SWCON201
Opensource & Software
Development Methods and Tools

Software Maintenance

Department of
Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Contents

- Agenda
- Class
- Summary
- Reference
- Homework

Agenda

- How multiple programmers co-work at the same time for the shared project?
  - GitHub (https://github.com/)
  - Google Docs (https://docs.google.com/)

Software Maintenance Tools

- Version Control
- Concurrent Versions System (CVS)
- Apache Subversion (SVN)
- Git
- GitHub

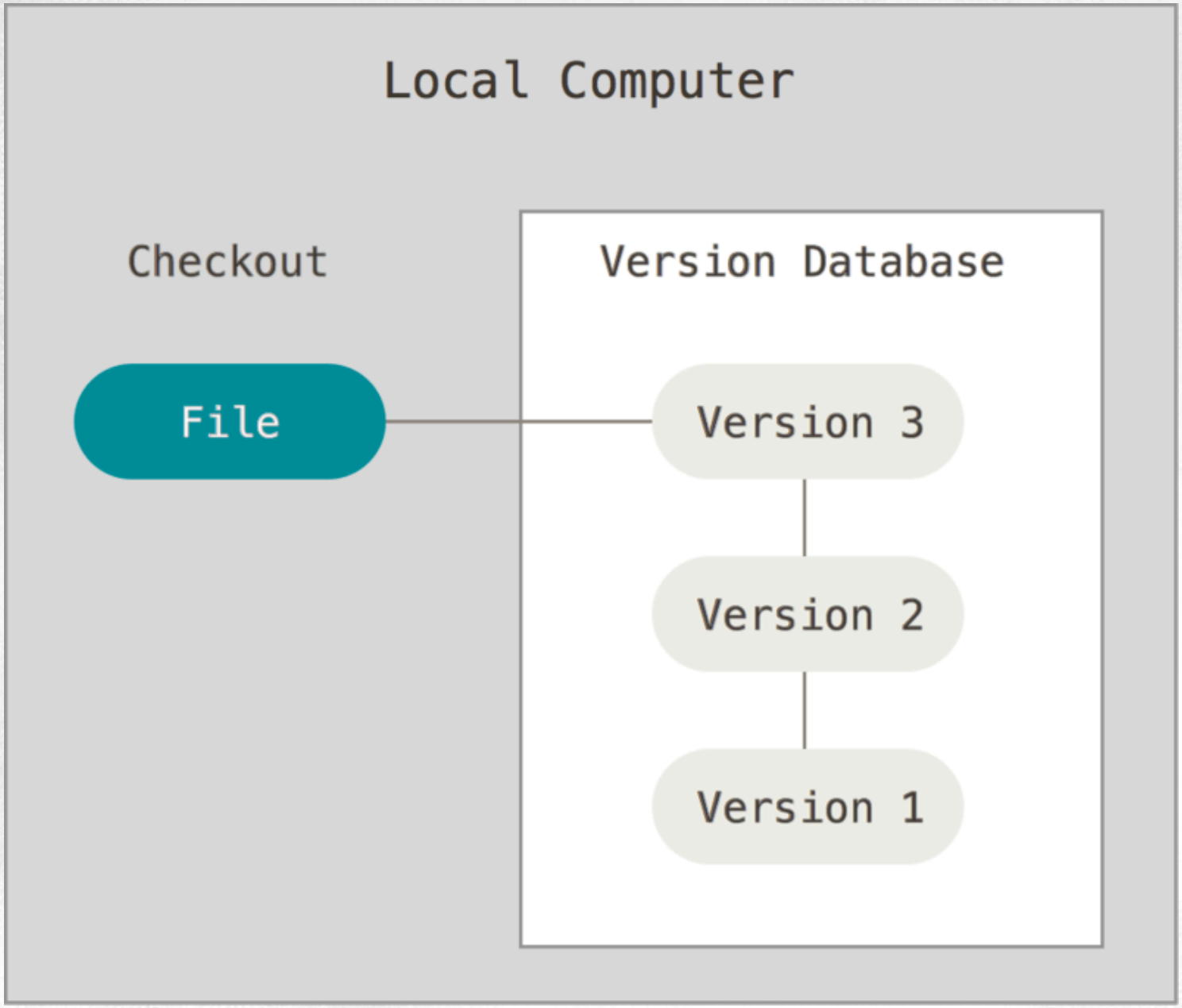
Version Control

- Also known as revision control or source control
- Management of changes to documents, computer programs, large web sites, and other collections of information
- Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision"
- For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on
- Each revision is associated with a time-stamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged
- Embedded in various softwares such as word processors, spreadsheets, collaborative web docs and in various content management systems (Wikipedia's page history)

Version Control like this?

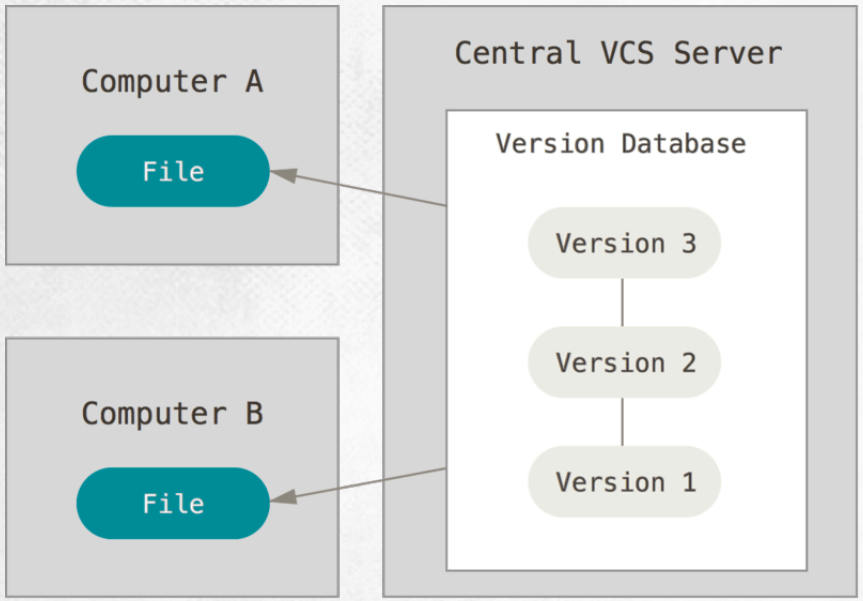
Reference: https://library.ucmerced.edu/node/66631

Version Control on Single Machine

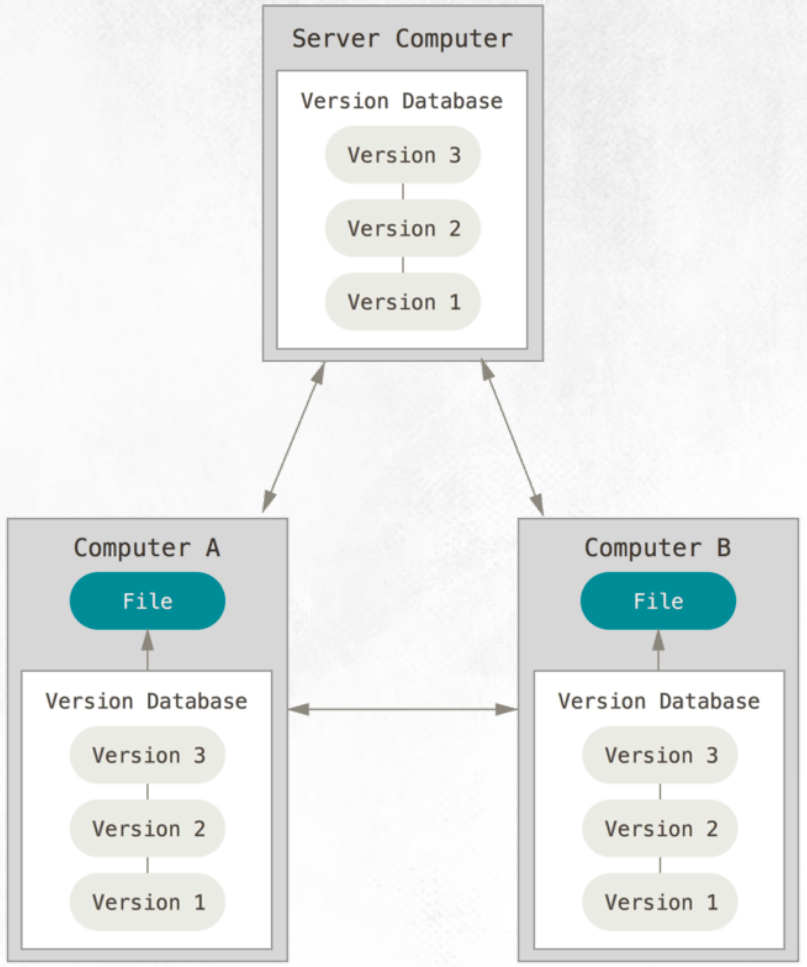


Reference: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Version Control on Multiple Machine



