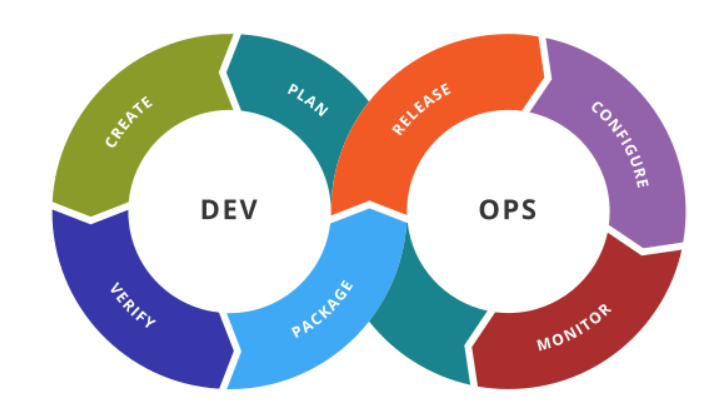
1.) DevOps คือ

DevOps เป็นคำที่ย่อมาจาก Development Operations เป็นศาสตร์ที่ช่วยเชื่อมกระบวนการพัฒนาโปรแกรม การส่งเอาขึ้นไป Deploy ให้ผู้ใช้เข้าถึงอย่างได้อย่างมีประสิทธิภาพ มีคุณภาพ ไปจนถึงการติดตามสถานะการทำงานของโปรแกรมเพื่อนำกลับมาพัฒนาปรับปรุงกระบวนการพัฒนาโปรแกรม

DevOps Engineer เปรียบได้กับ System Admin ในสมัยก่อน ที่ทำหน้าที่เอาโปรแกรมขึ้นไปรันบน Server ให้ได้ แล้วคอยดูไม่ให้มันล่มนั่นล่ะ แต่พัฒนาร่างขึ้นมาทำเรื่อง Automation ด้วย คือทำให้กระบวนการทั้งหมดที่ต้องทำมือในสมัยก่อน ทำงานโดยอัตโนมัติได้ นอกจากนั้นยังลงไปช่วยนักพัฒนาทำงานด้วย ในส่วนของการวางสภาพแวดล้อมในการพัฒนาโปรแกรมที่ดี

**DevOps Process**

DevOps นั้นมีรูปแบบการทำงานค่อนข้างจะตายตัวตามรูปที่อยู่ข้างล่าง ซึ่งจะพยายามทำให้ Process ต่างๆ ทำงานไปได้โดยอัตโนมัติ ไม่ต้องมีคนคอยเข้าไปกด Deploy เองอีกต่อไป ยกเว้นในส่วนของการการวางแผน การพัฒนา และดูผลลัพธ์ ในส่วนนี้ยังจำเป็นต้องอาศัยสมองคนต่อไปอยู่ ในเรื่องของรายละเอียด ผมขออธิบายลงไปในแต่ละขั้นตอน

#### Plan

การจะพัฒนาโปรแกรมขึ้นมา ต้องมีการวางแผนร่วมกันจากทุกฝ่ายที่เกี่ยวข้องอย่างละเอียดก่อน โปรแกรมเราจะทำอะไรบ้าง จะพัฒนาด้วยภาษาอะไร Framework อะไร จะรันบน Platform อะไร จะไปลงไว้ที่ไหน จะเก็บข้อมูลอย่างไร จะแก้ปัญหาที่เกิดขึ้น และจะพัฒนา Process การทำงานอย่างไร

#### Create

การสร้างหรือการพัฒนา นี่ก็คือขั้นตอนของการเขียนโปรแกรมขึ้นมา DevOps เข้ามาช่วยทำให้ Developers ทุกคนทำงานบน Environment แบบเดียวกัน ไม่ว่าเขาจะพัฒนาบน OS ใดก็ตาม

