รายงานวิจัยฉบับสมบูรณ์
โครงการการประยุกต์ใช้ Azure Devops เพื่อปรับปรุงกระบวนการพัฒนาระบบสารสนเทศของสำนักคอมพิวเตอร์

คณิต ธุระ

โครงการวิจัย ประเภทงบประมาณเงินรายได้เพื่อส่งเสริมการวิจัยและพัฒนา
นวัตกรรมประจำปีงบประมาณ พ.ศ. 2563 สำนักคอมพิวเตอร์
มหาวิทยาลัยบูรพา

สัญญาเลขที่ 001/2563

รายงานวิจัยฉบับสมบูรณ์
โครงการการประยุกต์ใช้ Azure Devops เพื่อปรับปรุงกระบวนการพัฒนาระบบสารสนo9iiเทศของสำนักคอมพิวเตอร์

คณิต ธุระ

มิถุนายน 2564

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับทุนสนับสนุนการวิจัยจากงบประมาณเงินรายได้ เพื่อส่งเสริมการวิจัยและการพัฒนานวัตกรรมประจำปีงบประมาณ พ.ศ. 2563 สำนักคอมพิวเตอร์ มหาวิทยาลัยบูรพา เลขที่สัญญา 001/2563

ขอขอบพระคุณ อาจารย์วิทวัส พันธุมจินดา ผู้อำนวยการสำนักคอมพิวเตอร์ ผู้ช่วยศาสตราจารย์ ดร.จักริน สุขสวัสดิ์ชน รองผู้อำนวยการสำนักคอมพิวเตอร์ที่ให้การสนับสนุนและเสียสละเวลาในการตรวจสอบงานวิจัย ช่วยเหลือ และให้ข้อเสนอแนะที่เป็นประโยชน์ต่อการทำงานวิจัยครั้งนี้

คณิต ธุระ

บทคัดย่อ

ชื่อเรื่อง : โครงการการประยุกต์ใช้ Azure Devops เพื่อปรับปรุงกระบวนการพัฒนาระบบสารสนเทศของสำนักคอมพิวเตอร์

ผู้วิจัย : นายคณิต ธุระ

ปีที่พิมพ์ : 2563

แหล่งทุน : ทุนสนับสนุนการวิจัยจากงบประมาณเงินรายได้ เพื่อส่งเสริมการวิจัยและการพัฒนานวัตกรรมประจำปีงบประมาณ พ.ศ. 2563 สำนักคอมพิวเตอร์ มหาวิทยาลัยบูรพา

ปัจจุบันการเปลี่ยนแปลงทางด้านเทคโนโลยีอย่างรวดเร็วส่งผลกระทบต่อภาคธุรกิจอย่างมาก ทำให้กระบวนการพัฒนาซอฟต์แวร์และกระบวนการส่งมอบซอฟต์แวร์ที่สนับสนุนธุรกิจนั้นจำเป็นต้องรวดเร็วและเกิดขึ้นอย่างต่อเนื่องมากขึ้นเพื่อตอบสนองได้ทันต่อความต้องการ แม้ว่าธุรกิจจะมีการใช้กระบวนการพัฒนาซอฟต์แวร์ในรูปแบบแอจไจล์มาทำงานมากขึ้น แต่กระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์มุ่งเน้นเพียงกระบวนงานเพื่อตอบสนองความต้องการฝั่งธุรกิจ ซึ่งไม่ได้ครอบคลุมไปถึงความต่อเนื่องของกระบวนการพัฒนาซอฟต์แวร์ที่ส่งมอบไปยังผู้ใช้งาน โดยปัจจุบันมีแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps เข้ามาใช้งานร่วมกับกระบวนการพัฒนาซอฟต์แวร์ด้วยการใช้เครื่องมือมาความสร้างความต่อเนื่องในการพัฒนาซอฟต์แวร์ให้เป็นอัตโนมัติมากขึ้น เพื่อเข้ามาช่วยการทำงานระหว่างทีมพัฒนาซอฟต์แวร์กับทีมปฏิบัติการ ในการปรับปรุงกระบวนงานให้ราบรื่นมากขึ้น โดยผลการวิจัยสรุปได้ดังนี้

(1) จำนวนขั้นตอนการพัฒนาระบบนั้นยังคงเท่ากันแต่กิจกรรมต่าง ๆ จะปรับเปลี่ยนใหม่ทั้งหมดเพื่อให้สอดคล้องกับหลักการของ devops

(2) ส่งมอบงานได้อย่างอย่างรวดเร็วและมีประสิทธิภาพ สามารถตรวจสอบคุณภาพและส่งซอร์สโค้ดไปยังเครื่องทดสอบและเครื่องส่งมอบโดยอัตโนมัติ

(3) มีความน่าเชื่อถือของซอฟต์แวร์ โดยใช้เครื่องช่วยค้นหาช่องโหว่ต่างๆ เพื่อลดช่องโหว่ต่าง ๆ ก่อนส่งมอบให้งานให้กับลูกค้า

(4) ลดขั้นตอนการดำเนินงานและจัดการโครงสร้างพื้นฐานอัตโนมัติทั้งหมด เพื่อลดข้อผิดพลาดในการทำงานจากคน

Abstract

Title : Applying of Azure Devops to improve the information system development process of the Computer Center
Name : Kanit Tura
Year : 2020
Funding Resource : Research and Development Fund of Computer Center Burapha University, Year 2020

Rapidly change in technology has tremendous impact on most organizations. Software development process and the delivery process of software that supports their business requirements need to be faster and more continuous to respond as needed. Many businesses have adopted Agile concept. But it only focuses on the business process which does not cover the continuity of the software development process delivered to the user. Currently, the concept of DevOps term is introduced to integrate with the software development process. With the DevOps tool chain, the creation of continuity in software development is more automated. It facilitates working between the Development Team and the Operations Team to improve the continuously development process.

Research results were as follows:

(1) The number of development steps remains the same, but activities will be completely modified to reflect devops principles.

(2) deliver work quickly and efficiently able to inspect the quality and automatically send the source code to the tester and delivery.

(3) the reliability of the software by using a tool to find vulnerabilities to reduce the vulnerabilities before delivering to the customer.

(4) Reduces operational procedures and manages all automated infrastructure. To reduce errors in work from people.

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหาวิจัย

ปัจจุบันสภาพแวดล้อมของธุรกิจเปลี่ยนแปลงอย่างรวดเร็ว ทำให้ธุรกิจต้องเพิ่มความรวดเร็วในการส่งมอบงานให้แก่ลูกค้า ส่งผลต่อระเบียบวิธีการพัฒนาและกระบวนการส่งมอบซอฟต์แวร์ที่ต้องปรับตัวตาม เพื่อตอบสนองการเปลี่ยนแปลงของธุรกิจ แม้ว่าองค์กรต่าง ๆ จะเริ่มใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มากขึ้น ทำให้กระบวนการพัฒนาซอฟต์แวร์คล่องตัวและเป็นไปได้อย่างมาก แต่องค์กรธุรกิจยังคงประสบปัญหา เนื่องด้วยระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มุ่งเน้นกระบวนการเพื่อตอบสนองต่อการเปลี่ยนแปลงความต้องการ แต่ไม่ครอบคลุมความต่อเนื่องถึงการส่งมอบซอฟต์แวร์ไปยังผู้ใช้งาน จึงทำให้เกิดปัญหาคอขวดในกระบวนการส่งมอบซอฟต์แวร์ นอกจากนี้ยังเกิดปัญหาการทำงานร่วมกันระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการอีกด้วย ปัจจุบันจึงมีการนำเสนอแนวคิดการใช้เครื่องมือเพื่อสร้างความต่อเนื่องในการพัฒนาซอฟต์แวร์ทำให้มีความอัตโนมัติมากขึ้น โดยเครื่องมือจะเข้ามาช่วยการทำงานระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการปรับปรุงกระบวนการทำงานระหว่างกันให้ราบรื่นเรียกว่า DevOps เป็นคำที่ย่อมาจาก Development Operations เป็นเครื่องมือที่ช่วยเชื่อมกระบวนการพัฒนาโปรแกรม การส่งเอาขึ้นไป Deploy ให้ผู้ใช้เข้าถึงอย่างได้อย่างมีประสิทธิภาพ มีคุณภาพ ไปจนถึงการติดตามสถานะการทำงานของโปรแกรมเพื่อนำกลับมาพัฒนาปรับปรุงกระบวนการพัฒนาโปรแกรม ส่วน DevOps Engineer เปรียบได้กับ System Admin ในสมัยก่อน ที่ทำหน้าที่เอาโปรแกรมขึ้นไปรันบนเซิร์ฟเวอร์ให้ได้ แล้วคอยติดตามให้ใช้งานได้ปกติ แต่พัฒนากระบวนการโดยทำเรื่อง Automation เข้าไปด้วย คือทำให้กระบวนการทั้งหมดที่ต้องทำมือในสมัยก่อน ทำงานโดยอัตโนมัติได้ นอกจากนั้นยังลงไปช่วยนักพัฒนาทำงานด้วย ในส่วนของการวางสภาพแวดล้อมในการพัฒนาโปรแกรมที่ดีของ DevOps Process

DevOps นั้นมีรูปแบบการทำงานค่อนข้างจะตายตัวมีกระบวนการดังต่อไปนี้ Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor ซึ่งจะพยายามทำให้ Process ต่าง ๆ ทำงานไปได้โดยอัตโนมัติ ไม่ต้องมีคนคอยเข้าไปกด Deploy เองอีกต่อไป ยกเว้นในส่วนของการการวางแผน การพัฒนา และดูผลลัพธ์ ในส่วนนี้ยังจำเป็นต้องอาศัยสมองคนต่อไปอยู่

แนวคิด DevOps จะใช้คอเทนเนอร์ (Container) เป็นการจำลองเพื่อควบคุมสภาพแวดล้อมสำหรับการทำงานเฉพาะเซอร์วิส โดยจะบรรจุสภาพแวดล้อมนั้นลงในคอนเทนเนอร์ก่อให้เกิดการใช้ทรัพยากรเท่าที่จำเป็น การทำงานจะอยู่ในระดับเซอร์วิสเท่านั้น แตกต่างจากการจำลองเซิร์ฟเวอร์เสมือนที่ใช้สภาพแวดล้อมในระบบปฏิบัติการทั้งหมดเพื่อสร้างเซิร์ฟเวอร์หนึ่งเครื่องที่มีหลายๆ เซอร์วิสทำงานร่วมกัน

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อปรับปรุงกระบวนการการพัฒนาระบบสารสนเทศในสำนักคอมพิวเตอร์ให้มีประสิทธิภาพมากยิ่งขึ้น

2. เพื่อช่วยให้สามารถปรับปรุงรุ่นของซอฟต์แวร์ได้อย่างรวดเร็ว และลดขั้นตอนในการติดตั้งซอฟต์แวร์ที่เครื่องแม่ข่าย (Deployment)

1.3 ขอบเขตของการวิจัย

1. มีระบบ Version Control ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจคมีการสำรองซอร์สโค้ด สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่าง ๆของโปรเจคที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นถูกเขียนโดยใครบ้างก็สามารถทำได้

2. มีระบบ CI/CD (Continuous Integration, Continuous Delivery) เป็นกระบวนการในการทำงาน ตั้งแต่การ Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor

3. มีเครื่องมือช่วยตรวจสอบคุณภาพของซอร์สโค้ดช่วยหาข้อบกพร่องในซอร์สโค้ดไม่ว่าจะเป็น Bug ที่น่าจะเกิดขึ้น ช่องโหว่ทางด้านความปลอดภัยหรือซอร์สโค้ดที่จะเป็นปัญหาในอนาคต (Code Smell) และช่วยตรวจสอบว่าเขียนซอร์สโค้ดทดสอบครอบคลุมหรือดีแล้วหรือไม่ (Code coverage)

4. มีระบบเปรียบเทียบ Database Schemas ระหว่างเครื่อง Database Develop และเครื่อง Database Production เพื่อแสดงส่วนที่แตกต่างกันพร้อมทั้งสร้างไฟล์ SQL เพื่อนำเข้าไปยังเครื่อง Database Production ได้

5. มีระบบจำลองเพื่อควบคุมสภาพแวดล้อมสำหรับการทำงานเฉพาะเซอร์วิส โดยจะบรรจุสภาพแวดล้อมนั้นลงในคอนเทนเนอร์ (Container) ก่อให้เกิดการใช้ทรัพยากรเท่าที่จำเป็น การทำงานจะอยู่ในระดับเซอร์วิสเท่านั้น

6. มีระบบ Proxy ทำหน้าที่เป็น HTTP reverse proxy ทำงานร่วมกับด๊อกเกอร์ (Docker) ทำให้สามารถสร้างคอนเทนเนอร์โดยไม่ต้องกำหนด Port การใช้งาน

7. มีระบบติดตาม (monitor) สามารถเห็นความเคลื่อนไหวของ Application ทุกตัว โดยเฉพาะเวลาที่มีข้อผิดพลาดของโปรแกรมเกิดขึ้น ก็สามารถรู้ได้อย่างทันที

1.4 วิธีการประเมินผล

1. วิธีการประเมินผลให้เปรียบเทียบว่าจำนวนขั้นตอนการพัฒนาระบบแบบเดิมกับขั้นตอนใหม่

2. ประเมินผลการส่งมอบงานได้อย่างอย่างรวดเร็วและมีประสิทธิภาพระหว่างระบบงานเดิมและระบบงานใหม่

3. ประเมินผลว่าซอฟต์แวร์มีความเชื่อถือระหว่างระบบงานเดิมและระบบงานใหม่

4. ประเมินผลการลดขั้นตอนการดำเนินงานและจัดการโครงสร้างพื้นฐานระหว่างระบบงานเดิมและระบบงานใหม่

บทที่ 2
แนวคิด ทฤษฎี วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในการศึกษาเรื่อง “การประยุกต์ใช้ Azure Devops เพื่อปรับปรุงกระบวนการพัฒนาระบบสารสนเทศของสำนักคอมพิวเตอร์” ผู้วิจัยได้ทบทวนวรรณกรรม แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้องเพื่อนำมากำหนดเป็นกรอบในการศึกษาดังนี้

2.1 แนวความคิดวงจรการพัฒนาซอฟต์แวร์

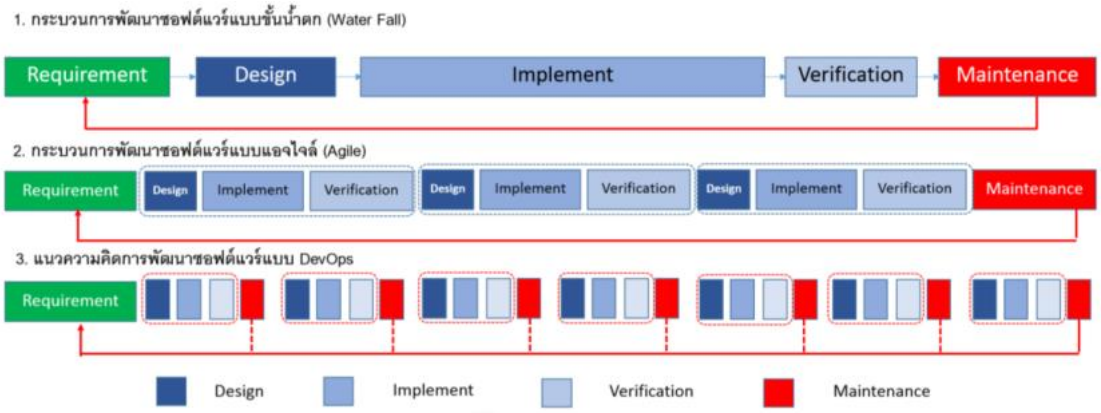
2.2 แนวคิดการพัฒนาซอฟต์แวร์แบบแอจไจล์

2.3 แนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps

2.4 แนวคิดการทำงานของเครื่องเสมือนกับคอนเทนเนอร์ (Container)

2.1 แนวความคิดวงจรการพัฒนาซอฟต์แวร์

กระบวนการพัฒนาซอฟต์แวร์ คือ ลำดับขั้นตอนการดำเนินการในการพัฒนาซอฟต์แวร์จนกระทั่งได้ซอฟต์แวร์ที่สำเร็จรูป ภายใต้ระยะเวลาที่กำหนดและได้ซอฟต์แวร์ที่มีคุณภาพตามมาตรฐานที่กำหนดไว้ (Kaur, 2015) ซึ่งเป็นที่รู้จักกันในนามของกระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall model) โดยกิจกรรมพื้นฐานของกระบวนการพัฒนาซอฟต์แวร์ประกอบด้วย 5 ขั้นตอนพื้นฐาน ดังนี้ 1) ขั้นตอนการเก็บรวบรวมและวิเคราะห์ความต้องการ (requirement) ซึ่งจะช่วยให้เข้าใจปัญหามากขึ้น 2) ขั้นตอนการออกแบบ (design) เป็นขั้นตอนการวางแผนเพื่อแก้ไขปัญหา 3) ขั้นพัฒนาซอฟต์แวร์ (implement) การพัฒนาซอฟต์แวร์ด้วยการเขียนโปรแกรม 4) ขั้นตอนการทดสอบโปรแกรม (verification) เป็นการทดสอบผลลัพธ์ที่ได้จากการเขียนโปรแกรม และ 5) ขั้นตอนการบำรุงรักษา (maintenance) เป็นขั้นตอนการใช้งานจริงและบำรุงรักษาซอฟต์แวร์ ดังภาพที่ 2.1 หมายเลข 1 ด้วยลักษณะของแนวคิดแบบขั้นน้ำตกที่เป็นลำดับขั้นตอนทำให้ง่ายต่อการวางแผน แต่ยากต่อการเปลี่ยนแปลง รวมทั้งยังส่งผลถึงความเสี่ยงต่อการล้มเหลวของโครงการ (Cois, Yankel, &Connell, 2014) จึงเกิดกระบวนการพัฒนาซอฟต์แวร์รูปแบบใหม่ขึ้นมาเพื่อปรับปรุงข้อเสียดังกล่าว



ภาพที่ 2.1 แสดงการเปรียบเทียบระหว่าง วงจรการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall model) กระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์ และแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps สนิธพรรณ จิตตั้งสมบูรณ์ (2559)

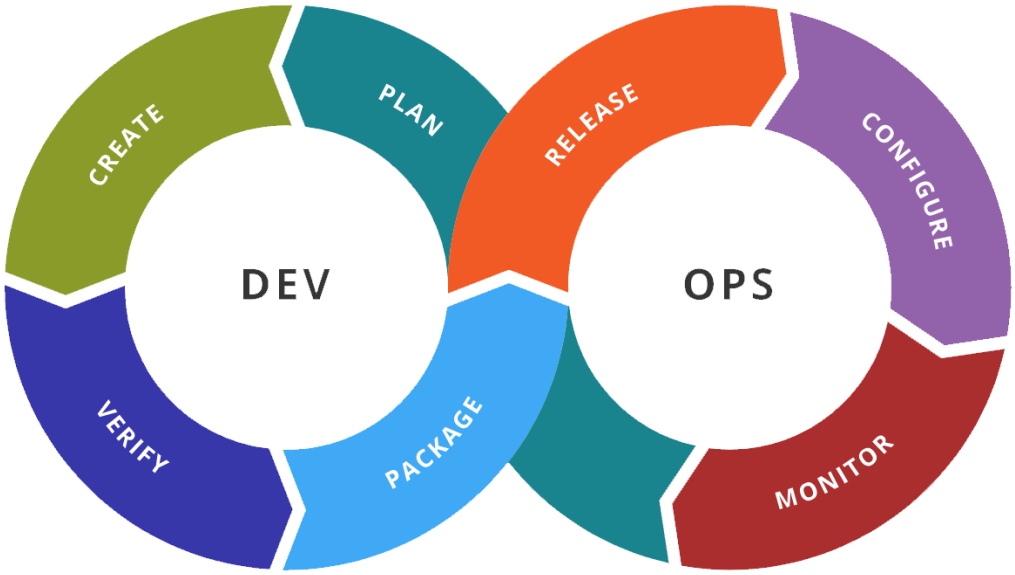
2.2 กระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์

เป็นกระบวนการพัฒนาซอฟต์แวร์ที่แก้ไขข้อบกพร่องของกระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก โดยนำกระบวนงานมาแบ่งเป็นส่วนเล็กเพื่อการทำรอบซ้ำ ๆ และเพิ่มจำนวนกาพัฒนาแต่ละรอบให้มากขึ้น ดังภาพที่ 2.1 หมายเลข 2 และเน้นถึงการติดต่อสื่อสารระหว่างผู้มีส่วนได้ส่วนเสีย การปรับปรุงความต้องการ และวางแผนงาน เพื่อให้สามารถส่งมอบซอฟต์แวร์ไปยังลูกค้าให้เร็วที่สุด (Schaller, 2016) โดยผู้ร่วมโครงการเชื่อว่ากระบวนการออกแบบต้องพร้อมที่จะปรับปรุงอยู่เสมอ ด้วยข้อมูลใหม่ที่มีประสิทธิภาพและช่วยลดความเสี่ยงในโครงการที่จะเกิดขึ้น รวมถึงเพิ่มความสำคัญของผลลัพธ์ที่จะได้จากโครงการ (Cois et al., 2014) ถึงแม้ว่ากระบวนการพัฒนาซอฟต์แวร์แอจไจล์จะช่วยให้การส่งมอบไปสู่ลูกค้าได้เร็วขึ้น แต่ก็ยังพบข้อจำกัดในกระบวนการบางอย่างที่ไม่สอดคล้องกับรูปแบบธุรกิจ เช่น การติดตามปัญหาและแก้ปัญหาที่ผู้ใช้รายงานให้ทราบ การเพิ่มความสามารถของซอฟต์แวร์จากฝ่ายบริหาร การส่งมอบที่ต้องปรับแต่งตลอดเวลา ซึ่งข้อจำกัดเหล่านี้ยังขาดกระบวนการที่ดีและความน่าเชื่อถือในการติดตามงาน (พงศกร ภูแสนคำ, 2559)

2.3 แนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps

แนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps พัฒนาอยู่บนพื้นฐานกระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์ โดยช่วยเสริมช่องว่างของแอจไจล์ซึ่งมุ่งเน้นเพียงด้านธุรกิจ เป็นการประสานความสัมพันธ์ระหว่างผู้จัดการโครงการ (project manager) กับนักพัฒนาซอฟต์แวร์ (developer) โดย DevOps จะช่วยให้ทีมปฏิบัติการ (operation team) สามารถส่งมอบซอฟต์แวร์ไปยังผู้ใช้งาน (user) ได้อย่างต่อเนื่องมากขึ้น (Mueller, 2016; Deshpande, 2016;Mohamed,

2016) ดังภาพที่ 2.1 หมายเลข 3 ทั้งนี้แนวความคิดที่อยู่เบื้องหลังของ DevOps คือการนำไปสู่ความร่วมมือที่ดีระหว่างทั้งสองทีม ประกอบด้วย ทีมนักพัฒนาซอฟต์แวร์และทีมปฏิบัติการ เพื่อให้แต่ละทีมสามารถดำเนินงานร่วมกันได้อย่างราบรื่น ทำให้กระบวนการพัฒนาซอฟต์แวร์เกิดความต่อเนื่องมากขึ้นด้วยความเป็นอัตโนมัติ ส่งผลให้ซอฟต์แวร์มีคุณภาพ และรองรับการเปลี่ยนแปลงได้อย่างรวดเร็วตามความต้องการ ก่อให้เกิดความน่าเชื่อถือในการส่งมอบซอฟต์แวร์ (delivery) (Babar, Lapouchnian, & Yu, 2011) รวมถึงการเตรียมซอฟต์แวร์เพื่อนำไปสู่การใช้งานจริง (deployment) (Fitzgerald & Stol, 2014) ซึ่งกิจกรรมที่เกิดขึ้นจากแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps จะมีลักษณะต่อเนื่องเป็นลูป (loop) ดังภาพที่ 2.2 โดยจะเห็นว่ากระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์ จะมีเพียงส่วนซ้ายซึ่งจะสนับสนุนความคล่องตัวในการทำงานด้านการพัฒนาซอฟต์แวร์ ที่รองรับการเปลี่ยนแปลงทางความต้องการ การพัฒนาซอฟต์แวร์ และการทดสอบในเชิงโปรแกรม แต่แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะเริ่มตั้งแต่ฝั่งซ้ายที่รวมกระบวนการพัฒนาซอฟต์แวร์แอจไจล์และให้ความสำคัญต่อเนื่องไปถึงการเตรียมสภาพแวดล้อมต่าง ๆ เช่น UAT, Production เพื่อนำไปสู่การใช้งานจริง การปรับแต่งค่าที่เหมาะสม รวมถึงติดตามผลการดำเนินงาน (Dingsøy & Lassenius, 2016)



ภาพที่ 2.2 แสดงแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps
จาก https://buffohero.com/development-concept/devops/, โดย Buffohero, 2019.

(Gartner, 2016) ได้บรรยายถึง 7 กิจกรรมหลักของ DevOps ที่จะสนับสนุนการทำงานที่เชื่อมโยงถึงกัน โดยแต่ละกิจกรรมที่เกิดขึ้น ไม่ได้แยกเป็นตามลำดับขั้นตอน แต่กิจกรรมเหล่านี้จะคาบเกี่ยวกันด้วยโครงการและทีมงาน ทั้งนี้โครงสร้างที่เกิดขึ้นจะไม่กระทบต่อโครงสร้างเดิมขององค์กร ซึ่งรายละเอียดของกระบวนการ (Higgins, 2016) ประกอบด้วย

กิจกรรมการวางแผน (plan) เป็นการกำหนดกิจกรรมที่เกิดขึ้นทั้งหมดตามความต้องการ มีการกำหนดมาตรวัดในการพัฒนาซอฟต์แวร์ การจัดลำดับความสำคัญของความต้องการทั้งความต้องการเพิ่มเติมและความต้องการใหม่ การวางแผนรักษาความปลอดภัย และแผนการส่งมอบ

กิจกรรมการสร้าง (create) เป็นกิจกรรมการพัฒนาซอฟต์แวร์ ตั้งแต่การออกแบบ เขียนโปรแกรม, บิ้วด์ (Build), การทดสอบระดับหน่วยฟังก์ชั่น และการบริหารจัดการการส่งมอบ

กิจกรรมการตรวจสอบ (verify) เป็นกิจกรรมการประกันคุณภาพ ที่รวมการทดสอบหลากหลายรูปแบบ เช่น การทดสอบในมุมมองผู้ใช้งาน การทดสอบเชิงถดถอย เป็นการทดสอบโดยใช้ข้อมูลทดสอบชุดเดิม เพื่อให้มั่นใจว่าการเปลี่ยนแปลงที่เกิดขึ้นจะไม่ส่งผลกระทบต่อการทำงานเดิม และ การทดสอบสมรรถนะของระบบในด้านต่าง ๆ เช่น ระยะเวลาในการตอบสนองการทำงาน (พรภัทรา ภัทรจารี,2551)

กิจกรรมการเตรียมก่อนการใช้งานจริง (pre-prod) หรือที่เรียกว่า Preproduction เป็นกิจกรรมที่เกิดขึ้นเมื่อซอฟต์แวร์พร้อมจะนำไปส่งมอบให้ใช้งาน ได้แก่ การทำ staging เป็นการเตรียมสภาพแวดล้อมของเซิร์ฟเวอร์ ให้คล้ายกับสภาพแวดล้อมที่ใช้งานจริง (production server) มากที่สุด เพื่อเป็นการทดสอบครั้งสุดท้าย

กิจกรรมการส่งมอบ (release) เป็นกิจกรรมที่พร้อมจะส่งมอบซอฟต์แวร์ไปยังสภาพแวดล้อมที่ใช้งานจริง รวมถึงเตรียมการกู้คืนระบบหากเกิดความล้มเหลวในการส่งมอบกิจกรรมการปรับแต่งเป็นกิจกรรมที่เกิดขึ้นได้ตลอดในแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps รวมถึงการปรับแต่งฮาร์ดแวร์และซอฟต์แวร์ให้เหมาะสมกับการทำงาน

กิจกรรมการติดตามผล (monitor) เป็นกิจกรรมที่มุ่งเน้นไปยังการดูแลการทำงานของซอฟต์แวร์บนสภาพแวดล้อมที่ใช้งานจริง (production environment) ซึ่งรวมไปถึงการวัดประสิทธิภาพการทำงาน ความพร้อมในการให้บริการ และความต้องการเพิ่มเติมอื่น ๆ เช่น การติดตามพฤติกรรมการใช้งานของผู้ใช้ โดยข้อเสนอแนะที่เกิดขึ้นเหล่านี้จะถูกส่งกลับมายังกิจกรรมวางแผนอีกครั้ง

França, Junior, and Travassos (2016) ได้ทำการศึกษาคุณลักษณะของ DevOps (characteristic of DevOps) และรวบรวมไว้ ประกอบด้วยหลักการ วิธีปฏิบัติ และการบริการ ดังนี้

2.3.1 หลักการของแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps คือ แนวความคิดและคุณค่าที่ได้จาก DevOps ประกอบด้วย

ความเป็นอัตโนมัติ (automation) เป็นหลักการที่สำคัญของ DevOps การสร้างความเป็นอัตโนมัติด้วยการปรับลดขั้นตอนที่ไม่จำเป็น เพื่อให้เกิดซอฟต์แวร์ที่มีคุณภาพ

