DevOps คืออะไร

DevOps คือการรวมคำว่า Developer กับ Operator เข้าด้วยกัน โดยสมัยก่อน Developer เป็นคนพัฒนา code แต่ไม่มีสิทธิ deploy code ทำให้การทำงานยุ่งยาก เกิดความผิดพลาด และใช้เวลานานในการ deploy แต่ละครั้ง เป็นการผสมผสานแนวความคิดเชิงวัฒนธรรม แนวทางปฏิบัติ และเครื่องมือต่างๆ ที่ช่วยเพิ่มความสามารถขององค์กรในการส่งมอบแอปพลิเคชันและบริการอย่างรวดเร็ว โดยพัฒนาและปรับปรุงผลิตภัณฑ์ต่างๆ ให้เร็วกว่ากระบวนการการพัฒนาซอฟต์แวร์และการจัดการโครงสร้างพื้นฐานแบบดั้งเดิม ความรวดเร็วนี้ช่วยให้องค์กรสามารถให้บริการแก่ลูกค้าของตนได้ดีขึ้น และสามารถแข่งขันในตลาดได้อย่างมีประสิทธิภาพมากขึ้น

สำหรับโมเดล DevOps ทีมพัฒนาและทีมปฏิบัติการจะไม่ทำงานแบบ “ต่างคนต่างทำ” อีกต่อไป บางครั้ง ทั้งสองทีมจะจับมือร่วมงานเป็นทีมเดียวกันโดยที่เหล่าวิศวกรจะทำงานตลอดวงจรการทำงานของแอปพลิเคชัน ตั้งแต่การพัฒนาและการทดสอบไปจนถึงการปรับใช้และการปฏิบัติงาน และพัฒนาขอบเขตความสามารถที่ไม่ได้จำกัดอยู่ที่การทำงานเพียงอย่างเดียว ในบางโมเดลของ DevOps ทีมประกันคุณภาพและทีมรักษาความปลอดภัยอาจทำงานร่วมกับทีมพัฒนาและทีมปฏิบัติการอย่างใกล้ชิดยิ่งขึ้นตลอดวงจรการทำงานของแอปพลิเคชัน ในเวลาที่การรักษาความปลอดภัยเป็นจุดสำคัญของทุกคนในทีม DevOps เราจะเรียกว่า DevSecOps

ทีมต่างๆ ใช้ข้อปฏิบัติในการเปลี่ยนกระบวนการต่างๆ ที่เคยทำงานแบบแมนนวลและเชื่องช้าให้ทำงานอัตโนมัติ พวกเขาใช้ชุดเทคโนโลยีและเครื่องมือต่างๆ ที่ช่วยให้พวกเขาดำเนินการและพัฒนาแอปพลิเคชันได้อย่างรวดเร็วและเชื่อถือได้ นอกจากนั้นเครื่องมือเหล่านี้ยังช่วยให้วิศวกรเหล่านี้ทำงานได้อย่างอิสระ (เช่น การปรับใช้โค้ด หรือการจัดเตรียมโครงสร้างพื้นฐาน) ที่โดยปกติแล้วจำเป็นต้องได้รับความช่วยเหลือจากทีมอื่น พร้อมทั้งยังช่วยทีมงานทำงานได้รวดเร็วยิ่งขึ้นอีกด้วย

