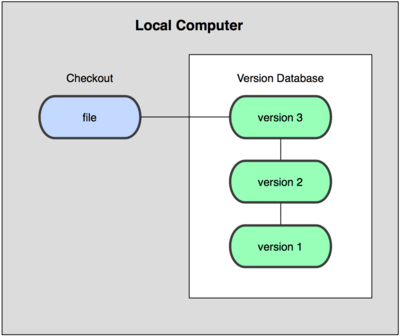
**ข้อ 11 โปรแกรม version control มีประโยชน์อย่างไร**

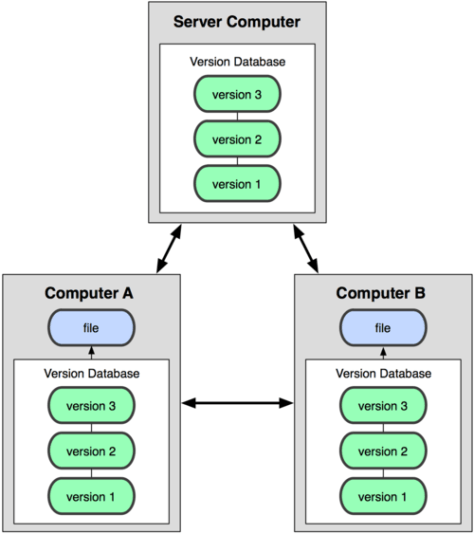
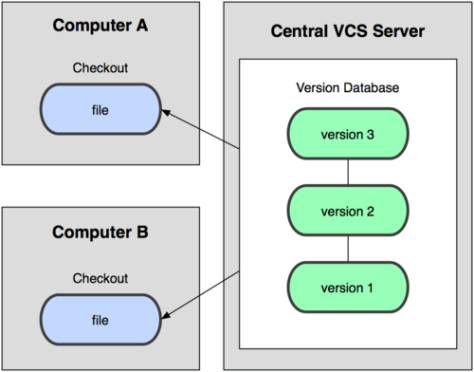
Version control คือ ระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่คุณสามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้ หนังสือเล่มนี้จะยกตัวอย่างจากไฟล์ที่เป็นซอร์สโค้ดของซอฟต์แวร์ แต่ขอให้เข้าใจว่าจริง ๆ แล้วคุณสามารถใช้ version control กับไฟล์ชนิดใดก็ได้

**ประโยชน์ของVersion Control** -จะช่วยให้คุณสามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้และใช้ในการควบคุมเวอร์ชั่นของเอกสารที่ใช้งานสามารถเก็บเอกสาร, รูปภาพ หรือคลิปวีดีโอได้อีกด้วย

Version Control System จะแบ่งออกเป็น 3 รูปแบบ คือ

**Local**

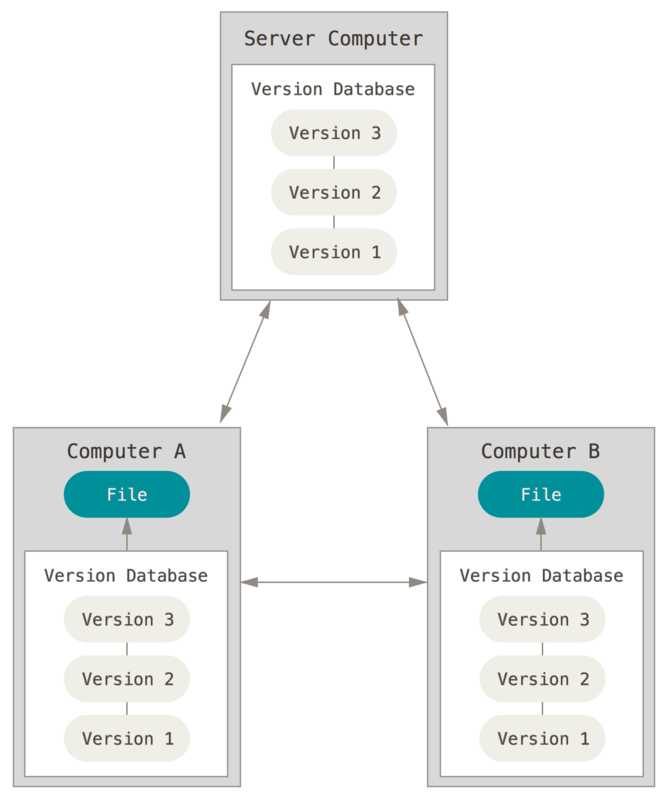


 **Central Distributed**

**ข้อที่ 12 ข้อได้เปรียบของ distributed Version Control เมื่อเทียบกับ centralized Version Control คืออะไร ?**

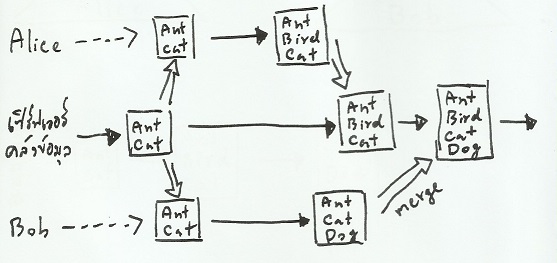
**ข้อได้เปรียบ**

* 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



**ข้อที่ 13 ข้อได้เปรียบของ centralized Version Control เมื่อเทียบกับ distributed Version Control คืออะไร ?**

**ยุคระบบจัดการซอร์สแบบรวมศูนย์ (Centralized Version Control System)**

คือระบบ 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/) -[Perforce](http://www.perforce.com/) -[SVN](https://subversion.apache.org/) -[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)

