Introduction to
open-Source Software (OSS)

Concepts, strategies, and methodologies
related to open-source software development

Week 04 – Lecture 07

Jamil Hussain
jamil@sejong.ac.kr
010-6252-8807

Office: 421, Innovation Center
Sejong University

Recap

- Getting start the OSS project
- Ingredients for starting new project
  - Naming and branding your project

FOSS project key items list

- ✓Naming and branding your project
- ✓Have a clear mission statement
- ✓State that the project is free
- ✓Features and requirements list
- ✓Development status
- ✓Downloads
- ✓Version control and bug tracker access
- ✓Communications channels
- ✓Developer guidelines
- ✓Documentation
- ✓Choosing a license and applying it

Today, Agenda

- Ingredients for starting new project
- Version control
  - Version control Type
  - Version control Systems(VCS)
- Git
  - Git workflow
  - Installation

Downloads

- The software should be downloadable as source code in standard formats
- When a project is first getting started, binary (executable) packages are not necessary, unless the software has such complicated build requirements or dependencies that merely getting it to run would be a lot of work for most people.
- (But if this is the case, the project is going to have a hard time attracting developers anyway!)

Version Control and Bug Tracker Access

- Downloading source packages is fine for those who just want to install and use the software, but it's not enough for those who want to debug or add new features.
- People need real-time access to the latest sources, and a way to submit changes based on those sources.
- The solution is to use a version control system
  - Specifically, an online, publicly-accessible version controlled repository, from which anyone can check out the project's materials and subsequently get updates.

Communications Channels

- Visitors usually want to know how to reach the human beings involved with the project
- Provide the addresses of mailing lists, chat rooms, IRC channels and any other forums where others involved with the software can be reached.
- Make it clear that you and the other authors of the project are subscribed to these mailing lists, so people see there's a way to give feedback that will reach the developers.
- Your presence on the lists does not imply a commitment to answer all questions or implement all feature requests.

Developer Guidelines

- If someone is considering contributing to the project, she'll look for developer guidelines.
- Developer guidelines are not so much technical as social:
  - they explain how the developers interact with each other and with the users, and ultimately how things get done.

Documentation

- Documentation is essential. There needs to be something for people to read, even if it's rudimentary and incomplete.
- Documentation should be available from two places: online (directly from the web site), and in the downloadable distribution of the software
- For online documentation, make sure that there is a link that brings up the entire documentation in one HTML page
  - (put a note like "monolithic" or "all-in-one" or "single large page" next to the link, so people know that it might take a while to load).
  - This is useful because people often want to search for a specific word or phrase across the entire documentation.

Documentation

- But this is not necessarily the most common way documentation is accessed.
- Often, someone who is basically familiar with the software is coming back to search for a specific word or phrase, and to fail to provide them with a single, searchable document would only make their lives harder.

Read *the* Docs

Hosting

- Where on the Internet should you put the project's materials?
- A web site, obviously — but the full answer is a little more complicated than that.

bluehost HostGator inmotion hosting

GoDaddy TSOHOST HOSTINGER

SiteGround Hostwinds WiX

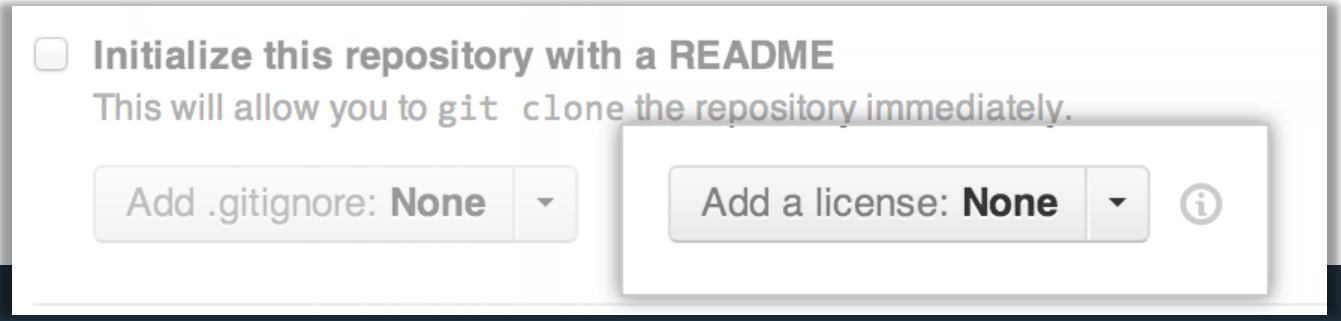
Previous web hosting experience ?

A Yes

B No

Choosing a License and Applying It

- An open source license guarantees that others can use, copy, modify, and contribute back to your project without repercussions. It also protects you from sticky legal situations. You must include a license when you launch an open source project.
- MIT, Apache 2.0, and GPLv3 are the most popular open source licenses, but there are other options to choose from.
- When you create a new project on GitHub, you are given the option to select a license. Including an open source license will make your GitHub project open source.

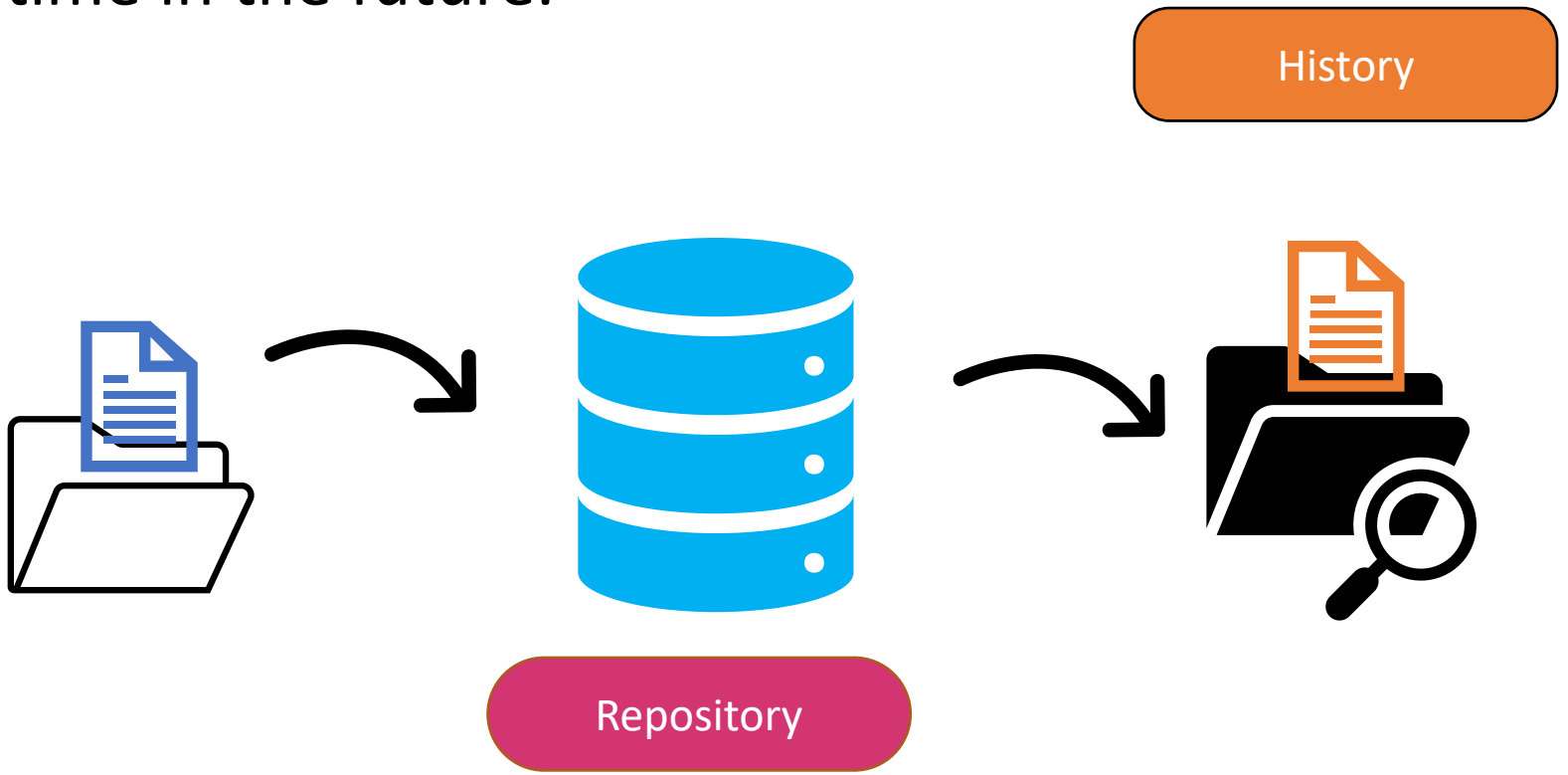


What is Version Control?

- Your Daily Tasks
  - Creating things
  - Save things
  - Edit things
  - Save the thing again and again

What is Version Control?

- That saving the thing again and again is the goal and where version control helps, providing you clarity as to when you did it, why you did it, and what the contents of the change were, open for review at any time in the future.



What is Version Control?

John's code

V1.0.0

V1.0.1

V1.0.2

Amy's code

V1.0.0

V1.0.1

V1.0.2

What is Version Control?

John's code

V1.0.2

Amy's code

V1.0.2

What is Version Control?