Centralized Version Control Systems

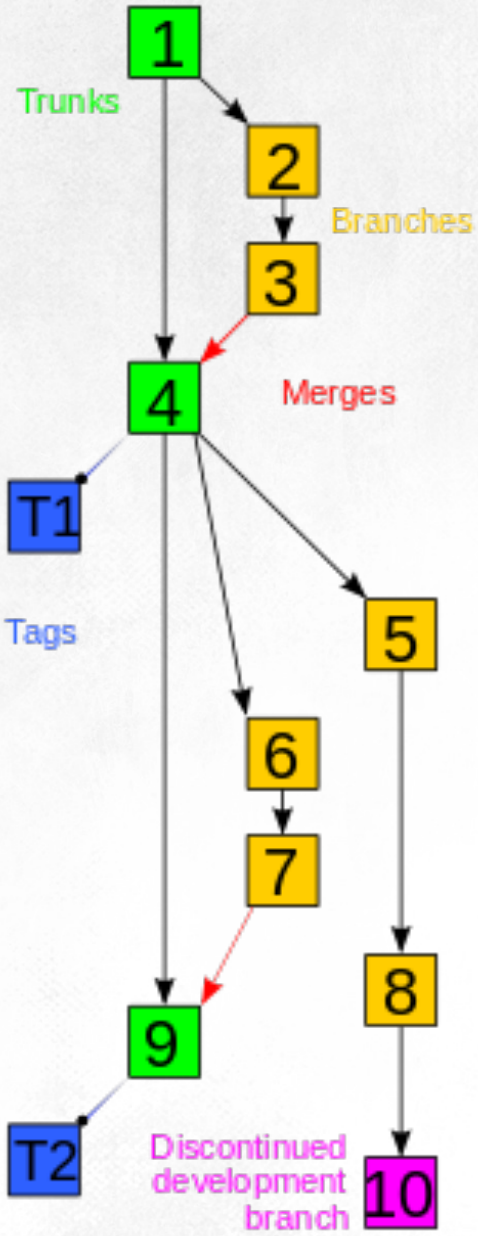


Distributed Version Control Systems

Reference: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

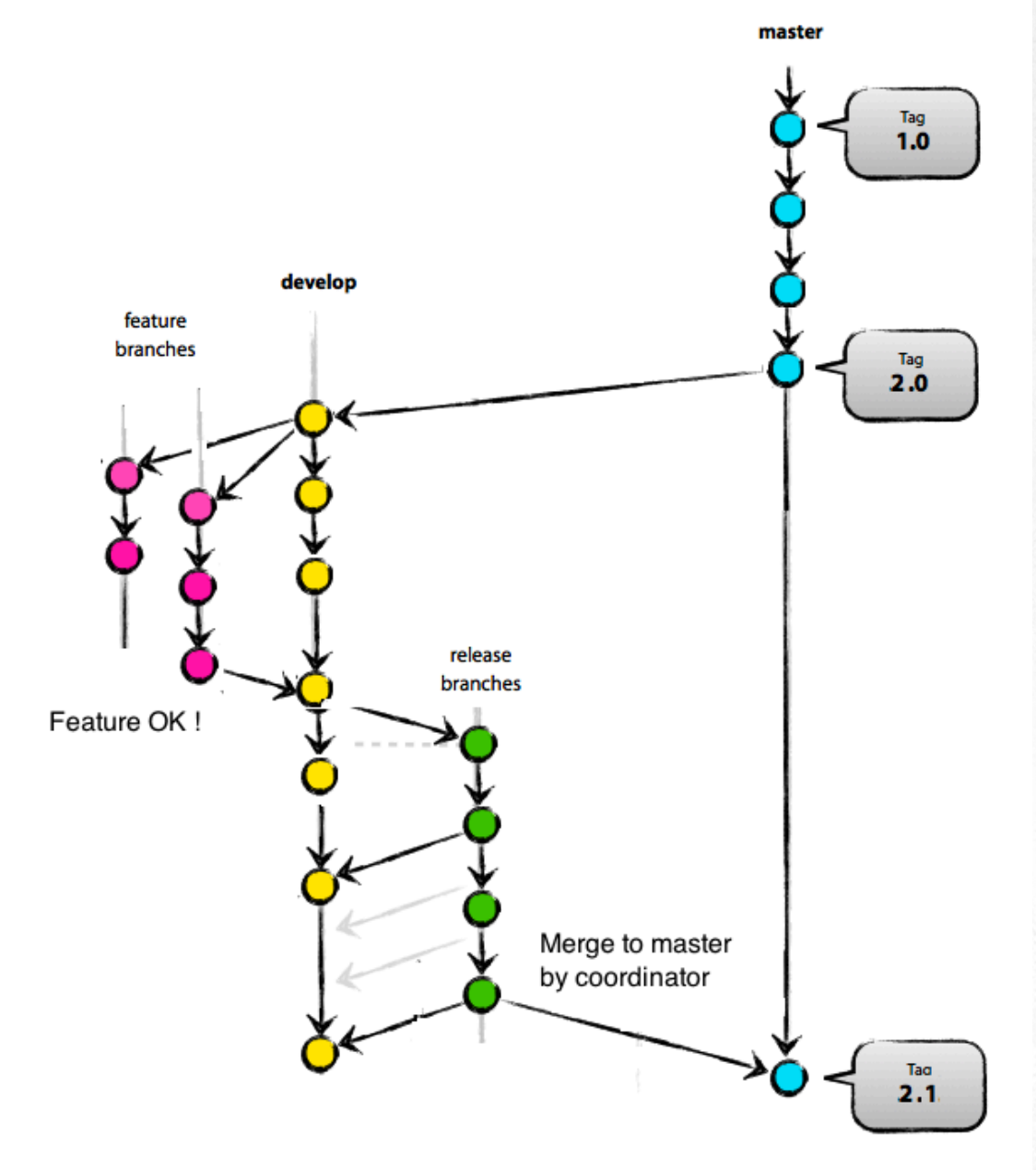
Version Control

- Revisions are generally thought of as a line of development (the trunk) with branches-off of this, forming a directed tree, visualized as one or more parallel lines of development (the "mainlines" of the branches) branching off a trunk
- In reality, the structure is more complicated, forming a directed acyclic graph, but for many purposes "tree with merges" is an adequate approximation



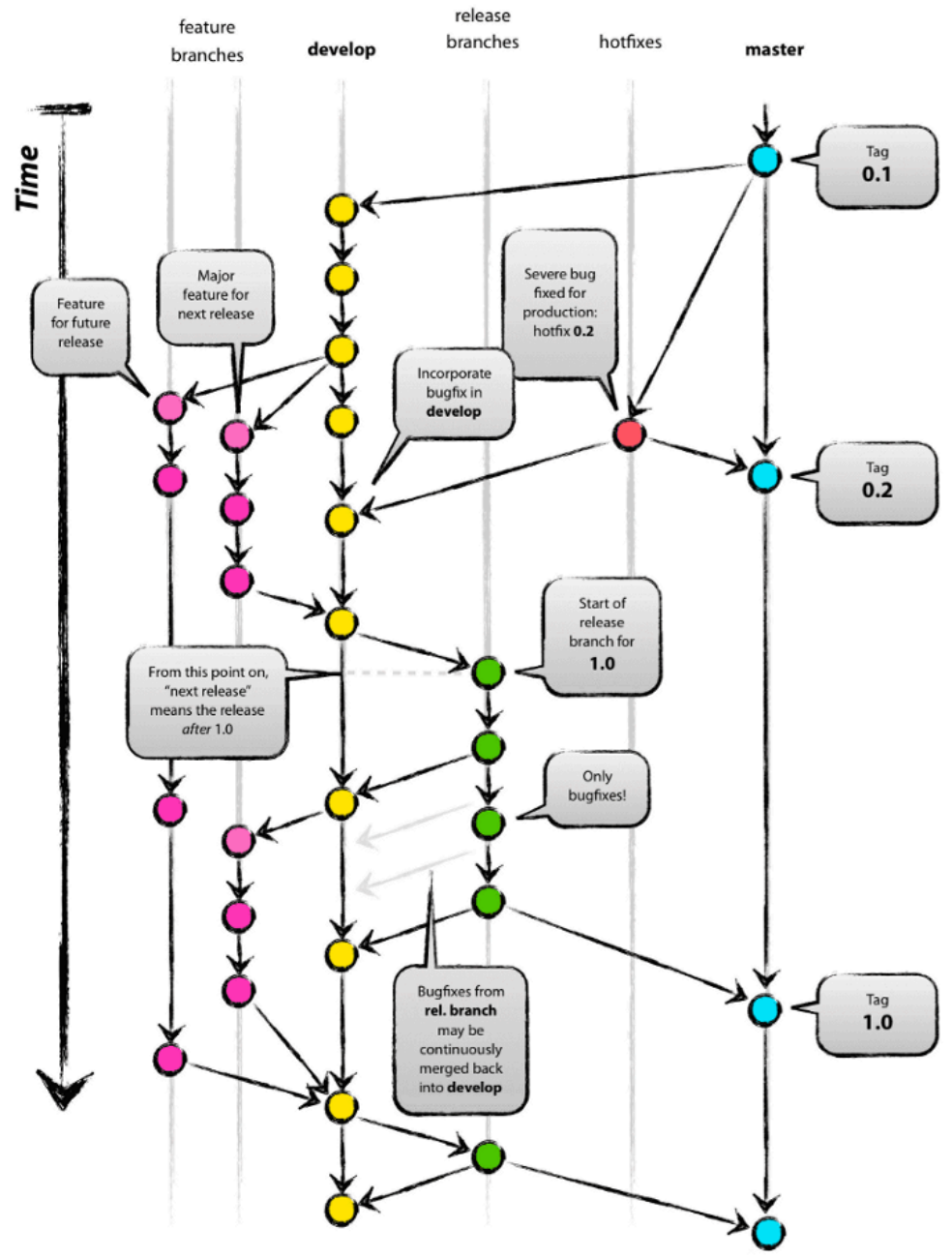
Reference: https://en.wikipedia.org/wiki/Version_control

Version Control example 1



Reference: https://www.crondose.com/2016/09/freelancers-guide-version-control/

Version Control example 2

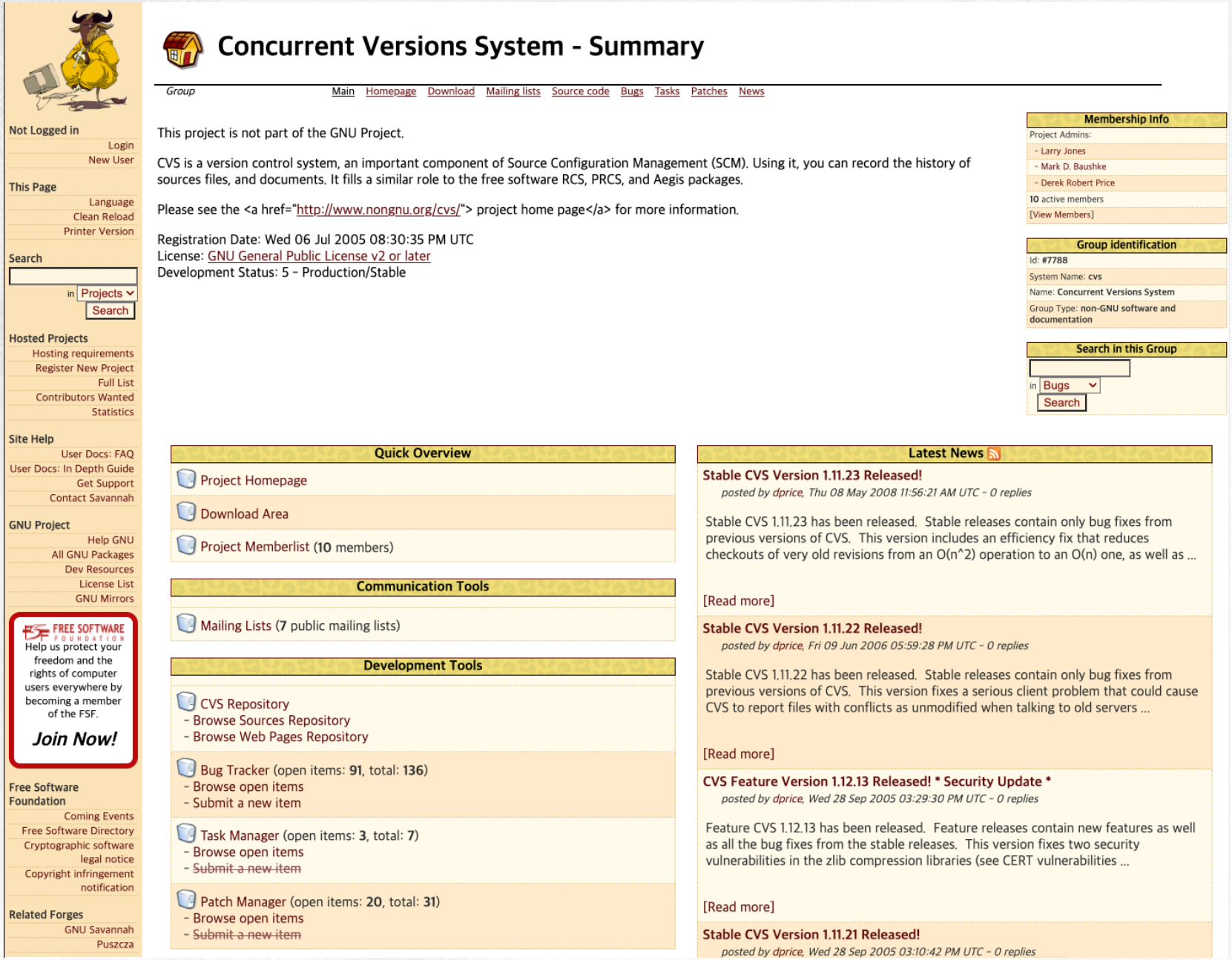


Reference: Reference: https://medium.com/trainingcenter/git-flow-as-control-system-version-844c242b19ab

