

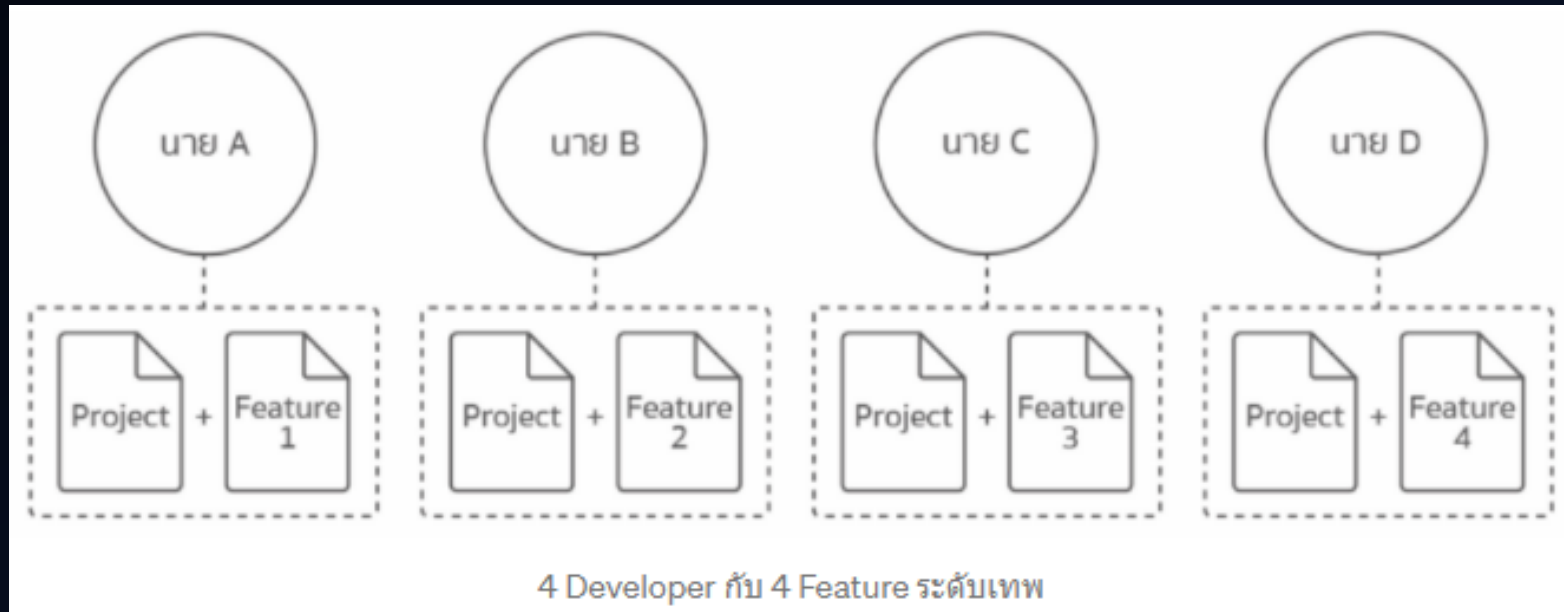
CPE101 – Week09

Git command, Github and Stackoverflow

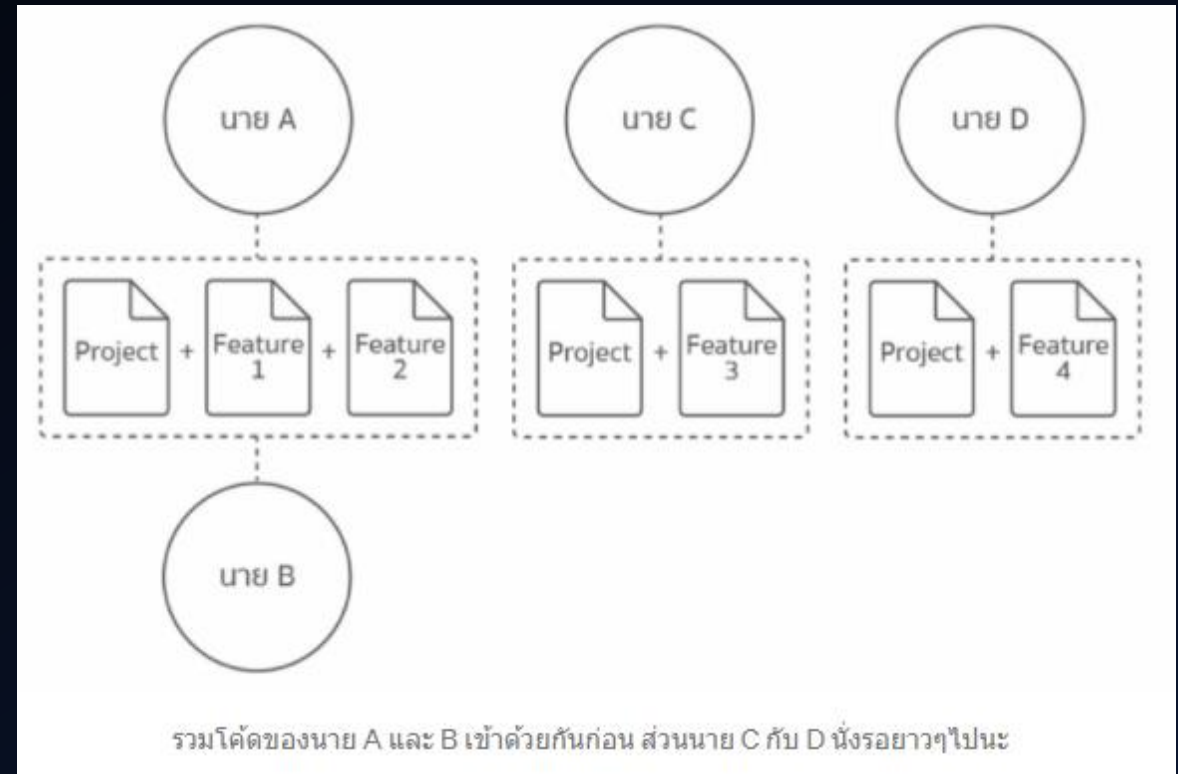
อ.จตุพล ศรีวิลาส (อ.ป๋อ)

Version Control Systems (VCS)

- ในยุคนี้ Version Control ถือเป็นหนึ่งในคุณสมบัติพื้นฐานของ Developer โดยเฉพาะอย่างยิ่งบริษัทที่ต้องทำงานร่วมกันเป็นทีม
- ถ้าต้องพัฒนาโปรเจกขนาดใหญ่ที่มี Dev 4 คนที่กำลังรวม Code ในโปรเจกนี้อยู่ จะใช้วิธีไหนเพื่อเอา Code ที่แต่ละคนเขียน มารวมเข้าด้วยกันในโปรเจก



- วิธีเก่าแก่สุดที่ใช้กัน คือ copy โปรเจกจากแต่ละคนมารวมไว้ในเครื่องเดียวกัน แล้วนั่งรวมหัวกัน โดยมี 1 คนที่เปิด Code ของแต่ละคนขึ้นมา
- สมมติว่าคนๆ นั้นคือ นาย A และนาย B เป็นคนเขียน Code ที่กำลังจะรวมไว้ในโปรเจกเดียวกัน นาย A ต้องถามนาย B ว่า เขียน Code ตรงไหนเพิ่มบ้าง แล้วค่อยนำไปรวมไว้ในโปรเจกหลัก
- ปัญหาที่เกิดขึ้นประจำคือ Code ที่นาย B แก้ไขไปทับซ้อนกับนาย C เพราะนาย C แก้ไขจุดนั้นเหมือนกัน ทำให้นาย B ต้องไปเรียกนาย C มานั่งคุย เพื่อบอกให้นาย A แก้ไขให้ Code ของนาย B และนาย C ทำงานร่วมกันได้



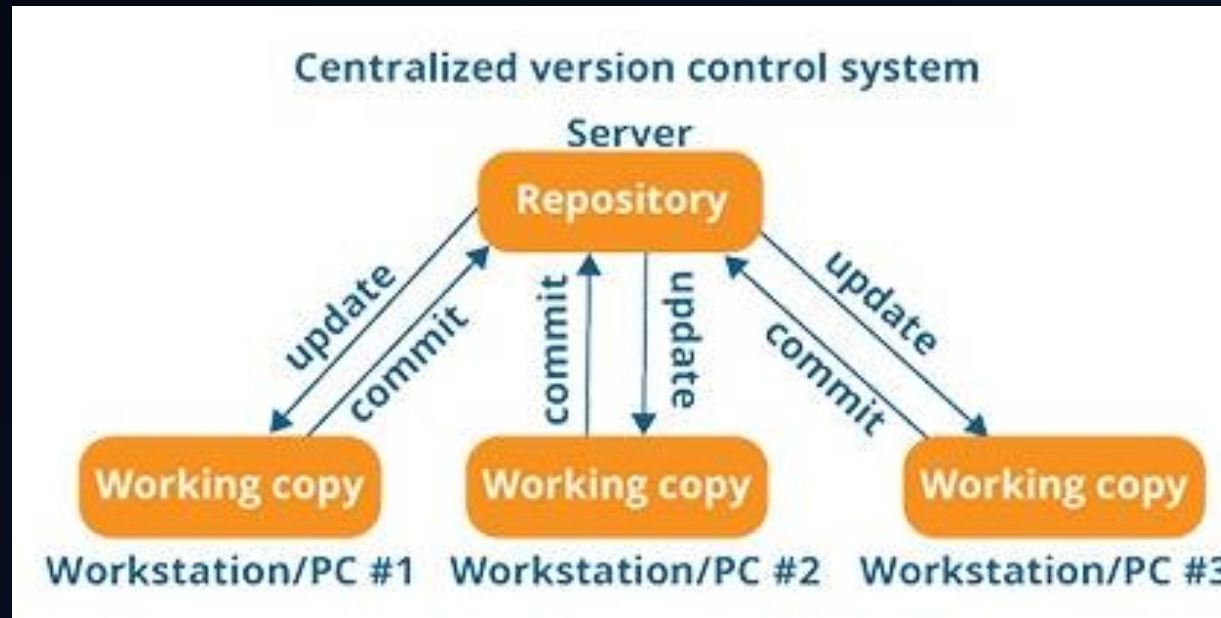
- ยังไม่รวมถึงกรณีที่นาย B จำไม่ได้ว่าตัวเองแก้ไข Code ตรงจุดไหนไปบ้าง เพราะ Feature ที่ทำนั้นใช้เวลาหลายวันและเขียน Code หลายบรรทัด **อ่านแล้วก็รู้เลยว่ามีคามวุ่นวายเกิดขึ้นมากมาย**

จากปัญหาดังกล่าว ทำให้เกิดสิ่งที่เรียกว่า Version Control ขึ้นมา เพื่อควบคุมการเปลี่ยนแปลงของ Code ในโปรเจกต์ โดยประโยชน์ของ Version Control มีดังนี้

- เก็บประวัติการแก้ไข Code ไว้ทุกครั้ง และรู้ว่า Code ตรงไหนใครเป็นคนเพิ่มเข้ามาหรือแก้ไข
- ช่วยรวม Code จากหลายๆ คนเข้าด้วยกันให้ง่ายขึ้น ดูได้ว่า Code เดิมคืออะไร และแก้ไขเป็นอะไร
- เมื่อเกิดปัญหา สามารถติดตามดูประวัติการแก้ไข Code ในแต่ละไฟล์แต่ละบรรทัดได้ง่าย
- เป็น Backup ไปในตัว ไม่ต้องกลัวเวลา Code มีปัญหาแล้วต้อง Rollback กลับไปใช้ Code ชุดเก่า และใช้พื้นที่ในการเก็บข้อมูลน้อยเมื่อเทียบกับการ Backup แบบเก็บทั้งโปรเจกต์ไว้ทุกครั้งที่ทำ การ Backup
- สามารถ Track การทำงานของทุกคนภายในทีมได้จาก History

