



Chiangmai, Thailand in 2010
by Usa Sammapun

Version Control เป็องตัน

Usa Sammapun

Introduction to Version Control System (VCS)

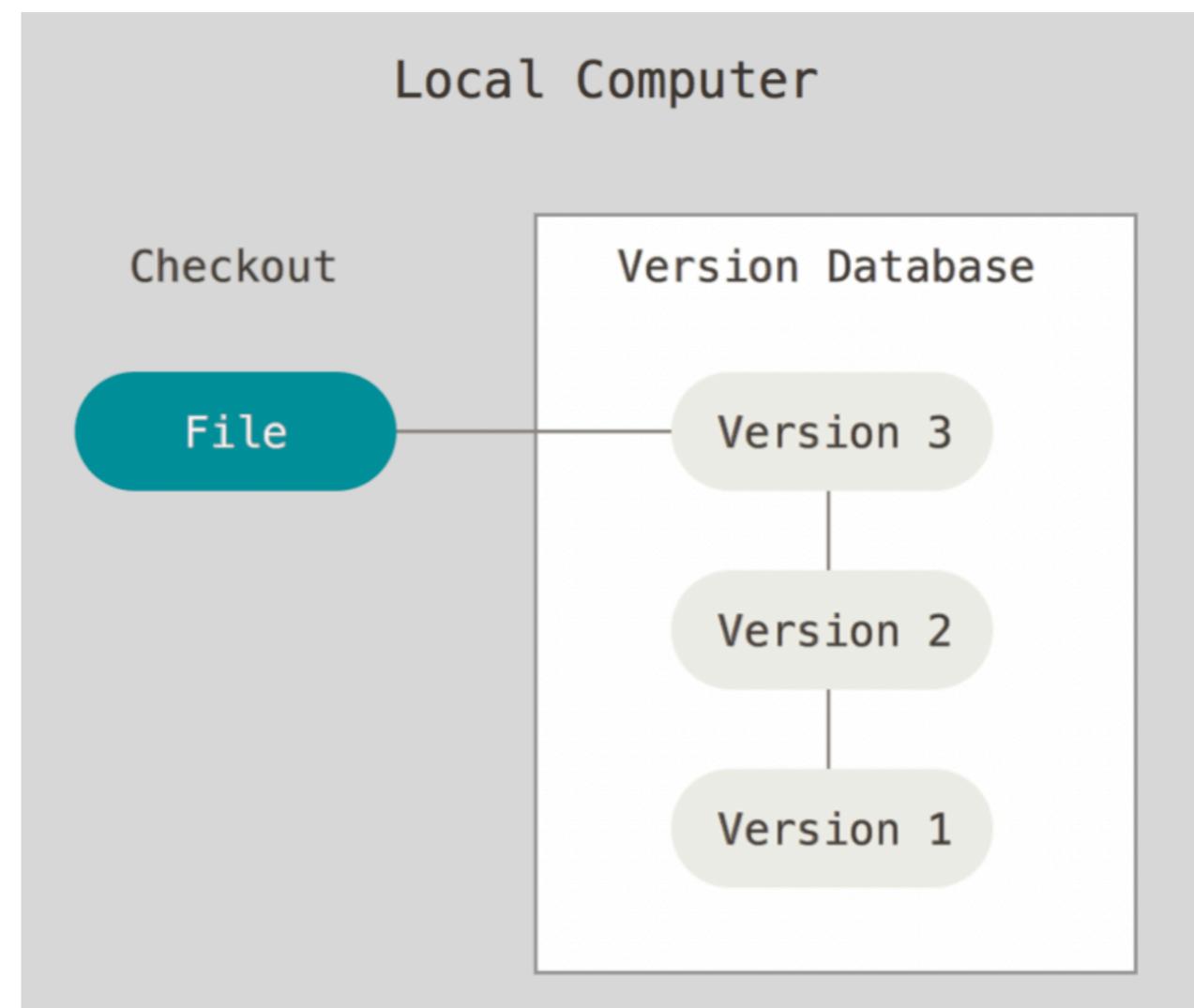
- **Versioning** and tracking changes
 - ติดตามการเปลี่ยนแปลงของไฟล์
- **Backup** and rolling back
 - นำไฟล์เวอร์ชันก่อนกลับมาใช้ได้
- **Collaboration** with multiple members and computers
 - ทำให้แบ่งงานและรวมงานกับผู้อื่นได้ง่ายขึ้น
- **Logging**
 - เก็บประวัติการเปลี่ยนแปลง

Introduction to Version Control System (VCS)

- นี่ 3 ประเภทหลัก
 - Local VCS
 - Centralized VCS
 - Distributed VCS

Introduction to Version Control System (VCS)

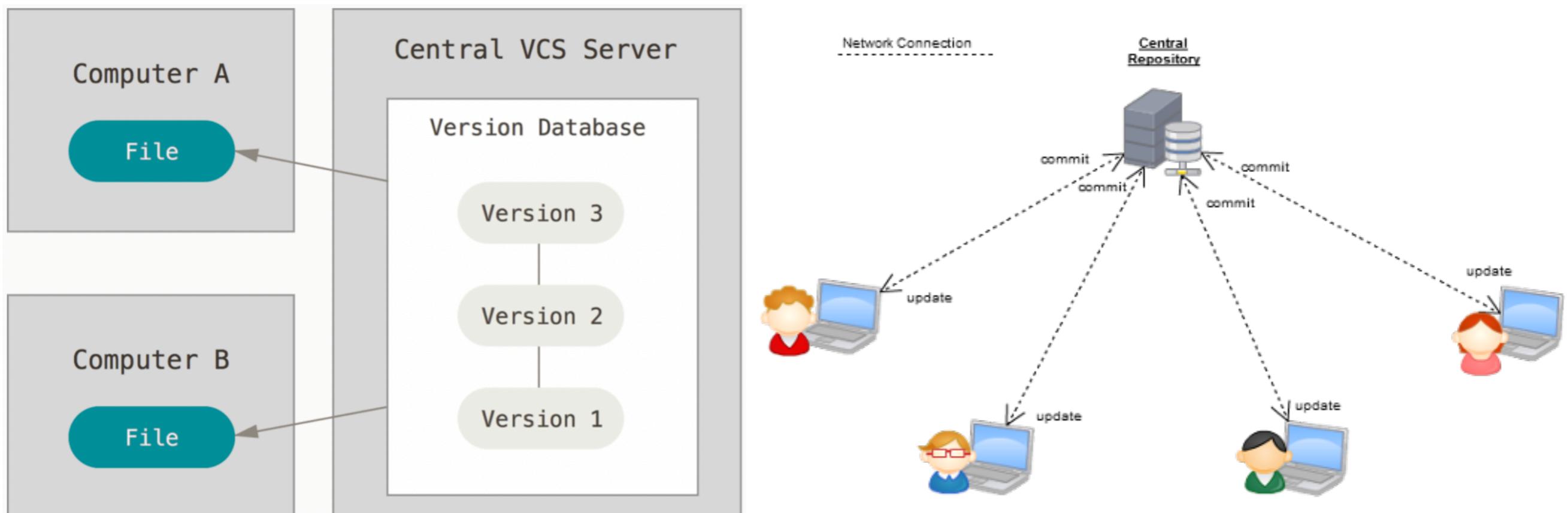
- Local VCS



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

Introduction to Version Control System (VCS)