จะทำหน้าที่เก็บฐานข้อมูลชิ้นงานแล้ว ยังมีหน้าที่ในการจัดเก็บรุ่นใหม่ๆ ลงฐานข้อมูล และเรียกคืนรุ่นเก่าออกมาเมื่อถูกร้องขออีกด้วยผู้ใช้แต่ละคนที่ทำงานบนระบบดังกล่าว จะส่งคำสั่งเรียกคืนรุ่นที่ต้องการไปยังคลังข้อมูลกลาง เพื่อเรียกคืนชิ้นงานรุ่นนั้นออกมาทำงานต่อยอด เมื่อทำงานสำเร็จและสั่งจัดเก็บชิ้นงาน คลังข้อมูลจะตรวจเช็กเนื้อหาส่วนที่เปลี่ยนแปลงเทียบกับรุ่นที่ผู้ใช้คนนั้นๆ เรียกคืนออกไป แล้วบันทึกส่วนต่างลงฐานข้อมูลด้วยวิธีการนี้ เมื่อมีผู้ใช้หลายคนจัดเก็บชิ้นงานพร้อมกัน คลังข้อมูลจะพยายามรวมเนื้อหา (merge) ของผู้ใช้ทั้งหลายเข้าด้วยกัน หากเนื้อหาที่นำมารวมกันนั้นอยู่ในรูปแบบที่เรียบง่ายมากพอ ระบบจะรวมเนื้อหาดังกล่าวให้โดยอัตโนมัติ แต่หากเนื้อหามีความซับซ้อนคาบเกี่ยวกันจนระบบไม่สามารถรวมเนื้อหาให้เองได้

**ข้อได้เปรียบ -**มี Server หลักเป็นศูนย์กลางในการเก็บข้อมูลของโปรเจคทั้งหมด

**-**สามารถดาวน์โหลด Update ไฟล์นั้นมายังเครื่องของตัวเองได้

-มีการจัดเก็บรุ่นใหม่ๆ ลงฐานข้อมูล

-คลังข้อมูลจะพยายามรวมเนื้อหา (merge) ของผู้ใช้

**ข้อที่ 14. แนวทางในการแก้ไข conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลาย ๆคนเข้าด้วยกัน**

**1.สร้าง ชื่อไฟล์ที่บ่งบอกความเป็นเจ้าของ เช่น รหัสนิสิต 57160539**

2.การพูดคุย การสื่อสาร มันสำคัญอย่างมาก (แก้ไขการ Merge conflict ได้ง่าย ๆ มากเลยนะเพียงแค่เราคุยกันในเรื่องที่ควรจะคุยกันมากขึ้น)

### 3. [Mob programming](http://mobprogramming.org/)เป็นวิธีการที่ทรงประสิทธิภาพอย่างมากและเชื่อได้เลยว่า แก้ไขปัญหา Merge conflict ได้ 100%เนื่องจากทุกคนมานั่งทำงานด้วยกันใช้เครื่องทำงานเดียวกัน

### 4.ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้นหรือในแต่ละ class แต่ละ method ควรมีเหตุผลเดียวในการเปลี่ยนแปลงเท่านั้น

### 5.สาเหตุของปัญหาเกิดจากอะไร

**ข้อที่ 15. แนวทางในการลด conflict ที่เกิดขึ้นเมื่อมีการ merge โปรแกรมของผู้พัฒนาหลาย ๆคนเข้าด้วยกัน**

1.มีการประชุมเพื่ออธิบายโครงสร้างของงาน

2.ทำการประเมิณผลงานและตรวจสอบความถูกต้อง ของผู้พัฒนา

3.ก่อนที่จะหาวิธีการแก้ไขปัญหาเราต้องรู้และเข้าใจก่อนว่า ที่ไปที่มาของปัญหามันคืออะไรกันแน่

### ****ข้อที่ 16 git คืออะไร ต่างจาก github อย่างไร****

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

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

**ความต่าง** คือ 1.หน้าคอมมิทของ Git แสดงให้เราเห็นการเปลี่ยนแปลงทั้งหมดที่เกิดขึ้นกับคอมมิทนี้ แต่บางครั้งเราอาจต้องการเห็นไฟล์ในตอนที่ยังไม่ถูกแก้ไขเพื่อให้แน่ใจว่าโค้ดมีหน้าตาเป็นอย่างไรในตอนนั้น