การประกันคุณภาพ (quality assurance) คุณภาพที่เกิดจากการทำงานร่วมกันระหว่างทีมนักพัฒนาซอฟต์แวร์และทีมปฏิบัติการ ก่อให้เกิดเป็นผลิตภัณฑ์ที่มีคุณภาพ โดยการเชื่อมโยงความแตกต่างระหว่างทีม ด้วยกิจกรรมที่มีประสิทธิภาพและวิธีการที่น่าเชื่อถือ

การปรับลดขั้นตอนที่ไม่จำเป็น (leanness) เป็นวิธีปฏิบัติที่อยู่บนพื้นฐานของแนวความคิดแบบลีน (Lean thinking) โดยใช้ข้อมูลเป็นตัวขับเคลื่อนการทำงาน การตัดสินใจ และการเลือกเฉพาะกระบวนการที่ก่อให้เกิดประโยชน์เข้ามาใช้งาน ทำให้กระบวนการเหล่านี้นำไปสู่การปรับปรุงทีละส่วนแบบมีคุณภาพ (Morris, 2016) จึงส่งผลให้เกิดข้อเสนอแนะกลับมาได้อย่างรวดเร็ว

การแบ่งปัน (sharing) เป็นการแลกเปลี่ยนองค์ความรู้ภายในองค์กรเพื่อเกิดวิธีปฏิบัติที่ดีที่สุดภายในบริบทของ DevOps ส่งผลให้บุคคลภายในทีมนำความรู้ความสามารถมาช่วยสนับสนุนการทำงานให้ราบรื่น

การวัดผล (measurement) เป็นตัวประเมินผลกระบวนการพัฒนาซอฟต์แวร์ เช่น การวัดผลความสำเร็จในการส่งมอบซอฟต์แวร์ อัตราการเกิดข้อผิดพลาด จำนวนข้อผิดพลาดที่เกิดขึ้น (Novak, 2016) ดังตารางที่ 2.1 ซึ่งแสดงให้เห็นถึงความแตกต่างโดยเปรียบเทียบกับกระบวนการพัฒนาซอฟต์แวร์แบบ Waterfall, Agile และ Devops

ตารางที่ 2.1 แสดงการเปรียบเทียบผลลัพธ์การดำเนินงานของ DevOps continuous delivery กับกระบวนการพัฒนาซอฟต์แวร์อื่น ๆ

เงื่อนไข	Waterfall	Agile	Devops
อัตราการส่งมอบซอฟต์แวร์โดยเฉลี่ยต่อเดือน	ทุกๆ 6 เดือน	ทุกๆ 3 สัปดาห์	ทุกวัน
จำนวนข้อผิดพลาด	10	8	2
หน่วยเวลาที่ใช้ในการเปลี่ยนแปลง	เดือน	วัน	นาที
อัตราเฉลี่ยระยะเวลาในความล้มเหลว	6 เดือน	16 ชั่วโมง	4 ชั่วโมง
ระยะเวลาในการแก้ไขความล้มเหลว	6 เดือน	24 ชั่วโมง	6 ชั่วโมง
ระยะเวลาที่ขัดข้อง	X	น้อยกว่า 5X	น้อยกว่า 10X
ทรัพยากรที่เกิดขึ้นต่อหนึ่งหน่วยเวลา	X	3X	7X

ตารางที่ 2.1 แสดงการเปรียบเทียบผลลัพธ์การดำเนินงานของ DevOps continuous delivery กับกระบวนการพัฒนาซอฟต์แวร์อื่น ๆ (ต่อ)

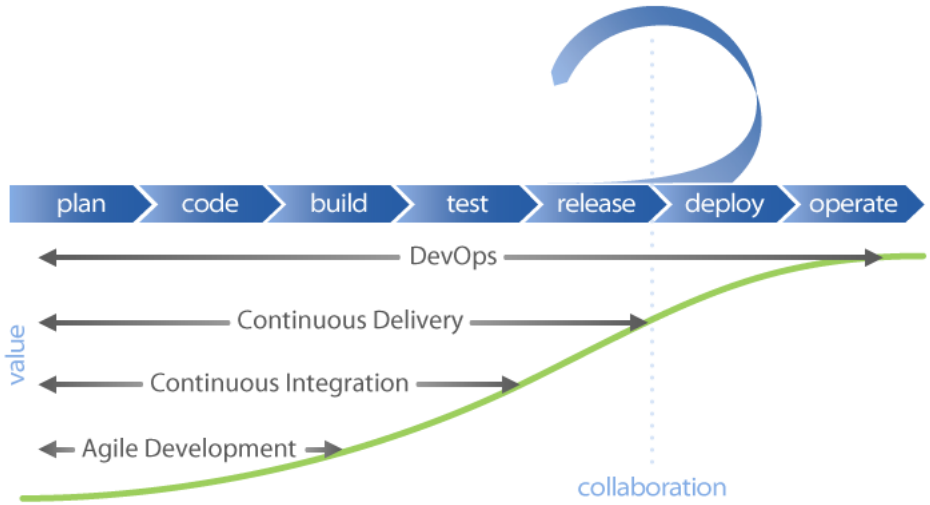
เงื่อนไข	Waterfall	Agile	Devops
การเปลี่ยนแปลง	X	5X	14X
อัตราความสำเร็จในการเปลี่ยนแปลง	X	80% X	99.5% X

หมายเหตุ. จาก “Software Release Management Evolution -Comparative Analysis across Agile and DevOps Continuous Delivery” โดย (Mohamed, 2016).

2.3.2 แนวทางปฏิบัติของแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps คือ การนำหลักการมาใช้ในทางปฏิบัติ ก่อให้เกิดกิจกรรมของทีมนักพัฒนาซอฟต์แวร์และทีมปฏิบัติการ ซึ่งวิธีปฏิบัติโดยทั่วไปจะประกอบด้วย 3 วิธีปฏิบัติ ดังนี้

ความร่วมมือระหว่างทีม (collaborative) เป็นแนวทางปฏิบัติเพื่อส่งเสริมให้เกิดการทำงานร่วมกันระหว่างทีมและภายในทีม ซึ่งจะช่วยให้สมาชิกแต่ละทีมเข้าใจปัญหาที่เกิดขึ้นอย่างแท้จริงและสามารถแก้ไขปัญหาที่เกิดขึ้นได้อย่างมีประสิทธิภาพ โดยมีการแลกเปลี่ยนประสบการณ์และร่วมกันแก้ปัญหาที่เกิดขึ้น

ขั้นตอนดำเนินการ (procedural) เป็นกลุ่มวิธีปฏิบัติของทีมนักพัฒนาซอฟต์แวร์และทีมปฏิบัติการ ในทำงานร่วมกันโดยอาศัยหลักการของความต่อเนื่องในการส่งมอบซอฟต์แวร์ (continuous software delivery) ประกอบด้วยขั้นตอน continuous integration,continuous delivery (CD) และ continuous deployment/release ดังภาพที่ 2.3 ซึ่งผลลัพธ์ปลายทางนำไปสู่การส่งมอบซอฟต์แวร์เพื่อให้บริการ (service) (Humble & Farley, 2011) ประกอบด้วย



ภาพที่ 2.3 แสดงกระบวนการความต่อเนื่องของวิธีปฏิบัติของแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps., จาก https://www.intellegus.nl/nieuws/devops-waar-en-hoe-beginne

(1) ขั้นตอนการผสมผสานความต่อเนื่อง (continuous integration)

เป็นจุดเริ่มต้นความต่อเนื่องของความเป็นอัตโนมัติ แม้ว่ากระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์จะมีการกล่าวถึงขั้นตอนผสมผสานความต่อเนื่อง โดยเกิดจากวิธีการ Extreme Programming ซึ่งเกิดจากที่นักพัฒนาซอฟต์แวร์หลายคนทำงานร่วมกัน เมื่อถึงเวลารวบรวมซอร์สโค้ดจำเป็นต้อง Build, Run และทดสอบในมุมมองวิธีปฏิบัติขั้นพื้นฐาน แต่ไม่ได้เหมาะสมสำหรับการใช้งานจริง (implement) ที่มีการเปลี่ยนแปลงบ่อยครั้ง (Dingsøy & Lassenius,2016) ซึ่งแตกต่างจากแนวคิดของ DevOps คือ ทุกครั้งที่ซอร์สโค้ดมีการเปลี่ยนแปลงจะถูกนำไปเก็บที่ระบบควบคุมเวอร์ชั่นของซอฟต์แวร์ (version control system) (Humble & Farley, 2011) หลังจากนั้น continuous integration server (CI server) จะมีกลไกประสานการทำงานแต่ละขั้นตอนอย่างเป็นอัตโนมัติ ตั้งแต่การคอมไพล์ (compile) การสร้างคำสั่ง (script) เพื่อนำซอฟต์แวร์ไปทดสอบในมิติต่าง ๆ ประกอบด้วย การทดสอบระดับหน่วยฟังก์ชั่น และการทดสอบระดับการทำงานของโปรแกรม ซึ่งการทดสอบในขั้นตอนนี้จะเป็นการทดสอบในมุมมองนักพัฒนาซอฟต์แวร์เพื่อทดสอบความเข้ากันได้ของซอฟต์แวร์ หากพบข้อผิดพลาดจะแจ้งเตือนแก่นักพัฒนาซอฟต์แวร์เพื่อให้สามารถแก้ไขได้ทันที (Morris, 2016)

(2) ขั้นตอนส่งมอบอย่างต่อเนื่อง (continuous delivery) (CD)

เป็นขั้นตอนต่อจาก continuous integration โดยจะนำซอฟต์แวร์เข้าสู่สภาพแวดล้อมในการทดสอบ (UAT Server) ประกอบด้วย การทดสอบระดับผู้ใช้งาน (user Acceptance Test: UAT) ในมุมมองธุรกิจ ซึ่งมีการจำลองสภาพแวดล้อมให้เสมือนการใช้งานจริง รวมทั้งทีมปฏิบัติการที่จะต้องเตรียมสภาพแวดล้อมสำหรับนำซอฟต์แวร์ไปใช้งานจริง (production server) (Sharma, 2013) ซึ่งกระบวนการ continuous delivery จะมีความคล้ายกับ continuous deployment แตกต่างเพียงแค่การนำซอฟต์แวร์เหล่านี้ ไปสู่สภาพแวดล้อมที่ใช้งานจริงโดยอัตโนมัติ (Humble & Farley, 2011) ทั้งนี้หากมีการเปลี่ยนแปลงใด ๆ ที่เกิดขึ้น จะต้องได้รับการแก้ไขเพื่อให้ระบบสามารถทำงานได้อย่างต่อเนื่อง (Morris, 2016)

(3) ขั้นตอนปรับปรุงอย่างต่อเนื่อง (continuous improvement)

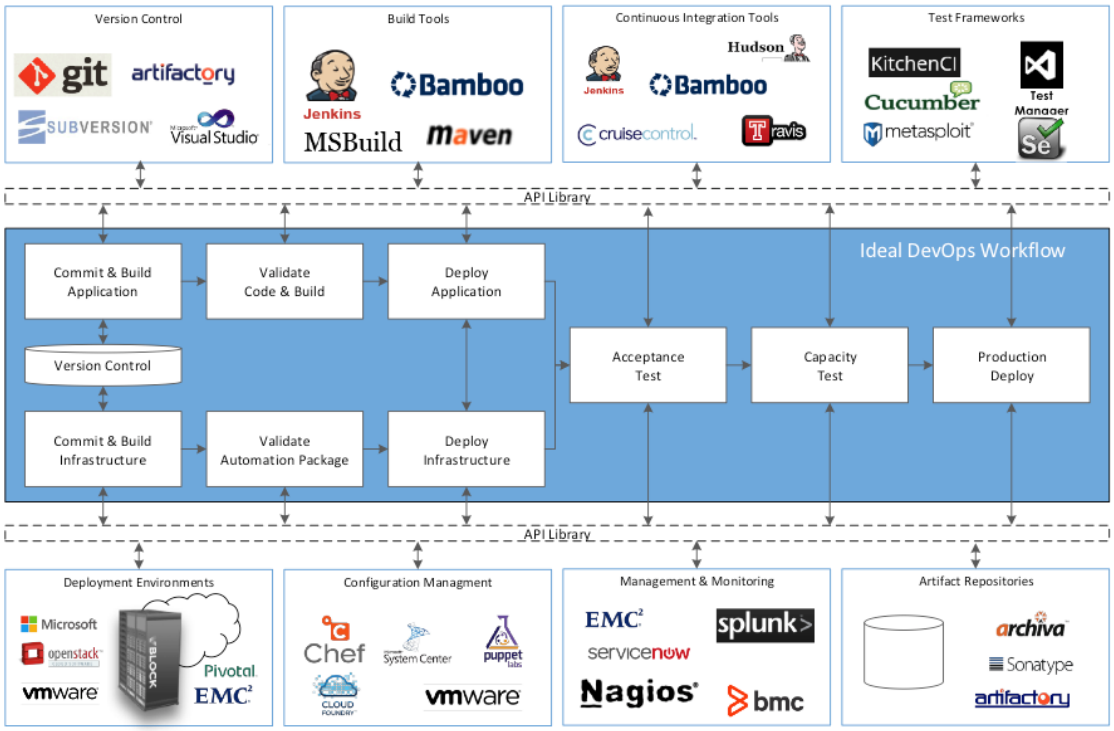
การปรับปรุงอย่างต่อเนื่องเป็นวัฒนธรรม (culture) อย่างหนึ่งของแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps เนื่องจากการรวบรวมข้อเสนอแนะ และการวัดผลอย่างสม่ำเสมออยู่บนพื้นฐานของข้อมูล ระเบียบแบบแผน ข้อกำหนด และการปฏิบัติตามแผนที่วางไว้อย่างต่อเนื่อง (Airaj, 2015)

2.3.3 การบริการ (service)

เป็นขั้นตอนการน าซอฟต์แวร์ไปสู่การให้บริการในรูปแบบที่หลากหลาย เช่น Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) โดยมีเทคโนโลยีแบบกลุ่มเมฆ (cloud technology) เป็นตัวสนับสนุนการทำงาน

2.3.4 กลุ่มของเครื่องมือที่สร้างความต่อเนื่องในกระบวนงาน (DevOps tool chain)

กลุ่มของเครื่องมือเป็นสิ่งที่สำคัญในแนวคิดการพัฒนาซอฟต์แวร์รูปแบบ DevOps โดยกลุ่มของเครื่องมือที่ใช้จัดการกระบวนการต่าง ๆ ที่เกิดขึ้นในการพัฒนาซอฟต์แวร์มีความครอบคลุมตั้งแต่การพัฒนาซอฟต์แวร์ จนกระทั่งส่งมอบซอฟต์แวร์ รวมถึงติดตามผลลัพธ์ที่เกิดขึ้น (Gartner, 2015) โดยส่งเสริมความร่วมมือในการทำงานระหว่างทีมงานในมุมมองทีมนักพัฒนาซอฟต์แวร์ และทีมปฏิบัติการ ซึ่งแต่ละกระบวนการจะมีเครื่องมือที่เหมาะสมแตกต่างกัน ทำให้สามารถตอบสนองความต้องการได้อย่างครบถ้วน (Gartner, 2015; Higgins, 2016; Menzel & Macaulay, 2014) โดยสามารถจำแนกเครื่องมือได้ตามกิจกรรมที่เกิดขึ้น ดังภาพที่ 2.4 และคำอธิบายขั้นตอนกิจกรรมที่สอดคล้องกับเครื่องมือ ดังตารางที่ 2.2



ภาพที่ 2.4 แสดงเครื่องมือสนับสนุนการดำเนินงานแต่ละขั้นตอน. จาก “Common DevOps Tool Chains Pitfalls,” https://infocus.emc.com/bart_driscoll/common-devopstool-chains-pitfalls., โดย Driscoll, 2015.

กิจกรรม	ลักษณะการทำงาน	ตัวอย่าง
1. การวางแผน	ใช้สำหรับเก็บรวบรวมความต้องการ สร้างแผนภาพ (diagram) และการออกแบบ (design) รวมถึงการติดต่อสื่อสารระหว่างทีม	JIRA , Github ,TeamForge , Team Foundation
2. การสร้าง	การบิ้วด์ การเก็บ และควบคุมเวอร์ของซอร์สโค้ดภายใต้ CI Server (continuous integration server)	Git, Bamboo , MS Build, SubVersion, Jenkins
3. การทดสอบ	ตรวจสอบคุณภาพของข้อมูลและท าการทดสอบโดยอัตโนมัติ (automated testing)	Kitchen CI, Junit, Cucumber
4. การเตรียมก่อนการใช้งานจริง	การจัดเก็บไฟล์และสร้างซอฟต์แวร์ขึ้นมาสำหรับติดตั้งบนเซิร์ฟเวอร์ และการเตรียมสภาพแวดล้อมสำหรับการทดสอบในรูปแบบต่าง ๆ	Apache Archiva, Inedo's ProGet Sonatype Nexus
5. การส่งมอบ	การส่งมอบระบบโดยอัตโนมัติ (automated deployment) รวมถึงเตรียมการกู้คืนระบบหากเกิดความล้มเหลวในการส่งมอบ	Packer, Dockers, VMware
6. การปรับแต่ง	กระบวนการที่จัดการกับความเปลี่ยนแปลงที่เกิดขึ้น	Chef, Puppet, VmWare
7. การติดตามผล	การดูแลและติดตามประสิทธิภาพของซอฟต์แวร์และเซิร์ฟเวอร์ รวมถึงการติดตามปัญหาและการเปลี่ยนแปลงที่เกิดขึ้น	Stats, jmxtrans, Metrics, Ganglia

ตารางที่ 2.2 “ตัวอย่างเครื่องมือสำหรับ devops” http://blogs.atlassian.com/2016/03/how-to-choose-devops-tools โดย Zorah, 2016.

2.4 แนวคิดการทำงานของเครื่องเสมือนกับคอนเทนเนอร์

รูปแบบการทำงานของเซิร์ฟเวอร์ของแต่ละกระบวนการพัฒนาซอฟต์แวร์นั้น โดยในยุคแรกของการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall) ใช้เซิร์ฟเวอร์เชิงกายภาพ (Physical

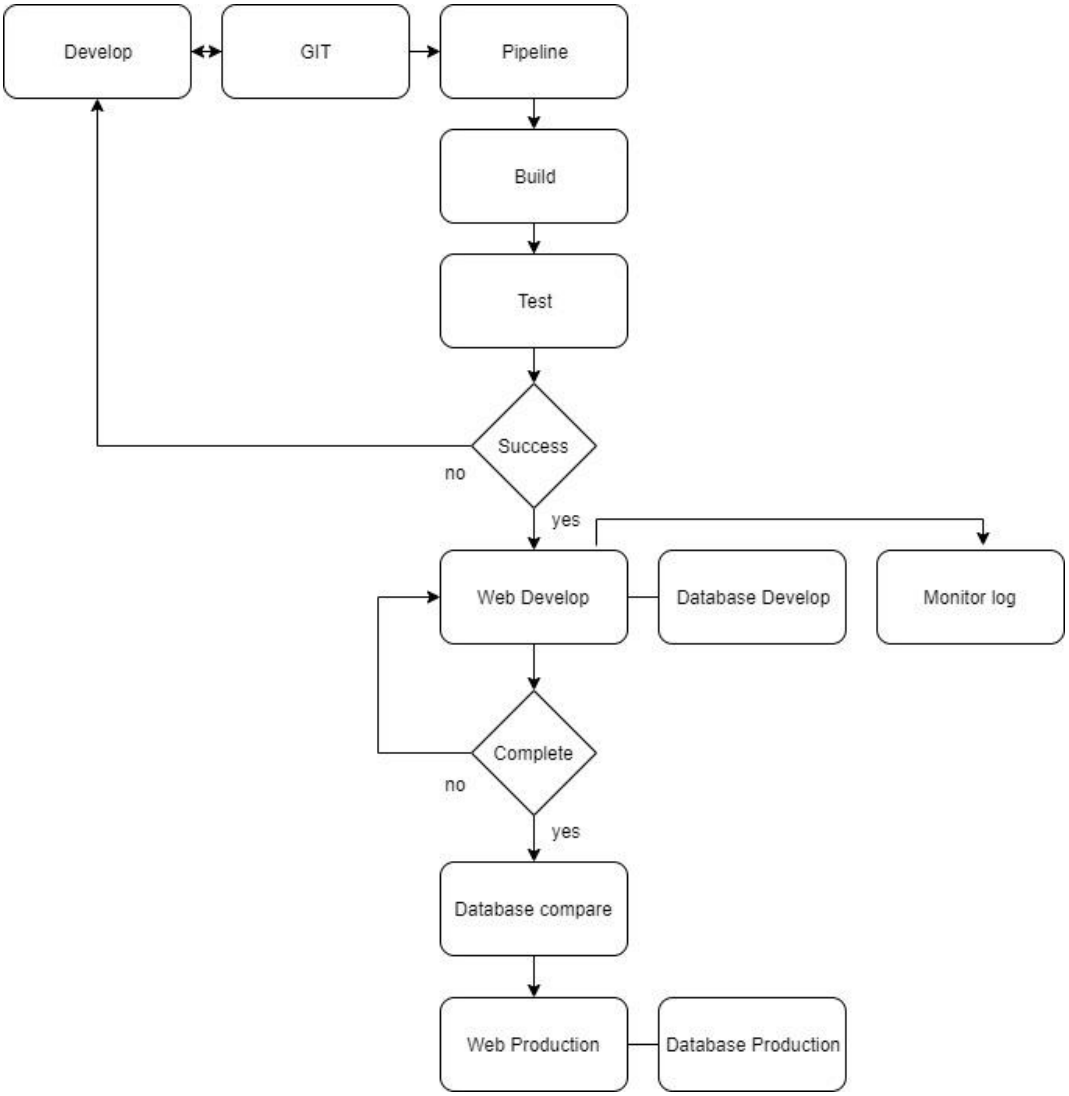
Server) ผู้ใช้จะต้องติดตั้งระบบปฏิบัติการของเซิร์ฟเวอร์และดูแลรักษาด้วยตนเอง ทำให้เกิดความยุ่งยากในการบริหารจัดการและมีค่าใช้จ่ายสูง (Rosehosting, 2016) ส่วนของระเบียบแบบแอจไจล์จะอยู่ในยุคของการใช้เครื่องเสมือน (Virtual Machine) ที่ช่วยในการบริหารจัดการทรัพยากรที่มีอยู่บนเซิร์ฟเวอร์ให้มีหลายเครื่อง แต่ทำางานอยู่ภายใต้เครื่องเดียวกัน ซึ่งจะมีความแตกต่างขอทรัพยากร (Resource) ประสิทธิภาพและระบบปฏิบัติการ (Operating System) ที่ใช้งานดังภาพที่ 2.5 (ซ้ายมือ) และเริ่มหันมาใช้บริการกลุ่มเมฆแบบส่วนตัว (Private Cloud) โดยจะบริหารจัดการภายในองค์กรเอง เนื่องจากความกังวลด้านความปลอดภัย ทั้งนี้จะช่วยลดความยุ่งยากในการดูแลรักษาลงไป (White,2016) และแนวคิด DevOps จะใช้คอนเทนเนอร์ (Container) เป็นการจำลองเพื่อควบคุมสภาพแวดล้อมสำหรับการทำงานเฉพาะเซอร์วิสโดยจะบรรจุสภาพแวดล้อมนั้นลงในคอนเทนเนอร์ ก่อให้เกิดการใช้ทรัพยากรเท่าที่จำเป็น การทำงานจะอยู่ในระดับระบบปฏิบัติการเท่านั้นดังภาพที่ 2.5 (ขวามือ) แตกต่างจากการจำลองเซิร์ฟเวอร์เสมือนที่ใช้สภาพแวดล้อมในระบบปฏิบัติการทั้งหมดเพื่อสร้างเซิร์ฟเวอร์หนึ่งเครื่องที่มีหลายๆ เซอร์วิสทำงานร่วมกัน ทั้งนี้คอนเทนเนอร์ จะมีด๊อกเกอร์ (Dockers) ถือว่าเป็นเครื่องมือที่ได้รับความนิยมในการเข้ามาจัดการคอนเทนเนอร์ ทำให้สามารถใช้งานได้ง่ายและยังสนับสนุนการใช้เทคโนโลยีกลุ่มเมฆทั้งแบบส่วนตัวและสาธารณะ โดยแบบสาธารณะนั้นองค์กรจะไม่สามารถบริหารจัดการได้เองทั้งหมด แต่จะมีผู้ให้บริการเป็นคนบริหารจัดการให้ (White, 2016)

ภาพที่ 2.5 ลักษณะการทำงานของเครื่องเสมือน (Virtual Machine) กับคอนเทนเนอร์ (Containers) จาก “ทำความรู้จัก Docker และ Software Container” https://medium.com/thothzocial-engineering/ทำความรู้จัก-docker-และ-software-container-c6338629da11 โดย Pattanapong Cherthong (2016)

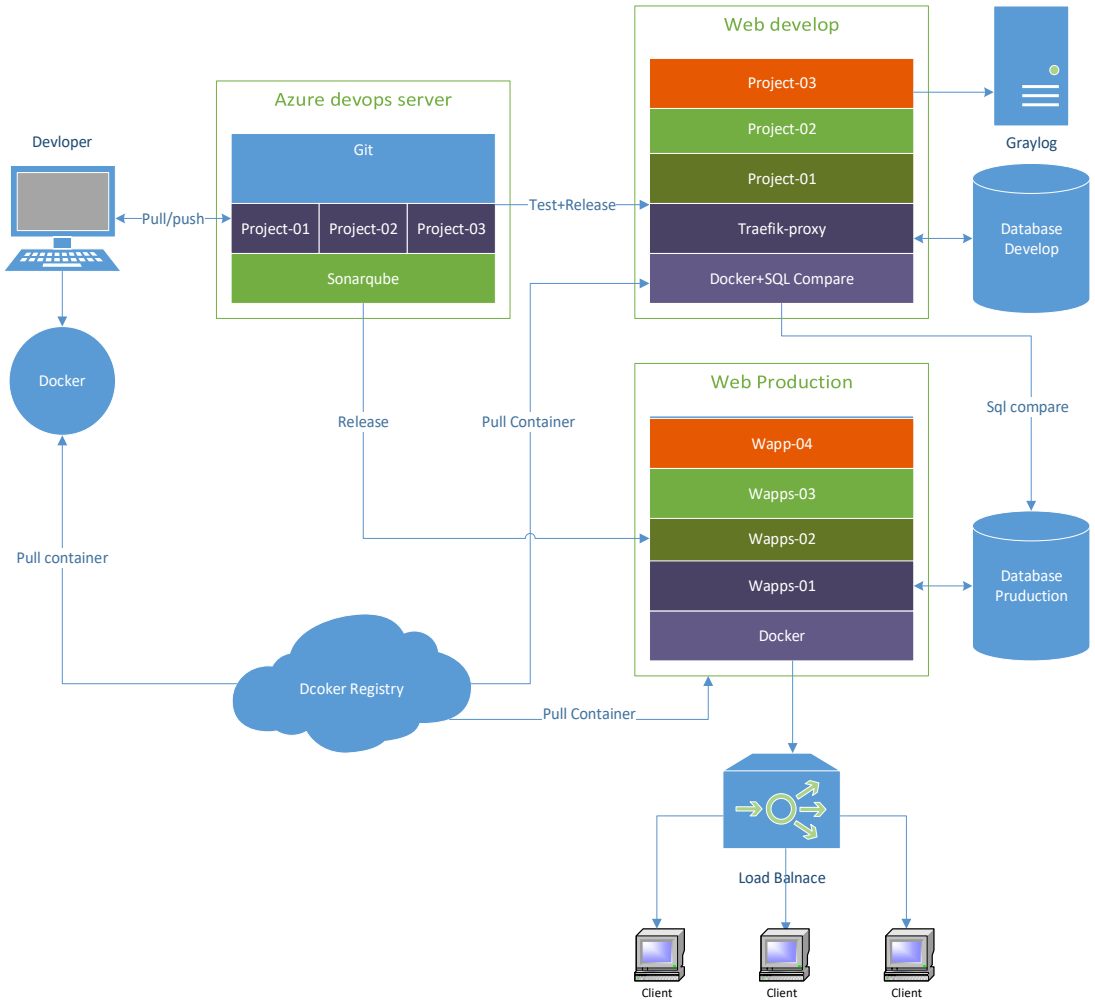
บทที่ 3
วิธีดำเนินการวิจัย

3.1 วิธีการดำเนินการวิจัย

ออกแบบและพัฒนาระบบ Azure Devops เพื่อปรับปรุงกระบวนการพัฒนาระบบสารสนเทศของสำนักคอมพิวเตอร์ประกอบไปด้วยเครื่อง Azure Devops server, Web Develop, Database Develop, Web Production, Database Production และ Monitor log สำหรับเครื่อง Azure Devops server ภายในจะประกอบไปด้วย GIT ที่ใช้จัดเก็บและควบคุมการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ ส่วนถัดมาก็คือ Pipeline ใช้สำหรับสร้างกระบวนการในการทำงาน Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor และ Sonarqube ใช้สำหรับตรวจสอบคุณภาพของซอร์สโค้ด (Source code) ช่วยหาข้อบกพร่องในซอร์สโค้ด ไม่ว่าจะเป็น Bug ที่น่าจะเกิดขึ้นพร้อมทั้งสร้างรายงานผลการตรวจให้นักพัฒนาทราบ โดยกระบวนการถัดมาหลังจากที่ซอร์สโค้ดดำเนินการบิ้วด์ (Build) และทดสอบ (Testing) เสร็จสิ้น ซอร์สโค้ดจะถูกคัดลอกและดำเนินการส่งไปที่เครื่องแม่ข่าย Web Develop และส่งคำสั่งเพื่อเรียกใช้งานเซอร์วิสด๊อกเกอร์ (Docker) เพื่อดำเนินการสร้างคอนเทนเนอร์ (Container) พร้อมทั้งตั้งค่าพร็อกซี (Proxy) สำหรับเรียกใช้งานเว็บไซต์เพื่อใช้ทดสอบระบบ ขั้นตอนถัดมาหลังจากดำเนินการสร้างคอนเทนเนอร์เสร็จสิ้นแล้วกระบวนการถัดไปจะทำการส่งคำสั่งเพื่อเรียกใช้งานเซอร์วิส Syslog และส่ง log ของระบบที่ใช้ทดสอบ ไปยังเครื่องแม่ข่าย Monitor log เพื่อใช้สำหรับทดสอบระบบว่ามีปัญหาอะไรหรือไม่ ส่วนลำดับสุดท้ายหากระบบถูกพัฒนาเสร็จโดยสมบูรณ์ซอร์สโค้ดจะถูกคัดลอกและดำเนินการส่งไปที่เครื่องแม่ข่าย Web production และทำการตรวจสอบฐานข้อมูลระหว่าง Database Develop และ Database Production ว่ามีโครงสร้างข้อมูล (Schemas) อะไรที่เปลี่ยนแปลงหรือไม่ โดยจะรายงานให้ผู้พัฒนาระบบทราบและดำเนินการสร้างคำสั่ง SQL Script เพื่อใช้สำหรับนำเข้าข้อมูลได้อีกด้วย โดยจะแสดงขั้นตอนกระบวนการทำงาน Devops ดังภาพที่ 3.1 และโครงสร้างของระบบ Devops ดังภาพที่ 3.2



