**DevOps**

DevOps คือ ชุดแนวทางการพัฒนาซอฟต์แวร์ที่รวมการพัฒนาซอฟต์แวร์และการดำเนินงานด้านเทคโนโลยีสารสนเทศ เพื่อลดวงจรชีวิตของการพัฒนาระบบในขณะที่นำเสนอคุณลักษณะการแก้ไขและการอัปเดตบ่อยครั้งเพื่อให้สอดคล้องกับวัตถุประสงค์ทางธุรกิจ DevOps มีหลักการคือ การนำความคล่องตัวมาสู่งานการพัฒนาและการดำเนินงานระบบ DevOps อาจไม่สามารถทดแทน Agile ได้แต่ DevOps จะสามารถนำมาปรับปรุงต่อยอดส่วนที่ยังขาดในเรื่องของการดำเนินงานได้ดี DevOps อาจไม่ใช่การแทนที่ Agile ได้ แต่ก็อาจใช้ร่วมเพื่อให้การปฏิบัติมีระบบดีขึ้น เมื่อเวลาผ่านไป Agile ก็ได้เพิ่มความท้าทายและ DevOps ก็กลายเป็นแนวทางปฏิบัติที่เหมาะสมที่สุด

- กระบวนการหรือการปฏิบัติ ของ DevOps เกี่ยวข้องกับชุดของกระบวนการทางเทคนิคเช่น การพัฒนาอย่างต่อเนื่อง, การรวมอย่างต่อเนื่อง (CI), การทดสอบอย่างต่อเนื่อง (CI), การปรับใช้อย่างต่อเนื่อง (CD) และการตรวจสอบอย่างต่อเนื่อง

- พื้นที่โฟกัสที่สนใจให้ความสำคัญ จะมุ่งเน้นไปที่การรับประกันซอฟต์แวร์ที่มีคุณภาพในเวลาที่เหมาะสม การรับประกันคุณภาพจะทำโดยการตรวจสอบซอฟต์แวร์อย่างต่อเนื่องหลังจากการปรับใช้

- มีรอบระยะเวลาทั้งในส่วนของการเผยแพร่และการพัฒนา จะมุ่งเน้นไปที่ระยะการปล่อยที่สั้นลง มุ่งให้มีการส่งมอบงานที่เร็วขึ้นแต่ติดตามผลการตอบรับอย่างต่อเนื่องทันที

- ผู้ให้ข้อเสนอแนะ จะมีการวัดผลจากทีมภายในพัฒนาภายใน โดยอาจใช้เครื่องมือการตรวจสอบมาช่วยอย่างต่อเนื่อง

**CI/CD**

**CI คืออะไร**  
CI (Continuous Integration)คือ กระบวนการรวม source code ของคนในทีมพัฒนาเข้าด้วยกัน และมีการ test ด้วย test script เพื่อให้แน่ใจว่าไม่มี error ในส่วนใดๆ ของโปรแกรม แล้วถึงทำการ commit ไปที่ branch master อีกต่อนึง  
  
โดยในการพัฒนานั้น มักใช้ Build Server มาช่วย กล่าวคือจะเริ่มทำการ Integration กันตั้งแต่เมื่อมีการเปลี่ยนแปลง Source Code ที่ Repository กลาง ระบบจะทำการตรวจสอบ Code หลังจากการเปลี่ยนแปลงว่าทำงานร่วมกันได้หรือไม่ตั้งแต่ Compile, Testing  
  
**CD คืออะไร**  
1. CD (Continuous Deployment) คือ การ Deploy ขึ้น production โดยจะทำทุกขั้นตอน ตั้งแต่ complie build ไปจนถึง deploy ขึ้น production แบบอัตโนมัติทั้งหมด

2. CD (Continuous Delivery) คือ การทำทุกขั้นตอนคล้ายกันกับ Continuous Deployment ต่างกันตรงที่จะไม่มีการ deploy ขึ้น production ขึ้นในทันที แต่จะเป็นการทำ manual deploy หรือจะเป็นแบบ one click deploy ก็ได้ หลังจาก QA หรือ ฝ่าย Business พอใจในตัว product ที่ทีมทำออกมา

**ข้อที่ 1 ทำงานได้อย่างรวดเร็ว**

ในขั้นตอนการทำงานของ CI/CD หรือ pipeline นั้น  
จะต้องทำงานได้อย่างรวดเร็ว ตั้งแต่การ build, test และ deploy  
เพื่อช่วยทำให้เราได้รับ feedback ต่าง ๆ จากการเปลี่ยนแปลงได้รวดเร็ว  
  
ถ้าทำงานช้ามาก ๆ  
ก็ต้องรอนาน หรือไม่อยากใช้งานไปในที่สุด  
อย่าลืมเป้าหมายหลังของ CI/CD