Common Vocabularies

- Go to: https://en.wikipedia.org/wiki/Version_control
- Visit “Common Vocabulary”

Concurrent Versions System (CVS)



CVS

- Developer(s): The CVS Team
- Initial release: November 19, 1990
- Stable release: 1.11.23 / May 8, 2008 [stopped]
- Written in: C
- Operating system: Unix-like, Windows
- Type: Revision control
- License: GNU General Public License
- Website: savannah.nongnu.org/projects/cvs

CVS

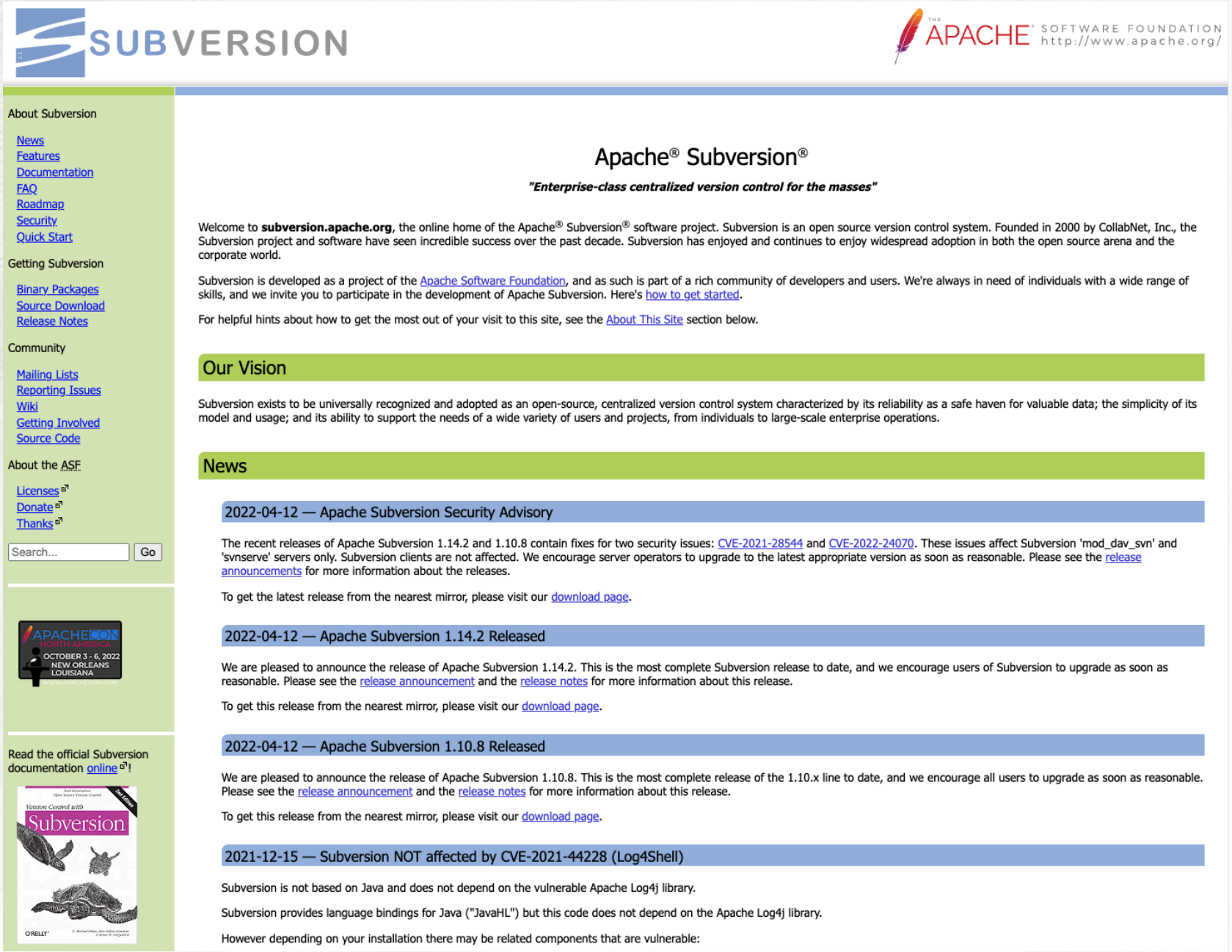
- Free software client-server revision control system in the field of software development
- A version control system keeps track of all work and all changes in a set of files, and allows several developers (potentially widely separated in space and time) to collaborate
- CVS uses a client–server architecture: a server stores the current version(s) of a project and its history, and clients connect to the server in order to "check out" a complete copy of the project, work on this copy and then later "check in" their changes
- No new releases since 2008

CVS

- "I created CVS to be able to cooperate with my students, Erik Baalbergen and Maarten Waage, on the ACK (Amsterdam Compiler Kit) C compiler. The three of us had vastly different schedules (one student was a steady 9-5 worker, the other was irregular, and I could work on the project only in the evenings). Their project ran from July 1984 to August 1985. CVS was initially called cmt, for the obvious reason that it allowed us to commit versions independently."

From: http://dickgrune.com/Programs/CVS.orig/#History

Apache Subversion (SVN)



SVN

- Original author(s): CollabNet
- Developer(s): Apache Software Foundation
- Initial release: 20 October 2000
- Stable release: 1.14.2 / 12 April 2022 [alive]
- Development status: Active
- Written in: C
- Operating system: Cross-platform
- Type: Revision control
- License: Apache License version 2.0
- Website: subversion.apache.org

Reference: https://en.wikipedia.org/wiki/Apache_Subversion

SVN

- Software versioning and revision control system distributed as open source under the Apache License
- Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation
- Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS)
- The open source community has used Subversion widely: for example in projects such as Apache Software Foundation, Free Pascal, FreeBSD, GCC and SourceForge
- Subversion was created by CollabNet Inc. in 2000, and is now a top-level Apache project being built and used by a global community of contributors

git --everything-is-local

Git

- Original author(s): Linus Torvalds
- Developer(s): Junio Hamano and others
- Initial release: 7 April 2005; 13 years ago
- Stable release: 2.38.1 / 18 October 2022 [alive]
- Repository: github.com/git/git
- Development status: Active
- Written in: C, Shell, Perl, Tcl, Python
- Operating system: POSIX: Linux, Windows, macOS
- Platform: IA-32, x86-64
- Available in: English
- Type: Version control
- License" GNU GPL v2 and GNU LGPL v2.1
- Website: git-scm.com

Reference: https://en.wikipedia.org/wiki/Git

Git

- Git (/ɡɪt/) is a version control system for tracking changes in computer files and coordinating work on those files among multiple people
- It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files
- As a distributed revision control system it is aimed at speed,and data integrity
  - As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server

Git

- Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano
- Git is free and open source software distributed under the terms of the GNU General Public License version 2

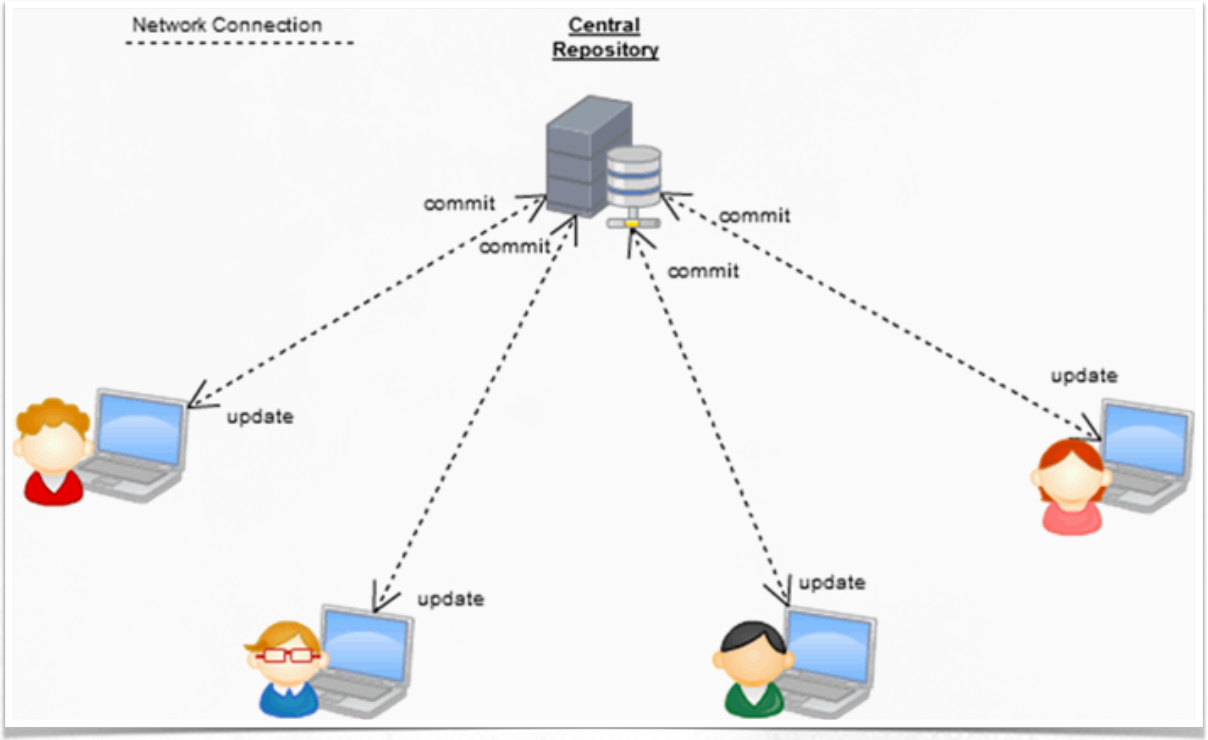
Reference: https://en.wikipedia.org/wiki/Git

Git - Distributed Version Control

- Distributed version control (also known as distributed revision control) is a form of version control where the complete codebase - including its full history - is mirrored on every developer's computer

Git - Distributed Version Control

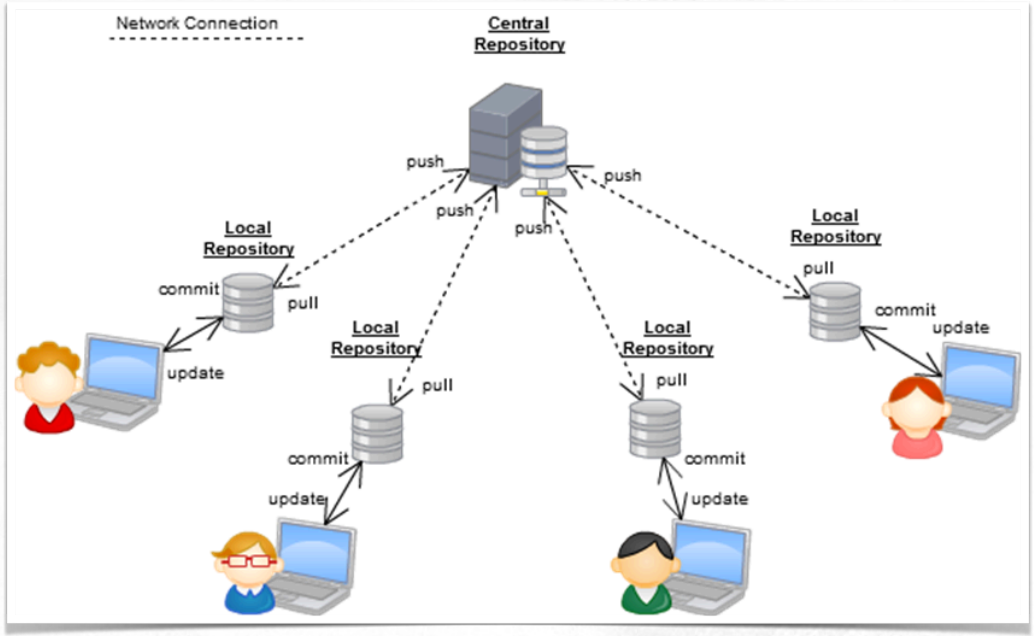
- Centralized Version Control System (CVCS)
  - We have the central server.
    - All the source-code of the team will be stored there.
  - Each member of the team will checkout the code directly from server.
  - After modifying code on their local, developers need to push their changes directly into server code. Thus, other member can see those changes.



