11.ตอบ

**-Source ไม่มีวันหาย สามารถย้อนคืนสิ่งที่ผิดได้ตลอดเวลา** การทำงานกับโปรเจคใหญ่ๆ Source Code เยอะๆ แล้วทำงานร่วมกันหลายคน มีโอกาสอยู่เสมอที่จะมีบางไฟล์เสียหายหรือถูกลบทิ้งอย่างไม่ได้ตั้งใจ แต่กับ Git แล้ว ถ้าคุณเอาข้อมูลใส่ไว้ใน Git Repository ถึงคุณจะลบไฟล์ทิ้งไปหมด มันก็สามารถเอากลับมาได้

-**Merge Code ให้อัตโนมัติ**

**-Track ได้ว่าใครทำอะไร**

**-ทำ Local ได้**

**-"มันกระจายความเสี่ยง"** โดยโครงสร้างของ Git จะเก็บทุกการเปลี่ยนแปลงไว้ใน Repository ตอนเรา clone มาลงในเครื่อง เราก็จะเก็บทั้งโปรเจคแบบ Full มาลงไว้เลย ดังนั้นทุกเครื่องที่มี Repository อยุ่ ก็คือ Backup ดีๆนี่เอง ดังนั้นหากเครื่องคอมพ์หาย ก็ไม่มีปัญหาอะไร ข้อมูลทั้งหมดอยู่ใน Server ถ้า Server พัง ก็ไม่มีปัญหาอะไร Source Code สมบูรณ์แบบก็ยังอยู่ในเครื่องเรา ตั้ง Server ใหม่แล้วเอาขึ้นใช้งานต่อได้เลย

12.ตอบ

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.ตอบ

มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้ ระบบการทำงานแบบรวมศูนย์นี้ได้ถูกนำมาใช้เป็นเวลานานหลายปี ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

14.ตอบ

ทำการ edit แล้ว commit ไปใหม่

15.ตอบ

ทำการ Merge บ่อย ๆ ปัญหาใหญ่ ๆ ของ Merge conflict เกิดจากจำนวน source code ที่ชนหรือขัดแย้งกันมาก ดังนั้น ถ้าต้องการหลีกเลี่ยงปัญหาเหล่านี้

ให้ทำการ merge บ่อย ๆ ไปเลย นั่นคือ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code นั่นเองจะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมาก

16.ตอบ

Git คือ revision control แบบ distributed (หมายความว่าไม่มีศูนย์กลาง) และ แบบ non-linear history (หมายความว่ามีประวัติการเปลี่ยนแปลงแบบไม่ใช่เส้นตรง)

แตกต่างจาก Github คือ การทำงานพื้นฐานของ Git ก็คือการ คอมมิท คุณสามารถมองได้ว่า คอมมิทก็คือภาพถ่ายโค้ดของโปรแกรมคุณในขณะหนึ่ง

แทนที่เราจะให้คุณเห็นโค้ดสำเร็จของแอพ Microscope ในครั้งเดียว เราก็ถ่ายภาพโค้ดของโปรแกรมไว้ทีละขั้นจนครบทุกขั้น ซึ่งคุณสามารถเข้าไปดูออนไลน์ได้ที่ GitHub

17.ตอบ

เพื่อให้เราเริ่มโค้ด features ใหม่ได้ ส่วน app ที่ stable ก็จะอยู่ใน master branch เพื่อให้งานดำเนินต่อไปได้โดยไม่ต้องรอแก้บั๊กใน master ให้เสร็จก่อน ใน Git เราใช้คำสั่ง branch เพื่อแตก branch ใหม่ออกมา Branch ใหม่ที่แยกออกมาก็จะมี history เป็นของมันเอง เราสามารถ checkout และ commit สิ่งใหม่ๆใน branch นี้ได้ นั่นคือ ก็เหมือนใน local repository หนี่งๆ เราสามารถมี sub repository ย่อยๆนั่นเอง เพียงแต่เราเรียก sub repository พวกนี้ว่า branch พอถีงเวลาที่เราเสร็จการแก้ master branch แล้ว เราสามารถใช้คำสั่ง merge เพื่อรวมงานของ master และ new features branch เข้าด้วยกัน

18.ตอบ

Fast forword merge คือ 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"

> Fast forward เกิดขึ้นเวลาเราใช้คำสั่ง Merge แล้ว Git พบว่ามันมีประวัติ

มาจากที่เดียวกัน มันก็จะพยายามทำให้เป็นเส้นตรงแนวเดียวกัน

19.ตอบ

สามารถดึงการเปลี่ยนแปลงล่าสุดจาก remote repository มายัง local repository ด้วยคำสั่ง git pull โดย git จะทำการ ดึงข้อมูล และ รวมข้อมูล การเปลี่ยนแปลงจาก remote repository ในเครื่องของคุณ

20.ตอบ