Version Control

@Chalach.mo @Vattanan.bu

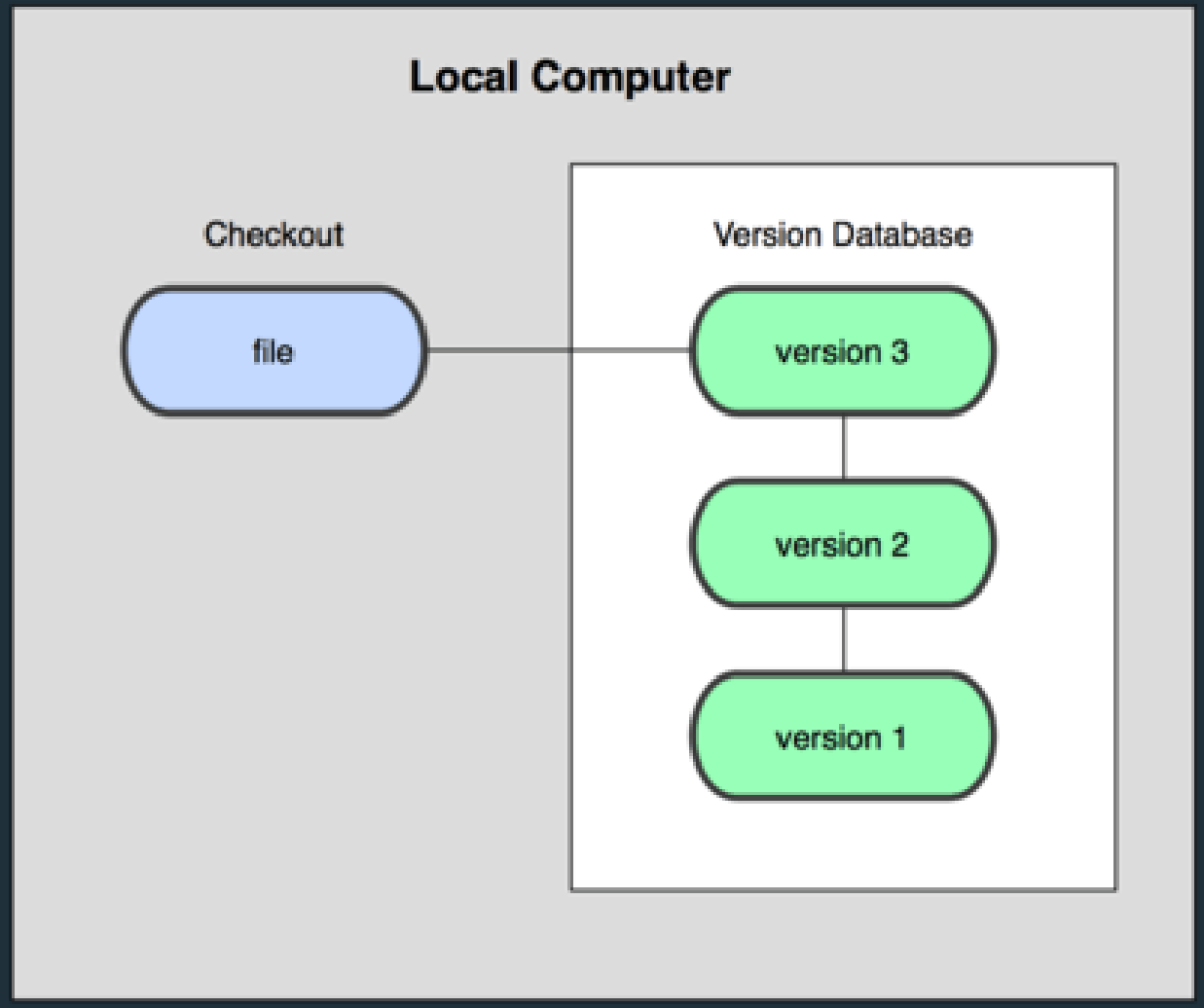
Version Control System(VCS)

- ระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์
- เพื่อที่จะสามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้
- version control กับไฟล์ชนิดใดก็ได้

Type Of Version Control

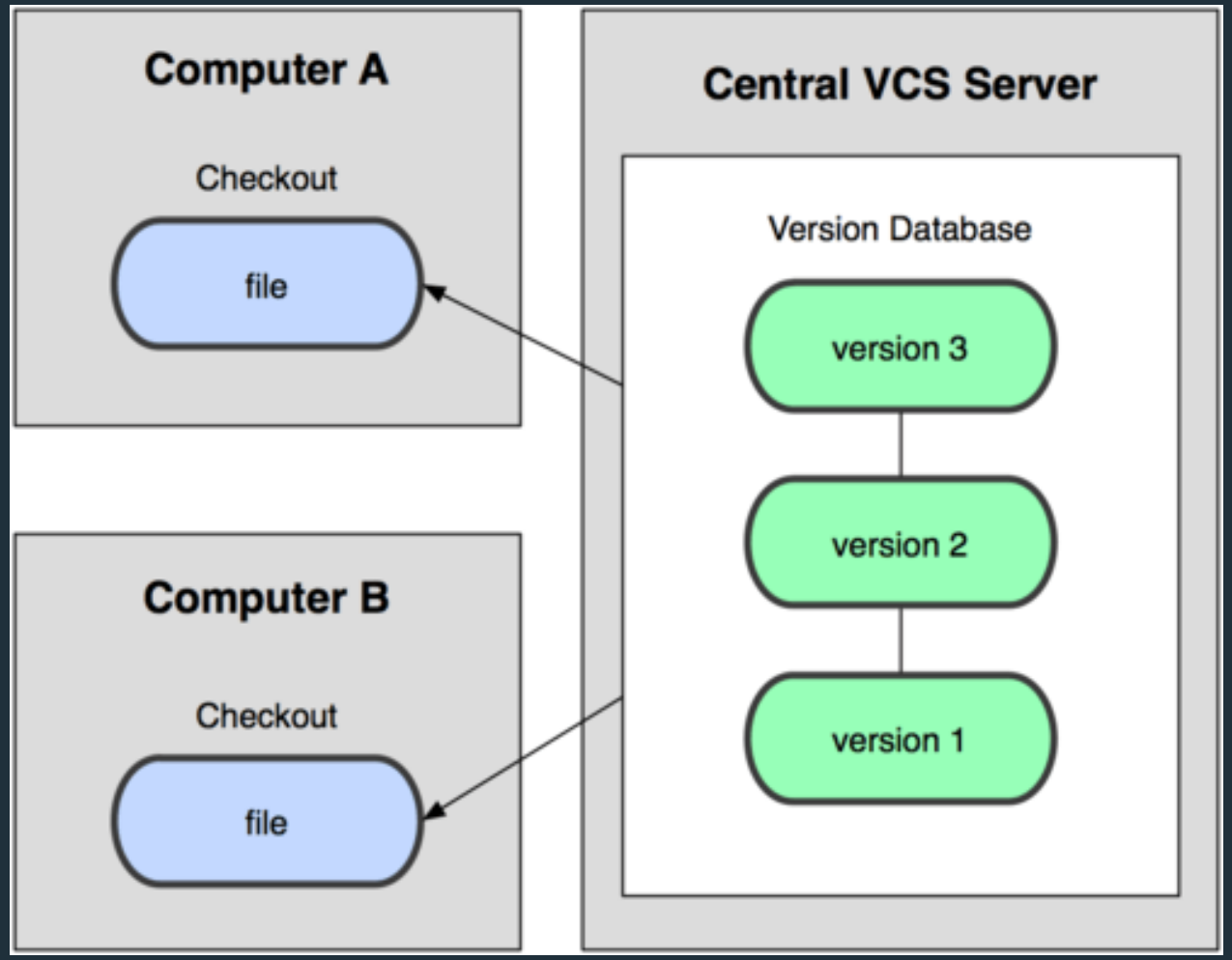
- Local Version Control (LVC)
- Centralized Version Control Systems (CVCSs)
- Distributed Version Control Systems (DVCSs)