Reference: https://www.codeproject.com/Articles/1165512/GIT-Branching-Model

Git - Distributed Version Control

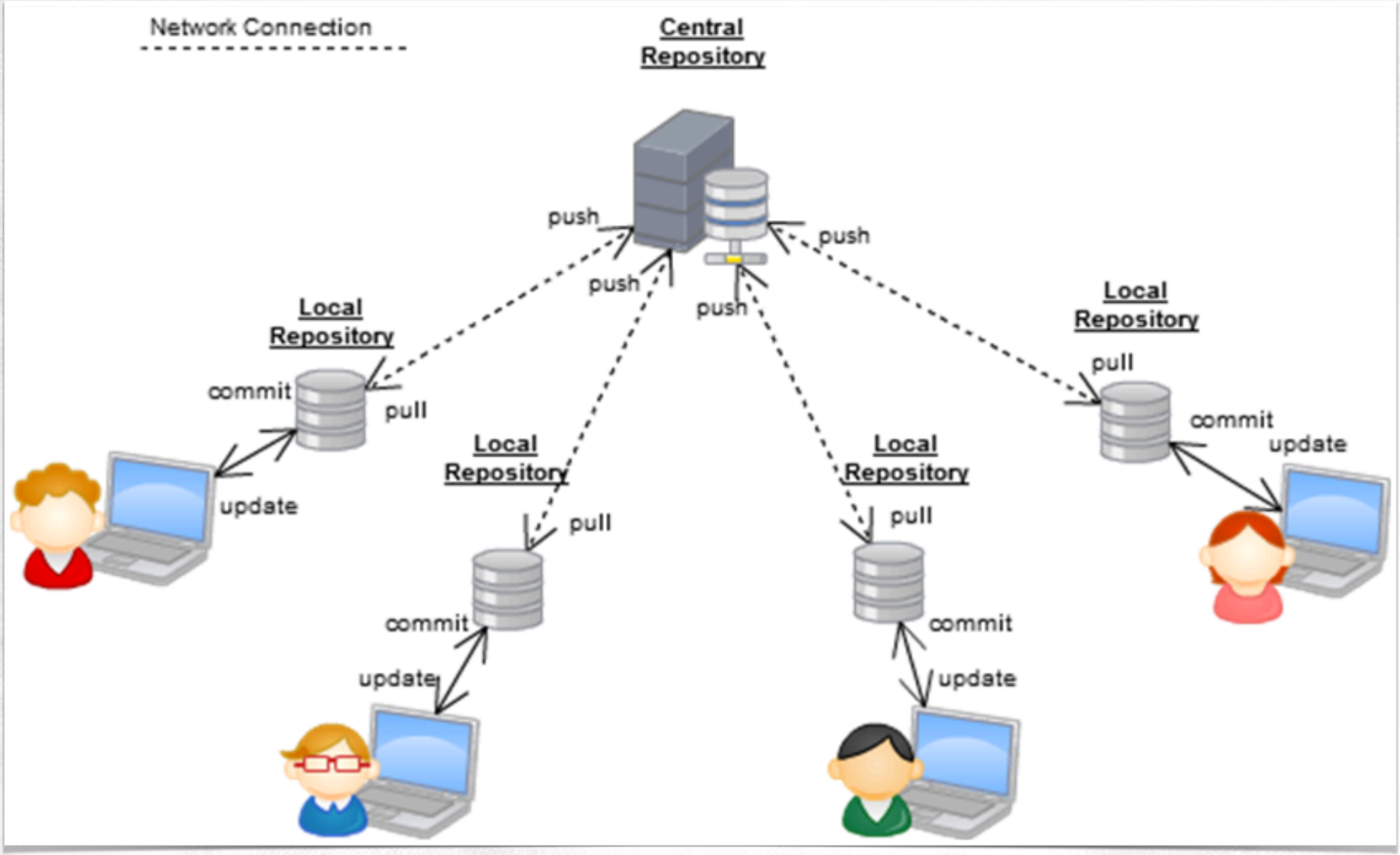
- Distributed Version Control System (DVCS)
  - All the source-code of teams will be stored on central server (remote) as in CVCS.
  - Developers need to checkout the code from this server if they wish to work on it.
  - There is another version of GIT on their local. So developers will temporarily push their changes to this local git server.
  - Developers can push all their local changes to remote at once if they want.



Reference: https://www.codeproject.com/Articles/1165512/GIT-Branching-Model

Git - Distributed Version Control

- Distributed Version Control System (DVCS)



Reference: https://www.codeproject.com/Articles/1165512/GIT-Branching-Model

Git - Distributed Version Control

- Central and local/branch repositories
  - Every project has a central repository that is considered as the official repository, which is managed by the project maintainers
  - Developers clone this repository to create identical local copies of the code base
  - Source code changes in the central repository are periodically synchronized with the local repository
  - The developer creates a new branch in his local repository and modifies source code on that branch
  - Once the development is done, the change needs to be integrated into the central repository

Reference: https://en.wikipedia.org/wiki/Distributed_version_control

Git - Distributed Version Control

- Pull requests
  - Contributions to a source code repository that uses a distributed version control system are commonly made by means of a pull request
  - The contributor requests that the project maintainer "pull" the source code change, hence the name "pull request"
  - The maintainer has to merge the pull request if the contribution should become part of the source base
  - The developer creates a pull request to notify maintainers of a new change; a comment thread is associated with each pull request
  - This allows for focused discussion of code changes

Reference: https://en.wikipedia.org/wiki/Distributed_version_control

Git - Distributed Version Control

- Pull requests (continued)
  - Submitted pull requests are visible to anyone with repository access
  - A pull request can be accepted or rejected by maintainers
  - Once the pull request is reviewed and approved, it is merged into the repository
  - Depending on the established workflow, the code may need to be tested before being included into official release
  - ·Therefore, some projects contain a special branch for merging untested pull requests
  - Other projects run an automated test suite on every pull request, and the reviewer checks that any new code has appropriate test coverage

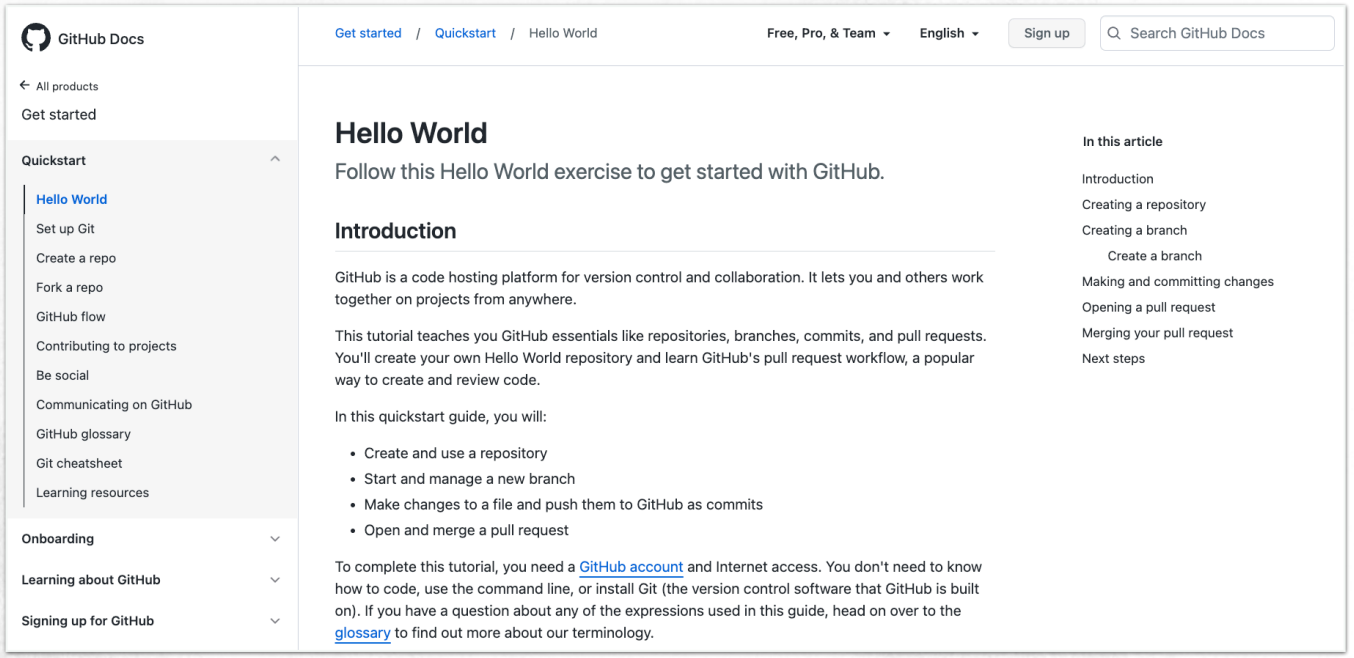
Reference: https://en.wikipedia.org/wiki/Distributed_version_control