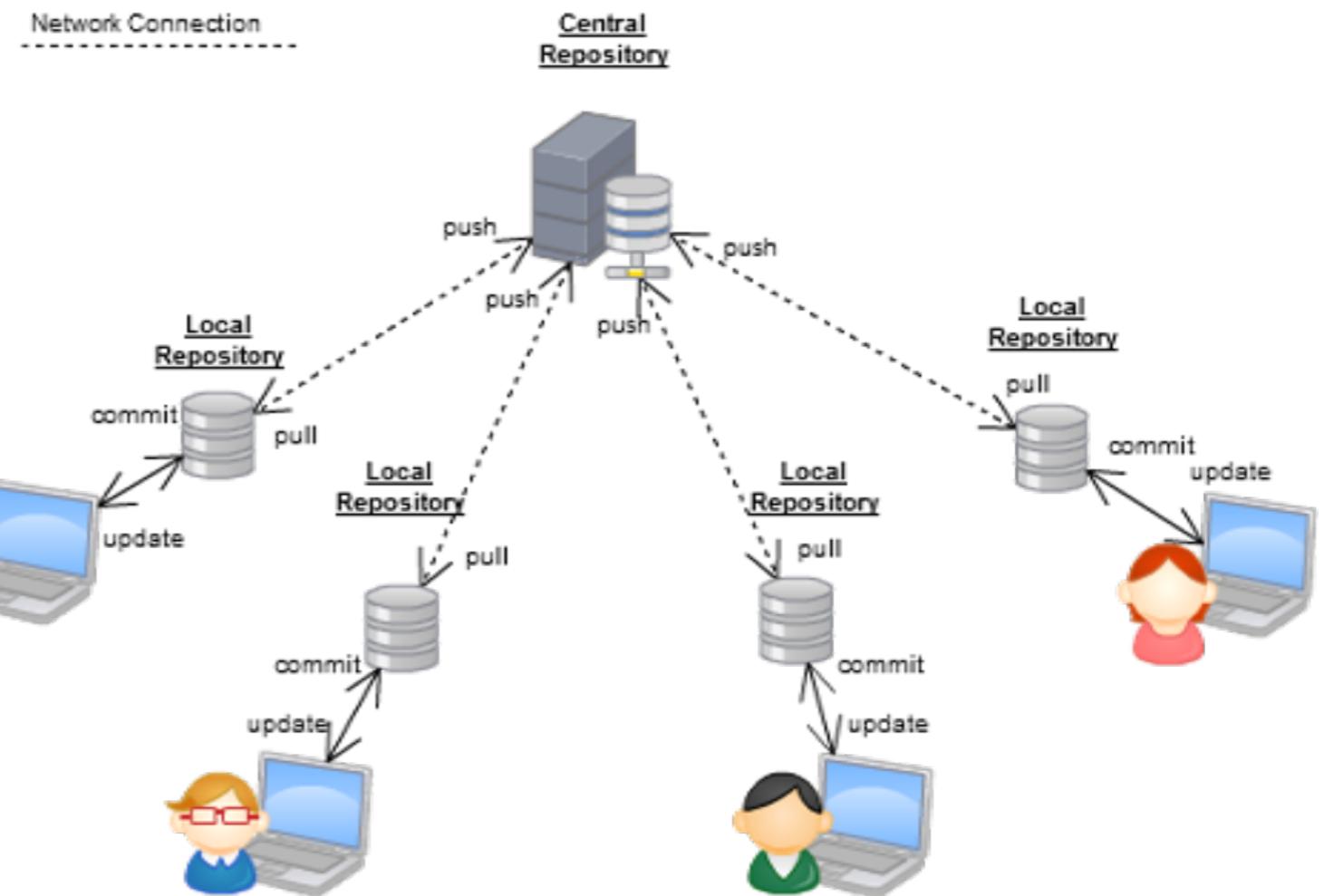
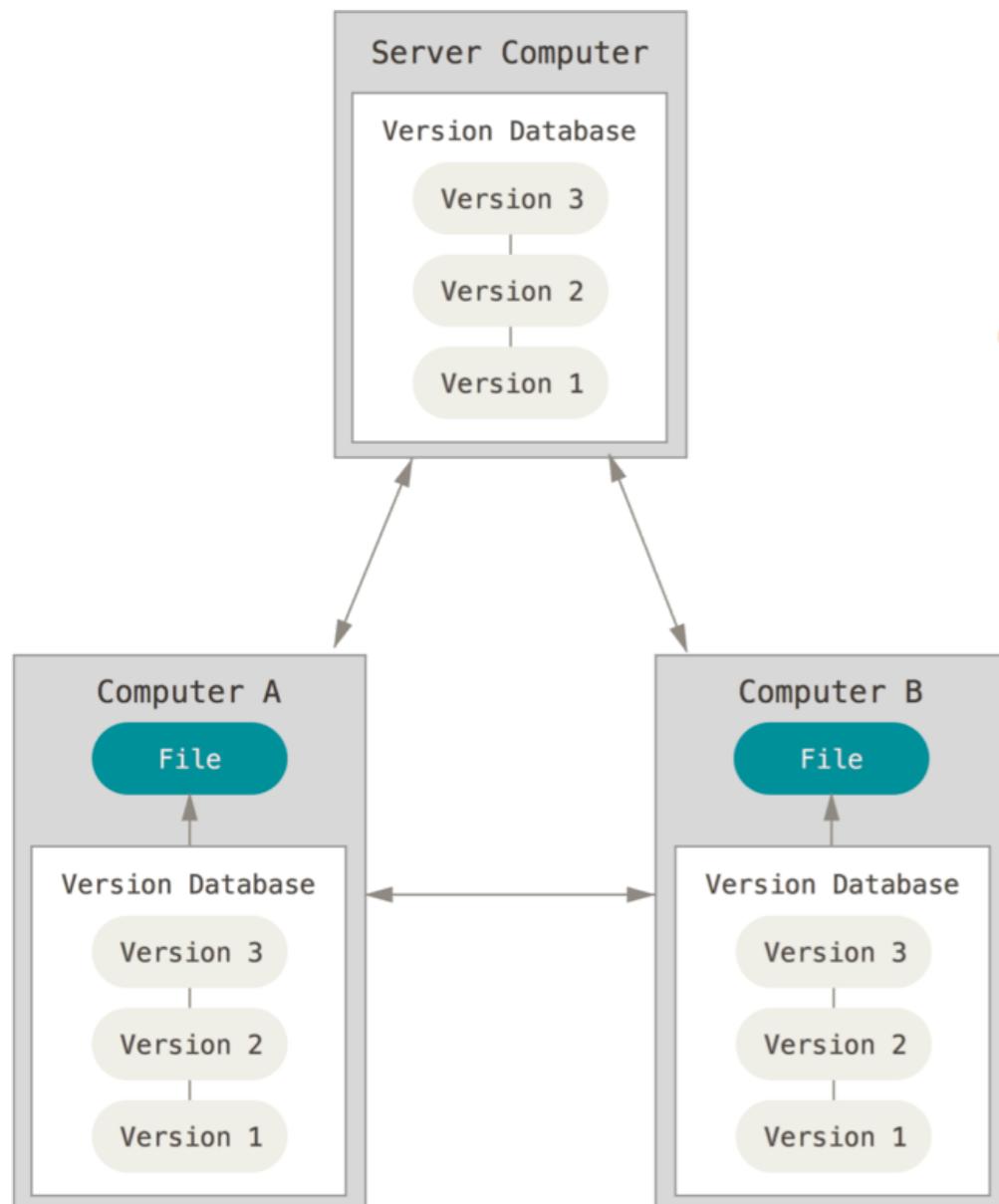
- Centralized VCS
 - เช่น CVS, Subversion (SVN)



Introduction to Version Control System (VCS)

- Distributed VCS

- ឧីវិញ Git, Mercurial



Git — Global Information Tracker

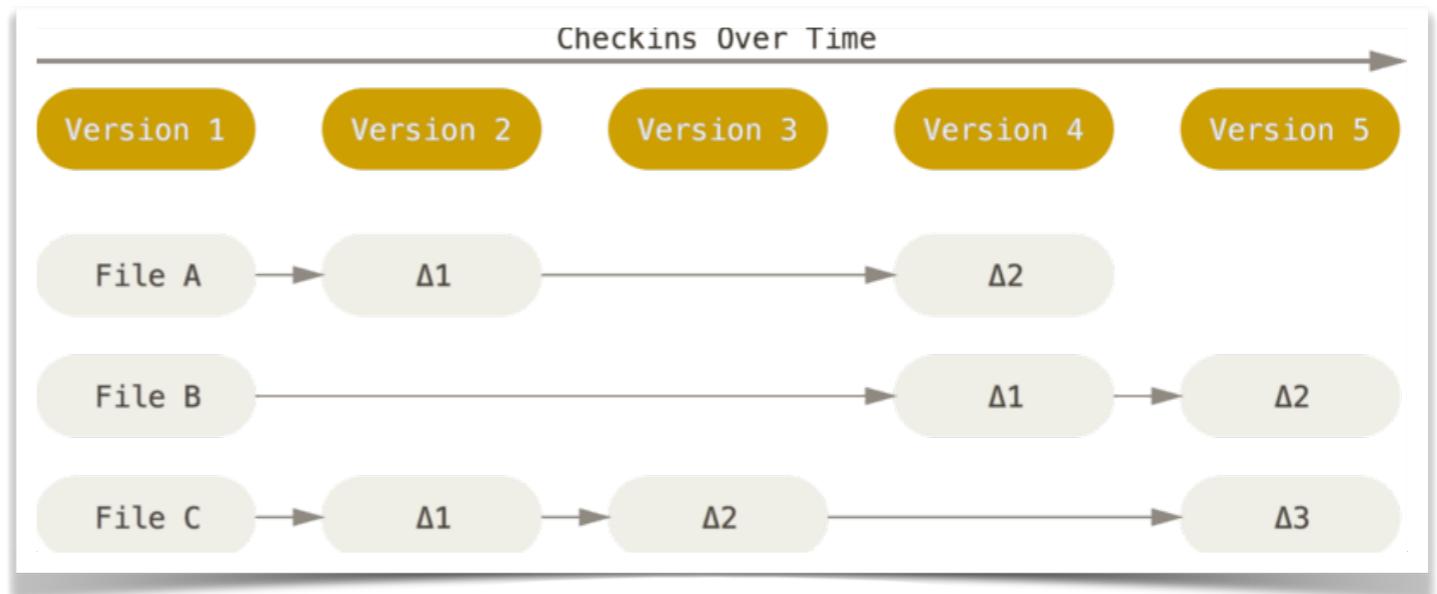
- Version control system แบบ **distributed**
 - พัฒนาโดย Linus Torvalds (ผู้พัฒนา Linux) ในปี 2005
- Goal
 - Speed
 - Collaboration with multiple members and computers
 - Distributed: Local vs central
 - Non-linear development: **Branching**
 - Can handle large project efficiently



การติดตาม change ใน VCS

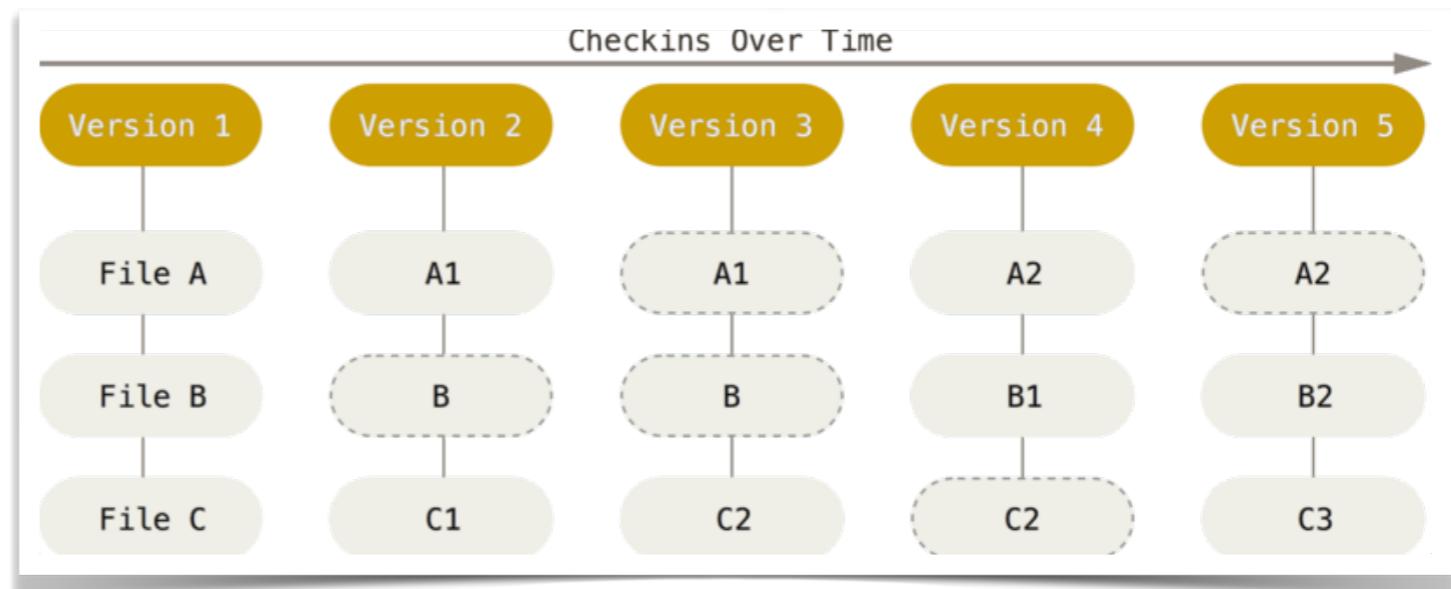
- **Delta** (เก็บการเปลี่ยนแปลงของไฟล์)

- SVN, etc.