ประโยชน์ของ DevOps

- ความรวดเร็ว ดำเนินการอย่างรวดเร็วเพื่อให้คุณสามารถสร้างสรรค์นวัตกรรมให้แก่ลูกค้าได้รวดเร็วยิ่งขึ้น ปรับตัวต่อการเปลี่ยนแปลงของตลาดได้ดียิ่งขึ้น และมีประสิทธิภาพยิ่งขึ้นในการกระตุ้นให้เกิดผลลัพธ์ทางธุรกิจ โมเดล DevOps จะทำให้นักพัฒนาและทีมปฏิบัติการของคุณบรรลุผลดังกล่าวได้ ตัวอย่างเช่น ไมโครเซอร์วิสและการส่งมอบอย่างต่อเนื่องจะทำให้ทีมมีความเป็นเจ้าของบริการและออกอัปเดตให้แก่ลูกค้าได้รวดเร็วยิ่งขึ้น
- การส่งมอบอย่างรวดเร็ว ออกรุ่นใหม่ๆ ถี่ขึ้นและรวดเร็วยิ่งขึ้น เพื่อให้คุณสามารถสร้างสรรค์นวัตกรรมและปรับปรุงผลิตภัณฑ์ของคุณได้รวดเร็วยิ่งขึ้น ยิ่งคุณสามารถออกคุณสมบัติใหม่ๆ และแก้ไขจุดบกพร่องได้เร็วเท่าไร คุณก็ยิ่งตอบสนองความต้องการของลูกค้าและสร้างความได้เปรียบในการแข่งขันได้เร็วเท่านั้น การบูรณาการอย่างต่อเนื่องและการส่งมอบอย่างต่อเนื่อง เป็นข้อปฏิบัติที่ทำให้กระบวนการออกซอฟต์แวร์ทำงานอัตโนมัติตั้งแต่การสร้างไปจนถึงการปรับใช้
- ความเชื่อถือได้ ทำให้แน่ใจถึงคุณภาพของการอัปเดตแอปพลิเคชันและการเปลี่ยนแปลงโครงสร้างพื้นฐาน เพื่อให้คุณสามารถส่งมอบได้อย่างน่าเชื่อถือในความเร็วที่มากขึ้น ในขณะที่ยังคงรักษาประสบการณ์ใช้งานเชิงบวกของผู้ใช้ปลายทาง ใช้ข้อปฏิบัติอย่างเช่น การบูรณาการอย่างต่อเนื่อง และการส่งมอบอย่างต่อเนื่อง เพื่อ

ทดสอบว่าการเปลี่ยนแปลงนั้นทำงานได้และปลอดภัยหรือไม่ ข้อปฏิบัติในการตรวจสอบและการบันทึกจะช่วยให้คุณทราบถึงประสิทธิภาพการทำงานในทันที

- ขนาด ดำเนินการและจัดการโครงสร้างพื้นฐานและกระบวนการพัฒนาของคุณในขนาดต่างๆ การทำงานอัตโนมัติและความสอดคล้องจะช่วยให้คุณจัดการระบบที่ซับซ้อนหรือมีการเปลี่ยนแปลงได้อย่างมีประสิทธิภาพโดยมีความเสี่ยงที่ลดลง ตัวอย่างเช่น โครงสร้างพื้นฐานเป็นโค้ดจะช่วยคุณในการจัดการสภาพแวดล้อมในการพัฒนา การทดสอบ และการทำงานจริงในลักษณะที่ทำซ้ำได้และมีประสิทธิภาพยิ่งขึ้น
- การทำงานร่วมกันที่ปรับปรุงดีขึ้น สร้างทีมงานที่มีประสิทธิผลมากขึ้นภายใต้โมเดลเชิงวัฒนธรรมของ DevOps ซึ่งให้ความสำคัญกับคุณ เช่น ความเป็นเจ้าของและความรับผิดชอบ นักพัฒนาและทีมปฏิบัติการจะทำงานร่วมกันอย่างใกล้ชิด รับผิดชอบร่วมกันในหลายกรณี และนำลำดับการทำงานมารวมเข้าด้วยกัน ซึ่งจะช่วยลดความไร้ประสิทธิภาพและประหยัดเวลา (เช่น ลดระยะเวลาการส่งมอบระหว่างนักพัฒนากับทีมปฏิบัติการ เขียนโค้ดที่คำนึงถึงสภาพแวดล้อมที่นำโค้ดไปใช้งาน)
- การรักษาความปลอดภัย ดำเนินการอย่างรวดเร็วในขณะที่ยังคงรักษาการควบคุมและการปฏิบัติสอดคล้องกับกฎระเบียบ คุณสามารถรับเอาโมเดล DevOps มาใช้โดยไม่ต้องสูญเสียความปลอดภัยโดยใช้นโยบายการปฏิบัติตามกฎระเบียบที่ทำงานอัตโนมัติ การควบคุมโดยละเอียด และเทคนิคการจัดการการกำหนดค่า ตัวอย่างเช่น คุณสามารถใช้โครงสร้างพื้นฐานเป็นโค้ดและนโยบายเป็นโค้ด เพื่อกำหนดและติดตามการปฏิบัติตามกฎระเบียบในขนาดต่างๆ

