



# Git

ผศ.ดร.สายชล ใจเย็น



What is Version Control?



What is Git?



Git Workflow

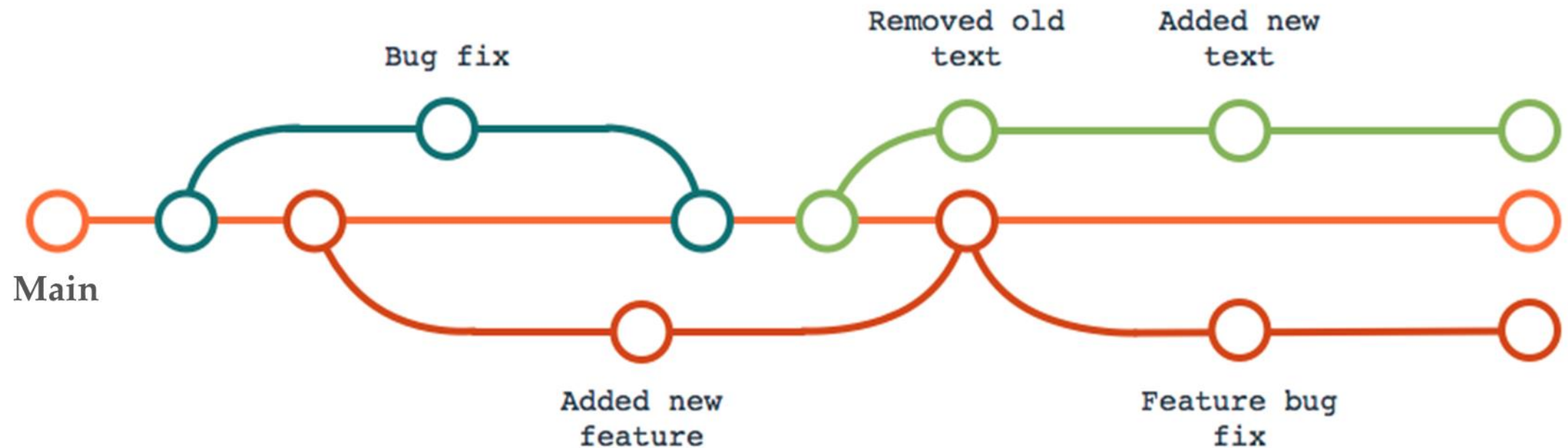


Git Commands

# What is Version Control?