Centralized Revision Control

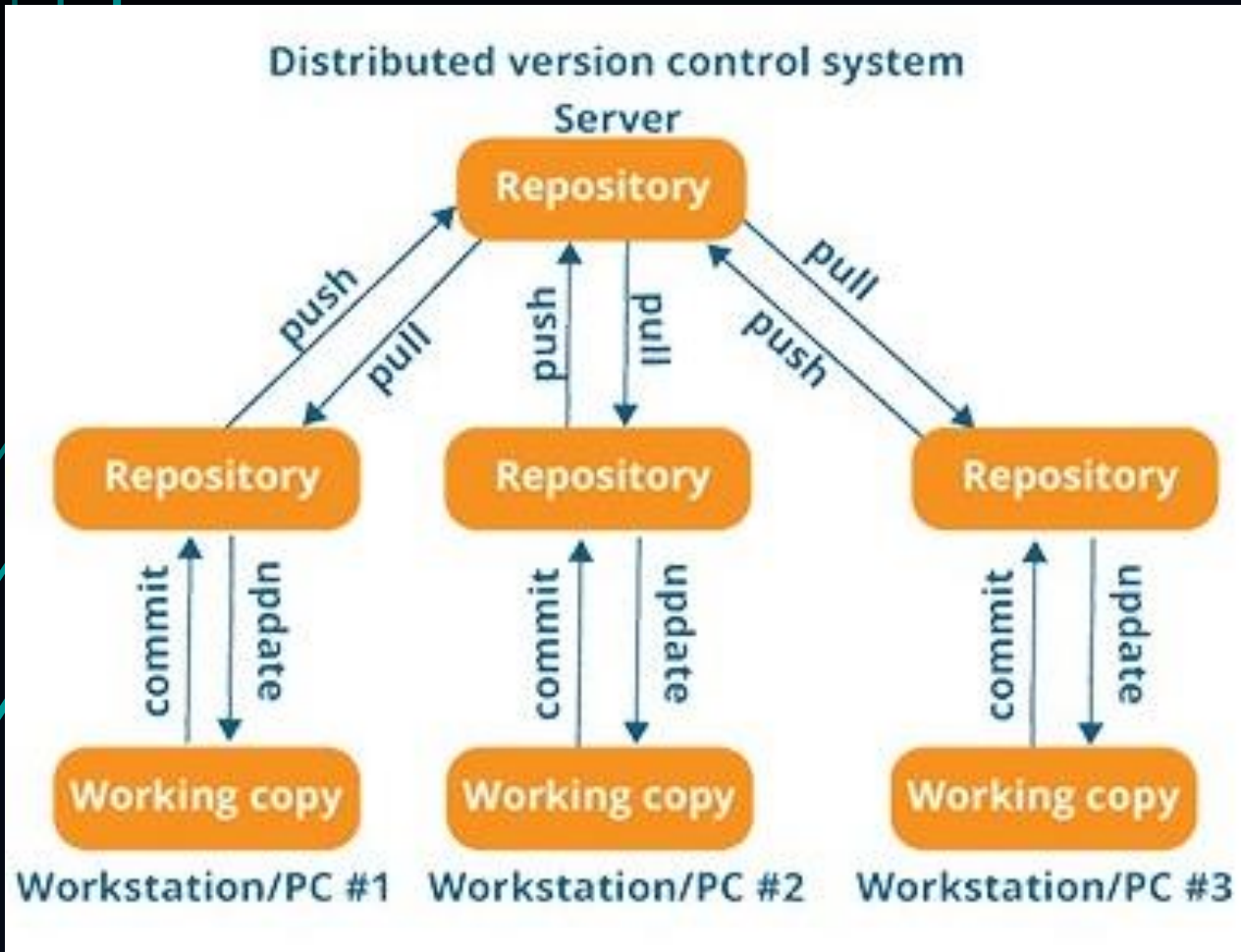
- แต่เดิม software Revision Control จะเป็นระบบ Centralized Revision Control เช่น SVN, CVS หรือ Sourcesafe คือ มี Server เพียงเครื่องเดียวเป็น Repository ในการเก็บ source code และข้อมูล version
- Source code ในแต่ละไฟล์ เครื่อง Repository กับเครื่องของนักพัฒนาแต่ละคนจึงทำงานในรูปแบบ Client-Server
- ข้อเสียของระบบนี้คือ หาก Server มีปัญหา เช่น เครื่องหรือข้อมูลเสียหาย ทีมพัฒนาจะไม่สามารถทำงานต่อได้



Git : Distributed Revision Control

- Git เป็นระบบจัดการกลุ่ม source code ในรูปแบบที่สามารถมีหลาย Repository หรือที่จัดเก็บ source code หลายที่กระจายกันไปเพื่อความสะดวกและรวดเร็วในการทำงานเป็นทีม
- Git ถูกคิดค้นโดย Linus Torvalds สำหรับใช้ภายในทีมพัฒนา kernel ของ Linux ซึ่งปัจจุบัน Git เป็น Open source Software (GNU) ซึ่งสามารถใช้งานได้ในเชิงพาณิชย์แบบไม่ติดลิขสิทธิ์ ทำให้มีการนำไปใช้เป็น Core Engine ของ GitHub
- Git จะแบ่ง Repository ออกเป็น 2 ส่วน คือ
- **Local Repository** ใช้จัดเก็บ source code ทั้งหมดภายในเครื่องนักพัฒนาแต่ละคน
- **Remote Repository** ใช้จัดเก็บ source code ทั้งหมด เป็น Server กลางที่ใช้ร่วมกันในทีมพัฒนาทุกคน
- นักพัฒนาแต่ละคนในทีมจะมี Local Repository อยู่ในเครื่องตนเองซึ่งจะเป็นการดึง (**PULL**) ชุด source code ทั้งหมดจาก Remote Repository มาไว้ในเครื่อง เมื่อมีการแก้ไขและบันทึกการเปลี่ยนแปลง source code จะทำการ (**COMMIT**) ไว้ที่ Local repository เพื่อทำการบันทึกเป็น version ใหม่ที่เครื่องตนเอง เมื่อมีการแก้ไขทั้งหมดแล้ว จึงทำการส่ง (**PUSH**) ชุดไฟล์ version ใหม่ไปยัง Remote Repository เพื่อเป็นการ update ชุดไฟล์ใหม่นั้นให้นักพัฒนาในทีมสามารถดึงไปใช้ได้

Git : Distributed Revision Control



การใช้งาน Git

- มีอยู่ 2 แบบหลักๆ คือ
- **Command Line** : พิมพ์คำสั่งของ Git จาก Terminal หรือ Command Prompt โดยตรง
- **GUI** : ใช้โปรแกรม Git GUI เช่น SourceTree, TortoiseGit หรือ Github Desktop เป็นต้น

```
MINGW64/c/Users/pocks/c-project
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   codena.txt

no changes added to commit (use "git add" and/or "git commit -a")

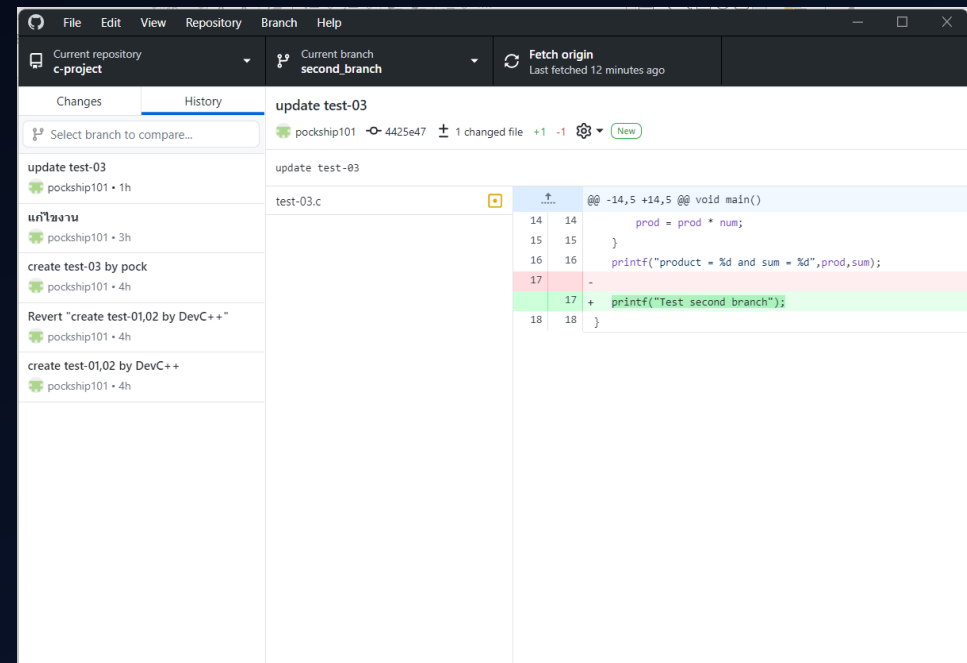
pocks@AREA-99 MINGW64 ~/c-project (master)
$ git log
commit 527726f3d6a7a8c303d11d9f9f3a7bb3226e2488 (HEAD -> master, origin/master)
Author: pockship101 <93063926+pockship101@users.noreply.github.com>
Date:   Sun Oct 24 15:07:14 2021 +0700

    add my age

commit b152b509a2074565d4f2208d99559d483dbad7f1
Author: pockship101 <93063926+pockship101@users.noreply.github.com>
Date:   Sun Oct 24 15:03:25 2021 +0700

    test

pocks@AREA-99 MINGW64 ~/c-project (master)
$
```



คำต่างๆ ที่ต้องรู้จัก เมื่อใช้งาน Git

- **Repository** : เวลาพัฒนาโปรแกรมจะต้องสร้างสิ่งที่เรียกว่าโปรเจก ซึ่งการสร้างโปรเจกสำหรับใช้งาน Git จะเรียกว่า Repository ซึ่ง Repository ของ Git คือ Folder ที่ใช้เก็บข้อมูล ส่วนใหญ่นิยมเก็บโปรเจก 1 ตัวต่อ 1 Repository
- **Clone** : เวลาที่ต้องการ Sync Repository จาก Remote มาลงที่เครื่องตนเอง
- **Commit** : เวลาที่มีข้อมูลที่แก้ไขเสร็จแล้ว ต้องการทำการ backup เก็บไว้ใน VCS จะเรียกว่า Commit โดยสามารถเลือกได้ว่าเอาไฟล์ไหนบ้าง ไม่จำเป็นต้องเลือกทุกไฟล์ ในการ Commit แต่ละครั้ง จะต้องใส่ Commit Message เพื่ออธิบายรายละเอียดใน Commit นั้นๆ ว่าทำอะไรไปบ้าง เพื่อที่มาดูภายหลังจะสามารถอ่านได้จาก Commit Message
- **Push** : เวลาที่มี Commit อยู่ในเครื่องและต้องการจะ Sync ขึ้นไปเก็บไว้ใน Remote
- **Pull** : เวลา Sync จาก Remote เพื่อดึงข้อมูล Commit ใหม่ๆ มาเก็บไว้ในเครื่องตนเอง

Git

การใช้งาน Git Command [https://git-scm.com]