เวลา Developer ทำงานบนคอมพิวเตอร์คนละ OS กัน มักจะมีปัญหา “ทำไมโปรแกรมรันบนเครื่องเธอไม่ได้ล่ะ บนเครื่องฉันก็รันได้อยู่นะ!” ซึ่งมันไม่ควรจะเกิดขึ้นแล้วในยุคนี้ที่เรามี Docker ที่ช่วยให้ทุกคนพัฒนาโปรแกรมอยู่บนพื้นฐาน Environment เดียวกันเป๊ะ ตั้งแต่ขั้นตอนการพัฒนาไปจนถึงการส่งโปรแกรมขึ้น Production

#### Verify

การตรวจสอบอยู่อย่างสม่ำเสมอว่าโปรแกรมที่พัฒนาขึ้นมา ไม่ว่าจะเพิ่มฟีเจอร์อะไรเข้ามาจะไม่ไปกระทบหรือไปก่อบั๊กทำให้การใช้งานเดิมพังเสียหายนั้นเป็นเรื่องสำคัญมาก โดยเฉพาะโปรแกรมขนาดใหญ่ที่ร่วมกันทำกับทีมงานจำนวนมาก เราจะมั่นใจได้อย่างไรว่า Code ที่ Developer X ส่งขึ้นมาจะเข้ากันได้ดีกับการทำงานเดิม

วิธีการที่จะรู้ได้เร็วที่สุดคือ เราจะต้องมีการทำ Testing อย่างครอบคลุมในทุกๆ Components ของโปรแกรม แล้วทำให้มันรันโดยอัตโนมัติทุกครั้งก่อนจะปล่อยโปรแกรมออกไป เมื่อใดที่ Test Fail ก็จะรู้ได้ทันทีว่ามีปัญหาเกิดขึ้นแล้ว ให้หยุดการส่งงานที่มีปัญหาไปก่อน สามารถเข้าไปหาสาเหตุและแก้ไขปัญหาได้อย่างทันท่วงที

#### Package

การจะส่งโปรแกรมขึ้นไปรันบน Server เพื่อความสะดวกรวดเร็ว จะต้องมีการเอาโปรแกรมไปใส่ใน Technology ที่ออกแบบมาเพื่อนำโปรแกรมไปรันได้อย่างราบลื่น ซึ่งในปัจจุบันนี้เราใช้ Docker กัน

เวลาจะใช้ เราจะสร้าง Docker Image ที่เอาตัวโปรแกรมพร้อมสำหรับการรันใส่อยู่ในนั้น แล้วเอา Docker Image นี้ไป Deploy ที่ไหนก็ได้สามารถรัน Docker ได้ โปรแกรมก็จะรันขึ้นมาพร้อมใช้งาน

#### Release

เมื่อมี Docker Image พร้อมแล้ว ก็พร้อมสำหรับการนำโปรแกรมไปรันบน Deployment Platform ที่ต้องการ ตอนนี้กำลังทำของ Development หรือทำหรือ Production อยู่ จะต้องเลือกไป Deploy ให้ถูกที่

#### Configure

โปรแกรมที่พัฒนานั้น **จำเป็นอย่างมาก** ที่จะต้องตั้งค่าได้ ตัวอย่างการตั้งค่าที่ควรมีเลยก็เช่น การเลือก Database ตั้งค่า Email เป็นต้น

ยิ่งถ้ามี Service แยกอื่นๆ ที่ต้องใช้แล้ว ยิ่งสำคัญ เพราะ Development Server และ Production Server ควรอยู่แยกกันอย่างเด็ดขาด เราจะต้องหาทางสร้าง Docker Image มา 1 อัน แล้วเราจะใช้มัน Deploy ลงบน Environments ที่ต่างกันให้ได้

