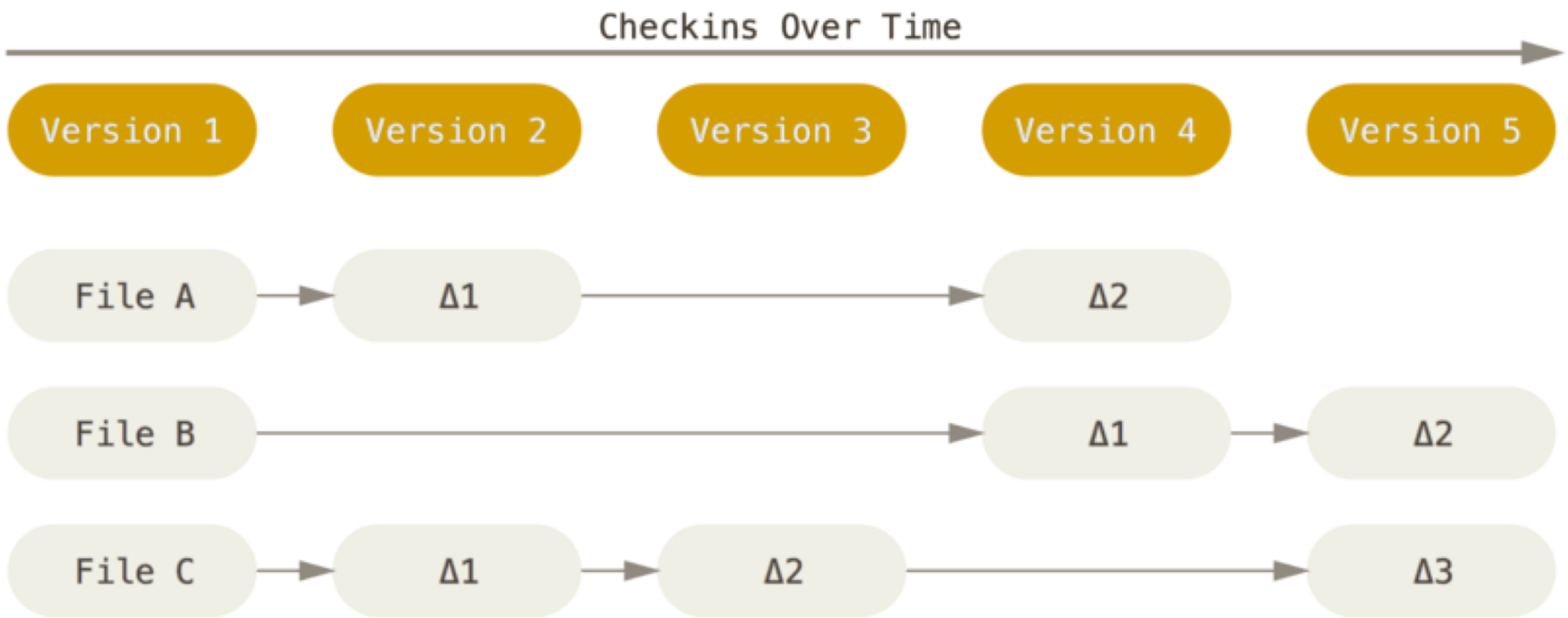
Git Basics

- A nutshell?