- **Snapshot** (เก็บ “snapshot”)

- Git



Repository

- เก็บ metadata ที่เกี่ยวข้องกับไฟล์ที่ต้องการ track
- ใน git
 - โฟลเดอร์ .git

ลักษณะสำคัญของ git

- **Nearly Every Operation Is Local**
 - การใช้งานส่วนใหญ่ เป็นแบบ local บนเครื่องผู้ใช้
- **Git Has Integrity**
 - Git เก็บข้อมูลเกี่ยวกับไฟล์ในรูปแบบ hash ของ file content ไม่ได้เก็บเป็น filename
 - มีการ checksum ทำให้แน่ใจว่าไฟล์มีการเปลี่ยนแปลงหรือไม่

ไฟล์ใน git มีได้ 3 สถานะ

- **Modified**

- ไฟล์มีการแก้ไข แต่ยังไม่ได้ commit

- **Staged**

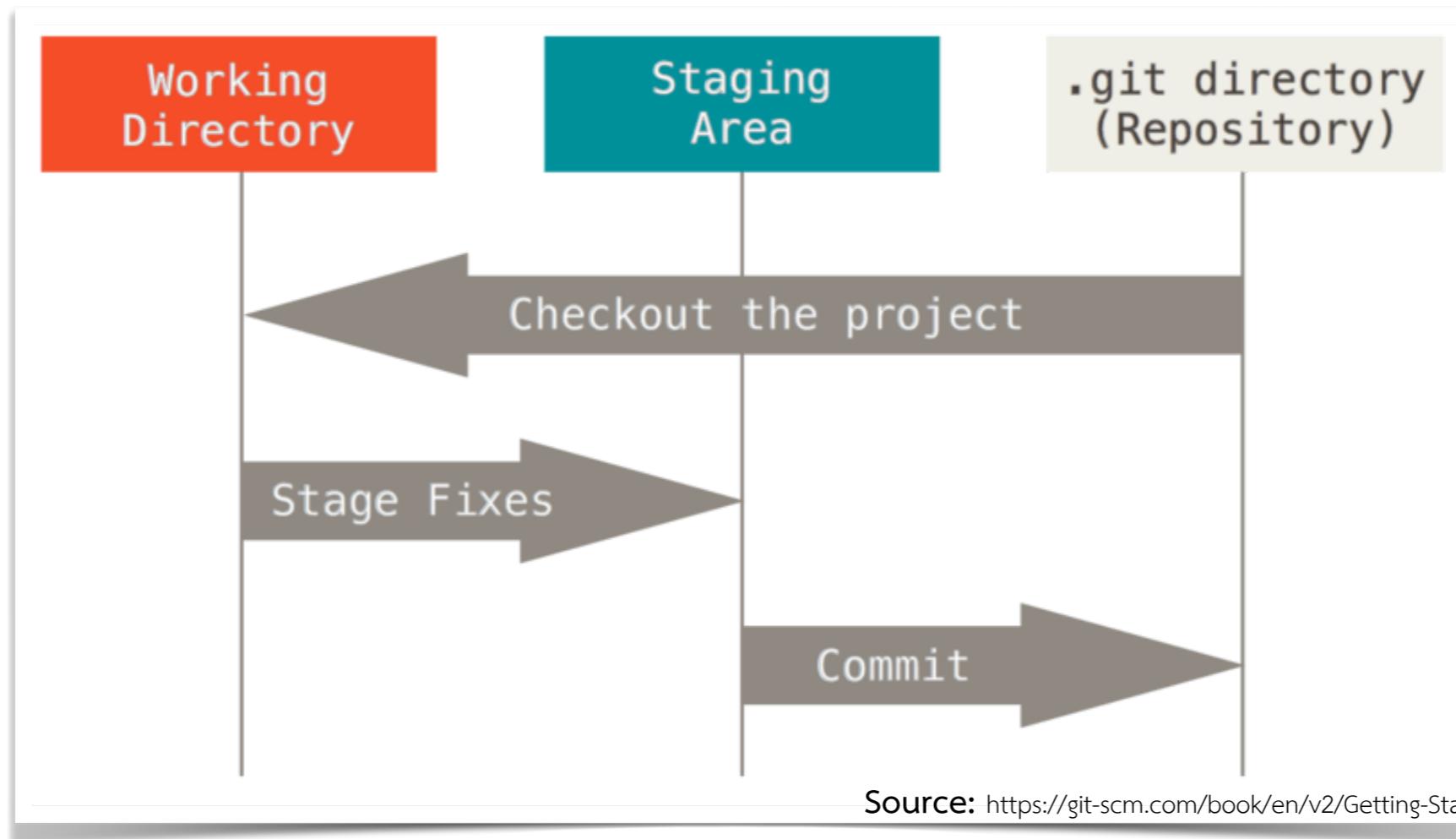
- mark ว่า ต้องการจะเก็บไฟล์นี้เข้า commit snapshot

- **Committed**

- เก็บข้อมูลการเปลี่ยนแปลงเข้า git แล้ว

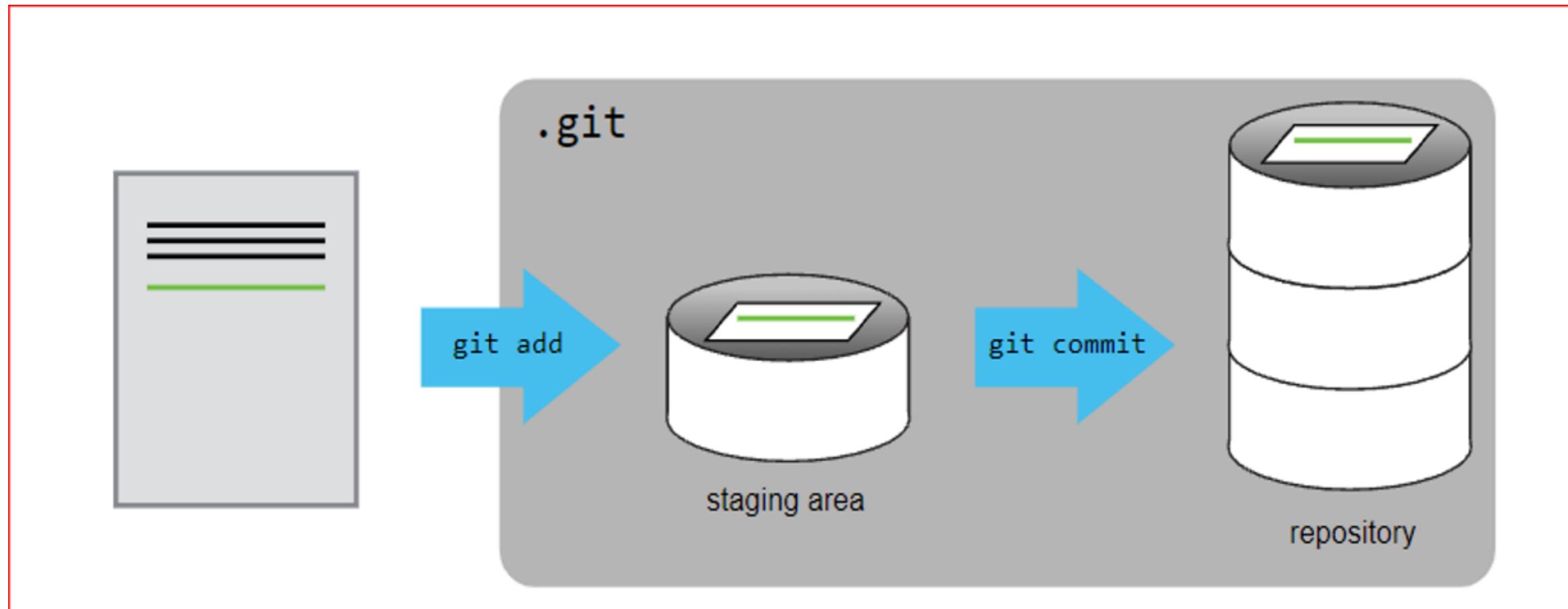
Git has 3 sections ของ git project

- **Working tree (working directory)** — พื้นที่เก็บไฟล์ที่ให้ git ช่วยจัดการ
- **Staging area** — พื้นที่เก็บไฟล์ที่จะเตรียม commit
- **.git directory** — git เก็บ metadata และ object database ไว้ที่ .git