Git by Example

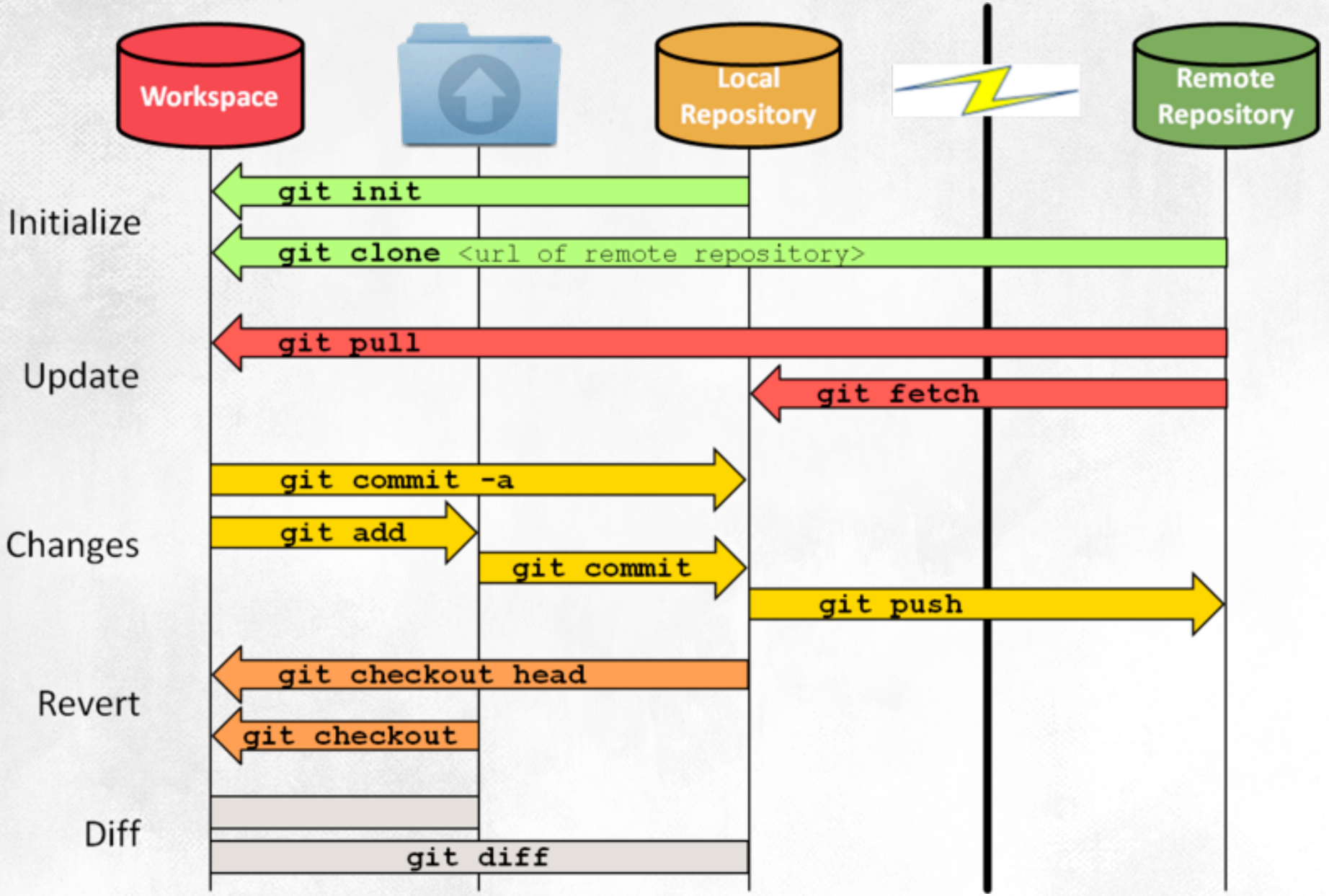
(1) Visit : https://docs.github.com/en/get-started/quickstart/hello-world

(2) Read and Experience :

- ☑ Create and use a repository
- ☑ Start and manage a new branch
- ☑ Make changes to a file and push them to GitHub as commits
- ☑ Open and merge a pull request



Git deep dive - big picture



Reference: https://medium.com/analytics-vidhya/git-most-frequently-used-commands-9df9f200c235

Git deep dive - terms

- Remote Repository
  - 여러 사람이 함께 공유하기 위한 파일을 보관하는 전용 저장소
- Local Repository
  - 내 PC에 파일이 저장되는 개인 저장소 (Commit 하는 저장소)
- Index (Staging Area)
  - Working Directory 에서 Repository로 정보가 저장되기 전 준비 영역
  - 파일 상태를 기록, 스테이징 한다고도 표현함Working Directory (작업 영역)
  - 실제 코드를 수정하고 추가하는 변경이 이루어지는 영역
  - 실제 프로젝트 디렉토리, git 이력과 관련된 정보가 저장되어있는 .git을 제외한 모든 영역

Reference:https://zoosso.tistory.com/714

Git deep dive - (a) initialize

- **git init** vs **git clone**
  - At a high level, they can both be used to initialize a new git repository.
  - However, **git clone** is dependent on **git init**. Internally, **git clone** first calls **git init** to create a new repository. It then copies the data from the existing repository.

Git deep dive - (a) initialize

- **git init**
  - Create a new empty git repository or convert an existing un-versioned project(workspace) to git repository.
  - Executing this command creates a .git subdirectory, which contains all the metadata for the new repository. This metadata includes subdirectories for objects, refs, and template files.
    A HEAD file is also created which points to the currently checked out commit.
- **git clone**
  - Download an existing git repository to your local computer.
  - `git clone -b branch_name <git url>`: The -b argument lets you specify a specific branch to clone instead of the branch the remote HEAD is pointing to, usually the master branch.

Git deep dive - (b) update

- **git pull**
  - Update local git repository from the corresponding remote git repository
  - git pull <remote> <local>:Local git repository ← Remote git repository
- **git fetch**
  - Update local git repository from the corresponding remote git repository. **git fetch** does not change your workspace, it keeps the fetched content separate until it is merged.
  - git fetch <remote> <local>
  - git checkout <remote>/<local>: To view the change
- **git pull** vs **git fetch**
  - **git pull** = **git fetch** + **git merge**

Reference: https://medium.com/analytics-vidhya/git-most-frequently-used-commands-9df9f200c235

Git deep dive - (c) change

- **git add**
  - Add changes in the workspace to the staging area.
- **git commit**
  - Add changes in the staging area to the local Git repository
  - git commit: Staging area → Local git repository
  - git commit -a: Workspace → Local git repository
    - ☑ Untracked files are not included, only those that have been added with git add at some point
- **git push**
  - Add changes in the local git repository to the remote repository
  - git push <remote> <local>: Local git repository → Remote git repository

Reference: https://medium.com/analytics-vidhya/git-most-frequently-used-commands-9df9f200c235

Git deep dive - (d) revert

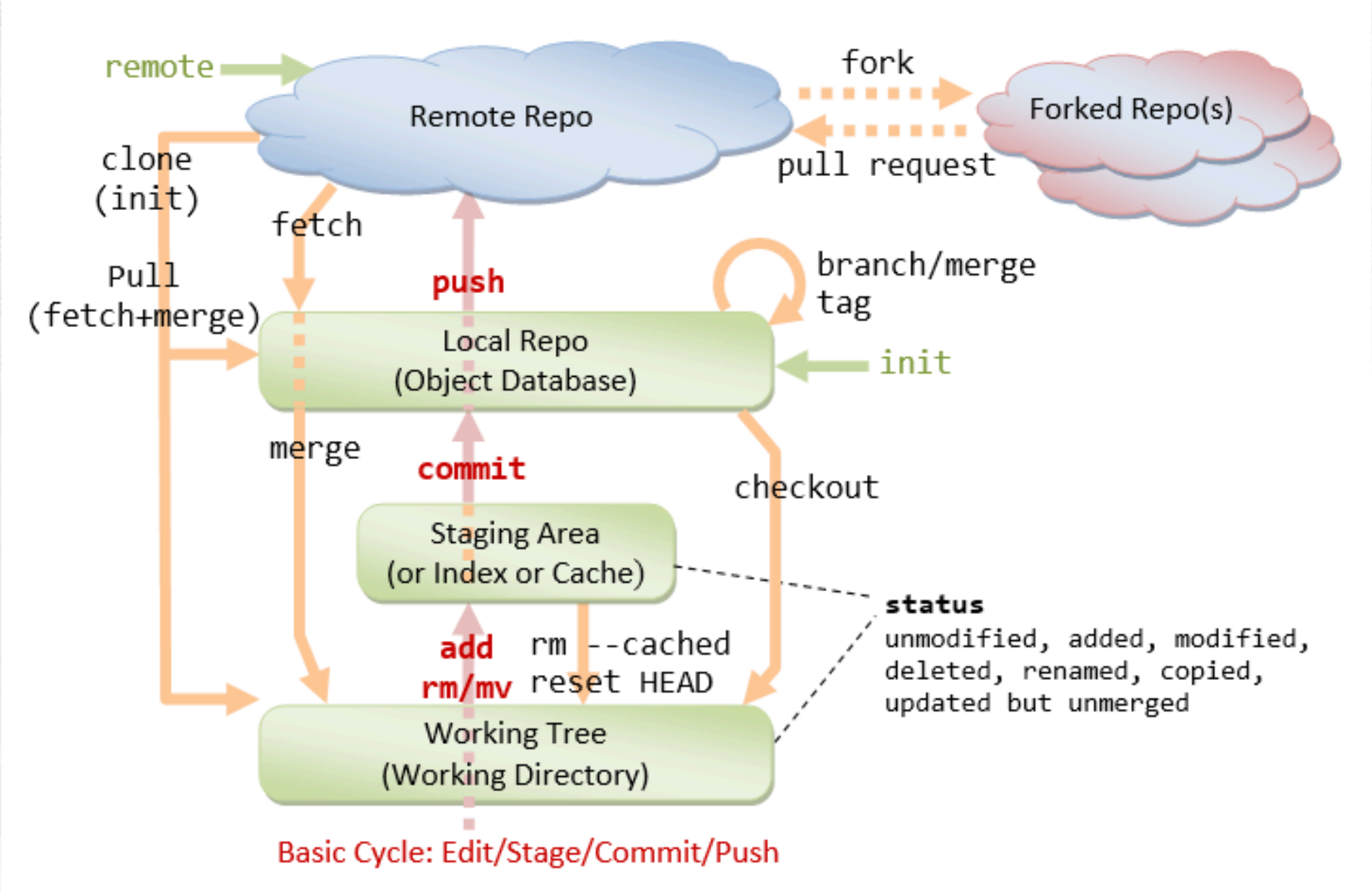
- **git checkout**
  - Navigate between different branches.
  - git checkout <branch>
  - git checkout -b <new branch>: Create a new branch from your current branch and switch to it.