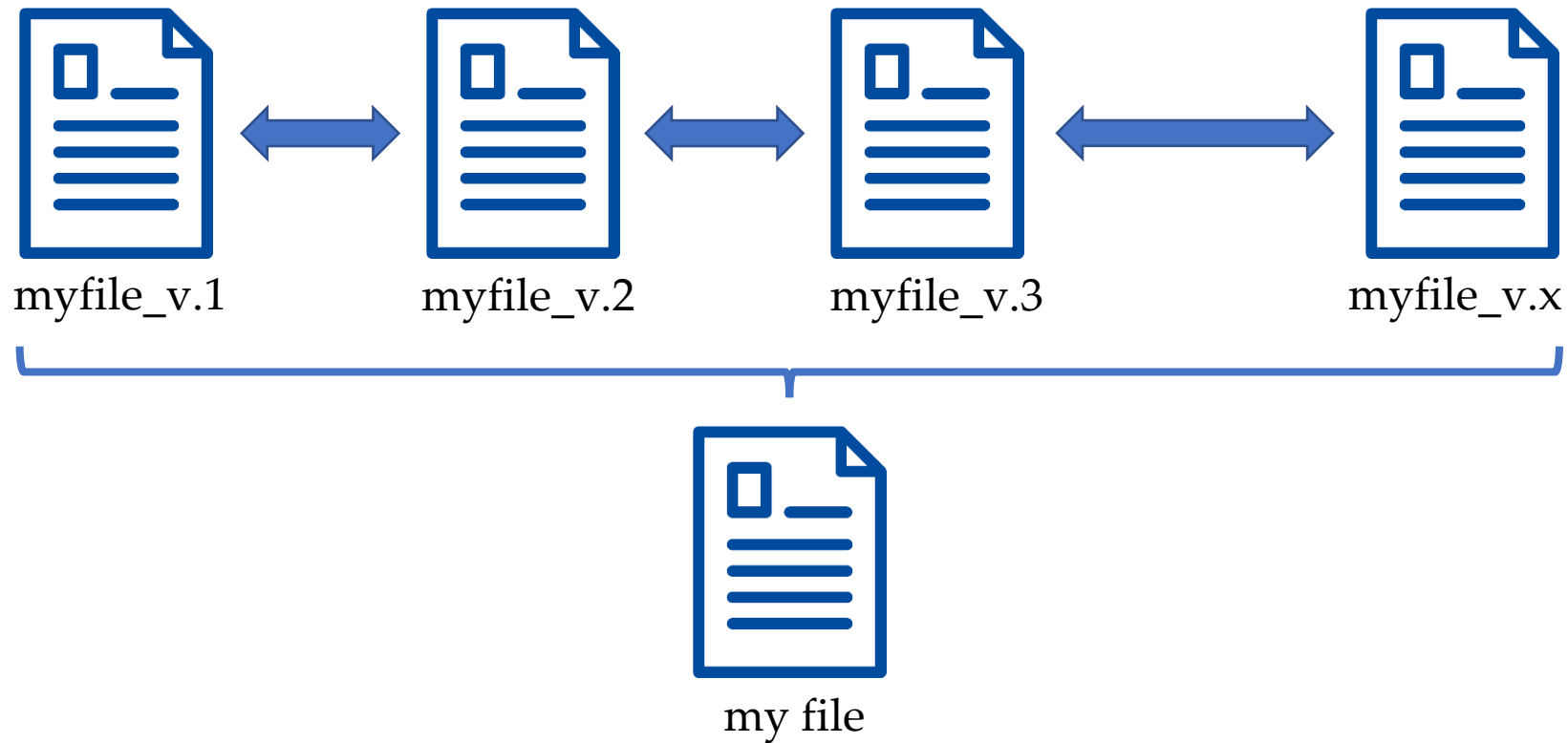
---

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



# Why is Version Control Important?

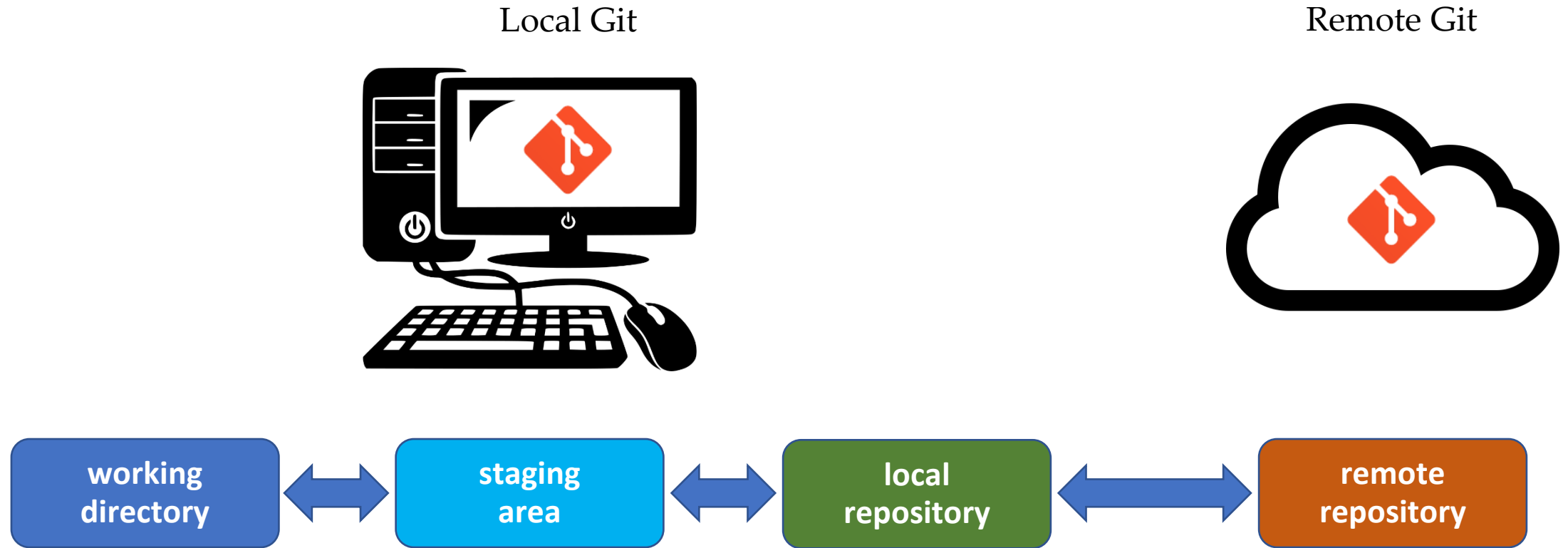
