การสร้างไปป์ไลน์เทมเพลต สำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI

Creating a template pipeline for CI/CD software development with Gitlab CI

นายมนัสวี นรบดีศุภกร

สหกิจนี้เป็นส่วนหนึ่งของการศึกษาตาม

หลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

Creating a template pipeline for CI/CD software development with Gitlab CI

MANASAVEE NARABADEESUPPHAKORN

A COOPERATIVE EDUCATION SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)

DEPARTMENT OF COMPUTER SCIENCE, SCHOOL OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

หัวข้อสหกิจศึกษา	การสร้างไปป์ไลน์เทมเพลต สำหรับการพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI Creating a template pipeline for CI/CD software development with Gitlab Ci
ชื่อนักศึกษา	นาย มนัสวี นรบดีศุภกร รหัสนักศึกษา 62050212
ปริญญา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2565
อาจารย์ที่ปรึกษา	อ.สันธนะ อู่อุดมยิ่ง

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.) อนุมัติให้สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2565

คณะกรรมการสอบ	ลายมือชื่อ
ดร.วิชญะ ต่อวงศ์ไพชยนต์ กรรมการ	
อ.สันธนะ อู่อุดมยิ่ง กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อสหกิจศึกษา การสร้างไปป์ไลน์เทมเพลตสำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI
ชื่อนักศึกษา นาย มนัสวี นรบดีศุภกร รหัสนักศึกษา 62050212
ปริญญา วิทยาศาสตร์บัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา วิทยาการคอมพิวเตอร์
คณะ วิทยาศาสตร์
มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)
ปีการศึกษา 2565
อาจารย์ที่ปรึกษา อ.สันธนะ อู่อุดมยิ่ง

บทคัดย่อ

สหกิจนี้มีไว้เพื่อให้ทำการศึกษาการสร้างไปป์ไลน์เทมเพลตสำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI ในการแก้ไขปัญหาความซ้ำซ้อนของขั้นตอนในการพัฒนาซอฟต์แวร์แบบเก่า โดยใช้ซอฟต์แวร์เดิมมาทำการพัฒนาแบบ CI/CD จำนวน 5 ซอฟต์แวร์ที่เขียนแบบ Monorepo ซึ่งแต่ละซอฟต์แวร์จะติดตั้งใน 4 Environment เครื่องมือที่ใช้ในการสหกิจได้แก่ DevSecOps Tools, Container technology, Gitlab CI, Opstella platform ผลสรุปว่าไปป์ไลน์เทมเพลตที่ได้นำไปใช้แทนที่ไปป์ไลน์ดั้งเดิมสามารถทำงานแทนที่ไปป์ไลน์ดั้งเดิมได้

คำสำคัญ : ไปป์ไลน์เทมเพลต, CI/CD, Gitlab CI, Monorepo, DevSecOps Tools, Container technology, Gitlab CI, Opstella platform

Title	Creating a template pipeline for CI/CD software development with Gitlab CI
Students	Mr. Manasavee Narabadeesuphakorn Student ID 62050212
Degree	Bachelor of Science (Computer Science)
Department	Computer Science
School	Science
University	King Mongkut's Institute of Technology Ladkrabang (KMITL)
Year	2022
Advisor	Aj.Suntana Oudomying

Abstract

This Cooperative educations aims to study the creation of a CI/CD-based software development template pipeline with Gitlab CI to resolve the redundancy of traditional software development workflows. Using the original software to develop CI/CD designs, 5 software is written in Monorepo, each of which will be installed in 4 environments. The tools used for co-op are DevSecOps Tools, Container technology, Gitlab CI, Opstella platform. Pipeline templates that have been applied in place of the original pipeline can work in place of the original pipeline.