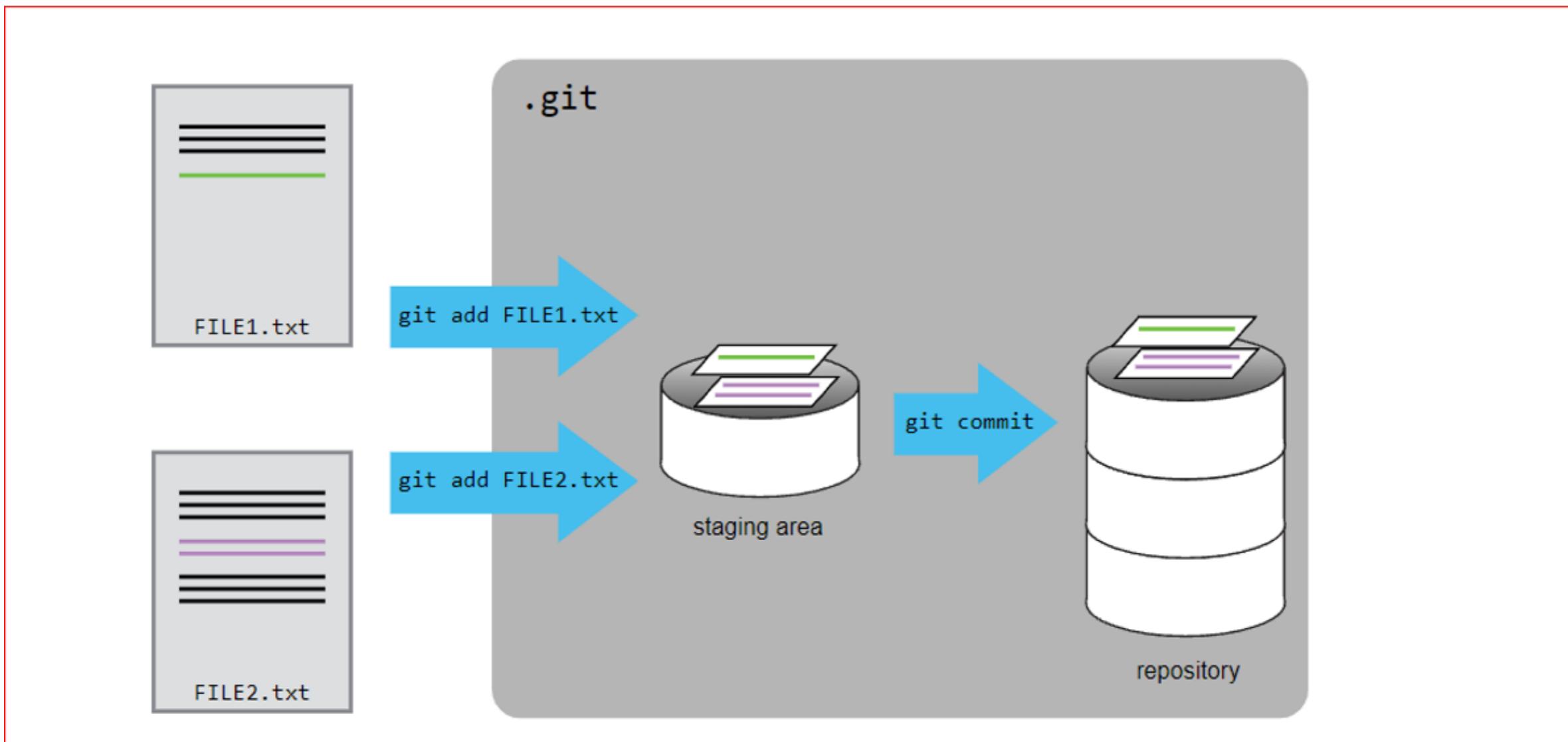


Git has 3 sections ของ git project

- **Working tree (working directory)** — พื้นที่เก็บไฟล์ที่ให้ git ช่วยจัดการ
- **Staging area** — พื้นที่เก็บไฟล์ที่จะเตรียม commit
- **.git directory** — git เก็บ metadata และ object database ไว้ที่ .git

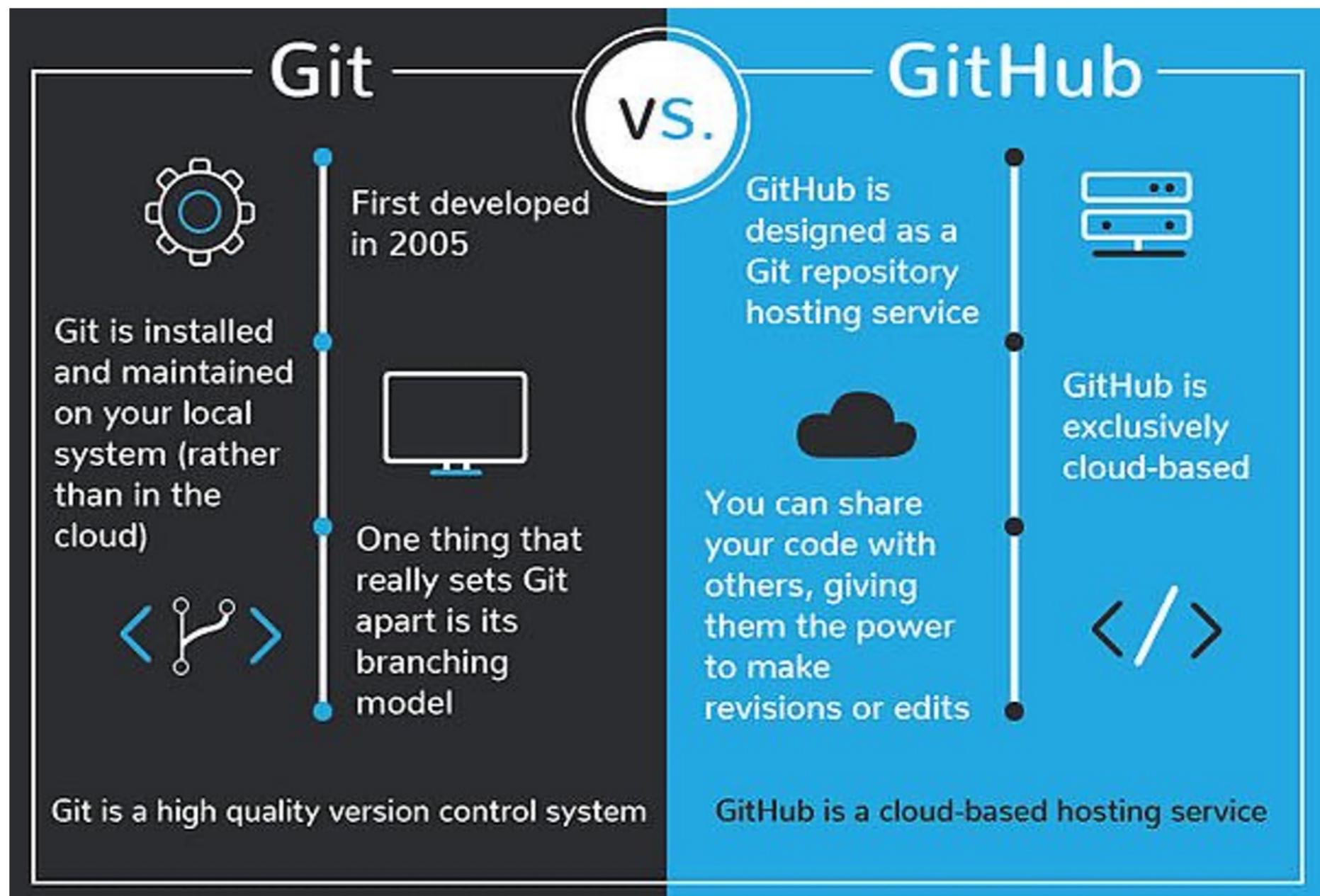


Git has 3 sections ຂອງ git project



VCS Server

- GitHub
- GitLab
- BitBucket



VCS Server

| | GitHub | GitLab | BitBucket |
|--------------------|--------|--------|-----------|
| Cloud-based | Y | Y | Y |
| On-Premise | N | Y | N |
| Private repository | Y | Y | Y |

การติดตั้ง

- 3 ขั้นตอน
 - 1. git
 - 2. เครื่องมือ GUI เพื่อช่วยให้ใช้ git ได้ง่ายขึ้น (Optional)
 - ถ้าไม่ใช้ GUI สามารถใช้ผ่าน command line ได้
 - 3. ลงทะเบียน cloud-based git hosting service / ติดตั้ง server on premise

การติดตั้ง git

- MacOS
 - มากับ Xcode
- Windows
 - ติดตั้ง git bash
 - <https://gitforwindows.org/>

การติดตั้ง git GUI

- **SourceTree**
 - <https://www.sourcetreeapp.com/>
 - Free to use
- **GitKraken**
 - <https://www.gitkraken.com/>
 - มี free version แต่การใช้งานจำกัดมาก

ลงทะเบียน server

- GitHub
 - ลองใช้ GitHub
 - <https://github.com/>

ขั้นตอนการใช้งาน

- flow การใช้งานต่าง ๆ
 - เริ่มต้นใช้งาน
 - Flow ทั่วไป ในการ commit และอัพชัน server
 - การดู log
 - ดึงไฟล์จาก server ลงมาใช้งาน
 - การ branch และ merge
 - การตั้งชื่อ version
 - การ remove ไฟล์