---



# What is Git?

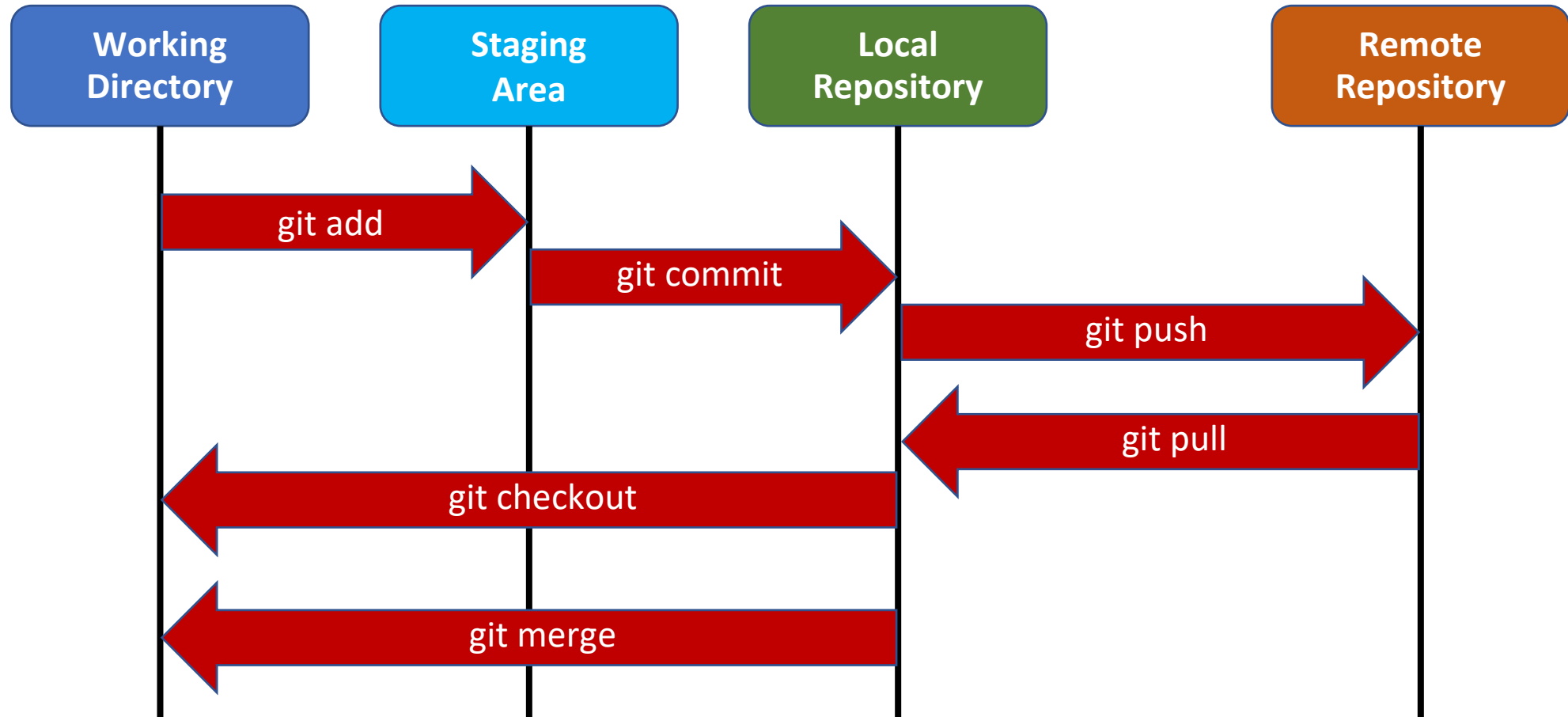
---

- Git is a distributed version-control system for tracking changes in source code during software development.



