11.โปรแกรม version control มีประโยชน์อย่างไร

ตอบ จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่คุณสามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้ หนังสือเล่มนี้จะยกตัวอย่างจากไฟล์ที่เป็นซอร์สโค้ดของซอฟต์แวร์ แต่ขอให้เข้าใจว่าจริง ๆ แล้วคุณสามารถใช้ version control กับไฟล์ชนิดใดก็ได้

ถ้าคุณเป็นนักออกแบบกราฟฟิคหรือเว็บดีไซเนอร์และต้องการเก็บทุกเวอร์ชั่นของรูปภาพหรือเลย์เอาต์ (ซึ่งคุณน่าจะอยากเก็บอยู่) การใช้ Version Control System (VCS) เป็นสิ่งที่ชาญฉลาดมาก เพราะมันช่วยให้คุณสามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้ นอกจากนั้นระบบ VCS ยังจะช่วยให้คุณเปรียบเทียบการแก้ไขที่เกิดขึ้นในอดีต ดูว่าใครเป็นคนแก้ไขคนสุดท้ายที่อาจทำให้เกิดปัญหา แก้ไขเมื่อไร ฯลฯ และยังช่วยให้คุณสามารถกู้คืนไฟล์ที่คุณลบหรือทำเสียโดยไม่ตั้งใจได้อย่างง่ายดาย

12.ข้อได้เปรียบของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร

ตอบ

* ทั้ง SCCS, RCS, CVS, SVN ที่กล่าวในช่วงแรกทั้งหมดนั้นใช้ central code repository model คือหมายความง่ายๆ ว่า changes ของ 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
* ดังนั้น distributed version control จึงเร็วกว่า และผู้ใช้งานก็สามารถแก้ได้ทุกที่ โดยไม่จำเป็นจะต้องติดต่อกับ central repo ก็ได้
* นอกจากนี้เรายังสามารถ fork projects ได้ด้วย แปลง่ายๆ คือดึงโปรเจกต์มาอยู่ในชื่อเราเพื่อแก้ไขในแบบที่เราต้องการหรือใช้สำหรับแก้โค้ดและส่งกลับไปให้กับเจ้าของที่พัฒนาโค้ดก็ได้ การ fork ในที่นี้คือการยอมให้ outside developers นำโค้ดของผู้พัฒนาไปแก้และ pull request กลับไปให้ผู้พัฒนานั่นเอง

13.ข้อได้เปรียบของ centralized version control เมื่อเทียบกับ distributed version control คืออะไร

### ตอบ Centralized Version Control ระบบ Version Control ที่มี Server หลักเป็นศูนย์กลางในการเก็บข้อมูลของโปรเจคทั้งหมด Client แต่ละเครื่องจะเป็นแค่ Working Copy เท่านั้น วิธีทำงานคร่าวๆ เริ่มต้นเราก็สร้างโปรเจคขึ้นมาบน Server หลังจากนั้นแต่ละคนก็มาดาวน์โหลดลงเครื่องของตัวเอง เวลามีการเปลี่ยนแปลงอะไรก็อัพโหลดกลับขึ้นไปบน Server แล้วคนอื่นๆ ก็จะสามารถดาวน์โหลด Update นั้นมายังเครื่องของตัวเองได้ ข้อเสียคือถ้า Server เดี้ยงไป Project History ทั้งหมดก็จะหายไปหมด ยี่ห้อที่เป็นที่นิยมกันในประเภทนี้ได้แก่

* [CVS](http://savannah.nongnu.org/projects/cvs/)
* [SVN](https://subversion.apache.org/)
* [Perforce](http://www.perforce.com/)
* [Team Foundation Server](http://www.google.co.th/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCMQFjAA&url=http%3A%2F%2Fmsdn.microsoft.com%2Fen-us%2Fvstudio%2Fff637362.aspx&ei=lvhiVP_SJsfiuQTE4ILAAw&usg=AFQjCNHYGbQJJK6uruHJixZ7sZH4yU-jpw&bvm=bv.79189006,d.c2E)