ภาพที่ 3.1 ขั้นตอนกระบวนการทำงาน Devops



ภาพที่ 3.2 โครงสร้างของระบบ Devops

3.2 เครื่องมือที่ใช้ในการวิจัย

1. เครื่องให้บริการ Devops

    1.1 Azure devops server 2019

    1.2 SQL Server 2017

    1.3 Sonarqube 8.0

    1.4 Node.js 12.13.0

    1.5 Java 8

2. เครื่องให้บริการ Web Develop

    2.1 Docker 19.03.4

    2.2 php-mysql-diff 1.0

    2.3 Apache 2.4.29

2.4 PHP 7.3

2.5 NXLog 2.10

2.6 Traefik 1.7.6

3. เครื่องให้บริการ Web Production

3.1 Docker 19.03.4

3.2 Apache 2.4.29

3.3 PHP 7.3

4. เครื่องให้บริการ Database Develop

4.1 MariaDB 10.4.12

5. เครื่องให้บริการ Database Production

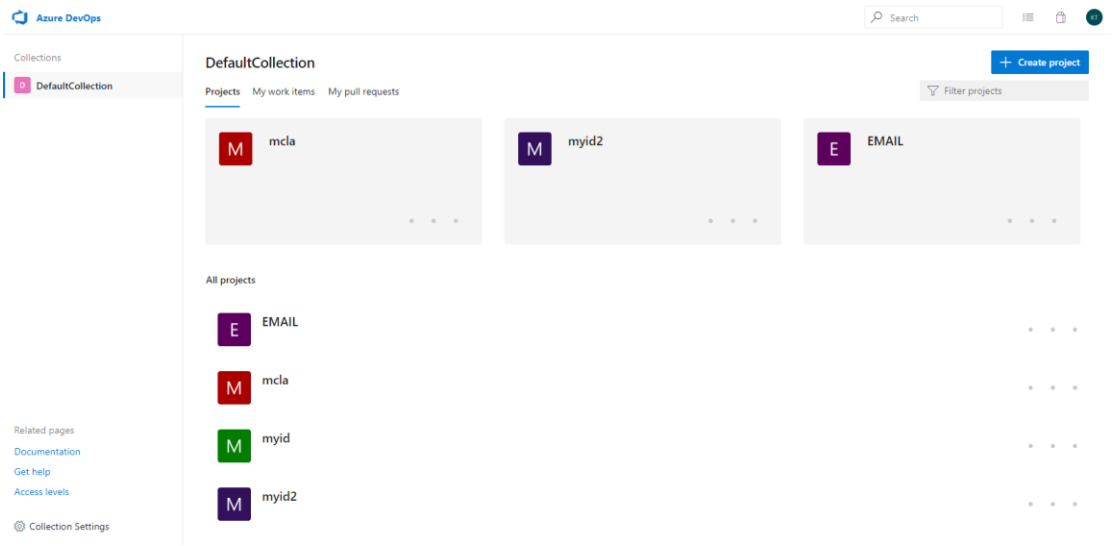
5.1 MariaDB 10.4.12

6. เครื่องติดตามการทำงานของซอร์สโค้ด (Monitor)

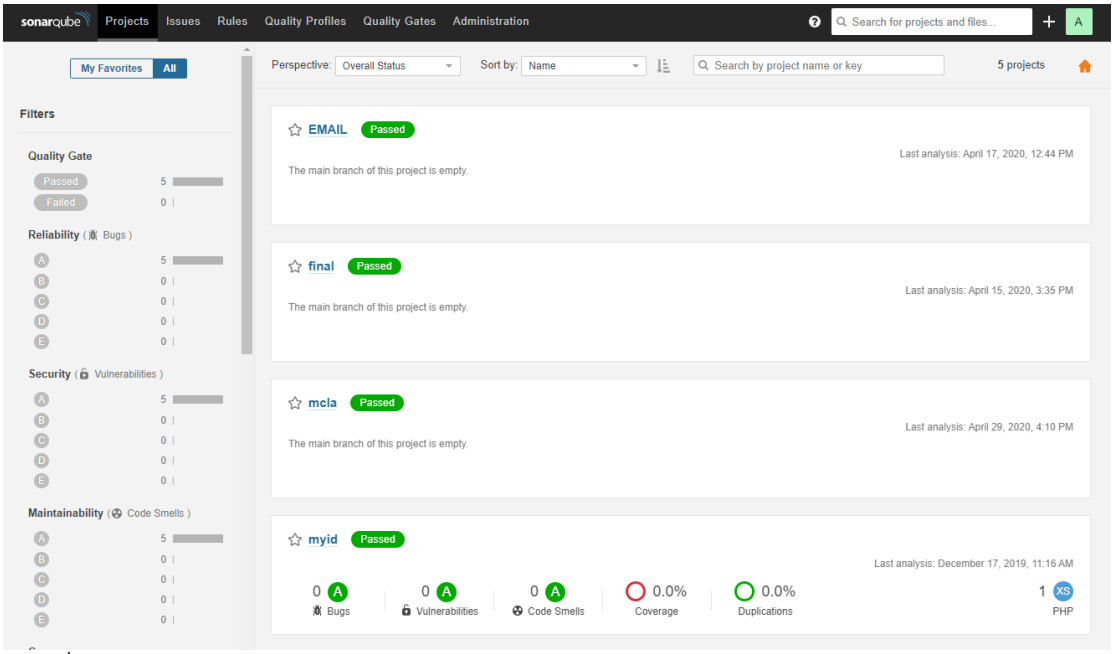
6.1 Graylog 3.1.1

3.3 ขั้นตอนดำเนินงานวิจัย

1. ดำเนินการติดตั้งเครื่องให้บริการ Azure Devops โดยทำการติดตั้ง Azure devops server 2019 และ SQL Server 2017 เพื่อใช้สำหรับเป็น Devops Service ดังภาพที่ 3.3 และขั้นตอนถัดมาได้ดำเนินการติดตั้ง Sonarqube และ Java JRE 8 เพื่อใช้สำหรับตรวจสอบคุณภาพของซอร์สโค้ด แต่ข้อจำกัด Sonarqube Community นั้นสามารถตรวจสอบคุณภาพของซอร์สโค้ดได้เพียง Branch master เท่านั้น จึงจำเป็นต้องทำการติดตั้ง Community Branch Plugin เพื่อให้ Sonarqube สามารถทำการตรวจสอบ Branch อื่น ๆ นอกเหนือจาก Branch master ได้ดังภาพที่ 3.4 และขั้นตอนสุดท้ายได้ดำเนินการติดตั้ง Node.js สำหรับทำงานร่วมกับ Sonarqube ในการตรวจสอบคุณภาพของซอร์สโค้ดภาษาต่าง ๆ เพิ่มมากขึ้น



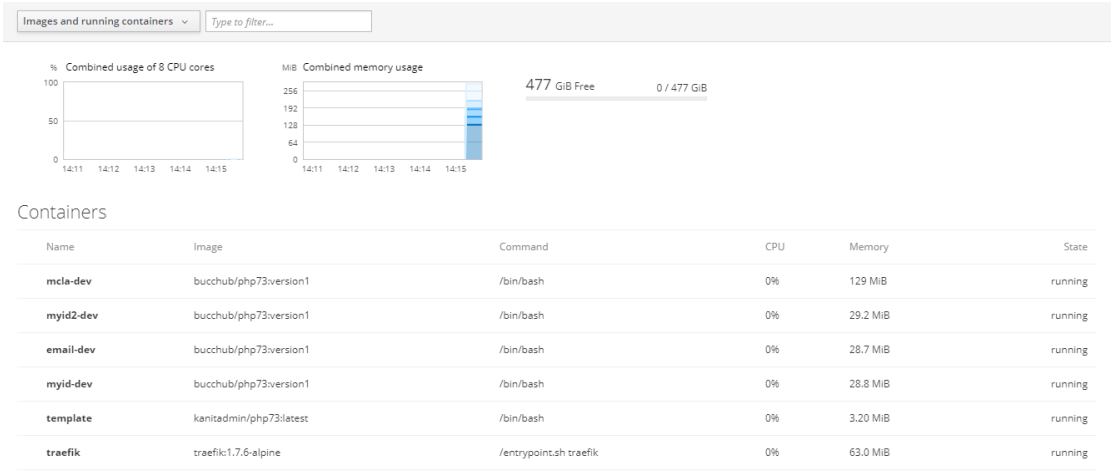
ภาพที่ 3.3 ระบบบริหารจัดการ Devops



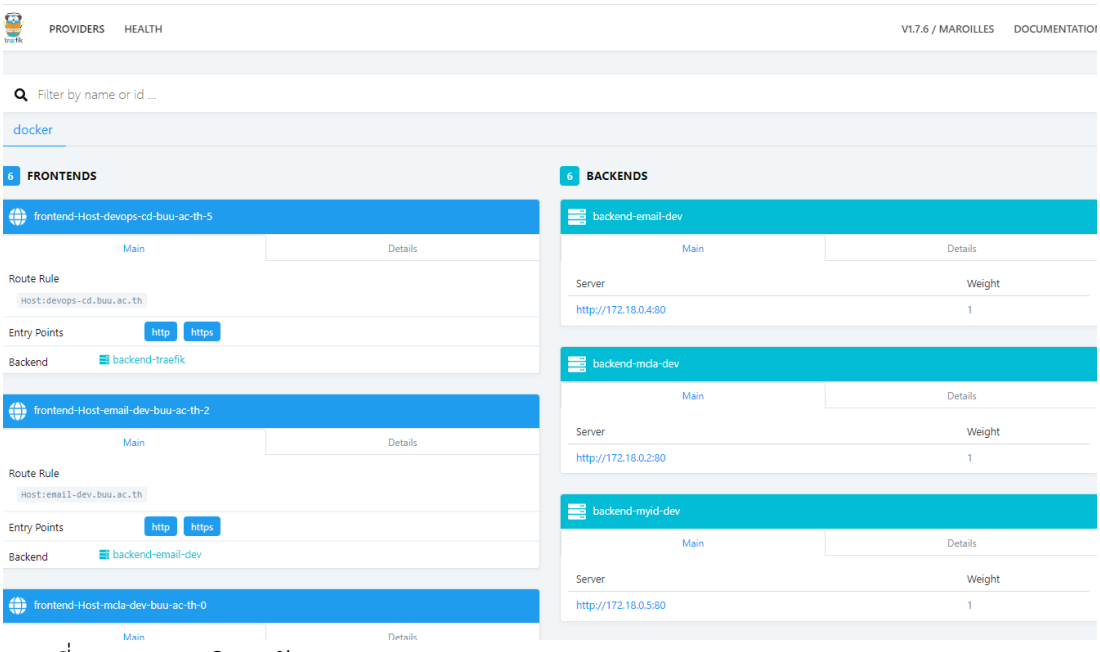
ภาพที่ 3.4 ระบบตรวจสอบคุณภาพของซอร์สโค้ด

2. ดำเนินการติดตั้งเครื่องให้บริการ Web Develop โดยทำการติดตั้ง Docker ใช้สำหรับสร้างคอนเทนเนอร์ (Container) เพื่อใช้ทดสอบการทำงานของ Web Develop ดังรูปที่ 3.5 และขั้นตอนถัดมาได้ดำเนินการสร้างคอนเทนเนอร์เซอร์วิส Traefik Proxy โดยที่ Traefik Proxy เพื่อทำหน้าเป็นตัวรับคำร้องขอการใช้งาน (request) มาประมวลผลและส่งต่อไปยังคอนเทนเนอร์ Web Develop ที่กำหนดไว้ พร้อมทั้งตั้งค่าให้สามารถใช้งาน SSL เพื่อความปลอดภัยในการรับส่งข้อมูลดังภาพที่ 3.6 ขั้นตอนถัดมาทำการติดตั้ง php-mysql-diff เพื่อใช้สำหรับตรวจสอบฐานข้อมูลระหว่าง Database Develop และ Database Production ว่ามีโครงสร้างข้อมูล (Schemas) อะไรที่เปลี่ยนแปลงหรือไม่ โดยจะรายงานให้ผู้พัฒนาระบบทราบและดำเนินการสร้างคำสั่ง SQL Script

เพื่อใช้สำหรับนำเข้าข้อมูลอีกด้วย และขั้นตอนสุดท้ายทำการติดตั้งโปรแกรม NXLog เพื่อใช้สำหรับส่งข้อมูลจราจรไปยังเครื่องเก็บข้อมูลจราจรส่วนกลาง เพื่อใช้สำหรับติดตามการทำงานของซอร์สโค้ด (Source code) ว่ามีปัญหาอะไรหรือไม่ดังภาพที่ 3.7



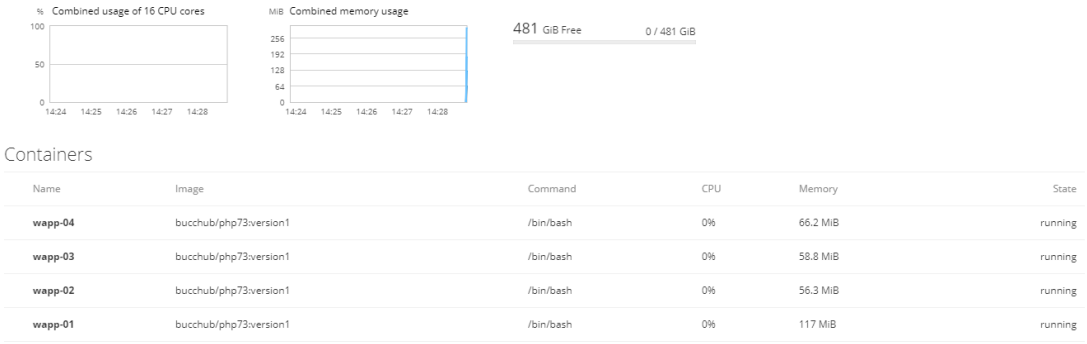
ภาพที่ 3.5 ระบบบริหารจัดการด๊อกเกอร์สำหรับเครื่อง Web Develop



ภาพที่ 3.6 ระบบบริหารจัดการ Traefik Proxy

ภาพที่ 3.7 ระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์ส่วนกลางสำหรับติดตามการทำงานของซอร์สโค้ด

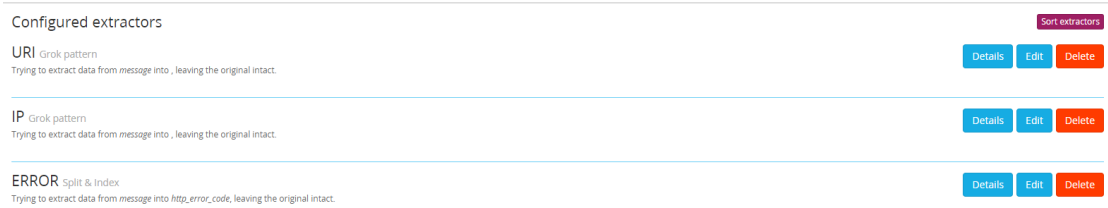
3. ดำเนินการติดตั้งเครื่องให้บริการ Web Production โดยทำการติดตั้งด๊อกเกอร์ (Docker) ใช้สำหรับสร้างคอนเทนเนอร์ (Container) เพื่อใช้สำหรับการให้บริการของ Web Production ขั้นตอนถัดมาได้ทำการสร้างคอนเทนเนอร์ จำนวน 4 คอนเทนเนอร์ โดยแต่ละคอนเทนเนอร์ได้ทำการติดตั้ง Apache 2.4, PHP 7.3 และส่วนเสริม (Extension) ที่จำเป็นเพื่อใช้งาน Web Service ดังภาพที่ 3.8



ภาพที่ 3.8 ระบบบริหารจัดการด๊อกเกอร์สำหรับเครื่อง Web Production

4. ดำเนินการติดตั้งเครื่องให้บริการ Database Develop และ Database Production โดยทำการติดตั้ง Mariadb 10.4.14 ขั้นตอนถัดมาได้ทำการตั้งค่าและปรับแต่งเพื่อให้ฐานข้อมูลทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น และขั้นตอนสุดท้ายได้ดำเนินการสร้างชุดคำสั่งเพื่อทำการสำรองฐานข้อมูลโดยแยกแต่ละฐานข้อมูลเพื่อง่ายต่อการกู้คืน

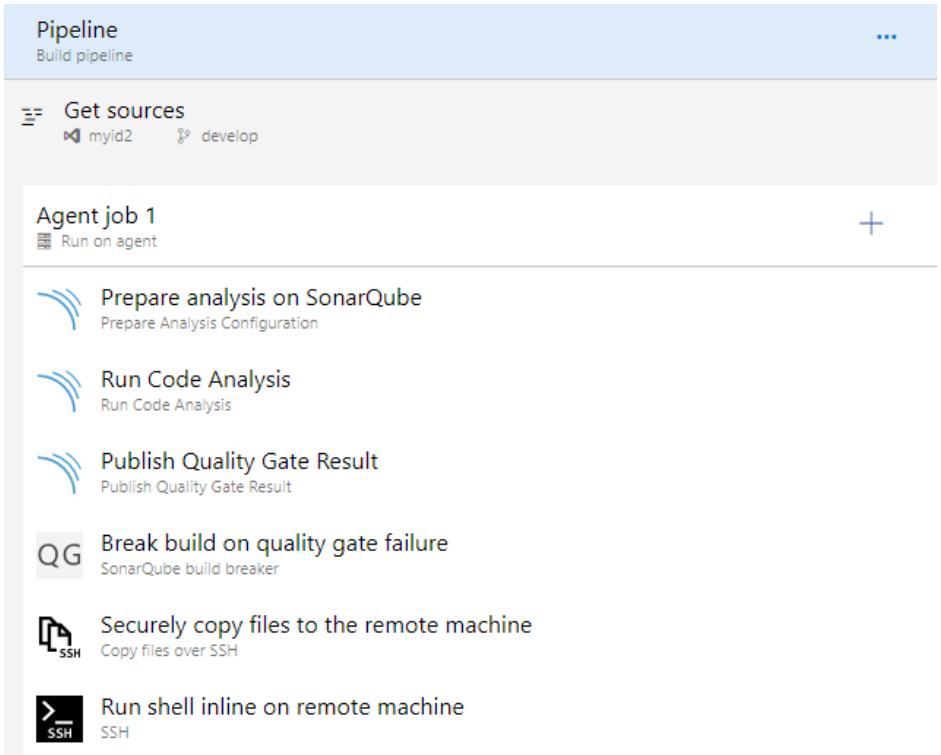
5. ดำเนินการตั้งค่าของเครื่อง Graylog Server โดยทำการสร้างกฎจำแนกข้อมูลจราจรทางคอมพิวเตอร์ (Extractor) ที่ได้รับมาจากเครื่อง Web Develop โดยจำแนกข้อมูลจราจรทางคอมพิวเตอร์เพื่อนำไปสร้าง Dashboard ได้ดังภาพที่ 3.9



ภาพที่ 3.9 กฎจำแนกข้อมูลจราจรทางคอมพิวเตอร์ (Extractor)

6. ดำเนินการสร้าง Pipeline โดยแยกเป็นสองส่วนดังนี้

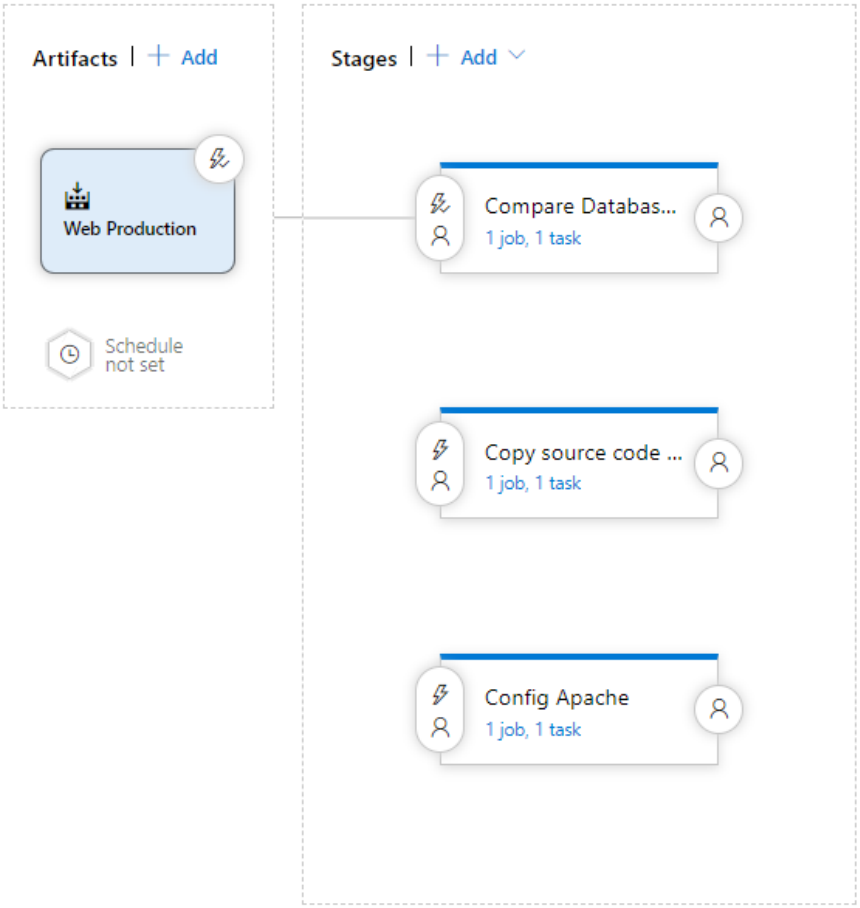
(1) Pipeline สำหรับ Web develop ซึ่งจะมีลำดับการทำงานดังนี้ 1) ตั้งค่าเพื่อให้ Azure devops สามารถติดต่อกับระบบ Sonarqube ได้ 2) ตรวจสอบคุณภาพของซอร์สโค้ด (Source code) 3) ส่งผลของการตรวจสอบคุณภาพของซอร์สโค้ดไปยังระบบ Sonarqube 4) ตรวจสอบผลของการตรวจสอบคุณภาพของซอร์สโค้ดหากไม่ผ่านให้หยุดการทำงานของ Pipeline 5) คัดลอกซอร์สโค้ดไปยังเครื่อง Web Develop 6) สร้างคอนเนอร์ (Container) โดยกำหนด Proxy ในการเรียกใช้งานเว็บไซต์ ดำเนินการทำการสร้าง Virtual Host ของ Apache และส่งคำสั่งให้ NXLog ดำเนินการส่งข้อมูลจราจรทางคอมพิวเตอร์ไปยังระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์ส่วนกลาง ดังภาพที่ 3.10



ภาพที่ 3.10 แสดงลำดับการทำงานของ Pipeline Web Develop

(2) Pipeline สำหรับ Web Production ซึ่งจะมีลำดับการทำงานดังนี้ 1) ทำการตรวจสอบฐานข้อมูลระหว่าง Database Develop และ Database Production ว่ามี

Schemas อะไรที่เปลี่ยนแปลงหรือไม่ โดยจะรายงานให้ผู้พัฒนาระบบทราบและดำเนินการสร้าง SQL Script เพื่อใช้สำหรับนำเข้าข้อมูล 2) คัดลอกซอร์สโค้ดไปยังเครื่อง Web Production 3) ทำการสร้าง Virtual Host ของ Apache ดังภาพที่ 3.11



ภาพที่ 3.11 แสดงลำดับการทำงานของ Pipeline Web Production

บทที่ 4
ผลการวิจัย

ผลการวิจัยจะเปรียบเทียบระหว่างการพัฒนาระบบแบบเดิมกับการพัฒนาระบบแบบใหม่ โดยจะแสดงรายละเอียดขั้นตอนการดำเนินงานทั้งสองกระบวนการว่ามีความแตกต่างอย่างไร และมีประสิทธิภาพดีขึ้นด้านใดบ้าง ซึ่งจะประเมินผลเปรียบเทียบดังนี้

1. เปรียบเทียบจำนวนขั้นตอนการพัฒนาระบบ
2. การส่งมอบงานได้อย่างอย่างรวดเร็วและมีประสิทธิภาพ
3. ความน่าเชื่อถือของซอฟต์แวร์
4. การลดขั้นตอนการดำเนินงานและจัดการโครงสร้างพื้นฐาน
5. การทดสอบกระบวนการทำงานจริงร่วมกับฝ่ายพัฒนาระบบ

4.1 เปรียบเทียบกระบวนการการพัฒนาระบบ

กิจกรรม	แบบเดิม	แบบใหม่
1. การวางแผน	เก็บรวบรวมความต้องการ สร้างแผนภาพ และการออกแบบ รวมถึงการติดต่อสื่อสารระหว่างทีมโดยการประชุม	เก็บรวบรวมความต้องการ สร้างแผนภาพ และการออกแบบ รวมถึงการติดต่อสื่อสารระหว่างทีมโดยระบบ Azure boards และสามารถติดตามการดำเนินงานได้
2. การสร้าง	เก็บซอร์สโค้ดบนเครื่องพัฒนาระบบ	เก็บและควบคุมเวอร์ชั่นของซอร์สโค้ดด้วย Version Control
3. การทดสอบ	ตรวจสอบการทำงานของโปรแกรมเพียงอย่างเดียว	ตรวจสอบการทำงานของโปรแกรมและตรวจสอบคุณภาพของซอร์สโค้ดโดยอัตโนมัติ
4. การเตรียมก่อนการใช้งานจริง	จัดเก็บไฟล์และสร้างซอฟต์แวร์ขึ้นมาสำหรับติดตั้งบนเครื่องพัฒนาระบบโดยตรงและการเตรียมสภาพแวดล้อมเดียวกันไม่สามารถ	จัดเก็บไฟล์และสร้างซอฟต์แวร์ขึ้นมาสำหรับติดตั้งบนเซิร์ฟเวอร์ และการเตรียมสภาพแวดล้อมสำหรับการทดสอบในรูปแบบต่าง ๆ โดยอิสระ

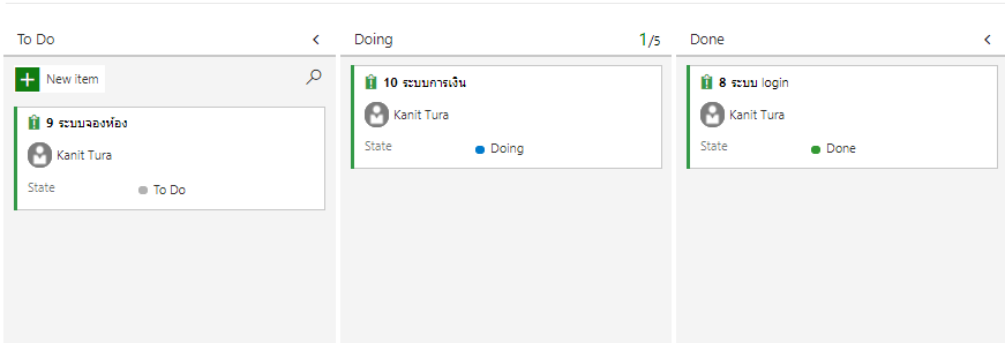
	ปรับเปลี่ยนสภาพแวดล้อมเดิม เพราะอาจจะส่งผลกระทบกับระบบอื่น ๆ	จากกันและไม่มีผลกระทบกับระบบอื่น ๆ
5. การส่งมอบ	ส่งมอบระบบโดยการคัดลอกซอร์สโค้ดด้วยตนเอง	ส่งมอบระบบโดยอัตโนมัติรวมถึงเตรียมการกู้คืนระบบหากเกิดความล้มเหลวในการส่งมอบ
6. การติดตามผล	ดูแลและติดตามประสิทธิภาพของซอฟต์แวร์และเซิร์ฟเวอร์ รวมถึงการติดตามปัญหาและการเปลี่ยนแปลงที่เกิดขึ้นที่เครื่องพัฒนาระบบ	ดูแลและติดตามประสิทธิภาพของซอฟต์แวร์และเซิร์ฟเวอร์ รวมถึงการติดตามปัญหาและการเปลี่ยนแปลงที่เกิดขึ้นด้วยระบบจัดเก็บข้อมูลส่วนกลาง