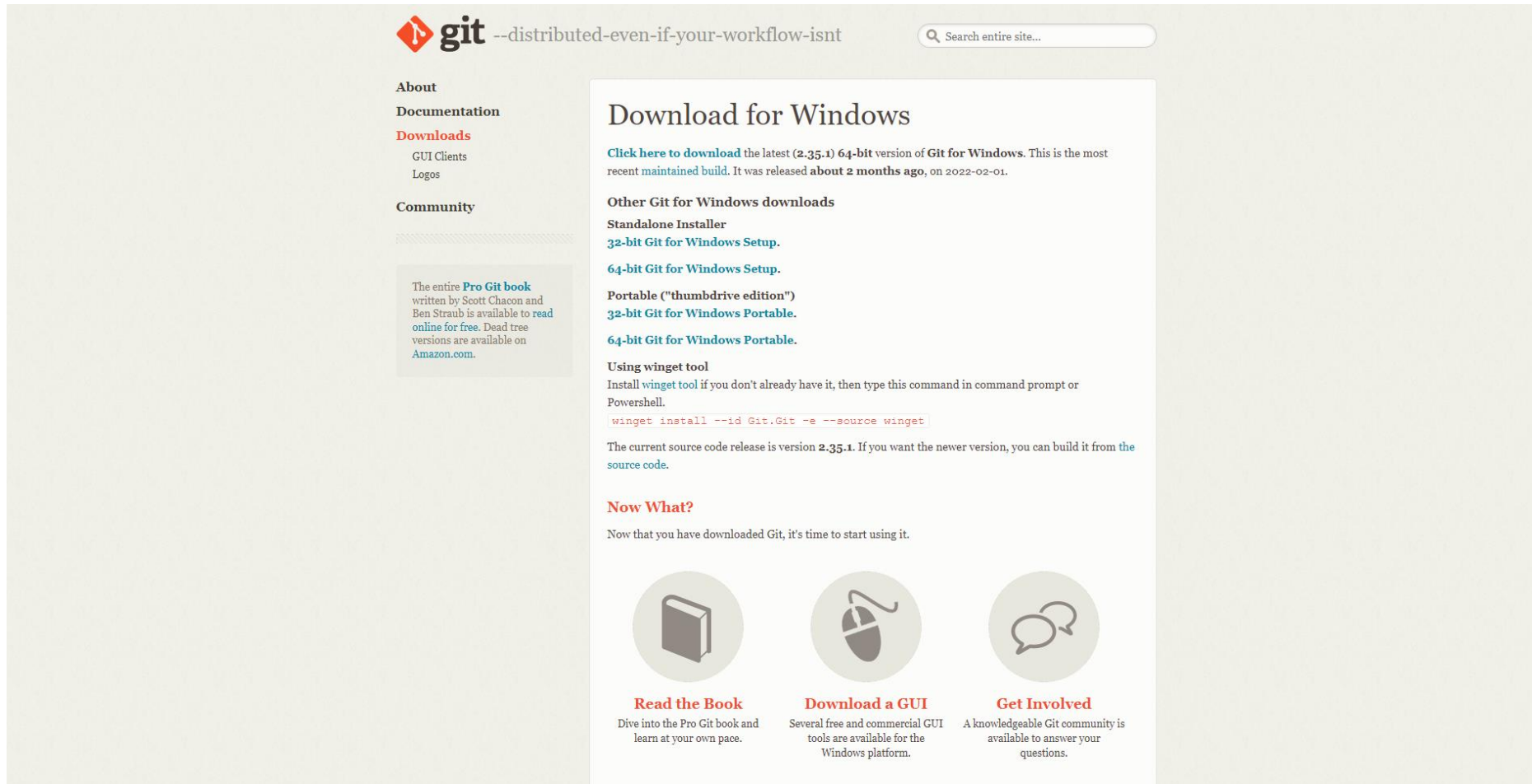
# Git Workflow

---



# Installing Git on Windows

- Go to the website: <https://git-scm.com/download/win>



The screenshot shows the Git website's 'Download for Windows' page. The header features the Git logo and the tagline '--distributed-even-if-your-workflow-isnt', along with a search bar. A left sidebar contains navigation links: 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. Below these is a box for the 'Pro Git book'. The main content area is titled 'Download for Windows' and includes a link to download the latest 64-bit version of Git for Windows, noting it was released about 2 months ago. It lists other download options: 'Standalone Installer' (32-bit and 64-bit setups) and 'Portable ("thumbdrive edition")' (32-bit and 64-bit). A section titled 'Using winget tool' provides instructions and a command to install the tool. Below this, it states the current source code release is version 2.35.1. A 'Now What?' section encourages users to start using Git. At the bottom, three circular icons represent 'Read the Book', 'Download a GUI', and 'Get Involved', each with a brief description.