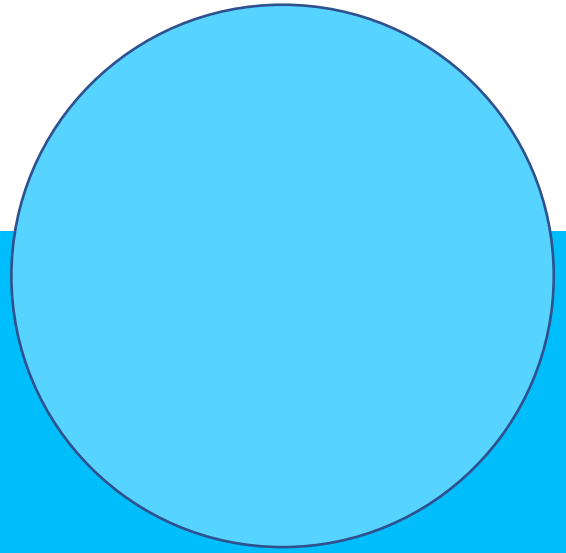
John's code

Amy's code

What is Version Control?

John's code

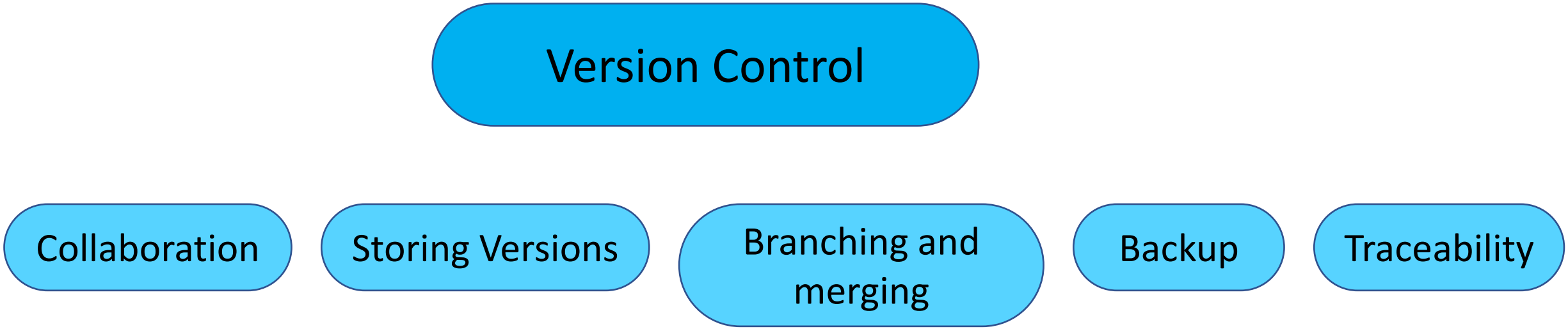
Amy's code



Version Control System (VCS)

- **A *version control system*** is a system that records changes to a set of one or more files so that earlier versions of any of these files can be restored at a later time.
- It is often used for managing software, but it can be used for any types of files, such as documentation, web pages, graphics, artwork in general, and chapters of a book.

Benefits of Version Control Systems



- Which helps us to know which change was made when and by who made it
- If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Benefits of Version Control Systems

- **Long-term change history**
  - The changes made by developers, including the creating, modification, and deletion of files over the years, can be seen in history. It will allow going back to the previous version for analyzing bugs and fixing problems.
- **Branching and merging**
  - Branching helps work in an independent manner and not interfere with each other's work. Merging brings the works together and allows seeing if there are conflicts between those works.
- **Traceability**
  - Ability to trace each change and connect it to project management and bug tracking software, as well as to annotate each change with a message describing the purpose of the change.

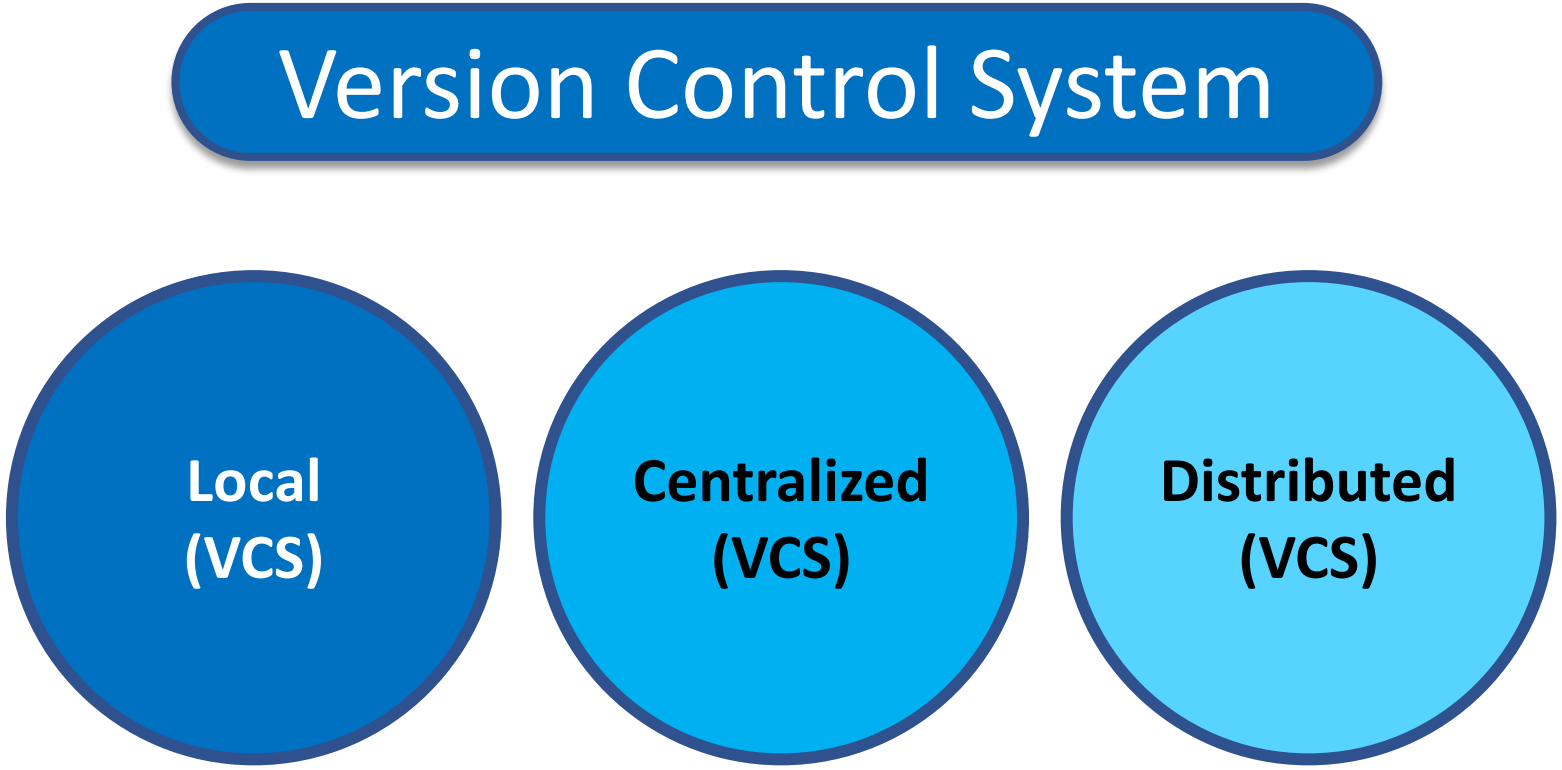
Version Control System (VCS)

https://students.iitj.ac.in/blog/2018/05/29/up-and-running-with-version-control-systems/

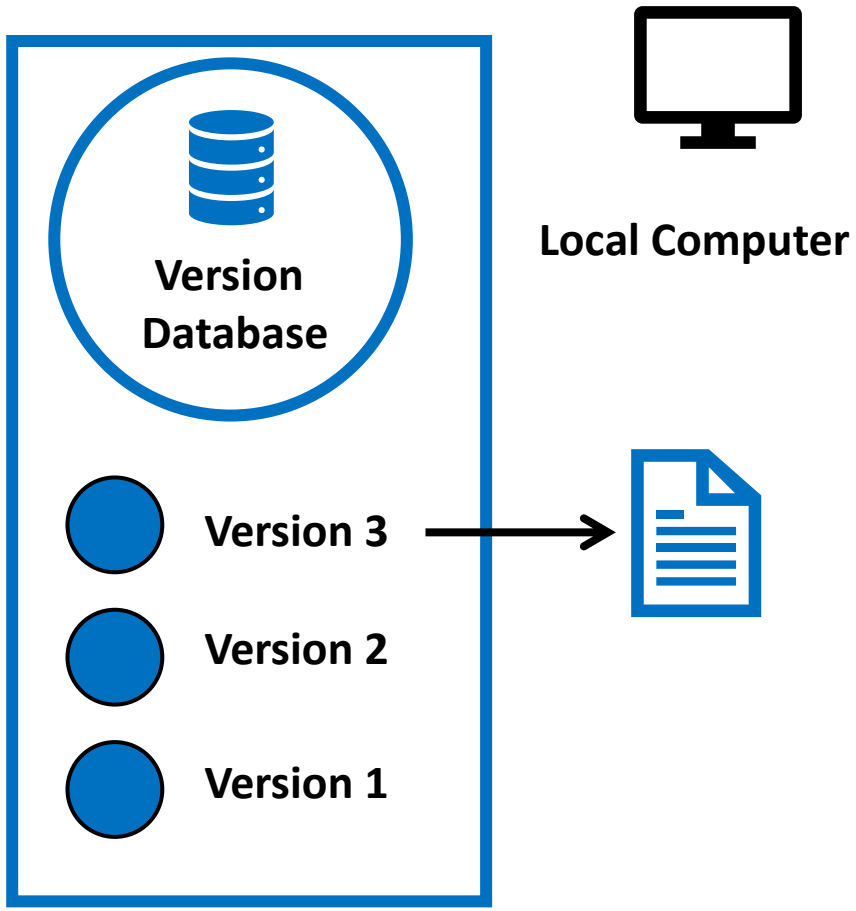
Why do we need version control system?