Reference: https://medium.com/analytics-vidhya/git-most-frequently-used-commands-9df9f200c235

Git deep dive - (e) diff

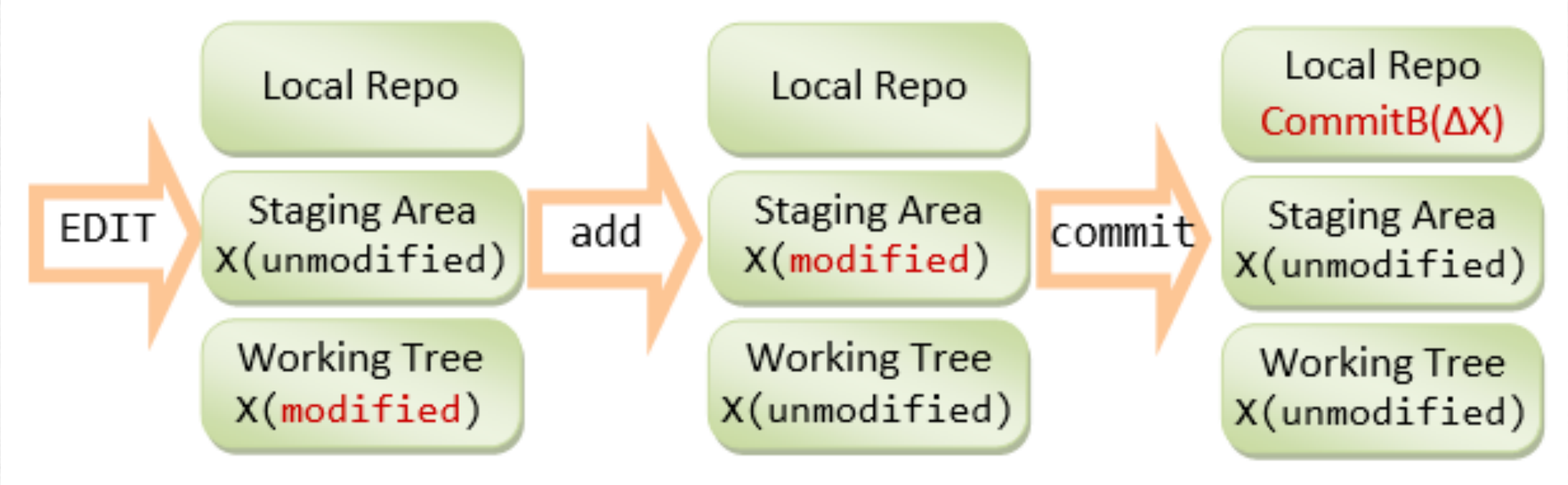
- **git status** displays:
  - Current branch
  - Files that have differences between Workspace ↔ Staging area (Untracked(new) files and Unstaged changes)
  - Files that have differences between Staging ↔ Local Git Repository (Uncommitted changes)

Git deep dive - another example



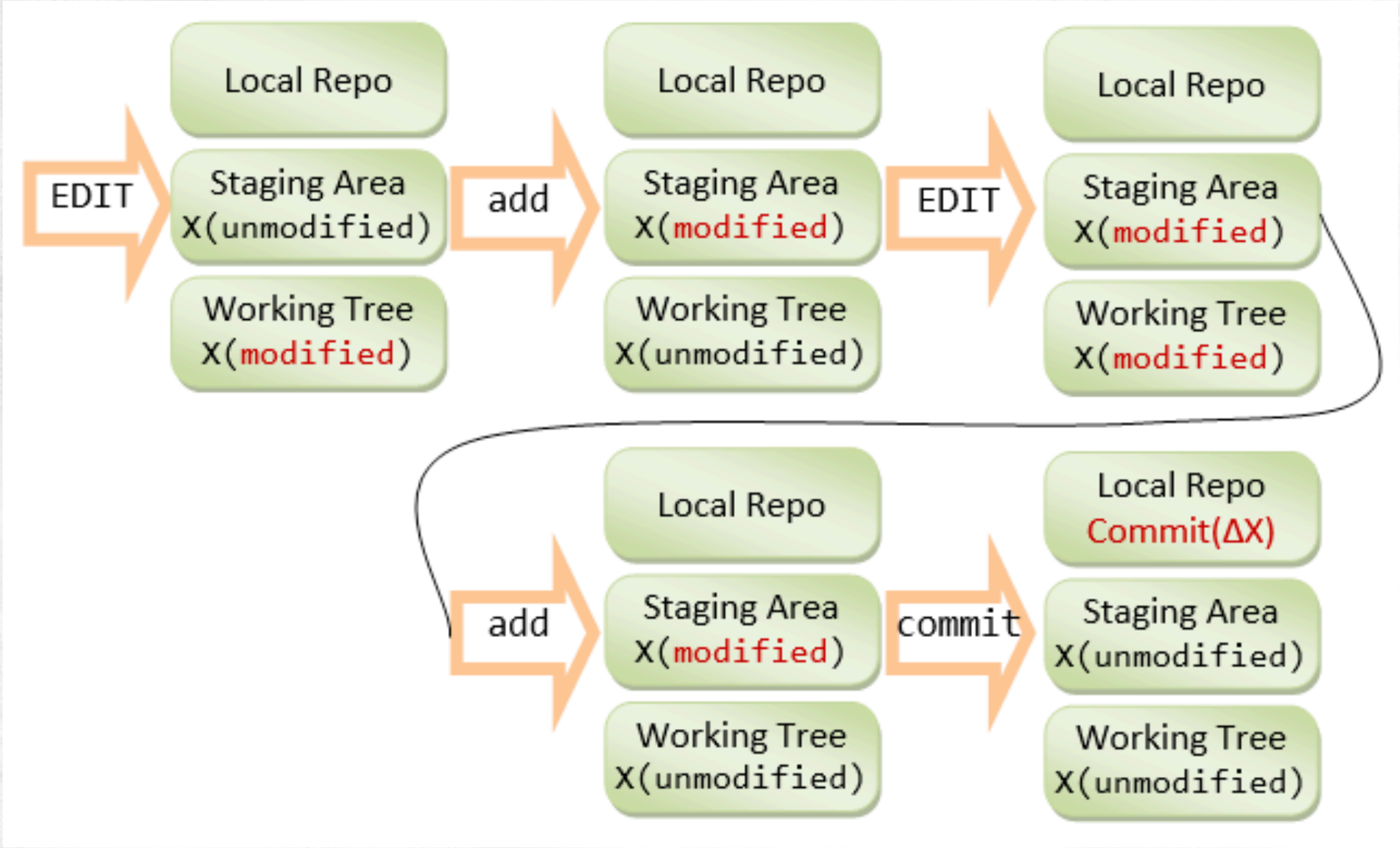
Reference: https://www3.ntu.edu.sg/home/ehchua/programming/howto/Git_HowTo.html

Git deep dive - more about add/commit



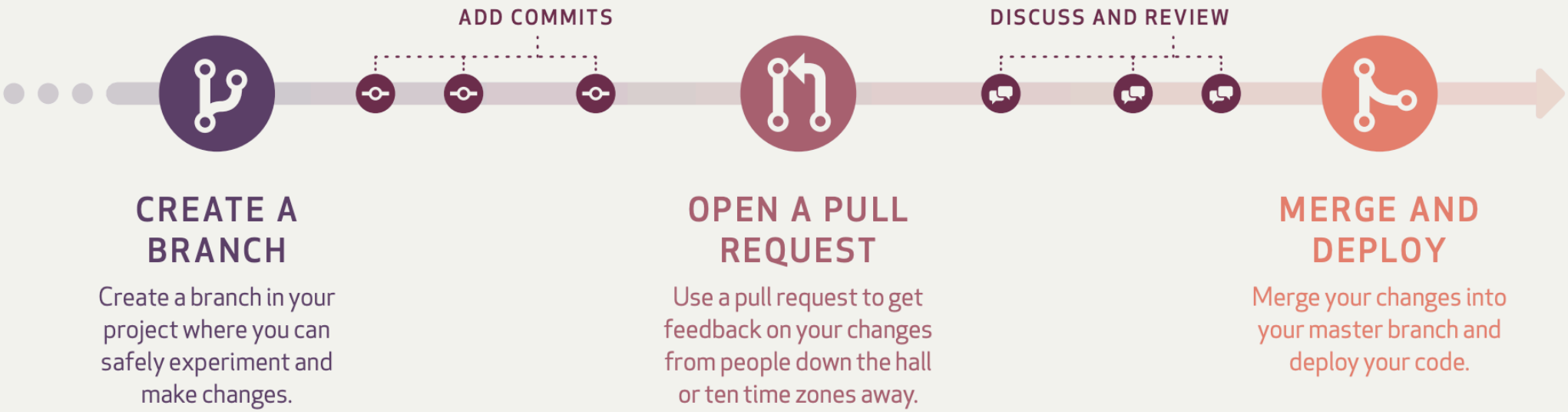
Reference: https://www3.ntu.edu.sg/home/ehchua/programming/howto/Git_HowTo.html

Git deep dive - more about add/commit



Reference: https://www3.ntu.edu.sg/home/ehchua/programming/howto/Git_HowTo.html

Git deep dive - more about pull request



Reference: https://subicura.com/git/guide/github-flow.html

GitHub

- Type of site: Git-repository hosting service
- Available in: English
- Founded: February 8, 2008; 11 years ago
- Headquarters: San Francisco, United States
- Employees: 2500
- Parent : Microsoft
- Website: github.com
- Users: 83 million (as of June 2022)
- Launched: 10 April 2008
- Current status: Active
- Written in: Ruby >> Ruby, JavaScript, Go, C

Reference: https://en.wikipedia.org/wiki/GitHub

GitHub

MICROSOFT | BREAKING | TECH

Microsoft completes GitHub acquisition

By Tom Warren | @tomwarren | Oct 26, 2018, 9:28am EDT

Microsoft revealed earlier this year that it's acquiring GitHub for $7.5 billion. Today, the software maker is confirming that this big acquisition is complete. Microsoft will operate GitHub independently as a business, to keep its platform and community in place and to allow the service to "retain its product philosophy."

As Microsoft's GitHub CEO, Nat Friedman, commented in a blog post announcing the completion of the deal, "Our vision is to serve every developer on the planet, by being the best place to build software. This is a dream opportunity for all of us at GitHub, and we couldn't be more excited to roll up our sleeves and start this next chapter."

Reference: https://www.theverge.com/2018/10/26/17954714/microsoft-github-deal-acquisition-complete

GitHub overall

- Web-based hosting service for version control using git
- It is mostly used for computer code
- It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features
- It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project
- As of June 2022, GitHub reported having over 83 million developers and more than 200 million repositories, including at least 28 million public repositories. It is the largest source code host as of November 2021.

Reference: https://en.wikipedia.org/wiki/GitHub

GitHub features

- Documentation, including automatically rendered README files in a variety of Markdown-like file formats
- Issue tracking (including feature requests) with labels, milestones, assignees and a search engine
- Wikis
- Pull requests with code review and comments
- Commits history
- Graphs: pulse, contributors, commits, code frequency, punch card, network, members
- Integrations Directory
- Unified and split diffs
- Email notifications

Reference: https://en.wikipedia.org/wiki/GitHub

GitHub features

- Option to subscribe someone to notifications by @ mentioning
- Emojis
- GitHub Pages: small websites can be hosted from public repositories on GitHub. The URL format is https://username.github.io
- Nested task-lists within files
- Visualization of geospatial data
- 3D render files that can be previewed using a new integrated STL file viewer that displays the files on a "3D canvas". The viewer is powered by WebGL and Three.js.
- Photoshop's native PSD format can be previewed and compared to previous versions of the same file.
- PDF document viewer

GitHub cheat sheet

Git 설치하기

GitHub은 일반적으로 많이 사용되는 저장소 관련 작업을 위한 데스크톱 클라이언트와 함께, 더 복잡한 작업을 위해 자동으로 업데이트되는 Git command line 에디션을 제공합니다.

Windows 사용자를 위한 GitHub

windows.github.com

Mac 사용자를 위한 GitHub

mac.github.com

리눅스와 POSIX 운영체제를 위한 Git 배포 버전은 Git의 공식 웹사이트인 Git SCM에서 확인하실 수 있습니다.

모든 플랫폼을 위한 Git

git-scm.com

환경 설정

모든 로컬 저장소에 적용할 사용자 정보를 설정합니다

`$ git config --global user.name "[name]"`

자신이 생성한 커밋(commit)에 들어갈 이름을 설정합니다

`$ git config --global user.email "[email address]"`

자신이 생성한 커밋에 들어갈 이메일 주소를 설정합니다

저장소 생성하기

새로운 저장소를 만들거나, 다른 저장소의 URL을 이용해 저장소를 복사합니다

`$ git init [project-name]`

새로운 로컬 저장소를 생성하고 이름을 정합니다.

