11

เพื่อที่จะลดปัญหา เก็บประวัติการแก้ไขต่าง ๆ ด้วยมือโดยการคัดลอกไฟล์ไปไว้ในไดเร็คทอรี่ใหม่ (อาจจะเป็นไดเร็คทอรี่ที่มีชื่อเป็นวันเดือนปีและเวลาก็ได้) และ การร่วมมือกันกับนักพัฒนาคนอื่น ๆ

12

centralized version control

file ทั้งหมดถูกเก็บอยู่ที่ central repo หรือศูนย์กลางที่เดียว ถ้าสมมติว่า central repo เกิดพังขึ้นมาหรือผู้ใช้ไม่สามารถติดต่อกับ central repo ได้ก็จะทำให้ไม่สามารถดึงไฟล์ที่พึ่งเปลี่ยนแปลงมาทีตัวเองได้

ส่วน distributed version control

ที่ใช้ใน BitKeeper และ git นั้น แต่ละคนจะมี copy ของไฟล์ตั้งแต่เริ่มแรกจนท้ายสุด ดังนั้นแต่ละทีมหรือแต่ละคนสามารถ maintain code ได้โดยที่ทุกคนมี copy ของไฟล์ทั้งหมดอยู่ที่ local computer และ change set ของแต่ละ version ที่เกิดขึ้นจากหลายๆ ทีมนั้นสามารถรวมกันได้ ไม่จำเป็นต้องรอ changes copy จาก central repo อีกต่อไป

13

สามารถ ดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้

ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

14

โดยการแตก branch แรกเพื่อเพิ่มความสามารถให้โปรแกรมรับ -v สำหรับบอกเวอร์ชั่นโปรแกรม

ตัวอย่าง ไฟล์ hello.py

$ git checkout -b show-version

Switched to a new branch 'show-version'

ที่ไฟล์ hello.py ส่วนที่เช็ค -h (แถวบรรทัดที่ 5) เพิ่ม code ให้เป็นอย่างนี้

if '-h' in names:

exit('usage: python hello.py [-h] [NAME [NAME ...]]')

elif '-v' in names:

exit('advance hello beta')

แล้วก็ commit

$ git commit -am 'add option -v as show version'

[show-version 02513a2] add option -v as show version

1 file changed, 2 insertions(+)

เรียร้อยกับ branch แรกไปแล้ว ก็มาทำ branch ที่สอง โดยเขียนให้โปรแกรมรับ -l เพื่อบอกลิขสิทธิ์โปรแกรม

$ git checkout master

Switched to branch 'master'

$ git checkout -b show-license

Switched to a new branch 'show-license'

แก้ไฟล์ hello.py ที่เดิมเลย (แค่เปลี่ยนหน้าตา code นิดหน่อย)

if '-h' in names:

exit('usage: python hello.py [-h] [NAME [NAME ...]]')

elif '-l' in names:

exit('license under WTFPL v2.0')

เช่นเดิม commit มันซะ

$ git commit -am 'add option -l as show license'

[show-license ec29013] add option -l as show license

1 file changed, 2 insertions(+)

คราวนี้กลับมาที่ master แล้วทำการ merge เหมือนตอนที่ผ่านมา

$ git checkout master

Switched to branch 'master'

$ git merge show-version show-license

Fast-forwarding to: show-version

Trying simple merge with show-license

Simple merge did not work, trying automatic merge.

Auto-merging hello.py

ERROR: content conflict in hello.py

fatal: merge program failed

Automatic merge failed; fix conflicts and then commit the result.

พบว่ามีฟ้อง conflict และส่งผลให้ merge ไม่ผ่าน ตอนนี้ถ้าดูสถานะจะพบว่า

$ git status

# On branch master

# Unmerged paths:

# (use "git add/rm <file>..." as appropriate to mark resolution)

#

# both modified: hello.py

#

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

หมายเหตุว่า Git จะไม่ยอมให้ย้าย branch ถ้าหากยังมี conflict เช่นนี้อยู่ ดังนั้นมาเก็บข้อผิดพลาดนี้กัน โดยเปิดไฟล์ hello.py ขึ้นมา จะเห็นดังนี้

import sys

names = sys.argv[1:]

if '-h' in names:

exit('usage: python hello.py [-h] [NAME [NAME ...]]')

<<<<<<< .merge\_file\_trt1cP

elif '-v' in names:

exit('advance hello beta')

=======

elif '-l' in names:

exit('license under WTFPL v2.0')

>>>>>>> .merge\_file\_XPZY4N

if not names:

print('Hello, world!')

if '-s' in names:

names.remove('-s')

names.sort()

for name in names:

print('Hi {}.'.format(name))

ส่วนที่เกิด conflict จะถูกคั่นด้วยบรรทัดที่ขึ้นต้นด้วยเครื่องหมาย <<<<<<<, =======, >>>>>>> ในตัวอย่างนี้ จัดการลบ 3 บรรทดนั้นทิ้ง

เรียบร้อยแล้วก็ลบ branch ทั้ง 2 ทิ้ง

15

Merge ให้น้อยลง

16

Git เป็น Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา

Github เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ Git

17

เพื่อให้งานดำเนินต่อไปได้โดยไม่ต้องรอแก้บั๊กใน master ให้เสร็จก่อน

18

ในการ Merge Branch บน Git นั้น หาก Commit สุดท้ายของ Branch ปลายทาง เป็น Commit เดียวกับจุดที่แยก Branch ออกมา การ Merge จะได้ผลเป็นแบบ Fast-forward

เพราะ

หากทำการ Merge Branch \* ไปยัง master จะเป็นการ Merge แบบ Fast-forward จะเห็นว่า Branch master ไปอยู่ที่ Commit เดียวกับ Branch test และสายของ Commit เป็นเส้นตรง!

19

git pull ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้ว

20

การทำงานของ git