ข้อปฏิบัติของ DevOps

ข้อปฏิบัติที่ดีที่สุดของ DevOps มีดังนี้

- การบูรณาการอย่างต่อเนื่อง คือข้อปฏิบัติในการพัฒนาซอฟต์แวร์โดยที่นักพัฒนานำการเปลี่ยนแปลงโค้ดของตนมารวมอยู่ในพื้นที่เก็บข้อมูลส่วนกลางอย่างสม่ำเสมอ ซึ่งหลังจากนั้นจะดำเนินการสร้างและทดสอบโดยอัตโนมัติ เป้าหมายหลักของการบูรณาการอย่างต่อเนื่อง คือเพื่อค้นหาและแก้ไขจุดบกพร่องอย่างรวดเร็ว ปรับปรุงคุณภาพของซอฟต์แวร์ และลดเวลาที่ใช้ในการตรวจสอบความถูกต้องและออกอัปเดตซอฟต์แวร์ใหม่
- การส่งมอบอย่างต่อเนื่อง คือข้อปฏิบัติในการพัฒนาซอฟต์แวร์โดยที่การเปลี่ยนแปลงโค้ดถูกสร้างขึ้น ทดสอบ และจัดเตรียมสำหรับการออกสู่การใช้งานจริงโดยอัตโนมัติ ซึ่งจะขยายตามการบูรณาการอย่างต่อเนื่องโดยนำการเปลี่ยนแปลงโค้ดทั้งหมดมาปรับใช้กับสภาพแวดล้อมการทดสอบและ/หรือสภาพแวดล้อมการใช้งานจริง ภายหลังขั้นตอนการสร้าง หากนำการส่งมอบอย่างต่อเนื่องมาใช้อย่างเหมาะสมแล้ว นักพัฒนาก็จะมีชิ้นงานจากการสร้างที่พร้อมสำหรับการปรับใช้งานที่ผ่านกระบวนการทดสอบที่เป็นมาตรฐาน
- ไมโครเซอร์วิส คือแนวทางการออกแบบในการสร้างแอปพลิเคชันเดียวโดยเป็นชุดของบริการขนาดเล็ก แต่ละบริการจะทำงานตามกระบวนการของตนและสื่อสารกับบริการอื่นผ่านอินเทอร์เฟซที่กำหนดไว้เป็นอย่างดีโดยใช้กลไกน้ำหนักเบา ซึ่งโดยทั่วไปจะเป็นอินเทอร์เฟซการเขียนโปรแกรมบน HTTP (API) ไมโครเซอร์วิสถูกสร้าง

ขึ้นตามขีดความสามารถของธุรกิจ โดยแต่ละบริการจะถูกกำหนดขอบเขตสำหรับจุดประสงค์เดียว คุณสามารถใช้งานเฟรมเวิร์กหรือภาษาการเขียนโปรแกรมที่แตกต่างกันในการเขียนไมโครเซอร์วิสได้ และสามารถนำไปปรับใช้ได้อย่างอิสระ โดยเป็นบริการเดี่ยวหรือกลุ่มบริการก็ได้

