Term Project Sprint 3

Continuous Integration/Continuous Deployment (CI/CD) Pipeline

จัดทำโดย

กลุ่ม หมูกรอบหมด

กชพร	มีณรงค์	6509650195
กรกฤต	พงศ์ปัญจศิล	6509650203
จิรัฏญา	ทั่งจันทร์	6509650278
เตชิต	จันทร์ลี	6509650419

เสนอ

ผู้ช่วยศาสตราจารย์ ดร.ประภาพร รัตนธำรง

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

คพ.360 หัวข้อเลือกสรรด้านวิศวกรรมซอฟต์แวร์

คณะวิทยาศาสตร์และเทคโนโลยี สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์

Repository Link

ผลการทดสอบ

Method	Integrate/Build Code	Deploy Code	Total
Manual	10-15 นาที	10-15 นาที	20-30 นาที
Semi-Automate	5-10 นาที	10-15 นาที	15-25 นาที
Automate (CI/CD)	5 นาที	5 นาที	5-10 นาที

1. Manual

วิธีนี้ต้องดำเนินการทุกขั้นตอนด้วยตัวเอง ตั้งแต่การเตรียมโค้ด ทดสอบโค้ด ติดตั้งเซิร์ฟเวอร์ และนำเว็บแอปพลิเคชันขึ้นใช้งาน

- **ส่วน Integrate/Build Code**

เนื่องจากต้องรันคำสั่งเองเพื่อดาวน์โหลด Dependency และเพื่อสร้างไฟล์สำหรับใช้งานจริง ขั้นตอนนี้จึงใช้เวลาประมาณ 10-15 นาที

- **ส่วน Deploy Code ขึ้น Server**

หลังจาก Build เสร็จ เราต้องเชื่อมต่อกับเซิร์ฟเวอร์ EC2 ผ่าน SSH และอัปโหลดไฟล์โค้ดจากนั้นจึงติดตั้งซอฟต์แวร์ และตั้งค่าเซิร์ฟเวอร์ ขั้นตอนนี้ใช้เวลาประมาณ 10-15 นาที

รวมใช้เวลาประมาณ 20-30 นาที

ข้อดี

- การ Manual ทำให้เราเข้าใจกระบวนการตั้งค่าอย่างละเอียด
- สามารถแก้ไขได้ทันทีหากพบปัญหา

ข้อเสีย

- ใช้เวลานาน
- มีโอกาสเกิดข้อผิดพลาดสูง

2. Semi-Automate

วิธีนี้ใช้ Bash Script เข้ามาช่วยลดขั้นตอนที่ต้อง Manual เช่น การติดตั้ง Dependency หรือการรันเว็บแอปพลิเคชัน แต่ยังงต้อง Manual บางส่วน

- **ส่วน Integrate/Build Code**

Bash Script จะช่วยรันคำสั่งสำคัญ

โดยเราสามารถรันสคริปต์เพียงคำสั่งเดียวเพื่อดำเนินการทุกอย่างในขั้นตอนนี้ ใช้เวลาประมาณ 5-10 นาที

- **ส่วน Deploy Code ขึ้น Server**

เรายังต้องเชื่อมต่อเซิร์ฟเวอร์ EC2 และเรียกใช้งาน Bash Script เพื่ออัปโหลดไฟล์และติดตั้งซอฟต์แวร์ ขั้นตอนนี้ใช้เวลาประมาณ 10-15 นาที

รวมใช้เวลาประมาณ 15-25 นาที

ข้อดี

- ลดข้อผิดพลาดจากการพิมพ์คำสั่งผิด
- ประหยัดเวลามากกว่าวิธี Manual

ข้อเสีย

- ยังต้อง Manual บางขั้นตอน

3. การ Deploy แบบ Automate

วิธีนี้ใช้ระบบ CI/CD Pipeline ในการจัดการทุกขั้นตอนอย่างอัตโนมัติ ตั้งแต่การทดสอบโค้ด การ Build ไปจนถึงการนำโค้ดขึ้นเซิร์ฟเวอร์

- **ส่วน Integrate/Build Code**

CI/CD Pipeline จะรันคำสั่งทั้งหมดโดยอัตโนมัติ เช่น การติดตั้ง Dependency และการ Build เว็บแอปพลิเคชัน ใช้เวลาประมาณ 5 นาที

- **ส่วน Deploy Code ขึ้น Server**

หลังจาก Build เสร็จ ระบบจะอัปโหลดไฟล์และติดตั้งซอฟต์แวร์โดยอัตโนมัติผ่าน Pipeline ใช้เวลาประมาณ 5 นาที