Local Version Control



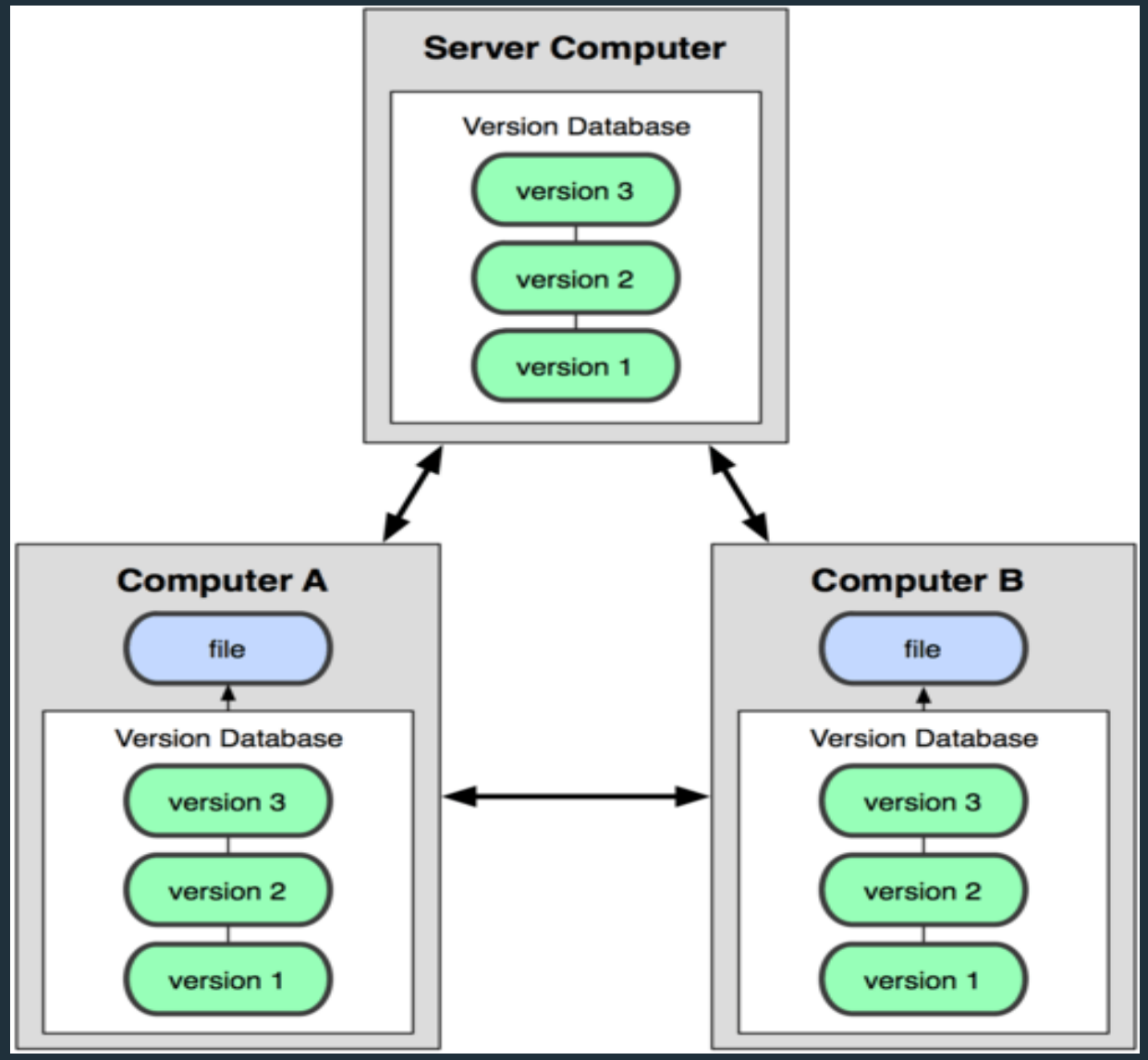
- การคัดลอกไฟล์ไปไว้ในไดเร็คทอรี่ใหม่
- เป็นวิธีที่เกิดข้อผิดพลาดได้ง่าย

Centralized Version Control



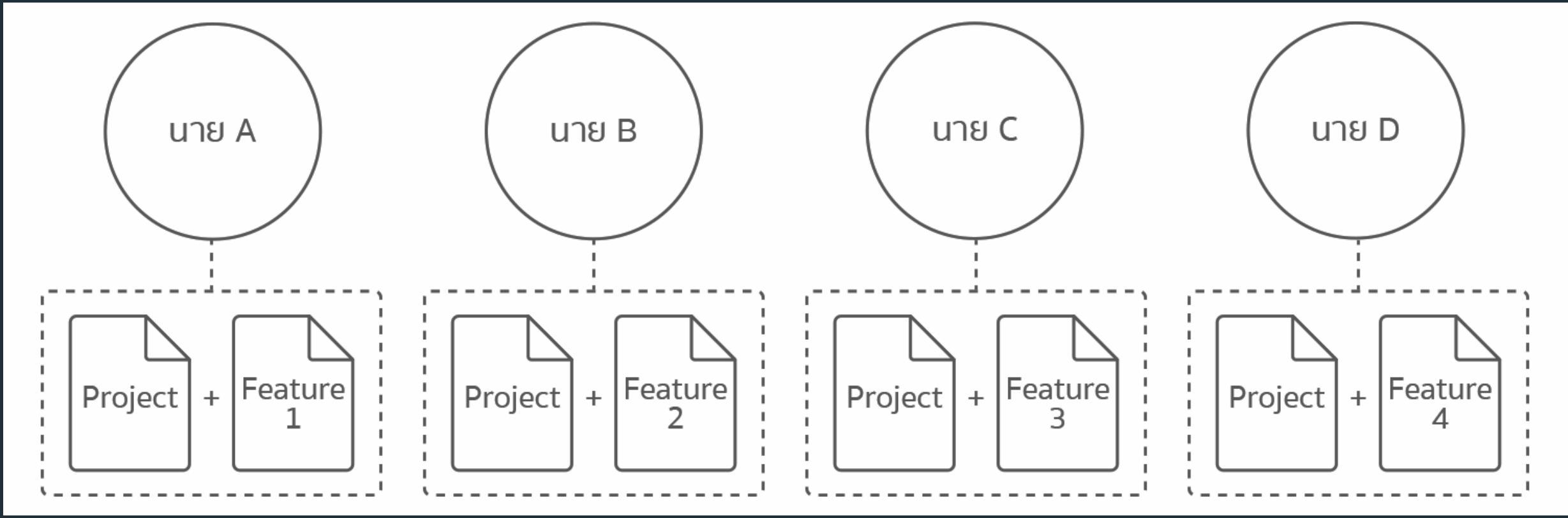
- แบบรวมศูนย์ เช่น CVS, SVN
- มี Server กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้
- ถ้า Server ล่ม ก็จะไม่มีใครสามารถใช้งานได้ หรือ หากเสีย ก็จะไม่สามารถนำข้อมูลกลับมาได้

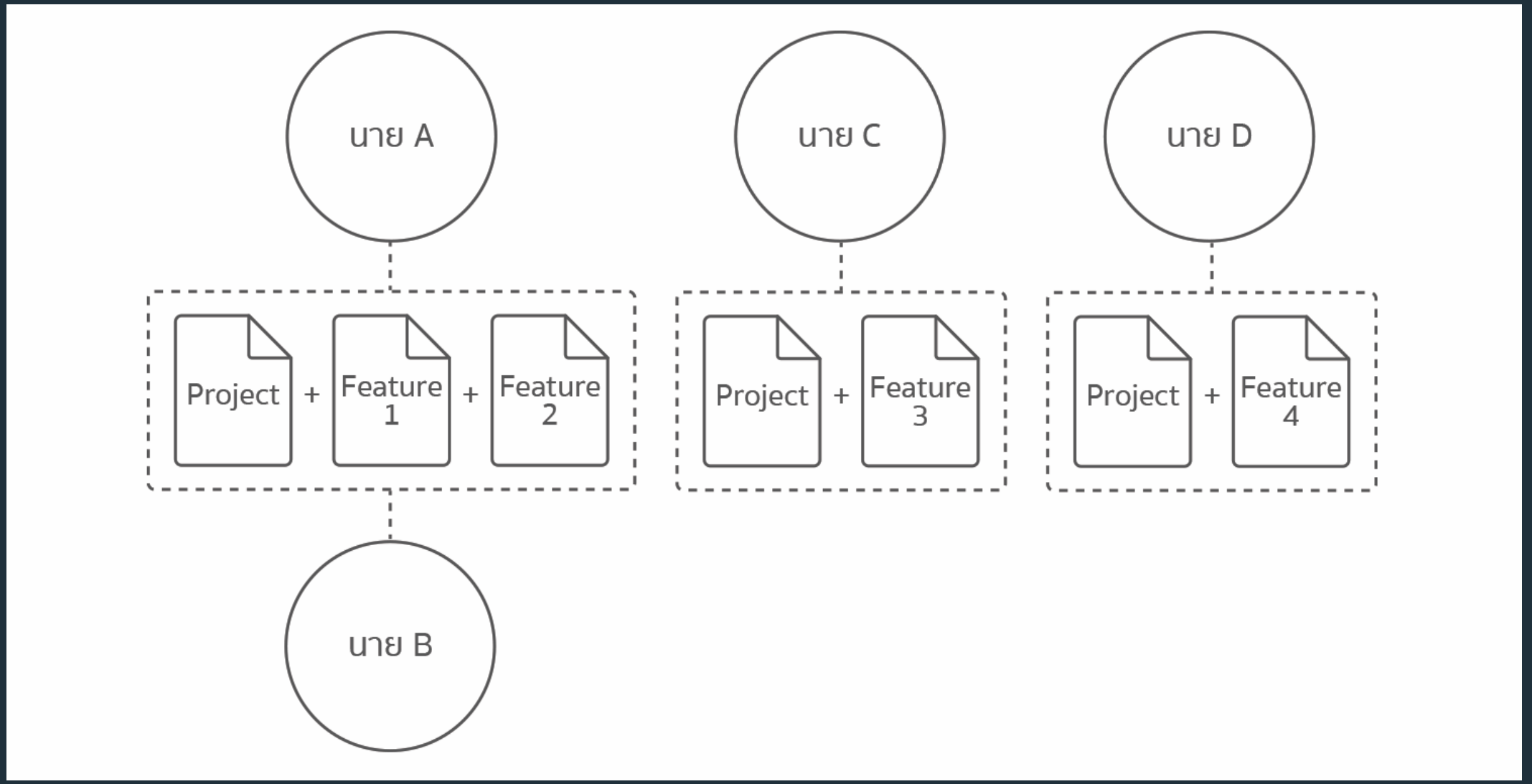
Distributed Version Control



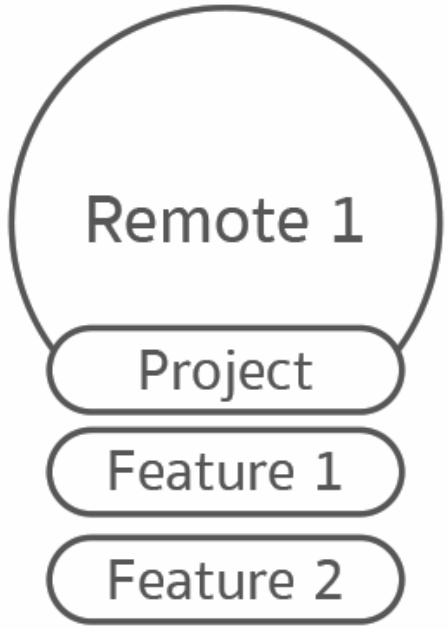
- แบบกระจายศูนย์ เช่น Git
- Copy มาทั้ง Repository
- Server เสีย Client ก็ยังทำงานบน Local ได้
- ทุกครั้งที่ Checkout คือ การ Backup file

Why Version Control ?