- โครงสร้างพื้นฐานเป็นโค้ด คือข้อปฏิบัติโดยที่โครงสร้างพื้นฐานได้รับการจัดเตรียมและจัดการโดยใช้โค้ดและเทคนิคการพัฒนาซอฟต์แวร์ เช่น การควบคุมเวอร์ชัน และการบูรณาการอย่างต่อเนื่อง โมเดลที่ขับเคลื่อนด้วย API ของคลาวด์ทำให้นักพัฒนาและผู้ดูแลระบบสามารถตอบโต้กับโครงสร้างพื้นฐานได้ในทางโปรแกรมในขนาดต่างๆ แทนที่จะต้องมาตั้งค่าและกำหนดค่าทรัพยากรด้วยตนเอง ด้วยเหตุนี้ วิศวกรจึงสามารถใช้เครื่องมือที่ใช้โค้ดในการสื่อสารกับโครงสร้างพื้นฐานได้ และทำงานกับโครงสร้างพื้นฐานได้ในลักษณะเดียวกับที่ทำงานกับโค้ดของแอปพลิเคชัน เนื่องจากโครงสร้างพื้นฐานดังกล่าวถูกกำหนดโดยใช้โค้ด จึงสามารถนำโครงสร้างพื้นฐานและบริการต่างๆ ไปปรับใช้ได้อย่างรวดเร็วโดยใช้รูปแบบที่เป็นมาตรฐาน อัปเดตด้วยโปรแกรมแก้ไขและเวอร์ชันล่าสุด หรือผลิตซ้ำในวิธีการที่ทำซ้ำได้
- การตรวจสอบและการบันทึก องค์กรจะตรวจสอบตัวชี้วัดและบันทึกต่างๆ เพื่อดูว่าประสิทธิภาพของแอปพลิเคชันและโครงสร้างพื้นฐานมีผลกระทบต่อประสบการณ์ของผู้ใช้ผลิตภัณฑ์ในขั้นปลายอย่างไร การรวบรวม จำแนกประเภท และวิเคราะห์ข้อมูลและบันทึกที่สร้างโดยแอปพลิเคชันและโครงสร้างพื้นฐาน จะทำให้องค์กรสามารถเข้าใจว่าการเปลี่ยนแปลงหรืออัปเดตมีผลกระทบต่อใช้อย่างไร โดยทำให้มองเห็นข้อมูลเชิงลึกถึงสาเหตุหลักของปัญหาหรือการเปลี่ยนแปลงที่ไม่คาดคิด การตรวจสอบอย่างจริงจังมีความสำคัญเพิ่มขึ้นเรื่อยๆ ในขณะที่บริการต้องพร้อมใช้งาน 24 ชั่วโมงทุกวัน และในขณะที่แอปพลิเคชันและโครงสร้างพื้นฐานมีอัปเดตในความถี่เพิ่มขึ้น การสร้างข้อความแจ้งเตือนหรือการดำเนินการวิเคราะห์ข้อมูลดังกล่าวในทันทียังช่วยให้องค์กรตรวจสอบบริการในเชิงรุกมากขึ้นอีกด้วย
- การสื่อสารและการทำงานร่วมกัน ที่เพิ่มขึ้นในองค์กรเป็นประเด็นเชิงวัฒนธรรมข้อหนึ่งของ DevOps การใช้เครื่องมือ DevOps และการทำงานอัตโนมัติของกระบวนการส่งมอบซอฟต์แวร์ก่อให้เกิดการทำงานร่วมกันโดยนำลำดับการทำงานและความรับผิดชอบของฝ่ายพัฒนาและฝ่ายปฏิบัติการเข้ามาอยู่ด้วยกัน จากนั้น ทีมเหล่านี้ก็จะกำหนดบรรทัดฐานเชิงวัฒนธรรมที่เข้มแข็งเกี่ยวกับการแบ่งปันข้อมูลและการอำนวยความสะดวกในการสื่อสารผ่านการใช้แอปพลิเคชันการสนทนา ระบบติดตามปัญหาหรือโครงการ และ Wiki ซึ่งทำให้นักพัฒนาฝ่ายปฏิบัติการ และแม้แต่ทีมอื่นๆ เช่น การตลาดหรือการขาย ติดต่อสื่อสารระหว่างกันได้เร็วขึ้น โดยทำให้ทุกส่วนขององค์กรปรับตัวสอดคล้องกับเป้าหมายและโครงการได้มากขึ้น

CI/CD

เพื่อแก้ปัญหาในการ deploy code จึงเกิดเป็นแนวทาง CI/CD โดยทำงานตั้งแต่การ Plan, Code, Build, Test, Release, Deploy, Operate, Monitor หรือบางทีเรียกสั้นๆ ว่า Pipeline

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

ตัวอย่างขั้นตอนการพัฒนาระบบตามแนวทาง CI/CD

1. Developer เมื่อทำการพัฒนา feature เสร็จ จะทำการ build, test และ run บนเครื่องของตัวเอง (Local) เพื่อทำให้แน่ใจว่าระบบทำงานได้ถูกต้องและให้แน่ใจว่าสิ่งที่เปลี่ยนแปลงไม่กระทบส่วนอื่น ๆ

2. ทำการดึง source code ล่าสุดจาก Repository ของระบบ เพื่อตรวจสอบว่ามีการเปลี่ยนแปลงหรือไม่ถ้ามีการเปลี่ยนแปลงก็ให้ทำการรวมหรือ merge ที่เครื่องของ Developer 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 sit environment