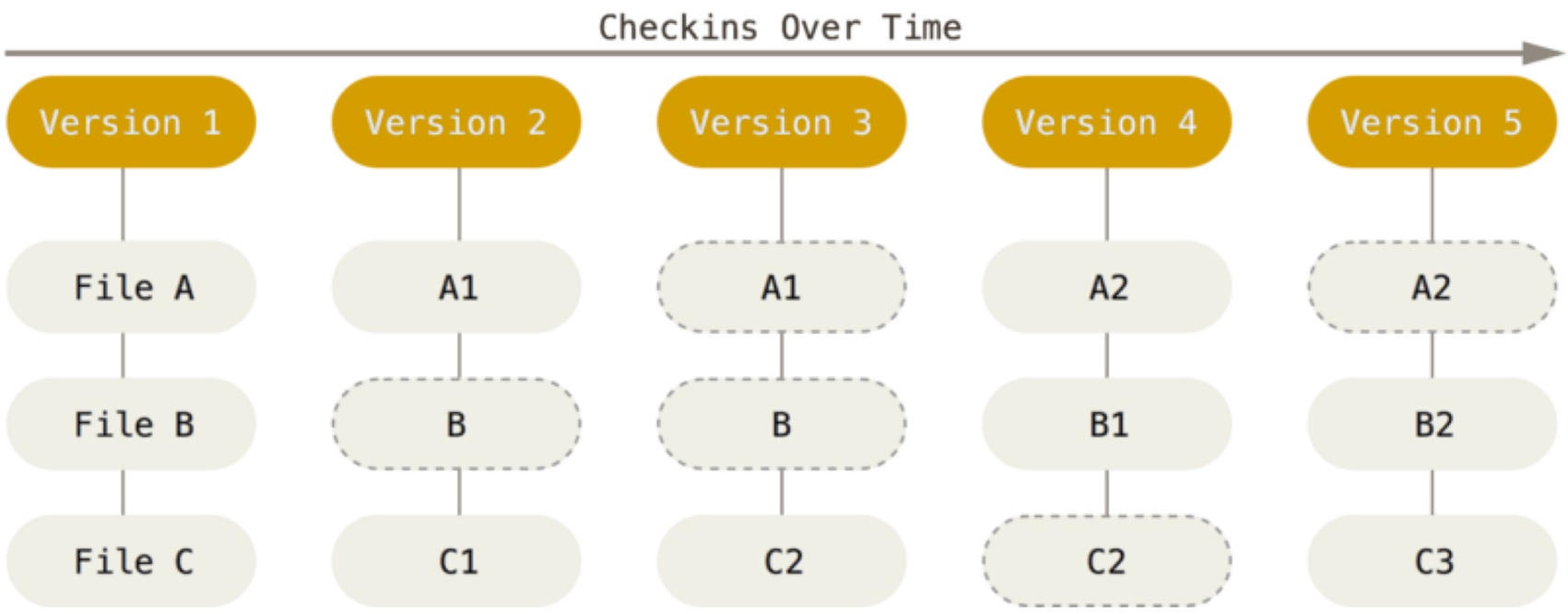
- Try to clear your mind of the things you may know about other VCSs, such as CVS, Subversion or Perforce

Snapshots, Not Differences

The other systems such as CVS, Subversion, Perforce, Bazaar, and so on think of the information they store as a set of files and the changes made to each file over time (this is commonly described as ***delta-based***version control).



- Git thinks of its data more like a series of snapshots of a miniature filesystem.
- Every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
- Git thinks about its data more like a **stream of snapshots**.



Nearly Every Operation Is Local

- Most operations in Git need only local files and resources to operate — generally no information is needed from another computer on your network.
- There is very little you can't do if you're offline or off VPN.

Git Has Integrity

- Everything in Git is check-summed before it is stored and is then referred to by that checksum.
- It's impossible to change the contents of any file or directory without Git knowing about it.
- The mechanism that Git uses for this checksumming is called a SHA-1 hash.
- A 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git.
- A SHA-1 hash looks something like this:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