วิธีการที่จะทำได้ก็คือการทำให้โปรแกรมของเราตั้งค่าได้เนี่ยแหละ คือโปรแกรมที่ไปรันเป็นตัวเดียวกัน แต่เปิดด้วยการตั้งค่าที่แตกต่างกัน ทำให้ได้การทำงานที่แตกต่างกัน โดยปรกติจะใช้ไฟล์ .env และ Environment Variable ในการตั้งค่ากันเป็นมาตรฐานที่ใช้กันบนทุก Platform

#### Monitor

สุดท้ายที่ขาดไม่ได้เลยคือการตรวจสอบว่าโปรแกรมมันขึ้นไปแล้วทำงานได้ปรกติ มีสุขภาพแข็งแรงดี การดู Log ว่าโปรแกรมเราทำงานมีปัญหาตรงไหน มีความสำคัญอย่างมากในการแก้ไขบั๊ก ยิ่งในตอนนี้ที่มักจะออกแบบให้โปรแกรมมันสามารถทำให้ Horizontal Scale ได้ (การสร้างเครื่องขึ้นมาหลายๆ เครื่องให้มาแบ่ง Load ทำงานแยกกันได้) จำเป็นอย่างยิ่งที่จะต้องมี Centralized Logging System รวบรวม Log เอามาไว้ในที่เดียว เพราะการ Remote เข้าไปอ่าน Log ในแต่ละเครื่องเป็นไปไม่ได้อีกแล้ว

อีกส่วนหนึ่งที่สำคัญไม่แพ้กันคือการทำ Monitoring ดูโหลดต่างๆ ของระบบ ตอนนี้มีกี่เครื่องเปิดขึ้นมาใช้งาน เครื่องล่มหรือเปล่า ให้บริการได้ทันที่ลูกค้าเข้ามาใช้หรือไม่ เรื่องเหล่านี้ล้วนมีเครื่องมือออกมาแก้โจทย์เหล่านี้แล้วทั้งสิ้น เพียงแต่ต้อง Implement โปรแกรมให้ไปรองรับการใช้งานกับเครื่องมือเหล่านั้น

2.) CI/CD คือ

**Continuous Integration(CI)**คือ กระบวนการรวม source code ของคนในทีมพัฒนาเข้าด้วยกัน และมีการ test ด้วย test script เพื่อให้แน่ใจว่าไม่มี error ในส่วนใดๆ ของโปรแกรม แล้วถึงทำการ commit ไปที่ branch master อีกต่อนึง

**Continuous Deployment** ต้องมีการ Deploy ขึ้น production แน่ๆ ซึ่งก็เป็นอย่างที่ทุกคนเข้าใจ คือ Continuous Deployment จะทำทุกขั้นตอน ตั้งแต่ complie build ไปจนถึง deploy ขึ้น production แบบอัตโนมัติทั้งหมด

**Continuous Delivery** มีการทำทุกขั้นตอนคล้ายกันกับ**Continuous Deployment**ต่างกันตรงที่จะไม่มีการ deploy ขึ้น production ขึ้นในทันที แต่จะเป็นการทำ manual deploy หรือจะเป็นแบบ one click deploy ก็ได้ หลังจาก QA หรือ ฝ่าย Business พอใจในตัว product ที่ทีมทำออกมา

3.) ทางทีมพัฒนาระบบของเราจะแบ่ง environment เป็น 4 ส่วนคือ Local , Alpha , Staging , Production มีขั้นตอนดังนี้  
1. Developer เมื่อทำการพัฒนา feature เสร็จ จะทำการ build, test และ run บนเครื่องของตัวเอง (Local) เพื่อทำให้แน่ใจว่าระบบทำงานได้ถูกต้องและให้แน่ใจว่าสิ่งที่เปลี่ยนแปลงไม่กระทบส่วนอื่น ๆ