2.แทนที่เราจะให้คุณเห็นโค้ดสำเร็จของแอพ Microscope ในครั้งเดียว เราก็ถ่ายภาพโค้ดของโปรแกรมไว้ทีละขั้นจนครบทุกขั้นซึ่งคุณสามารถเข้าไปดูออนไลน์ได้ที่ GitHub

**17. จุดประสงค์หลักในการ branch คือ**

git branch เป็น feature ที่ช่วยให้นักพัฒนาสามารถที่จะทำงานได้สะดวกขึ้น ยกตัวอย่างเช่น เรามีโค๊ดที่ดีอยู่แล้ว แต่อยากจะทดลองอะไรนิดๆหน่อย หรือแก้ไขอะไรก็ตาม ไม่ให้กระทบกับตัวงานหลัก ก็เพียงแค่สร้าง branch ใหม่ขึ้นมา เมื่อแก้ไขหรือทำอะไรเสร็จแล้ว ก็ค่อยเซฟกลับมาที่ master เหมือนเดิม

**18. Fast forward merge คืออะไร และทำไมการ push ไปที่ Remote Repo**

จะเห็นคำว่า "Fast forward" ใน merge นั้น เพราะ commit ที่ถูกชี้โดย branch ที่ merge มันเป็น upstream ของ commit ที่คุณอยู่โดยตรง Git ก็เลยขยับ pointer ไปข้างหน้า พูดอีกนัยหนึ่งก็คือ เวลาที่พยายามจะ merge commit ซักอันเข้ากับ commit ที่สามารถไปถึงได้โดยการตาม history ของ commit อันแรก Git จะทำให้ทุกอย่างง่ายขึ้นโดยการขยับ pointer ไปข้างหน้าเพราะมันไม่มีงานที่ถูกแยกออกไปให้ merge สิ่งนี้เรียกว่า "fast forward".

จำเป็นต้องมีการ pushไปที่ Remote Repo เพื่อที่จะอ้างอิงถึงที่อยู่ที่เราจะต้องการเก็บข้อมูล เช่น

**git remote add origin** [**https://github.com/topblitz/midterm\_887342.git**](https://github.com/topblitz/midterm_887342.git)

**19.หน้าที่หลักของ git pull คือ อะไร**

Git Pull จริงๆแล้ว git pull ก็คือรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้วนั่นเอง ซึ่งจริงๆแล้ว git pull มันก็คือการทำ git fetch และต่อด้วยgit merge อัตโนมัตินั่นเอง

**20.แผนภาพสื่อความหมายอะไร**



**GITFLOW**

1. เมื่อจะพัฒนาฟีเจอร์ใหม่ ให้แตก branch Feature มาจาก Develop ให้ทีมพัฒนาโค้ดกันใน branch นั้น เมื่อพัฒนาเสร็จให้ merge โค้ดเข้า Develop แล้วลบ branch Feature ทิ้ง

2. เมื่อจะเอาโค้ดขึ้นโปรดักชั่น ให้แตก branch Release ออกมาจาก Develop ให้ Tester ตรวจสอบว่าโปรแกรมทำงานถูกต้องหรือเปล่า มีบักหรือเปล่า หากมีบักก็แก้ใน Branch Release เลย จนเมื่อโปรแกรมถูกต้องสมบูรณ์จึง merge เข้า Master เพื่อเอาขึ้นโปรดักชั่นต่อไป อย่าลืม Tag เวอร์ชั่น และ merge เข้า Develop ด้วยจากนั้นจึงลบ branch Release ทิ้ง

3. ทีนี้หลังจากที่เอาโค้ดขึ้นโปรดักษ์ชั่นแล้ว มันอาจจะพบบักที่ไม่คาดฝัน อาจจะเพราะ environment เครื่องหรืออะไรก็แล้วแต่ ซึ่งจำเป็นต้องรีบแก้ ให้แตก branch Hotfix ออกมาจาก Master แล้วแก้ไขบักซะ หลังจากแก้ไขเสร็จแล้ว ให้ merge โค้ดเข้าไปยัง Master แล้วลบ branch Hotfix ทิ้ง

จากภาพ เราจะเห็นวิธีการตั้งเวอร์ชั่นของโปรแกรมที่เราเขียนด้วย ถ้าหากเป็นการเอาโค้ดจาก Release ขึ้นโปรดักชั่น เขาจะใช้หมายเลขเวอร์ชั่นใหญ่ เช่น 1.0, 2.0, 3.0 แต่ถ้าเป็นการ Hotfix เขาจะใช้เลขเวอร์ชั่นย่อย เช่น 1.1,  1.2, 2.1, 3.1 เป็นต้น

เบื้องหลังการทำงานของคำสั่งก็คือ แตก branch, merge โค้ด, ลบ branch ให้นั่นเอง