Git Generally Only Adds Data

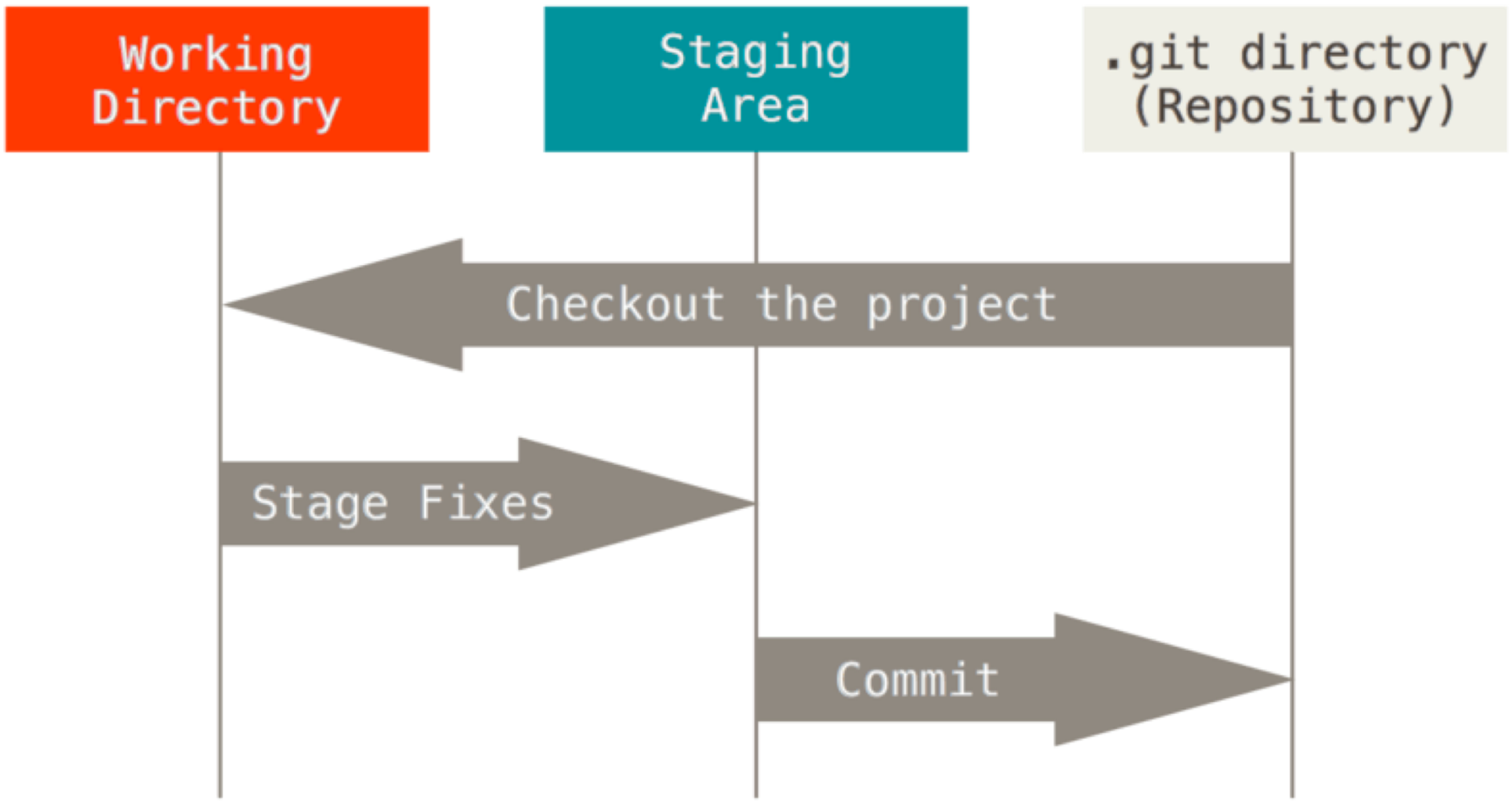
- When you do actions in Git, nearly all of them only *add* data to the Git database.
- It is hard to get the system to do anything that is not undoable or to make it erase data in any way.
- This makes using Git a joy because we know we can experiment without the danger of severely screwing things up.

The Three States

Git has three main states that your files can reside in:

- *Committed*: the data is safely stored in your local database
- *Modified:* you have changed the file but have not committed it to your database yet
- *staged*: you have marked a modified file in its current version to go into your next commit snapshot.

The three main sections of a Git project: the working directory, the staging area, and the Git directory, .



- The working tree(directory) is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.
- The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit. Its technical name in Git parlance is the “index”, but the phrase “staging area” works just as well.
- The Git directory(`.git`) is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you *clone* a repository from another computer.