Keywords : Pipeline Template, CI/CD, Gitlab CI, Monorepo, DevSecOps Tools, Container technology, Gitlab CI, Opstella platform

กิตติกรรมประกาศ

การสหกิจเรื่อง การศึกษาการสร้างไปป์ไลน์เทมเพลตสำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI สามารถดำเนินการจนประสบความสำเร็จลุล่วงไปด้วยดี เนื่องจากความกรุณาและความร่วมมือของทุก ๆ ท่านที่มีส่วนร่วมในการให้คำแนะนำ ให้ความช่วยเหลือ และการสนับสนุนตลอดมา ขอขอบพระคุณ อ.สันธนะ อู่อุดมยิ่ง ที่ให้ความอนุเคราะห์ในการดูแลเป็นอาจารย์ที่ปรึกษาในการสหกิจศึกษารวมถึงให้คำปรึกษาและแนะนำเพื่อปรับปรุงในส่วนที่บกพร่องเพื่อให้การสหกิจศึกษาครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณ คุณ วัชรินทร์ ยางงาม และ คุณ นันทวัฒน์ ไชยรัตน์ ที่คอยให้คำปรึกษาตลอดทั้งการช่วยเหลือและชี้แนะแนวทางในการทำสหกิจศึกษาครั้งนี้จนสำเร็จลุล่วง

ขอขอบคุณ ดร.วิชญะ ต่อวงศ์ หัวหน้าภาควิชาวิทยาการคอมพิวเตอร์ ที่ให้ความอนุเคราะห์ในการดูแลเป็นกรรมการในการสหกิจ

ขอขอบพระคุณทางบริษัท บริษัท ออพซ์ตา (ประเทศไทย) จำกัด ที่สนับสนุนโครงงานสหกิจศึกษาในครั้งนี้เพื่อให้นักศึกษาเรียนรู้กระบวนการในการทำงานจริงก่อนที่จะก้าวเข้าสู่โลกของการทำงานและผลักดันให้สหกิจศึกษานี้สำเร็จไปได้ด้วยดี ผู้จัดทำขอขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

มนัสวี นรบดีศุภกร

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

บริการที่ออพซ์ตาให้กับลูกค้าคือการจัดเตรียมโครงสร้างของ DevSecOps ไว้ด้วยเครื่องมือต่าง ๆ ให้เป็นแพลทฟอร์มให้ลูกค้านำเซอร์วิสของตนขึ้นมาพัฒนา/ทำงาน เป็นการลดภาระของลูกค้าในการติดตั้ง ดูแล และจัดการทรัพยาการเหล่านี้ กระบวนการทั้งหมดในการพัฒนาของ DevSecOps มีดังนี้ การวางแผน การพัฒนาซอร์สโค้ด การบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ด การทดสอบซอฟต์แวร์ การส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ การกำหนดสภาพแวดล้อมในการส่งออกซอฟต์แวร์ และการติดตามการทำงานของซอฟต์แวร์ ซึ่งในสหกิจนี้จะมีขอบเขตอยู่ที่กระบวนการ CI/CD ซึ่งประกอบไปด้วยการบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ด การทดสอบซอฟต์แวร์ การส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ การกำหนดสภาพแวดล้อมในการส่งออกซอฟต์แวร์

การพัฒนาซอฟต์แวร์ที่มีโดยใช้การพัฒนาแบบ CI/CD โดยใช้ไปป์ไลน์ใน Gitlab CI ในปัจจุบันเกิดการความยุ่งยากและความซ้ำซ้อน เนื่องจากการตั้งค่าต่าง ๆ ที่ไม่เกี่ยวข้องกับการส่งออกซอฟต์แวร์มีความซ้ำซ้อนเหมือนกันในทุก ๆ พื้นที่ ถ้าหากต้องการใช้งานไปป์ไลน์ในพื้นที่ใหม่ก็จำเป็นต้องกำหนดการตั้งค่าเองใหม่ทุกครั้ง

ผู้จัดทำและบริษัทออพซ์ตา ประเทศไทย จำกัด จึงได้ร่วมกันคิดวิธีแก้ไขปัญหาโดยสร้างเทมเพลตสำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI เพื่อรวมศูนย์การตั้งค่าที่เหมือนกันไว้ในที่เดียว โดยทำให้เป็นเทมเพลตสำหรับเรียกใช้ได้ในทุก ๆ พื้นที่ ถ้าหากต้องการเปลี่ยนแปลงแก้ไขสิ่งใด เพียงแก้ไขเทมเพลต การตั้งค่าต่าง ๆ ก็จะถูกแก้ไขในทุก ๆ พื้นที่ด้วย

