พื้นฐาน version control และ git

11.ตัวควบคุมเวอชั่น คือตัวที่จะคอยตรวจสอบว่าในแต่ละครั้งซอสร์สโค้ดมันมีอะไรเปลี่ยนแปลง ไปบ้าง แล้วเก็บความเปลี่ยนแปลงนั้นไว้ ตั้งเป็น version ตามด้วยตัวเลขที่เพิ่มขึ้นไปเรื่อยๆ เหมือนกับการที่เราเซฟโค้ดไว้เป็นก้อปปี้ๆนั่นแหละ แต่ version control มันมีระบบการทำ งานภายในที่เจ๋งกว่าแค่การก้อปปี้ข้อมูลไว้เป็นช่วงๆ มันยังบันทึกไว้ด้วยใครเป็นคนเขียน โค้ดบรรทัดไหน เวลาเมื่อไร

12.Distributed Version Control Systems (DVCSs) หรือระบบ VCS แบบกระจายศูนย์ ในระบบ แบบนี้ (เช่น Git, Mercurial, Bazaar หรือ Darcs) แต่ละคนไม่เพียงได้ก๊อปปี้ล่าสุดของไฟล์ เท่านั้น แต่ได้ทั้งก๊อปปี้ของ repository เลย หมายความว่าถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้ กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ การ checkout แต่ละครั้งคือการทำสำรอง ข้อมูลทั้งหมดแบบเต็ม ๆ

นอกจากนั้นระบบเหล่านี้ยังทำงานกับหลาย ๆ repository ได้อย่างดี ทำให้คุณ สามารถทำงานกับคนหลายกลุ่มซึ่งทำงานในรูปแบบต่างกันในโปรเจคเดียวกันได้อย่างง่าย ดาย เนื่องจากระบบเหล่านี้สนับสนุนการทำงานได้หลากหลายรูปแบบ ซึ่งอาจทำได้ยากใน ระบบแบบรวมศูนย์

13.Centralized Version Control Systems (CVCSs) หรือระบบ Version Control Systems แบบรวม ศูนย์ ระบบเหล่านี้ เช่น CVS, Subversion และ Perforce มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้

การทำงานแบบนี้มีประโยชน์เหนือ local VCS ใน หลายด้าน เช่น ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละ เครื่องเยอะ

14. 1.ทำการ merge บ่อย ๆ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code นั่นเองจะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมาก ถึงจะเกิดข้อขัดแย้ง ก็เป็นเพียงปัญหาเล็ก ๆ ซึ่งสามารถแก้ไขได้อย่างง่ายดาย

2.การออกแบบที่ดีมันช่วยให้ทีมทำงานร่วมกันได้อย่างดี ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้นหรือในแต่ละ class แต่ละ method ควรมีเหตุผลเดียวในการเปลี่ยนแปลงเท่านั้น

15. 1.ทำการ merge บ่อย ๆ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code

2.ช่ายออกแบบ code ร่ามกัน เพื่อลดความขัดแย้งในทีม

16.Git คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่น ต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือ ว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้าง ก็ยิ่งจะมีประสิทธิภาพมากหากเป็นการพัฒนาเ ป็นที

GitHub คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ Git โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ ในบทแทรกนี้ เรา จะมาดูวิธีการหลายๆอย่างที่คุณสามารถใช้ GitHub เพื่อทำตามที่หนังสือ Discover Meteor บอกได้

สรุปคือ git คือ Version Control ตัวหนึ่ง แต่ github คือ เว็ปไซต์ที่ให้บริการ

17. เป็น feature ที่ช่วยให้นักพัฒนาสามารถที่จะทำงานได้สะดวกขึ้น ยกตัวอย่างเช่น เรามีโค๊ดที่ดีอยู่แล้ว แต่อยากจะทดลองอะไรนิดๆหน่อย หรือแก้ไขอะไรก็ตาม ไม่ให้กระทบกับตัวงานหลัก ก็เพียงแค่สร้าง branch ใหม่ขึ้นมา เมื่อแก้ไขหรือทำอะไรเสร็จแล้ว ก็ค่อยเซฟกลับมาที่ master เหมือนเดิม

18.เป็นพฤติกรรมของการที่แตก Branch ออกไปทำงาน และในการ Merge เข้ามาที่ Branch ใดๆ นั้น Git คำนวณแล้วว่า ไม่มีการเกิด Conflict ระหว่างไฟล์ใดๆ ก็ตามในการ Merge จึงสามารถนำ commit ที่เกิดขึ้นใน Branch ที่แตกออกไปทำงาน มาต่อได้เลย

19.git pull ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วย git merge อัตโนมัติ

20.การแตกสายการพัฒนาจากต้นฉบับ Git จะใช้แนวคิดกระจายศูนย์การเก็บซอร์ส ทำให้ผู้ใช้หลายคนสามารถสร้างสรรค์ชิ้นงานพร้อมกันตามแนวทางที่ได้ตกลงกันไว้ Git ยังมีแนวคิดของการแตกกิ่งก้าน (branch) ที่เปรียบเสมือนการแตกสายการพัฒนาชิ้นงานชิ้นนั้น เพื่อทดลองนำแนวทางอื่นๆ มาใช้พัฒนาชิ้นงานโดยไม่ต้องสนใจแนวทางเดิม หรือดัดแปลงนำความสามารถนี้ไปกำหนดสายการพัฒนาก็ย่อมได้