**ข้อที่ 2 มีความน่าเชื่อถือ**

ความเร็วอย่างเดียวยังไม่เพียงพอ แต่ต้องมีความน่าเชื่อถือหรือเสถียรด้วย  
ไม่ใช่ทำงานผ่านบ้าง ไม่ผ่านบ้าง โดยที่ไม่แก้ไขอะไรเลย  
มันทำให้ขาดความน่าเชื่อถือ  
รวมทั้งยังรองรับการ scale และรองรับ use case ต่าง ๆ มากขึ้นด้วย

**ข้อที่ 3 environment ต่าง ๆ ควรมีรูปแบบเดียวกัน**

ปัญหาอย่างหนึ่งของ environment คือ ความแตกต่างกันมาก  
สิ่งหนึ่งที่ควรจะเป็นคือ environment ต่าง ๆ ควรเหมือนหรือคล้ายกันมากที่สุด  
เพื่อให้การทำงานมีความน่าเชื่อถือ  
ไม่ใช่ไปเจอแต่บน production environment อย่างเดียว  
ส่วน environment อื่น ๆ กลับไม่เจอ  
ถ้าเป็นแบบนี้ แสดงว่าน่ากลัวมาก ๆ

**ข้อที่ 4 ควรอธิบายให้ทีมและส่วนงานที่เกี่ยวข้องเกี่ยวกับ CI/CD**

ปัญหาที่เจอบ่อย ๆ คือ ความไม่เข้าใจของการนำ CI/CD มาใช้งาน  
ว่านำมาใช้งานเพื่ออะไร  
ว่านำมาใช้งานเพื่อแก้ไขปัญหาอะไร  
จากนั้นตองทำอะไรบ้าง  
สุดท้ายจึงช่วยกันหาวิธีการว่าต้องทำอย่างไร  
น่าจะทำให้ผลที่ออกมาดีสำหรับทุก ๆ ฝ่าย

**หากองค์กรต้องการนำเอา DevOps และ CI/CD เข้ามาเป็นส่วนเสริมในกระบวนการพัฒนาซอฟต์แวร์จะต้องทำอย่างไรบ้าง**

**ขั้นตอนที่ 0 ระบุปัญหาก่อนว่ามีอะไรบ้าง** ?

จากนั้นจึงทำการเรียกลำดับตามความสำคัญหรือผลกระทบที่เกิดขึ้น  
ว่าจะแก้ไขอะไรก่อนหลัง  
แนะนำให้ค่อย ๆ แก้ไขทีละเรื่อง  
ถ้ามันสำคัญหมด คงจะมั่วกันน่าดู   
เนื่องจากเราจะไร้ทิศทางมาก ๆ      
  
ถ้าไม่มีปัญหา จะปรับปรุงไปทำไม จริงไหม ?

**ขั้นตอนที่ 1 ลดความหลากหลาย**

สิ่งที่พบเจอบ่อยมาก ๆ ในการพัฒนา software คือ  
ความหลากหลายของการทำงาน ไม่ว่าจะเป็น

* ขั้นตอนการทำงาน
* Environment ต่าง ๆ ที่ใช้งาน
* เครื่องมือที่หลายหลาย ตามใจใครหลาย ๆ คน
* การ configuration ที่ไร้ทิศทาง
* ข้อมูลที่หลายหลายชนิด อยู่หลายที่

**คำถามคือ ถ้ายังมีความหลากหลายแบบนี้แล้ว**  
เราจะจัดการอย่างไร ?  
ยิ่งเริ่มเอาระบบการทำงานแบบอัตโนมัติเข้ามาช่วยปรับปรุง มันยิ่งลำบากหรือไม่ ?  
  
ดังนั้น ควรต้องลดความหลากหลาย  
หรือลองหาสักแนวทางสำหรับการเริ่มต้นดีไหม ?  
ยิ่งมีความหลากหลาย ยิ่งก่อให้เกิดปัญหาหรือไม่ ?  
ดังนั้นลองดูว่า อะไรบ้างที่ตัดออกไปแล้ว  
มันทำให้เราทำงานง่ายขึ้นบ้าง ชีวิตน่าจะดีขึ้นนะ

**ขั้นตอนที่ 2 ต้องรู้ขั้นตอนการทำงานตั้งแต่ต้นจนจบ**

เราไม่สามารถจะปรับปรุงอะไรได้  
ถ้าเราไม่รู้กระบวนการทำงานตั้งแต่ต้นจนจบ  
เนื่องจากเรามักจะมีคนที่รู้เฉพาะจุดหรือบางส่วน  
คำถามหรือแล้วภาพใหญ่ละ  
บอกเลยว่า ไม่มีใครรู้หรือรู้ก็แบบลาง ๆ  
ต้องทำการนัดประชุมกันใหญ่โต หรือใช้เวลานานมาก ๆ