**git** --distributed-even-if-your-workflow-isnt

Search entire site...

**About**  
**Documentation**  
**Downloads**  
GUI Clients  
Logos  
**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

## Download for Windows

[Click here to download](#) the latest (2.35.1) 64-bit version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 2 months ago**, on 2022-02-01.

### Other Git for Windows downloads

**Standalone Installer**  
[32-bit Git for Windows Setup](#).  
[64-bit Git for Windows Setup](#).

**Portable ("thumbdrive edition")**  
[32-bit Git for Windows Portable](#).  
[64-bit Git for Windows Portable](#).

### Using winget tool


Install **winget tool** if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```


The current source code release is version **2.35.1**. If you want the newer version, you can build it from [the source code](#).

### Now What?


Now that you have downloaded Git, it's time to start using it.



**Read the Book**  
Dive into the Pro Git book and learn at your own pace.



**Download a GUI**  
Several free and commercial GUI tools are available for the Windows platform.



**Get Involved**  
A knowledgeable Git community is available to answer your questions.

# Initial Setup

---

- Once you have installed Git, the first thing you need to do is to tell Git your **name** and **email** (particularly before creating any commits).

```
C:\> git config --global user.name "Your Name"  
C:\> git config --global user.email "email@gmail.com"
```

- If you want to check your configuration settings, you can use the git config --list command to list all the settings.

```
C:\> git config --list
```



# LAB 1

---

1. ติดตั้ง Git บนเครื่องคอมพิวเตอร์
2. เปิด Command Prompt แล้วตรวจสอบการติดตั้งด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\> git --version
```

3. กำหนดค่าเริ่มต้น name และ email ให้กับ Git ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\> git config --global user.name "Your Name"
```

```
C:\> git config --global user.email "email@gmail.com"
```

# Getting a Git Repository

---

- A **Git repository** is the local collection of all the files related to a particular Git version control system and contains a `.git` subdirectory in its root.
- You can obtain a Git repository in one of two ways:
  1. You can take a local directory that is currently not under version control, and turn it into a Git repository, or
  2. You can clone an existing Git repository from elsewhere.

# Initializing a Repository in an Existing Directory

---

- To initialize a git repository, use this command while inside the project folder. This will create a .git folder.

```
C:\myproject> git init
```

# Cloning an Existing Repository

---

- You can clone a repository with `git clone <url>`. For example, if you want to clone the repository named “myproject”, you can use this command:

```
C:\> git clone https://github.com/username/myproject.git
```

# git status

---

- This command will list files in green or red colors. Green files have been added to the stage but not committed yet. Files marked as red are the ones not yet added to the stage.

```
C:\myproject> git status
```

# LAB 2

---

1. เปิดโปรแกรม NetBeans แล้วสร้าง Project ใหม่ชื่อ MyProject
2. เปิด Command Prompt แล้วเข้าไปที่ไดเรกทอรีที่เก็บไฟล์โปรแกรม .java
3. พิมพ์คำสั่งสร้าง git repository ดังนี้

```
C:\myproject> git init
```

4. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

# git add

---

- This command adds one or all changed files to the staging area.

```
C:\myproject> git add file_name
```

```
C:\myproject> git add *.java
```

```
C:\myproject> git add .
```

# LAB 3

---

1. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```

2. เพิ่มไฟล์ไปยัง Staging Area ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git add MyProject.java
```

3. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

5. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```



# git commit

---

- This command records the file in the version history. The -m means that a commit message follows. This message is a custom one and you should use it to let your colleagues, or your future self know what was added in that commit.

```
C:\myproject> git commit -m "commit message"
```

# LAB 4

---

1. บันทึกการเปลี่ยนแปลงของไฟล์ลงใน Repository ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git commit -m "new file"
```

2. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

3. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```

# LAB 5

---

1. แก้ไขไฟล์ MyProject.java ด้วยการเพิ่มคำสั่ง ดังนี้

```
System.out.println("Hello Worlds!");
```

2. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

3. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```

# LAB 6

---

1. เพิ่มไฟล์ไปยัง Staging Area ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git add MyProject.java
```

2. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

3. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```

# LAB 7

---

1. บันทึกการเปลี่ยนแปลงของไฟล์ลงใน Repository ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git commit -m "modified file"
```