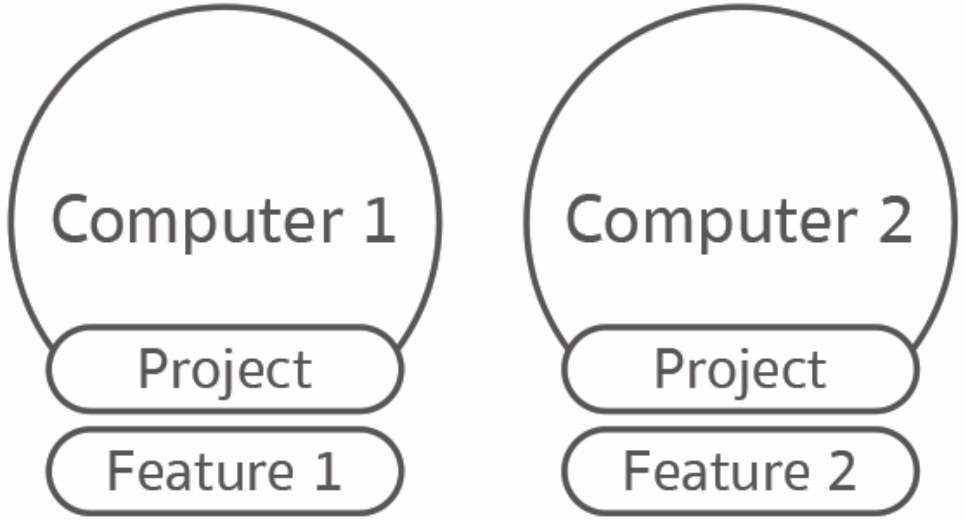

นาย A
นาย C
นาย D
Project
+
Feature 1
+
Feature 2
Project
+
Feature 3
Project
+
Feature 4
นาย B