`$ git clone [url]`

기존 프로젝트의 모든 커밋 내역을 가져와 저장소를 만듭니다

변경점을 저장하기

수정 사항을 검토하고 커밋을 생성합니다

`$ git status`

커밋할 수 있는 새로운 파일과 수정된 파일의 목록을 보여줍니다

`$ git diff`

수정하였으나 아직 stage하지 않은 파일의 변경점을 보여줍니다

`$ git add [file]`

커밋을 준비하기 위해 파일을 stage합니다

`$ git diff --staged`

stage하였으나 아직 커밋하지 않은 파일과 가장 최근에 커밋한 파일을 비교합니다

`$ git reset [file]`

파일의 내용은 유지한 채로 stage한 내역만을 제거합니다

`$ git commit -m"[descriptive message]"`

stage한 내용을 커밋으로 영구히 저장합니다

변경점을 묶어 관리하기

일련의 커밋에 이름을 붙이고 여러 변경점을 합칩니다

`$ git branch`

현재 저장소의 모든 로컬 브랜치를 보여줍니다

`$ git branch [branch-name]`

새로운 브랜치를 생성합니다

`$ git switch -c [branch-name]`

특정 브랜치로 전환하고 워킹 디렉토리를 업데이트합니다

`$ git merge [branch-name]`

현재 브랜치에 특정 브랜치의 히스토리를 병합시킵니다

`$ git branch -d [branch-name]`

브랜치를 삭제합니다

파일 이름 바꾸기

버전 관리 중인 파일을 옮기거나 삭제합니다

`$ git rm [file]`

워킹 디렉토리에 있는 파일을 제거하고 삭제한 내역을 stage합니다

`$ git rm --cached [file]`

현재 파일은 그대로 두고 버전 관리 체계에서만 제거합니다

`$ git mv [file-original] [file-renamed]`

파일명을 변경하고 해당 내역을 stage합니다

특정 파일을 저장소에서 제외하기

임시 파일과 경로를 제외시킵니다

```
*.log
build/
temp-*
```

`.gitignore` 이라는 텍스트 파일에 제외할 문자열 패턴을 지정하여 실수로 엉뚱한 파일이나 경로가 저장되지 않게 방지할 수 있습니다

`$ git ls-files --others --ignored --exclude-standard`

이 프로젝트에서 제외된 모든 파일을 보여줍니다

변경점 일부분을 저장하기

불완전한 변경 사항을 임시로 저장하거나 복원합니다

`$ git stash`

버전 관리 중인 모든 파일의 변경점을 임시로 저장합니다

`$ git stash pop`

가장 최근에 임시 저장한 내용을 복원합니다

`$ git stash list`

임시 저장된 모든 변경점의 목록을 보여줍니다

`$ git stash drop`

가장 최근에 임시 저장한 내용을 지웁니다

변경 기록 검토

프로젝트 내 파일의 변경 기록을 살펴보고 검토합니다

`$ git log`

현재 브랜치의 변경 기록을 보여줍니다

`$ git log --follow [file]`

특정 파일의 변경 기록을 보여줍니다(파일명 변경 포함)

`$ git diff [first-branch]...[second-branch]`

두 브랜치의 차이점을 비교합니다

`$ git show [commit]`

특정 커밋에 포함된 변경 사항과 메타데이터를 표시합니다

커밋 되돌리기

실수한 내용을 지우고 기록을 바꿉니다

`$ git reset [commit]`

현재 파일의 변경 사항은 그대로 두고 '[커밋]' 이후의 모든 커밋 내용을 되돌립니다

`$ git reset --hard [commit]`

모든 변경점과 기록을 버리고 특정 커밋으로 되돌아갑니다

변경점을 동기화하기

원격 저장소(의 URL)을 등록하고 저장소 기록을 주고받습니다

`$ git fetch [remote]`

원격 저장소로부터 모든 기록을 받아옵니다

`$ git merge [remote]/[branch]`

원격 브랜치를 현재 사용 중인 로컬 브랜치와 병합합니다

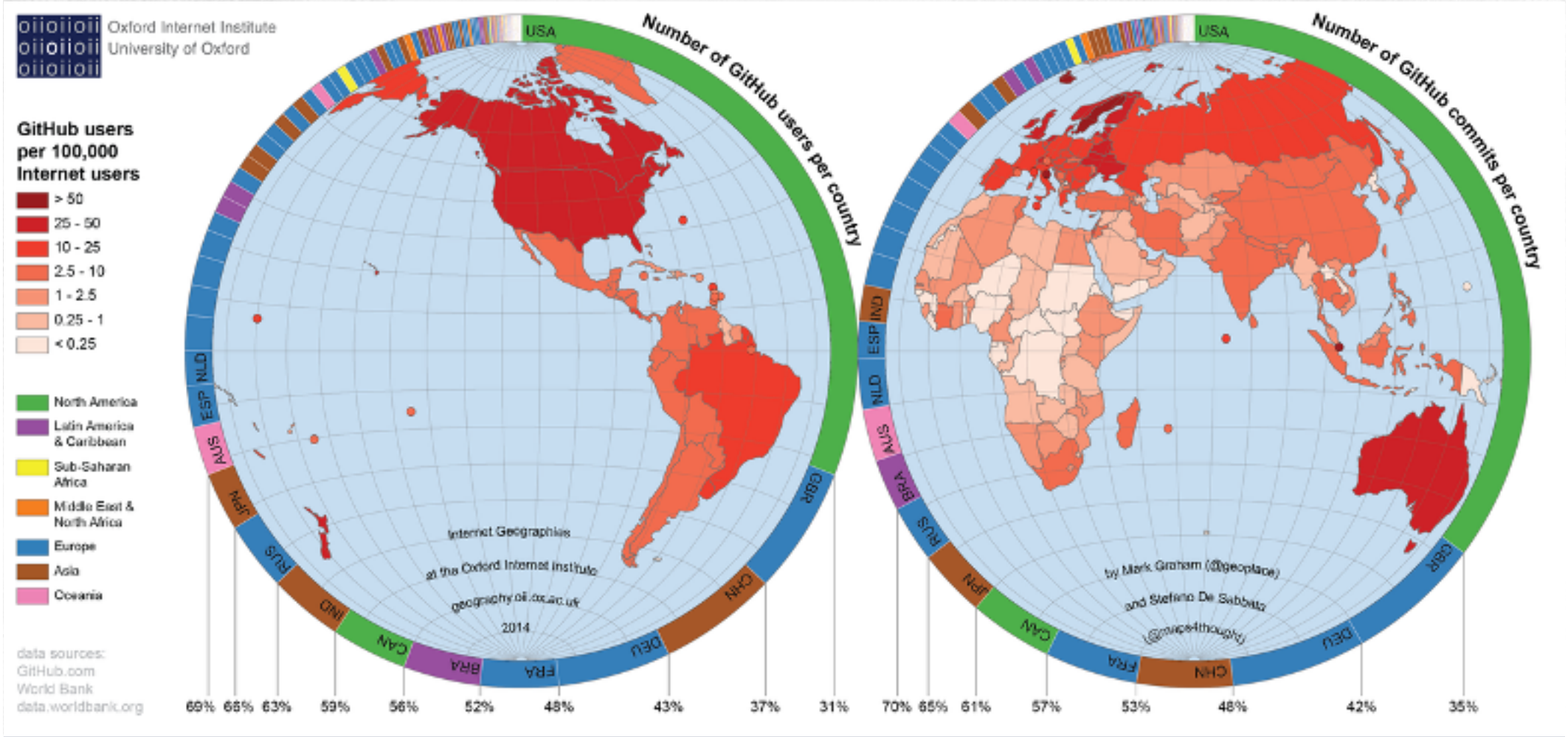
`$ git push [remote] [branch]`

모든 로컬 브랜치의 변경점을 GitHub에 업로드합니다

`$ git pull`

북마크된 원격 브랜치의 기록을 다운로드하여 변경점을 병합합니다

GitHub statistics

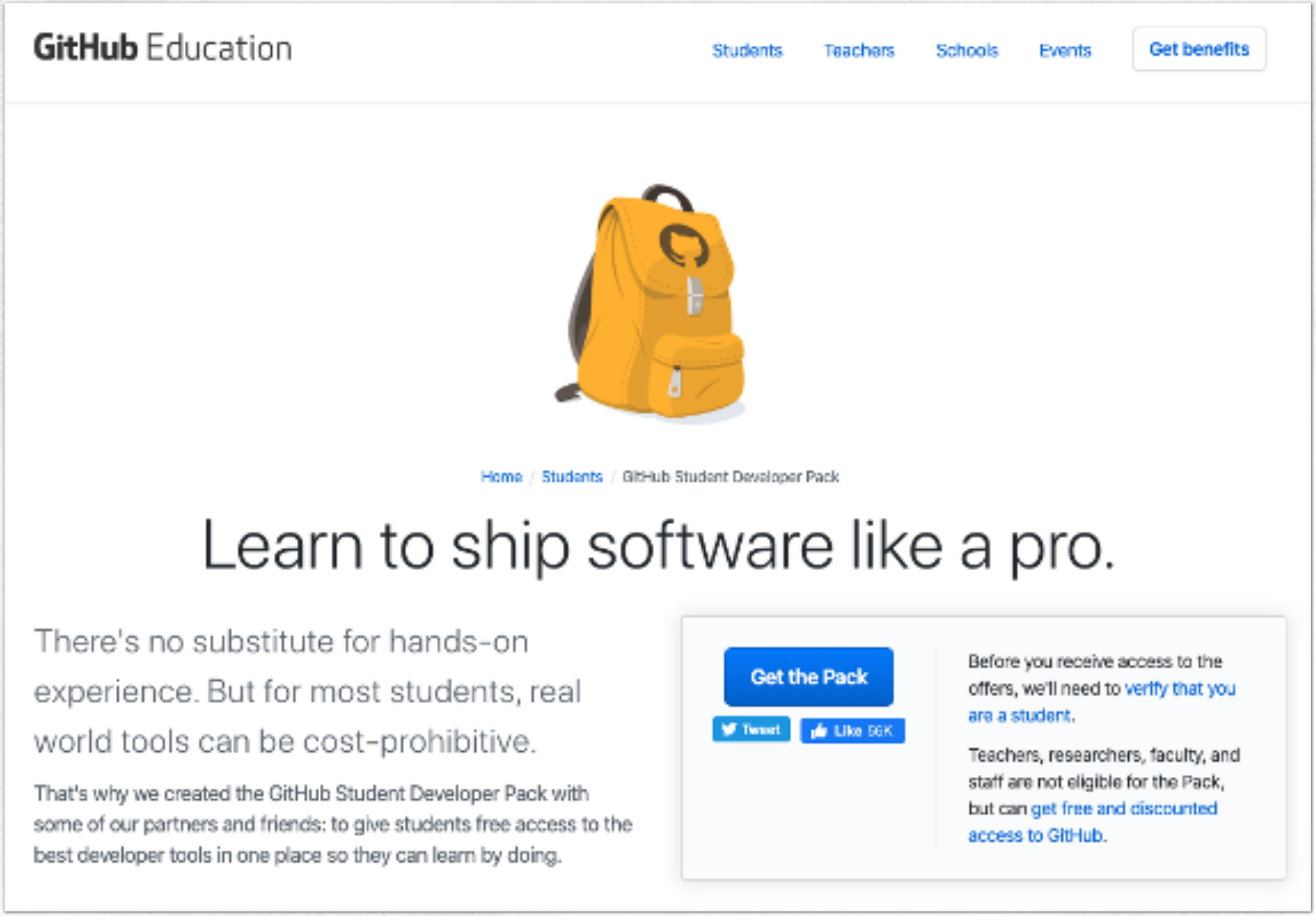


GitHub | Mapping collaborative software

Reference: https://commons.wikimedia.org/wiki/File:Mapping_collaborative_software_on_GitHub.png