ตารางที่ 4.1 เปรียบเทียบกระบวนการการพัฒนาระบบ

จากตารางที่ 4.1 จะแสดงความแตกต่างระหว่างการพัฒนาระบบแบบเดิมกับการพัฒนาระบบแบบใหม่ ซึ่งจะมีกิจกรรมได้แก่ การวางแผน การสร้าง การทดสอบ การเตรียมก่อนการใช้งานจริง การส่งมอบ และการติดตามผล ซึ่งจะเห็นว่าทุกกิจกรรมมีการปรับปรุงการทำงานทั้งหมดเพื่อลดข้อจำกัดต่าง ๆ ของการพัฒนาระบบที่ผ่านมาให้ดียิ่งขึ้น โดยจะสรุปแต่ละกิจกรรมได้ดังนี้

4.1.1 กิจกรรมการวางแผน

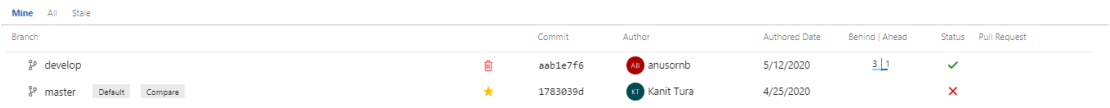
กระบวนการพัฒนาระบบแบบเดิมจะเก็บรวบรวมความต้องการ สร้างแผนภาพและการออกแบบ รวมถึงการติดต่อสื่อสารระหว่างทีมโดยการประชุมภายในทีม แต่ยังขาดเครื่องมือที่ช่วยในการแบ่งงาน ระบบติดตามการทำงานของลูกทีม แต่การพัฒนาระบบแบบใหม่ได้มีเครื่องมือที่เรียกว่า Azure Boards เป็นหน้าบอร์ดกำหนดแผนการทำงานต่าง ๆ เช่นการทำ Product Backlog การทำ Planning ต่าง ๆ โดยจะออกมาเป็นลักษณะของ Kanban boards โดยมีตัวเลือกให้เราหลายแบบเช่นแบบของ Scrum ซึ่งตัวมันสามารถกำหนดงานได้หลายระดับ เช่น Epic > Feature > Story > Task และแต่ละตัวสามารถกำหนดคนรับผิดชอบได้มากกว่า 1 คน และยังสามารถติดตามการทำงานของลูกทีมว่าระบบอะไรดำเนินการเสร็จแล้วบ้าง และยังเหลืออีกกี่ระบบ ทำให้หัวหน้าโครงการสะดวกต่อบริหารจัดการมากยิ่งขึ้น



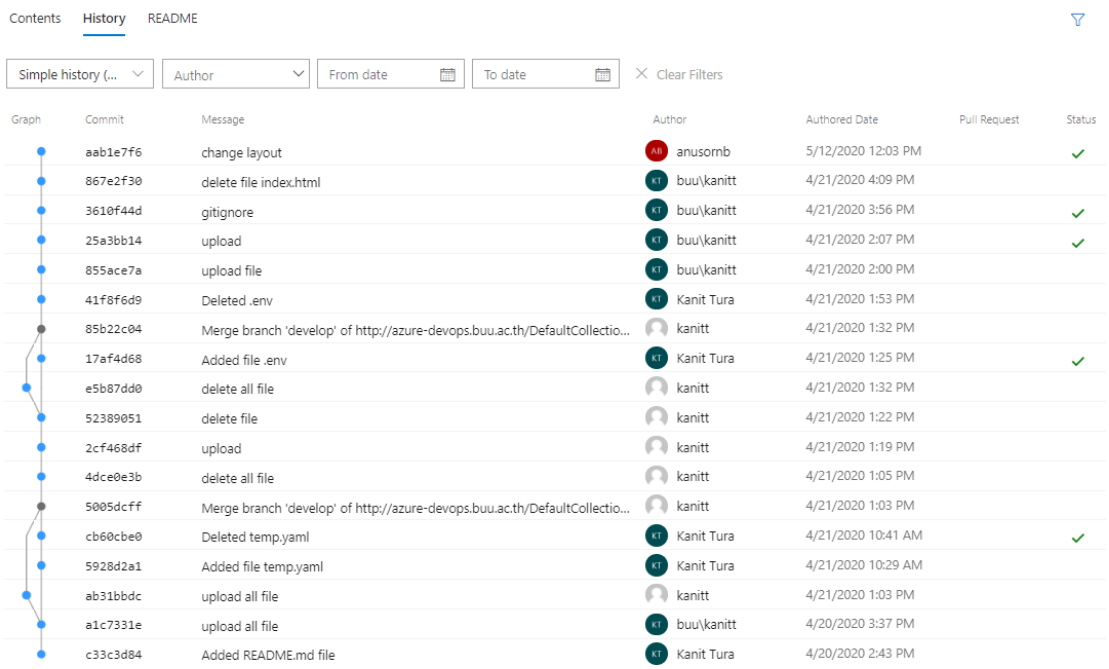
ภาพที่ 4.1 ตัวอย่างการใช้งาน Azure Boards ของกระบวนการพัฒนาระบบแบบใหม่

4.1.2 กิจกรรมการสร้าง

กระบวนการพัฒนาระบบแบบเดิมจะเก็บซอร์สโค้ดบนเครื่องพัฒนาระบบโดยตรง โดยที่ไม่มีระบบจัดเก็บและควบคุมซอร์สโค้ดทำให้ยากต่อบริหารจัดการ แต่การพัฒนาระบบแบบใหม่จะทำงานบนระบบ Version Control ซึ่งสามารถทำงานในไฟล์เดียวกันได้โดยไม่กระทบซึ่งกันและกัน สามารถติดตามย้อนหลังของซอร์สโค้ดว่าทำไปวันที่เท่าไร เวอร์ชั่นนี้มีอะไรบ้าง เวอร์ชั่นใหม่ได้แก้ไขอะไรเพิ่มขึ้นมาบ้าง มีซอร์สโค้ด Master โดยที่สามารถสร้าง Branch ไปทำงานต่อได้และตัว Master ไม่พัง แต่ยังคงลักษณะที่เหมือนแม่ใน Master ไว้ทุกประการ และยังสามารถดูการอัพเดทในทีมได้ ว่าลูกทีม Commit อัพเดทไฟล์อะไรมาบ้าง ใส่ comment แก้ไขอะไรไปแล้ว และเมื่อมีซอร์สโค้ดเสียหายหรือทำงานพลาดเกิดขึ้น ก็สามารถไปย้อนเอาเวอร์ชั่นเก่ากลับมาใช้งานได้โดยซอร์สโค้ดจะยังอยู่บนระบบเสมอ



ภาพที่ 4.2 ตัวอย่าง Branch บนระบบ Version Control ของกระบวนการพัฒนาระบบแบบใหม่

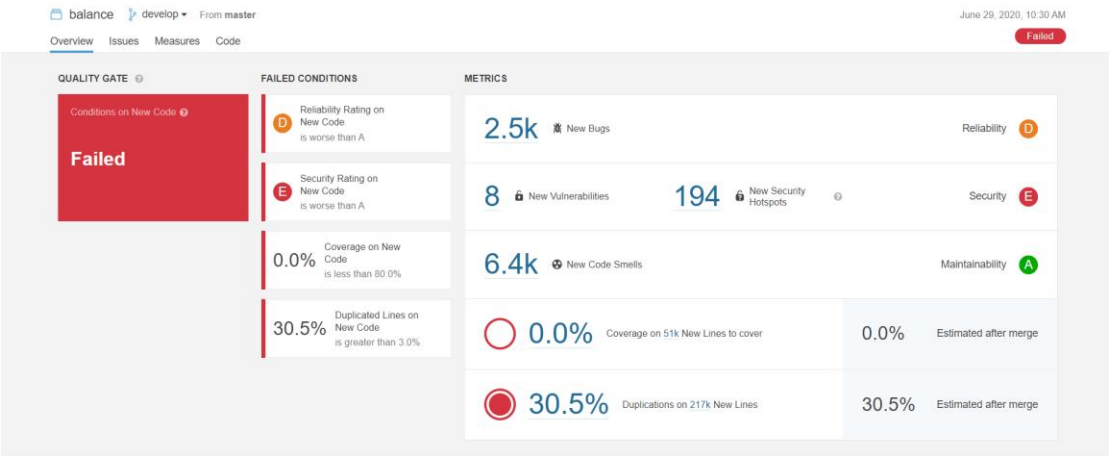


ภาพที่ 4.3 ตัวอย่างการติดตามย้อนหลังบนระบบ Version Control ของกระบวนการพัฒนาระบบแบบใหม่

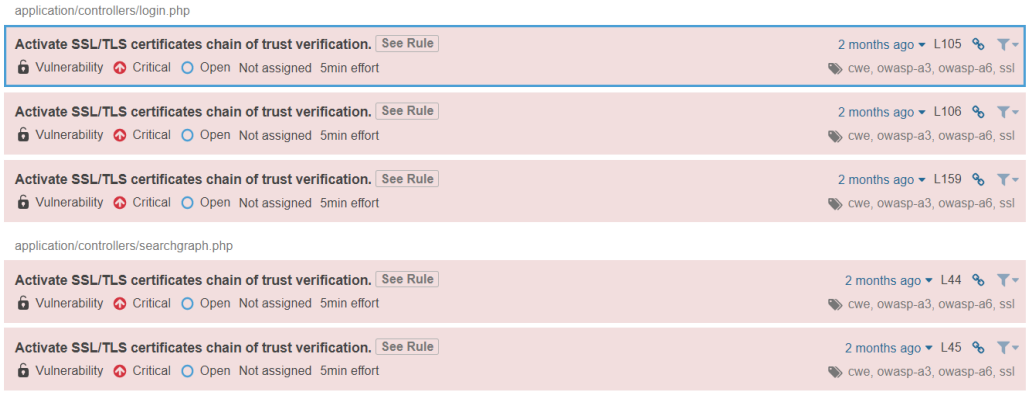
ภาพที่ 4.4 ตัวอย่างการเปรียบเทียบการเปลี่ยนแปลงและการย้อนกลับของซอร์สโค้ดบนระบบ Version Control ของกระบวนการพัฒนาระบบแบบใหม่

4.1.3 กิจกรรมการทดสอบ

กระบวนการพัฒนาระบบแบบเดิมจะทดสอบระบบว่าทำงานได้ถูกต้องได้ผลตามวัตถุประสงค์หรือไม่เพียงอย่างเดียว แต่การพัฒนาระบบแบบใหม่นั้นจะเพิ่มความสามารถในการตรวจสอบคุณภาพของซอร์สโค้ด ช่วยหาข้อบกพร่องในซอร์สโค้ดไม่ว่าจะเป็น Bug ที่น่าจะเกิดขึ้น ช่องโหว่ทางด้านความปลอดภัยหรือซอร์สโค้ดที่จะเป็นปัญหาในอนาคต (Code Smell) และ ช่วยตรวจสอบการเขียนโปรแกรมครอบคลุมหรือดีแล้วหรือไม่ (code coverage)



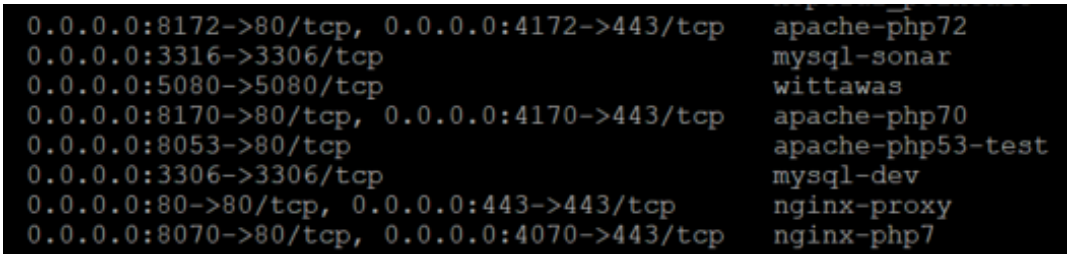
ภาพที่ 4.5 ตัวอย่างการทดสอบคุณภาพของซอร์สโค้ดของกระบวนการพัฒนาระบบแบบใหม่



ภาพที่ 4.6 ตัวอย่างช่องโหว่ของซอร์สโค้ดของกระบวนการพัฒนาระบบแบบใหม่

4.1.4 กิจกรรมการเตรียมก่อนการใช้งานจริง

กระบวนการพัฒนาระบบแบบเดิมจะจัดเก็บไฟล์และสร้างซอฟต์แวร์ขึ้นมาสำหรับติดตั้งบนเครื่องพัฒนาระบบโดยตรง และการเตรียมสภาพแวดล้อมเดียวกันไม่สามารถปรับเปลี่ยนสภาพแวดล้อมเดิมเพราะอาจจะส่งผลกระทบกับระบบอื่น ๆ ได้ แต่การพัฒนาระบบแบบใหม่นั้นสามารถเตรียมสภาพแวดล้อมสำหรับการทดสอบในรูปแบบต่าง ๆ โดยอิสระจากกันและไม่มีการส่งผลกระทบกับระบบอื่น ๆ



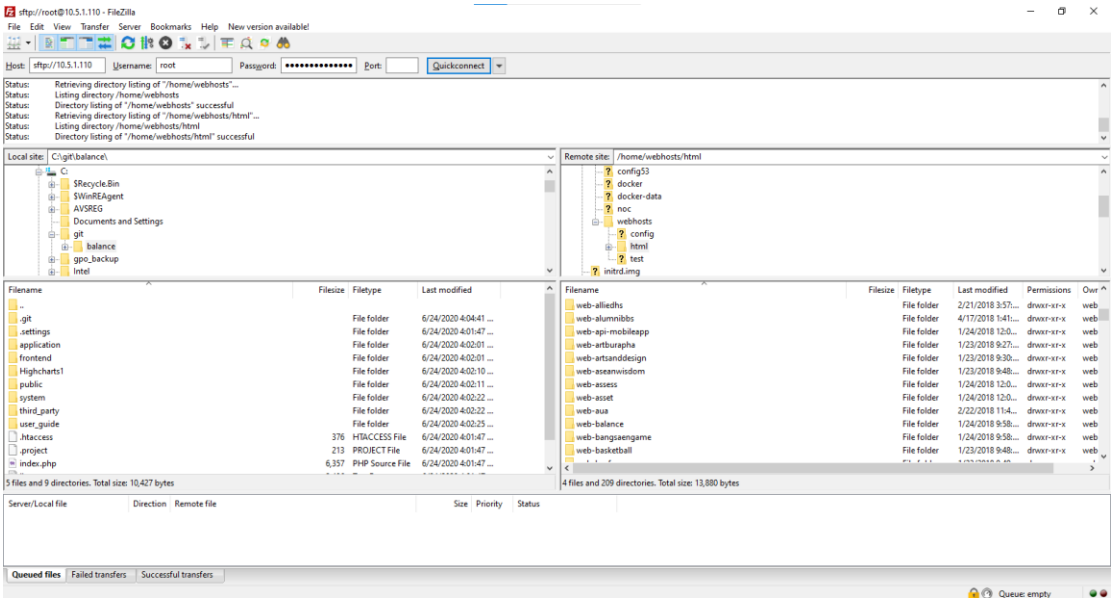
ภาพที่ 4.7 รายการสภาพแวดล้อมเครื่องพัฒนาระบบของระบบแบบเดิม

ภาพที่ 4.8 รายการสภาพแวดล้อมเครื่องพัฒนาระบบของระบบแบบใหม่

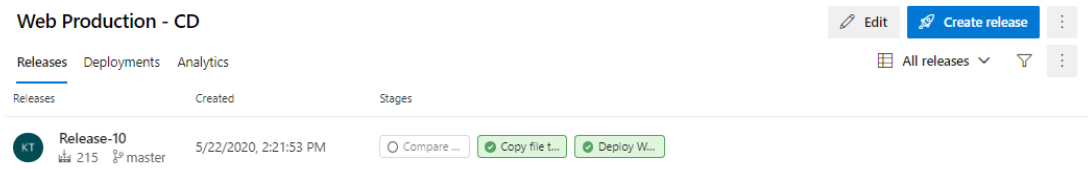
จากภาพที่ 4.7 และ 4.8 จะแสดงให้เห็นว่าการพัฒนาระบบแบบเดิมจะแบ่งสภาพแวดล้อมตามเวอร์ชั่นของ PHP แต่การพัฒนาระบบใหม่จะแยกตามโปรเจค ซึ่งทำให้แต่ละโปรเจคสามารถปรับเปลี่ยนสภาพแวดล้อมได้อย่างอิสระจากกัน

4.1.5 กิจกรรมการส่งมอบ

กระบวนการพัฒนาระบบแบบเดิมจะดำเนินการส่งมอบโดยให้ผู้รับผิดชอบทำการคัดลอกซอร์สโค้ดจากเครื่องพัฒนาระบบไปยังเครื่องใช้งานจริงด้วยตัวเอง ก็อาจส่งผลให้เกิดความล่าช้าและความผิดพลาดเกิดขึ้นได้ กระบวนการพัฒนาระบบแบบใหม่ เป็นกระบวนการในการส่งมอบทำงานได้แบบราบรื่นโดยอัตโนมัติ ทำให้ลดความผิดพลาด ในขณะเดียวกันก็เกิดความรวดเร็วในการทำงานอีกด้วย รวมถึงเตรียมการกู้คืนระบบหากเกิดความล้มเหลวในการส่งมอบ



ภาพที่ 4.9 ตัวอย่างการดำเนินการส่งมอบงานของกระบวนการพัฒนาระบบแบบเดิม

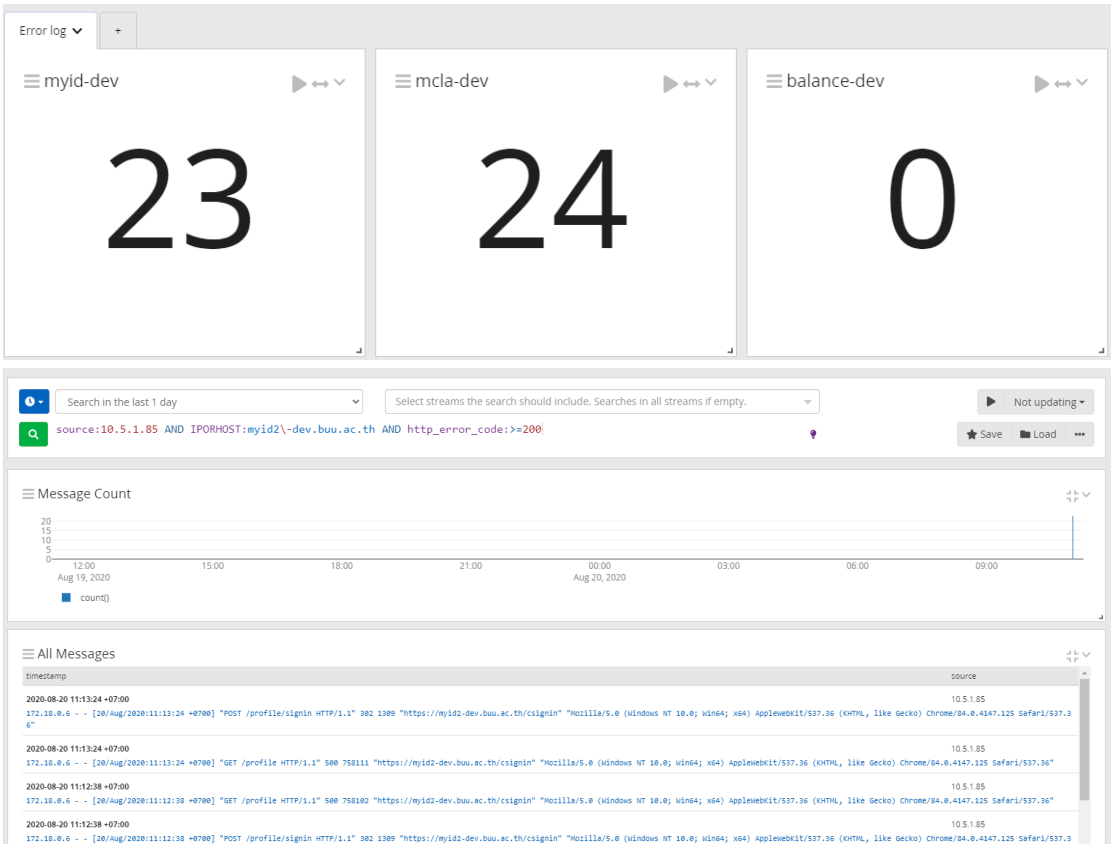


ภาพที่ 4.10 ตัวอย่างการดำเนินการส่งมอบงานของกระบวนการพัฒนาระบบแบบใหม่

4.1.6 กิจกรรมการติดตามผล

กระบวนการพัฒนาระบบแบบเดิมการดูแลและติดตามประสิทธิภาพของซอฟต์แวร์และเซิร์ฟเวอร์ รวมถึงการติดตามปัญหาและการเปลี่ยนแปลงที่เกิดขึ้นบนเครื่องพัฒนาระบบโดยตรง โดยที่ไม่มีระบบจัดเก็บข้อมูลส่วนกลาง ทำให้ผู้พัฒนาระบบยากต่อการค้นหาและเข้าถึง แต่กระบวนการพัฒนาระบบแบบใหม่จะเพิ่มระบบจัดเก็บส่วนกลาง เพื่อรวบรวมข้อมูลจราจรทางคอมพิวเตอร์เอามาไว้ในที่เดียว เพราะการรีโมทเข้าไปอ่านข้อมูลจราจรทางคอมพิวเตอร์ในแต่ละเครื่องเป็นเรื่องที่ยาก โดยระบบจะสามารถตรวจสอบว่าโปรแกรมทำงานได้ปกติหรือไม่ การทำงานมีปัญหาตรงไหน มีความสำคัญอย่างมากในการแก้ไขบั๊กของโปรแกรม

ภาพที่ 4.11 ตัวอย่างการติดตามผลของกระบวนการพัฒนาระบบแบบเดิม

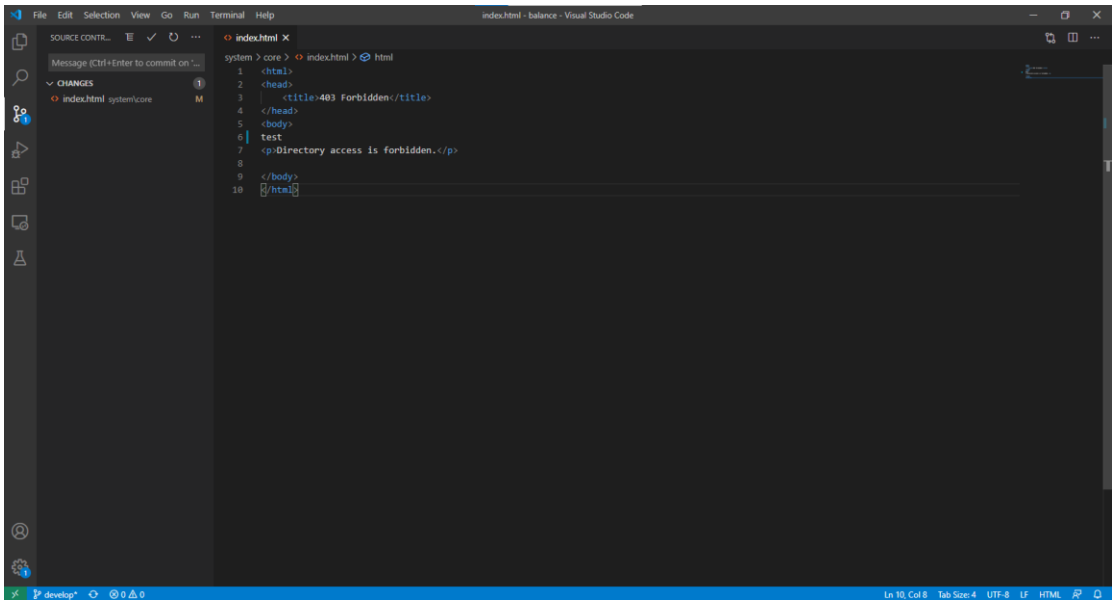


ภาพที่ 4.12 ตัวอย่างการติดตามผลของกระบวนการพัฒนาระบบแบบใหม่

4.2 การส่งมอบงานได้อย่างอย่างรวดเร็วและมีประสิทธิภาพ

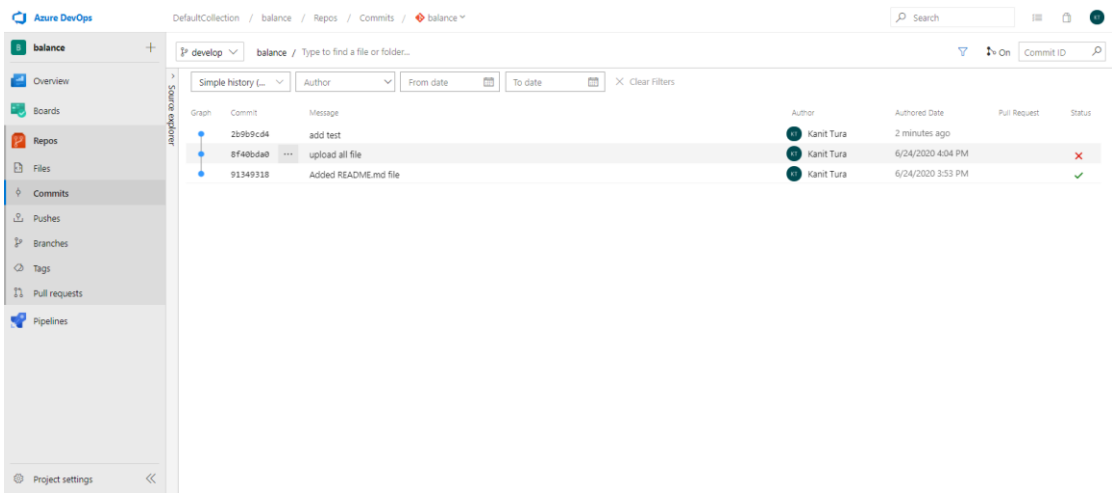
เวลาคือสิ่งที่สำคัญที่สุดในการทำธุรกิจ ดังนั้นการทำซอฟต์แวร์เลยต้องผลิตงานให้เร็วและมีคุณภาพสูงสุดนั่นเอง ดังนั้นจากที่ว่ามา DevOps คือการรวมทุกอย่างที่จำเป็นในการทำซอฟต์แวร์เข้ามาทำงานด้วยกัน โดยชื่อของมันเกิดจากการเอาของ 2 อย่างมารวมกันคือ Software Development (Dev) มารวมกับ Information-Technology Operations (Ops) และการทำ DevOps นั้นจะต้องรวมถึงเรื่องการทำ Agile planning, Continuous Integration (CI), Continuous Delivery (CD) และ Monitoring application อีกด้วย โดยจะแสดงลำดับการพัฒนาระบบจนถึงการส่งมอบงานได้ดังนี้

1. ผู้พัฒนาระบบเมื่อทำการพัฒนา feature เสร็จ จะทำการ build, test และ run บนเครื่องของตัวเอง เพื่อทำให้แน่ใจว่าระบบทำงานได้ถูกต้องและให้แน่ใจว่าสิ่งที่เปลี่ยนแปลงไม่กระทบส่วนอื่น ๆ



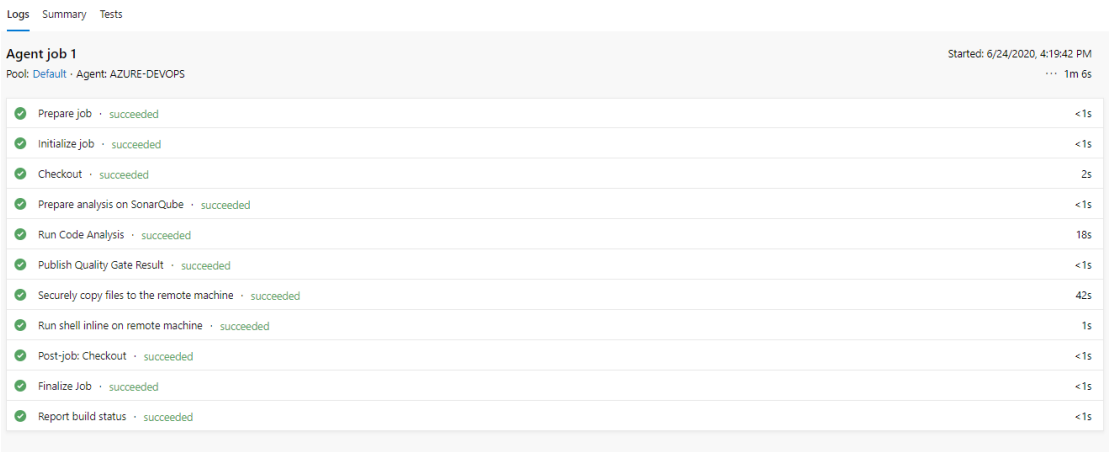
ภาพที่ 4.13 ตัวอย่างโปรแกรม VS CODE ใช้สำหรับทำงานบน Version control

2. ทำการดึง source code ล่าสุดจาก Repository ของระบบ เพื่อตรวจสอบว่ามีการเปลี่ยนแปลงหรือไม่ถ้ามีการเปลี่ยนแปลงก็ให้ทำการรวมหรือ merge ที่เครื่องของผู้พัฒนาระบบก่อน จากนั้นจึงทำการ build, test และ run อีกรอบ เมื่อทุกอย่างผ่านทั้งหมด ให้ทำการส่งการเปลี่ยนแปลงไปยัง Repository กลาง