The basic Git workflow

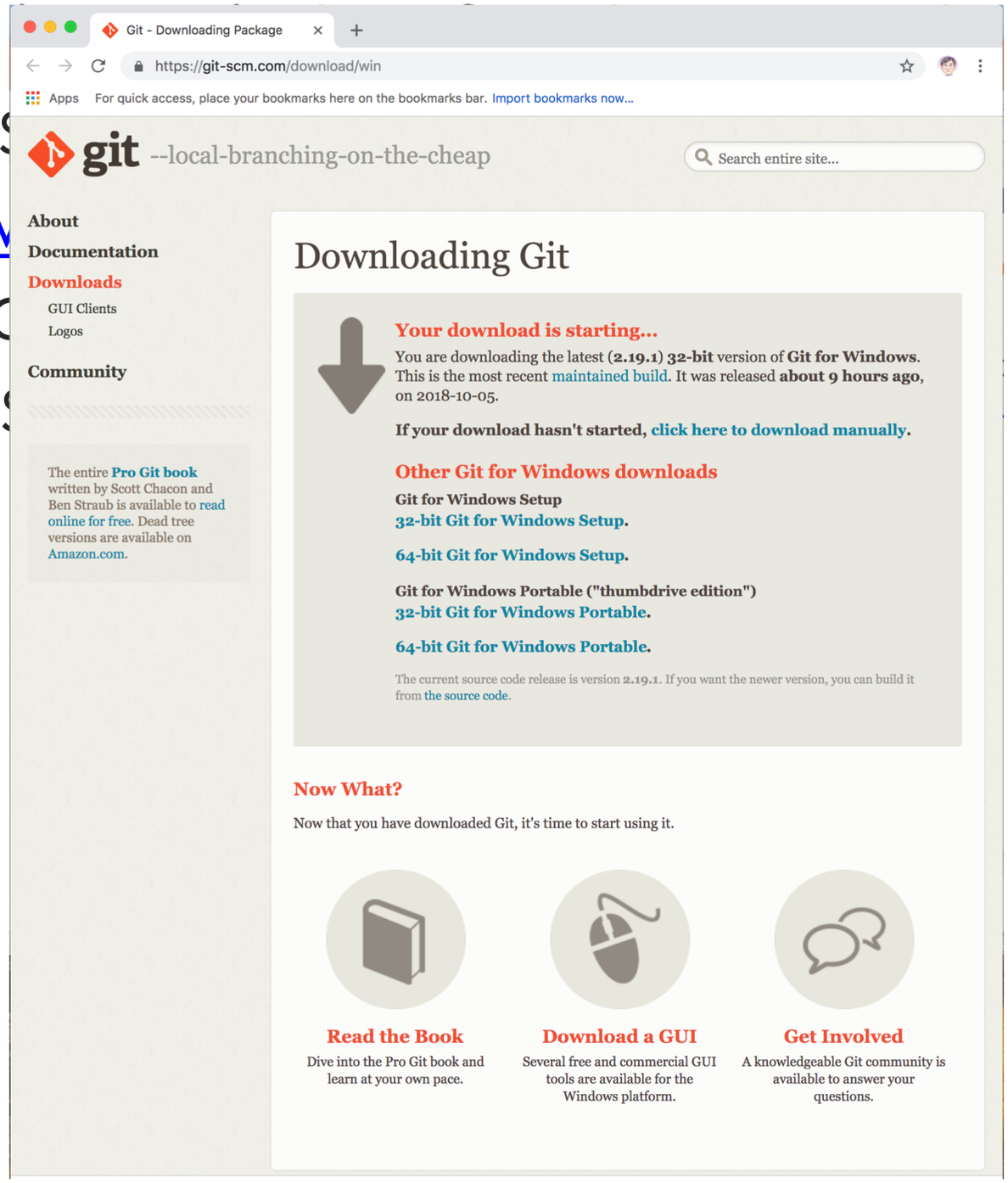
1. You modify files in your working tree (**modified**).
2. You selectively stage just those changes you want to be part of your next commit, which adds *only* those changes to the staging area(**staged**).
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory(**committed**).

The Command Line

- There are a lot of different ways to use Git. There are the original command-line tools, and there are many graphical user interfaces of varying capabilities.
- We will be using Git on the command line. the command line is the only place you can run *all* Git commands
- If you know how to run the command-line version, you can probably also figure out how to run the GUI version.