- Software is a precious asset. You spend hours working on it and need to make sure you do not lose important work.
- Sometimes you make mistakes, overwrite things, replace good ideas with bad ones, and so on.
- Version control allows us to safely go back to different versions.
- Also, when groups of people work on the same project files, a version control system helps prevent lost or conflicting work.
- It tracks every individual change by each contributor.

Version Control System Type

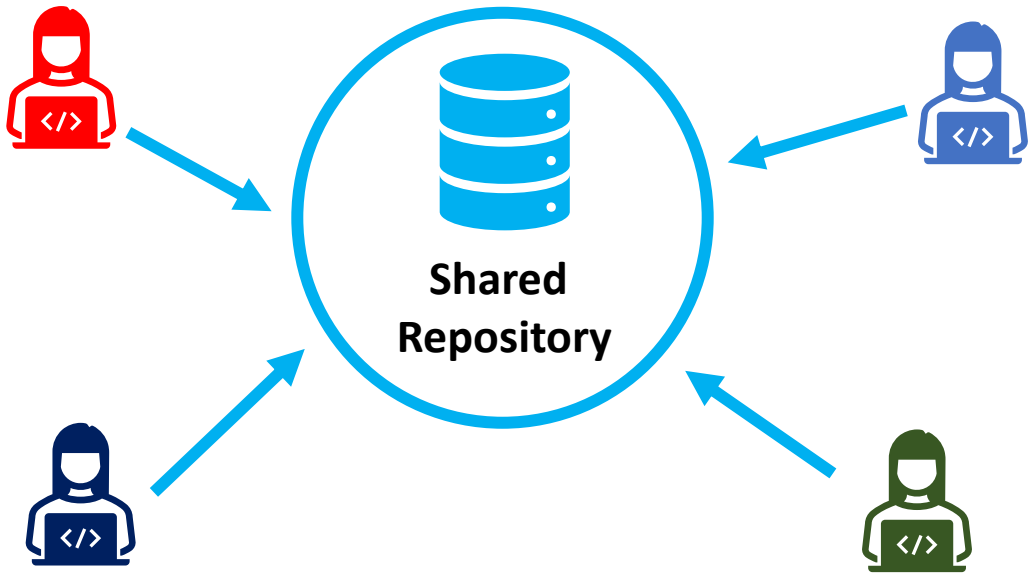


Local Version Control Systems



- This approach is very common because it is so simple, but it is also incredibly error prone.
- It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.
- To deal with this issue, programmers long ago developed local VCSs that had a simple database that kept all the changes to files under revision control.
- Just a Local Database
  - RCS (https://www.gnu.org/software/rcs/)

Centralized Version Control Systems



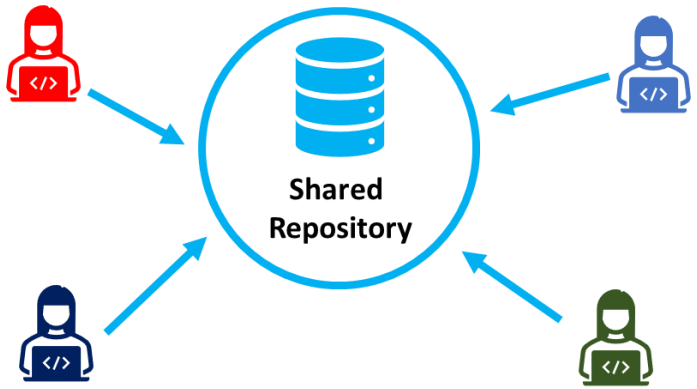
Subversion

Team Foundation Server

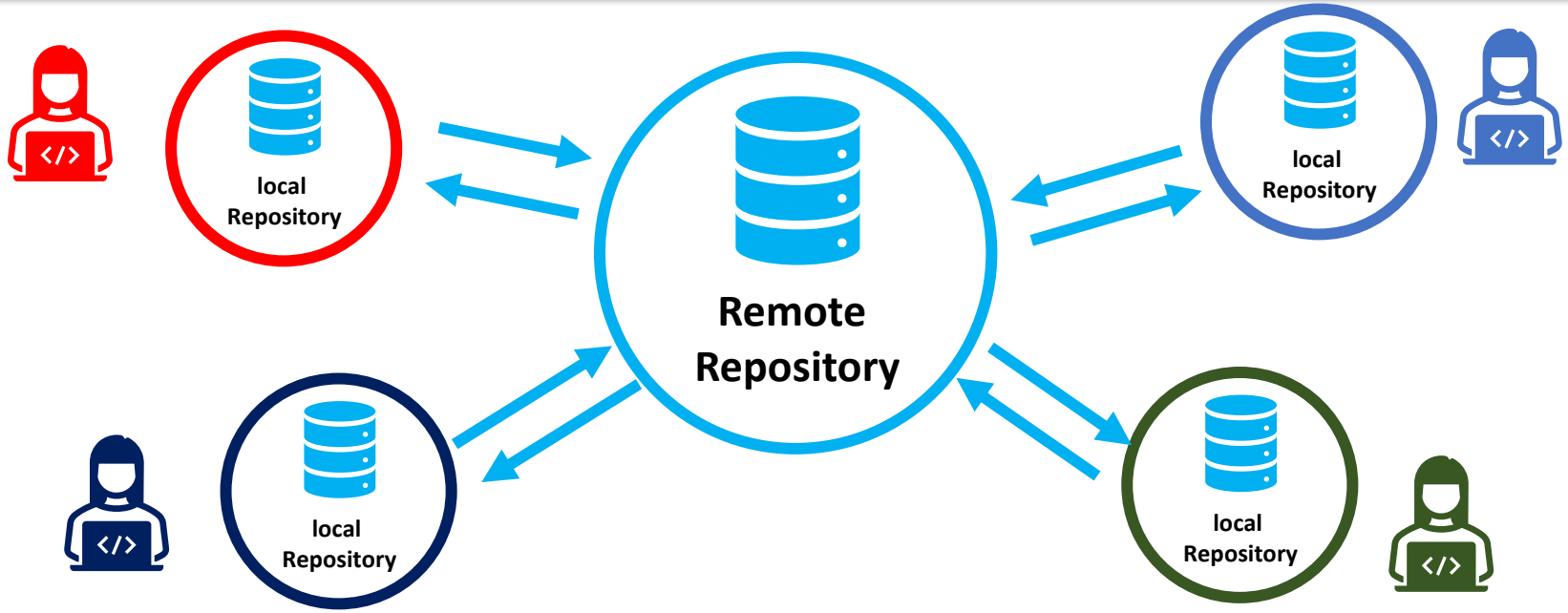
- The next major issue that people encounter is that they need to collaborate with developers on other systems ,
- To deal with this problem, Centralized Version Control Systems (CVCSs) were developed.

Centralized Version Control Systems

- However, this setup also has some serious downsides.
- The most obvious is the **single point of failure** that the centralized server represents. If that server **goes down** for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on.
- If the **hard disk the central database is on becomes corrupted**, and proper backups haven't been kept, you **lose absolutely everything** — the entire history of the project except whatever single snapshots people happen to have on their local machines.



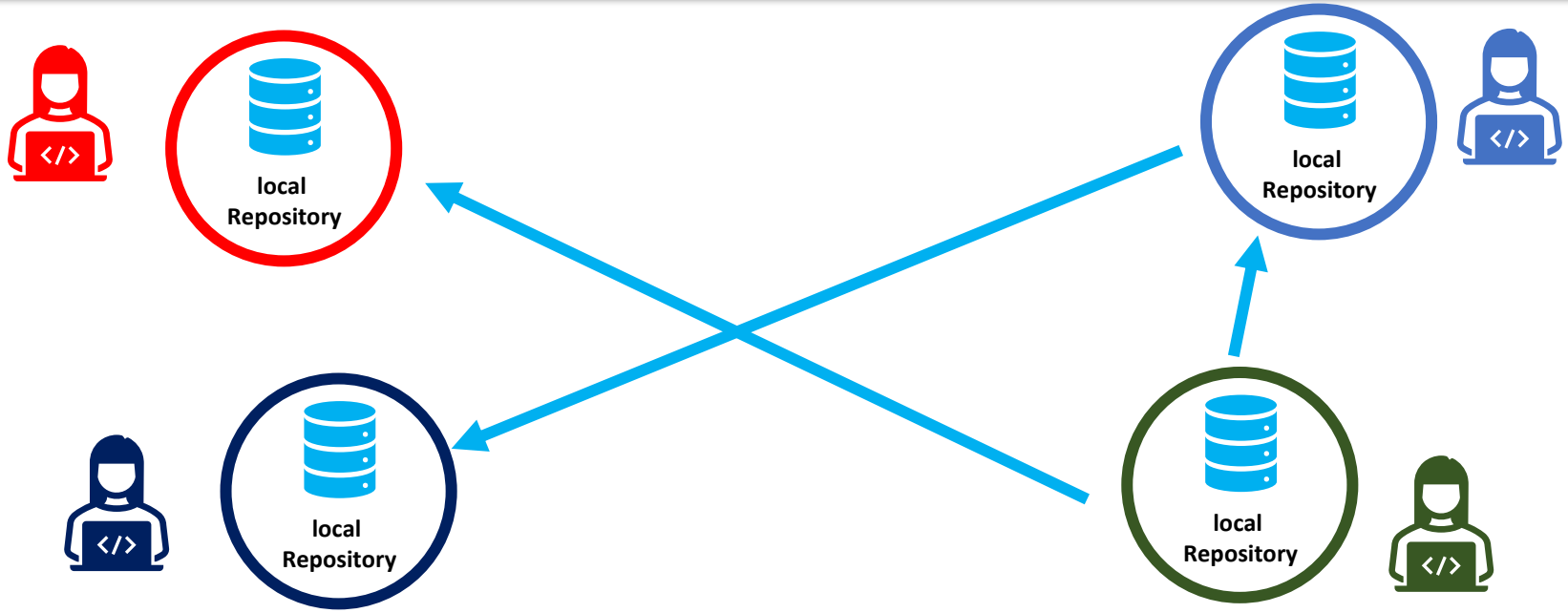
Distributed Version Control Systems (DVCSs)



- Git
- Mercurial
- Bazaar or Darcs

- In DVCS, clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.
- Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it.