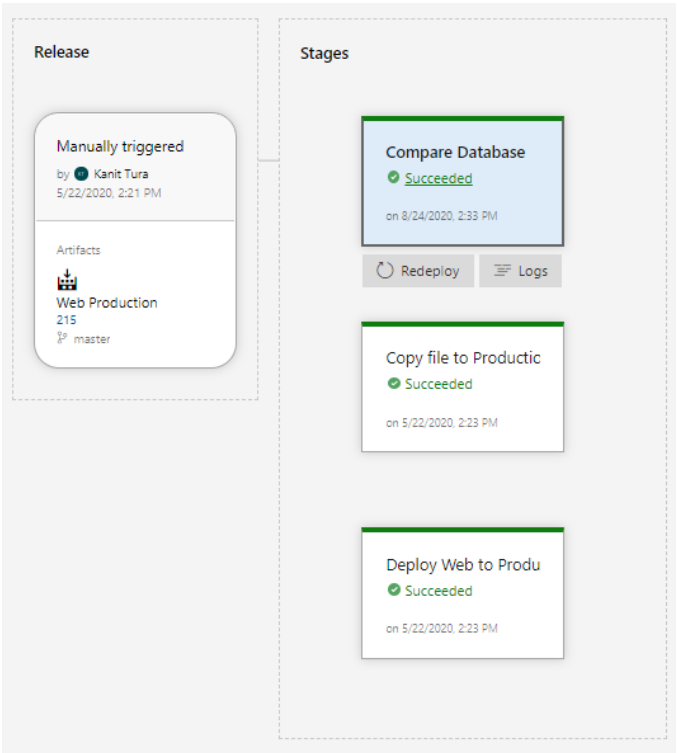
ภาพที่ 4.14 ตัวอย่างการ Commit ซอร์สโค้ดไปยัง Repository กลาง

3. เมื่อ Repository กลางมีการเปลี่ยนแปลง จะต้องมีระบบ CI ทำการ build หลังจาก build จะส่งต่อไปสแกนหาข้อบกพร่องในซอร์สโค้ดไม่ว่าจะเป็น Bug ที่น่าจะเกิดขึ้น ช่องโหว่ทางด้านความปลอดภัยก่อนถ้าผ่านหมดถึงจะส่งต่อไปยังระบบ CD เพื่อส่งซอร์สโค้ดไปยังเครื่องพัฒนาระบบแบบอัตโนมัติ



ภาพที่ 4.15 ตัวอย่างการทำ ci/cd

4. หลังจากทดสอบโปรแกรมบนเครื่องพัฒนาระบบผ่านทั้งหมดจะส่งต่อไปยังระบบ Continuous Delivery ซึ่งดำเนินการส่งซอร์สโค้ดไปยังเครื่อง Production สำหรับส่งมอบให้ลูกค้าโดยอัตโนมัติ

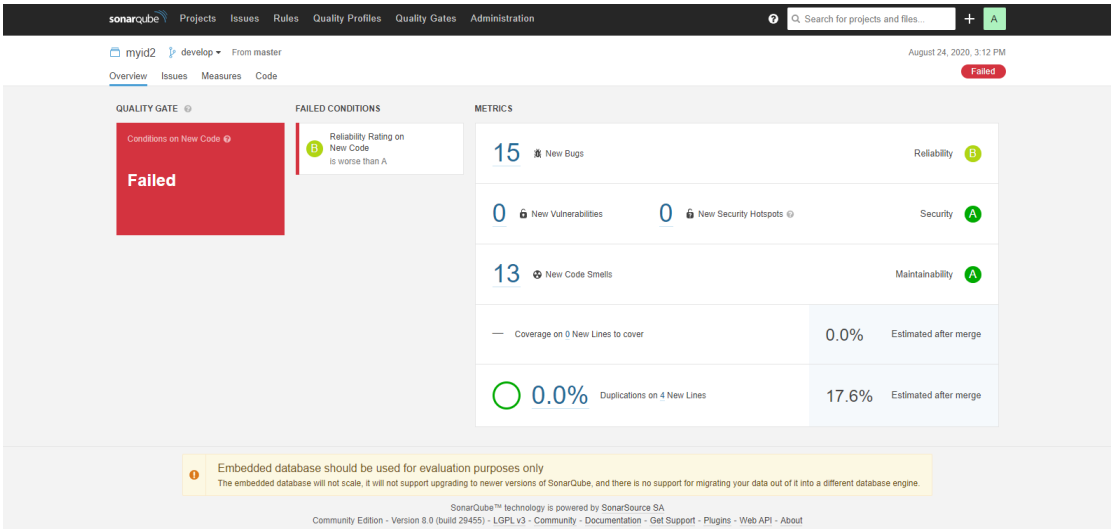


ภาพที่ 4.16 ตัวอย่างการส่งมอบงานให้กับลูกค้า

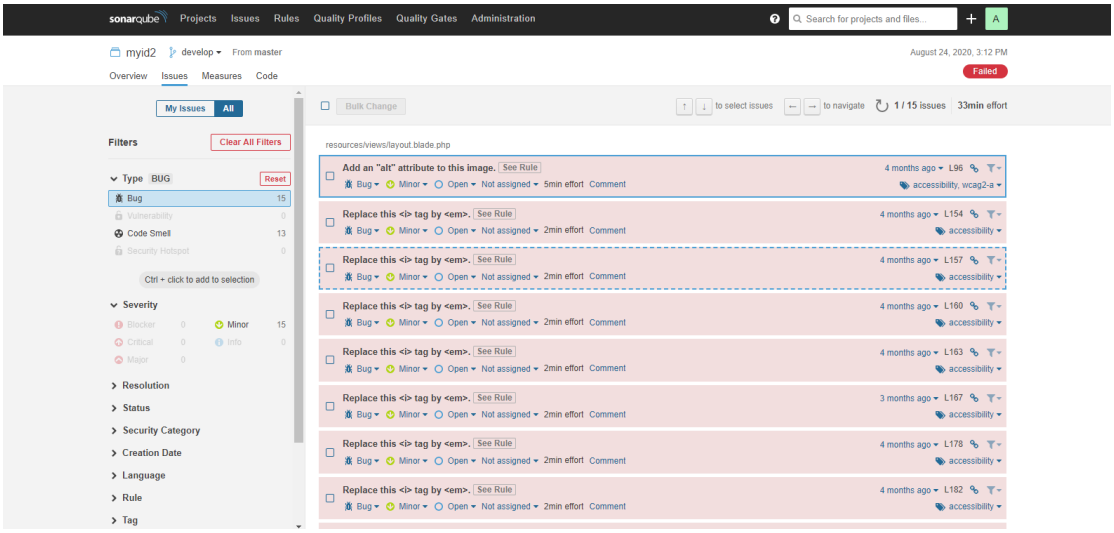
จากขั้นตอนการพัฒนาระบบข้างต้นหากเปรียบเทียบกับกระบวนการพัฒนาระบบแบบเดิมนั้นจะเพิ่มในส่วนของ version control และ ci/cd เพื่อลดระยะเวลาในการพัฒนา ลดต้นทุนในการพัฒนา รองรับการเปลี่ยนแปลงแก้ไขได้ง่าย ดูแลรักษาได้ง่าย เนื่องจากมีการดำเนินไปอย่างต่อเนื่องเป็นไปอย่างอัตโนมัติและทราบผลลัพธ์ได้ทันที ทำให้เราสามารถส่งมอบงานให้กับลูกค้าได้เร็วยิ่งขึ้น

4.3 ความน่าเชื่อถือของซอฟต์แวร์

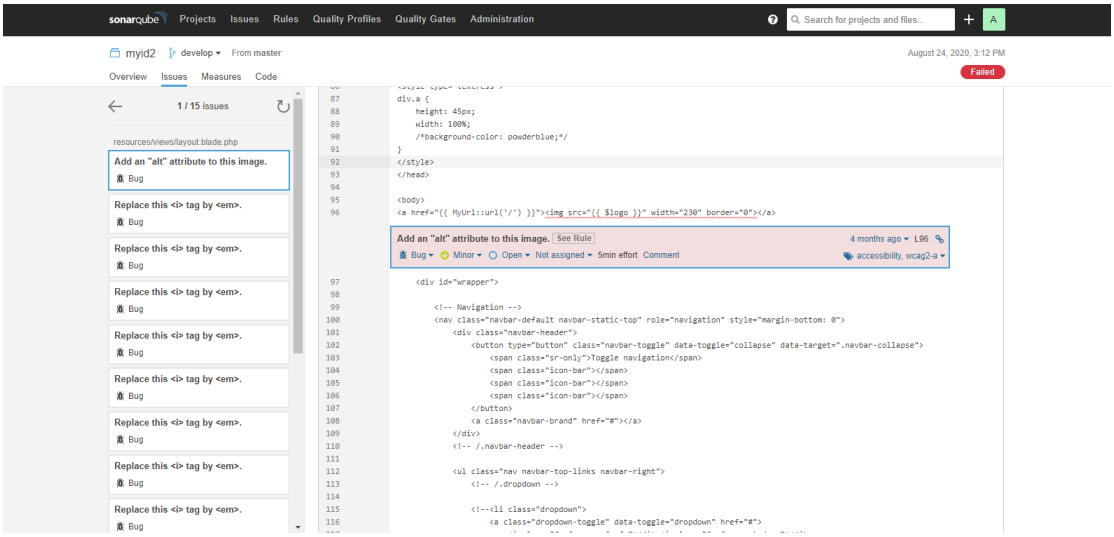
กระบวนการการพัฒนาระบบส่วนที่สำคัญลำดับต้น ๆ ก็คือการทำ code review เพื่อตรวจสอบคุณภาพของซอร์สโค้ด ช่วยหาข้อบกพร่องในซอร์สโค้ดไม่ว่าจะเป็น Bug ที่น่าจะเกิดขึ้น ช่องโหว่ทางด้านความปลอดภัยหรือซอร์สโค้ดที่จะเป็นปัญหาในอนาคต (Code Smell) และ ช่วยตรวจสอบเราเขียนโค้ดทดสอบครอบคลุมหรือดีแล้วยังยัง (code coverage) ซึ่งในกระบวนการการพัฒนาระบบแบบเดิมนั้นไม่มีเครื่องมือที่ใช้ในการตรวจสอบคุณภาพของซอร์สโค้ดทำให้งานที่ส่งให้กับลูกค้าไม่มีความน่าเชื่อถือ แต่ในกระบวนการพัฒนาระบบแบบใหม่นั้นได้เพิ่มเครื่องมือนี้ลงไปในกระบวนการของ Continuous Integration (CI) เพื่อทำการตรวจสอบซอร์สโค้ดก่อนส่งไปยังเครื่องพัฒนาระบบ โดยจะแสดงตัวอย่างการทำงานของ Sonarqube ดังรูป



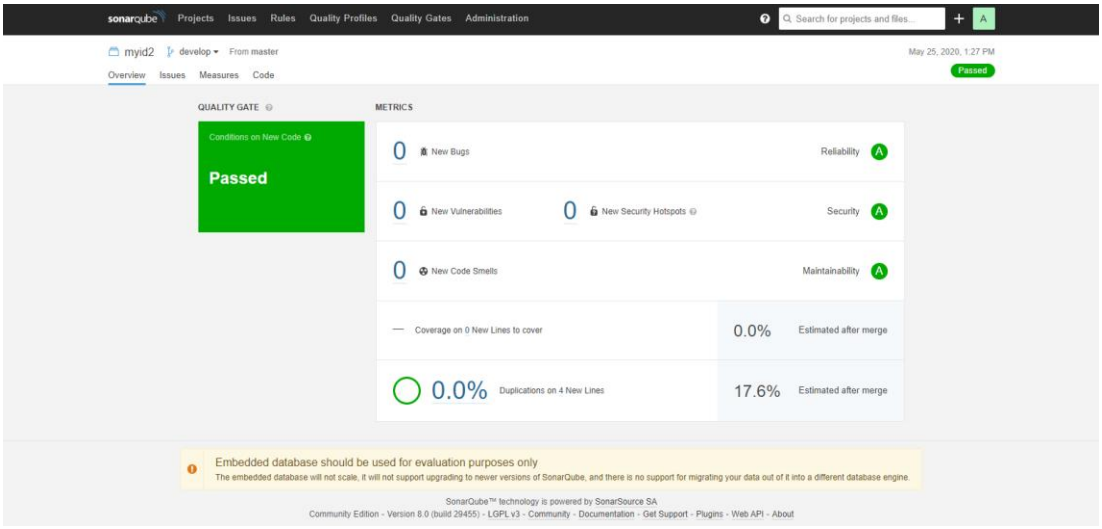
ภาพที่ 4.17 ตัวอย่างผลการตรวจสอบคุณภาพซอร์สโค้ดของกระบวนการพัฒนาระบบแบบเดิม



ภาพที่ 4.18 ตัวอย่างรายละเอียดช่องโหว่ซอร์สโค้ดของกระบวนการพัฒนาระบบแบบเดิม



ภาพที่ 4.19 ตัวอย่างรายละเอียดช่องโหว่ซอร์สโค้ดของกระบวนการพัฒนาระบบแบบเดิม (ต่อ)



ภาพที่ 4.20 ตัวอย่างผลการตรวจสอบซอร์สโค้ดหลังจากดำเนินการแก้ไขช่องโหว่ของซอร์สโค้ด

สรุปผลจากตัวอย่างข้างต้นได้แสดงผลของการตรวจสอบคุณภาพของซอร์สโค้ดก่อนและหลังการแก้ไข ซึ่งจะพบว่าซอร์สโค้ดที่ได้มากจากกระบวนการพัฒนาระบบแบบเดิมนั้นมีช่องโหว่เป็นจำนวนมากทำให้งานที่ส่งมอบไม่มีความน่าเชื่อถือและไม่ปลอดภัย ซึ่งอาจจะส่งผลให้ผู้ไม่หวังดีใช้ช่องโหว่เหล่านี้ในการโจมตีได้

4.4 การลดขั้นตอนการดำเนินงานและจัดการโครงสร้างพื้นฐาน

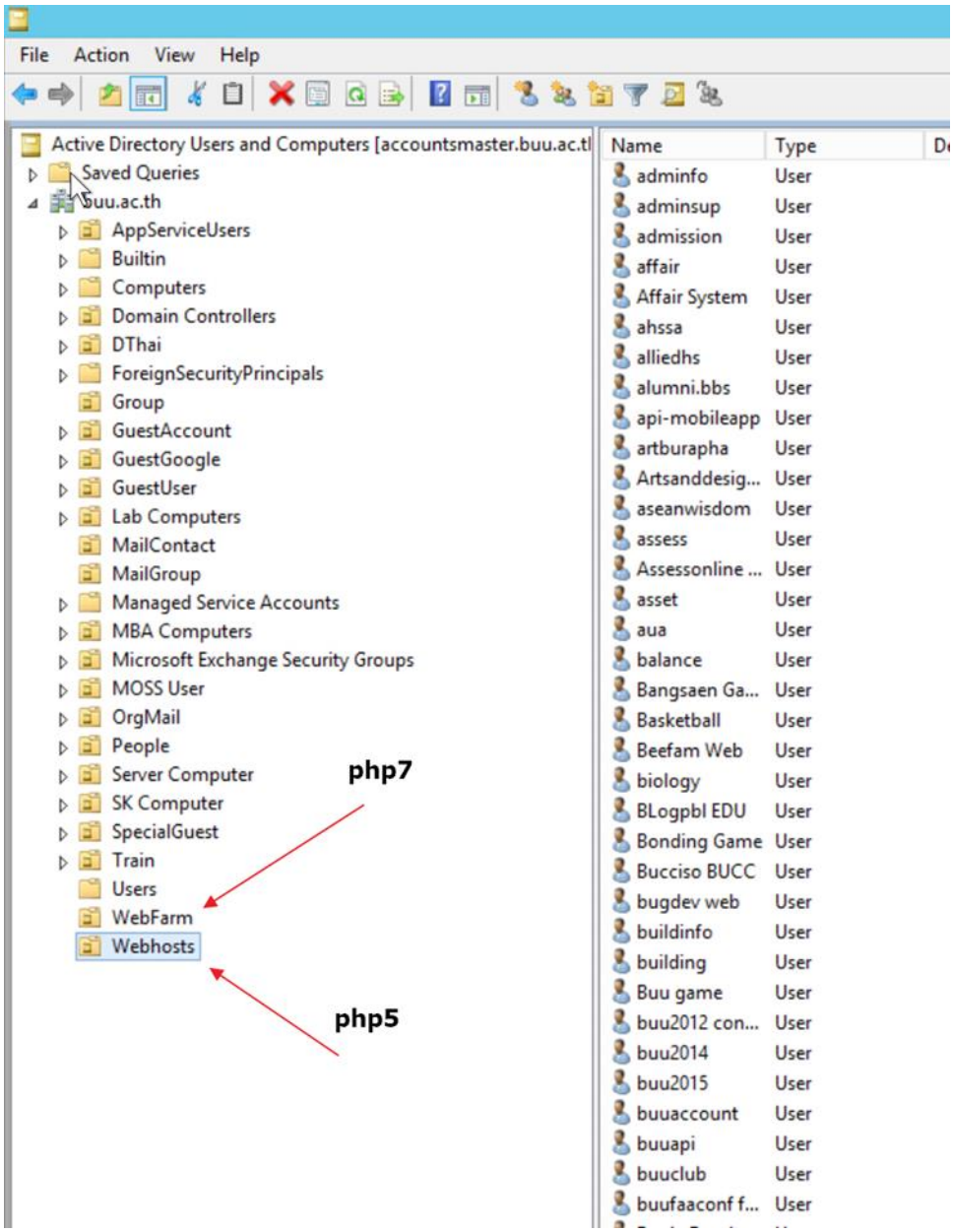
ขั้นตอน	แบบเดิม	แบบใหม่
1	ดำเนินการสร้างบัญชีผู้ใช้งานสำหรับพัฒนาระบบ	ดำเนินการสร้างโปรเจคพัฒนาระบบและกำหนดทีมงาน
2	ดำเนินการกำหนดสิทธิ์ผู้ใช้งานเพื่อให้สามารถเข้าเครื่องพัฒนาระบบได้	ดำเนินการตั้งค่าเริ่มต้นสำหรับโปรเจคพัฒนาระบบ
3	ดำเนินการสร้างไดเรคทอรี่สำหรับเก็บซอร์สโค้ดและกำหนดสิทธิ์การเข้าถึง	ดำเนินการจด DNS สำหรับเรียกใช้งานผ่านชื่อโดเมน
4	ดำเนินการสร้าง virtual host สำหรับเรียกใช้งานระบบที่ได้ทำการพัฒนา	
5	ดำเนินการเปิดใช้งาน virtual host เพื่อเรียกใช้งานระบบที่ทำการพัฒนาได้	
6	ดำเนินการรีสตาร์ทเซอร์วิสเว็บเซิร์ฟเวอร์	
7	ดำเนินการจด DNS สำหรับเรียกใช้งานผ่านชื่อโดเมน	

ตารางที่ 4.2 เปรียบเทียบจำนวนขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐาน

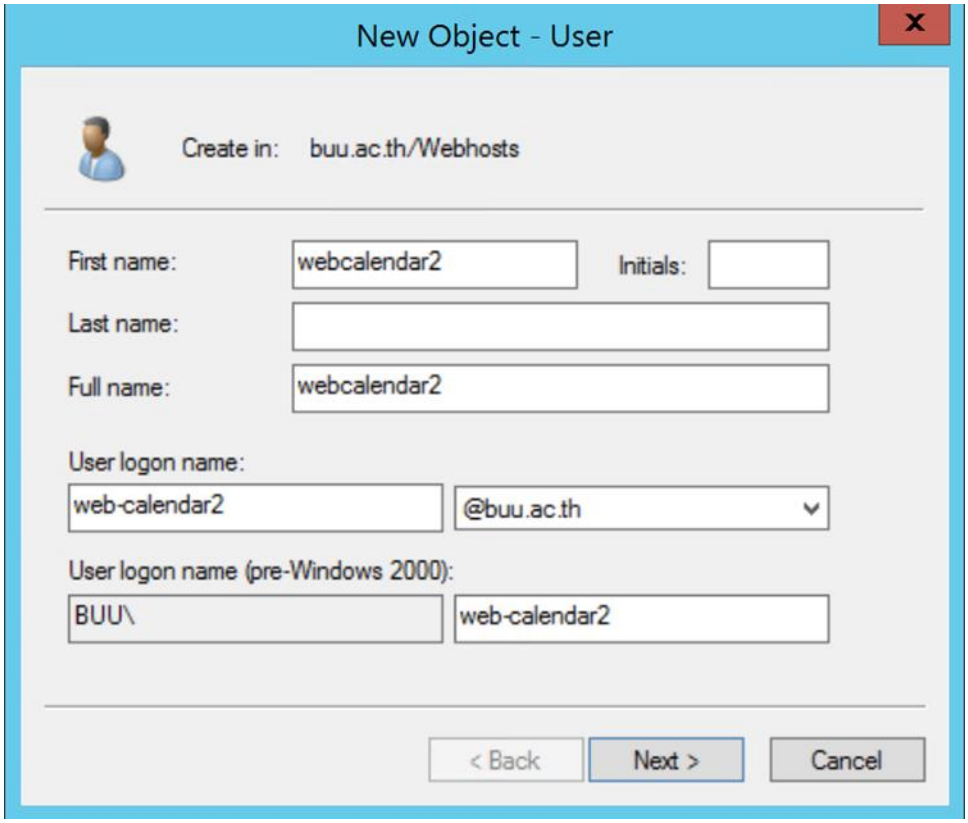
จากตารางที่ 4.2 จะแสดงความแตกต่างระหว่างการพัฒนาระบบแบบเดิมกับการพัฒนาระบบแบบใหม่ ซึ่งจะเปรียบเทียบจำนวนขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐาน ซึ่งจะแสดงรายละเอียดขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐานระหว่างการพัฒนาระบบแบบเดิมและระบบแบบใหม่ดังนี้ดังนี้

4.4.1 ขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐานแบบเดิม

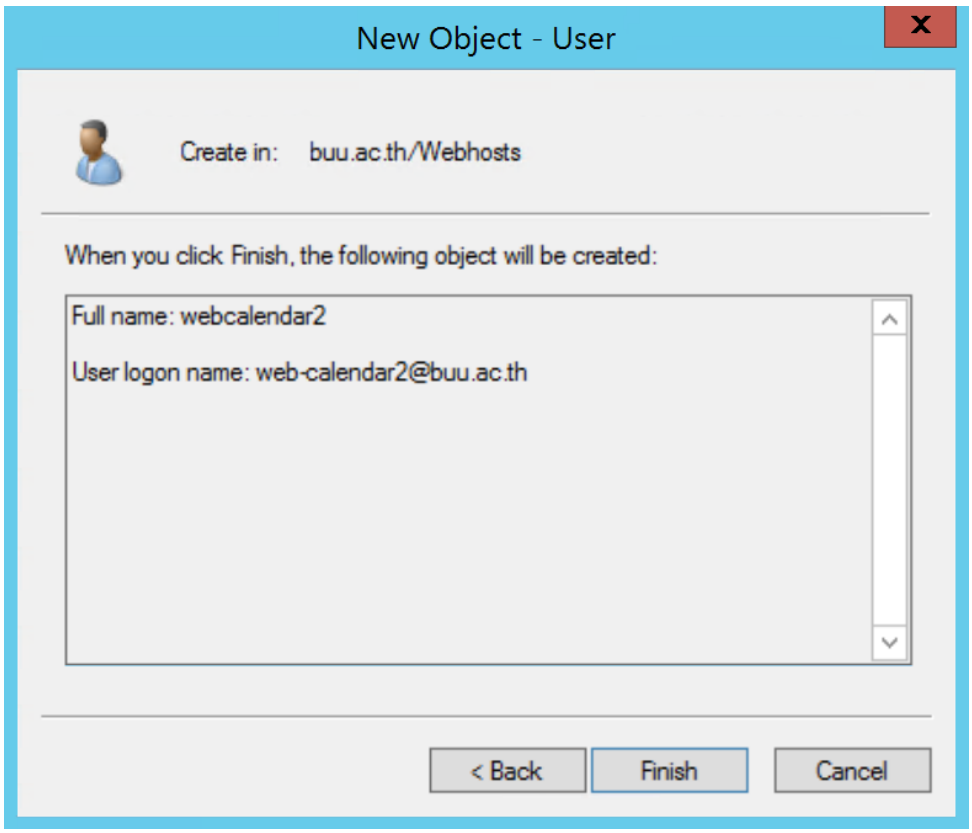
1.ดำเนินการสร้างบัญชีผู้ใช้งานสำหรับพัฒนาระบบบนเครื่องบริหารจัดบัญชีผู้ใช้ โดยระบุชื่อบัญชีผู้ใช้และรหัสผ่านสำหรับเข้าใช้งานเครื่องพัฒนาระบบดังนี้



ภาพที่ 4.21 การสร้างบัญชีผู้ใช้งานสำหรับพัฒนาระบบบนเครื่องบริหารจัดบัญชีผู้ใช้



ภาพที่ 4.22 การสร้างบัญชีผู้ใช้งานสำหรับพัฒนาระบบบนเครื่องบริหารจัดบัญชีผู้ใช้ (ต่อ)

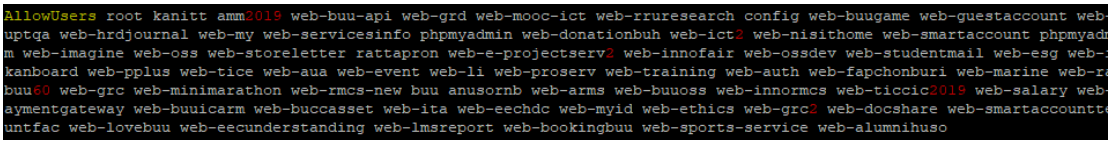


ภาพที่ 4.23 การสร้างบัญชีผู้ใช้งานสำหรับพัฒนาระบบบนเครื่องบริหารจัดบัญชีผู้ใช้ (ต่อ)

2. ดำเนินการกำหนดสิทธิ์ผู้ใช้งาน โดยกำหนดให้บัญชีผู้ใช้ใหม่สามารถเข้าเครื่องพัฒนาระบบได้ ดังนี้

2.1) เข้าสู่ระบบที่เครื่องพัฒนาระบบ

2.2) แก้ไขไฟล์ /etc/ssh/sshd_config โดยการเพิ่มบัญชีผู้ใช้ที่สร้างใหม่ลงในส่วนของ AllowUsers



ภาพที่ 4.24 การกำหนดสิทธิ์ผู้ใช้งานให้สามารถเข้าเครื่องพัฒนาระบบได้

2.3) หลักจากแก้ไขไฟล์ sshd_config ให้ดำเนินการรีสตาร์ทเซอร์วิส sshd

ภาพที่ 4.25 การรีสตาร์ทเซอร์วิส sshd

3. ดำเนินการสร้างไดเรคทอรี่สำหรับเก็บซอร์สโค้ดและกำหนดสิทธิ์การเข้าถึงตาม CMS ที่ใช้ในการพัฒนาระบบดังนี้

3.1) ใช้คำสั่ง chmod 715 <ชื่อไดเรคทอรี่ของระบบ>

3.2) กรณีที่ใช้ CMS ของ Laravel ให้ดำเนินการสร้างไดเรคทอรี่ public_html/public

3.3) กรณีที่ใช้ CMS ของ YII ให้ดำเนินการสร้างไดเรคทอรี่ public_html/web

3.4) กรณีที่ไม่ใช้ CMS ให้ดำเนินการสร้างไดเรคทอรี่ public_html

4. ดำเนินการสร้าง virtual host สำหรับเรียกใช้งานระบบที่ได้ทำการพัฒนาได้

```
server {
        listen 80;
        listen [::]:80;

        root /home/webhosts/web-id/public_html/public/;
        index index.php index.html;

        server_name id.buu.ac.th;

        access_log /var/log/nginx/id.buu.ac.th_access.log;
        error_log /var/log/nginx/id.buu.ac.th_error.log;

        location / {
#               try_files $uri $uri/ =404;
                try_files $uri $uri/ /index.php?$query_string;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;

                # With php7.0-cgi alone:
                #fastcgi_pass 127.0.0.1:9000;
                # With php7.0-fpm:
                fastcgi_pass unix:/run/php/php7.0-fpm.sock;
        }
}
```

ภาพที่ 4.26 ตัวอย่างการสร้าง virtual host สำหรับเรียกใช้งานระบบ

5. ดำเนินการเปิดใช้งาน virtual host เพื่อเรียกใช้งานระบบที่ทำการพัฒนาได้

```
root@webhost2-01:/etc/nginx/sites-enabled# ln -s ../sites-available/id .
```

ภาพที่ 4.27 ตัวอย่างการเปิดใช้งาน virtual host เพื่อเรียกใช้งานระบบที่ทำการพัฒนา

6. ดำเนินการรีสตาร์ทเซอร์วิสเว็บเซิร์ฟเวอร์

```
root@webhost2-01:~# service nginx restart
```