**Distributed Version Control** ระบบ Version Control ที่ไม่มี Server หลักเป็นศูนย์กลาง แต่ละเครื่องของแต่ละคนก็จะเก็บข้อมูลของโปรเจคทั้งหมดไว้ หมายความว่าข้อมูล History ของโปรเจคก็จะอยู่ในทุกเครื่อง และก็มีการ Share Changes กัน คล้ายๆ ระบบ Peer-to-Peer ครับ แนวคิดคือเวลาข้อมูลของเครื่องหนึ่งมีการเปลี่ยนแปลงก็จะมีการติดต่อไปยังเครื่องๆ อื่นๆ เพื่ออัพเดทให้ทุกคนเหมือนกัน ข้อดีคือมันกระจายความเสี่ยงของ Project History จะหายได้ครับ เพราะพังไปเครื่องหนึ่งเครื่องอื่นๆ ก็ยังมีอยู่ไม่หายไปไหน

* [Git](http://git-scm.com/)
* [Mercurial](http://mercurial.selenic.com/)
* [Aegis](http://aegis.sourceforge.net/)
* [BitKeeper](http://www.bitkeeper.com/)

14.บอกแนวทางในการแก้ไข conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

15.บอกแนวทางในการลด conflict ที่เกิดขึ้นจากการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

16.Git คืออะไร แตกต่างจาก Github อย่างไร

ตอบ  **Git** คือ ส่วนนี้เป็นส่วนสำคัญที่คุณต้องพยายามทำความเข้าใจเพราะถ้าคุณเข้าใจว่า Git คืออะไรและทำงานอย่างไร คุณจะสามารถใช้งาน Git ได้อย่างมีประสิทธิภาพและง่ายดายมาก เวลาคุณเรียนรู้ Git ให้พยายามลืมสิ่งต่าง ๆ ที่คุณอาจจะรู้อยู่แล้วจาก VCS อื่น ๆ เช่น Subversion หรือ Perforce เพราะคุณอาจสับสนคอนเซ็ปต์จากเครื่องมือเหล่านั้นได้ เหตุผลก็เพราะ Git เก็บและมองข้อมูล่างจากระบบอื่น ๆ เป็นอย่างมากถึงแม้ว่าจะทำงานคล้ายกันก็ตาม

[**GitHub**](https://github.com/) คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ **[Git](http://git-scm.com/)** โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ ในบทแทรกนี้ เราจะมาดูวิธีการหลายๆอย่างที่คุณสามารถใช้ GitHub เพื่อทำตามที่หนังสือ Discover Meteor บอกได้

17.จุดประสงค์หลักในการ branch คืออะไร

ตอบ ก็เหมือนกับการก้อปปี้โค้ดที่อยู่ภายใน master ทั้งหมดไปเป็นอีกโฟลเดอร์หนึ่งแล้วตั้งชื่อใหม่ ทุกคนที่ร่วมกันทำงานก็ให้ตกลงกันว่าจะ commit, push โค้ดลงไปที่นี่ ทดสอบโค้ดกันที่นี่ พอหลังจากพัฒนาโค้ดจนดี ไม่มีบัก อยากจะขึ้นโปรดักชั่นแล้ว ค่อย merge เขาเรียกว่าการ merge โค้ดกลับเข้าไปยัง master

ซึ่งตัว git มันจะเช็กของมันเองว่ามีโค้ดส่วนไหนเปลี่ยนแปลงไปบ้าง มันจะทำการผสานโค้ดของมันเอง เราไม่ต้องไปยุ่งเกี่ยวอะไรกับมันเลย แล้ว git เองมันก็ฉลาดถึงขนาดที่ว่า มันรู้ว่า ลบบรรทัดไหน เพิ่มบรรทัดไหนแจ่ม

18.Fast forword path คืออะไร และทำไมการ push ไปที่ remote repo จึงควรจะต้อง merge แบบนี้

ตอบ commit ที่ถูกชี้โดย branch ที่คุณ merge มันเป็น upstream ของ commit ที่คุณอยู่โดยตรง Git ก็เลยขยับ pointer ไปข้างหน้า พูดอีกนัยหนึ่งก็คือ เวลาที่คุณพยายามจะ merge commit ซักอันเข้ากับ commit ที่สามารถไปถึงได้โดยการตาม history ของ commit อันแรก Git จะทำให้ทุกอย่างง่ายขึ้นโดยการขยับ pointer ไปข้างหน้าเพราะมันไม่มีงานที่ถูกแยกออกไปให้ merge สิ่งนี้เรียกว่า "fast forward".

19.หน้าที่หลักของคำสั่ง git pull คืออะไร

ตอบ ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วยgit merge อัตโนมัตินั่นเอง

20.แผนการด้านล่างนี้ต้องการสื่อความหมายอะไร

ตอบ SourceTree