1.2 วัตถุประสงค์ของโครงงานสหกิจศึกษา

เพื่อออกแบบเทมเพลตสำหรับไปป์ไลน์ด้วย Gitlab CI แทนการทำงานด้วยไปป์ไลน์เดิมที่ต้องกำหนดการทำงานเองทุกโปรเจคที่ใช้งานบนแพลทฟอร์มของออพซ์ตา

1.3 ขอบเขตของโครงงานสหกิจศึกษา

1) เป็นไปป์ไลน์เทมเพลตที่ใช้กับ Gitlab CI เท่านั้น
2) ทำเฉพาะขั้นตอนการกำหนดสภาพแวดล้อม, การตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ด, การทดสอบความครอบคลุมของซอร์สโค้ด, การบีบอัดซอร์สโค้ดและการสร้าง

อิมเมจของซอฟต์แวร์จากซอร์สโค้ด, การส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ เท่านั้น

3) มีการนำไปใช้งานจริงกับซอฟต์แวร์ของลูกค้าในภาษา .NET เท่านั้น

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ประโยชน์ต่อองค์กร

1) ได้รับไปป์เทมเพลตที่ช่วยลดเวลาในการดูแลลำดับการทำงานของไปป์ไลน์โดยกำหนดไว้ในเทมเพลต เมื่อพื้นที่ใดเรียกใช้เทมเพลตจะได้รับลำดับการทำงานเดียวกันด้วย
2) ได้รับไปป์เทมเพลตที่ช่วยลดเวลาในการจัดการการตั้งค่าโดยไม่จำเป็นต้องกำหนดการตั้งค่าใหม่ในทุก ๆ พื้นที่

1.4.2 ประโยชน์ต่อผู้จัดทำ

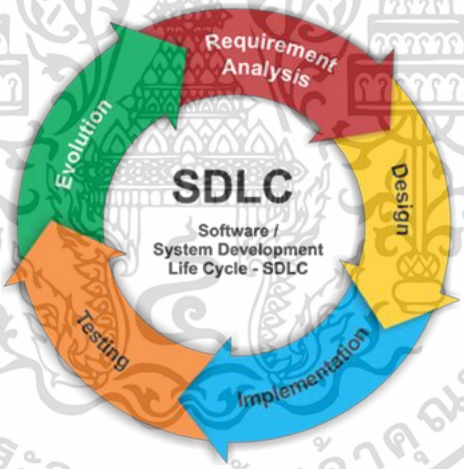
1) ได้เรียนรู้การทำไปป์ไลน์ CI/CD ด้วย Gitlab Ci
2) ได้พัฒนาทักษะในการทำงานร่วมกับผู้อื่น ทั้งทางด้านการสื่อสารและการแสดงออก

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในการพัฒนาไปป์ไลน์เทมเพลตสำหรับการพัฒนาแบบ CI/CD ด้วย Gitlab CI ผู้จัดทำได้ค้นคว้าเอกสารที่เกี่ยวข้องสำหรับการพัฒนา โดยแบ่งเป็น 2 หัวข้อดังนี้