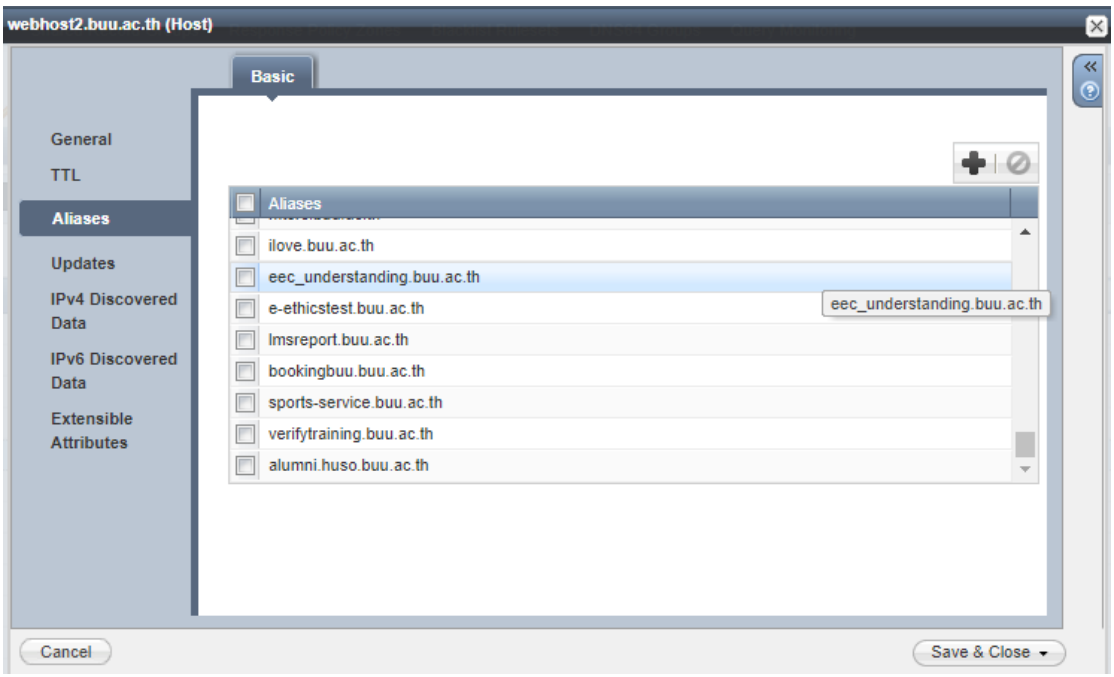
ภาพที่ 4.28 ตัวอย่างการรีสตาร์ทเซอร์วิสเว็บเซิร์ฟเวอร์

7. ดำเนินการจด DNS สำหรับเรียกใช้งานผ่านชื่อโดเมนโดยมีขั้นตอนดังนี้

7.1) เข้าสู่ระบบบริหารจัดการ DNS

ภาพที่ 4.29 ตัวอย่างการเข้าสู่ระบบบริหารจัดการ DNS

7.2) เพิ่มชื่อของโดเมนของระบบที่ทำการพัฒนา

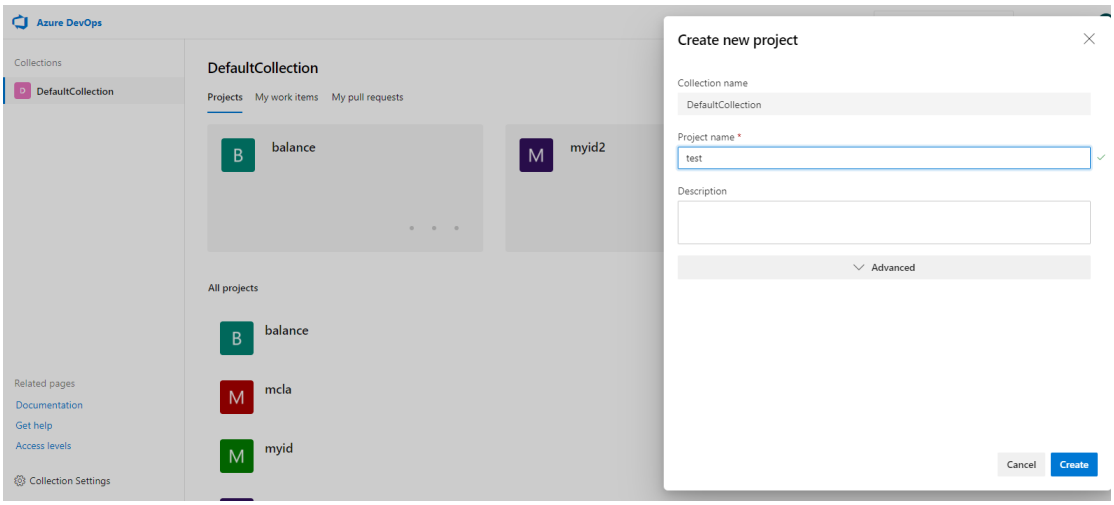


ภาพที่ 4.30 ตัวอย่างการเพิ่มชื่อโดเมนของระบบที่ทำการพัฒนา

4.4.2 ขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐานแบบใหม่

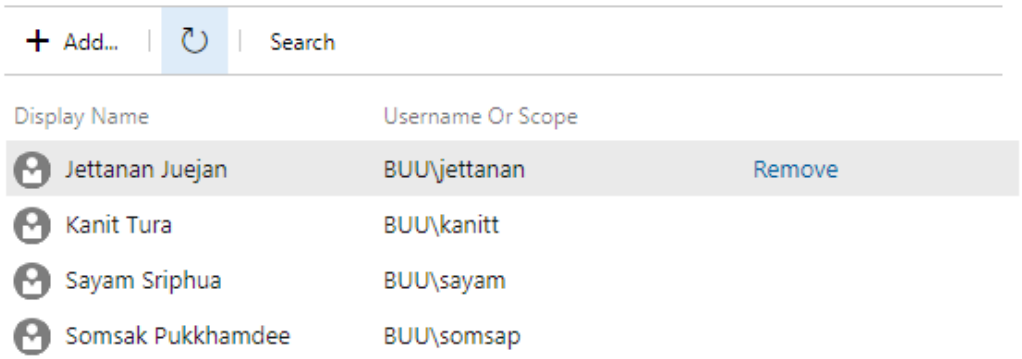
1. ดำเนินการสร้างโปรเจคพัฒนาระบบและกำหนดทีมงานได้ดังนี้

1.1) ระบุชื่อของโปรเจคดังรูป



ภาพที่ 4.31 ตัวอย่างการสร้างโปรเจค

1.2) กำหนดสมาชิกของโปรเจคดังรูป



ภาพที่ 4.32 ตัวอย่างการกำหนดสมาชิกของโปรเจค

2. ดำเนินการตั้งค่าเริ่มต้นสำหรับโปรเจคพัฒนาระบบซึ่งจะมีรายละเอียดดังนี้

- containerName : ชื่อของ Container
- frontendName : ชื่อเว็บไซต์ในการเรียกใช้งาน
- path : ที่อยู่ของไฟล์ ซึ่งจะแยกตาม CMS ที่ใช้งาน
- phpVersion : เวอร์ชั่นของ PHP ที่จะใช้งาน

Name ↑	Value	Settable at que
containerName	mcla-dev	
frontendName	mcla-dev.buu.ac.th	
path		
phpVersion	bucchub/php73:version1	
system.collectionId	68828b3b-7a9c-4256-8094-9167e880d512	
system.debug	false	☑
system.definitionId	32	
system.teamProject	mcla	

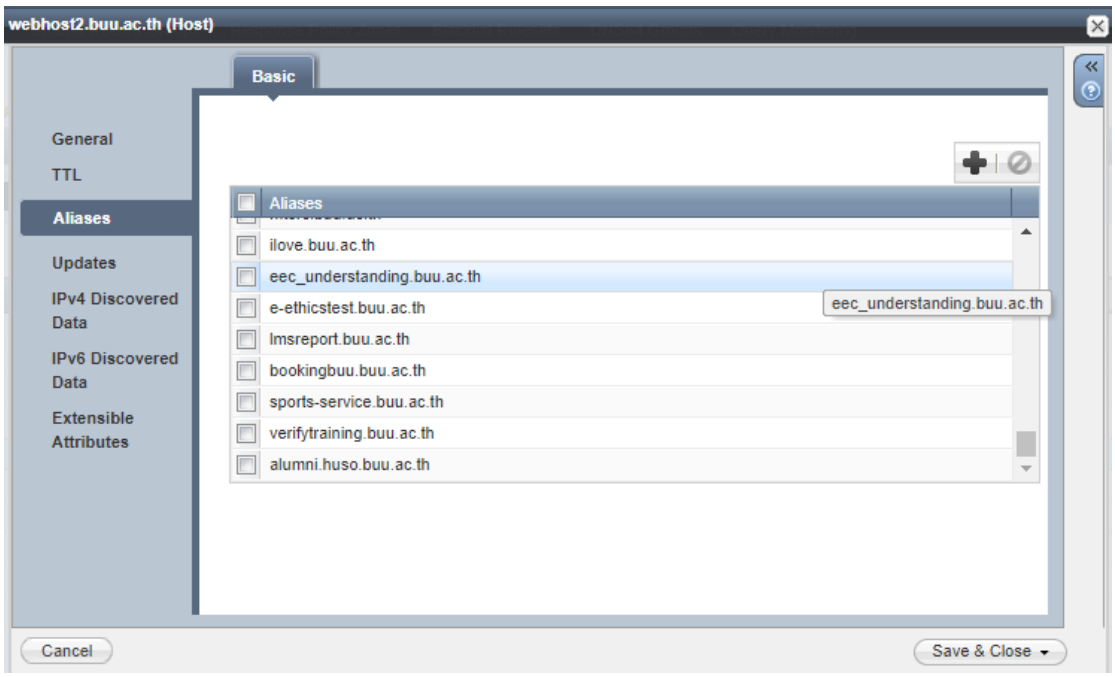
ภาพที่ 4.33 ตัวอย่างการตั้งค่าเริ่มต้นสำหรับโปรเจคพัฒนาระบบ

3. ดำเนินการจด DNS สำหรับเรียกใช้งานผ่านชื่อโดเมนโดยมีขั้นตอนดังนี้

3.1) เข้าสู่ระบบบริหารจัดการ DNS

ภาพที่ 4.34 ตัวอย่างการเข้าสู่ระบบบริหารจัดการ DNS

3.2) เพิ่มชื่อของโดเมนของระบบที่ทำการพัฒนา



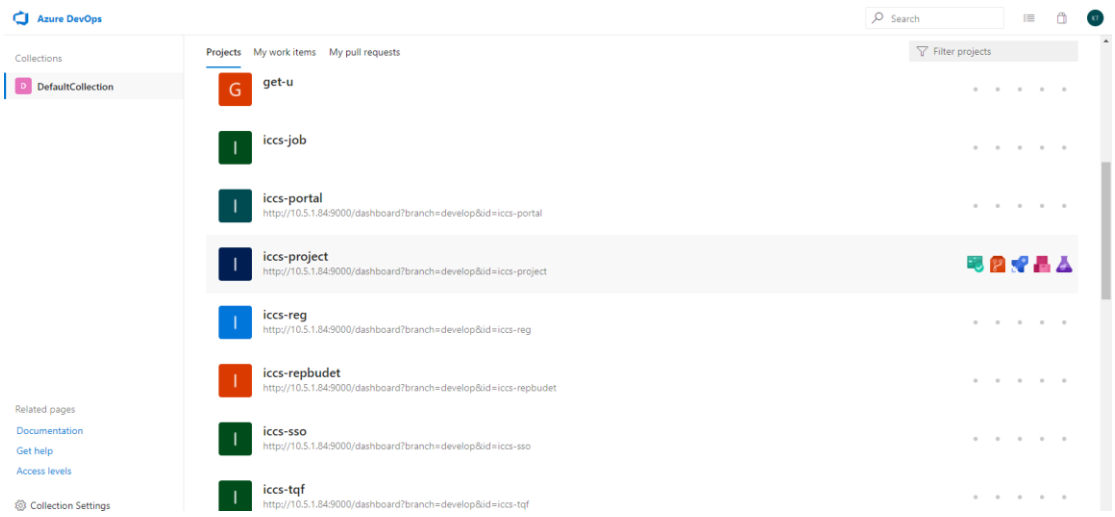
ภาพที่ 4.35 ตัวอย่างการเพิ่มชื่อโดเมนของระบบที่ทำการพัฒนา

สรุปผลจากตัวอย่างข้างต้นได้แสดงให้เห็นว่ากระบวนการพัฒนาระบบแบบใหม่นั้นได้ลดจำนวนขั้นตอนในการดำเนินงานและจัดการโครงสร้างพื้นฐานลง ลดการใช้คำสั่งต่าง ๆ ด้วยมือในการดำเนินงาน ซึ่งกระบวนการพัฒนาระบบแบบใหม่นั้นทางผู้ดูแลระบบจะกำหนดค่าพื้นฐานต่าง ๆ เท่านั้น โดยส่วนที่เหลือจะทำงานเบื้องหลังแบบอัตโนมัติทั้งหมด เพื่อลดข้อผิดพลาดในการทำงานจากคน

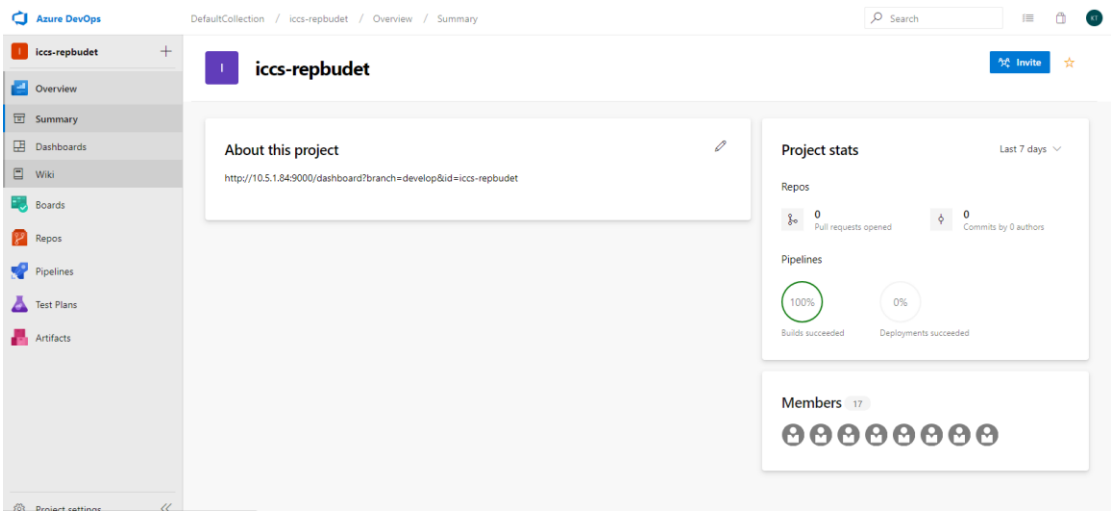
4.5 การทดสอบกระบวนการทำงานจริงร่วมกับฝ่ายพัฒนาระบบ

หลังจากผู้วิจัยได้ดำเนินการจัดทำระบบ Devops เสร็จสมบูรณ์แล้ว ผู้วิจัยได้ทำการเปิดอบรมให้ความรู้ร่วมกับบุคลากรของฝ่ายพัฒนาระบบเพื่อให้รู้วิธีการใช้งานเบื้องต้นและดำเนินการสร้างโปรเจคสำหรับระบบจริง โดยผู้วิจัยได้ทำการปรับปรุงกระบวนการต่างๆ และ Pipeline ของ Devops เพื่อให้สอดคล้องกับการทำงานของฝ่ายพัฒนาระบบ ดังนั้นจึงขอยกตัวอย่างกระบวนการดำเนินการและการตั้งค่าของระบบจริงได้ดังนี้

4.5.1 สร้างโปรเจค



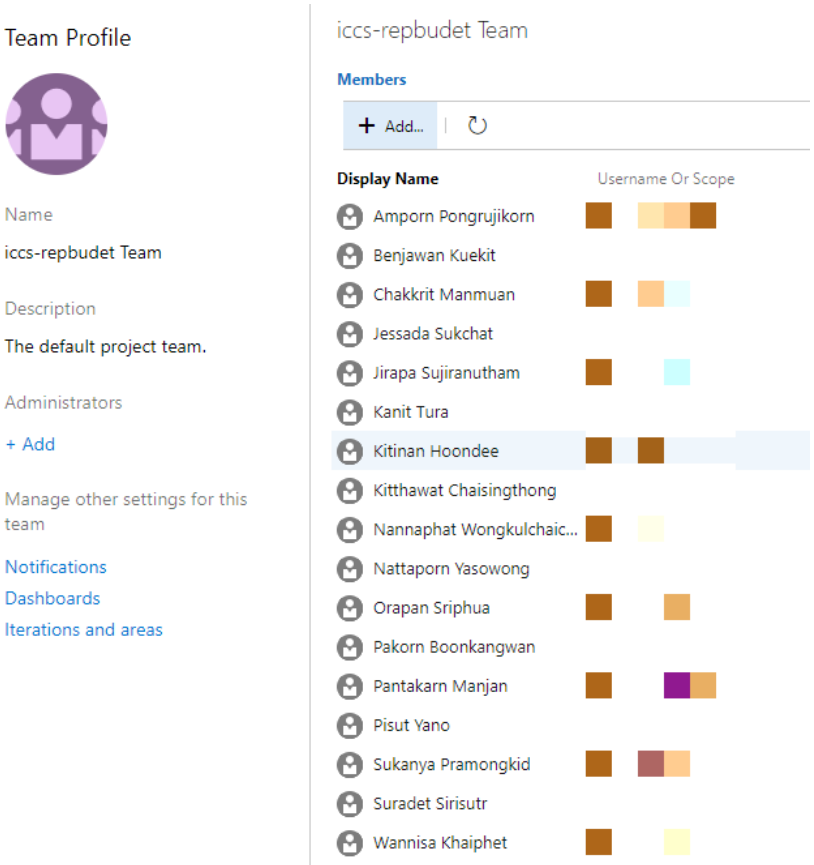
ภาพที่ 4.36 ตัวอย่างรายชื่อโปรเจคทั้งหมด



ภาพที่ 4.37 ตัวอย่างโปรเจคที่ใช้งานจริง

4.5.2 กำหนดสมาชิกและสิทธิ์ของโปรเจค

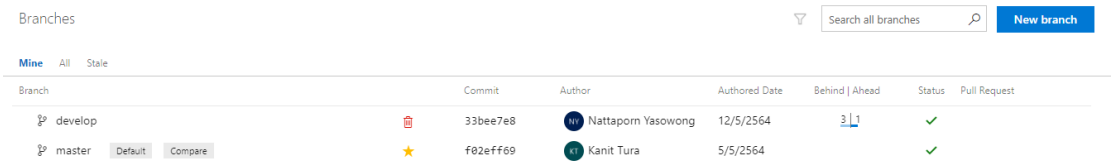
จากภาพข้างล่างแสดงถึงรายชื่อของสมาชิกที่สามารถเข้ามาใช้งานของโปรเจคได้



ภาพที่ 4.38 ตัวอย่างรายชื่อสมาชิกของโปรเจคที่ใช้งานจริง

4.5.3 สร้าง Branches สำหรับระบบทดสอบและระบบจริง

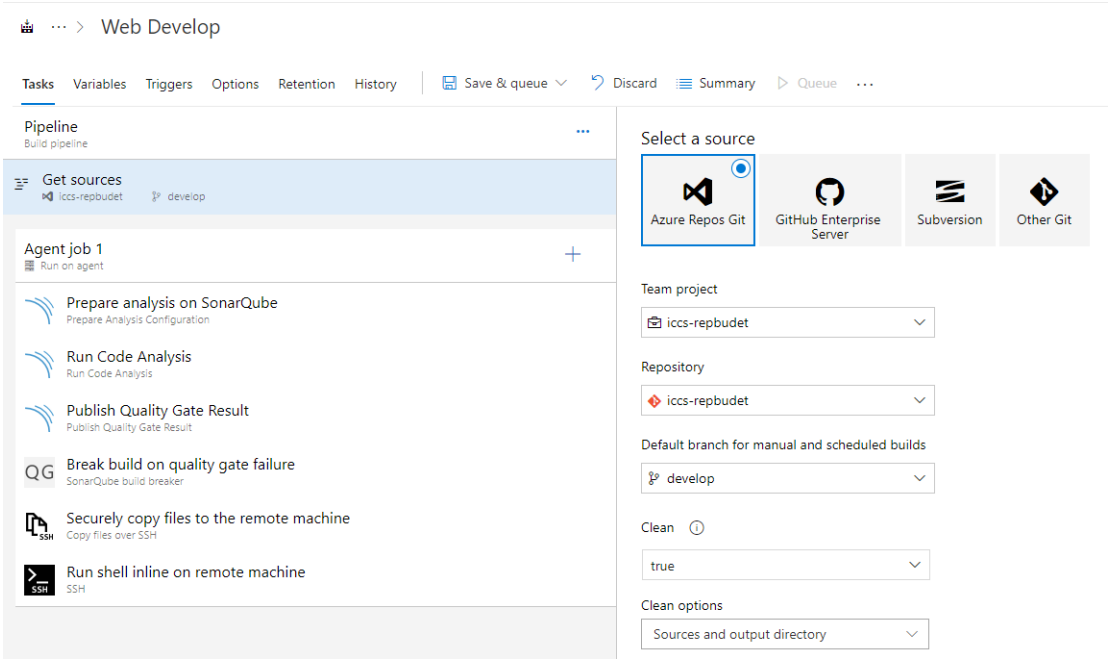
หลังจากสร้างโปรเจคและกำหนดสิทธิ์การใช้งานภายในทีมแล้ว ได้ดำเนินการสร้าง branches จำนวนสอง Branches ได้แก่ branches develop ใช้สำหรับพัฒนาโปรแกรมและใช้ในการทดสอบการทำงานของโปรแกรมรวมถึงตรวจสอบคุณภาพของซอร์สโค้ด (Source code) ส่วน branches master ใช้สำหรับ release ซอร์สโค้ด (Source code) ไปยังเครื่องที่ใช้งานจริง (Production)



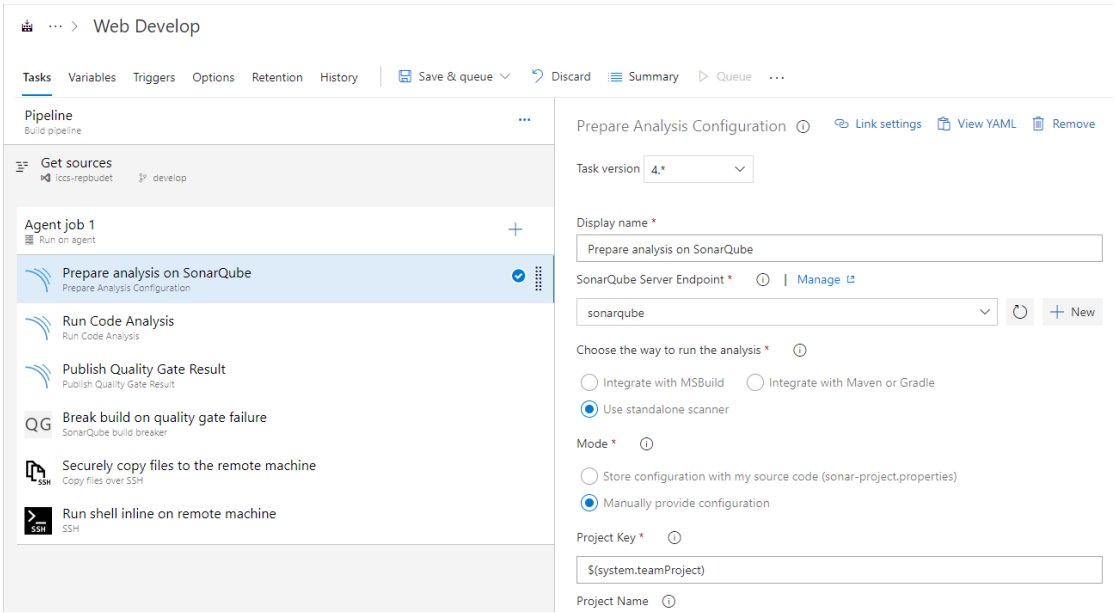
ภาพที่ 4.39 รายชื่อ Branches ทั้งหมดของโปรเจค

4.5.4 สร้าง Pipeline สำหรับระบบทดสอบและระบบจริง

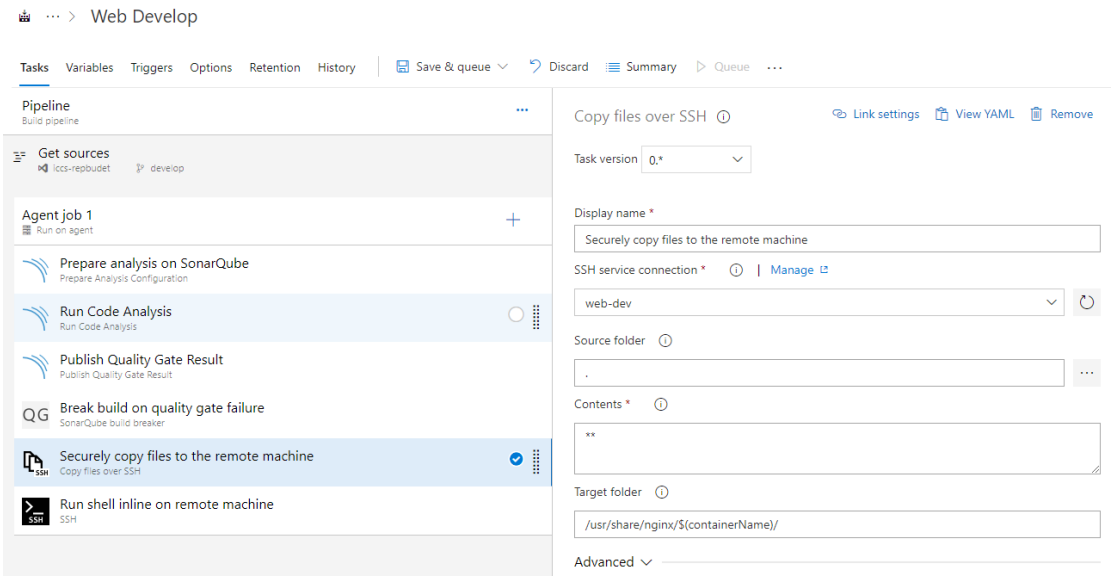
Pipeline การทำงานจะถูกแยกเป็น 2 ส่วน โดยส่วนแรกจะเป็น Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องทดสอบระบบ (Develop) เพื่อตรวจสอบคุณภาพของซอร์สโค้ด (Source code) และทดสอบการทำงานของโปรแกรมนั้นทำงานถูกต้องหรือไม่ และส่วนที่สองจะเป็น Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องใช้งานจริง (Production) โดยมีการเพิ่มส่วนของ Pipeline การแก้ไขการกำหนดค่าต่างๆ ของซอร์สโค้ด เนื่องจากว่าตัวแปรต่างๆ ของโปรแกรมระหว่างเครื่องทดสอบระบบและเครื่องใช้งานจริงอาจจะไม่เหมือนกัน เช่น แก้ไขบัญชีผู้ใช้และรหัสผ่านของฐานข้อมูลโดยอัตโนมัติ เพื่อลดขั้นตอนการแก้ไขการกำหนดค่าและลดข้อผิดพลาดการทำงานด้วยตัวเอง ทำให้สะดวกต่อการทำงานของฝ่ายพัฒนาระบบ



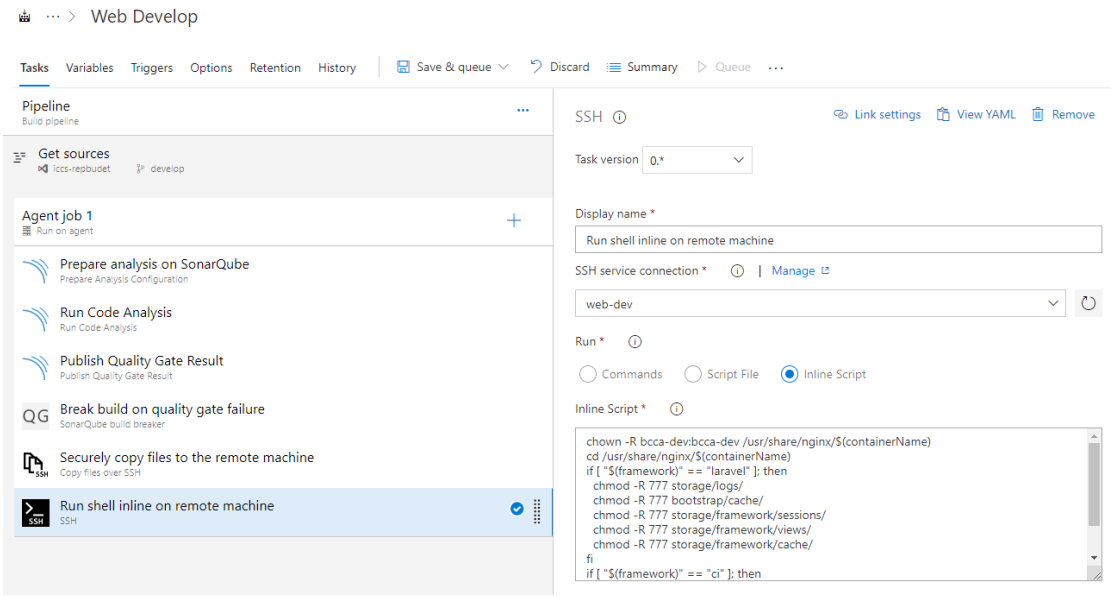
ภาพที่ 4.40 Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องทดสอบระบบ (Develop)