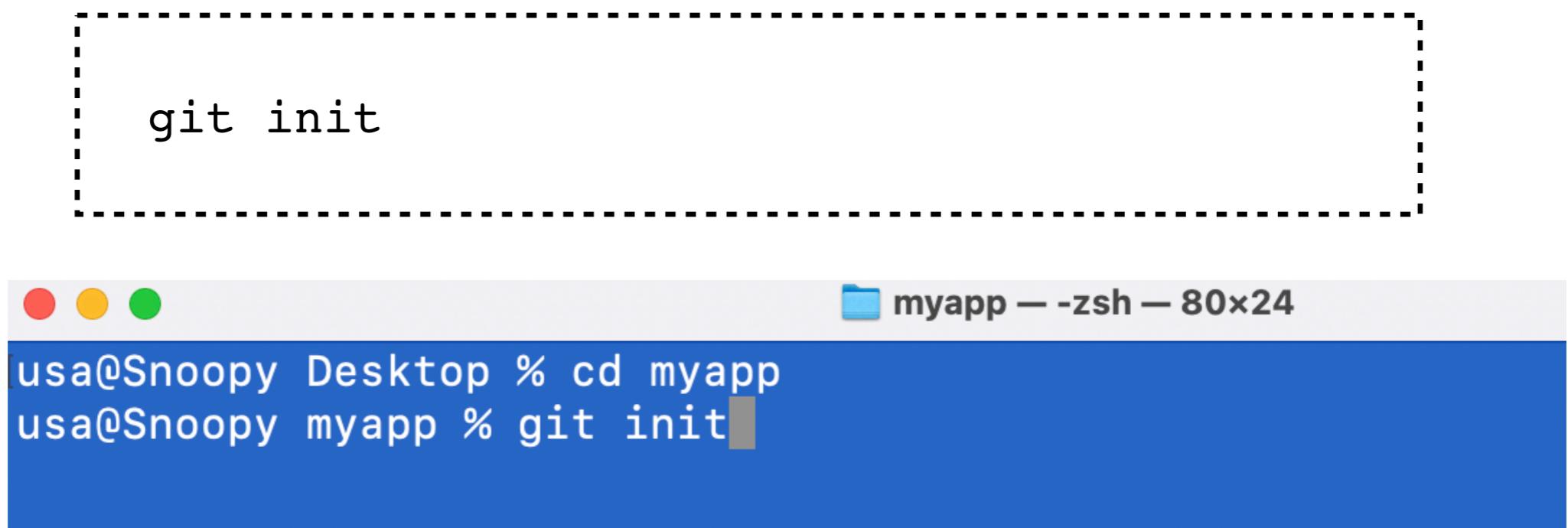
Flow 1: เริ่มต้นการใช้งาน

- Step 1: สร้าง repository เพื่อเริ่มใช้งาน
- Step 2: กำหนดชื่อและอีเมลของเรา

Step 1: สร้าง repository เพื่อเริ่มใช้งาน

- แบบ command line
 - ต้องอยู่ในโฟลเดอร์ที่ (จะ) มีไฟล์ก่อน จึงจะรัน command

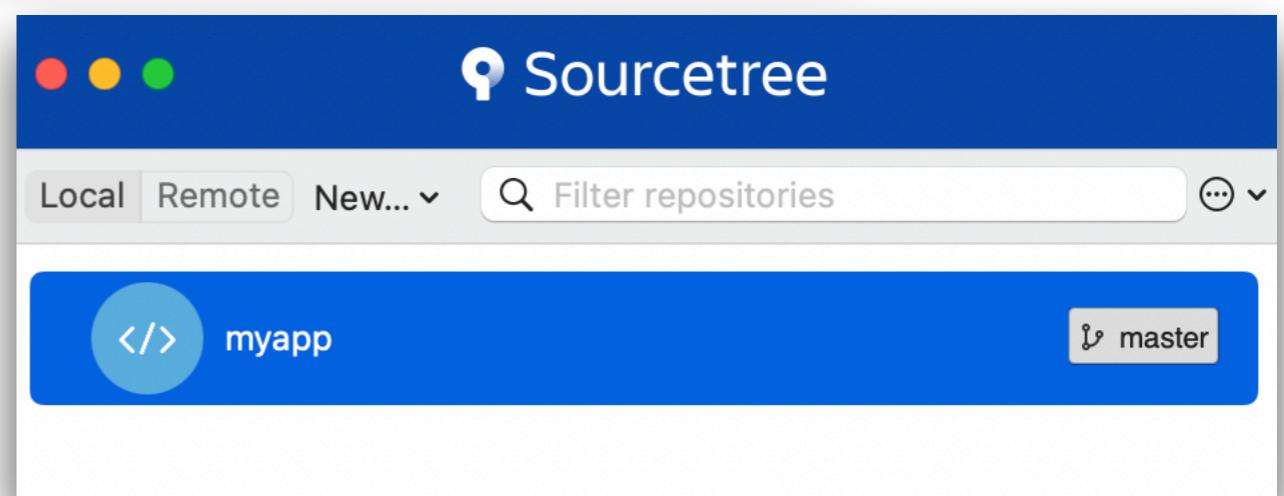
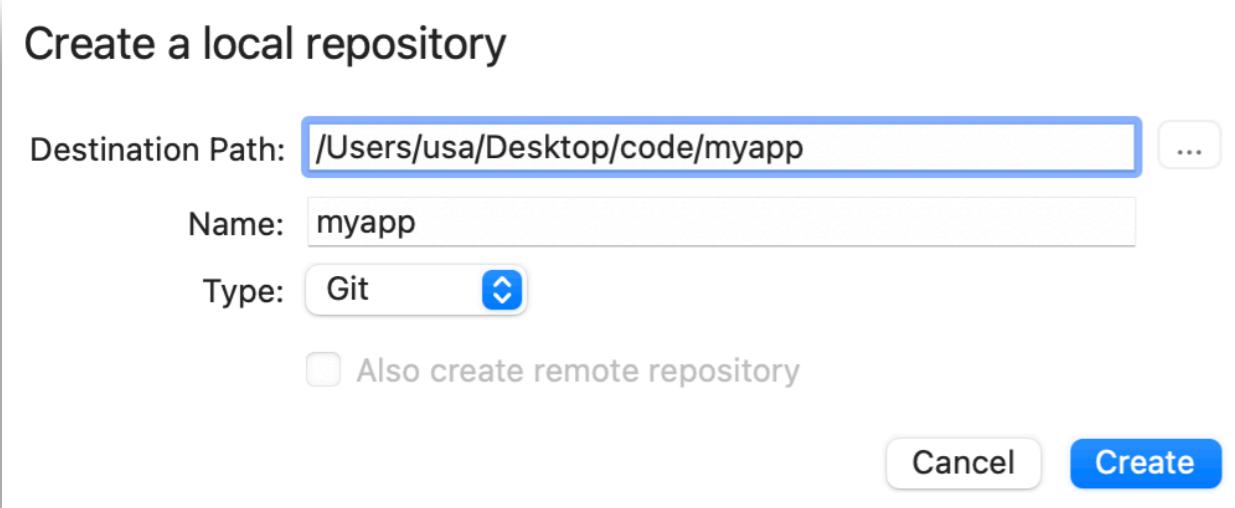
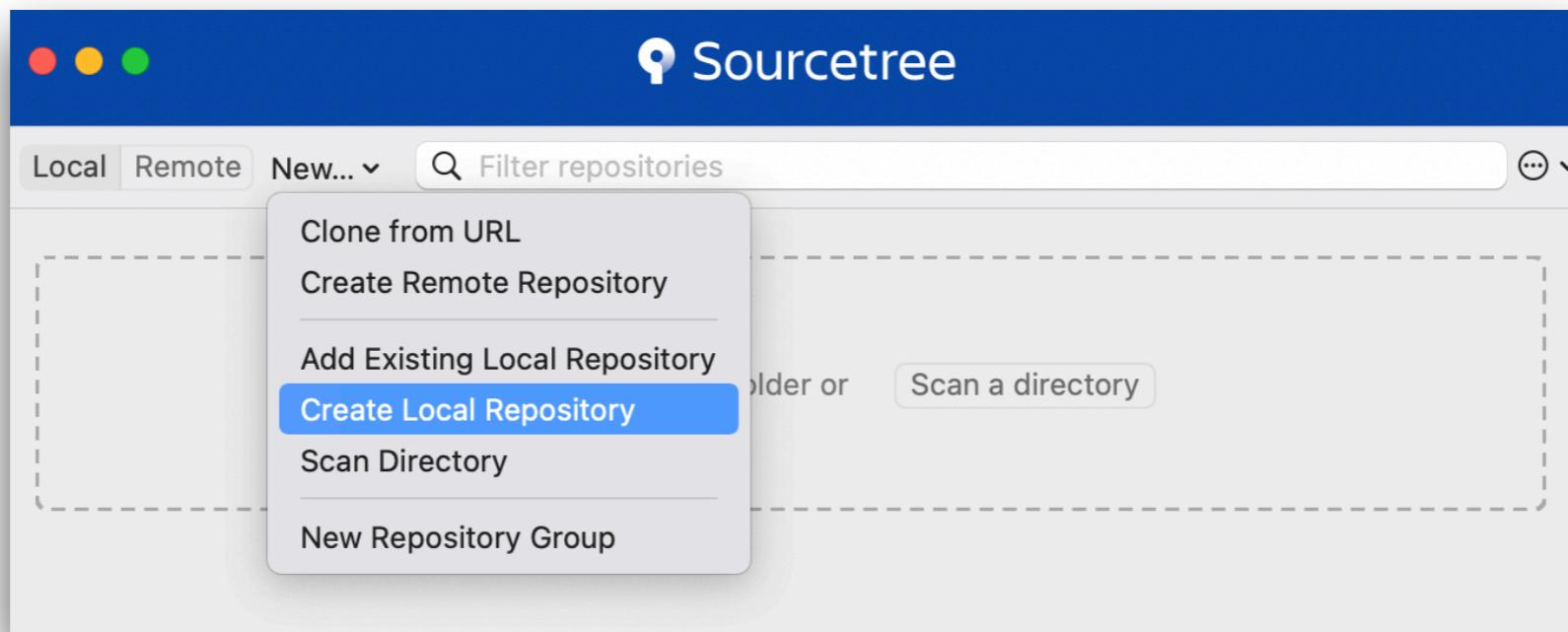
```
git init
```



```
[usa@Snoopy Desktop % cd myapp
usa@Snoopy myapp % git init]
```

Step 1: สร้าง repository เพื่อเริ่มใช้งาน

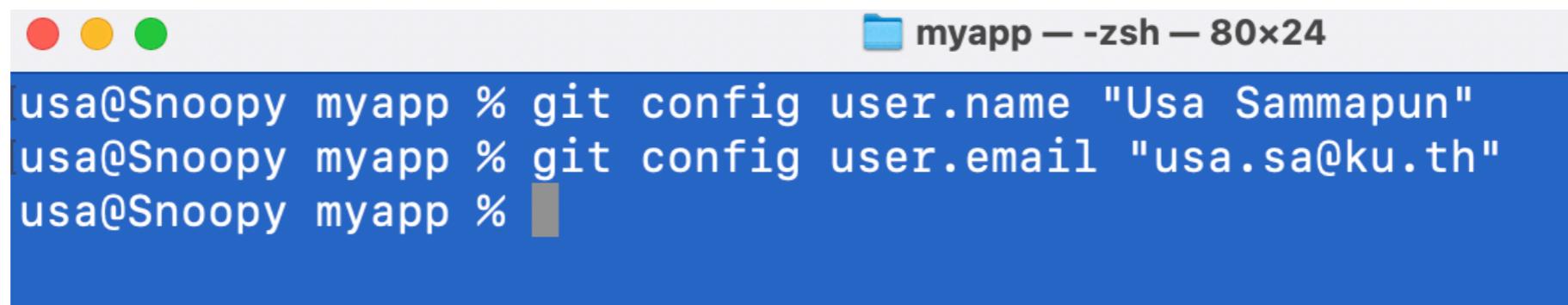
- แบบ source tree



Step 2: กำหนดชื่อและอีเมลของเรา

- แบบ command line
 - ถ้าใช้ --global จะ set ให้สำหรับทุก repo

```
git config user.name "FIRST_NAME LAST_NAME"
git config user.email "MY_NAME@example.com"
```