- **Git** คือ Version Control ตัวหนึ่งซึ่งเป็นระบบที่มีหน้าที่ในการจัดการการเปลี่ยนแปลงของไฟล์ในโปรเจก มีการ Backup Code สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชันต่างๆ ของโปรเจกที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆ ใครเป็นคนเพิ่มหรือแก้ไข หรือดูว่าไฟล์นั้นๆ ถูกเขียนโดยใครบ้าง



```
MINGW64:/c/Users/pocks/c-project
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   codena.txt

no changes added to commit (use "git add" and/or "git commit -a")

pocks@AREA-99 MINGW64 ~/c-project (master)
$ git log
commit 527726f3d6a7a8c303d11d9f9f3a7bb3226e2488 (HEAD -> master, origin/master)
Author: pockship101 <93063926+pockship101@users.noreply.github.com>
Date:   Sun Oct 24 15:07:14 2021 +0700

    add my age

commit b152b509a2074565d4f2208d99559d483dbad7f1
Author: pockship101 <93063926+pockship101@users.noreply.github.com>
Date:   Sun Oct 24 15:03:25 2021 +0700

    test

pocks@AREA-99 MINGW64 ~/c-project (master)
$
```

https://git-scm.com

git --fast-version-control

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

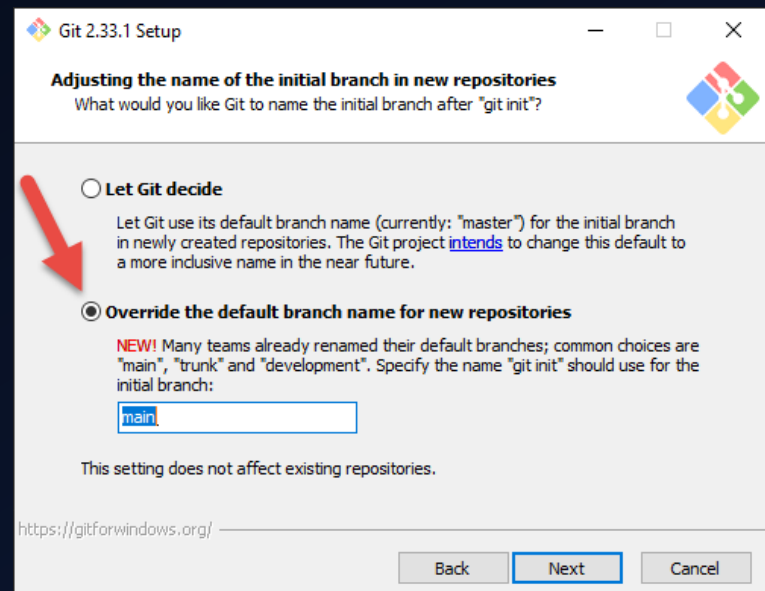
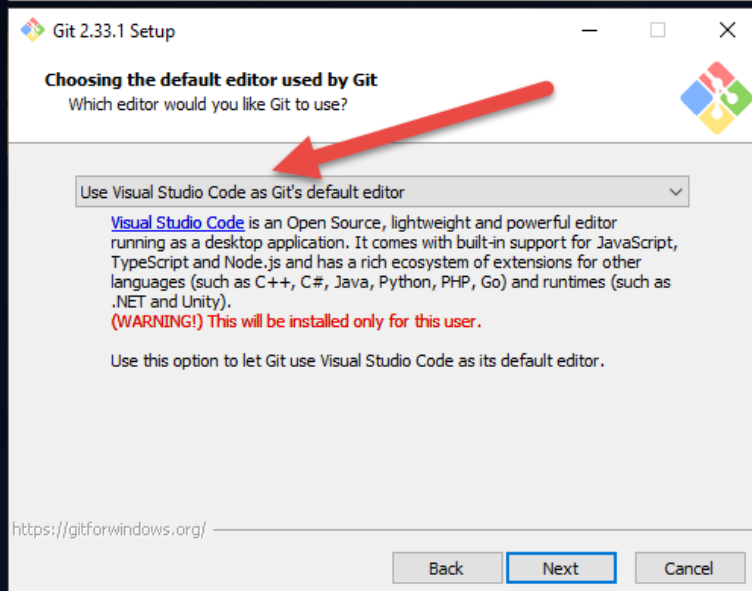
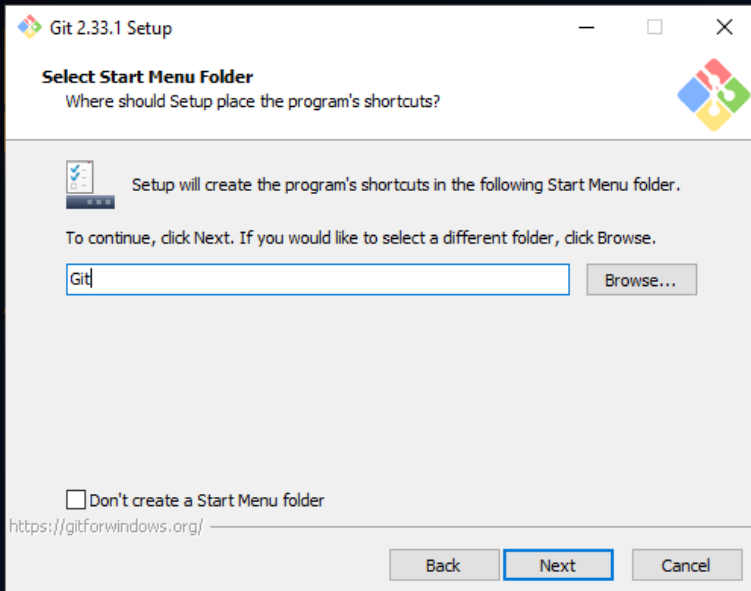
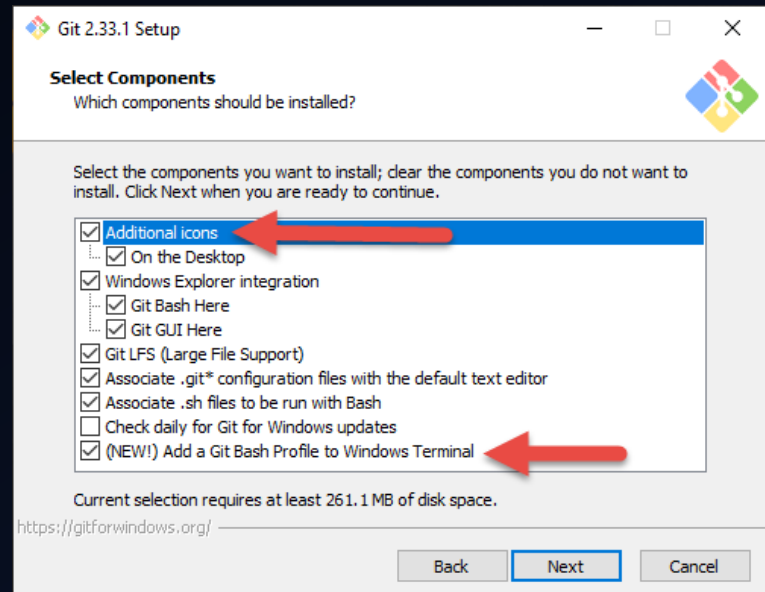
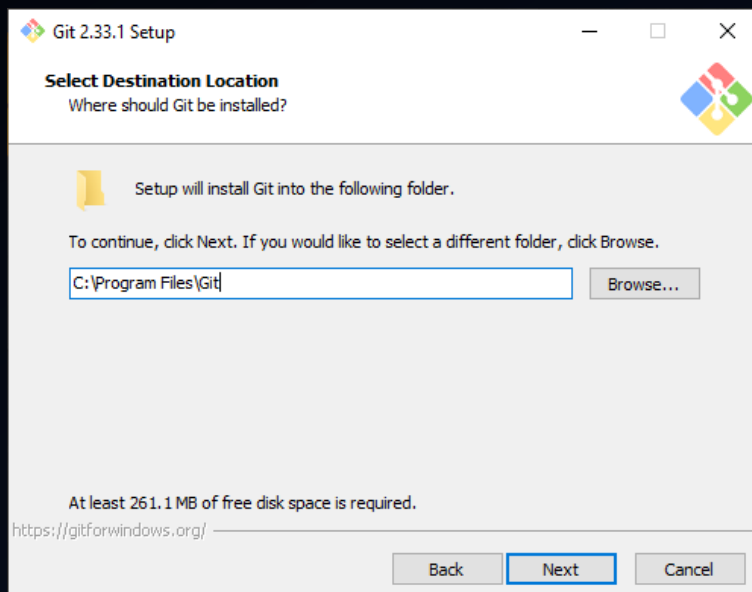
Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.33.1
Release Notes (2021-10-12)
Download for Windows

การติดตั้ง Git



การติดตั้ง Git

Git 2.37.3 Setup

Adjusting your PATH environment
How would you like to use Git from the command line?

☐ Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

☐ Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

<https://gitforwindows.org/>

Back Next Cancel

Git 2.33.1 Setup

Choosing the SSH executable
Which Secure Shell client program would you like Git to use?

☒ Use bundled OpenSSH
This uses ssh.exe that comes with Git.

☐ Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

<https://gitforwindows.org/>

Back Next Cancel

Git 2.33.1 Setup

Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?

☒ Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.

☐ Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

<https://gitforwindows.org/>

Back Next Cancel

Git 2.33.1 Setup

Configuring the line ending conversions
How should Git treat line endings in text files?

☒ Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

☐ Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

☐ Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

<https://gitforwindows.org/>

Back Next Cancel

Git 2.33.1 Setup

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

☒ Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via "winpty" to work in MinTTY.

☐ Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

<https://gitforwindows.org/>

Back Next Cancel

Git 2.33.1 Setup

Choose the default behavior of 'git pull'
What should 'git pull' do by default?

☒ Default (fast-forward or merge)
This is the standard behavior of 'git pull': fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