GitHub as Programmer SNS & Portfolio Repository

GitHub Student Developer Pack



Reference: https://education.github.com/pack

Summary

- CVS → SVN → Git & GitHub

- GitHub for
  - Access to global open source softwares
  - Make portfolio for your career management
  - Contribution to global projects
  - Enjoy rich student developer benefits

Reference

- GitHub Guides

  https://docs.github.com/en/get-started

- 소셜 코딩으로 이끄는 GitHub 실천기술

  오오츠카 히로키 지음, 제이펍

- GitHub Guides

  https://guides.github.com/

- Cheat Keys in GitHub

  https://education.github.com/git-cheat-sheet-education.pdf

Reference

- GitHub Guides

  https://docs.github.com/en/get-started

- 소셜 코딩으로 이끄는 GitHub 실천기술

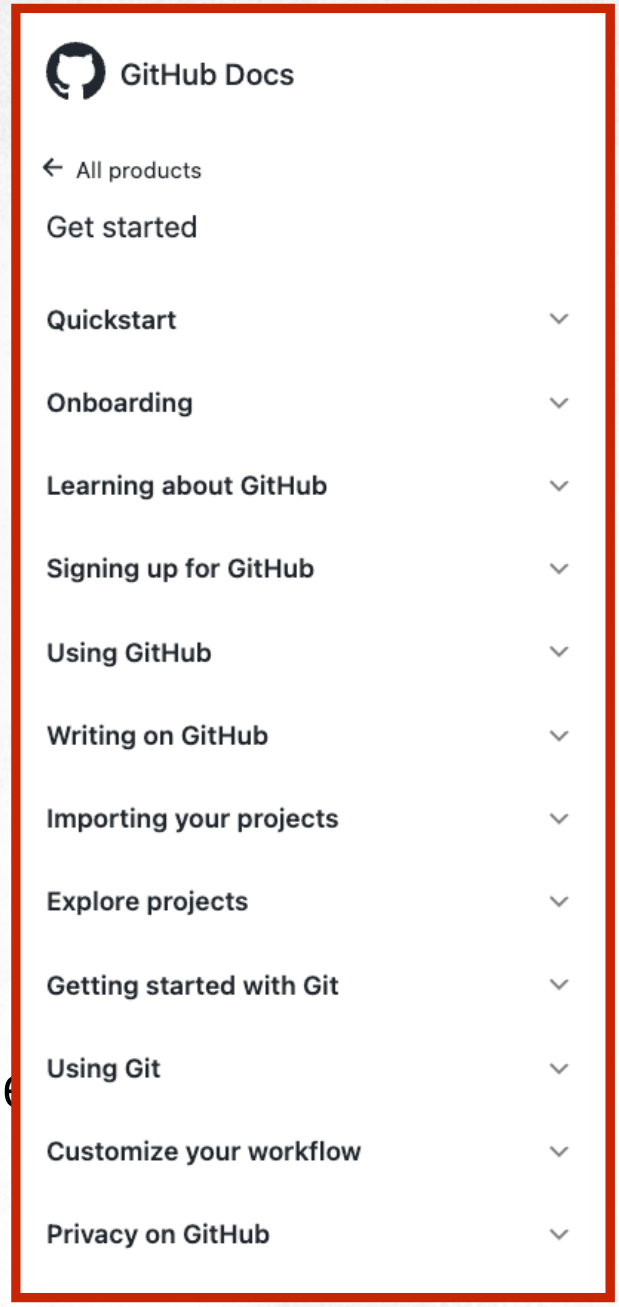
  오오츠카 히로키 지음, 제이펍

- GitHub Guides

  https://guides.github.com/

- Cheat Keys in GitHub

  https://education.github.com/git-cheat-she



Reference


Overview
Release Notes
Help
GitHub Desktop
Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.
Download for macOS
Feeling brave?
Try new features in the Beta Channel before they're released.
Apple silicon?
Download for an Apple silicon Mac. See the Apple docs about Apple vs Intel chips.
Windows?
Download for Windows
By downloading, you agree to the Open Source Applications Terms.

Reference

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.83 is now available! Read about the new features and fixes from September.

OVERVIEW
SETUP
GET STARTED
USER GUIDE
SOURCE CONTROL
Overview
Introduction to Git
Collaborate on GitHub
FAQ
TERMINAL
LANGUAGES
NODE.JS / JAVASCRIPT
TYPESCRIPT
PYTHON
JAVA
C++
C#
DOCKER
DATA SCIENCE
AZURE
REMOTE
DEV CONTAINERS

Working with GitHub in VS Code

Edit

GitHub is a cloud-based service for storing and sharing source code. Using GitHub with Visual Studio Code lets you share your source code and collaborate with others right within your editor. There are many ways to interact with GitHub, for example, via their website at https://github.com or the Git command-line interface (CLI), but in VS Code, the rich GitHub integration is provided by the GitHub Pull Requests and Issues extension.

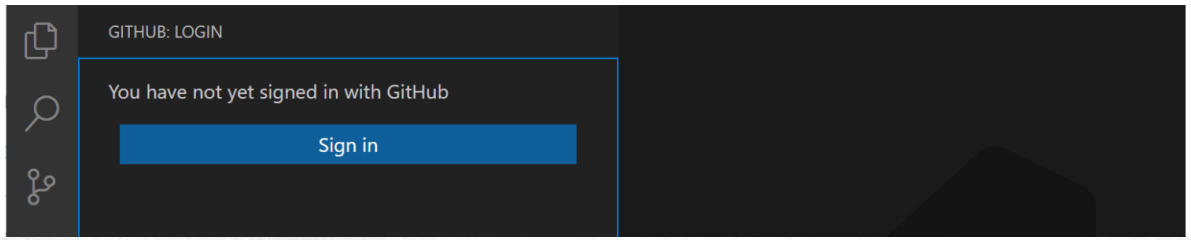
Install the GitHub Pull Requests and Issues extension

To get started with the GitHub in VS Code, you'll need to install Git, create a GitHub account and install the GitHub Pull Requests and Issues extension. In this topic, we'll demonstrate how you can use some of your favorite parts of GitHub without leaving VS Code.

If you're new to source control or want to learn more about VS Code's basic Git support, you can start with the Source Control topic.

Getting started with GitHub Pull Requests and Issues

Once you've installed the GitHub Pull Requests and Issues extension, you'll need to sign in. Follow the prompts to authenticate with GitHub in the browser and return to VS Code.



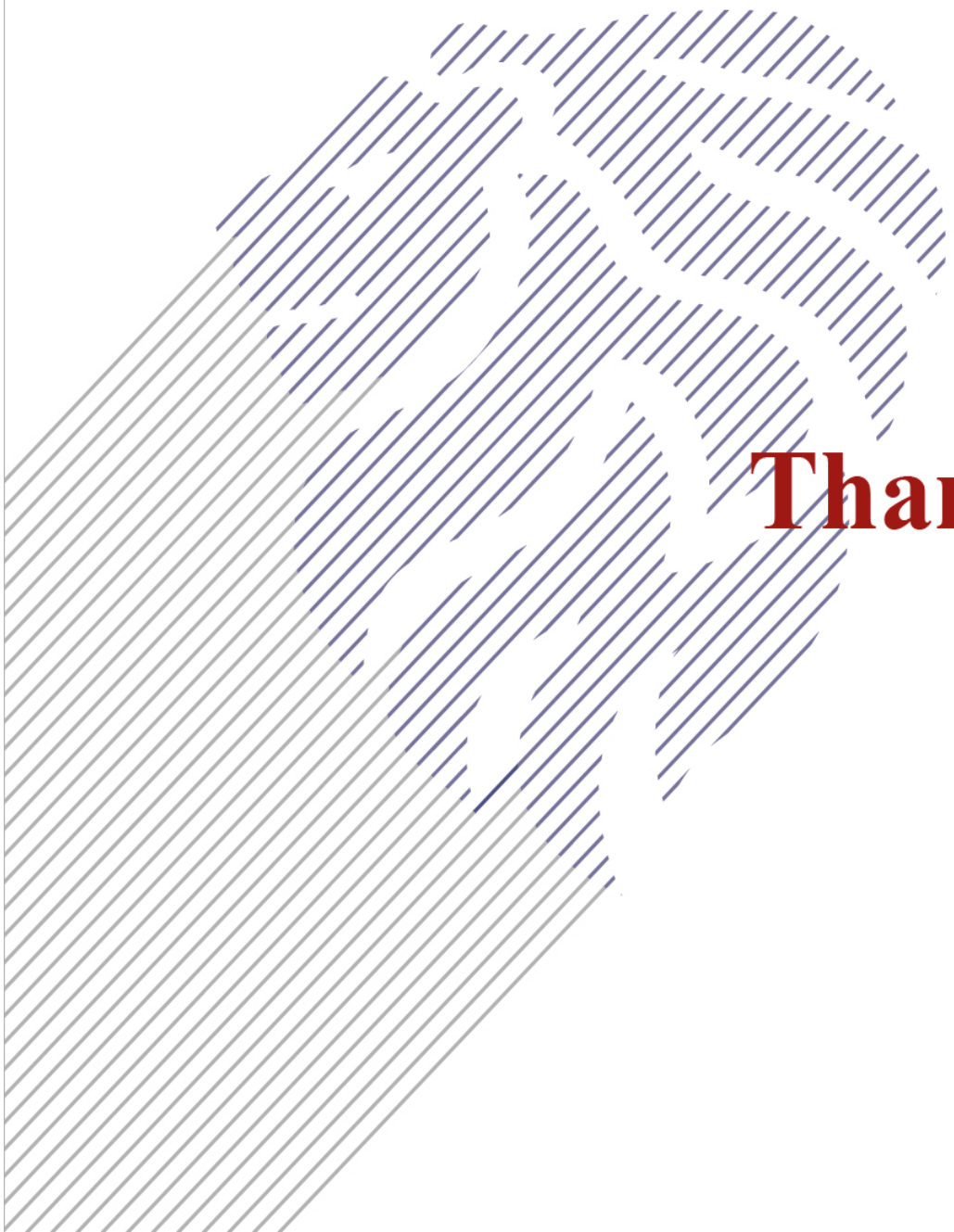
IN THIS ARTICLE

Getting started with GitHub Pull Requests and Issues
Setting up a repository
Editor integration
Pull requests
Issues
GitHub Repositories extension
GitHub Copilot

Subscribe
Ask questions
Follow @code
Request features
Report issues
Watch videos

Homework

- Create your GitHub account
- Experience “Hello World” with GitHub using your GitHub account
  https://guides.github.com/activities/hello-world/
- Submit URL of “Hello World” repository



Thank you

경희대학교
KYUNG HEE UNIVERSITY