11. โปรแกรม version control มีประโยชน์อย่างไร

- ช่วยให้สามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้
- ช่วยให้เปรียบเทียบการแก้ไขที่เกิดขึ้นในอดีต ดูว่าใครเป็นคนแก้ไขคนสุดท้ายที่อาจทำให้เกิดปัญหา แก้ไขเมื่อไร ฯลฯ
- ช่วยให้สามารถกู้คืนไฟล์ที่ลบหรือทำเสียโดยไม่ตั้งใจได้อย่างง่ายดาย

12. ข้อได้เปรียบของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร

Distributed Version Control Systems (DVCSs) หรือระบบ VCS แบบกระจายศูนย์ ในระบบแบบนี้ (เช่น Git, Mercurial, Bazaar หรือ Darcs) แต่ละคนไม่เพียงได้ copy ล่าสุดของไฟล์เท่านั้น แต่ได้ทั้ง copy ของ repository เลย หมายความว่าถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูก copy กลับไปที่เซิร์ฟเวอร์เพื่อกู้ข้อมูลกลับคืนก็ได้ และการ checkout แต่ละครั้งคือการทำสำรองข้อมูลทั้งหมดแบบเต็ม ๆ

13. ข้อได้เปรียบของ centralized version control เมื่อเทียบกับ distributed version control คืออะไร

Centralized Version Control Systems (CVCSs) หรือระบบ Version Control Systems แบบรวมศูนย์ ระบบเหล่านี้ เช่น CVS, Subversion และ Perforce มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้

การทำงานแบบนี้มีประโยชน์เหนือ local VCS ในหลายด้าน เช่น ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง และการจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

14. บอกแนวทางในการแก้ไข conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

- ผู้พัฒนาอาจจะทำงานในบริเวณเดียวกันของไฟล์ อัลกอริธึมในการ Merge ที่ SVN ใช้นั้น สามารถติดตามได้ว่า Source Code แต่ละบรรทัด ถูกย้ายไปยังตำแหน่งไหนบ้าง มีอะไรที่ต่างกันบ้าง แต่ว่าการเปลี่ยนแปลงที่เกิดขึ้นในบรรทัดเดียวกัน หรือการเปลี่ยนแปลงที่มากจนทำให้อัลกอริธึมสับสน ก็ไม่สามารถ Merge ได้ แต่ข้อดีของ SVN ก็คือ Source Code ที่อยู่ใน Repository จะไม่มีทางอยู่ในสภาพที่เสียหายโดยเด็ดขาด เพราะว่า ก่อนการ Commit Code เข้าไปยัง Repository ตัว SVN จะตรวจว่า มี

การเปลี่ยนแปลงที่ไฟล์เดียวกันหรือไม่ ถ้ามี ผู้พัฒนา จะต้องทำการ Update เพื่อดาวน์โหลด Source Code มาทำการ “ทดลอง Merge” ในเครื่องก่อน ถ้าเกิดว่า Merge ไม่ผ่าน ก็จะไม่สามารถ Commit ได้ แก้ไข โดยการ Merge Source Code ด้วยมือ เรียกอย่างเป็นทางการ ก็คือ Conflict Resolution หรือ การแก้ Conflict

- การแก้ code ในที่ๆ ซ้อนทับกัน จะเกิด conflict ซึ่งโปรแกรมไม่สามารถหยั่งรู้ได้ว่า เราต้องการ code ตอนสุดท้ายออกมาเป็นยังไงกันแน่ และเราต้องลงมือจัดการกับ conflict โดยการแตก branch แรกเพื่อเพิ่มความสามารถให้โปรแกรม

15. บอกแนวทางในการลด conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

- การ Merge ไม่สามารถการันตีเรื่อง Semantic/Logic ได้ แต่ทีนี้ ถึงแม้การ Merge จะสำเร็จ เราก็ไม่สามารถการันตีได้ว่า Source Code ที่ Merge สำเร็จแล้ว จะคอมไพล์ผ่าน และทำงานได้ถูกต้อง (ซึ่งก็เป็นปัญหาเดียวกัน กับ Source Control ในทุกรูปแบบ รวมไปถึง แบบที่ไม่มี Source Control ด้วย) ดังนั้น ทีมพัฒนาบางแห่ง จึงจะมี Server อีกตัว เรียกว่า Continuous Integration Server ที่จะคอยดาวน์โหลด Source Code จาก Repository มาเป็นระยะๆ เพื่อคอมไพล์ และรัน Test (บางที อาจจะทั้ง Unit และ Integrated) เพื่อให้มั่นใจว่า Source Code ยังอยู่ในสภาพดีเสมอ และจะได้ชี้นิ้วถูกว่าใครคือคนสุดท้าย ที่ Commit Code เข้าไป แล้วทำพัง

16. Git คืออะไร แตกต่างจาก Github อย่างไร

Git คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้างก็สามารถทำได้ ฉะนั้น Version Control ก็เหมาะอย่างยิ่งสำหรับนักพัฒนาไม่ว่าจะเป็นคนเดียวโดยเฉพาะอย่างยิ่งจะมีประสิทธิภาพมากหากเป็นการพัฒนาเป็นทีม

Git แตกต่างจาก Github คือ การำงานด้วย Git จะต้องพิมพ์ผ่าน command หรือ Terminal เท่านั้น Github จึงถูกพัฒนาขึ้นมาเพื่อทำให้การทำงานกับ Git สะดวกและง่ายดายมากขึ้น Github คือ application ทำงานร่วมกันกับ Git โดยมี interface ที่ทำใช้งานง่าย

17. จุดประสงค์หลักในการ branch คืออะไร

- แยกการพัฒนาความสามารถใหม่ๆ ออกมาจากส่วนหลัก โดยที่ git มี master branch มาให้อยู่แล้วหลังจากทำการสร้าง git repository ขึ้นมา
- การแยก branch ใหม่เพื่อให้เราเริ่มโค้ด features ใหม่ได้
- เพื่อแยกการพัฒนา จะต้องทำการ merge การเปลี่ยนแปลงเข้ามายัง master branch หลังจากที่พัฒนาเสร็จแล้วเสมอ
- สร้าง master branch เพื่อให้งานดำเนินต่อไปได้โดยไม่ต้องรอแก้บั๊กใน master ให้เสร็จก่อน
- เพื่อแตก branch ใหม่ออกมา

18. Fast forward merge คืออะไรและทำไมการ push ไปที่ remote repo จึงควรจะต้อง merge แบบนี้

Fast-forward ซึ่งหมายถึงหมายเลข commit ของ branch master ย้ายไปใช้ตัวเดียวกับของ hi-name เท่านั้นเอง ยังไม่ใช่การ merge ที่แท้จริงแต่อย่างใด

19. หน้าที่หลักของคำสั่ง git pull คืออะไร

git จะทำการ ดึงข้อมูล และ รวมข้อมูล การเปลี่ยนแปลงจาก remote repository ในเครื่อง

20. แผนภาพด้านล่างนี้ต้องการสื่อความหมายอะไร

workflow คือแนวทางที่จะทำให้การทำงานมันราบรื่นนั่นแหละ แต่มันมีแนวทางแนวทางหนึ่งที่เป็นที่นิยม เขาตั้งชื่อมันว่า gitflow เมื่อจะพัฒนาฟีเจอร์ใหม่ ให้แตก branch Feature มาจาก Develop ให้ทีมพัฒนาโค้ดกันใน branch เมื่อพัฒนาเสร็จแล้วให้ merge โค้ดเข้า develop