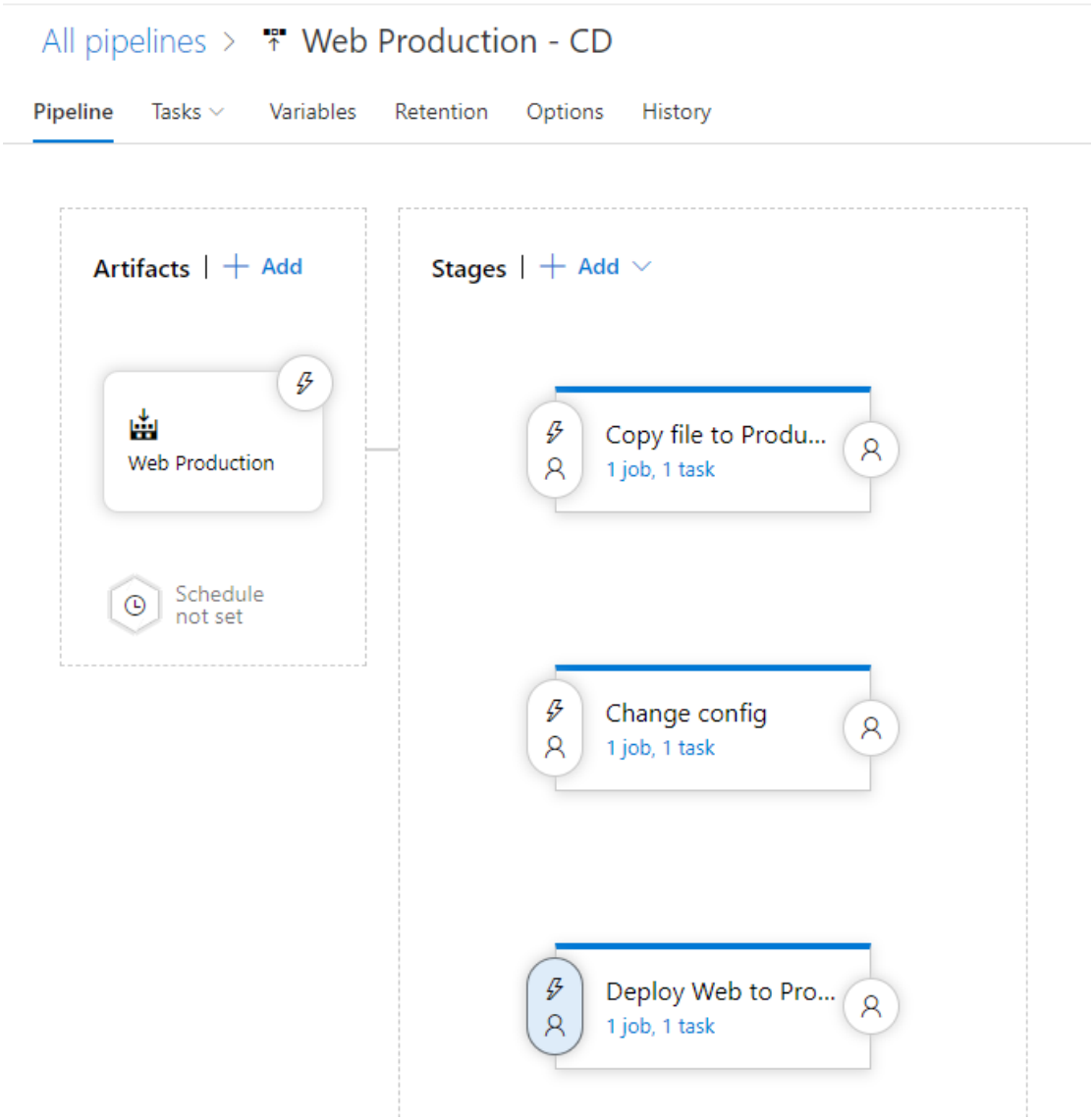
ภาพที่ 4.41 Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องทดสอบระบบ (Develop) (ต่อ)



ภาพที่ 4.42 Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องทดสอบระบบ (Develop) (ต่อ)



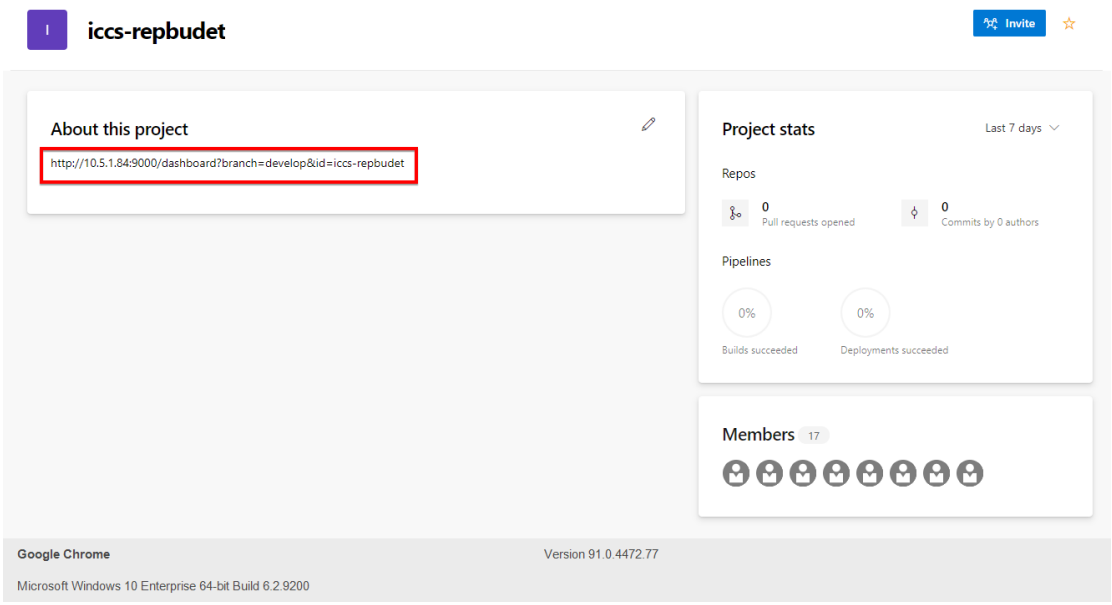
ภาพที่ 4.43 Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องทดสอบระบบ (Develop) (ต่อ)



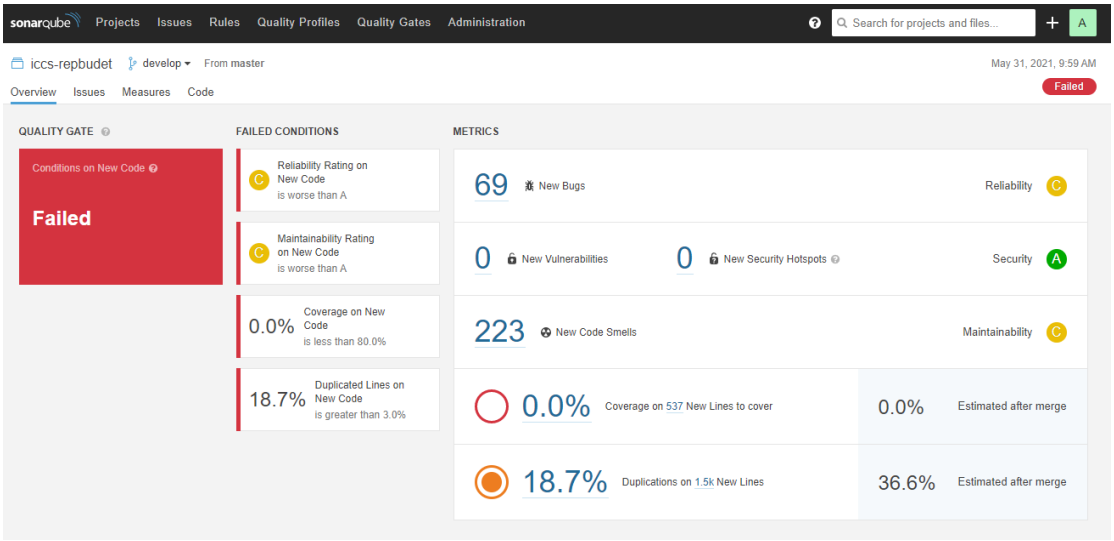
ภาพที่ 4.44 Pipeline สำหรับ Build และ Release โปรเจคไปยังเครื่องใช้งานจริง (Production)

4.5.5 สร้างลิงค์สำหรับตรวจสอบคุณภาพซอร์สโค้ด

ภายหลังจากที่นักพัฒนาระบบ Build โปรเจคไปยังเครื่องทดสอบระบบ (Develop) เพื่อตรวจสอบคุณภาพของซอร์สโค้ด (Source code) และทดสอบการทำงานของโปรแกรม ระบบจะสร้างรายงานผลการตรวจสอบคุณภาพของซอร์สโค้ด (Source code) ไปยังระบบ Sonarqube และได้ทำการแนบลิ้งค์ (Link) ไว้ที่โปรเจค เพื่อสำหรับให้ผู้พัฒนาระบบได้ดูผลการตรวจสอบคุณภาพของซอร์สโค้ดและดำเนินการแก้ไขในครั้งถัดไป



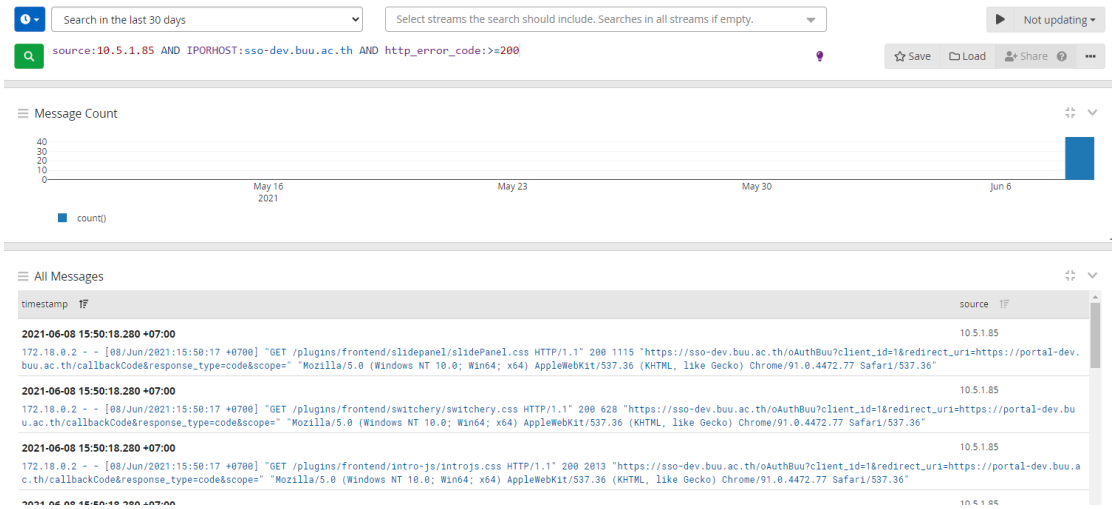
ภาพที่ 4.45 ตัวอย่างลิ้งสำหรับดูผลการตรวจสอบคุณภาพของซอร์สโค้ด



ภาพที่ 4.46 ตัวอย่างผลการตรวจสอบคุณภาพของซอร์สโค้ด

4.5.6 สร้างหน้าจอแสดงข้อมูลข้อผิดพลาดการทำงานของโปรแกรม

เนื่องจากการพัฒนาระบบแบบเก่าไม่มีระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์ส่วนกลาง ทำให้ผู้พัฒนาระบบต้องดำเนินการเข้าไปยังเครื่องพัฒนาระบบโดยตรง ซึ่งในความเป็นจริงนั้นเครื่องสำหรับพัฒนาระบบมีเป็นจำนวนมาก ทำให้ผู้พัฒนาระบบยากต่อการค้นหาข้อมูล ผู้วิจัยได้ทำระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์ส่วนกลาง และทำการสร้างหน้ารายงานผลสำหรับแต่ละโปรเจค เพื่อให้ทางผู้พัฒนาระบบสามารถเข้าตรวจสอบข้อมูลได้ในที่เดียว



ภาพที่ 4.47 ตัวอย่างผลการทำงานของโปรแกรม

บทที่ 5
สรุปการวิจัย

งานวิจัยนี้สามารถนำไปประยุกต์ในองค์กรที่มีการพัฒนาซอฟต์แวร์ ที่ต้องการนำแนวคิด DevOps เข้าไปใช้เพื่อเพิ่มความต่อเนื่องให้แก่การดำเนินงานพัฒนาและส่งมอบซอฟต์แวร์ หากพิจารณาจาก DevOps ซึ่งจะช่วยเพิ่มศักยภาพในการทำงานด้านการส่งมอบซอฟต์แวร์ให้รวดเร็วขึ้นและตรงความต้องการผู้ใช้ได้มากขึ้น อีกทั้งยังช่วยสร้างความต่อเนื่องในกระบวนการพัฒนาซอฟต์แวร์ให้ได้มากขึ้น รวมถึงประสานการทำงานระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการให้ราบรื่นมากยิ่งขึ้น โดยจะสรุปการวิจัยโดยการจัดทำการสนทนากลุ่ม (Focus group) กับฝ่ายพัฒนาระบบได้ดังนี้

ส่วนที่ 1 เปรียบเทียบจำนวนขั้นตอนการพัฒนาระบบ

ผลของการวิจัยสรุปได้ว่าจำนวนขั้นตอนของกระบวนการพัฒนาระบบแบบเดิมกับกระบวนการพัฒนาระบบแบบใหม่นั้นยังคงเท่ากันแต่กิจกรรมต่าง ๆ จะปรับเปลี่ยนใหม่ทั้งหมดเพื่อให้สอดคล้องกับหลักการของ devops ไม่ว่าจะเป็นเครื่องมือที่ใช้สำหรับการวางแผน การพัฒนาระบบ การทดสอบระบบที่เพิ่มความสามารถในการตรวจสอบคุณภาพของซอร์สโค้ด การใช้ด๊อกเกอร์ (Docker) ในการสร้างสภาพแวดล้อมการทำงานของเครื่องพัฒนาระบบและเครื่องสำหรับส่งมอบเหมือนกันเพื่อช่วยให้จัดส่งซอร์สโค้ดได้เร็วขึ้น สร้างมาตรฐานการดำเนินการแอปพลิเคชัน ย้ายซอร์สโค้ดได้อย่างราบรื่น และประหยัดค่าใช้จ่ายโดยการพัฒนาการใช้ทรัพยากรเมื่อใช้ด๊อกเกอร์ทำให้สามารถเรียกใช้ได้ทุกแห่งอย่างเชื่อถือได้ รวมถึงส่งมอบระบบโดยอัตโนมัติและเตรียมการกู้คืนระบบหากเกิดความล้มเหลวในการส่งมอบ และสุดท้ายได้พัฒนาระบบที่ใช้สำหรับติดตามผลเพื่อดูแลและติดตามประสิทธิภาพของซอฟต์แวร์และเซิร์ฟเวอร์ รวมถึงการติดตามปัญหาและการเปลี่ยนแปลงที่เกิดขึ้นด้วยระบบจัดเก็บข้อมูลส่วนกลาง

ส่วนที่ 2 การส่งมอบงานได้อย่างอย่างรวดเร็วและมีประสิทธิภาพ

ผลของการวิจัยสรุปได้ว่าจากขั้นตอนการพัฒนาระบบแบบเดิมและการพัฒนาระบบแบบใหม่นั้นจะเพิ่มเครื่องมือ version control ซึ่งเป็นซอร์ฟแวร์ที่จะคอยบันทึกเวอร์ชั่นการเปลี่ยนแปลงของซอร์สโค้ดหรือเอกสารต่างๆ โดยจะทำการบันทึกไว้ด้วยว่าการเปลี่ยนแปลงแต่ละครั้งนั้นทำเพื่ออะไร และทำโดยใคร อีกทั้งยังสามารถตรวจจับและแจ้งเตือน หากมีนักพัฒนามากกว่าหนึ่งคนแก้ไข

ไฟล์เดียวกันและอัฟโหลดขึ้นต้นฉบับ นอกจากนี้ version control อนุญาตให้สมาชิกเปลี่ยนซอร์สโค้ดกลับไปเป็นเวอร์ชั่นต่างๆ ก่อนหน้าได้ด้วยคำสั่งเดียว (ในกรณีที่เวอร์ชั่นใหม่มีปัญหา) และเพิ่มกระบวนการของ pipeline เพื่อทำขั้นตอนของ ci/cd เพื่อทำให้การพัฒนาซอฟท์แวร์ในแต่ละขั้นตอนมีความเชื่อมต่อกัน เริ่มตั้งแต่มีการเปลี่ยนแปลงในซอร์สโค้ดไปจนถึงการบิ้วด์ (build) และตรวจสอบคุณภาพและส่งซอร์สโค้ดไปยังเครื่องทดสอบและเครื่องส่งมอบโดยอัตโนมัติ รวมถึงการแก้ไขการกำหนดค่าต่างๆ ของซอร์สโค้ด ระหว่างเครื่องทดสอบระบบและเครื่องใช้งานจริงที่เปลี่ยนแปลงไปตามที่ลูกค้ากำหนด

ส่วนที่ 3 ความน่าเชื่อถือของซอฟต์แวร์

ผลของการวิจัยสรุปได้ว่าการพัฒนาระบบแบบเดิมและการพัฒนาระบบแบบใหม่นั้นมีกระบวนการที่ไม่เหมือนกัน เนื่องจากว่าการพัฒนาระบบแบบเดิมนั้นจะมีแค่ในส่วนของการทดสอบการทำงานของซอฟต์แวร์ถูกต้องและเป็นไปตามความต้องการหรือไม่เท่านั้น แต่ในการพัฒนาระบบแบบใหม่นั้นจะเพิ่มกระบวนการในส่วนของการตรวจสอบคุณภาพของซอร์สโค้ด ตรวจหาข้อผิดพลาด (Bugs) ในซอร์สโค้ด ช่วยค้นหาซอร์สโค้ดที่ซับซ้อน ช่องโหว่ทางด้านความปลอดภัยหรือซอร์สโค้ดที่จะเป็นปัญหาในอนาคต (Code Smells) ช่วยค้นหาช่องโหว่ต่างๆ ที่เป็นข้อผิดพลาดหรือปัญหาด้านความปลอดภัย (Vulnerabilities) เพื่อลดช่องโหว่ต่าง ๆ ก่อนส่งมอบให้งานให้กับลูกค้า

ส่วนที่ 4 การลดขั้นตอนการดำเนินงานและจัดการโครงสร้างพื้นฐาน

ผลของการวิจัยสรุปได้ว่าจากขั้นตอนการพัฒนาระบบแบบเดิมและการพัฒนาระบบแบบใหม่นั้นได้ลดจำนวนขั้นตอนการดำเนินงานและจัดการโครงสร้าง โดยลดการใช้คำสั่งต่าง ๆ ด้วยมือในการดำเนินงาน ซึ่งกระบวนการพัฒนาระบบแบบใหม่นั้นทางผู้ดูแลระบบจะกำหนดค่าพื้นฐานต่าง ๆ เท่านั้น โดยกระบวนการที่เหลือจะทำงานเบื้องหลังแบบอัตโนมัติทั้งหมด เพื่อลดข้อผิดพลาดในการทำงานจากคน

บรรณานุกรม

Ben Day. (2019). Azure DevOps Server 2019 Install Guide. Retrieved 12 September 2019, from https://www.benday.com/2019/04/04/azure-devops-server-2019-install-guide/

Pariwat Saknimitwong. (2017). Learn DevOps ตอนที่ 2 : DevOps คืออะไร ?. Retrieved 13 September 2019, from https://medium.com/@pariwat_s/learn-devops-ตอนที่-2-devops-คืออะไร-18ac48d73625

สนิธพรรณ จิตตั้งสมบูรณ์ มฑุปายาส ทองมาก (๒๕๖๑) , ปัจจัยที่ส่งผลต่อความตั้งใจใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

Rachata Tongpagdee. (2017). Docker คืออะไร ใช้งานอย่างไร. Retrieved 15 September 2019, from https://medium.com/@rachatatongpagdee/docker-คืออะไร-ใช้งานอย่างไร-7e77145967b6

Kriangkrai Chaonithi. (2019). DevOps คืออะไร นำมาประโยชน์ได้ยังไง และตัวอย่างการทำ DevOps ที่ Credit OK. Retrieved 15 September 2019, from https://www.spicydog.org/blog/introduction-to-devops-and-the-practical-use-cases-at-credit-ok/

ภาคผนวก

ภาคผนวก ก

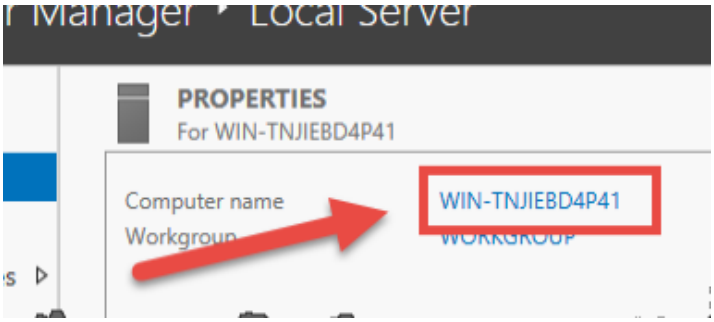
การติดตั้ง Azure devops server 2019

กระบวนการติดตั้งและตั้งค่า Azure devops server 2019 ประกอบไปด้วย

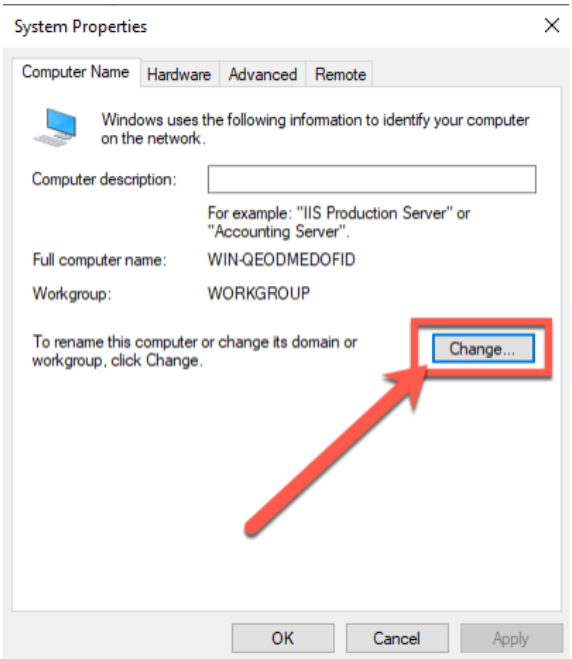
- การ Join Domain
- การติดตั้ง SQL Server 2017
- การติดตั้ง Azure DevOps Server 2019
- การตั้งค่า SMTP Server for Azure DevOps Server 2019

1. การ Join Domain

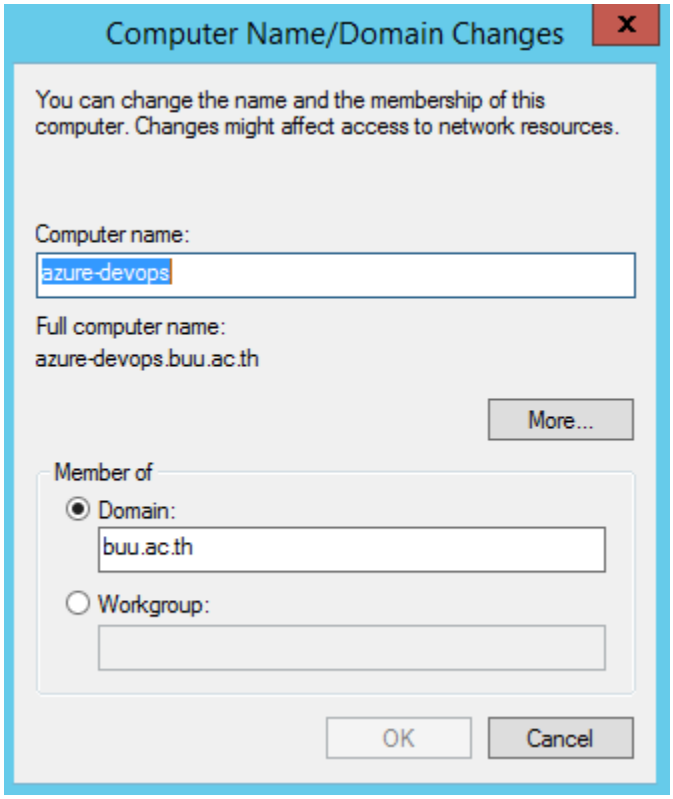
1.1 Join Domain ไปยังเครื่อง Active Directory Domain โดยไปที่ Server manager และคลิกที่ลิงค์ Computer name



1.2 จะปรากฏหน้าต่าง System Properties จากนั้นคลิกที่ปุ่ม Change

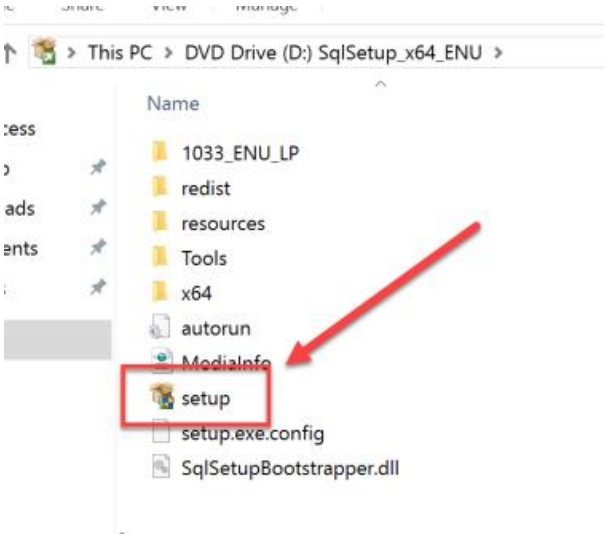


1.3 หลังจากคลิกปุ่ม Change จะปรากฏหน้าต่าง Computer Name/Domain Change ให้ทำการกำหนดชื่อ Computer name และ Domain เป็น buu.ac.th

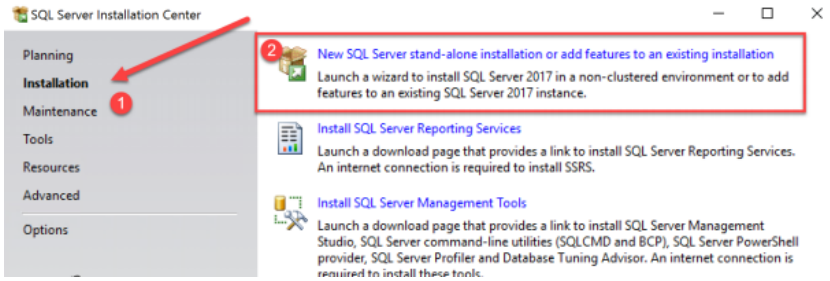


2. การติดตั้ง SQL Server 2017

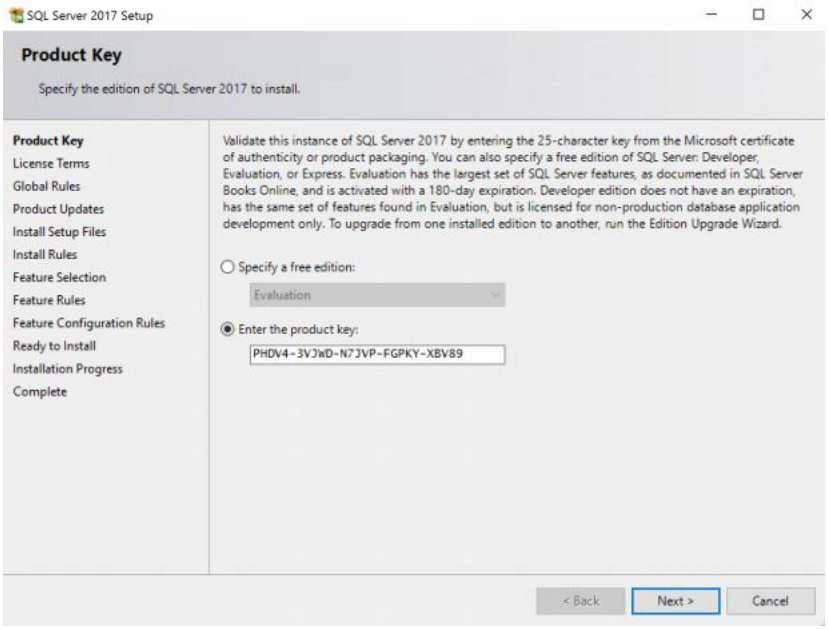
2.1 เปิดที่แผ่น CD ,USB หรือ ไฟล์ ISO แล้วคลิกที่ setup



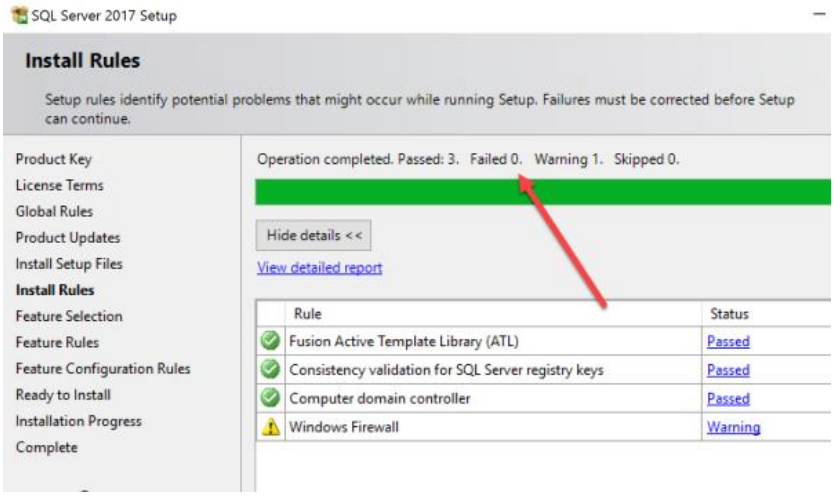
2.2 หลังจากคลิกไฟล์ setup จะปรากฏหน้าต่าง SQL Server Installation Center ให้ทำการคลิกลิงค์ Installation คอลัมด้านซ้ายมือ แล้วเลือกที่เมนู New SQL Server stand-alone installation or add features to an existing installation