Distributed Version Control Systems (DVCSs)



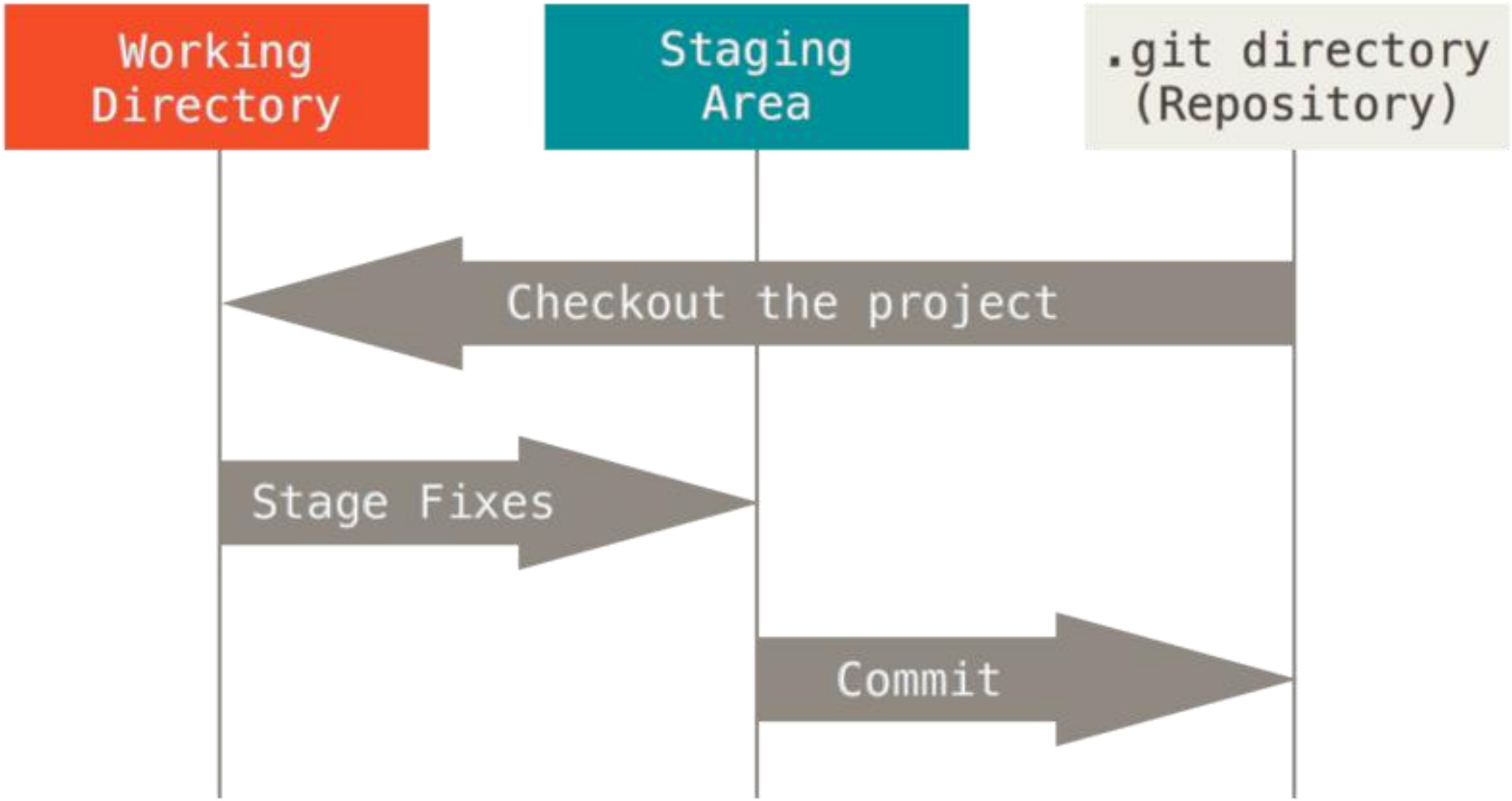
- Git
- Mercurial
- Bazaar or Darcs

- In DVCS, clients don’t just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history.
- Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it.

Why Git

- **Free and Open Source**
- **Git** is a fast and modern implementation of Version control
- Git provide a **history** of content changes
- Git facilitates **collaborative changes** to files
- Git is easy to use for any type of **knowledge worker**
- Fully distributed