2.1 กระบวนการพัฒนาแบบ DevSecOps

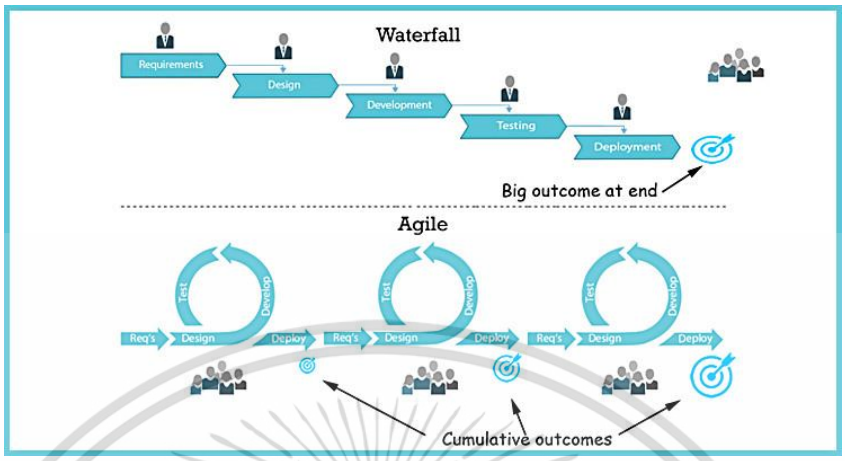
กระบวนการพัฒนาซอฟต์แวร์ในปัจจุบัน ยังคงสามารถกล่าวได้ว่าเป็นไปตามวงจรการพัฒนาซอฟต์แวร์ตามโมเดล SDLC แสดงในรูปที่ 1 โดยที่รูปแบบการพัฒนาซอฟต์แวร์ที่เป็นที่นิยมในปัจจุบันที่ให้คุณภาพของซอฟต์แวร์ที่ดีคือ กระบวนการพัฒนาซอฟต์แวร์แบบ DevSecOps ที่ทำให้การพัฒนาซอฟต์แวร์เป็นไปอย่างสะดวก รวดเร็ว มีมาตรฐาน กระบวนการพัฒนาแบบ DevSecOps ต่างจาก DevOps ทั่วไปคือจะมีการ security เพิ่มมา เช่น การตรวจสอบช่องโหว่ของซอร์สโค้ด การทดสอบความครอบคลุมของซอร์สโค้ด การทดสอบช่องโหว่ของอิมเมจ



รูป 1 วงจรการพัฒนาแบบ SDLC

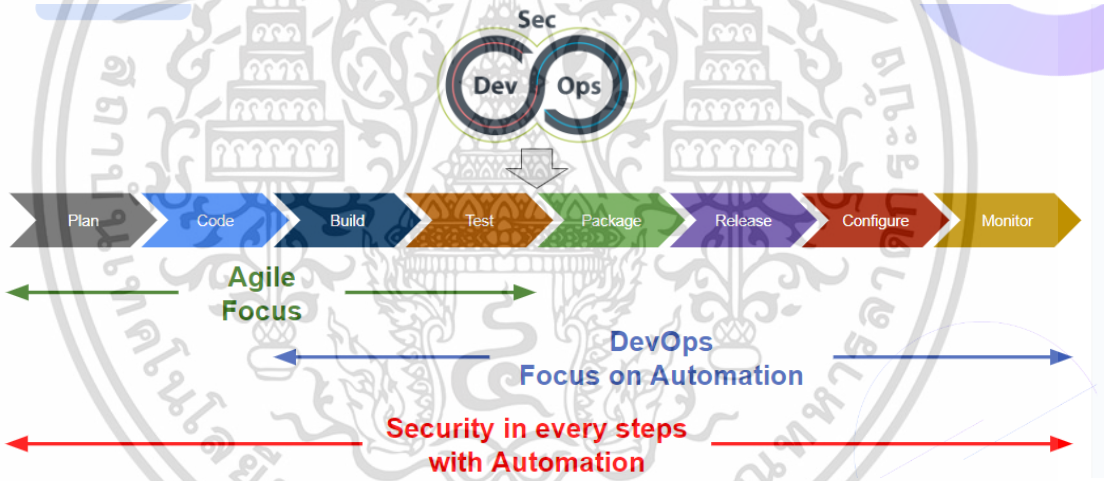
โดยที่วงจรการพัฒนาซอฟต์แวร์ตามโมเดล SDLC มีรูปแบบในการพัฒนาที่เป็นที่นิยมอยู่สองรูปแบบ แสดงในรูปที่ 8 นั่นคือรูปแบบ waterfall เป็นการพัฒนาซอฟต์แวร์ที่เปรียบได้กับน้ำตกที่ไหลลงไปข้างล่างเรื่อย ๆ ตามลำดับไม่ไหลย้อนกลับ จะทำการพัฒนาในแต่ละขั้นตอนให้แล้วเสร็จจึงจะไปทำยังขั้นถัดไป มักใช้เวลาที่ยาวนานในการพัฒนาซอฟต์แวร์ในรูปแบบนี้ ส่วนอีกรูปแบบหนึ่งคือ Agile เป็นการนำวงจรการพัฒนาซอฟต์แวร์ตามโมเดล SDLC มาทำการวนเวียนเป็นรอบ ๆ โดยในแต่ละรอบจะทำทุกขั้นตอนในวงจรการพัฒนาตามโมเดล SDLC และใช้เวลาประมาณ 2 – 4 ต่อรอบหนึ่ง ๆ โดยแบ่งส่วนของซอฟต์แวร์ออกเป็นส่วนย่อย ๆ เพื่อให้สามารถทำการพัฒนาสำเร็จได้ในเวลาแต่ละ