รวมใช้เวลาประมาณ 10 นาที

ข้อดี

- รวดเร็วและแม่นยำ
- ลดความผิดพลาดเกือบทั้งหมด
- เหมาะกับการทำงานซ้ำ ๆ

ข้อเสีย

- ต้องมีความเชี่ยวชาญในการตั้งค่า Pipeline
- ระบบค่อนข้างซับซ้อน

สรุป

จากการเปรียบเทียบทั้ง 3 วิธีแล้ว แสดงให้เห็นว่าวิธี Manual จะใช้เวลาดำเนินการมากที่สุด แต่เหมาะสำหรับผู้เริ่มต้นหรือผู้ที่ต้องการเรียนรู้กระบวนการอย่างละเอียด ในขณะที่ Semi-Automate และ Automate เหมาะสำหรับการลดเวลาและเพิ่มประสิทธิภาพ โดย Automate มีความสะดวกและรวดเร็วที่สุด สำหรับการเลือกใช้งานนั้น หากต้องการความรวดเร็วและความแม่นยำ Automate จะตอบโจทย์มากที่สุด แต่หากเป็นงานเล็กๆ Manual อาจเป็นตัวเลือกที่เหมาะสมกว่า

การอภิปรายเกี่ยวกับประโยชน์ของ CI/CD

ประโยชน์ของ CI/CD

1. **ลดเวลาในการ Deploy**

   จากตารางเปรียบเทียบจะเห็นได้ว่า **CI/CD** ลดเวลารวมจาก **~20-30 นาที (Manual)** เหลือเพียง **~5-10 นาที (Automate)** ซึ่งเป็นการลดเวลาได้ถึง 70-80% โดย:

   - ลดข้อผิดพลาดจากการติดตั้งและตั้งค่าด้วยตนเอง (Human Error) เช่น การพลาดขั้นตอนหรือการใช้คำสั่งที่ผิดพลาด
   - ทำให้การ Integrate/Build/Deploy รวดเร็วและเหมาะสมในโครงการที่มีการอัปเดตโค้ดบ่อย

2. **เพิ่มประสิทธิภาพของทีมพัฒนา**

   - ทีมพัฒนาสามารถโฟกัสไปที่การปรับปรุงฟีเจอร์หรือแก้ไข Bug แทนที่จะเสียเวลาในการตั้งค่าติดตั้งเซิร์ฟเวอร์หรือ Deploy แบบ Manual
   - กระบวนการอัตโนมัติช่วยให้ทีมทำงานพร้อมกันได้มากขึ้น เช่น ทดสอบโค้ดในขณะที่อีกทีมพัฒนาฟีเจอร์ใหม่

3. **ปรับปรุงคุณภาพของแอปพลิเคชัน**

   - การทดสอบอัตโนมัติ (Automated Tests) ที่เป็นส่วนหนึ่งของ CI/CD Pipeline ช่วยจับข้อผิดพลาดก่อน Deploy
   - ลดความเสี่ยงของปัญหาใน Production Environment เพราะ Pipeline สามารถทดสอบทั้ง Unit Test, Integration โดยอัตโนมัติ

ควรทำ CI/CD หรือไม่?

ในมุมมองของทีมคิดว่า

1. **ควรทำ CI/CD** โดยเฉพาะในโครงการที่ต้องการความรวดเร็ว เช่น
   - โครงการที่ใช้ Agile Development หรือมีการอัปเดตฟีเจอร์บ่อย
   - โครงการที่มีทีมขนาดใหญ่ที่ต้องการกระบวนการพัฒนาแบบ Collaborative

2. **เหตุผลที่ควรทำ CI/CD**
    - แม้การสร้าง Pipeline อาจใช้เวลาในช่วงแรก แต่เมื่อดำเนินการแล้ว จะช่วยประหยัดเวลาในระยะยาว
    - ลดข้อผิดพลาดที่เกิดจากมนุษย์ (Human Error) และเพิ่มความเสถียรให้กระบวนการพัฒนา
    - เพิ่มความมั่นใจให้ทีมพัฒนา ว่าฟีเจอร์ที่อัปเดตจะไม่ส่งผลกระทบต่อระบบหลัก
3. **ข้อควรระวัง**
    - ในโครงการขนาดเล็กที่ไม่ค่อยมีการอัปเดตโค้ด การสร้าง Pipeline อาจใช้ทรัพยากรเกินความจำเป็น
    - ควรประเมินความคุ้มค่าก่อนลงทุนในระบบ CI/CD

สรุป

CI/CD เป็นเครื่องมือสำคัญในกระบวนการพัฒนาซอฟต์แวร์ที่ต้องการความรวดเร็วและความถูกต้องแม่นยำ การนำระบบนี้มาใช้งานช่วยประหยัดเวลา เพิ่มความเสถียร และเพิ่มความมั่นใจในผลลัพธ์ของโครงการ อย่างไรก็ตาม ควรคำนึงถึงขนาดและความซับซ้อนของโครงการ เพื่อให้การตัดสินใจใช้งาน CI/CD เหมาะสมและเกิดประโยชน์สูงสุดในระยะยาว