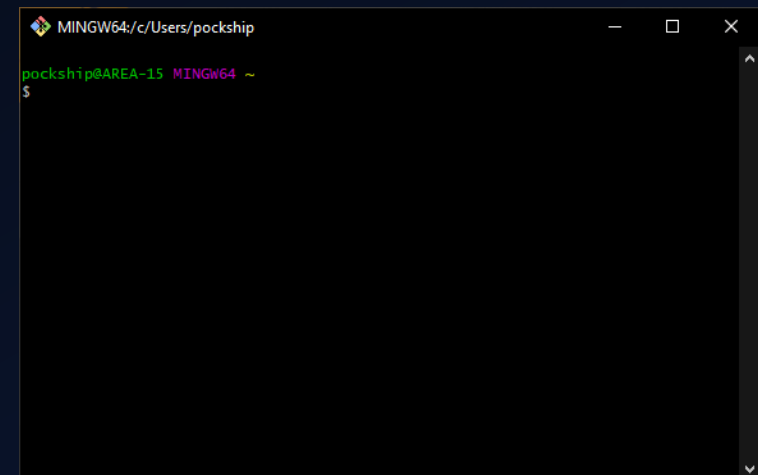
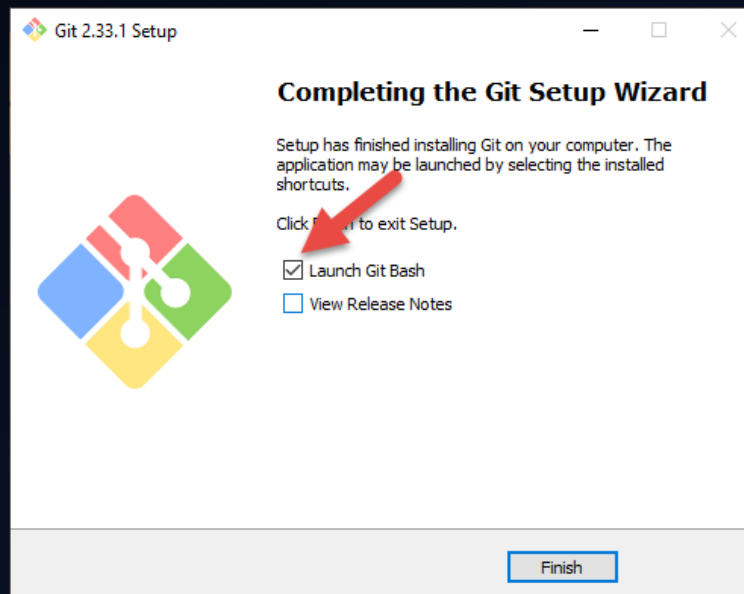
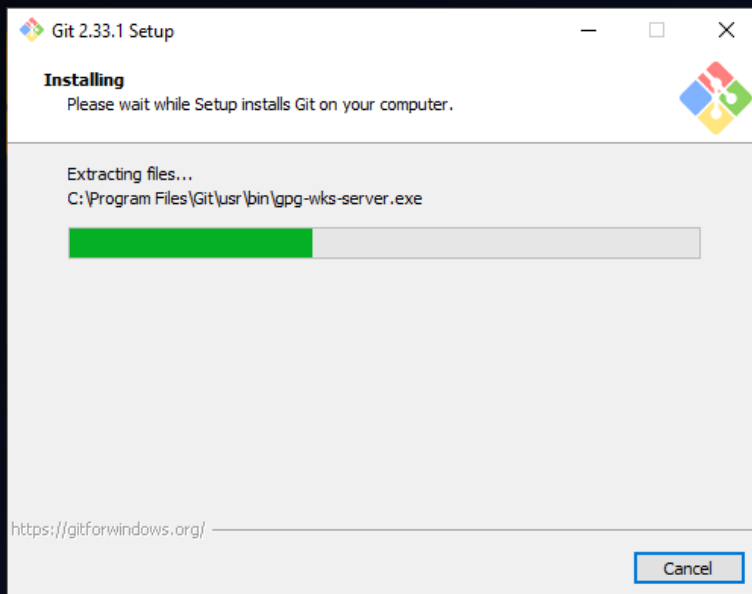
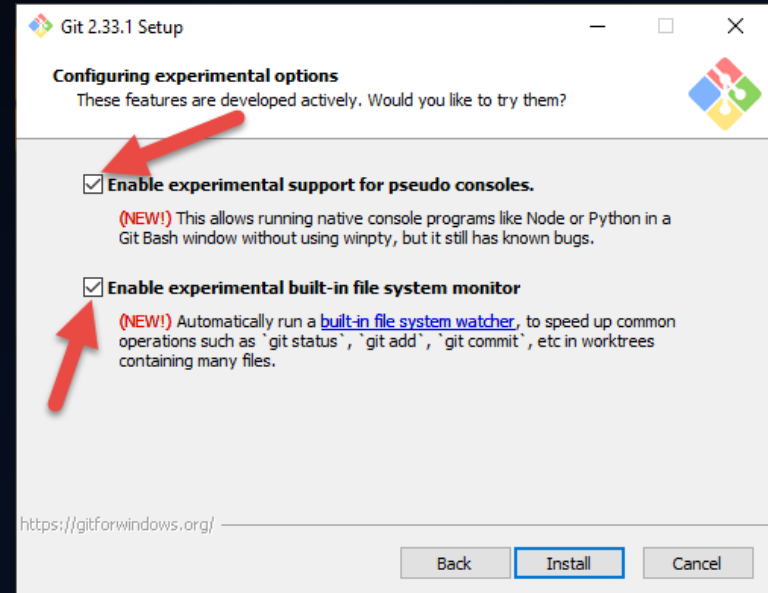
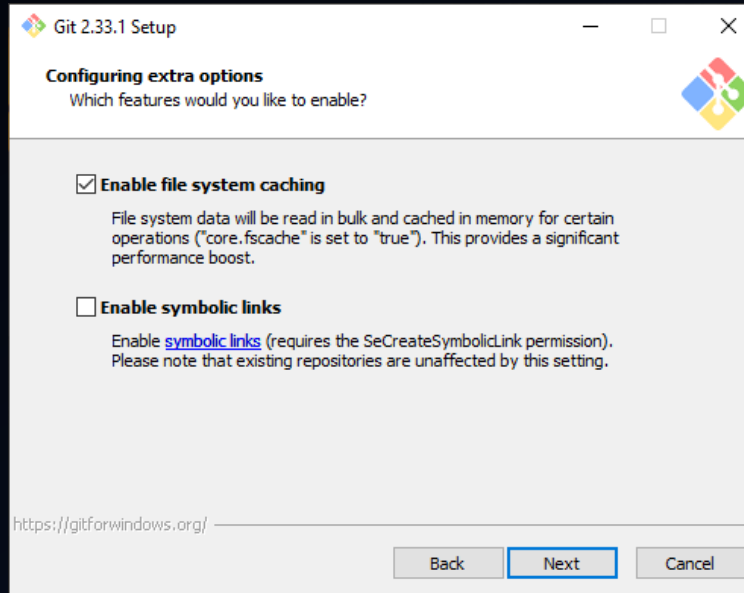
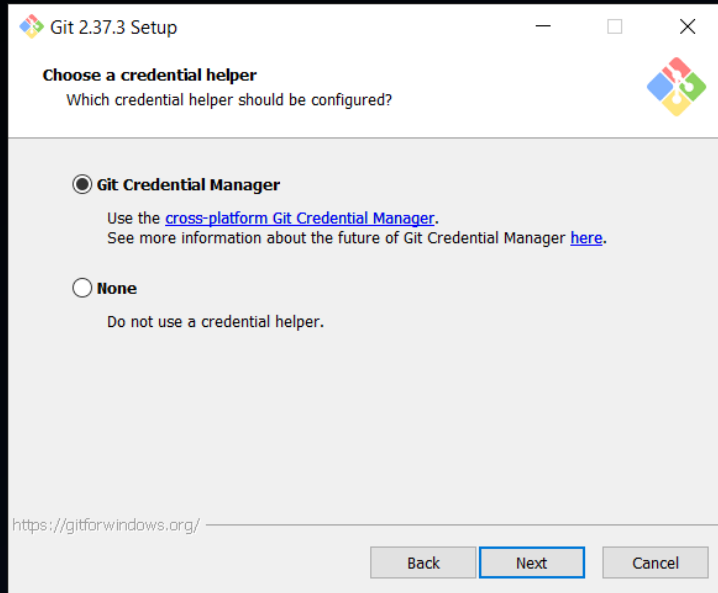
☐ Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

☐ Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible.

<https://gitforwindows.org/>

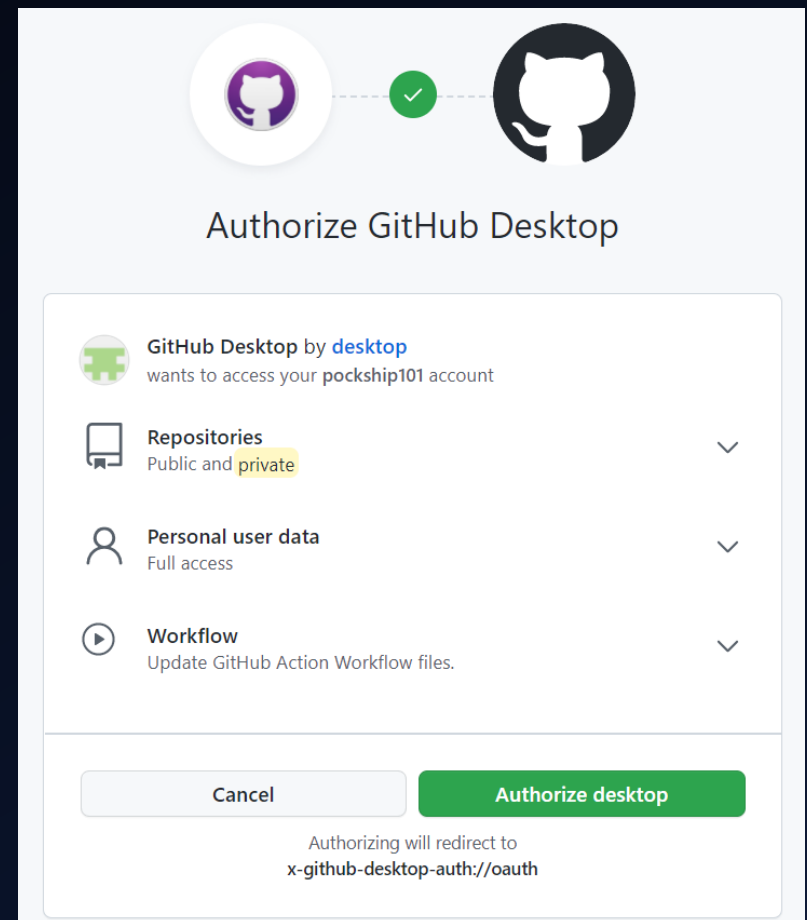
Back Next Cancel

การติดตั้ง Git

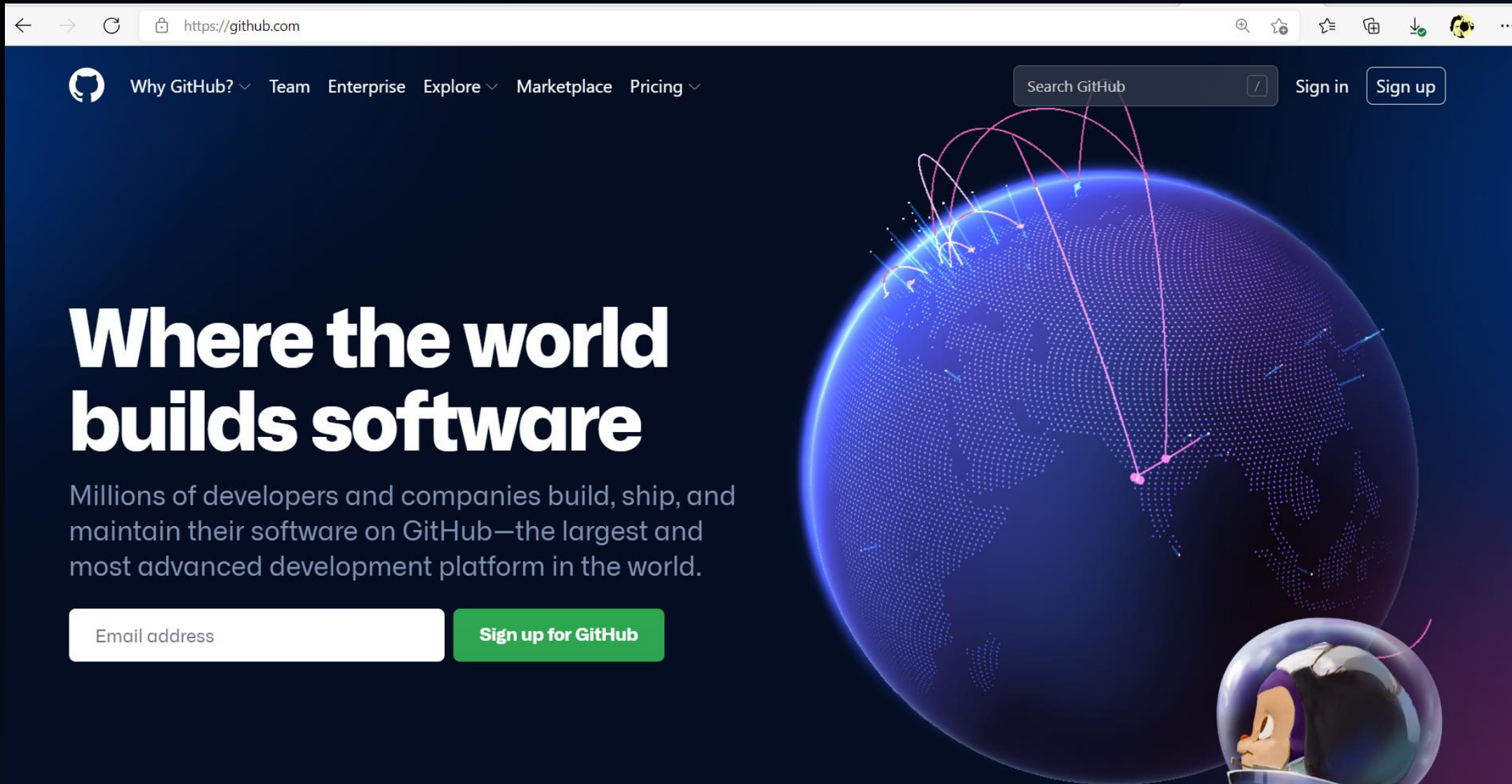


GitHub

- GitHub เป็น Webserver ที่ให้บริการในการฝากไฟล์ Git (ทั่วโลกนิยมใช้ในการเก็บ Project Open Source ต่างๆ ไม่ว่าจะเป็น Bootstrap, Rails, Node.js, Angular เป็นต้น)
- ขั้นตอนแรก ต้องทำการสมัครสมาชิกกับ GitHub จากนั้นจึงสามารถที่จะสร้าง Repository และไปฝากไว้บน GitHub ได้
- จุดเด่นของ GitHub คือใช้ฟรี และสร้าง Repository ได้ไม่จำกัด แต่ต้องเป็น Public Repository เท่านั้น หากอยาก Private ต้องเสียเงิน



การสมัครใช้งาน GitHub



การสมัครใช้งาน GitHub

https://github.com/join



Product ▾ Solutions ▾ Open Source ▾ Pricing

You're almost done!

We sent a launch code to `pockship@hotmail.com`

→ Enter code

Didn't get your email? [Resend the code](#) or [update your email address](#).

Your GitHub launch code



ถึง: คุณ

๑ 13/9/2022 11:43



Here's your GitHub launch code, @ramza1315!



Continue signing up for GitHub by entering the code below:



Open GitHub

Join GitHub

First, let's create your user account

Username *

ramza1315



Email address *

pockship@hotmail.com



Password *

.....



Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

[Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Welcome to GitHub



Welcome to GitHub

We are glad you're here.

How many team members will be working with you?

This will help us guide you to the tools that are best suited for your projects.

Just me

2 - 5

5 - 10

10 - 20

20 - 50

50+

Are you a student or teacher?



Student

Teacher

Continue

เลือก Features ของ GitHub

เลือกการใช้งานแบบฟรี










The tools you need to build what you want.