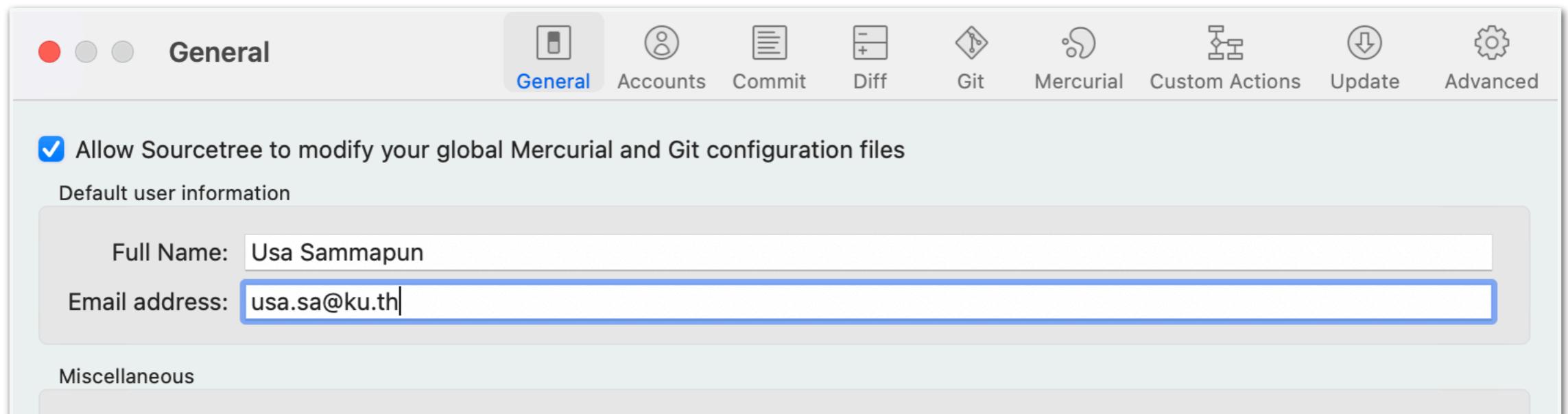
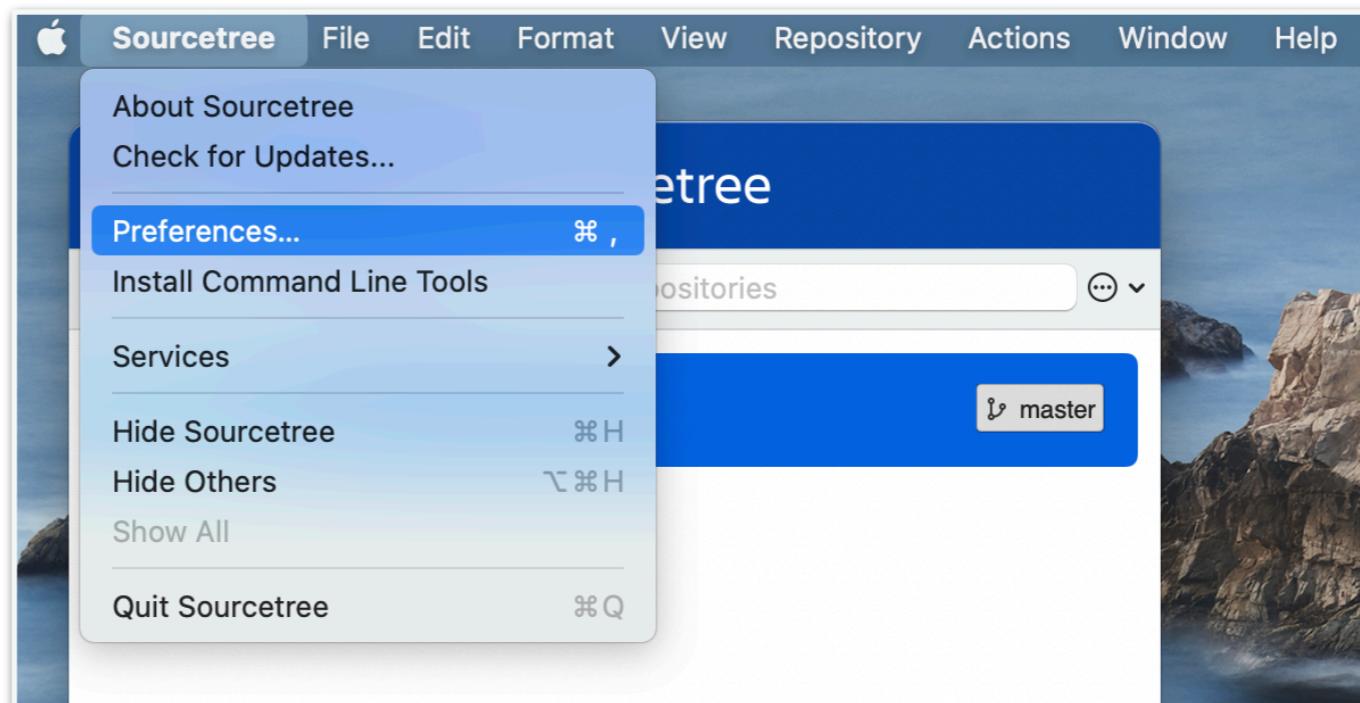


The screenshot shows a terminal window with the title "myapp — -zsh — 80x24". The user has run the following commands:

```
usa@Snoopy myapp % git config user.name "Usa Sammapun"
usa@Snoopy myapp % git config user.email "usa.sa@ku.th"
usa@Snoopy myapp %
```

Step 2: กำหนดชื่อและอีเมลของเรา

- แบบ source tree



Flow ทั่วไป ในการ commit และอัพขึ้น server

- git มีคำสั่ง status, pull, add, commit, และ push
 - ใช้ผ่าน command line หรือผ่าน source tree ได้
- ระหว่างเขียนโค้ด คุณจะมีขั้นตอนการทำงานต่อไปนี้

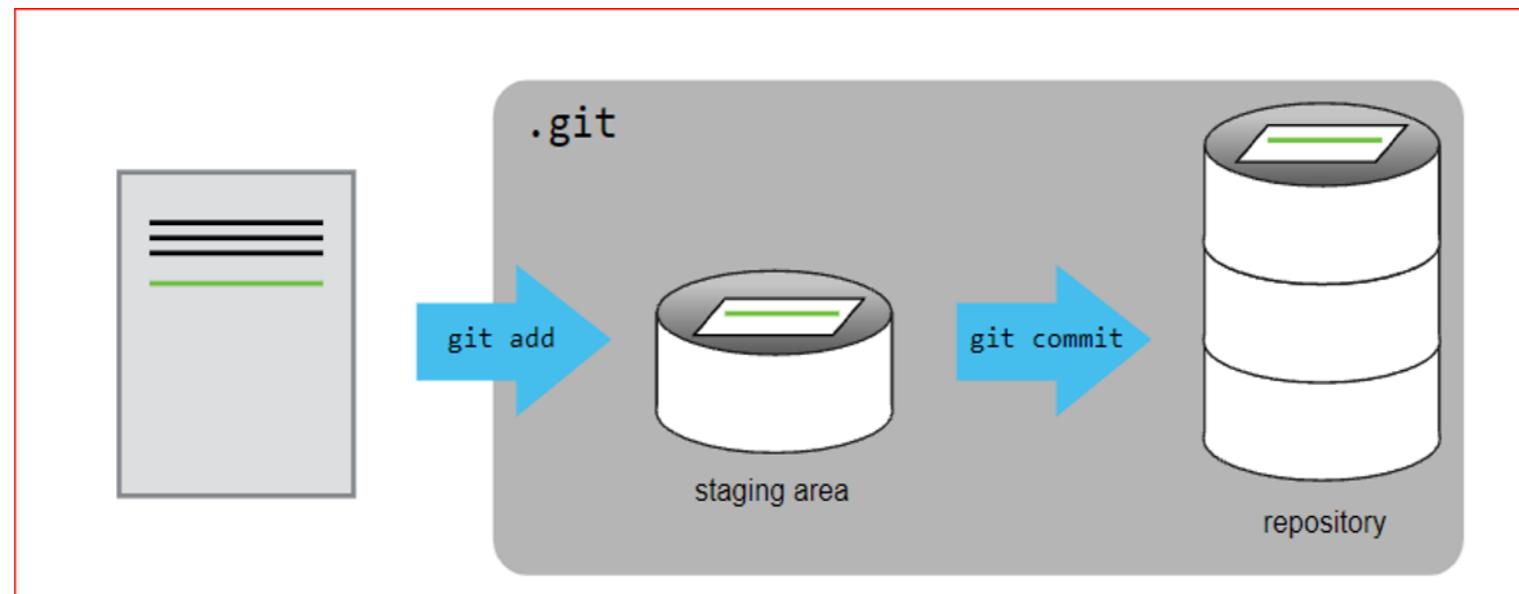
```
while true do
    1. status เพื่อดูว่ามีอะไรเปลี่ยนแปลงบ้าง
    2. pull ความเปลี่ยนแปลงที่อยู่บน server มา
    3. แก้ไขชอร์สโค้ดบนเครื่องส่วนตัว
    4. add ไฟล์ที่มีความเปลี่ยนแปลง
    5. commit ความเปลี่ยนแปลงที่คุณสร้าง
    6. push ความเปลี่ยนแปลงที่อยู่บนเครื่องส่วนตัวเข้าเครื่องเซิร์ฟเวอร์
```