**ดังนั้นสิ่งที่ควรทำคือ**  
อะไรที่ไม่รู้ทำให้รู้ (มักจะไม่รู้ว่า ไม่รู้อะไร !!)  
จากนั้นทำการสร้างเอกสาร เพื่ออธิบายการทำงานตั้งแต่ต้นจนจบไว้  
จากนั้น share ให้ทุกคนที่เกี่ยวข้อง  
จะได้ทำความเข้าใจ แก้ไขหรือปรับปรุงกันต่อไป  
มันจะทำให้เราเห็นว่า การทำงานปัจจุบันเป็นอย่างไร  
ตรงไหนมีปัญหา ตรงไหนที่มันเป็นคอขวด

***ให้จำไว้ว่า ถ้าขั้นตอนมันห่วยแล้ว  
จะเอาระบบการทำงานแบบอัตโนมัติมาใช้ มันก็ห่วย  
ดังนั้นปรับปรุงการทำงานก่อนนะ***

มันช่วยทำให้เราพูดคุยกัน ทำงานร่วมกัน  
เพื่อเข้าใจซึ่งกันและกัน ได้ช่วยกัน review ขั้นตอนการทำงานอีกด้วย

**เมื่อถึงขั้นตอนตรงนี้ จะเห็นว่า**

มันคือเรื่องพื้นฐานมาก ๆ ที่เรามักไม่ทำกัน ใช่ไหมนะ ?  
ไม่ได้สนใจเทคโนโลยีหรือเครื่องมือหรือ framework อะไรเลย  
มันคือเรื่องของการพูดคุย ทำงานร่วมกัน เปิดเผย  
จากนั้นก็มาดูขั้นตอนการทำงานตั้งแต่ต้นจนจบ  
เพื่อช่วยกันปรับปรุงให้ดีขึ้น  
เป้าหมายหลักคือ การแก้ไขปัญหาที่ตั้งไว้ตั้งแต่ต้นนั่นเอง

ต่อจากนี้ต่อไป เริ่มนำเครื่องมือและเทคโนโลยีมาใช้แล้ว

**ขั้นตอนที่ 3 เข้าสู่กระบวนการพัฒนาและส่งมอบ software**

การพัฒนาและส่งมอบ software ที่ดีประกอบไปด้วย

* การจัดการ source code หรือ Source Control Management (SCM)
* กระบวนการ build ของ software
* กระบวนการ deploy software
* การ configuration หรือ provisioning environment ต่าง ๆ สำหรับการ deploy software

ซึ่งเราอาจจะต้องกำหนดการทำงานเหล่านี้ให้ชัดเจน  
หรือดีที่สุดคือ ช่วยกันสร้างระบบที่ทำงานแบบอัตโนมัติขึ้นมา  
เพื่อลดงานที่ต้องมีคนเข้าไปยุ่งเกี่ยวให้มากที่สุด  
เพราะว่า มันคืองานที่ทำซ้ำ ๆ บ่อยมาก  
และที่สำคัญคือ เพื่อลดปัญหาที่เกิดจากคนให้มากที่สุด  
เรามักจะได้ยินคำว่า **work on my machine !!**

**ขั้นตอนที่ 4 กระบวนการทดสอบแบบอัตโนมัติ**

การทดสอบนั้นเป็นหัวใจของการพัฒนา software และ DevOps เลย  
มันสะท้อนในเรื่องของคุณภาพ software ที่ส่งมอบมากพอสมควร  
โดยการทดสอบควรต้องเป็นแบบอัตโนมัติให้ได้มากที่สุด  
เพื่อช่วย validate ว่า software ของเรา  
ยังคงทำงานได้อย่างถูกต้องตามที่คาดหวัง  
  
**ที่สำคัญต้องทำงานได้รวดเร็วด้วย**  
มิเช่นนั้น ปัญหาต่าง ๆ ก็ยังคงอยู่เสมอ มันน่ากลัวมาก ๆ  
ถ้าเราทำการ deploy software บ่อย ๆ แต่ผิดเยอะบ่อย ๆ  !!!  
มันไม่น่าจะใช่ผลที่เราต้องการใช่ไหม ?

**เมื่อเราทำตามมาจนถึงขั้นตอนนี้แล้ว**

เราได้ผ่านความต้องการพื้นฐานตามแนวคิด DevOps มาแล้ว  
สามารถแก้ไขปัญหาที่เรากำหนดได้แล้ว  
จากนี้ก็เป็นเรื่องที่เราต้องไปต่อแล้ว  
ไม่ว่าจะเริ่มสร้างระบบอื่น ๆ ขึ้นมา เพื่อให้ใช้งานง่ายขึ้น  
เช่นระบบแบบ **self-service** ไม่ต้องมาผ่านขั้นตอนที่มากมายหรือช้ากันแล้ว  
อยากได้อะไรไปสร้างเองได้เลย  
มันก็อยู่ที่ความต้องการขององค์กรต่อไปแล้ว