4. เมื่อ source code ถูก deploy to sit environment แล้วจะ trigger ไปสั่งให้ run job automated testing ใน level ของเทสเคส ซึ่งเป็นชุดเทสเคสย่อย ๆ ไม่เยอะมากเฉพาะในส่วนของ feature code ที่ถูก deploy มาเท่านั้น

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

6. กรณีหลังจาก run test ผ่านทั้งหมดจะส่งต่อไปยังระบบ Continuous Delivery เพื่อ deploy to uat (staging) environment เมื่อ source code ถูก deploy to uat (staging) แล้ว จะ trigger ไปสั่งให้ run job automated testing ใน level ของเทสเคส regression test และ QA ก็ทำการทดสอบ Acceptance testing ไปด้วยพร้อมๆ กันที่ uat (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

ข้อดีของการพัฒนาระบบตามแนวทาง CI/CD

ลดระยะเวลาในการพัฒนา ลดต้นทุนในการพัฒนา รองรับการเปลี่ยนแปลงแก้ไขได้ง่าย ดูแลรักษาได้ง่าย เนื่องจากมีการดำเนินไปอย่างต่อเนื่อง เป็นไปอย่างอัตโนมัติและทราบผลลัพธ์ได้ทันที ทำให้เราสามารถส่งมอบแอพฯ รุ่นใหม่ให้กับลูกค้าได้ไวขึ้น

หากองค์กรต้องการนำเอา DevOps และ CI/CD เข้ามาเป็นส่วนเสริมในกระบวนการพัฒนาซอฟต์แวร์จะต้องทำอย่างไรบ้าง

ในแต่ละรอบของการทำ CI/CD อาจประกอบด้วย Job ต่างๆ เช่น การ Compile การ Build การ Test และการ Deploy ซึ่งเราเรียก Job ทั้งหมดที่ทำงานว่า Pipeline

Pipeline จะถูกกระตุ้นให้ทำงานเมื่อสมาชิกในทีมมีการ Push Source Code ไปยัง Remote Repository ซึ่ง Software ที่จัดการ Git Repository อย่างเช่น GitLab นั้น มีเครื่องมือในการเขียน Script ทำ CI/CD แบบ Built-in ที่ชื่อว่า GitLab CI โดยไม่ต้องใช้โปรแกรมอื่น อย่างเช่น Jenkins ดังนั้น GitLab จึงเป็น Solution อย่างง่ายที่เราจะนำมา Implement CI/CD ในบทความนี้

ซึ่งการ Config GitLab CI ให้สามารถ Integrate และ Deploy Software ได้แบบอัตโนมัติ เป็นหน้าที่ของ DevOps Engineer

หลัก ๆ แล้ว DevOps Engineer จะเป็นคนเขียน CI/CD Pipeline Script เขียน Dockerfile และ docker-compose.yml วางสภาพแวดล้อมในการพัฒนาโปรแกรม รวมทั้งคอยดูแล Infrastructure และ Config Cluster (เช่น Docker Swarm/Kubernetes) เป็นต้น

สำหรับ Software ที่ Deploy แล้ว DevOps Engineer จะเป็นผู้ Monitor ซึ่งเมื่อพบปัญหาเขาจะต้องรีบแจ้ง Developer ให้หาทางแก้ไข

ดังนั้นหน้าที่ของ DevOps Engineer คือการ Support การทำงานของ Developer ในทำนองเดียวกันกับ System Admin แต่มีขอบเขตกว้างขวางกว่าการทำหน้าที่ Operation

อ้างอิง

ณัฐโชติ พรหมฤทธิ์. (2020). การทำ CI/CD Pipeline ด้วย GitLab Server ของตัวเอง สำหรับ DevOps Team. สืบค้นเมื่อ 21 กุมภาพันธ์ 2020, จาก https://blog.pjjop.org/ci-cd-pipeline-with-gitlab-server-for-devops-team/

Pariwat Saknimitwong. (2017). Learn DevOps ตอนที่ 2 : DevOps คืออะไร ?. สืบค้นเมื่อ 21 กุมภาพันธ์ 2020, จาก https://medium.com/@pariwat_s/learn-devops-ตอนที่-2-devops-คืออะไร-18ac48d73625

SoftMelt. (2011). การพัฒนาระบบตามแนวทาง CI/CD และ DevOps คืออะไร?. สืบค้นเมื่อ 21 กุมภาพันธ์ 2020, จาก https://www.softmelt.com/article.php?id=664