พื้นฐานเกี่ยวกับ Version Control และ git

11.โปรแกรม version control มีประโยชน์อย่างไร

* ระบบที่คอยจัดการ backup source code ของเรา โดยเก็บเป็นลักษณะ version ต่างๆ
* ในกรณีเกิดปัญหาขึ้น ก็สามารถหยิบเอา source code ตัวเก่าที่เคยใช้งานได้มาแทนได้
* Version Control ยังเป็นตัวกลางที่ทำให้ source code ของแต่ละเครื่อง (แต่ละ programmer) มี source code ที่ตรงกัน ซึ่งโดยปกติทั่วไปแล้ว programmer มักจะทำโปรแกรมให้เสร็จสมบูรณ์ และใช้งานได้ก่อน จึงค่อยโยนขึ้นไปที่ repository เพื่อให้คนอื่นดึงไปใช้ต่อไป

12.ข้อได้เปรียบ ของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร

* เป็นระบบจัดการซอร์สแบบรวมศูนย์อาจนำเซิร์ฟเวอร์สำรอง (redundant server) ซึ่งมีข้อมูลเหมือนกันกัเซิร์ฟเวอร์หลักทุกประการ มาใช้งานยามฉุกเฉินเมื่อเซิร์ฟเวอร์หลักเสียหาย
* ผู้ใช้แต่ละคนจะคัดลอกคลังข้อมูลชิ้นงานทั้งหมดจากเซิร์ฟเวอร์มาเก็บไว้ในเครื่องตนเอง ทำให้สามารถเข้าถึงประวัติชิ้นงานได้ตลอดเวลาแม้จะขาดการเชื่อมต่อ
* ผู้ใช้ยังสามารถส่งข้อมูลให้กันเองโดยไม่ต้องผ่านเซิร์ฟเวอร์
* นอกจากจะทำให้แต่ละเครื่องเปรียบเสมือนที่สำรองไฟล์ชิ้นงานแล้ว การทำงานหลายๆ อย่างยังมีความเร็วสูงขึ้นอย่างมีนัยสำคัญ
* 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.ข้อได้เปรียบ ของ centralized version control เมื่อเทียบกับ distributed version control คืออะไร



* ระบบ Version Control Systems แบบรวมศูนย์ ระบบเหล่านี้ เช่น CVS, Subversion และ Perforce มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้ ระบบการทำงานแบบรวมศูนย์นี้ได้ถูกนำมาใช้เป็นเวลานานหลายปี
* การทำงานแบบนี้มีประโยชน์เหนือ local VCS ในหลายด้าน เช่น ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ
* แต่ระบบแบบนี้ก็มีจุดอ่อนเหมือนกัน ตรงที่การรวมศูนย์ทำให้มันเป็นจุดอ่อนจุดเดียวที่จะล่มได้เหมือนกันเพราะทุกอย่างรวมกันอยู่ที่เซิร์ฟเวอร์ที่เดียว ถ้าเซิร์ฟเวอร์นั้นล่มซักชั่วโมงนึง หมายความว่าในชั่วโมงนั้นไม่มีใครสามารถทำงานร่วมกันหรือบันทึกการเปลี่ยนแปลงงานที่กำลังทำอยู่ไปที่เซิร์ฟเวอร์ได้เลย หรือถ้าฮาร์ดดิสก์ของเซิร์ฟเวอร์เกิดเสียขึ้นมาและไม่มีการสำรองข้อมูลเอาไว้ คุณก็จะสูญเสียข้อมูลประวัติและทุกอย่างที่มี

14.บอกแนวททางการแก้ไข conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

* ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code ให้ทำการ merge บ่อย ๆ จะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมาก
* หนึ่งในแนวทางการออกแบบระบบงานที่ดี คือ [SOLID](http://www.somkiat.cc/testable-design-part-2-about-solid/)นั่นคือ ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้นหรือในแต่ละ class แต่ละ method ควรมีเหตุผลเดียวในการเปลี่ยนแปลงเท่านั้นผลที่ได้ก็คือ นักพัฒนาจะไม่ทำงาน หรือ เปลี่ยนแปลง source code ที่เดียวกันอย่างแน่นอนยกเว้นจะทำงานเดียวกัน หรือ ทำงานด้วยกัน
* [Mob programming](http://mobprogramming.org/) เป็นวิธีการที่ทรงประสิทธิภาพอย่างมากและเชื่อได้เลยว่า แก้ไขปัญหา Merge conflict ได้ 100% เนื่องจากทุกคนมานั่งทำงานด้วยกันใช้เครื่องทำงานเดียวกันดังนั้น ไม่มีทางที่ source code จะขัดแย้งกันแต่มันมีค่าใช้จ่ายที่สูงมาก ๆ