How Git Works

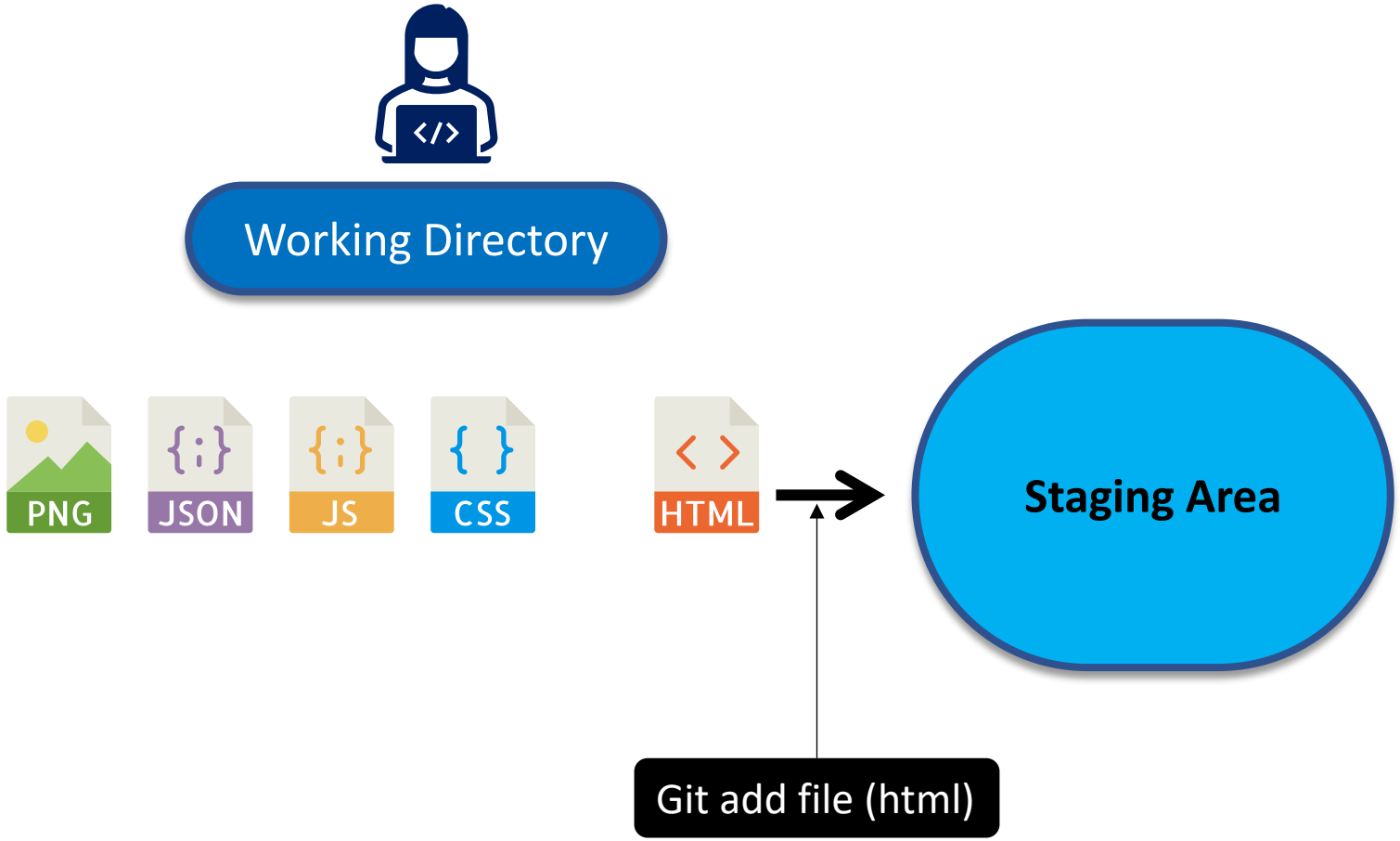


How Git Works

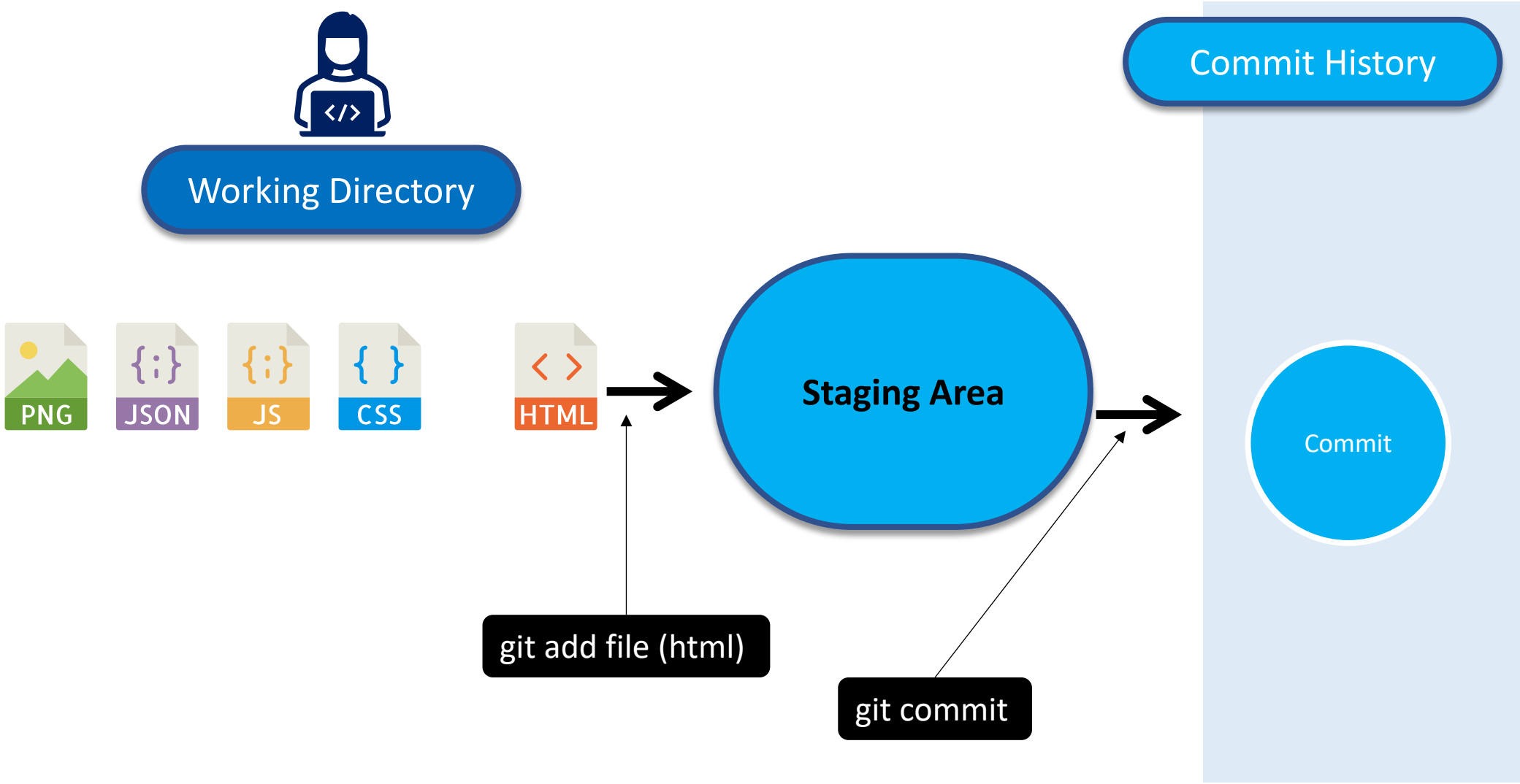
How Git Works


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
</body>
</html>
```

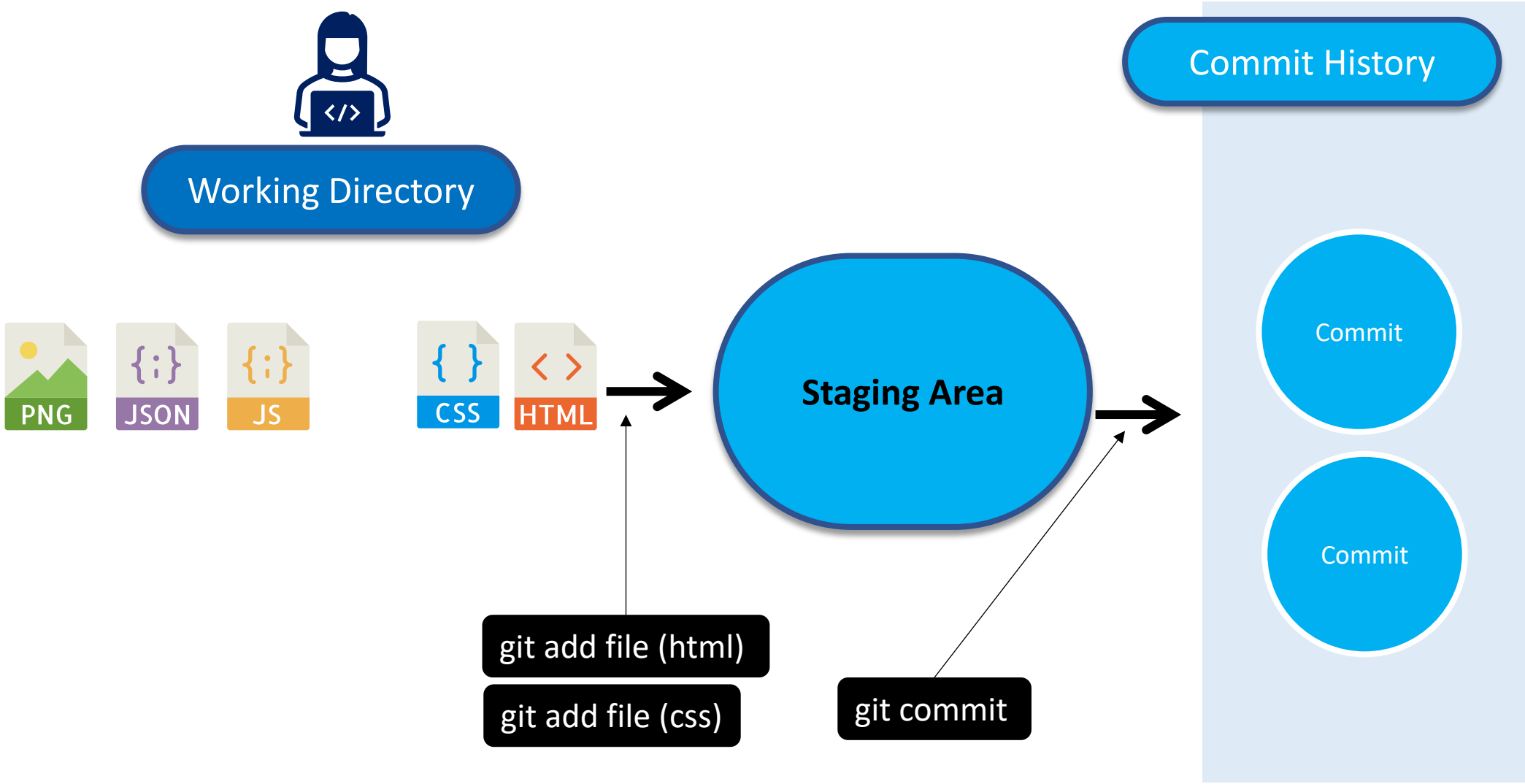
How Git Works



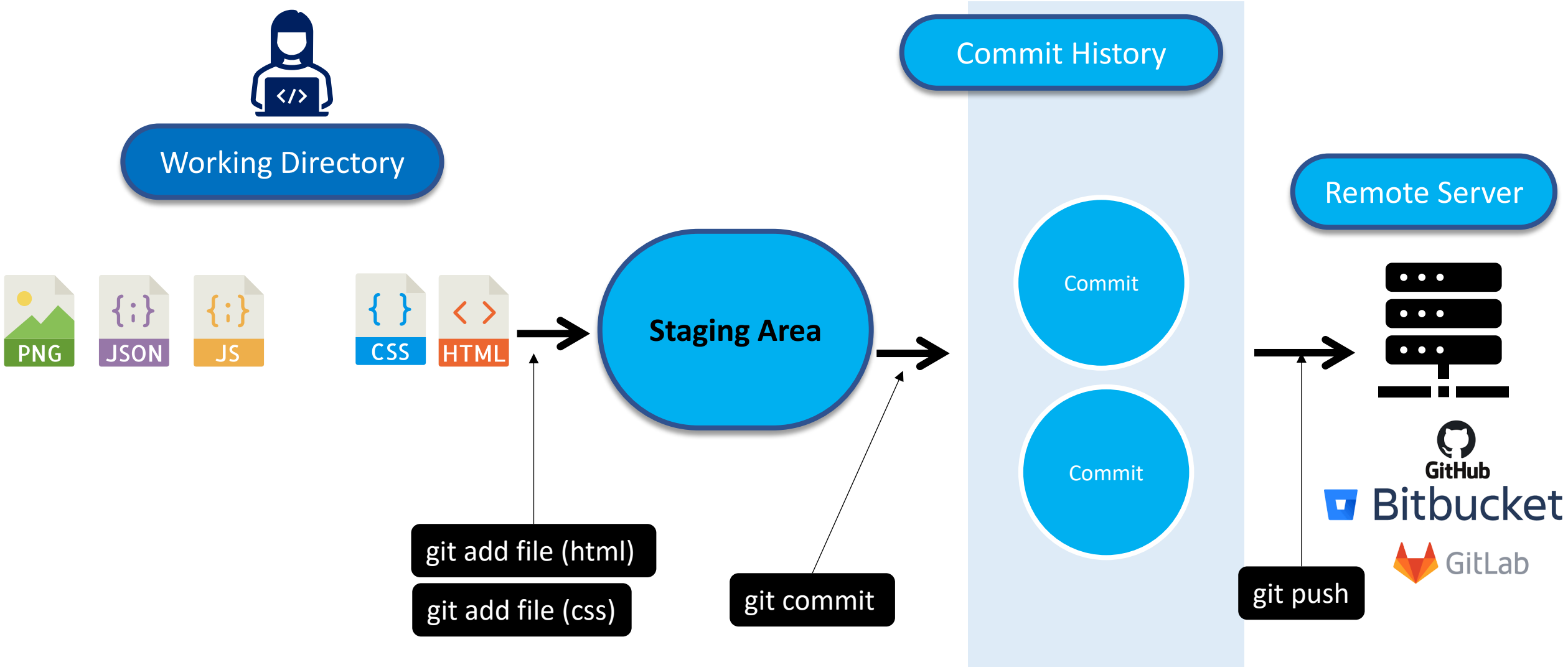
How Git Works



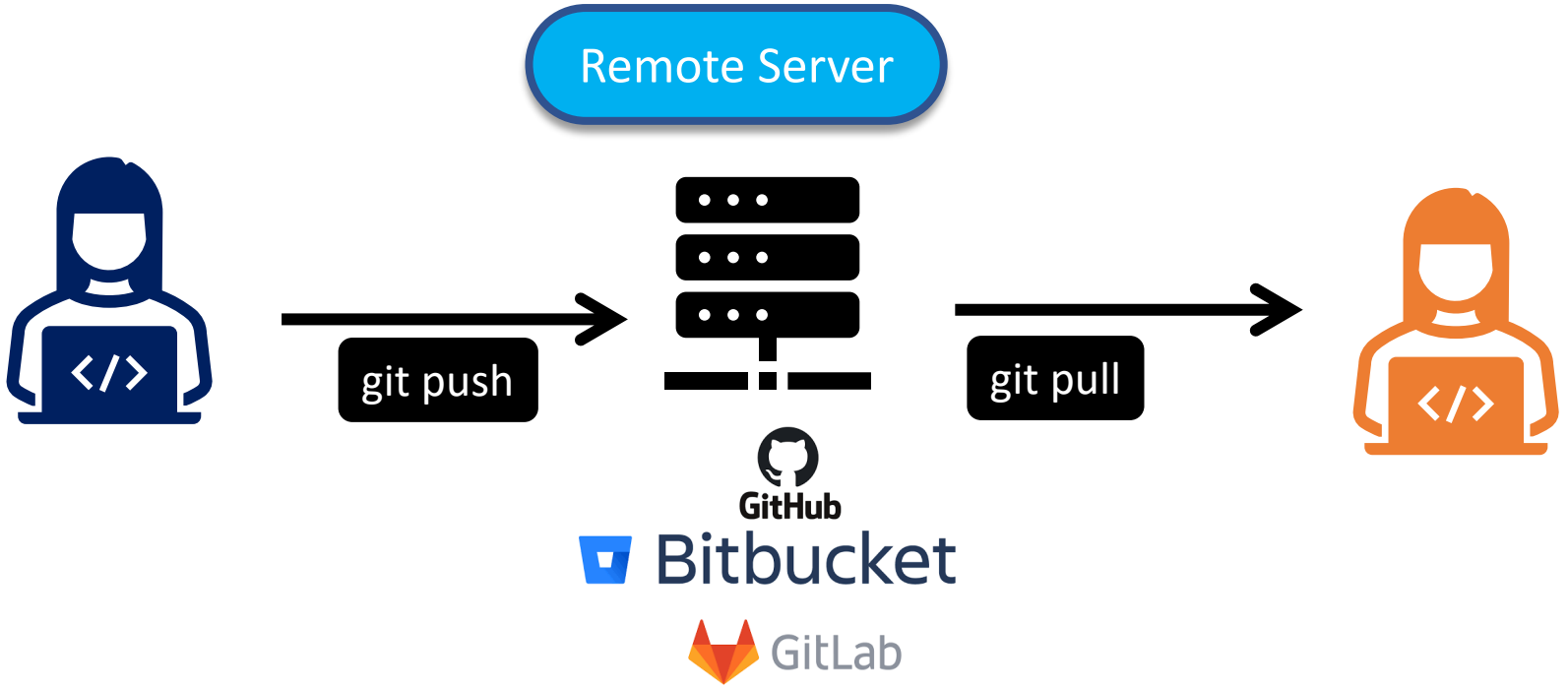
How Git Works



How Git Works

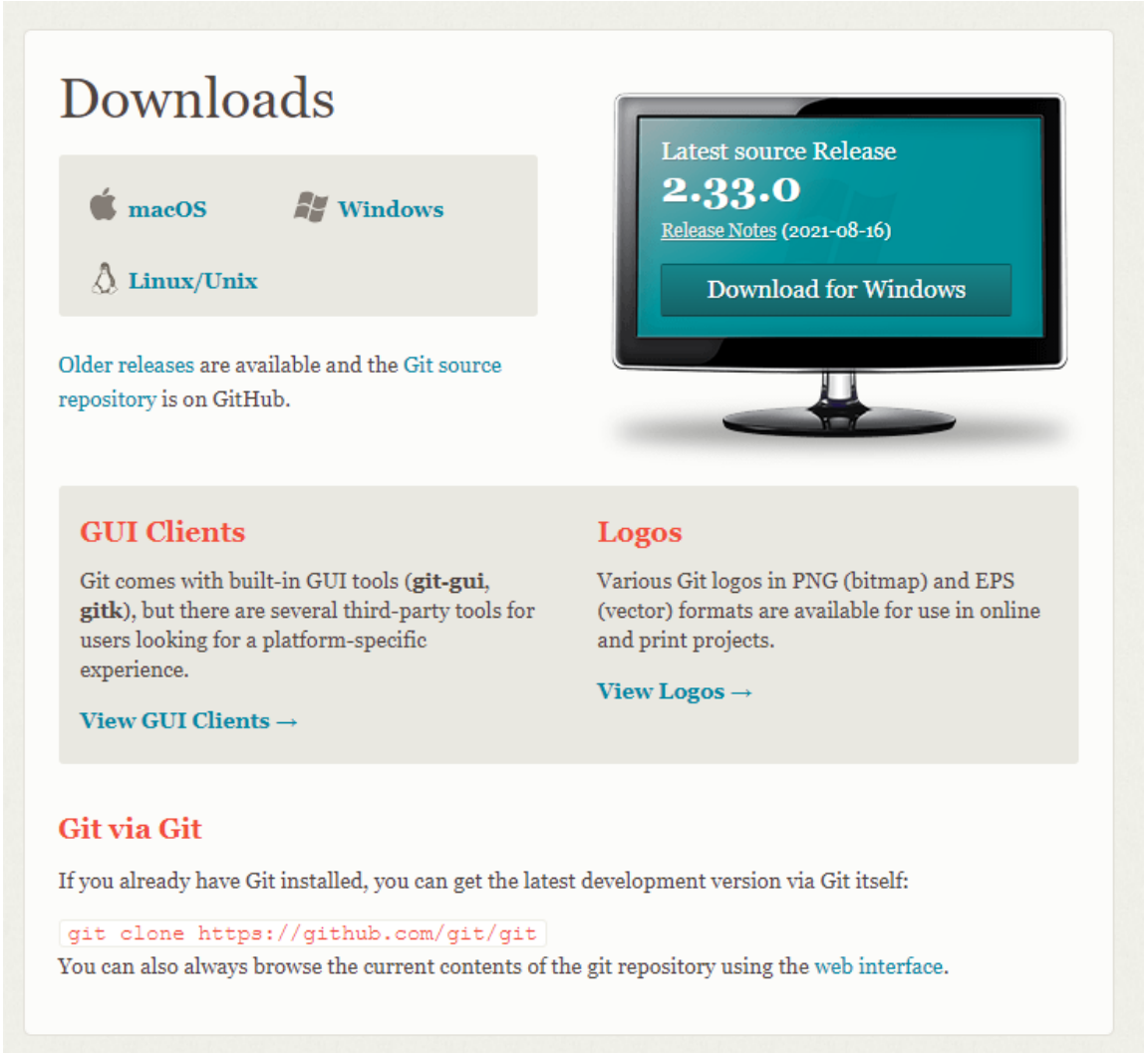


Git Push & Clone



Git Installing