Installing Git: Installing on Windows

- The most official build
on the Git website. Jus
scm.com/download/v
start automatically. No
called Git for Windows

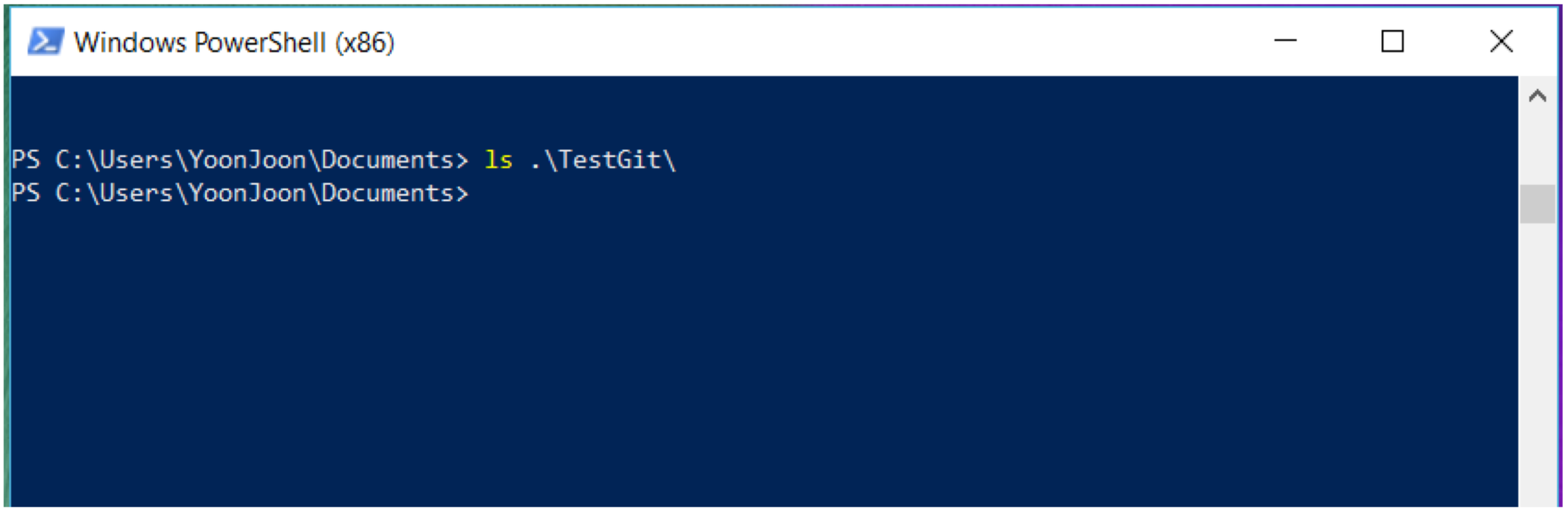


First-Time Git Setup

You need to do a few things to customize your Git environment.