Flow ทั่วไป ในการ commit และอัพขึ้น server

- ตัวอย่าง -- สร้างไฟล์ HelloWorld.java

- แบบ command line

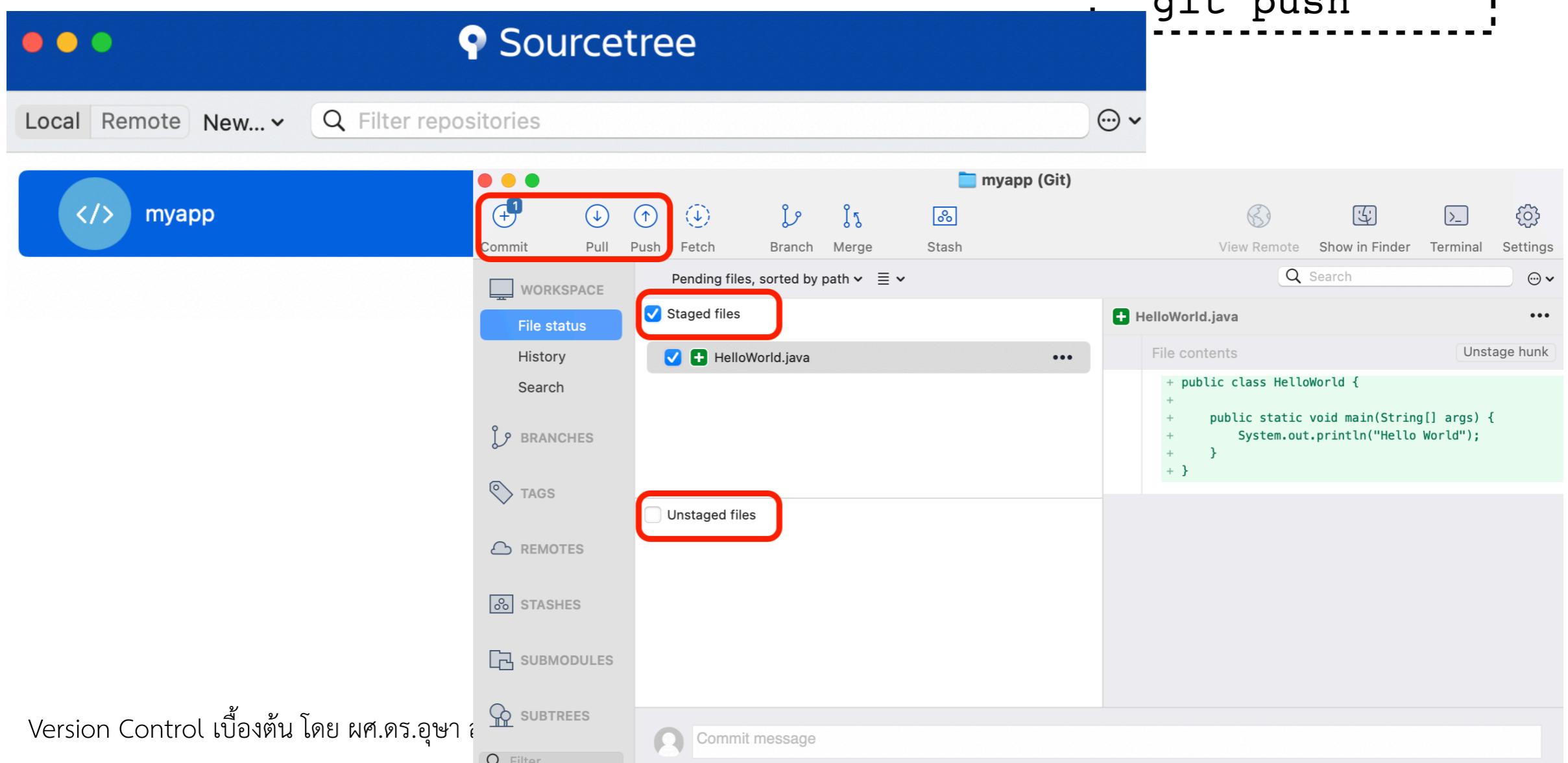
```
git add .
git commit -m "คำอธิบายการเปลี่ยนแปลง"
git push -u origin master
```



Flow ทั่วไป ในการ commit และอัพขึ้น server

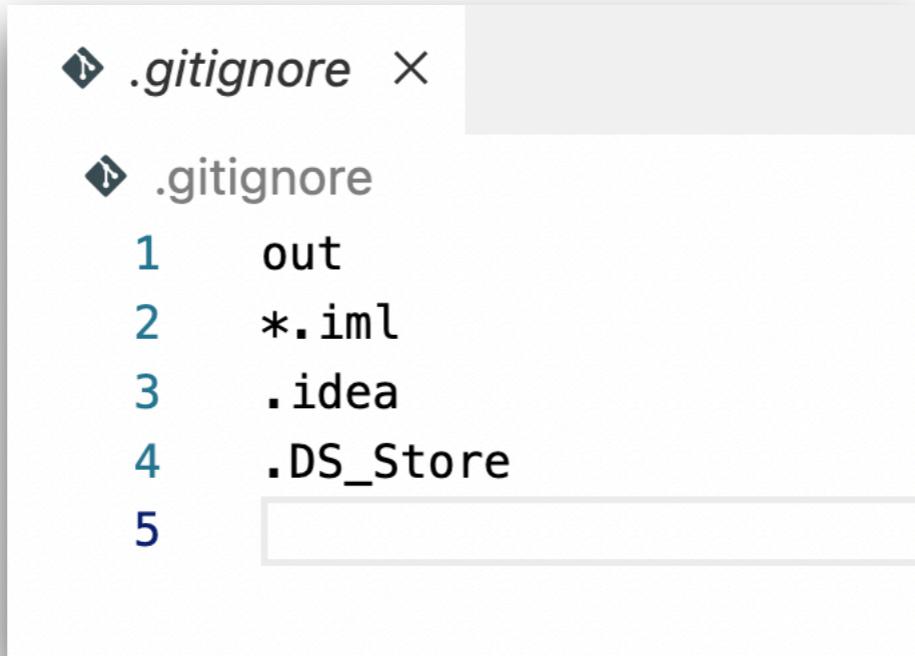
- ตัวอย่าง -- สร้างไฟล์ HelloWorld.java

- แบบ source tree



ไฟล์ .gitignore

- เป็นไฟล์ที่ระบุว่า จะไม่ commit ไฟล์ใดบ้าง
 - <https://github.com/github/gitignore>
- เช่น



```
❶ out
❷ *.iml
❸ .idea
❹ .DS_Store
❺
```

การดู log

- ดูประวัติการเปลี่ยนแปลงไฟล์
 - แบบ command line

```
git log
```

```
usa@Snoopy myapp % git log
commit 1c336a5a6c588dbe44510a78a65fc9d200f5d7c8 (HEAD -> master)
Author: Usa Sammapun <usa.sa@ku.th>
Date:   Tue Aug 23 21:22:29 2022 +0700

    Update Cat and add Dog

commit 63c822497295b975867725c7a55361f78c79de6c
Author: Usa Sammapun <usa.sa@ku.th>
Date:   Tue Aug 23 21:21:10 2022 +0700

    initial commit : add Cat
```

การดู log

- ดูประวัติการเปลี่ยนแปลงไฟล์
 - แบบ command line

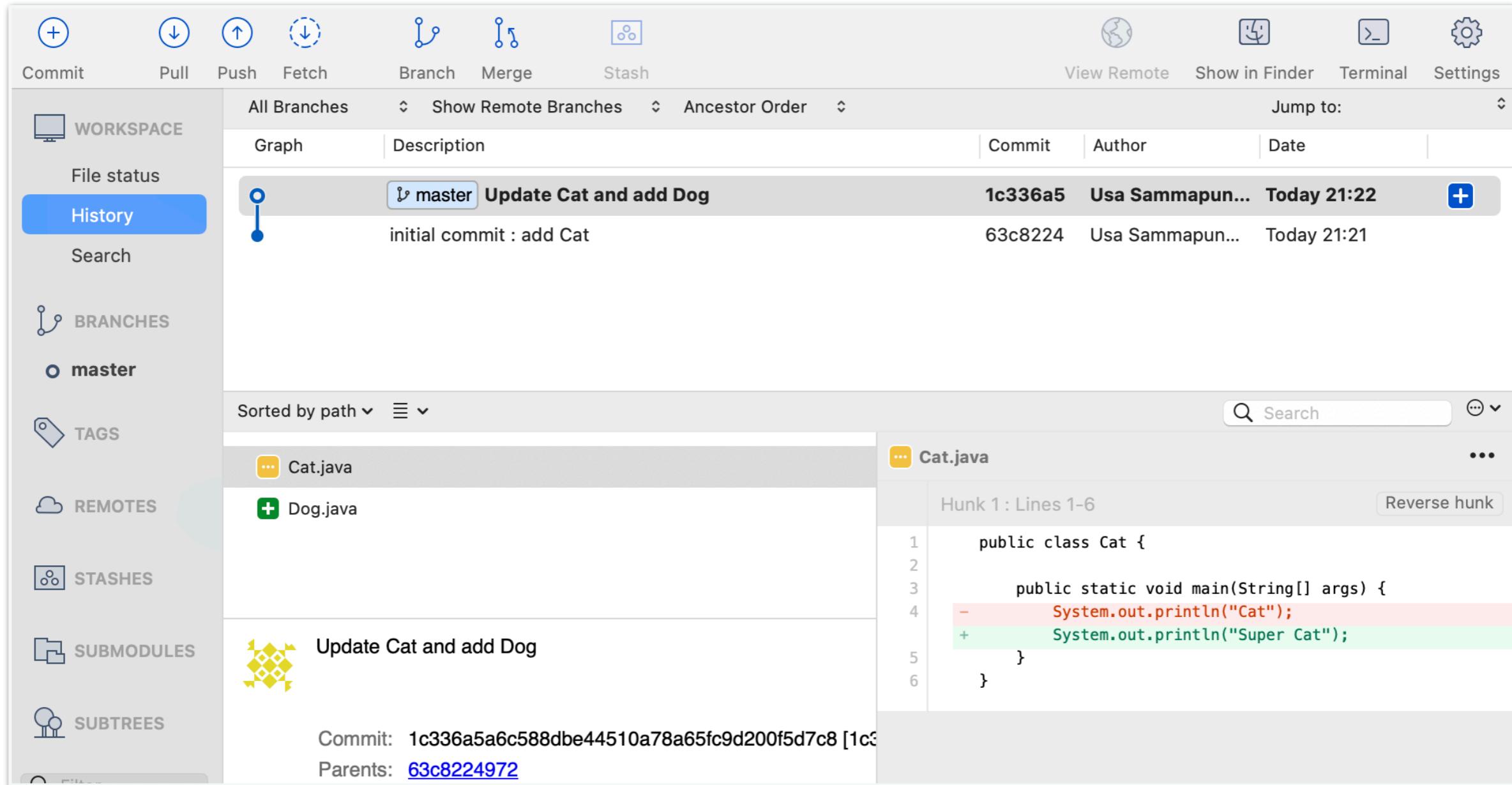
```
git log --all --decorate --oneline --graph
```

```
[usa@Snoopy myapp % git log --all --decorate --oneline --graph
* 1c336a5 (HEAD -> master) Update Cat and add Dog
* 63c8224 initial commit : add Cat
```