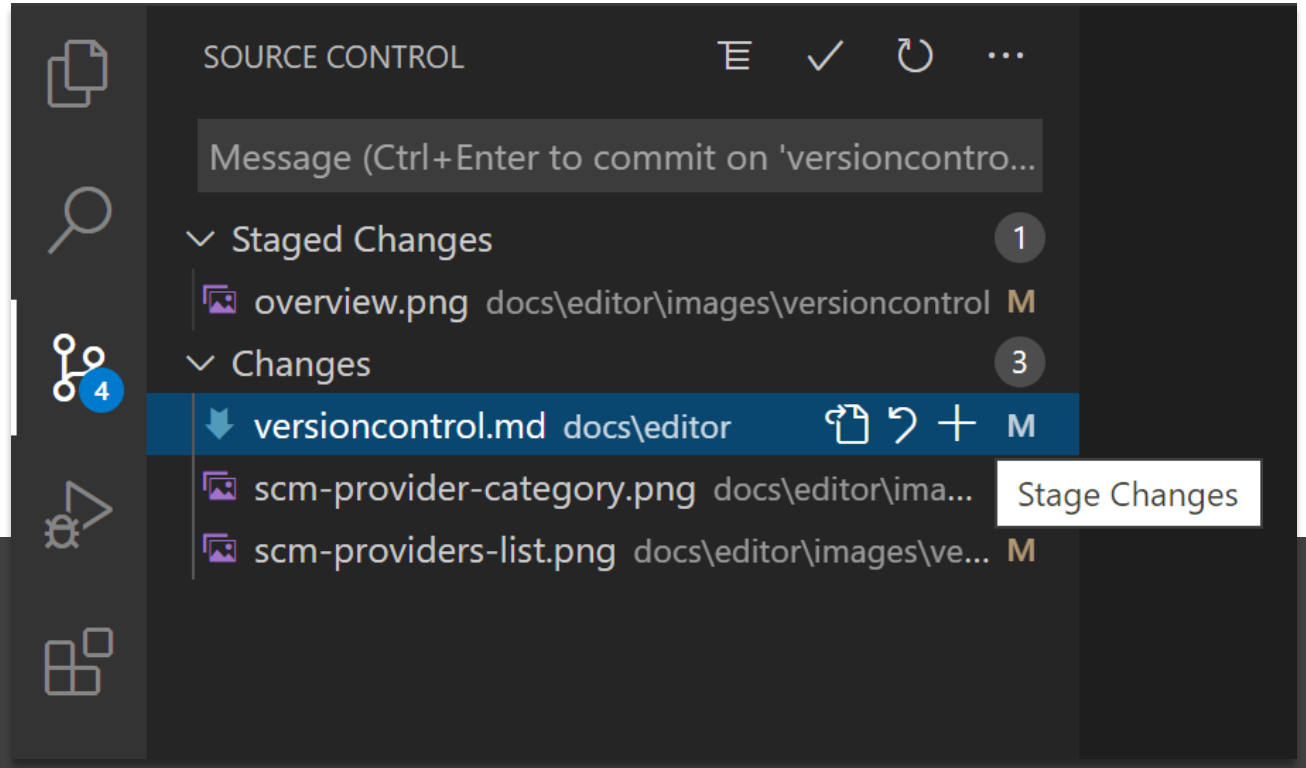
- Installing Git
  - This depends on your particular system. It can be downloaded from https://git-scm.com/downloads
  - It can usually be installed by the package manager in Linux systems.

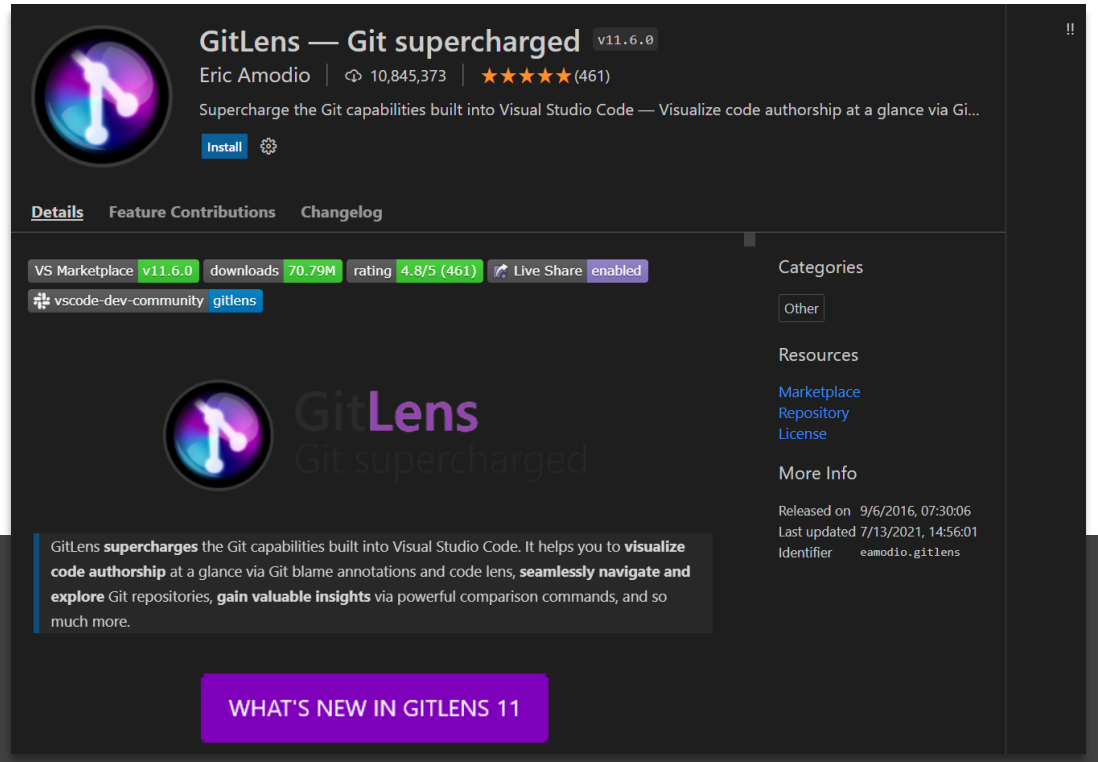


Using Git

- The command line
- Code Editors & IDEs
- Graphical User Interfaces

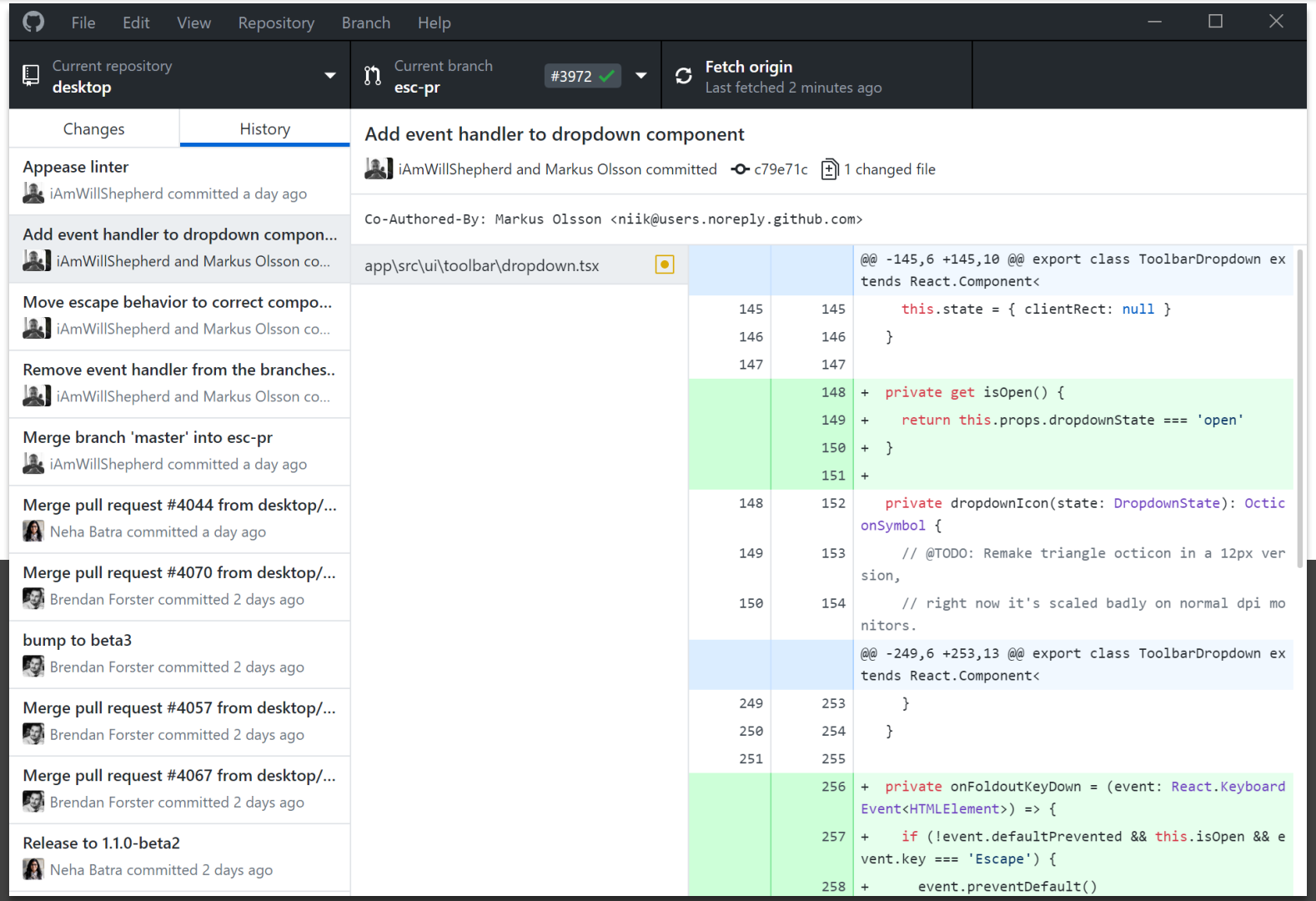
Code Editors & IDEs





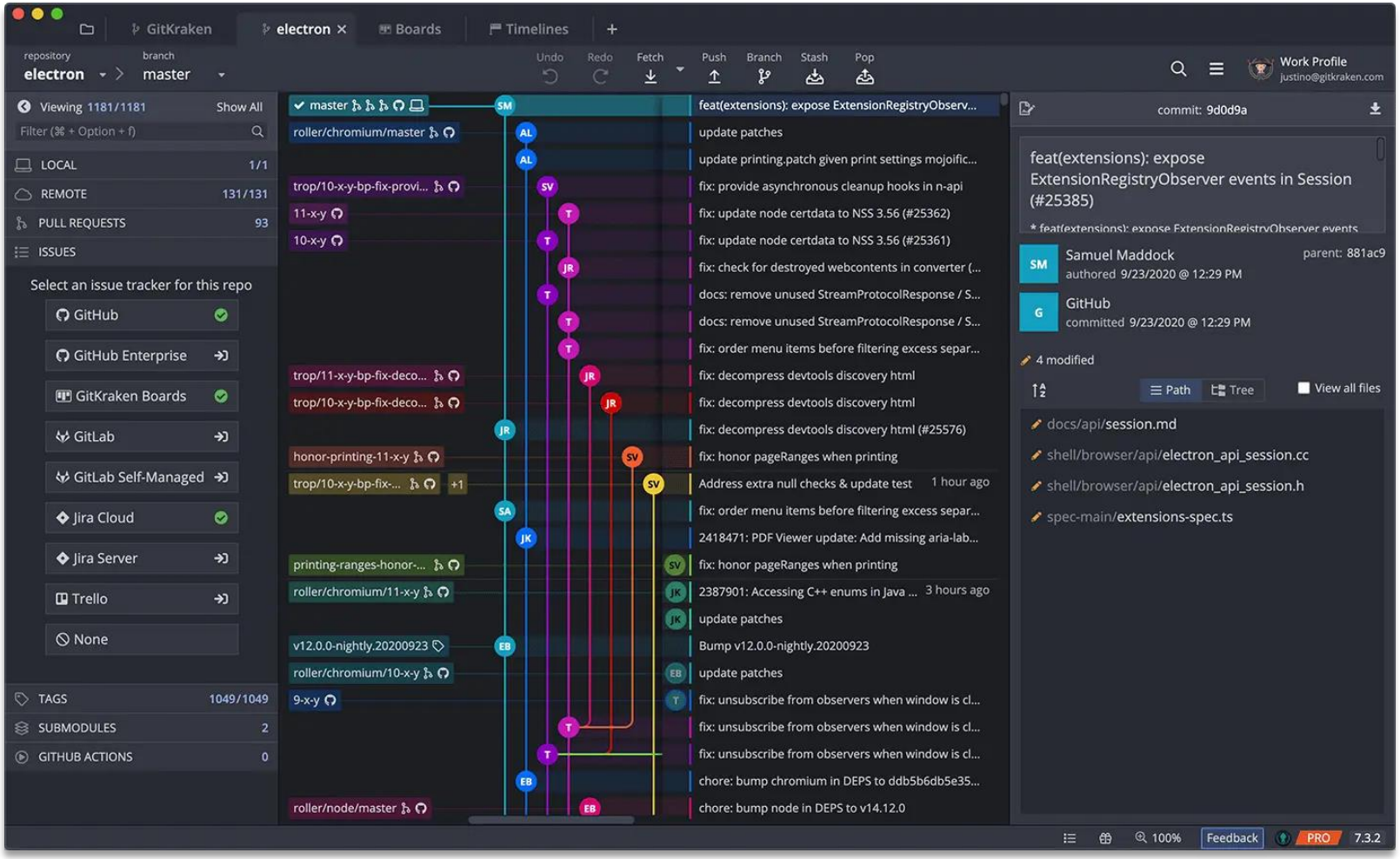