15.บอกแนวททางการลด conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

การพูดคุย การสื่อสารกันในทีม นั้นมีความสำคัญอย่างยิ่ง

* แต่ละคนในทีมพูดคุยกันหรือไม่ ?
* แต่ละคนในทีมรู้หรือไม่ว่า เพื่อน ๆ แต่ละคนทำงานอะไร ?
* รู้หรือไม่ว่า แต่ละคนแก้ไข class อะไรกันอยู่ ?
* รู้หรือไม่ว่า สิ่งที่แก้ไขไปนั้นจะไปกระทบใครบ้าง ?

ดังนั้น ถ้าไม่รู้หรือไม่แน่ใจ ก็ควรพูดคุยกันให้เข้าใจเพื่อจะได้ลดปัญหาที่เกิดขึ้นได้อย่างง่ายและตรงจุด โดยที่ต้องไม่ปทำงานซ้ำๆกัน เพียงเพราะไม่รู้มาก่อนว่า มีคนทำงานชิ้นนั้นไปก่อนแล้ว

16. Git คืออะไร แตกต่างจาก Github อย่างไร

**Git** คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็ค มีการ backup code สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็ค ที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้างก็สามารถทำได้ เหมาะอย่างยิ่งสำหรับนักพัฒนาไม่ว่าจะเป็นคนเดียวโดยเฉพาะอย่างยิ่งจะมีประสิทธิภาพมากหากเป็นการพัฒนาเป็นทีม

แตกต่างจาก Github เพราะ Github เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ของ Git อีกที ( มักนิยมใช้ในการเก็บโปรเจ็ค Open Source ต่างๆ ไม่ว่าจะเป็น Bootstrap, Rails, Node.js, Angular )

17.จุดประสงค์หลักในการ brance คืออะไร

เพื่อแยกตัวออกมาจาก main line ของการพัฒนาและทำงานต่อไปบนบนนั้นโดยไม่ไปยุ่งเกี่ยวกับ main line ในหลายๆ VCS การทำแบบนี้ค่อนข้างจะเปลือง ส่วนใหญ่จะเป็นการ copy ทั้ง directory ของ source code ซึ่งจะกินเวลานานมากบน project ใหญ่ๆ

18.Fast forward merge คืออะไร และทำการ push ไปที่ remote repo จึงควรจะต้อง merge แบบนี้

Fast forward merge คือ การ Merge ที่สายของ Commit เป็นเส้นตรงสวยงาม แต่นั่นทำให้เราดูไม่ออกว่า Branch test นั้น แยกออกและรวมกลับ ที่จุดไหน

19.หน้าที่หลักของคำสั่ง git pull คืออะไร

pull ซี่งจะดึงสิ่งใหม่ๆจาก origin ลงมา merge ทั้งบน clone repository และ working directory โดยทันที หากเป็นมี conflict จากการ merge ใน working directory เราต้อง resolve conflict นั้นๆก่อนจะ commit ได้ต่อไป นั่นคือ pull แท้จริงคือ fetch ต่อเนื่องด้วย merge ในท่าเดียวนั่นเอง

20.แผนภาพด้านล่างนี้ต้องการสือความหมายอะไร



การทำงานส่วน Gitflow Workflow จะกำหนดรูปแบบ การออกแบบการเปิดตัวโครงการ มีความซับซ้อนมากขึ้นกว่าเดิม Workflowเ หมาะสำหรับการจัดการโครงการหรือทำโปรเจคขนาดใหญ่

ขั้นตอนการทำงานนี้ไม่ได้มีการเพิ่มแนวความคิดใหม่ ๆ มีการกำหนดบทบาทและวิธีการที่เฉพาะเจาะจง ยังมีการโต้ตอบกันได้ การบำรุงรักษาดูแลรักษาและการบันทึกข้อมูล การเผยแพร่ข้อมูล จะเป็นสิทธิของแต่ละบุคคล ซึ่งเป็นประโยชน์จากการทดลองแยกการทำงานร่วมกัน เพื่อให้มีงานที่มีประสิทธิภาพมากขึ้น