2.3 หลังจากนั้นจะปรากฏหน้าต่าง product key โดยที่จะมี product key อยู่แล้ว แล้วกดปุ่ม Next

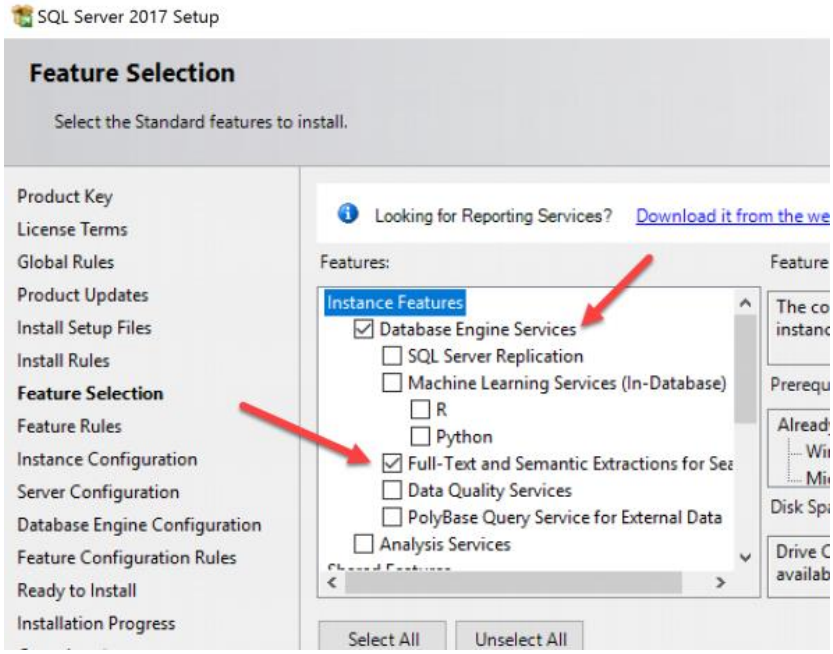


2.4 หลังจากนั้นจะปรากฏหน้าต่าง install rule โปรแกรมจะดำเนินการตรวจสอบความต้องการของระบบ หากผ่านจะขึ้นเครื่องหมายสีเขียว แล้วกดปุ่ม next

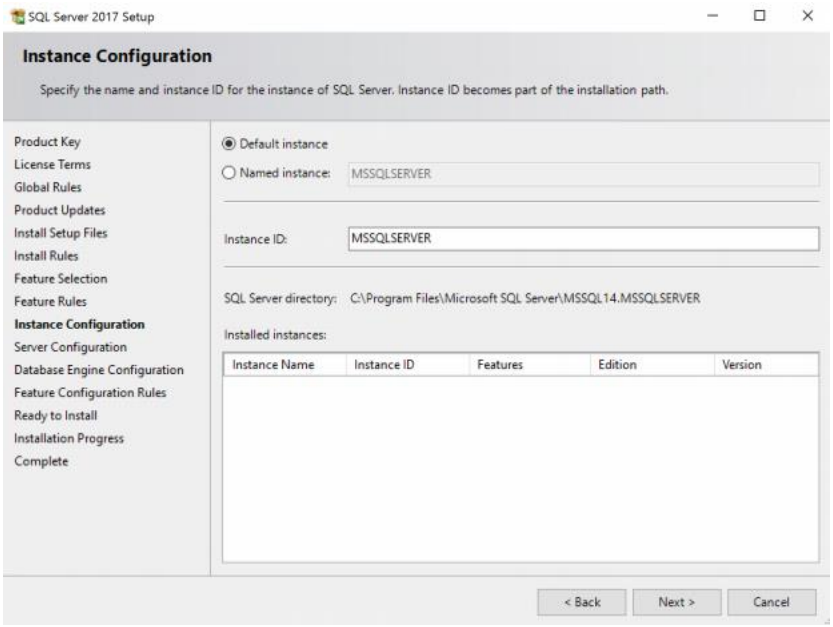


2.5 หลังจากนั้นจะปรากฏหน้าต่าง Feature selection ให้ทำการเลือกดังนี้แล้วกดปุ่ม next

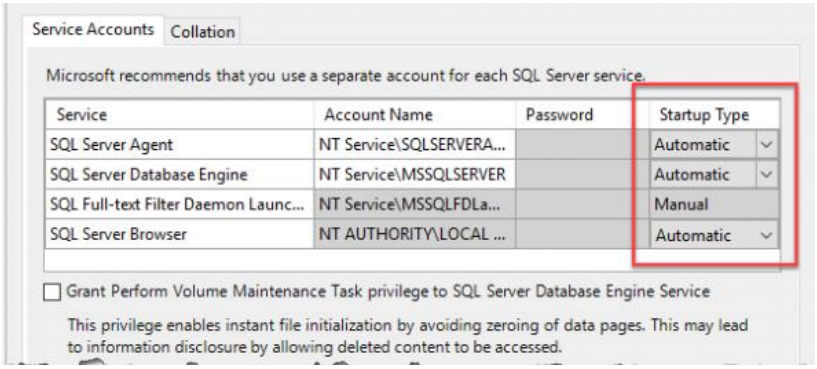
- Database Engine Services
- Full-Text and Semantic Extractions for Search
- Client Tools Connectivity



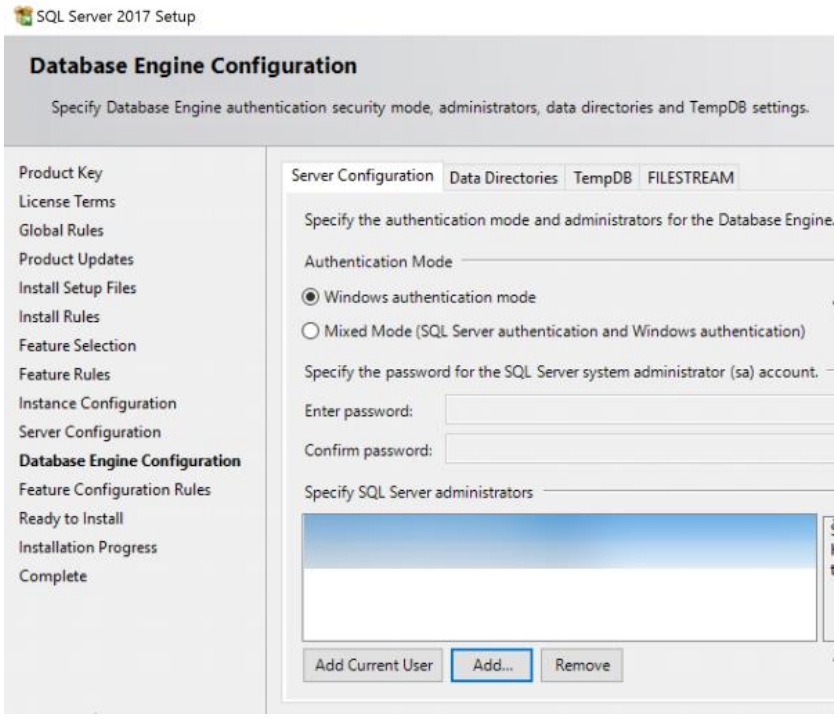
2.6 หลังจากนั้นจะปรากฏหน้าต่าง Instance Configuration โดยกำหนด instance ในตัวอย่างทำการเลือกเป็น Default แล้วกดปุ่ม next



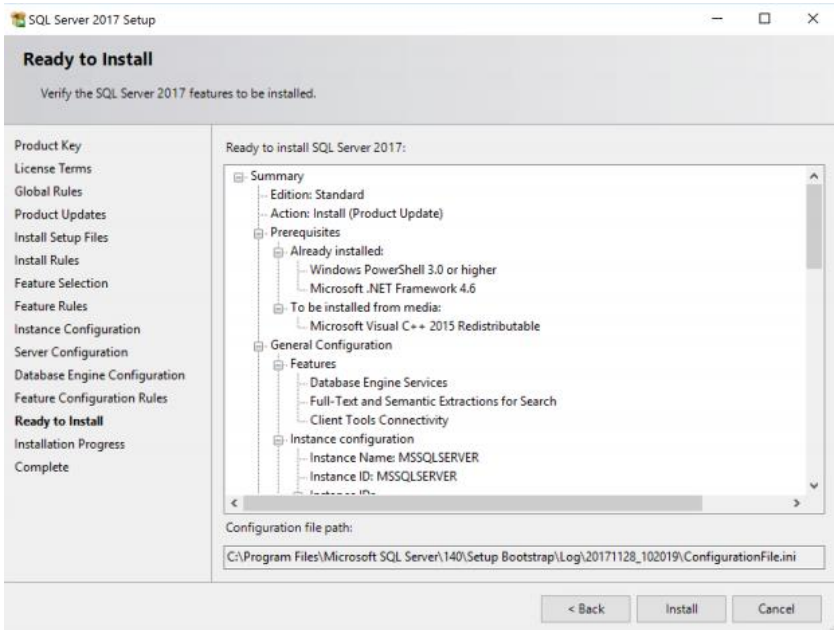
2.7 หลังจากนั้นจะปรากฏหน้าต่าง Server Configuration โดยกำหนดค่า Startup type ดังรูป แล้วกดปุ่ม next



2.8 หลังจากนั้นจะปรากฏหน้าต่าง Database Engine Configuration ให้ทำการกำหนดบัญชีผู้ใช้ที่สามารถเข้าใช้งานแล้วกดปุ่ม next

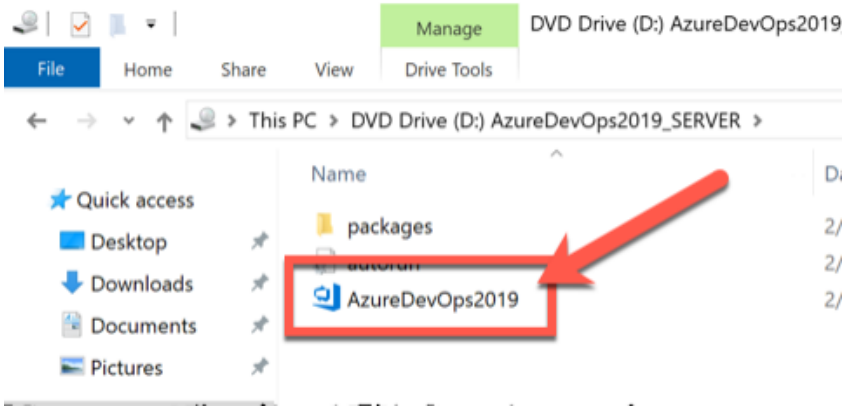


2.9 หลังจากนั้นจะปรากฏหน้าต่าง Ready to Install ให้ทำการกดที่ปุ่ม Install หากขึ้น complete ให้กดปุ่ม close เพื่อเสร็จสิ้นการติดตั้ง

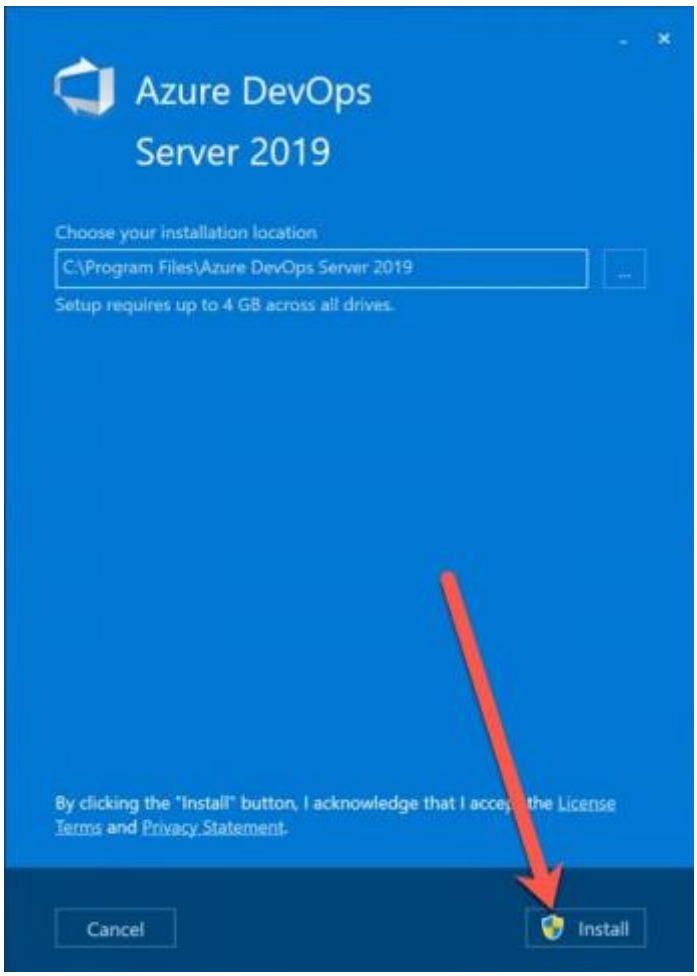


3. การติดตั้ง Azure DevOps Server 2019

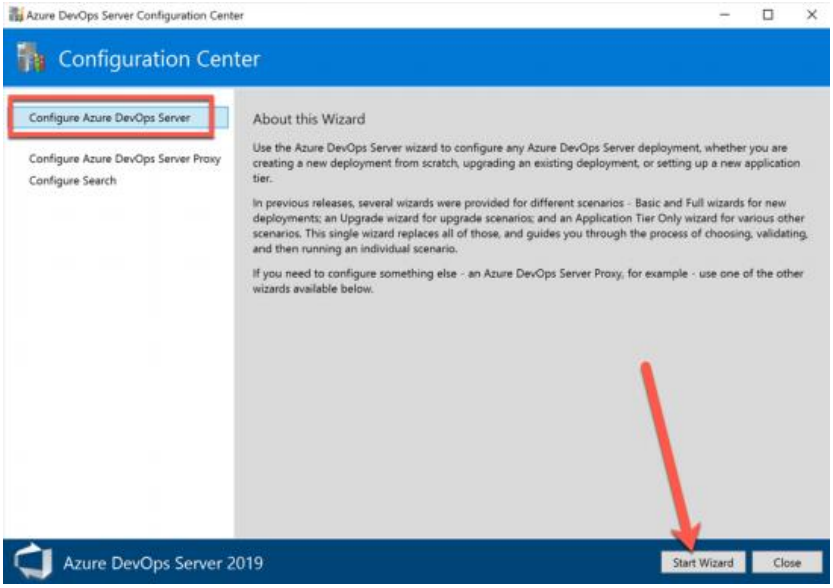
3.1 เปิดที่แผ่น CD ,USB หรือ ไฟล์ ISO แล้วคลิกที่ AzureDevOps2019



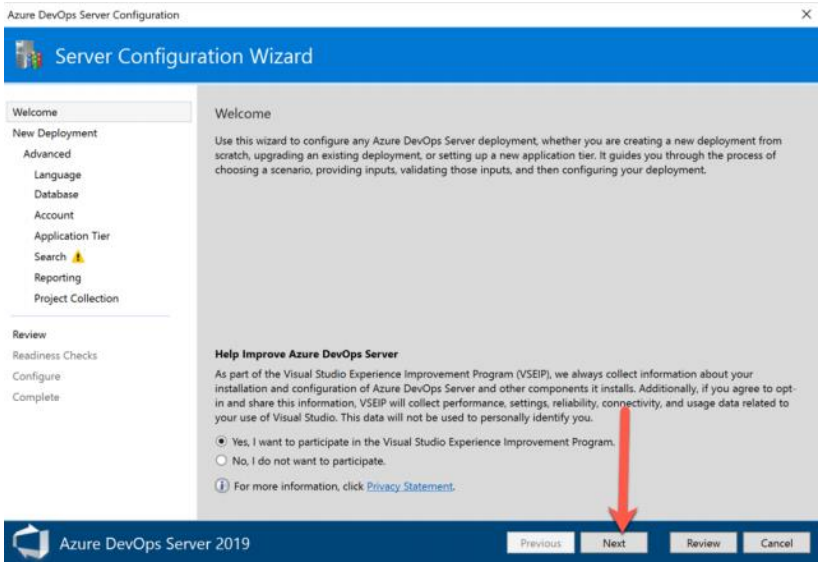
3.2 หลังจากนั้นจะปรากฏหน้าต่าง Azure DevOps Server Setup ให้ทำการกดปุ่ม Install



3.3 หลังจากติดตั้งเสร็จจะปรากฏหน้าต่าง Azure DevOps Server Configuration Center ให้ทำการกดปุ่ม Start Wizard

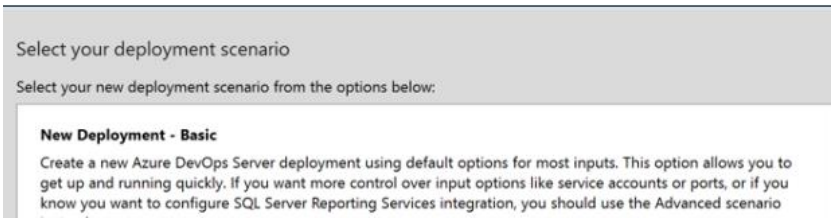


3.4 หลังจากนั้นจะปรากฏหน้าต่าง Welcome ให้ทำการเลือก Yes, I want to participate in the Visual Studio Experience Improvement Program แล้วกดปุ่ม next

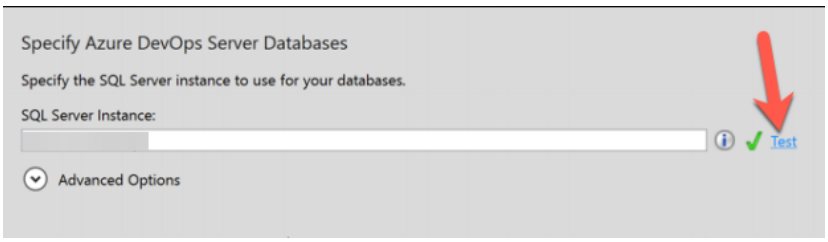


3.5 หลังจากนั้นจะปรากฏหน้าต่าง Deployment Type ให้ทำการเลือกที่ This is a new Azure DevOps Server deployment แล้วกด next

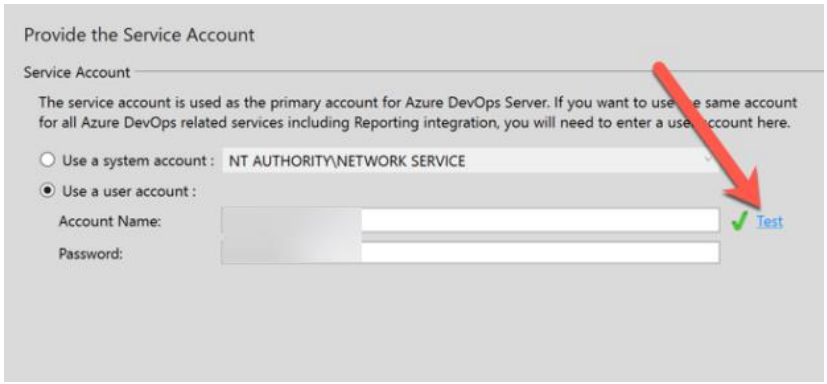
3.6 หลังจากนั้นจะปรากฏหน้าต่าง Select your deployment scenario ให้ทำการเลือก New Deployment – Basic แล้วกดปุ่ม next



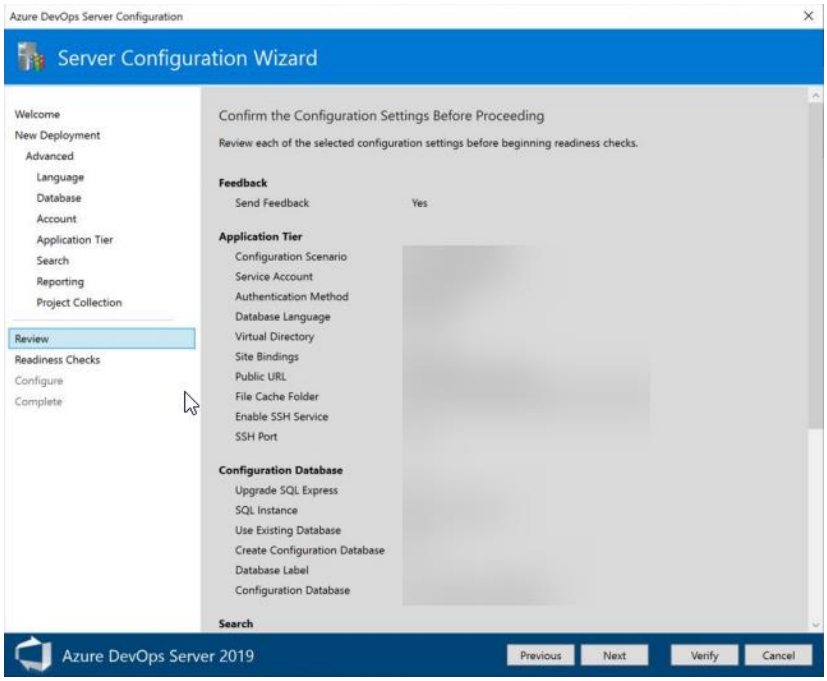
3.7 หลังจากนั้นจะปรากฏหน้าต่าง Specify Azure DevOps Server Databases โดยกำหนด IP หรือชื่อของเครื่องของ SQL Server แล้วกดปุ่ม next



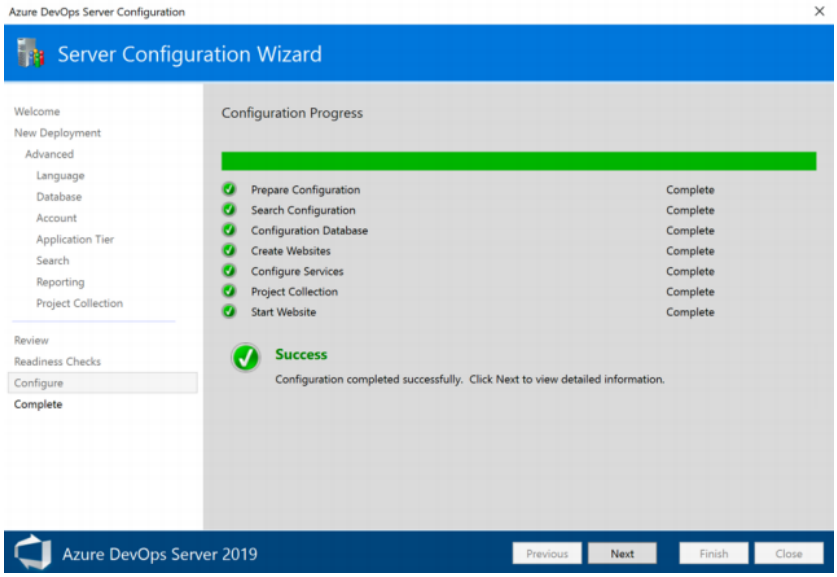
3.8 หลังจากนั้นจะปรากฏหน้าต่าง Provide Service Account โดยทำการระบุบัญชีผู้ใช้สำหรับ Azure devops service แล้วกดปุ่ม Test หากดำเนินการสำเร็จจะขึ้นเครื่องหมายถูก และกดปุ่ม next



3.9 หลังจากนั้นจะปรากฏหน้าต่าง Confirm the Configuration Settings Before Proceeding ซึ่งจะแสดงรายละเอียดการตั้งค่า หากไม่มีการแก้ไขให้กดปุ่ม next



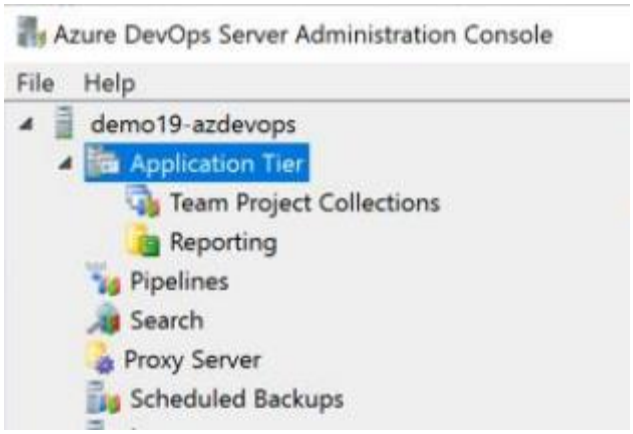
3.10 หลังจากนั้นระบบจะทำการติดตั้งโปรแกรม หากดำเนินการติดตั้งสำเร็จจะขึ้นเครื่องหมายถูกทุกขั้นตอนแล้วกดปุ่ม next แล้วก็ close ตามลำดับ



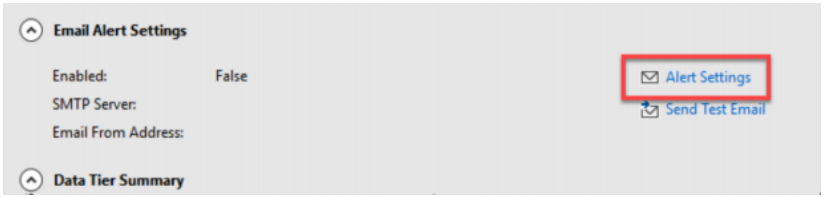
4. การตั้งค่า SMTP Server for Azure DevOps Server 2019

4.1 เข้าสู่ระบบของเครื่อง azure devops server

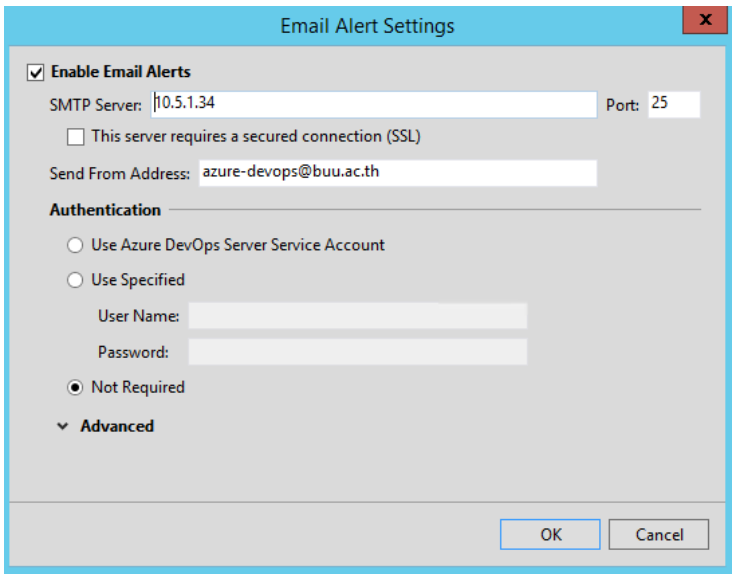
4.2 ไปที่เมนู start แล้วค้นหาคำว่า Azure DevOps Server Administrator Console



4.3 หน้าต่างด้านขวามือ เลื่อนหาในส่วนของ Email Alert Settings แล้วทำการคลิกที่ Alert Settings



4.4 หลังจากนั้นจะปรากฏหน้าต่าง Email Alert Settings แล้วทำการคลิกที่ช่อง Enable Email Alerts แล้วกำหนด SMTP Server ที่ใช้สำหรับส่งอีเมล แล้วกดปุ่ม OK



ภาคผนวก ข
การติดตั้งด๊อกเกอร์

ความต้องการของระบบ

- Ubuntu Focal 20.04 (LTS)
- Ubuntu Bionic 18.04 (LTS)
- Ubuntu Xenial 16.04 (LTS)

ขั้นตอนการติดตั้งด๊อกเกอร์

1. อัพเดท apt แพ็คเกจและติดตั้งแพ็คเกจที่อนุญาตใช้ repository ผ่าน HTTPS

   sudo apt-get update

   sudo apt-get install \

   apt-transport-https \

   ca-certificates \

   curl \

   gnupg-agent \

   software-properties-common

2. เพิ่ม Docker’s official GPG key

   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

–

3. ใช้คำสั่งข้างล่างเพื่อติดตั้ง repository ของด๊อกเกอร์

   sudo add-apt-repository \

   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

   $(lsb_release -cs) \

   stable"

4. อัพเดท apt แพ็คเกจและดำเนินการติดตั้งด๊อกเกอร์เวอร์ชั่นล่าสุด

   sudo apt-get update

   sudo apt-get install docker-ce docker-ce-cli containerd.io

ภาคผนวก ค
การติดตั้งเว็บเซิร์ฟเวอร์

กระบวนการติดตั้งและตั้งค่าเว็บเซิร์ฟเวอร์ประกอบไปด้วย

- Apache2
- PHP
- Mariadb

1. การติดตั้ง Apache2 และเพิ่มกฎของ firewall ด้วยคำสั่งดังนี้

   sudo apt update

   sudo apt install apache2

   sudo ufw allow in "Apache"

2. การติดตั้ง PHP ด้วยคำสั่งดังนี้

   sudo apt install php libapache2-mod-php php-mysql

   sudo systemctl restart apache2

3. การติดตั้ง MARIADB ด้วยคำสั่งดังนี้

   sudo apt update

   sudo apt install mariadb-server