การดู log

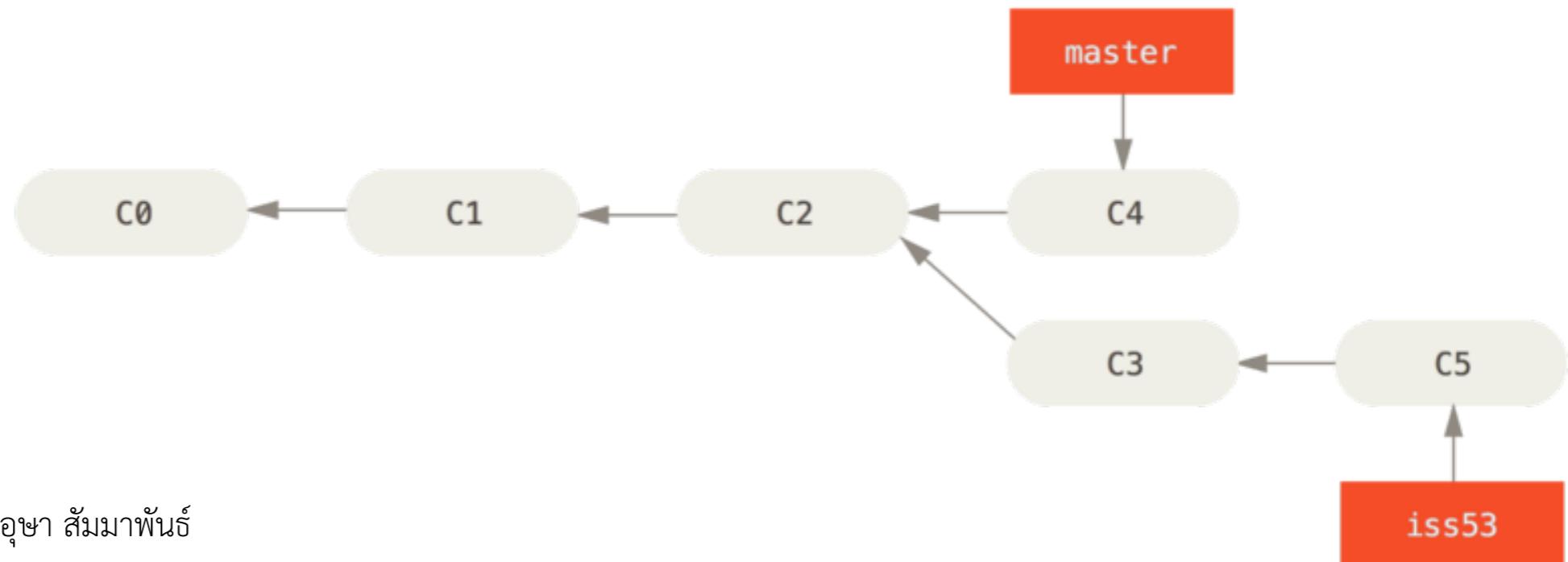
- ดูประวัติการเปลี่ยนแปลงไฟล์

- แบบ source tree



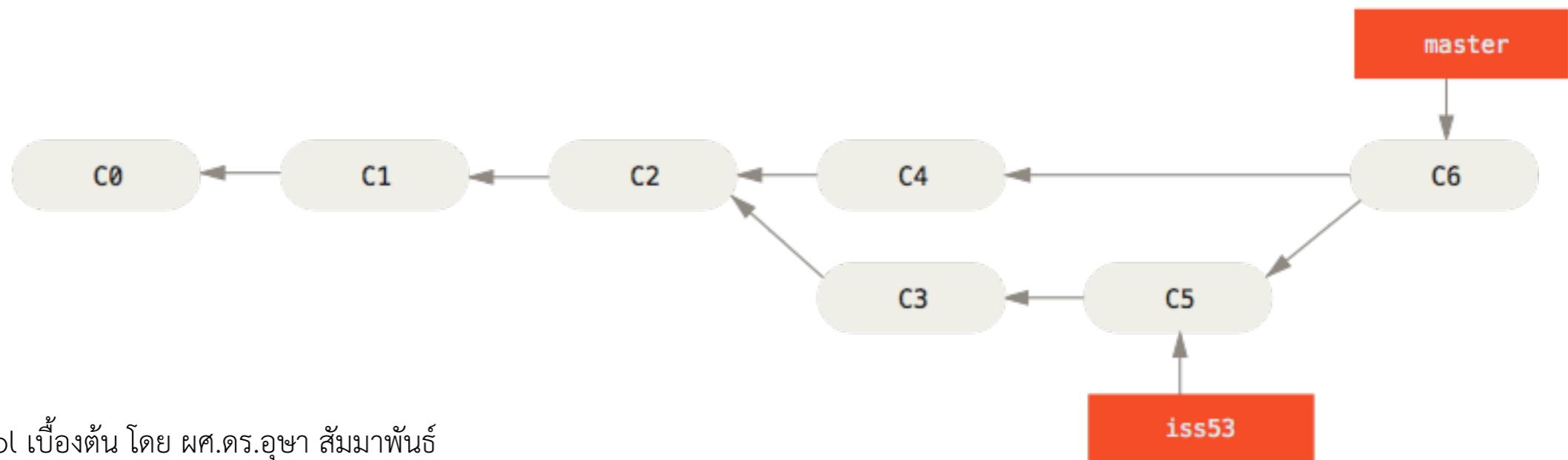
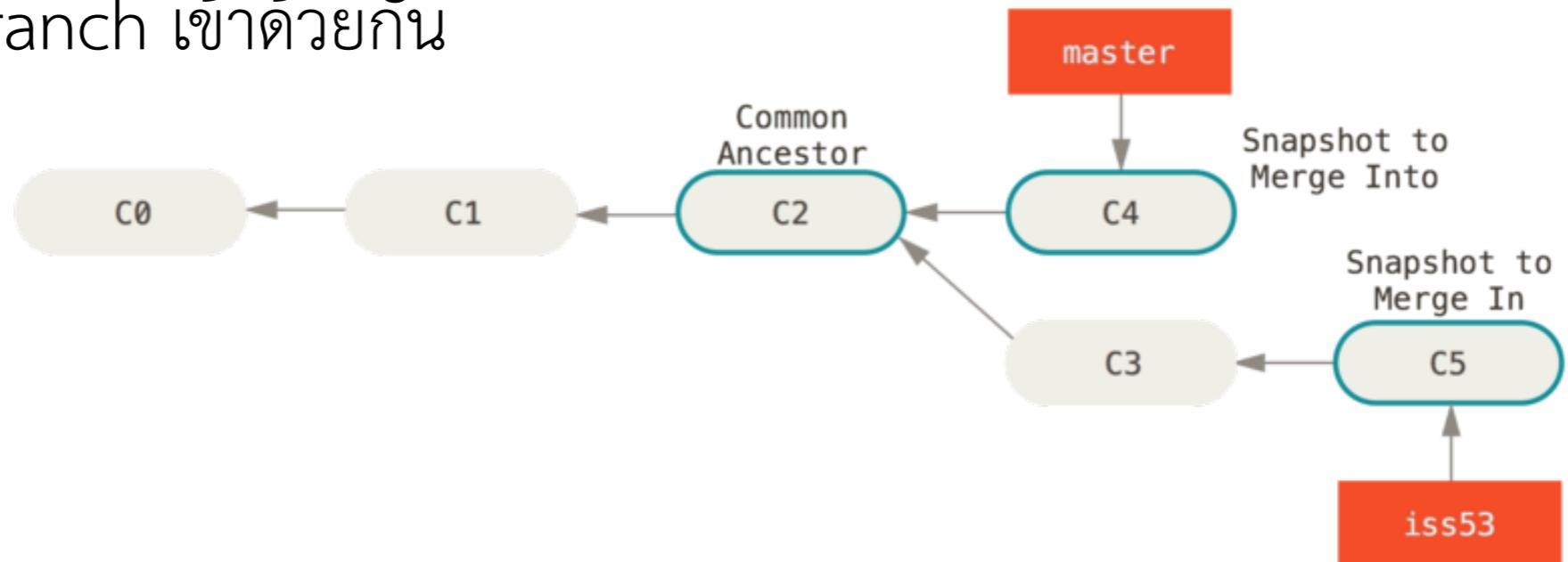
Branch

- Branch หลัก
 - Git — master
 - GitHub — main
- แตก branch ออกมาจาก branch หลักได้ เช่น
 - สำหรับ feature ใหม่
 - สำหรับแก้ bug



Merge

- รวมไฟล์จาก 2 branch เข้าด้วยกัน



Lab

- Git Basic
 - <https://docs.google.com/document/d/1pSczA4hPQ-32QNzfQkHL4VX2jghZmFBFeCf2TBi9k9E/edit?usp=sharing>
- Git Tag
 - <https://docs.google.com/document/d/1KVBV63HyOotPFRv0Thi2ic7Te9avFAVLV1SNwDK8HUM/edit?usp=sharing>
- Git Branch, Merge, Pull
 - <https://docs.google.com/document/d/1IVIjG6-kPL8JVPwePVO6ImPdEYCkE48WojYFmgrHcME/edit?usp=sharing>

Git Cat

- <https://girliemac.com/blog/2017/12/26/git-purr/>

Semantic versioning

- <https://semver.org/>

การทำงานร่วมกันผ่าน pull request

- <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/getting-started/about-collaborative-development-models>

Workflow แบบต่าง ๆ

- <https://www.atlassian.com/git/tutorials/comparing-workflows>

Key Takeaway

- การ merge file ใช้แรงกายแรงใจสูง
 - ถ้าออกแบบ workflow เพื่อให้คนจำนวนน้อยแก้ไฟล์เดียวกันจะดีมาก
 - และควร merge file บ่อย ๆ เพื่อให้เกิด conflict น้อย ๆ