2. ตรวจสอบสถานะ (status) ของไฟล์ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git status
```

3. ตรวจสอบว่ามีไฟล์อยู่ใน Staging Area หรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git diff --name-only --cached
```

# Git Server

---

- Github

- <https://github.com/>

- Gitlab

- <https://about.gitlab.com/>

- BitBucket

- <https://bitbucket.org/>

# Connecting a Remote Repository

---

- git remote command is used to connect your local repository to the remote server.

```
C:\myproject> git remote add origin https://gitlab.com/username/myproject.git
```

# LAB 8

---

1. ไปที่ <https://github.com/>
2. Sign in เข้าสู่ระบบ (ถ้ายังไม่มี account ของ github ให้ Sign up ก่อน)
3. สร้าง repository ใหม่ชื่อ myproject
4. เชื่อมโยง Local Repository กับ Remote Repository ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git remote add origin https://gitlab.com/username/myproject.git
```



# git push

---

- This command sends the committed changes to a remote repository.

```
C:\myproject> git push origin main
```

# git fetch

---

- Sometimes you may wish to download the new commits from the remote repository without merging them into your current branch (or without merging them yet). To do this, you can use the git fetch command.

```
C:\myproject> git fetch
```

# LAB 9

---

1. ส่งการเปลี่ยนแปลงของไฟล์ใน Local Repository ไปยัง Remote Repository ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git push origin main
```

2. ตรวจสอบการเปลี่ยนแปลงของไฟล์บน Local Repository กับ Remote Repository ว่าตรงกันหรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git fetch
```

# git pull

---

- To pull the changes from the remote server to your local computer.

```
C:\myproject> git pull origin main
```

# LAB 10

---

1. ไปที่ <https://github.com/>
2. เข้าไปที่ repository ชื่อ myproject แล้วเพิ่มไฟล์ใหม่ชื่อ test.txt
3. ตรวจสอบการเปลี่ยนแปลงของไฟล์บน Local Repository กับ Remote Repository ว่าตรงกันหรือไม่ ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git fetch
```

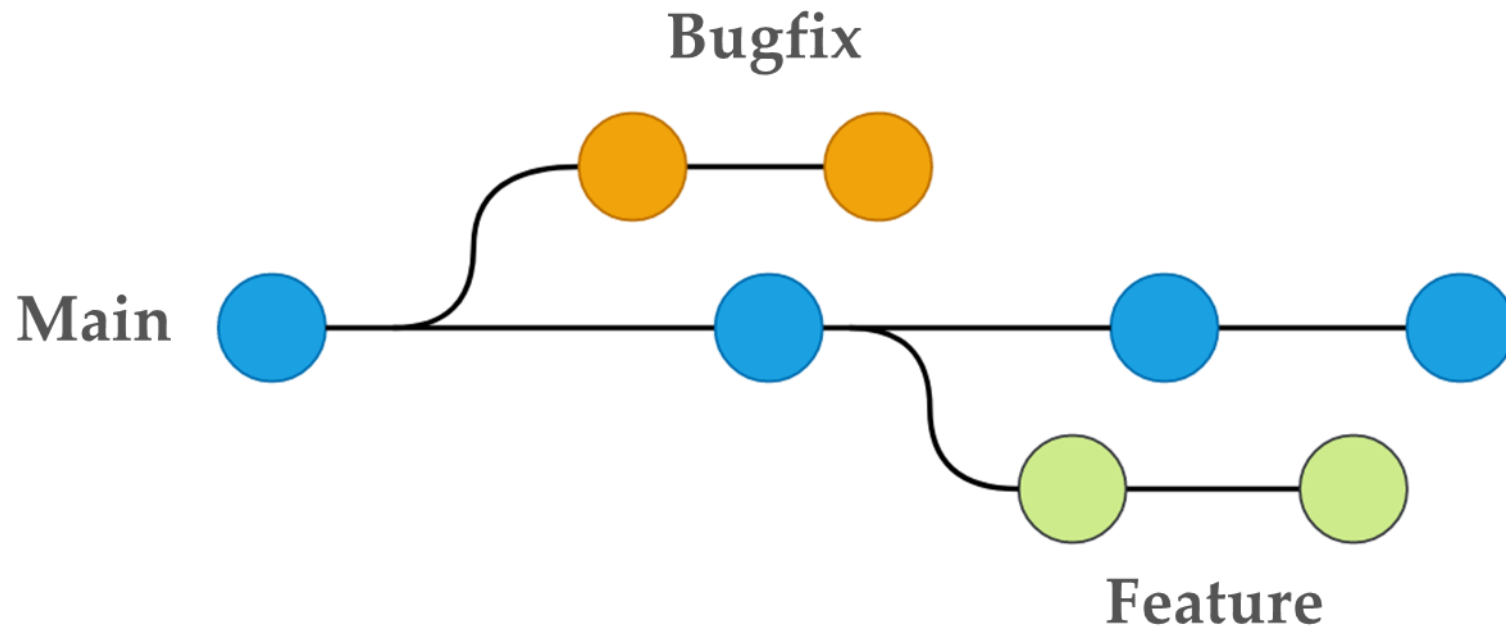
4. ดึงการเปลี่ยนแปลงของไฟล์จาก Remote Repository มา merge เข้ากับ Local Repository ด้วยการพิมพ์คำสั่ง ดังนี้

```
C:\myproject> git pull origin main
```

