**answer**

1. ใช้ในการควบคุมเวอร์ชั่นของเอกสารที่ใช้ร่วมกัน ด้วยความเข้าใจของคนส่วนใหญ่ คิดว่า Version Control เอาไว้เก็บซอร์สโค้ดเท่านั้น แต่จริงแล้ว เราสามารถเก็บเอกสาร, รูปภาพ หรือคลิปวีดีโอได้อีกด้วย
2. Distributed Version Control มันกระจายความเสี่ยงของ Project History จะหายได้ครับ เพราะพังไปเครื่องหนึ่งเครื่องอื่นๆ ก็ยังมีอยู่ไม่หายไปไหน
3. Centralized Version Control ทุกๆคนในทีมจะรู้ว่าคนอื่นๆในทีมที่เหลือกำลังทำอะไรอยู่ ส่วนผู้ที่ดูแลระบบก็สามารถจัดการกับสิทธิ์การทำงานของ user ทุกคนได้ซึ่งนี้ก็เป็นข้อดีที่เหนือกว่า
4. - ทำการ Merge บ่อย ๆ

* ลองกลับไปดู source code ของคุณดูสิว่า มีการแก้ไข source code อย่างไรกันบ้าง เช่น มี class ไหนที่ต้องแก้ไขบ่อยมาก ๆ จากทุก ๆ คน ทุก ๆ feature
* แก้ไขการ Merge conflict ได้ง่าย ๆ มากเลยนะ เพียงแค่เราคุยกันในเรื่องที่ควรจะคุยกันมากขึ้น
* วันนี้คุณทำการ merge source code กันบ่อย ๆ หรือไม่ ? ถ้าไม่ ก็เริ่มได้แล้ววันนี้

1. - Short-Living Branches

* Small Modules
* Strong Communication
* Mob Programming

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

**Github** เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ Git (ทั่วโลกมักนิยมใช้ในการเก็บโปรเจ็ค Open Source ต่างๆ ที่ดังๆ ไม่ว่าจะเป็น Bootstrap, Rails, Node.js, Angular เป็นต้น)

1. ช่วยให้นักพัฒนาสามารถที่จะทำงานได้สะดวกขึ้น ยกตัวอย่างเช่น เรามีโค๊ดที่ดีอยู่แล้ว แต่อยากจะทดลองอะไรนิดๆหน่อย หรือแก้ไขอะไรก็ตาม ไม่ให้กระทบกับตัวงานหลัก ก็เพียงแค่สร้าง branch ใหม่ขึ้นมา เมื่อแก้ไขหรือทำอะไรเสร็จแล้ว ก็ค่อยเซฟกลับมาที่ master เหมือนเดิม
2. เพราะ 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".
3. คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วย git merge อัตโนมัตินั่นเอง
4. Release Branches