1. ช่วยให้คุณสามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้

ช่วยให้คุณเปรียบเทียบการแก้ไขที่เกิดขึ้นในอดีต ดูว่าใครเป็นคนแก้ไขคนสุดท้ายที่อาจทำให้เกิดปัญหา แก้ไขเมื่อไร

ช่วยให้คุณสามารถกู้คืนไฟล์ที่คุณลบหรือทำเสียโดยไม่ตั้งใจได้อย่างง่ายดาย

1. แม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ การ checkout แต่ละครั้งคือการทำสำรองข้อมูลทั้งหมดแบบเต็ม ๆ นั่นเอง
2. ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ
3. ทำ merge บ่อยๆ เพราะปัญหาใหญ่ ๆ ของ Merge conflict เกิดจากจำนวน source code ที่ชนหรือขัดแย้งกันมากเหลือเกิน ในหนึ่ง class ใน หนึ่ง method นั้นควรจะมีหน้าที่การทำงานเพียงอย่างเดียวเท่านั้นหรือในแต่ละ class แต่ละ method ควรมีเหตุผลเดียวในการเปลี่ยนแปลงเท่านั้น และควรจะพูดคุยกันในทีม
4. มีรูปแบบพื้นฐานอะไรกันบ้าง เพื่อนำไปใช้พิจารณาต่อไป
5. Mainline Branch Strategy จะทำงานบน local branch ของตัวเอง  
   เมื่อทำการพัฒนาเสร็จแล้วให้ทำการ merge code เขาไปยัง branch หลัก

### Feature Branch Strategy ทำการจัดการ code แต่ละ branch แยกตาม feature ไปเมื่อ developer ทำการพัฒนา และ ทดสอบเสร็จแล้วจะทำการ merge code จากfeature branchไปยัง integration branch จากนั้นทำการทดสอบอีกครั้งบน integration branch เมื่อทุกอย่างเรียบร้อย ก็ทำการ merge กลับไปยัง branch หลักต่อไป

**3. Environment Branch Strategy เ**ป็นอีกรูปแบบหนึ่งของการจัดการ branch  
ซึ่งจะทำการสร้าง branch ขึ้นมาตามจำนวนของ environment ของระบบ เช่น

* Development
* Testing
* Staging
* Pre-production
* Production

### 4. Released Branch Strategy ในแต่ละ branch คือ แยกไปตามแต่ละ version ของระบบงานแต่ส่วนการทำงานหลักยังคงอยู่ที่ branch หลัก

### สุดท้าย แต่ละ productล้วนมีรูปแบบที่แตกต่างกันดังนั้น Branching Strategy ก็ย่อมแตกต่างกันไปอาจจะทำการเพิ่ม หรือ ลด กระบวนการลงไปอีกก็ได้

### **Git** คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ ส่วน[GitHub](https://github.com/) คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ [Git](http://git-scm.com/) โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ

1. เพื่อแยกการพัฒนาความสามารถใหม่ๆ ออกมาจากส่วนหลัก โดยที่ git มี master branch มาให้อยู่แล้ว หลังจากทำการสร้าง git repository ขึ้นมาในการใช้ branch เพื่อแยกการพัฒนานั้น จะต้องทำการ merge การเปลี่ยนแปลงเข้ามายัง master branch หลังจากที่พัฒนาเสร็จแล้วเสมอ
2. Branch **master** ไปอยู่ที่ Commit เดียวกับ Branch **test** และสายของ Commit เป็นเส้นตรง จึงควรจะต้อง merge แบบนี้เพราะตัว HEAD ของเรา independent กับ merged commit, ดังนั้นกรณนี้จะเกิดการ merge จริง และมีการสร้าง commmit object ใหม่ขึ้นมา
3. git pull ใช้ดึงความเปลี่ยนแปลงจาก remote มายัง local และรวมเข้าด้วยกัน (มีค่าเท่ากับfetch+merge)
4. เป็นการ การแตก branch และการ merge การสร้าง branch ใหม่และย้ายไปทำงานบนนั้น, หรือว่าย้ายกลับไปกลับมา หรือว่า merge local branches เข้าด้วยกัน เพื่อให้ง่ายต่อการทำงานร่วมกัน