Soup to nuts, GitHub has it all.

What specific features are you interested in using?


Select all that apply so we can point you to the right GitHub plan.





- ☐  Collaborative coding
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- ☐  Automation and CI/CD
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- ☐  Security
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- ☐  Client Apps
GitHub Mobile, GitHub CLI, and GitHub Desktop.
- ☐  Project Management



Real-world tools, engaged students.






GitHub gives teachers free access to industry-standard tools for training developers.

Free 


-  **Unlimited public/private repositories**
-  **2,000 CI/CD minutes/month**
Free for public repositories
-  **500MB of Packages storage**
Free for public repositories
-  **Community support**

Get additional teacher benefits



GitHub Team

-   **Protect your branches**
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
-  **Draft pull requests**
-  **Required reviewers**
-  **3,000 CI/CD minutes/month**
Free for public repositories

หน้า Home ของ GitHub



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.


[Create repository](#) [Import repository](#)


Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

The home for all developers — including you.

Welcome to your personal dashboard, where you can find an introduction to how GitHub works, tools to help you build software, and help merging your first lines of code.


 Start writing code



Start a new repository

Collaborate on code with others and track your work in a repository.


[Create a new repository](#)



Create your profile README



Create a file in a repository that tells the GitHub community who you are.

[Create a README](#)



Contribute to an existing repository

[Find repos that need your help](#)



 **Universe 2022** 

Let's build from here

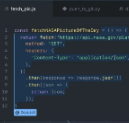
Join the global developer event for cloud, security, community, and AI.

Register today and get a 20% off early bird discount.

[Register now](#)

 **GitHub Copilot** 

Get suggestions for lines of code and entire functions in real-time



[Learn more about Copilot](#)

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

[Create repository](#) [Import repository](#)

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

[Following](#) [For you](#) [Beta](#)

Introduce yourself

The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:

ramza1315 / README.md


```
1 - 🙋 Hi, I'm @ramza1315
2 - 🤖 I'm interested in ...
3 - 🧑 I'm currently learning ...
4 - 🤝 I'm looking to collaborate on ...
5 - 📬 How to reach me ...
6
```


[Dismiss this](#) [Continue](#)


Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you [watch](#) or [star](#) and people you [follow](#).


[Explore GitHub](#)

 **ProTip!** The feed shows you events from people you [follow](#) and repositories you [watch](#) or [star](#).

 [Subscribe to your news feed](#)


 Use tools of the trade

Write code in your web browser




Use the [github.dev web-based editor](#) from your repository or pull request to create and commit changes.

Install a powerful code editor



Visual Studio Code is a multi-platform code editor optimized for building and debugging software.


Set up your local dev environment



After you set up Git, simplify your dev workflow with [GitHub Desktop](#), or bring GitHub to the command line.

Latest changes

- 15 hours ago
Merge commits now created using the merge-ort strategy
- 4 days ago
New Audit Log events and event context



การใช้งาน GitHub Desktop



[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 Windows (64bit)

Download for [macOS](#) or [Windows \(msi\)](#)

By downloading, you agree to the [Open Source Applications Terms](#).

FileEditViewRepositoryBranchHelp

Current repository
c-project

Current branch
second_branch

Fetch origin
Last fetched 12 minutes ago

ChangesHistory

Select branch to compare...

update test-03
pockship101 • 1h

แก้ไขงาน
pockship101 • 3h

create test-03 by pock
pockship101 • 4h

Revert "create test-01,02 by DevC++"
pockship101 • 4h

create test-01,02 by DevC++
pockship101 • 4h

update test-03
pockship101 4425e47 1 changed file +1 -1 New

update test-03

test-03.c

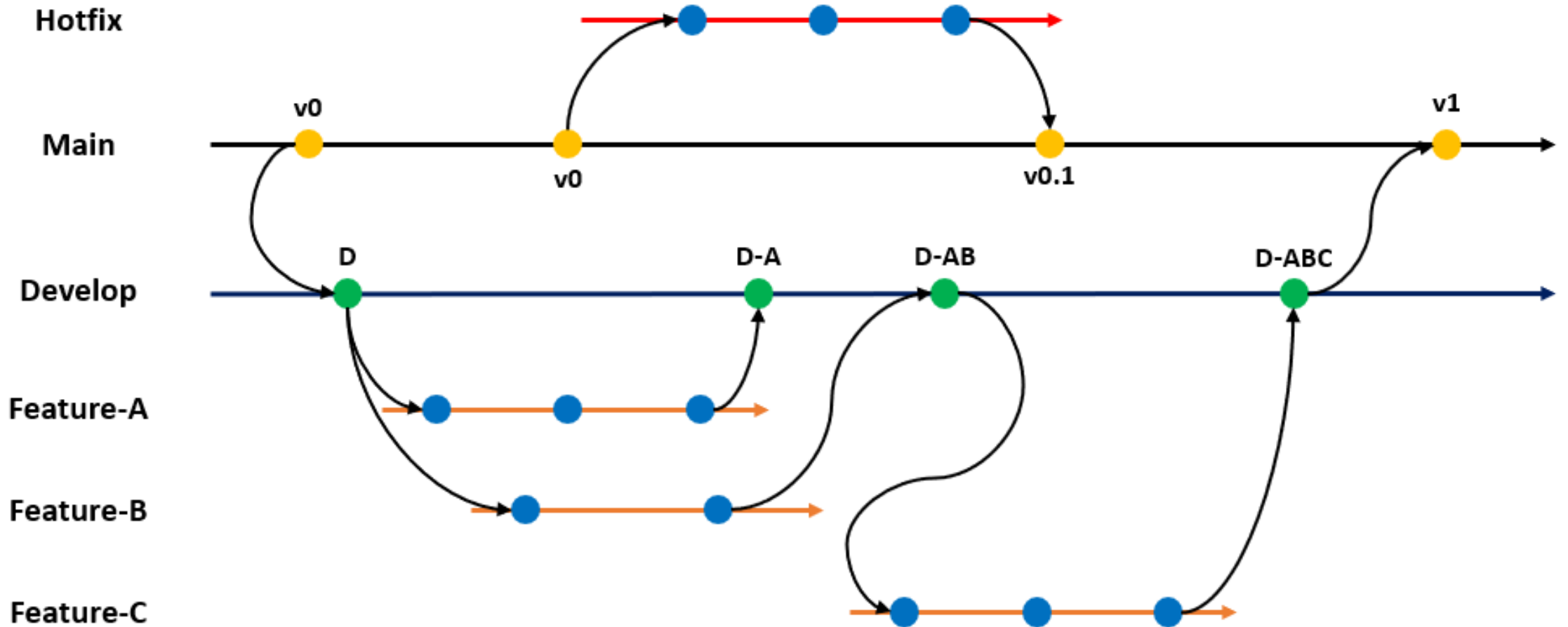
↑...

@@ -14,5 +14,5 @@ void main()
14 14 prod = prod * num;
15 15 }
16 16 printf("product = %d and sum = %d",prod,sum);
17 -
17 + printf("Test second branch");
18 18 }

Branch และ Main (Master) ในระบบ Git

- Branch ในระบบ Git หมายถึง ไฟล์ซอร์สโค้ดชุดหนึ่ง ซึ่งมีคุณสมบัติหรือ feature การทำงานเฉพาะ โดยไฟล์ซอร์สโค้ดชุดนั้นจะมีการระบุ version ของตัวเองในแต่ละ Branch นั้นๆ และเมื่อมีการพัฒนาเพิ่มเติมกับโค้ดชุดนั้น ก็จะมีการระบุ version เพิ่มขึ้นไปเรื่อยๆ จึงเรียกว่า Branch เปรียบได้กับกิ่งของต้นไม้ที่มีการงอกเป็น version เพิ่มขึ้นเรื่อยๆ
- โดยเริ่มแรกไฟล์ซอร์สโค้ดที่นำมาเก็บไว้ในระบบ Git จะต้องมีส่วน Default ซึ่งเหมือนเป็น version เริ่มแรกของซอร์สโค้ดโปรแกรมที่กำลังพัฒนา รวมไปถึงซอร์สโค้ดชุดหลักที่เป็นการรวมคุณสมบัติหรือ feature ต่างๆ จากทุก Branch จะเรียกซอร์สโค้ดชุดนี้ของ Repository ว่า Main (หรือ Master ในชื่อเรียกเดิม)
- ซอร์สโค้ดที่เป็นชุด Main จะใช้เป็น version หลักของซอร์สโค้ด ซึ่งนักพัฒนาสามารถเรียกดูไปทำงานที่ Local Repository ของตัวเอง เมื่อพัฒนาเสร็จแล้วสามารถนำไปสร้างเพิ่มเป็น Branch ใหม่ของตัวเองเพื่อเพิ่มการทำงานในซอร์สโค้ดไฟล์นั้นๆ
- การทำ Branch จะทำให้ใช้ประโยชน์ของหลักการ Distributed Revision Control เนื่องจากแต่ละคนแยกไปทำงาน เช่น การพัฒนาคนละ Module ที่มีการทำงานคนละอย่าง แต่มีการอ้างอิง Library หรือซอร์สโค้ดของการทำงานหลัก (Core features) จาก Main โดยที่ Main อาจจะมีการอัปเดต version เพิ่มขึ้นเรื่อยๆ จากนั้นนำแต่ละ Branch กลับมารวมกันด้วยการ Merge กลับมายัง version Main เพื่อรวม feature จากทุก Branch

Branch และ Main (Master) ในระบบ Git



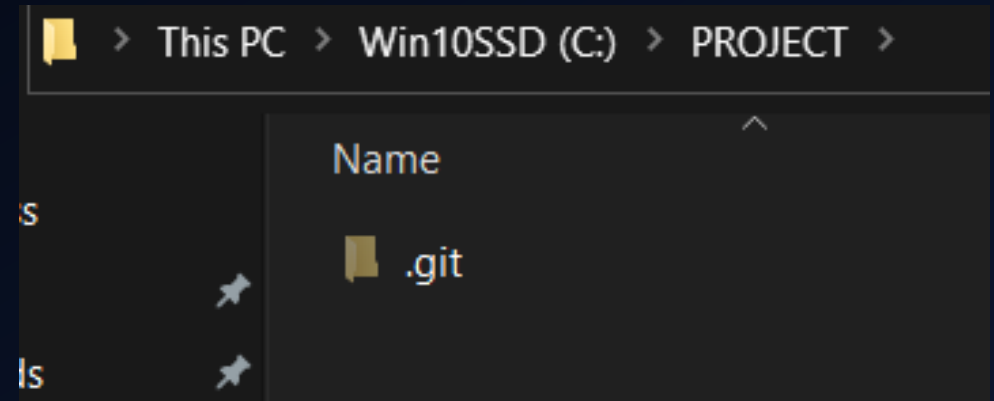
ขั้นตอนการสร้าง Repository และนำ Code ขึ้นระบบ Git

การสร้าง Local Repository ใหม่

คู่มือการใช้งาน Git : <https://git-scm.com/book/en/v2>

- ในการเริ่มสร้าง Local Repository ที่เครื่องตนเอง จะใช้คำสั่ง init โดยเปิด Git Bash จากนั้นเปลี่ยนไป Directory ที่ต้องการสร้างเป็น Local Repository ในที่นี้จะใช้ C:\PROJECT

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 ~  
$ cd c:  
  
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c  
$ mkdir PROJECT  
  
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c  
$ cd PROJECT  
  
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT  
$ git init  
Initialized empty Git repository in C:/PROJECT/.git/  
  
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT (main)  
$
```



ขั้นตอนการสร้าง Repository และนำ Code ขึ้นระบบ Git

- เมื่อทำการสร้าง Local Repository เสร็จแล้ว จะพบว่า มี Directory แบบ hidden ชื่อ **.git** เกิดขึ้นใน Directory C:\PROJECT ซึ่งเป็น Directory ที่ใช้เก็บ configuration และข้อมูล version ของไฟล์และ Directory ย่อยภายใต้ตำแหน่งนั้นของ Git นั่นเอง

การตั้งค่าชื่อผู้ใช้ให้กับ Local Repository (git config)

- ก่อนเริ่มใช้งาน Local Repository จะต้องทำการตั้งค่า config โดยใช้ git config ซึ่งจะมีผลต่อการเชื่อมต่อไปยัง Remote Repository ดังนี้

