11.โปรแกรม version control มีประโยชน์อย่างไร

- Version control คือ ระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่คุณสามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้ หนังสือเล่มนี้จะยกตัวอย่างจากไฟล์ที่เป็นซอร์สโค้ดของซอฟต์แวร์ แต่ขอให้เข้าใจว่าจริง ๆ แล้วคุณสามารถใช้ version control กับไฟล์ชนิดใดก็ได้

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

12.ข้อได้เปรียบของ distributed version control เมื่อเทียบกับ centralized version control คืออะไร

Performing actions other than pushing and pulling changesets is extremely fast because the tool only needs to access the hard drive, not a remote server.

Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.

Everything but pushing and pulling can be done without an internet connection. So you can work on a plane, and you won’t be forced to commit several bugfixes as one big changeset.

Since each programmer has a full copy of the project repository, they can share changes with one or two other people at a time if they want to get some feedback before showing the changes to everyone.

13.ข้อได้เปรียบของ centralized version control เมื่อเทียบกับ distributed version control คืออะไร

If your project contains many large, binary files that cannot be easily compressed, the space needed to store all versions of these files can accumulate quickly.

If your project has a very long history (50,000 changesets or more), downloading the entire history can take an impractical amount of time and disk space.

14.บอกแนวทางในการแก้ไข conflict ที่เกิดขึ้นเมี่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

1.การแก้ Conflict ทำได้โดยการคลิ๊กขวาที่ไฟล์ แล้วเลือก Edit Conflict

2.เราก็จะพบหน้าต่างแบบนี้ครับ เรียกว่า TortoiseMerge

3.สิ่งที่เราต้องทำก็คือ กด Ctrl+ลูกศรขึ้น หรือ ลง เพื่อไล่ดูจุดที่มัน Conflict กันทั้งหมด ซึ่งบางครั้ง อาจจะไม่ได้เป็นแค่บรรทัดเดียว แต่เป็นทั้งบล็อค หลายๆ บรรทัด แล้วจัดการแก้ไขครับ โดยการดูจากหน้าต่างซ้ายมือ (ไฟล์นี้ ใน SVN) กับทางขวามือ (ไฟล์ในเครื่องเรา) แล้วเลือกว่า จะให้ผลลัพธ์การ Merge มาจากด้านไหน โดยการกด Use “Theirs” หรือ Use “Mine” บนทูลบาร์

4.แล้วตรงที่ Conflict ในหน้าต่าง Merged ก็จะกลายเป็นสีเขียวครับ

หรือถ้าไม่สามารถเลือกข้างได้ จะพิมพ์ลงไปเลยก็ได้เหมือนกันครับ เมื่อเรียบร้อยแล้ว กด Save แล้วทำการ Flag บอก SVN ว่า ไฟล์นี้แก้ไขเรียบร้อยแล้ว

15.บอกแนวทางในการลด conflict ที่เกิดขึ้นเมี่อมีการ merge โปรแกรมของผู้พัฒนาหลายๆคนเข้าด้วยกัน

- ใช้ VSTS (Visual Studio Team Services) เพื่อลดความสับสนของผู้ใช้งาน

16. git คืออะไร แตกต่างจาก github อย่างไร

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

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

- การทำงานพื้นฐานของ Git ก็คือการ คอมมิท คุณสามารถมองได้ว่า คอมมิทก็คือภาพถ่ายโค้ดของโปรแกรมคุณในขณะหนึ่ง

แทนที่เราจะให้คุณเห็นโค้ดสำเร็จของแอพ Microscope ในครั้งเดียว เราก็ถ่ายภาพโค้ดของโปรแกรมไว้ทีละขั้นจนครบทุกขั้น ซึ่งคุณสามารถเข้าไปดูออนไลน์ได้ที่ GitHub

17.จุดประสงค์หลักในการ branch คืออะไร

- คือการแยก branch ใหม่เพื่อให้เราเริ่มโค้ด 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

18.fast forward merge คืออะไรและทำไมการ push ไปที่ remote repo ถึงควรจะต้อง merge แบบนี้

fast forward merge คืออะไร

- ถ้า Head ของเราอยู่ใน commits ที่ดึงมา, case นี้มักเกิดจากคำสั่ง "git pull" เพื่อดึง code จากต้นน้ำมา update code(ที่ไม่มีการเปลี่ยนแปลง) ของเรา, สิ่งที่เกิดขึ้นก็คือ git จะ update HEAD ของเราให้ตรงตาม HEAD ของ merged commit (โดยไม่มีการสร้าง commit object ใหม่ขึ้นมา) มีศัพท์เฉพาะสำหรับกรณีนี้ว่า "Fast-forward”

ทำไมการ push ไปที่ remote repo ถึงควรจะต้อง merge แบบนี้

- ช่วยลดปัญหาที่อาจจะเกิดขึ้นจากการเลือกใช้การพัฒนาในลักษณะ Centralize Workflow ตามที่ Chujai ใช้ในการ push ข้อมูล

19.หน้าที่หลักของคำสั่ง git pull คืออะไร

- คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วย git merge อัตโนมัตินั่นเอง

20.