git



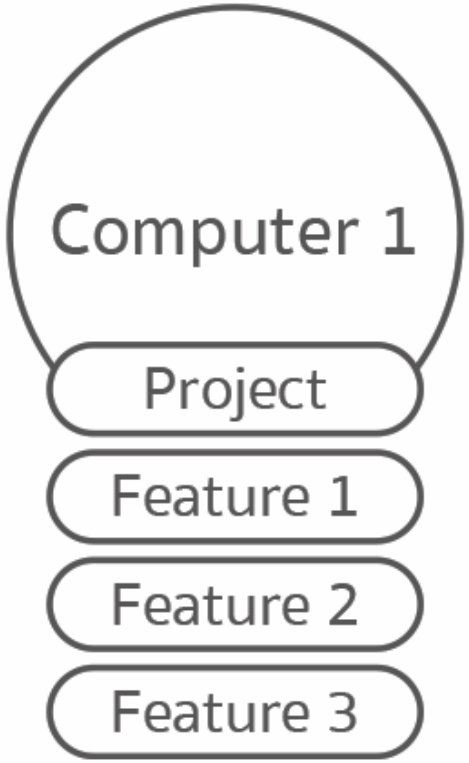
Server

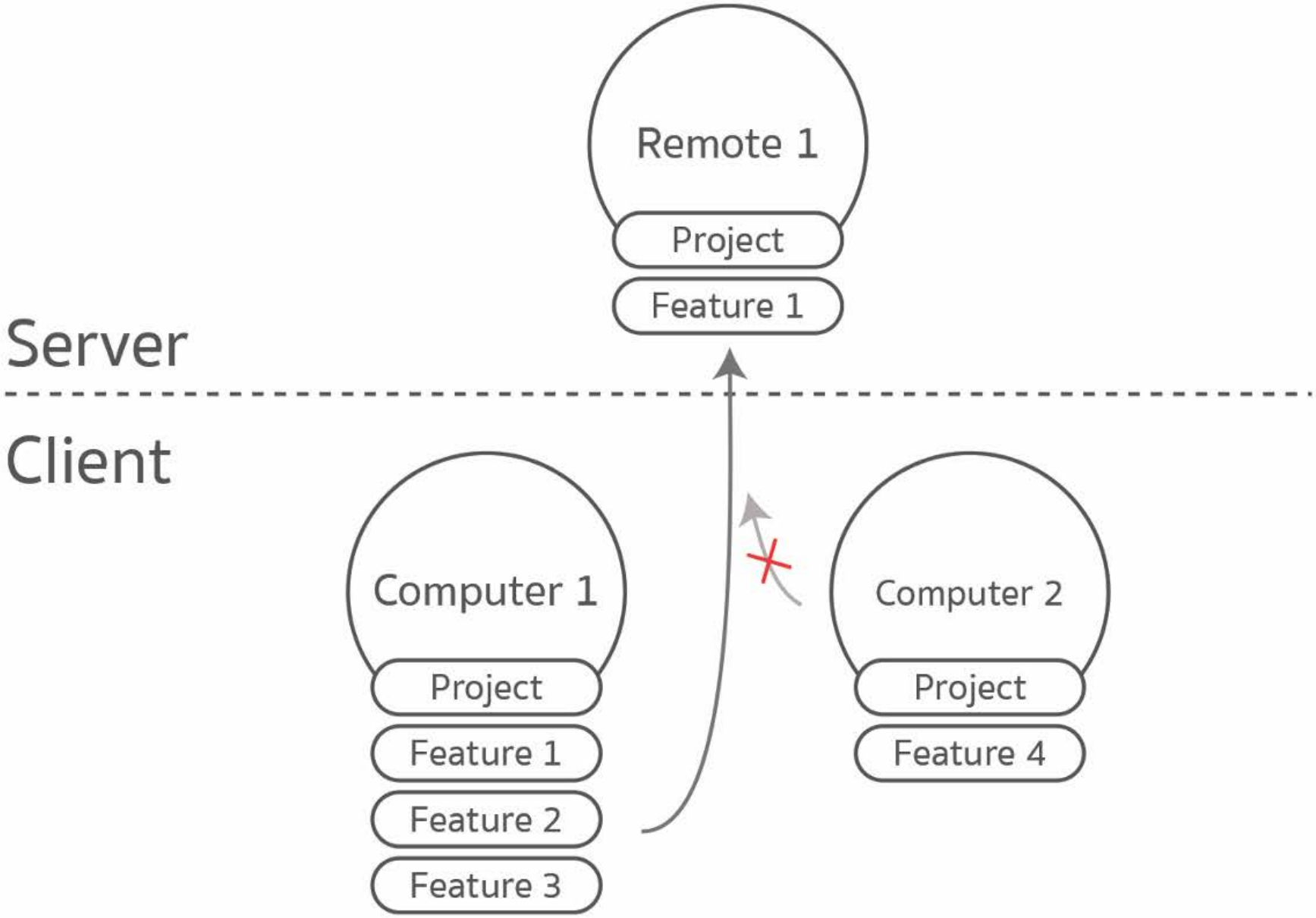
Client

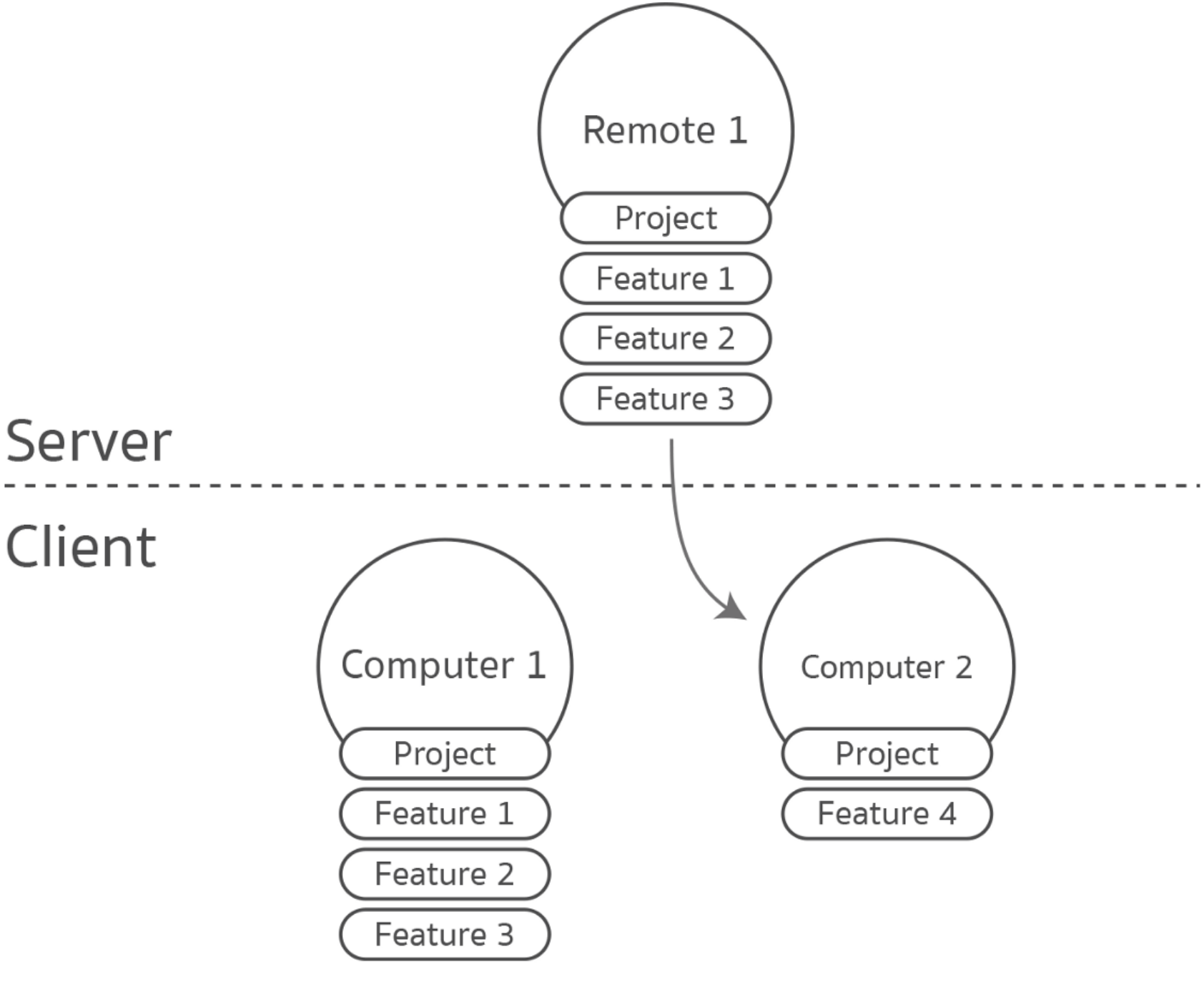


Server

Client




Remote 1
Project
Feature 1
Server
Client
Computer 1
Project
Feature 1
Feature 2
Feature 3
Computer 2
Project
Feature 4


Remote 1
Project
Feature 1
Feature 2
Feature 3
Server
Client
Computer 1
Project
Feature 1
Feature 2
Feature 3
Computer 2
Project
Feature 4

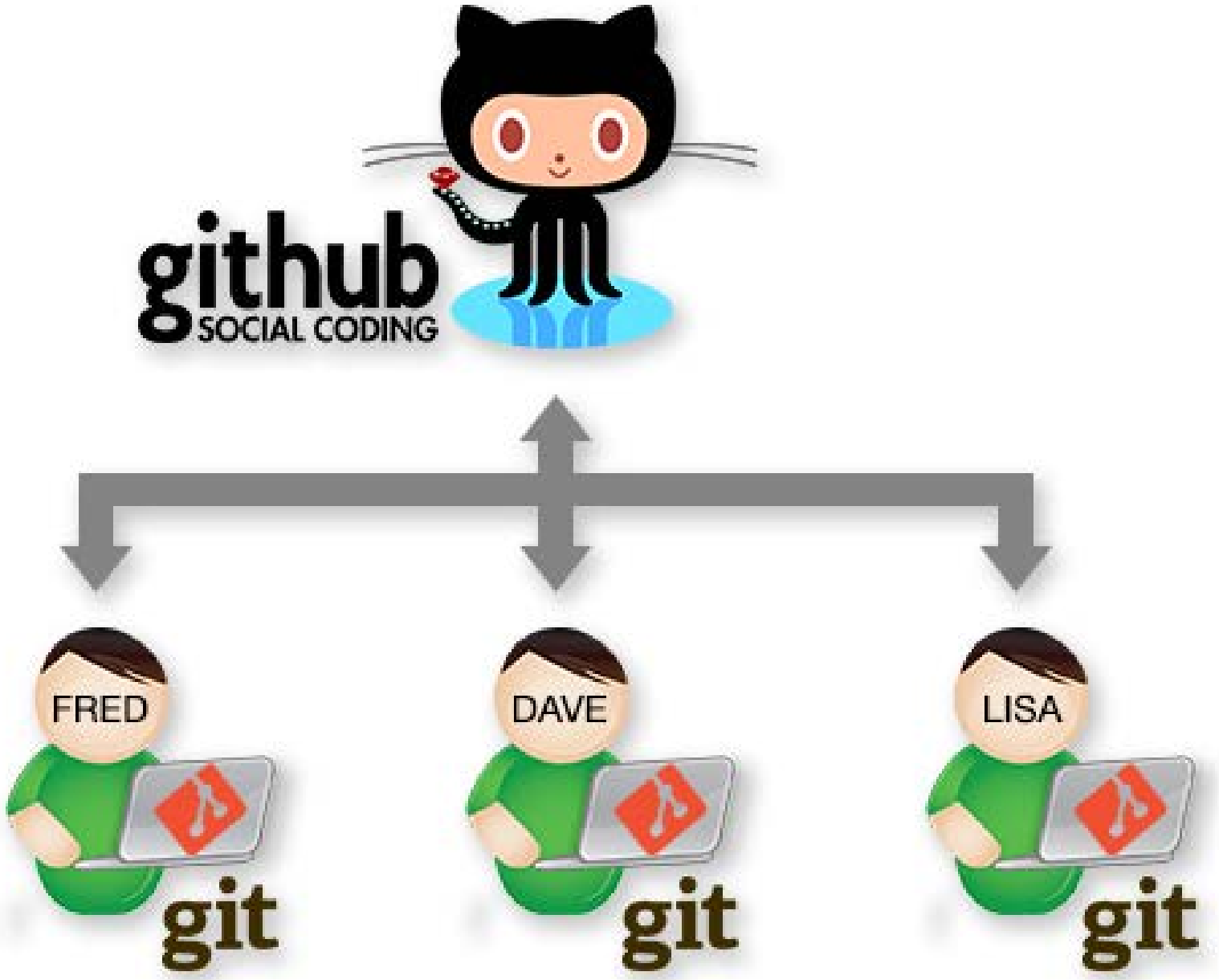

ALL RIGHT. WHO WROTE THE CODE THAT TANKED THE SERVER ?!!
WASN'T ME


V 1
V 2
V 3

git

github
SOCIAL CODING

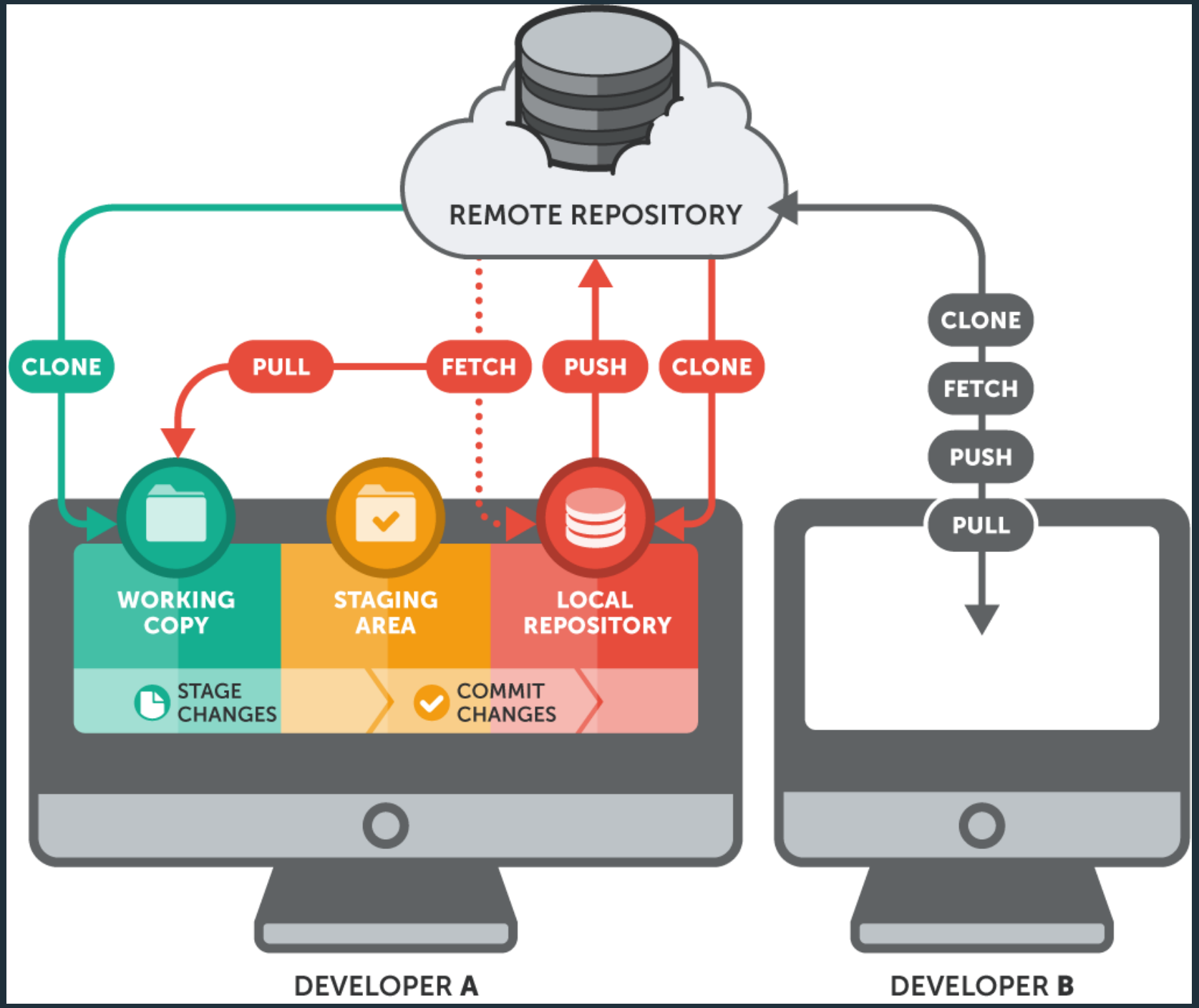
Whats the difference between Git & Github ?