```
git config (--global) <ชื่อ attribute> <ค่า attribute>
```

- user.name** ใช้กำหนดชื่อผู้ใช้งาน Repository
- user.email** กำหนด email ของผู้ใช้ (email ที่ใช้สมัคร GitHub)
- user.password** กำหนด password ของ account ที่ Remote Repository ซึ่งหากไม่ได้กำหนดค่าในส่วนนี้ Git จะสอบถามอีกครั้งตอนทำขั้นตอน Push หรือ Pull กับ Remote Repository
- ถ้าใส่ **--global** ทุกๆ Local Repository บนเครื่องจะใช้ค่า config นี้ เป็นค่าเดียวกันหมด

ขั้นตอนการสร้าง Repository และนำ Code ขึ้นระบบ Git

- ในส่วนนี้ จะกำหนด user.name เป็นชื่อตนเอง ดังนี้

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT (main)  
$ git config --global user.name "pockship"
```

การเพิ่มไฟล์ source code เข้าสู่ Local Repository (stage/untracked & git add)

- ทำการเพิ่มไฟล์ให้ Local Repository โดยใช้คำสั่ง add ดังนี้

```
git add <ชื่อไฟล์ หรือ Directory>
```

- การใช้คำสั่ง add เป็นการทำให้ไฟล์หรือ directory อยู่ในสถานะ “staged” หมายถึง ไฟล์หรือ directory นั้นจะถูกนำเข้าสู่ระบบ Git และสามารถนำไป commit เพื่อการบันทึก version ในลำดับถัดไป

ขั้นตอนการสร้าง Repository และนำ Code ขึ้นระบบ Git

- ทำการสร้าง directory ชื่อ **C_Project** ใน directory C:\PROJECT จากนั้นเปิดโปรแกรม Dev-C++ เพื่อเขียนโค้ดภาษา C เพื่อทดสอบการใช้งาน Git โดยให้ตั้งชื่อไฟล์ **test_01.c** และ **test_02.c**

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT (main)
$ git add C_Project
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT (main)
$ git status
On branch main
```

```
No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   C_Project/test_01.c
    new file:   C_Project/test_02.c
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT (main)
$ git status
On branch main
```

```
No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   C_Project/test_01.c
    new file:   C_Project/test_02.c
```

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    C_Project/test_03.c
```

- ทดสอบทำการสร้างไฟล์ **test03.c** จากนั้นใช้คำสั่ง **git status** เพื่อตรวจสอบสถานะของไฟล์
- ทำการ add ไฟล์ test_03.c โดยใช้คำสั่ง **git add test_03.c**

ขั้นตอนการสร้าง Repository และนำ Code ขึ้นระบบ Git

การลบไฟล์ออกจาก Local Repository (git rm)

- หากไฟล์หรือ directory อยู่ในสถานะ Staged แล้วถูกลบหรือย้ายไปที่ directory อื่น จะทำให้ Git ไม่สามารถทราบถึงสถานะและ version ของไฟล์หรือ directory นั้นๆ ได้ เนื่องจาก Git ยังคงจำได้ว่าไฟล์หรือ directory นั้นๆ อยู่ใน directory เดิม จึงต้องใช้คำสั่งเพื่อแจ้งให้ Git ไม่จำเป็นต้องติดตามสถานะของไฟล์หรือ directory นั้นๆ อีกต่อไป

`git rm <ชื่อไฟล์ หรือ directory>`

- ทดสอบลบไฟล์ test_01.c ออกจาก directory C_Project และใช้คำสั่ง `git status` และใช้คำสั่ง `git rm test_01.c` เพื่อลบไฟล์ออกจากการติดตาม

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
```

```
No commits yet
```

```
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   test_01.c
    new file:   test_02.c
    new file:   test_03.c
```

```
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
    deleted:    test_01.c
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git rm test_01.c
rm 'C_Project/test_01.c'
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
```

```
No commits yet
```

```
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   test_02.c
    new file:   test_03.c
```


การแก้ไขไฟล์และ commit การเปลี่ยนแปลงเข้าสู่ Local Repository (modified, restore & git command)

- เมื่อไฟล์ source code อยู่ในสถานะ staged จะสามารถทำการเปลี่ยนแปลงแก้ไขและบันทึกได้ตามต้องการ ซึ่ง Git จะยังไม่ทำการเก็บ version แต่จะจัดไฟล์หรือ directory อยู่ในสถานะ modified จนกว่าจะรันคำสั่ง add อีกครั้ง เพื่อบอกให้ Git ปรับสถานะเข้าสู่สถานะ staged สำหรับการ commit เป็น version ใหม่ในลำดับถัดไป
- ทดสอบแก้ไขไฟล์ `test_02.c` และใช้คำสั่ง `git status`

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test_02.c
        new file:   test_03.c

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test_02.c
```

การแก้ไขไฟล์และ commit การเปลี่ยนแปลงเข้าสู่ Local Repository (modified, restore & git command)

- การใช้คำสั่ง commit จะทำให้ทุกไฟล์ใน directory ที่รันคำสั่ง git commit และอยู่ในสถานะ staged ถูกนำไปบันทึกเป็น version ใหม่ โดยคำสั่ง commit มีรูปแบบดังนี้

`git commit -m <คำอธิบายหรือหมายเหตุการบันทึก version>`

- หลังจากใช้คำสั่ง git commit และ git status จะไม่พบรายการไฟล์ใดๆ เนื่องจากไฟล์ที่เป็นสถานะ staged ถูก commit ไปแล้ว จึงไม่มีไฟล์หรือ directory ที่ Git ติดตามสถานะอีกต่อไป

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git add test_02.c
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
```

```
No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
       new file:   test_02.c
       new file:   test_03.c
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git commit -m "initial C language"
[main (root-commit) d8280fd] initial C language
2 files changed, 15 insertions(+)
create mode 100644 C_Project/test_02.c
create mode 100644 C_Project/test_03.c
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
nothing to commit, working tree clean
```

การแก้ไขไฟล์และ commit การเปลี่ยนแปลงเข้าสู่ Local Repository (modified, restore & git command)

- ถ้าไฟล์เคยอยู่ในสถานะ staged และมีการแก้ไขเนื้อหาไฟล์ จะทำให้ไฟล์อยู่ในสถานะ modified ซึ่งหากต้องการยกเลิกการแก้ไขเพื่อย้อนกลับไป version ที่เป็น staged ก่อนหน้านั้น โดยที่ไฟล์ยังไม่ได้ถูก add อีกครั้งเป็น staged เพื่อ commit จะสามารถทำได้โดยใช้คำสั่ง `git restore` เพื่อย้อนกลับไปยัง version ก่อนหน้า

`git restore <ชื่อไฟล์หรือ directory>`

```
test_01.c
1 #include <stdio.h>
2
3 main()
4 {
5     printf("Hello World\n");
6     printf("Hello pockship");
7     printf("Hello CPE101");
8 }
```

```
test_01.c
#include <stdio.h>
main()
{
    printf("Hello World\n");
    printf("Hello pockship");
    printf("Hello CPE101");
    printf("Ramza");
}
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test_01.c

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

POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git add test_01.c

POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   test_01.c

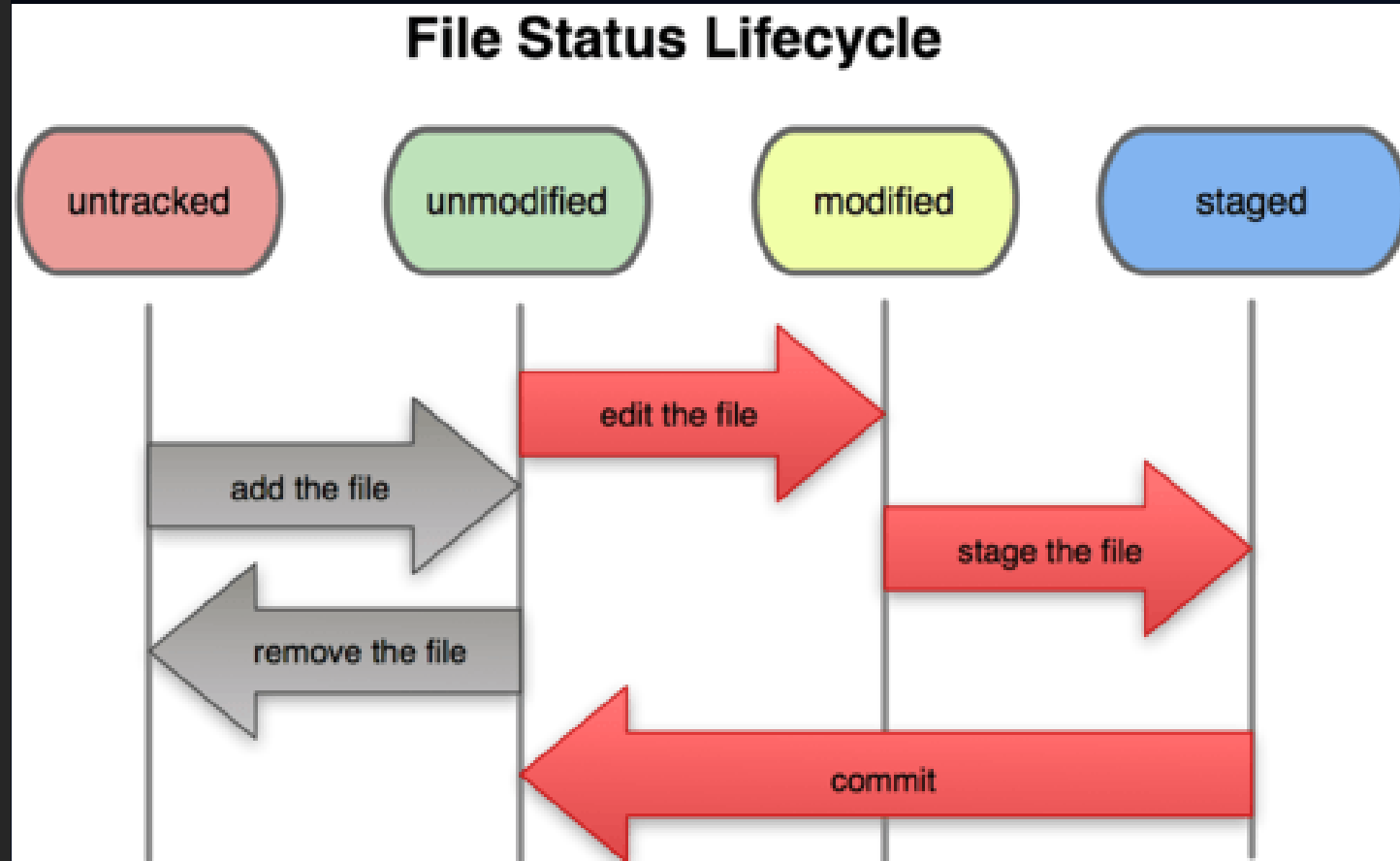
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   test_01.c