# Branching

---

- Branches are used to develop features isolated from each other. The master branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



# git branch

---

- To list all the branches and see on what branch you currently are:

```
C:\myproject> git branch
```

# Creating a New Branch

---

- The git branch command can be used to create a new branch. When you want to start a new feature, you create a new branch off master using the following command:

```
C:\myproject> git branch new_branch
```



# Switching Branches

---

- To switch from one branch to another:

```
C:\myproject> git checkout branch_name
```

# Merging Branches

---

- Suppose you've decided that your new\_branch work is complete and ready to be merged into your master branch. In order to do that, you'll merge your new\_branch into master using the following commands:

```
C:\myproject> git checkout main
```

```
C:\myproject> git merge new_branch
```

# Deleting Branches

---

- We can delete a branch by calling the branch command and passing in the -d option, followed by the branch name.

```
C:\myproject> git branch -d branch_name
```

# git log

---

- The git log command displays all the commits in a repository's history.

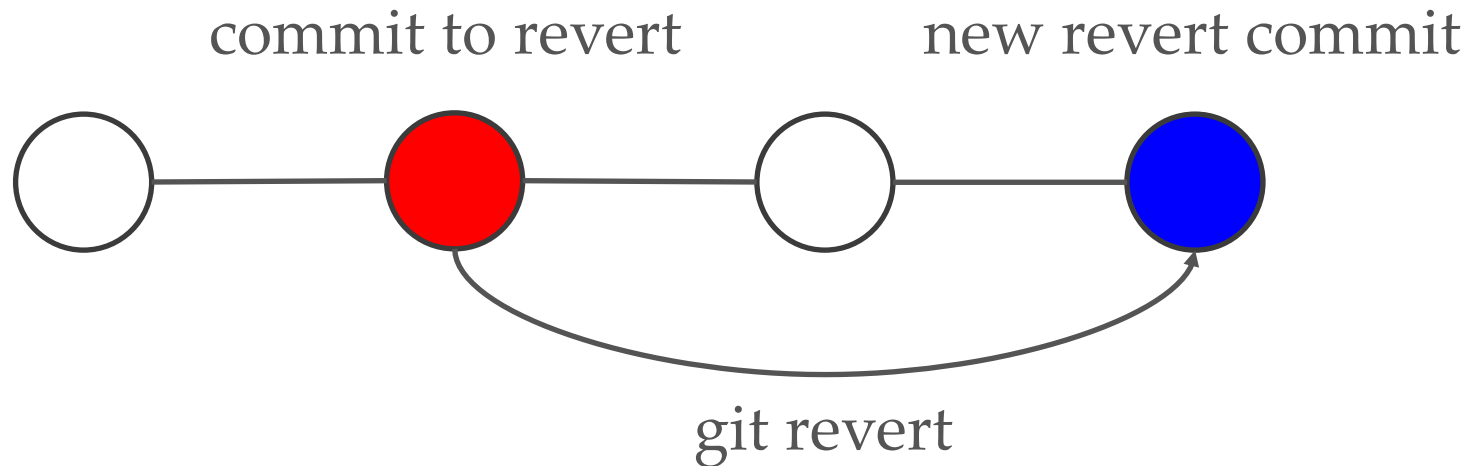
```
C:\myproject> git log
```

# git revert

---

- The git revert command is used for undoing changes to a repository's commit history.

```
C:\myproject> git revert <commit-hash>
```



**THANK YOU**