พื้นฐานเกี่ยวกับ version control และ git

11. ประโยชน์ => Version Control System คือ ระบบที่คอยจัดการ backup source code ของเรา โดยเก็บเป็นลักษณะ version ต่างๆ เช่นในกรณีเกิดปัญหาขึ้น ก็สามารถหยิบเอา source code ตัวเก่าที่เคยใช้งานได้มาแทน พูดง่ายๆ ก็คล้ายๆ กับ Undo ที่เราทำกัน แต่ว่า Version Control ยังเป็นตัวกลางที่ทำให้ source code ของแต่ละเครื่อง (แต่ละ programmer) มี source code ที่ตรงกันด้วย

หากเครื่องคอมพ์หาย ก็ไม่มีปัญหาอะไร ข้อมูลทั้งหมดอยู่ใน Server

12. Distributed Version Control ถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ การ checkout แต่ละครั้งคือการทำสำรองข้อมูลทั้งหมดแบบเต็มๆ ส่วน Centralized Version Control มีจุดอ่อนตรงที่ทุกอย่างรวมกันอยู่ที่เซิร์ฟเวอร์ที่เดียว ถ้าเซิร์ฟเวอร์นั้นล่มซักชั่วโมงนึง หมายความว่าในชั่วโมงนั้นไม่มีใครสามารถทำงานร่วมกันหรือบันทึกการเปลี่ยนแปลงงานที่กำลังทำอยู่ไปที่เซิร์ฟเวอร์ได้เลย หรือถ้าฮาร์ดดิสก์ของเซิร์ฟเวอร์เกิดเสียขึ้นมาและไม่มีการสำรองข้อมูลเอาไว้ คุณก็จะสูญเสียข้อมูลประวัติและทุกอย่างที่มี จะเหลือก็แค่ก๊อปปี้ของงานบนเครื่องแต่ละเครื่องเท่านั้นเอง

13. Centralized Version Control มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลายๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้ คือ ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

14. วิธีแก้ไขก็คือ ทำการ edit คือ แค่ลบพวกโค๊ดส่วนเกินออก แล้วแก้ไขใหม่ให้เรียบร้อย จากนั้นลองเช็คสถานะ แล้ว commit ไปใหม่

15. แนวทางในการลด conflict

- ทำการ Merge บ่อย ๆ นั่นคือ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code นั่นเอง

จะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมาก

ถึงจะเกิดข้อขัดแย้ง ก็เป็นเพียงปัญหาเล็ก ๆ

* การออกแบบที่ดีมันช่วยให้ทีมทำงานร่วมกันได้อย่างดี ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้น ยิ่งแต่ละส่วนการทำงานเล็ก ๆ แล้วก็ยิ่งทำให้คุณภาพของการออกแบบระบบดี รวมทั้ง ลดความเสี่ยงจาก Merge conflict อีกด้วย
* การพูดคุย การสื่อสาร ต้องมีการพูดคุยสื่อสารกับคนในทีมให้เข้าใจว่าใครทำหน้าที่อะไรบ้าง
* Mob programming เป็นวิธีการที่ทรงประสิทธิภาพอย่างมากและเชื่อได้เลยว่า แก้ไขปัญหา Merge conflict ได้ 100% เนื่องจากทุกคนมานั่งทำงานด้วยกัน ใช้เครื่องทำงานเดียวกัน ดังนั้น ไม่มีทางที่ source code จะขัดแย้งกัน

16. Git เป็นสิ่งที่เรียกว่า Version Control หรือตัวที่เอาไว้ดูแลและควบคุมการเปลี่ยนแปลงของ Source Code (a.k.a. เวอร์ชั่น) ไม่ว่าจะเป็น Text หรือ Binary File (JPG, PNG) โดยสิ่งที่เราเอาไปฝากไว้บน Git เราจะเรียกว่า Repository และวิธีการทำงานจะมี Git Server รันอยู่บน Server เราสามารถเข้าถึง Git ได้ผ่าน URL ที่สร้างขึ้นมาตอนเราสร้าง Repository บน Server แตกต่างจาก GitHub เพราะ GitHub คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ Git โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ

17.การแตก branch จะเป็นการก๊อปปี้กลุ่มของไฟล์ที่มีอยู่ปัจจุบันไปแตกเป็นกิ่งก้านสาขาอันใหม่ ช่วยให้นักพัฒนาสามารถพัฒนาบน branch ใหม่ได้โดยไม่มีการเข้าไปยุ่งกับส่วนของระบบที่ทำงานได้ใน branch หลัก

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

19. git pull ใช้ดึงความเปลี่ยนแปลงจาก remote มายัง local และรวมเข้าด้วยกัน (มีค่าเท่ากับ fetch+merge)

20. Feature Branch Workflow