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   test_01.c

POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git restore test_01.c
```

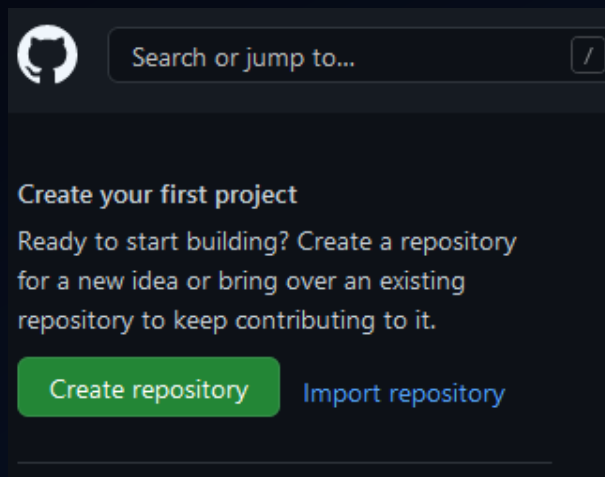
สถานะของไฟล์ในระบบ Git (File status life cycle)

- **Untracked** : ไฟล์ที่ยังไม่ถูกเพิ่มเข้าสู่การติดตามในระบบ Git แต่อยู่ใน directory เดียวกับ Local Repository
- **Unmodified** : สถานะของไฟล์แบบปกติที่ถูกเพิ่มเข้ามาสู่ระบบ Git โดยใช้คำสั่ง add ในครั้งแรกหรือไฟล์ที่ถูก commit ไปแล้วและยังไม่มีเปลี่ยนแปลงใดๆ หลังจากนั้น
- **Modified** : ไฟล์ที่มีการแก้ไขหรือเปลี่ยนแปลงจากเดิมหลังจากใช้คำสั่ง add มาแล้ว
- **Staged** : ไฟล์ที่มีการใช้คำสั่ง add หลังจากสร้างใหม่ หรือมีการแก้ไขและใช้คำสั่ง add อีกครั้งมาแล้ว จะทำให้ระบบ Git ติดตามการเปลี่ยนแปลงและสามารถ restore กลับมาที่ staged เดิมได้อีกครั้ง



การสร้าง Remote Repository

- ก่อนที่จะทำการ Push ชุด Source code จาก Local Repository จะต้องทำการสร้าง Remote Repository ก่อน ในที่นี่จะใช้ GitHub เป็น Remote Repository
- ทำการ login เข้า GitHub.com จากนั้นคลิกที่ปุ่ม **Create Repository**
- กำหนดชื่อ Remote Repository ในที่นี้กำหนดเป็น **Project_C**



The screenshot shows the GitHub homepage with the 'Create your first project' section. It includes a search bar, a description of creating a repository, and two buttons: 'Create repository' (highlighted in green) and 'Import repository'.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [redesigned-eureka?](#)

Description (optional)

☒ ☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ ☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license


A license tells others what they can and can't do with your code. [Learn more.](#)

You are creating a public repository in your personal account.

การสร้าง Remote Repository

- GitHub แสดงขั้นตอนในการสร้าง Local Repository หรือ Push ข้อมูลขึ้นสู่ Remote Repository

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

https://github.com/ramza1315/Project_C.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Project_C" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ramza1315/Project_C.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ramza1315/Project_C.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)


การ Push source code จาก Local Repository ขึ้นสู่ Remote Repository (git push)

- เมื่อนักพัฒนาแก้ไข source code และ commit ที่ Local Repository เรียบร้อยแล้ว สามารถนำขึ้น Remote repository ได้ โดยก่อนที่จะรันคำสั่ง push จะต้องใช้คำสั่ง git remote เพื่อให้ Git ที่เครื่องตนเองรู้จัก Remote Repository ที่สร้างไว้

```
git remote add origin <URL ของ Remote Repository>
```

- URL ของ Remote Repository จะถูกสร้างขึ้นอัตโนมัติ โดยระบบจะสร้างขึ้นมา สามารถคลิกที่ Repository ที่สร้างไว้เพื่อดูรายละเอียดได้

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

https://github.com/ramza1315/Project_C.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository inclu

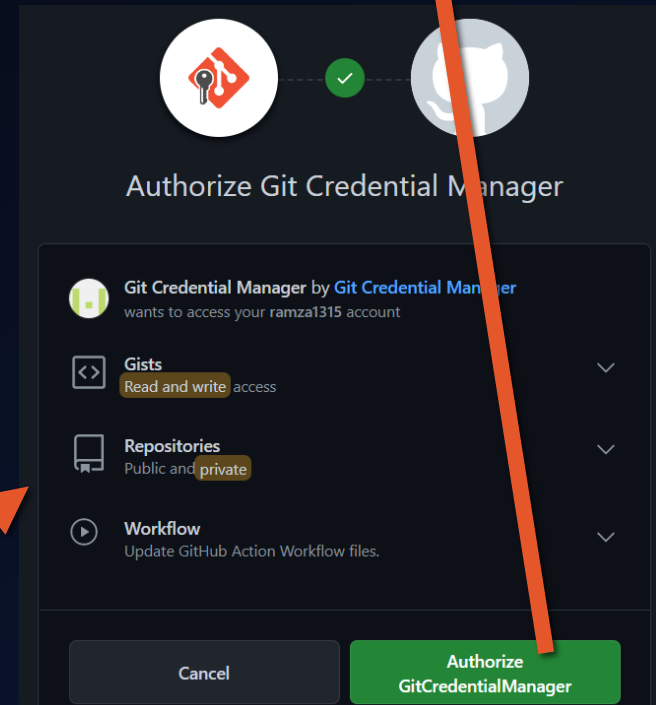
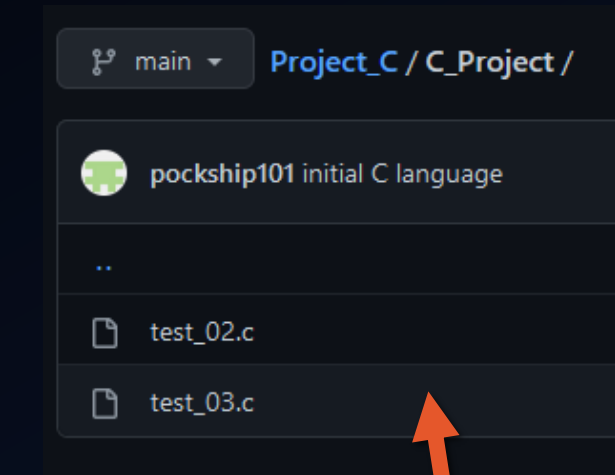
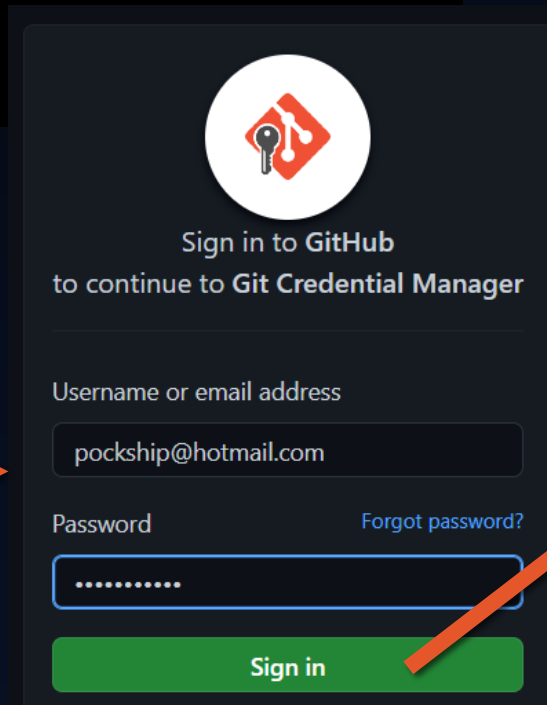
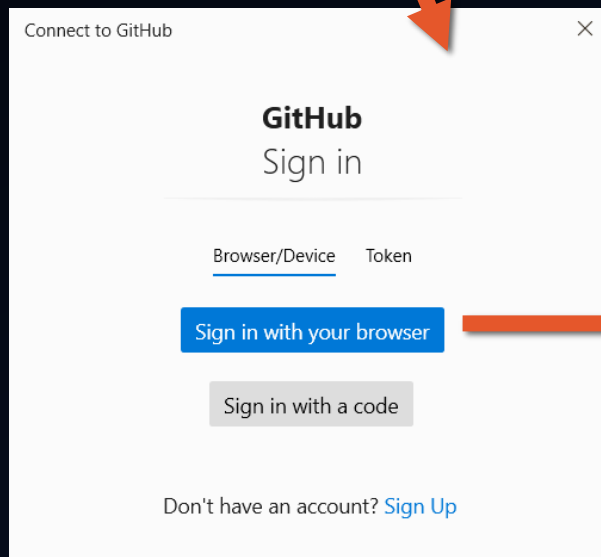
- จากนั้นจะใช้คำสั่ง git push เพื่อนำชุด source code ขึ้นไปที่ Remote Repository

```
git push -u origin <ชื่อ branch>
```

การ Push source code จาก Local Repository ขึ้นสู่ Remote Repository (git push)

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git remote add origin https://github.com/ramza1315/Project_C.git
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECT/C_Project (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 449 bytes | 449.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ramza1315/Project_C.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



การดึงข้อมูลจาก Remote มาสู่ Local Repository (git clone & git pull)

การดึงชุด source code จาก Remote Repository ที่มีอยู่ มาลงที่ Local Repository จะใช้ 2 คำสั่งสำหรับ 2 กรณี ดังนี้

- หากมี Remote Repository อยู่แล้ว และต้องการนำชุด source code ลงมาสร้างเป็น Local Repository ครั้งแรก บนเครื่องตนเอง จะใช้คำสั่ง clone

```
git clone <URL ของ Remote Repository>
```

- หากมี Local Repository อยู่แล้ว และต้องการนำชุด source code ลงมา update ให้ตรงกัน จะใช้คำสั่ง pull

```
git pull <URL ของ Remote Repository>
```

❶ Clone source code from Remote Repository


```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c
$ cd PROJECTS
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECTS
$ git init
Initialized empty Git repository in C:/PROJECTS/.git/
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECTS (main)
$ git clone https://github.com/ramza1315/Project_C.git
Cloning into 'Project_C'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECTS (main)
$ dir
Project_C
```

> Win10SSD (C:) > PROJECTS > Project_C > C_Project

Name	Date modified
 test_02.c	9/20/2022
 test_03.c	9/20/2022

การ Pull source code จาก Remote Repository

- ทดลอง upload ไฟล์ **test_01.c** ไปที่ Remote Repository ผ่านทาง **Website GitHub**

Project_C / C_Project

Drag additional files here to add them to your repository
Or choose your files

test_01.c

Commit changes

Add files via upload

Add an optional extended description...

☒ Commit directly to the `main` branch.
☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

main Project_C / C_Project /

ramza1315 Add files via upload

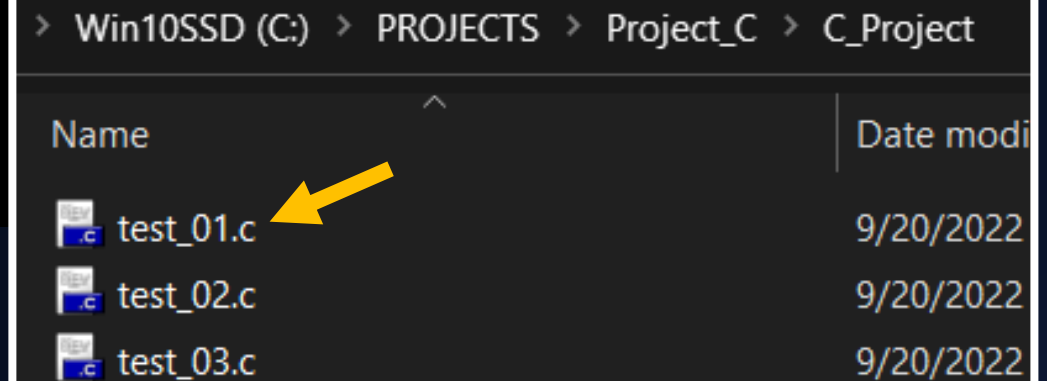

..




test_01.c	Add files via upload
test_02.c	initial C language
test_03.c	initial C language

How Pull source code from Remote Repository

```
POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECTS/Project_C/C_Project (main)
$ dir
test_02.c  test_03.c