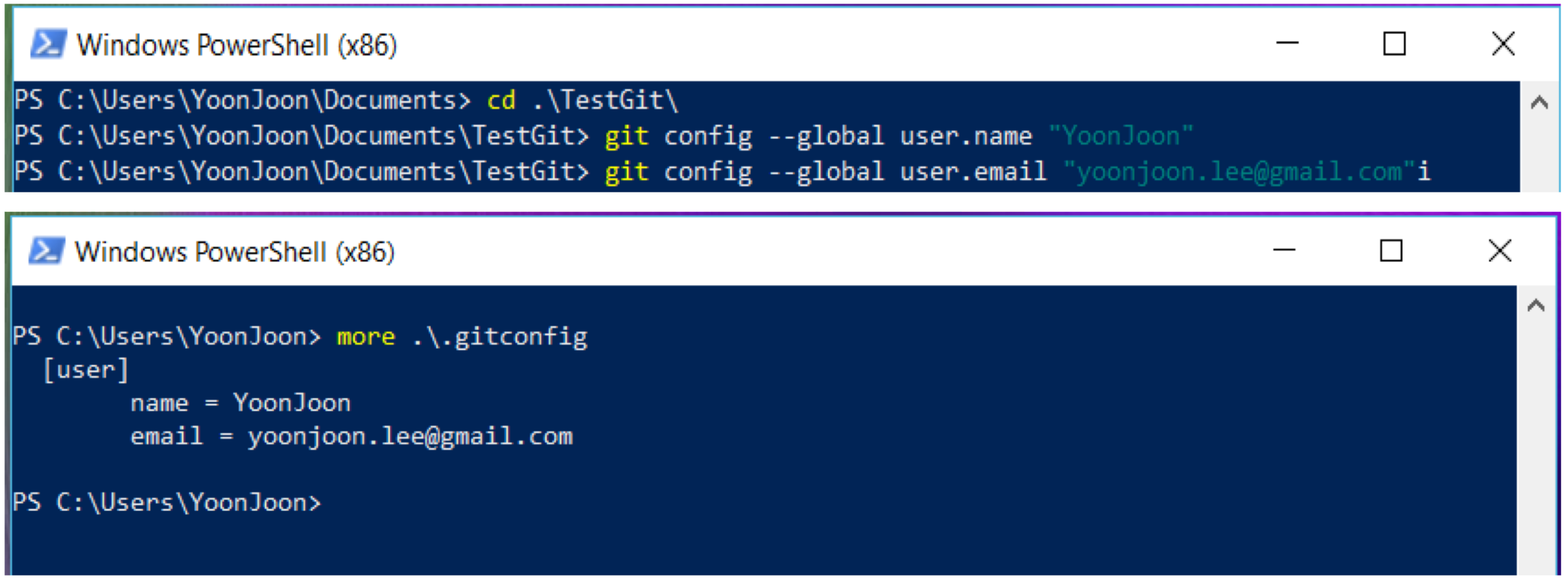
Git comes with a tool called git config that lets you get and set configuration variables that control all aspects of how Git looks and operates.

1. `C:\etc\gitconfig` file: Contains values applied to every user on the system and all their repositories. If you pass the option --system to git config, it reads and writes from this file specifically. (Because this is a system configuration file, you would need administrative or superuser privilege to make changes to it.)
2. `C:/Users/$USER/.gitconfig` file: Values specific personally to you, the user. You can make Git read and write to this file specifically by passing the --global option, and this affects *all* of the repositories you work with on your system.
3. config file in the Git directory (that is, `.git\config`) of whatever repository you're currently using: Specific to that single repository. You can force Git to read from and write to this file with the --local option, but that is in fact the default. (Unsurprisingly, you need to be located somewhere in a Git repository for this option to work properly.)


Windows PowerShell (x86)
PS C:\Users\YoonJoon\Documents> ls .\TestGit\
PS C:\Users\YoonJoon\Documents>

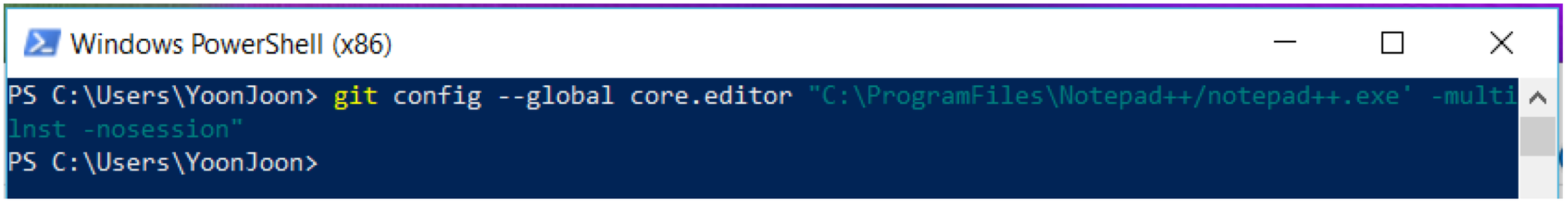
Your Identity

The first thing you should do when you install Git is to set your user name and email address.



Your Editor

On a Windows system, if you want to use a different text editor, you must specify the full path to its executable file.