2. ทำการดึง source code ล่าสุดจาก Repository ของระบบ เพื่อตรวจสอบว่ามีการเปลี่ยนแปลงหรือไม่ถ้ามีการเปลี่ยนแปลงก็ให้ทำการรวม หรือ merge ที่เครื่องของ Devleoper ก่อน จากนั้นจึงทำการ build, test และ run อีกรอบ เมื่อทุกอย่างผ่านทั้งหมด ให้ทำการส่งการเปลี่ยนแปลงไปยัง Repository กลาง

3. เมื่อ Repository กลางมีการเปลี่ยนแปลง จะต้องมีระบบ CI ทำการ build, หลังจาก build จะส่งต่อไป run unit testing ก่อนถ้าผ่านหมดถึงจะส่งต่อไปยังระบบ Continuous Delivery เพื่อ deploy to alpha environment

4. เมื่อ source code ถูก deploy to alpha environment แล้วจะ trigger ไปสั่งให้ run job automated testing ใน level ของเทสเคส smoke test ซึ่งเป็นชุดเทสเคสย่อยๆไม่เยอะมากเฉพาะในส่วนของ feature code ที่ถูก deploy มาเท่านั้น

5. หลังจาก run smoke test เสร็จแล้วถ้าเกิดว่า run มีบางส่วนไม่ผ่านทั้งหมดจะไม่ส่งต่อไปยังระบบ Continuous Delivery เพื่อ deploy to staging environment QA จะทำการ investigate ว่าเกิดจากอะไร เป็นที่ระบบมี Bug เกิดขึ้นจริงหรือไม่ ถ้ามี bug ก็ให้ dev แก้ไข และ deploy มาใหม่ วน loop ใหม่

6. กรณีหลังจาก run smoke test ผ่านทั้งหมดจะส่งต่อไปยังระบบ Continuous Delivery เพื่อ deploy to Staging environment เมื่อ source code ถูก deploy to staging แล้ว จะ trigger ไปสั่งให้ run job automated testing ใน level ของเทสเคส regression test และ QA ก็ทำการทดสอบ Acceptance testing ไปด้วยพร้อมๆกันที่ Staging environment นี้ เมื่อมีการ deploy ซ้ำๆ เพื่อ fixed bug จากที่ QA เจอ หรือที่พบเจอจากการ run regression test แล้ว fail ก็จะเป็นการวน loop ตั้งแต่ต้นจนจบ จนกระทั้ง ทุกอย่างผ่านหมด Business ฟันธง!! มาว่าเอาขึ้น production ได้ เป็นการ confirm ว่าเราจะเอา code version สุดท้ายนี้ขึ้นไปที่ production environment

**ข้อดี&ข้อเสียจากการ ใช้ automated testing มาใช้กับ CI/CD**

**ข้อดี** สามารถทดสอบได้บ่อย ๆ รู้สถานะของระบบได้ตลอดเวลา และ ได้ feedback ของระบบอยู่อย่างเสมอ ไม่ว่าจะดีหรือร้าย ทำให้ทุกคนเห็นความคืบหน้าของการพัฒนา

**ข้อเสีย**ถ้าเขียน script test ไม่ดีก็อาจทำให้ block การ deploy ระบบได้ เช่น เมื่อ run test แล้ว fail แต่เป็นการ fail จากการเขียน script ผิด ไม่ได้เกี่ยวกับระบบมี Bug ก็จะทำให้ไปขั้นตอนต่อไปไม่ได้

อ้างอิง

# เกรียงไกร เชาว์นิธิ. DevOps คืออะไร นำมาประโยชน์ได้ยังไง และตัวอย่างการทำ DevOps ที่ Credit OK. [ออนไลน์]. เข้าถึงได้จาก : <https://www.spicydog.org/blog/introduction-to-devops-and-the-practical-use-cases-at-credit-ok/?fbclid=IwAR2RVWoRsE13QRXuWUc_m5TkpfyfIFE-qc7tImjd_bShqzABxT_Wez7BVb4>. (วันที่ค้นข้อมูล : 25 กุมภาพันธ์ 2564).