https://code.visualstudio.com/docs/editor/versioncontrol

Graphical User Interfaces



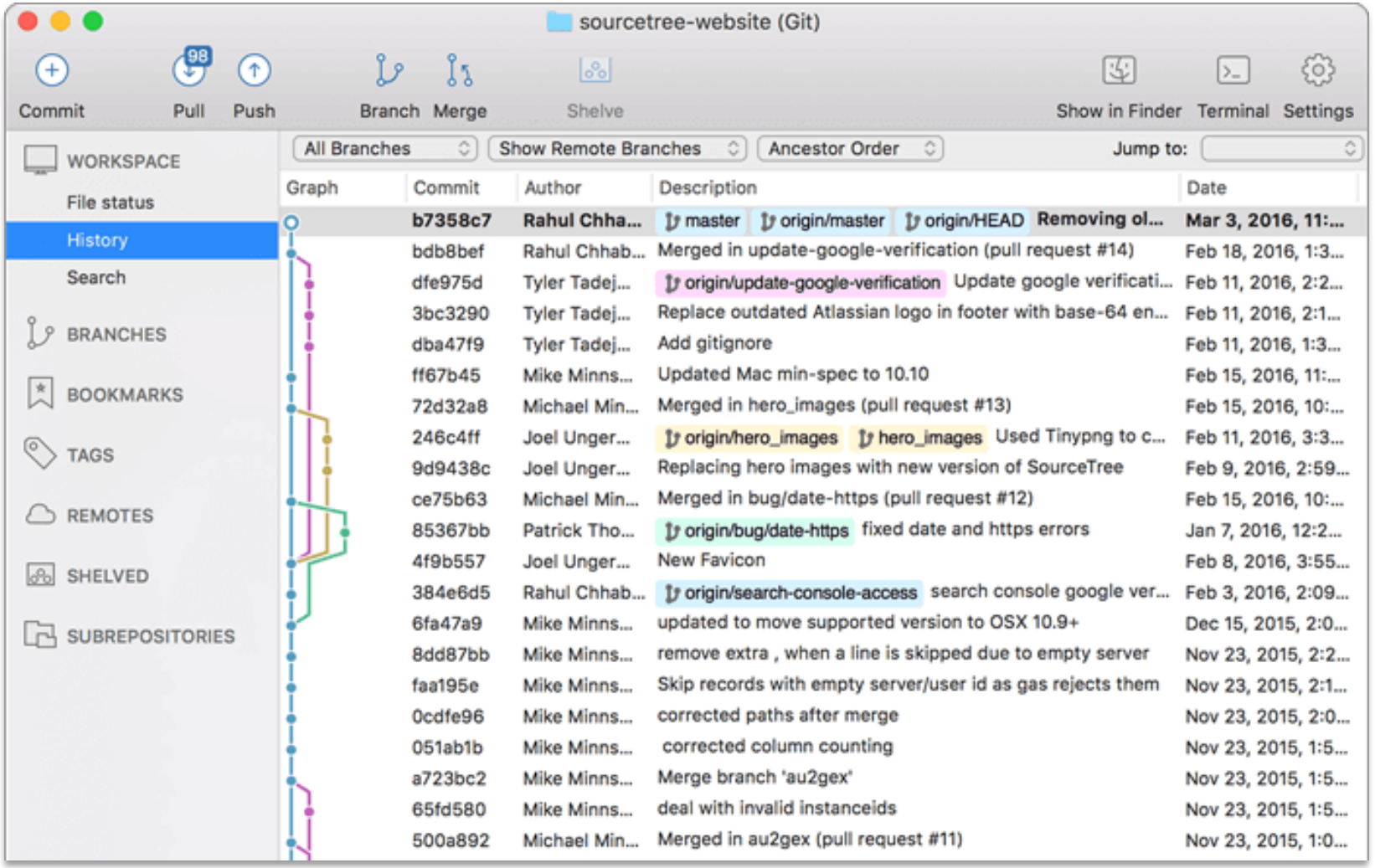
https://desktop.github.com/

Graphical User Interfaces



https://www.gitkraken.com/

Graphical User Interfaces

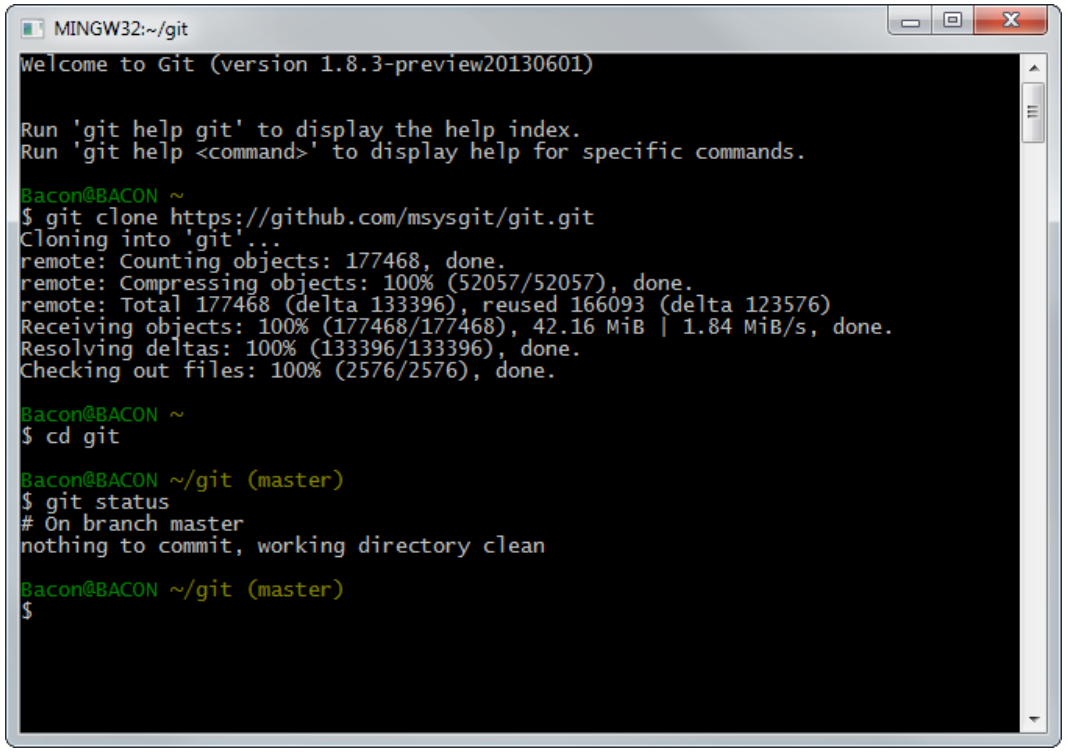


https://www.sourcetreeapp.com/

Why Command Line

- GUI Tools have limitations
- GUI tools are not always available

- Git SCM to Window
  - https://gitforwindows.org/
  - Git BASH



Terminal

- **macOS**
  - Press **Command** + **Space** and type **terminal**
- **Window**
  - Click the search icon and type **cmd**

Reading Materials

- Karl Fogel, *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media, 2009.
- https://choosealicense.com/
- https://opensource.guide/starting-a-project/
- Book : Pro Git Scott Chacon, Ben Straub

Thanks

Office Time: Monday-Friday (1000 - 1800)

You can send me an email for meeting, or any sort of discussion related to class matters.

jamil@sejong.ac.kr