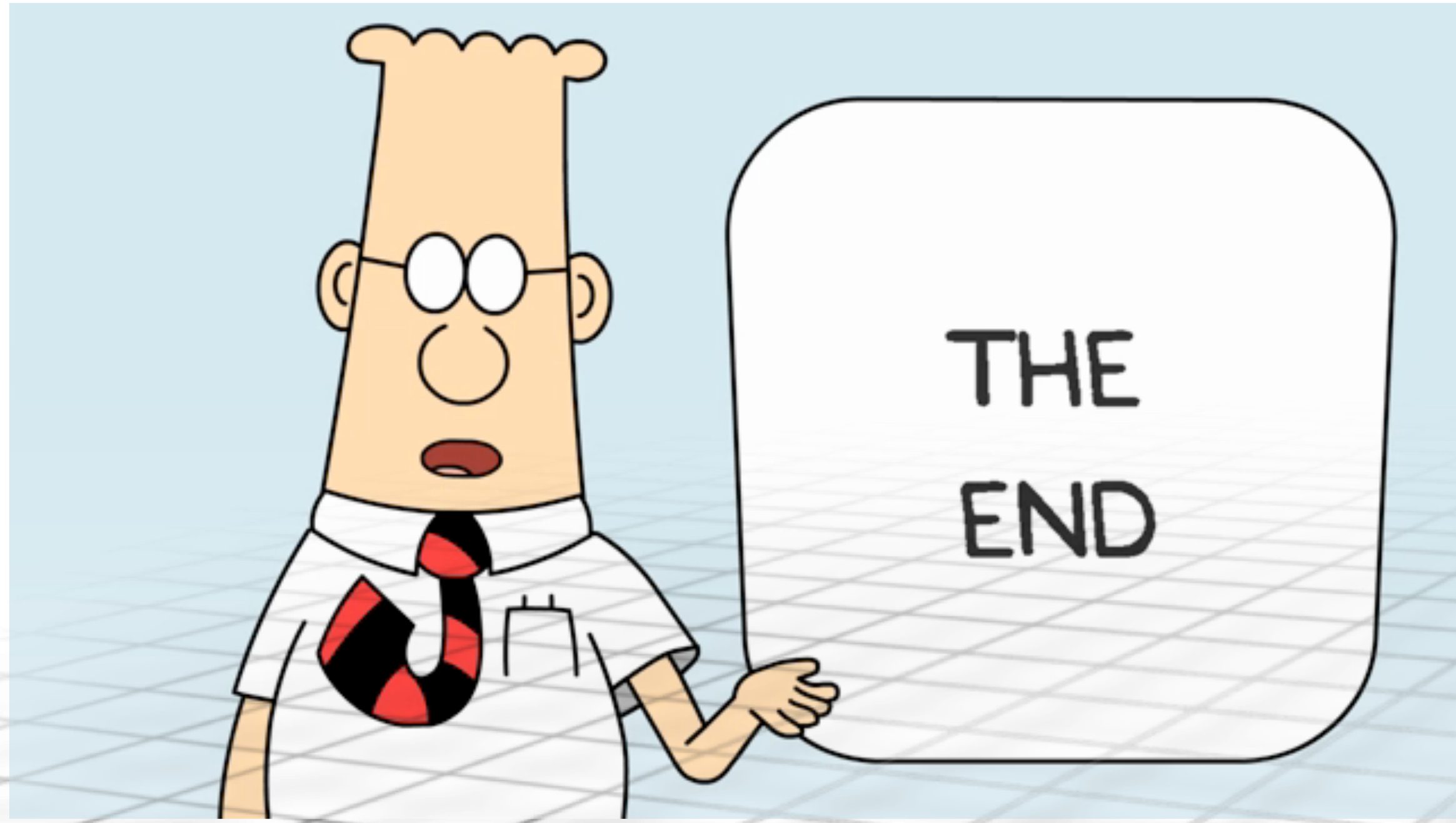


Checking Your Settings

Check your configuration settings

Windows PowerShell (x86)

```
PS C:\Users\YoonJoon> git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.name=YoonJoon
user.email=yoonjoon.lee@gmail.com
core.editor=C:\ProgramFiles\Notepad++/notepad++.exe' -multilnst -nosession
PS C:\Users\YoonJoon>
```



출처: metachannels.com

감사합니다