github
SOCIAL CODING
FRED
git
DAVE
git
LISA
git


GitHub

Keywords of Git

- Repository
- Clone
- Commit
- Unstaged
- Staged
- Push
- Pull
- Fetch
- Conflict
- Merge Commit


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
CLONE
FETCH
PUSH
PULL
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
DEVELOPER B

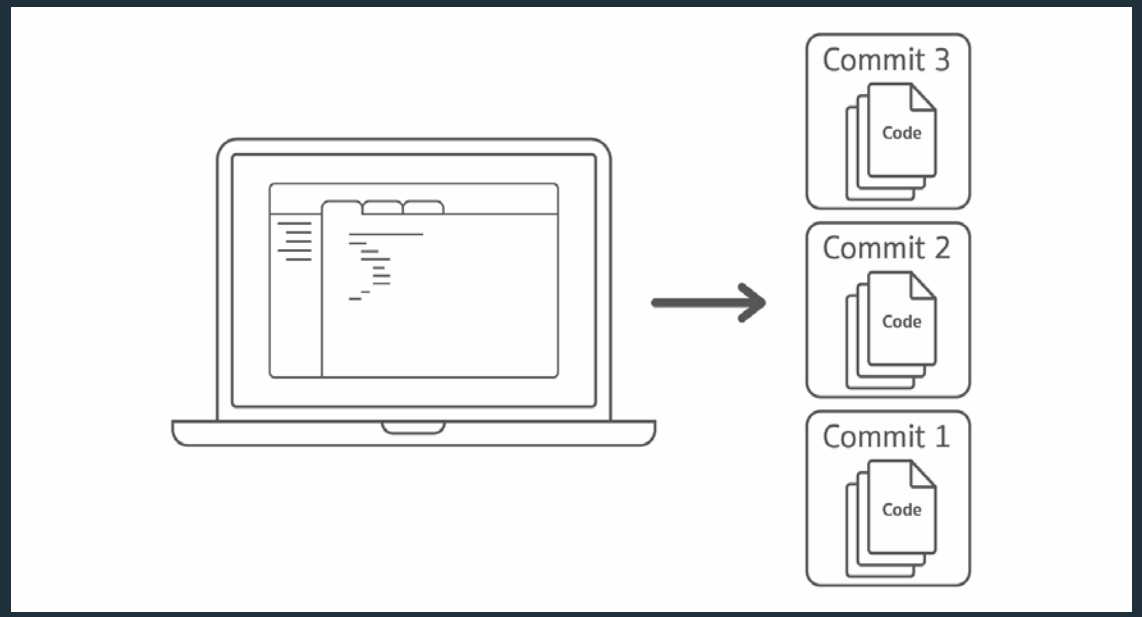
Repository

- Folder ที่เก็บ Project ของงานนั้นๆ
- 1 Repository สามารถเก็บ Project เท่าไรก็ได้ตามที่ต้องการ
- ส่วนใหญ่ก็นิยมเก็บ Project 1 ตัวต่อ 1 Repository

Clone

- คือ การ Sync Code มาลงเครื่องของเรา
- เรียกว่า *Clone* Repository
- หรือการ Copy Repository จาก Remote มาลงเครื่อง

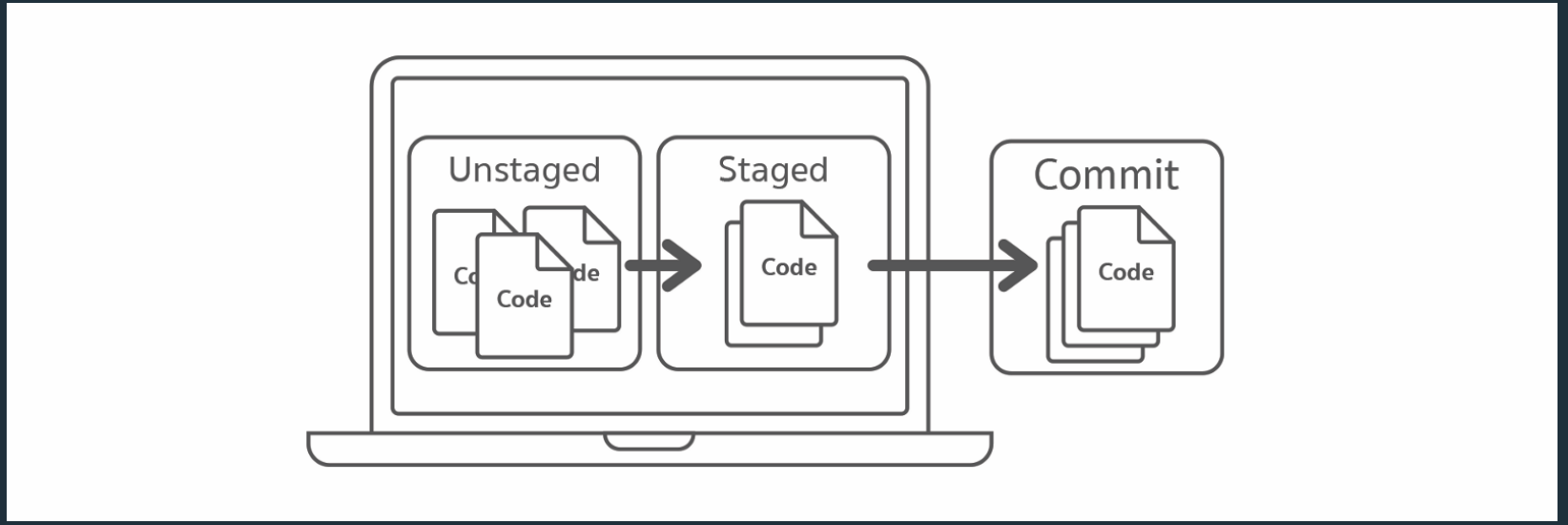
Commit



- การ Backup เก็บไว้ใน VCS จะเรียกกันว่า *Commit*
- การ Commit จะสามารถเลือกได้ว่าจะเอาไฟล์ไหนบ้าง (ไม่จำเป็นต้องเลือกทุกไฟล์)
- แต่ละ Commit จะไม่มีไฟล์ข้อมูลฉบับเต็ม
- แต่ละครั้งที่ทำการ Commit จะจำแค่ว่ามีตรงไหนของข้อมูลที่ถูกเปลี่ยนแปลงไป
- สามารถย้อนดู History ได้ว่ามีการแก้ไขอะไรบ้าง ทำให้รู้ว่าใน Commit นั้นๆ แต่ละไฟล์มีข้อมูลเป็นอย่างไร