POCKSHIP@DESKTOP-TOLSNBA MINGW64 /c/PROJECTS/Project_C/C_Project (main)
$ git pull https://github.com/ramza1315/Project_C.git
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 804 bytes | 57.00 KiB/s, done.
From https://github.com/ramza1315/Project_C
 * branch            HEAD              -> FETCH_HEAD
Updating d8280fd..29d5ccf
Fast-forward
 C_Project/test_01.c | 8 ++++++++
 1 file changed, 8 insertions(+)
 create mode 100644 C_Project/test_01.c
```

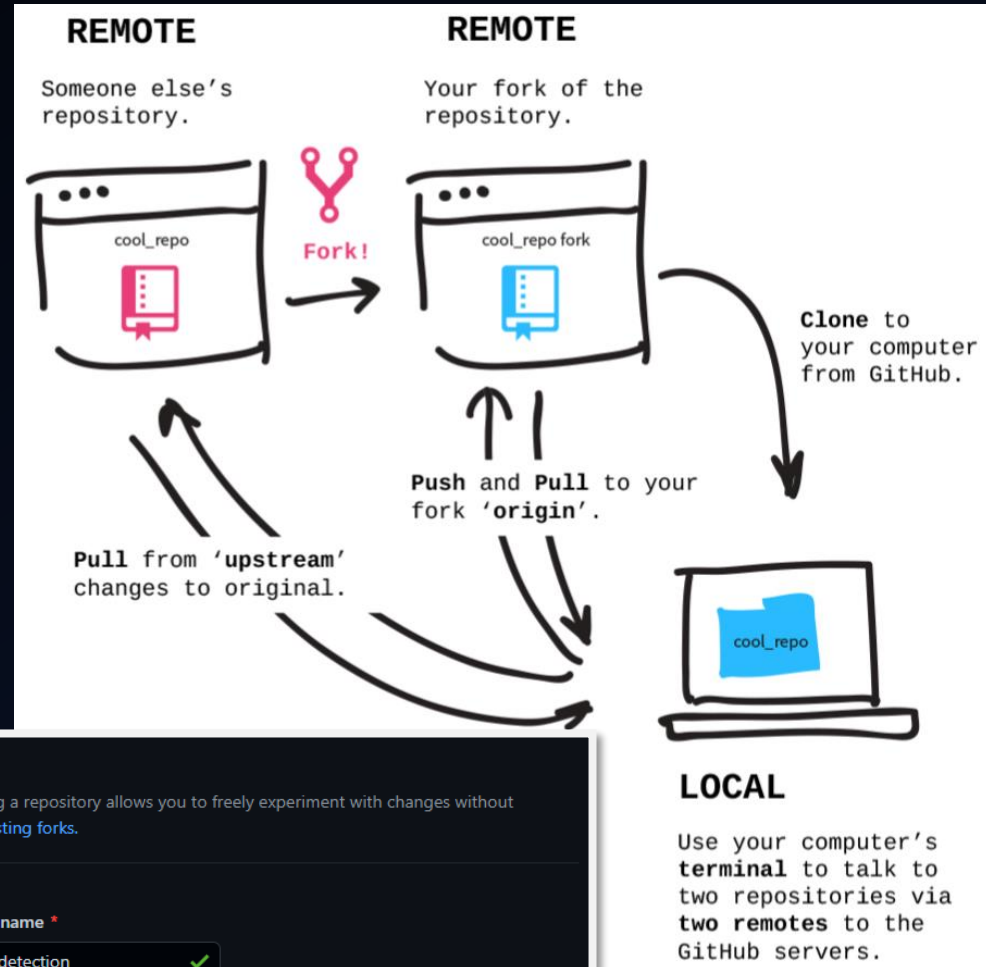


> Win10SSD (C:) > PROJECTS > Project_C > C_Project	
Name	Date modified
 test_01.c	9/20/2022
 test_02.c	9/20/2022
 test_03.c	9/20/2022

การทำ Git fork

- การ fork คือ การ (Clone Project) ที่เราต้องการ มาสร้าง Repository ใหม่ภายใต้ user ของเรา โดยสามารถนำมาพัฒนาต่อยอด เพิ่มฟีเจอร์ ตามที่เราต้องการได้

The screenshot shows the GitHub repository page for 'atulapra / Emotion-detection'. The repository is public and has 376 forks and 837 stars. The 'Fork' button is highlighted with a red arrow. The repository contains files like 'imgs', 'src', 'LICENSE', 'README.md', and 'requirements.txt'. The README.md file is selected, showing the title 'Emotion detection using deep learning'.



The screenshot shows the 'Create a new fork' form on GitHub. The form includes fields for 'Owner' (ramza1315), 'Repository name' (Emotion-detection), and 'Description' (Real-time Facial Emotion Detection using deep learning). The 'Copy the master branch only' checkbox is checked. The 'Create fork' button is visible at the bottom.

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner * **Repository name ***

ramza1315 / **Emotion-detection** ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

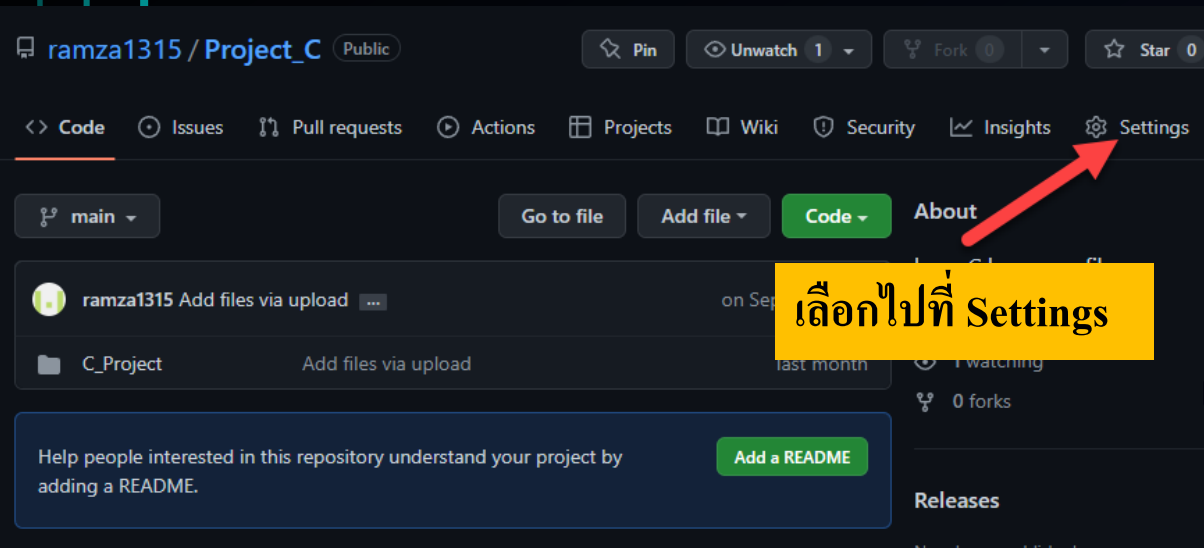
Real-time Facial Emotion Detection using deep learning

☒ **Copy the master branch only**
Contribute back to atulapra/Emotion-detection by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

Create fork

การลบ Repository ที่สร้างไว้ใน GitHub



Danger Zone

Change repository visibility

This repository is currently public.

Change visibility

Transfer ownership

Transfer this repository to another user or to an organization where you have the ability to create repositories.

Transfer

Archive this repository

Mark this repository as archived and read-only.

Archive this repository

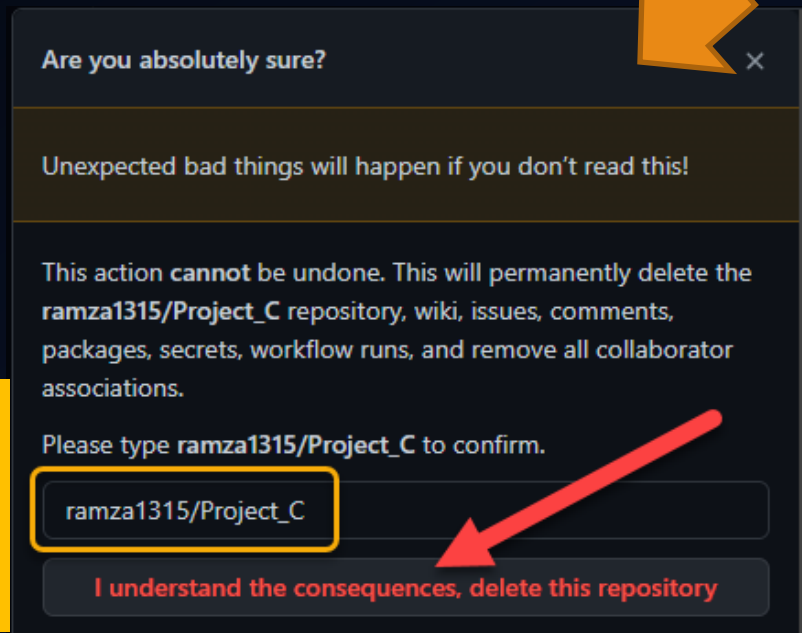
Delete this repository

Once you delete a repository, there is no going back. Please be certain.

Delete this repository

เลือกลบ repository

พิมพ์ link repository ของ
ตนเอง เพื่อยืนยันการลบ
จากนั้นกดปุ่มเพื่อลบ



Git Life Cycle

Staged : คือสถานะที่ไฟล์พร้อมจะ commit

unstaged : คือไฟล์ที่ถูกแก้ไขแต่ยังไม่ได้เตรียม commit

untracked : คือไฟล์ที่ยังไม่ถูก track โดย Git
(ส่วนมากจะเป็นไฟล์ที่เพิ่งสร้างใหม่)

deleted : ไฟล์ที่ถูกลบไปแล้ว

สรุปคำสั่ง

git init ไว้สร้าง Git repository

git status ตรวจสอบสถานะไฟล์ของ working directory และ staging area

git add เพิ่มไฟล์ the working directory เข้าสู่ staging area

git diff แสดงความแตกต่างของไฟล์ระหว่าง working directory กับ staging area

git commit เก็บประวัติการแก้ไขแบบถาวรจาก staging area ไว้ใน repository

git log แสดงรายการที่ commit มาทั้งหมด

สรุปคำสั่ง

git branch ใช้แสดงรายชื่อ branch ทั้งหมด

git branch <branch_name> สร้าง branch ใหม่

git checkout <branch_name> สลับไปใช้งาน branch ที่ระบุ

git merge <branch_name> ใช้รวมไฟล์จาก branch ที่ระบุ มายัง branch ปัจจุบัน

git branch -d <branch_name> ลบ branch ที่ระบุ

สรุปคำสั่ง

git clone สร้าง local repository จาก remote repository

git remote -v แสดงรายการ remote address ทั้งหมด

git fetch ดึงข้อมูลทั้งหมดจากฝั่ง remote มายัง local

git merge origin/master สัมรวมไฟล์จาก origin/master มายัง local branch

git pull สัมรวมไฟล์จาก ฝั่ง remote มายัง local (git fetch + git merge)

git push origin <branch_name> ส่งข้อมูลจาก local branch ไปรวมกับ originremote.

stackoverflow เพื่อนักเขียนโปรแกรม

- การเลือกบทความจาก Stackoverflow.com ซึ่งเป็นแหล่งรวมคำตอบทุกคำถามของโปรแกรมเมอร์ทั่วโลก จะมีจุดสังเกต 4 จุด ดังนี้
- 1. สังเกตคะแนนโหวตของคำถาม ซึ่งยิ่งมากยิ่งดี ถ้าเป็น 0 กด Back กลับไปหาเข้าหน้าบทความอื่นได้เลย
- 2. จำนวนคำตอบ Answers อันนี้ช่วยให้มีตัวเลือกที่หลากหลายมากขึ้น
- 3. จำนวนคะแนนโหวตของแต่ละคำตอบ ช่วยให้ Code ที่ดีที่สุด ณ ตอนนั้น
- 4. ส่วนสุดท้าย เครื่องหมายถูกสีเขียว หมายถึงเจ้าของคำถามได้รับคำตอบที่พึงพอใจ

stackoverflow Questions Developer Jobs Tags Users Search...

Redirect html5 video after play

I have an html 5 video which i remove the control buttons and added a js code in order for the user to play the video when the video is clicked. What I need to do is to bind an additional script that will redirect the page after the video plays without a page reload. Below is my js code.

```
function play(){
  var video = document.getElementById('video');
  video.addEventListener('click',function(){
    video.play();
  },false);
}
```

And Here is my HTML5 Video Code

```
<video id="video" width="770" height="882" onclick="play();"
<source src="video/Motion.mp4" type="video/mp4" />
</video>
```

javascript html5 html5-video

share improve this question asked May 2 '12 at 20:48

stackoverflow Questions Developer Jobs Tags Users Search...

Without a page reload? – AlienWebguy May 2 '12 at 20:51

add a comment

6 Answers active oldest votes

```
<script src="text/javascript">
// Do not name the function "play()"
function playVideo(){
  var video = document.getElementById('video');
  video.play();
  video.addEventListener('ended',function(){
    window.location = 'http://www.google.com';
  });
}
</script>
<video controls id="video" width="770" height="882" onclick="playVideo()"
<source src="video/Motion.mp4" type="video/mp4" />
</video>
```

Here is a list of media events to which you can bind:
<http://dev.w3.org/html5/spec/Overview.html#mediaevents>

share improve this answer edited May 2 '12 at 21:24 answered May 2 '12 at 20:53