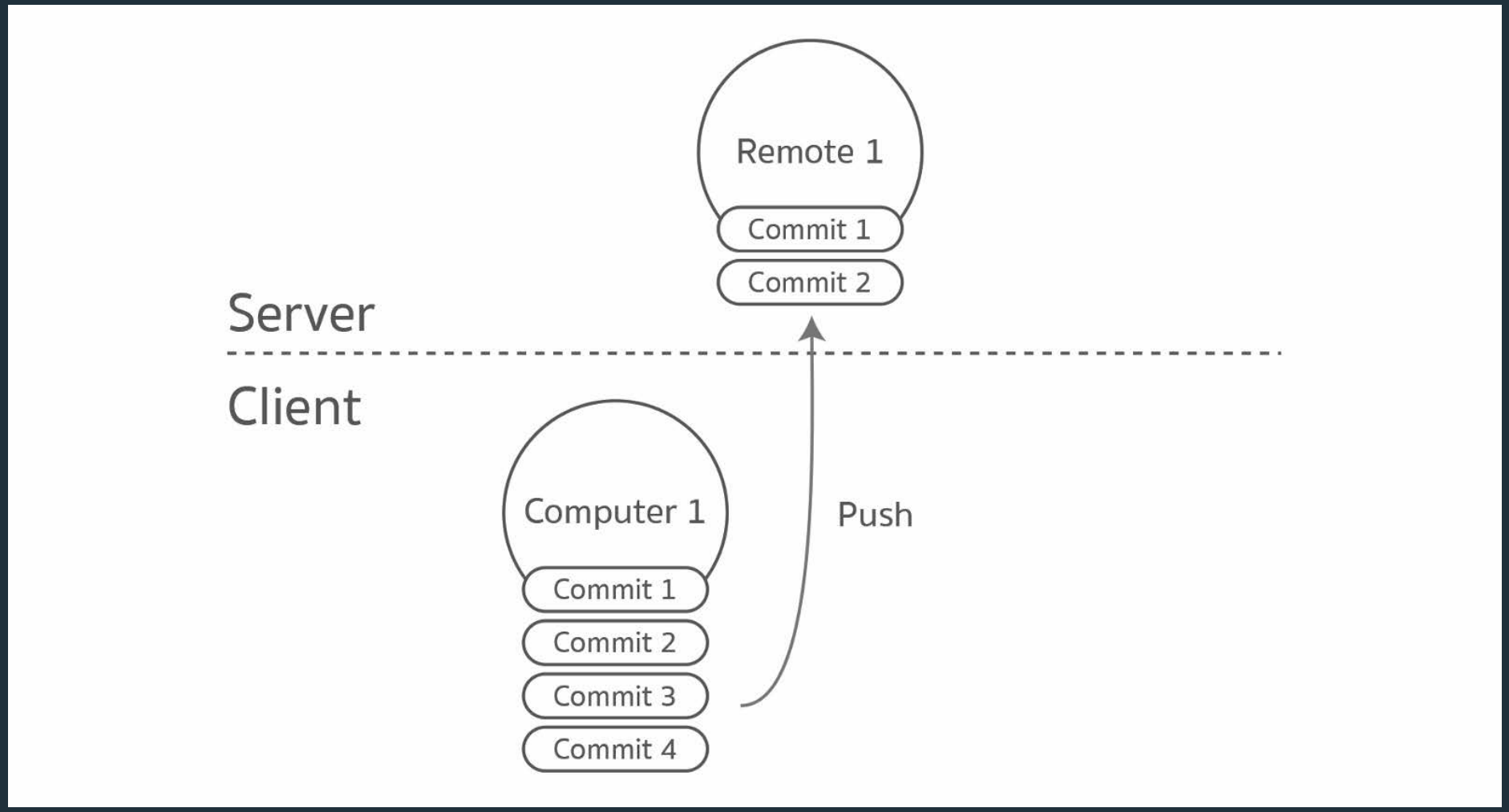
Unstaged & Staged

- ไฟล์ที่ถูกแก้ไขจะอยู่ในสถานะ *Unstaged*
- จะต้องเลือกไฟล์ที่ต้องการเพื่อย้ายเข้าสู่ในสถานะ *Staged* ก่อน ถึงจะทำการ Commit ได้
- Unstaged และ Staged ทำให้เราสามารถเลือกเฉพาะบางไฟล์ สำหรับ Commit ได้



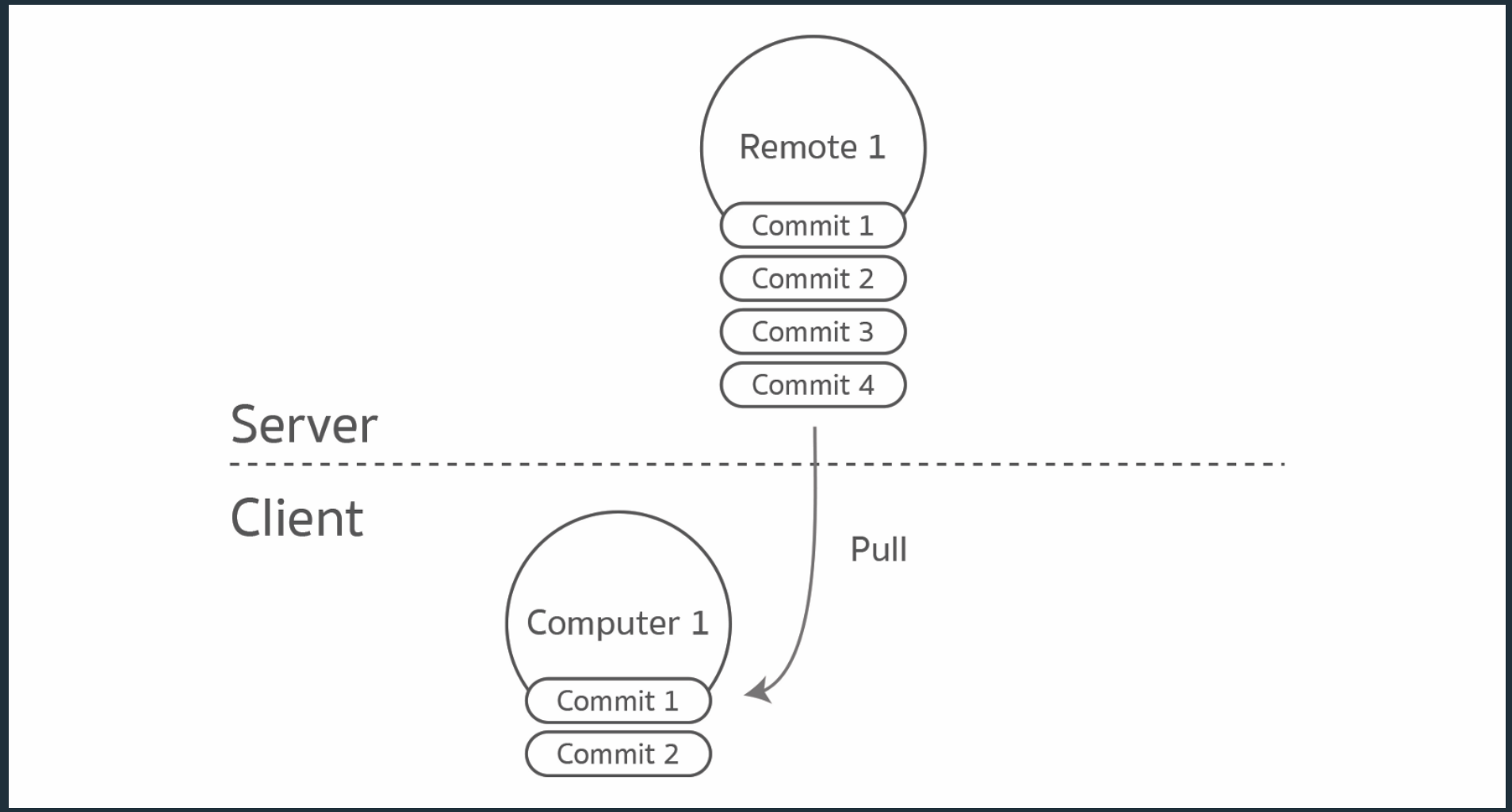
Push

- เวลาที่มี Commit อยู่ในเครื่องและต้องการจะ Sync ขึ้นไปเก็บไว้ใน Remote จะเรียกขั้นตอนนี้ว่า *Push*



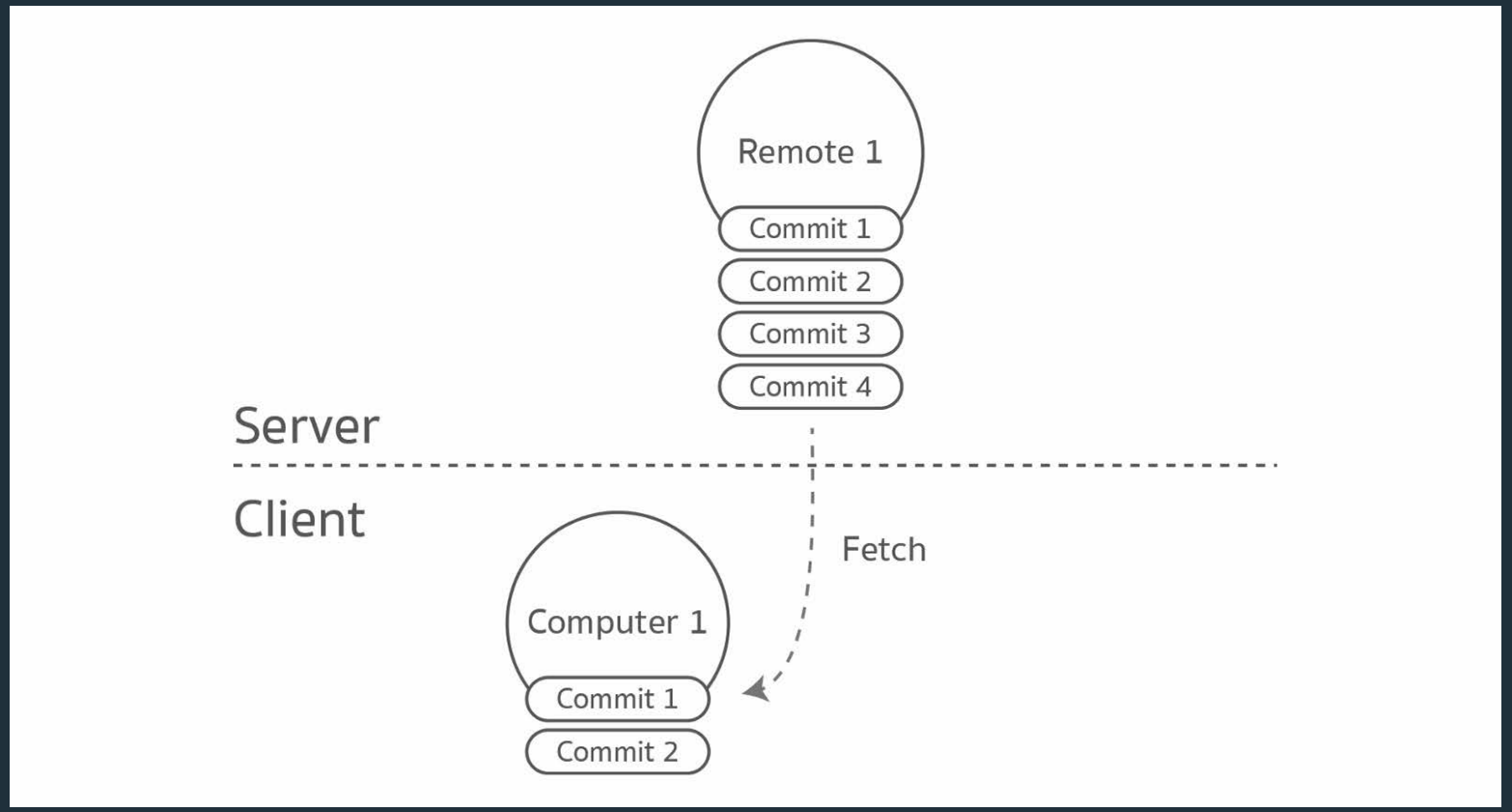
Pull

- เวลา Sync จาก Remote เพื่อดึงข้อมูล Commit ใหม่ๆลงมาเก็บไว้ในเครื่องจะเรียกขั้นตอนนี้ว่า *Pull*

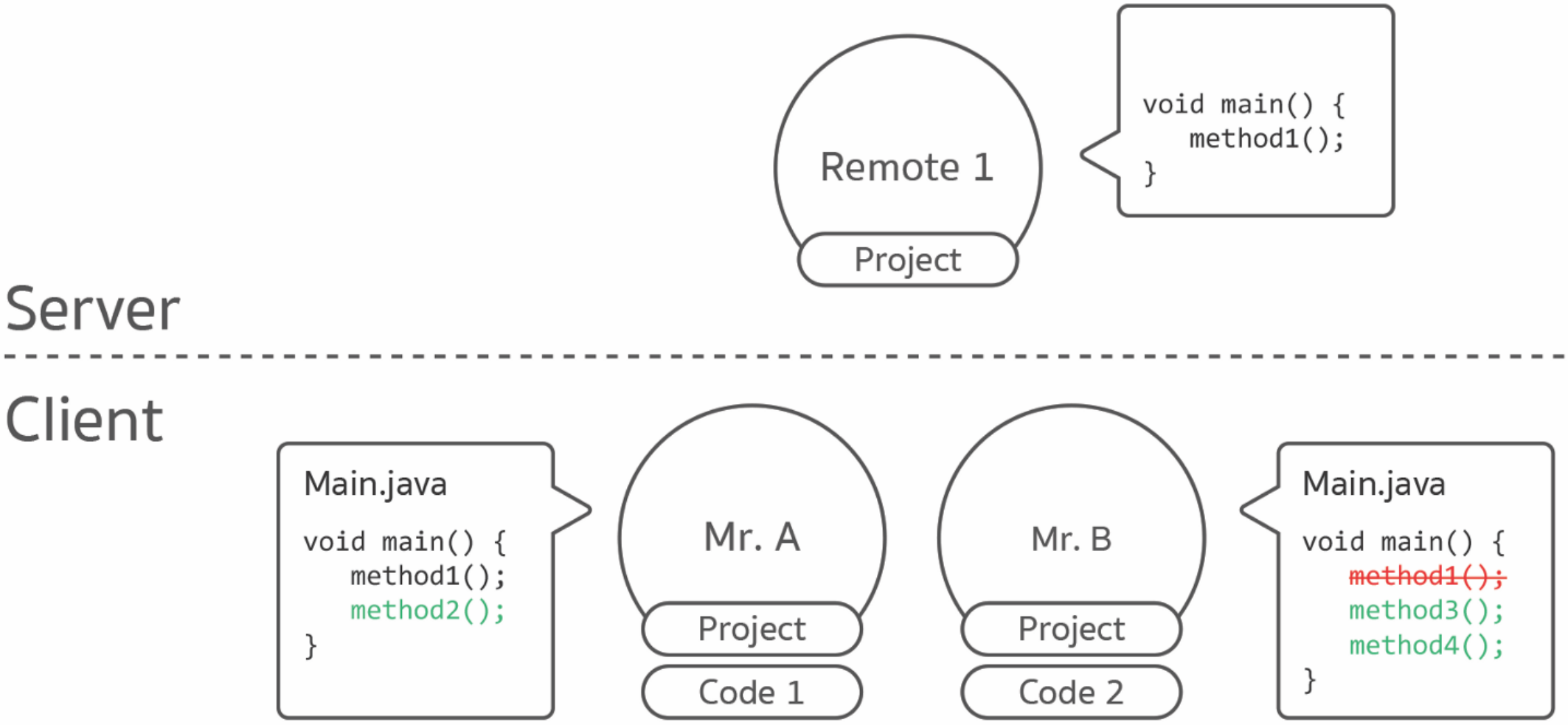


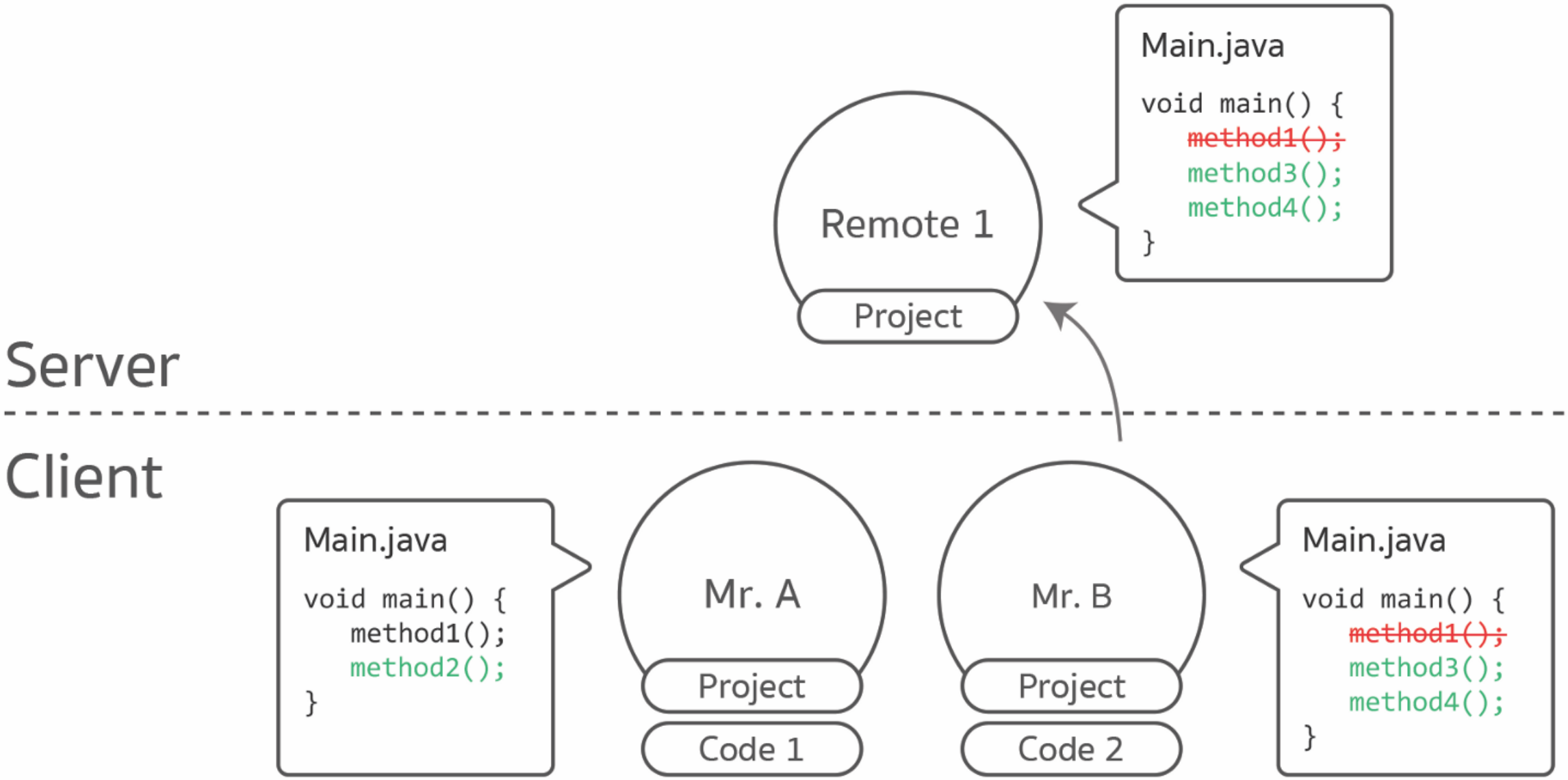
Fetch

- บางครั้ง อาจจะไม่ต้องการ Pull ข้อมูลลงมาเก็บไว้ในเครื่องทันที แค่อยากเช็คสถานะของ Remote ว่ามีใคร Push ข้อมูลใหม่ขึ้นไปที่ Remote หรือไม่ เราเรียกวิธีนี้ว่า ***Fetch***
- การ Fetch ทำให้สามารถอัพเดทและดู History ทั้งหมดที่อยู่บน Remote ได้ โดยไม่ต้องดึงข้อมูลลงมา
- การ Fetch จะถูกเรียกทุกครั้งที่ทำการ Pull

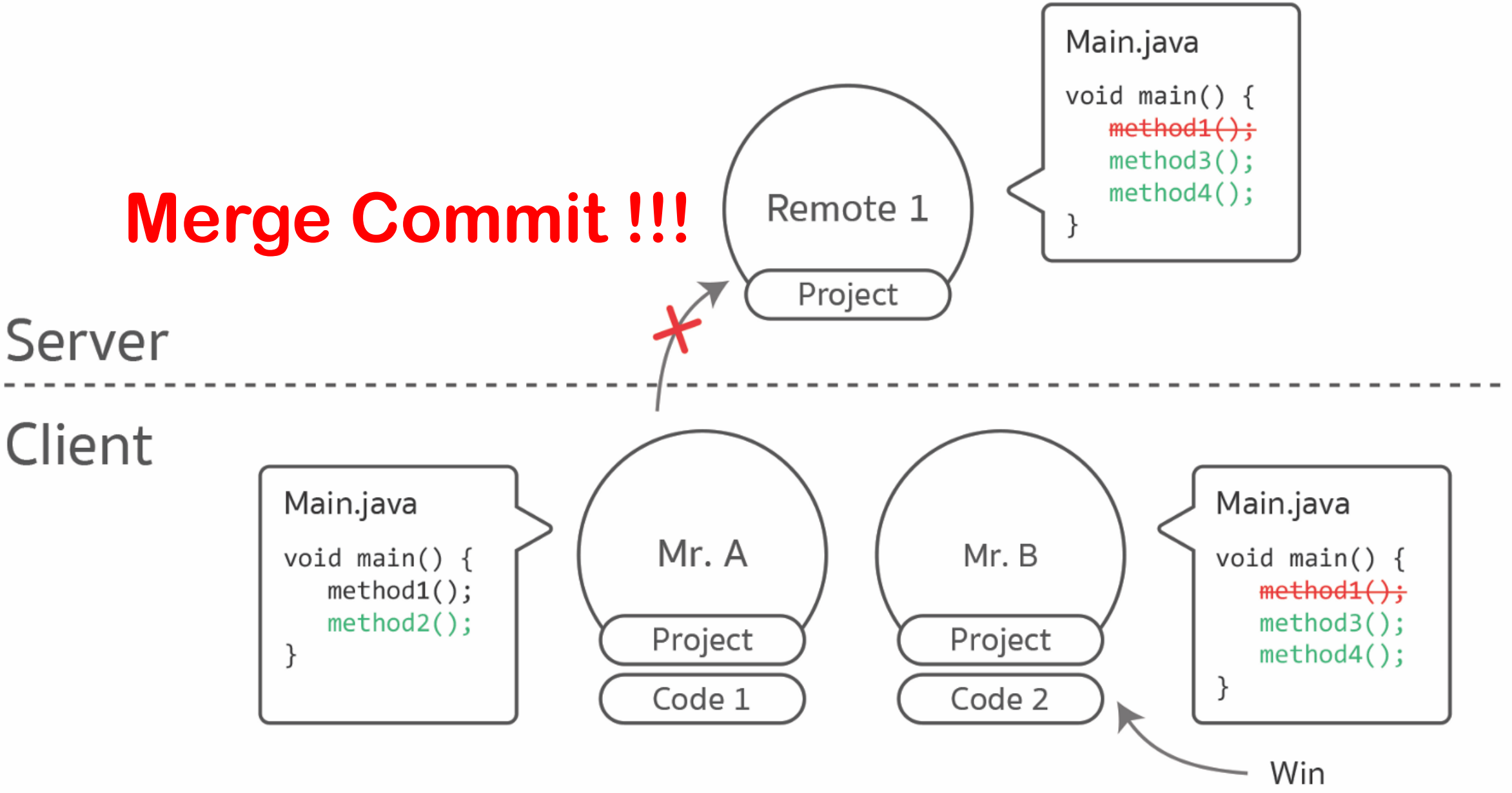


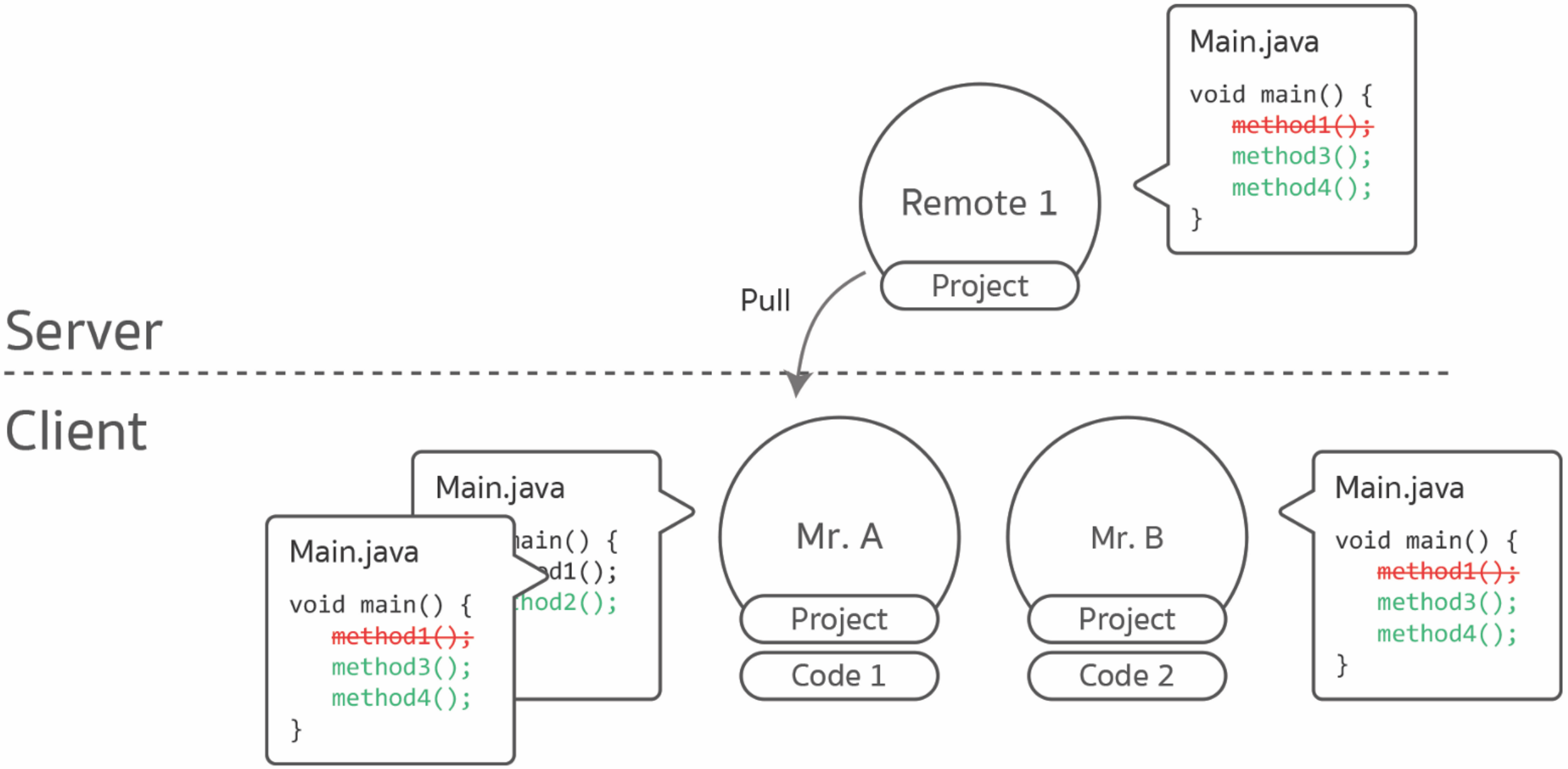
Merge Commit




Main.java
void main() {
method1();
method3();
method4();
}
Remote 1
Project
Server
Client
Main.java
void main() {
method1();
method2();
}
Mr. A
Project
Code 1
Mr. B
Project
Code 2
Main.java
void main() {
method1();
method3();
method4();
}

Merge Commit !!!

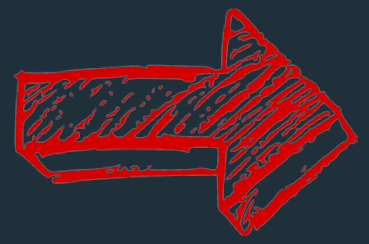



Server
Client
Remote 1
Project
Main.java
void main() {
method1();
method3();
method4();
}
Pull
Mr. A
Project
Code 1
Mr. B
Project
Code 2
Main.java
Main.java
void main() {
method1();
method3();
method4();
}
Main.java
void main() {
method1();
method3();
method4();
}

```
// โค้ดเดิม
void main() {
  method1();
}

// โค้ดที่นาย A แก้ไข
void main() {
    method2();
    method1();
}

// โค้ดที่นาย B แก้ไข
void main() {
    method1();
    method3();
}
```



```
void main() {
    method2();      // โค้ดของนาย A
    method1();
    method3();      // โค้ดของนาย B
}
```

- กรณีที่โค้ดสามารถ Merge รวมกันได้โดยไม่มีปัญหา เมื่อ Merge เสร็จ ก็จะกลายเป็น Commit ตัวหนึ่ง สำหรับ Push ขึ้น Remote

แต่ทว่าโค้ดที่นาย A กับนาย B
เขียนมันไม่ใช่แบบนั้นน่ะสิ !!!

Conflict

Main.java

```
void main() {
    method1();
    method3();
    method4();
}
```

B

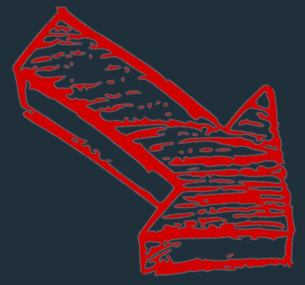
Main.java

```
void main() {
    method1();
    method2();
}
```

A

```
void main() {
<<<<<<< HEAD
    method1();
    method2();
=======
    method3();
    method4();
>>>>>>> 322d39a003e4d9...
}
```

- โค้ดที่อยู่ระหว่าง <<< และ === คือโค้ดของนาย A
- โค้ดที่อยู่ระหว่าง === และ >>> คือโค้ดของนาย B ส่วนตัวเลขต่อท้ายคือหมายเลขของ Commit ที่ทำการ Merge



```
void main() {
    method2();
    method3();
    method4();
}
```

Question ?

Homework ?