รอบของการทำงาน การพัฒนาในรูปแบบนี้ปัจจุบันเป็นที่นิยมมากกว่าเนื่องจากสามารถมองเห็นผลลัพธ์ของซอฟต์แวร์ได้ไวขึ้น ซึ่งวงจรการพัฒนารูปแบบ Agile มีการนำมาปรับใช้กับ DevSecOps



รูป 2 Waterfall และ Agile

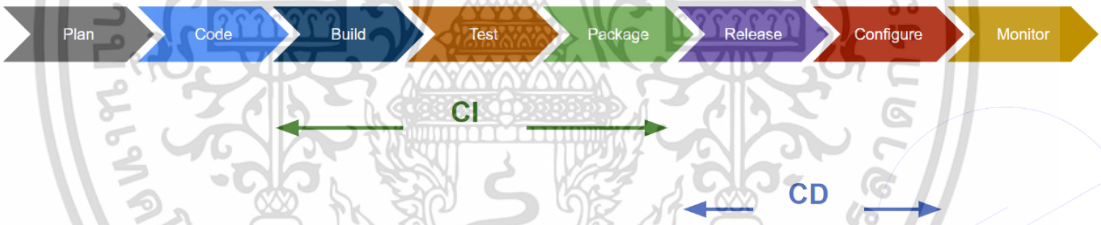
โดยมีความสัมพันธ์กันดังที่แสดงในรูปที่ 2



รูป 3 DevSecOps Scope

Agile และ DevSecOps ใช้ทำงานร่วมกันโดยมีจุดโฟกัสคนละส่วน ในส่วนของ Agile จะโฟกัสตั้งแต่ขั้นตอน Plan หรือการวางแผน ซึ่งในที่นี้รวมถึงขั้นตอนการ เก็บ requirement การ design software การวางแผนจัดการการทำงานต่าง ๆ ด้วย ต่อมาคือขั้นตอน Code หรือขั้นตอนที่ใช้ภาษาคอมพิวเตอร์ต่าง ๆ ในการพัฒนาขึ้นมาเป็นซอฟต์แวร์ให้สำเร็จตามแผนที่ได้วางไว้ในขั้นตอน Plan ต่อมาขั้นตอนการ Build เป็นขั้นตอนที่ทับซ้อนทั้งส่วนของ Agile และ DevOps โดยขั้นตอนนี้จะเป็นการนำซอร์สโค้ดที่ได้ทำการพัฒนาในขั้นตอน Code มาทำการบีบอัด และสร้างเป็น image โดยใช้ Dockerfile ในการกำหนดจำลองขั้นตอนการทำงานของซอฟต์แวร์ เพื่อเตรียมความพร้อมในการนำ image ไปสร้างเป็น Containerต่อไป ถัดมาเป็นขั้นตอนสุดท้ายของ Agile และขั้นตอนที่สองของ DevOps นั่นคือขั้นตอนในการ Test หรือการทดสอบ ซึ่งในที่นี้รวมถึงทั้งการทดสอบซอฟต์แวร์

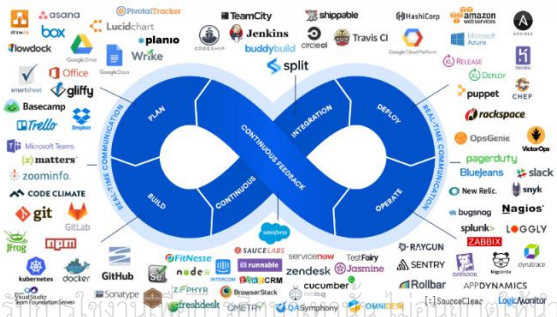
โดยทดสอบความครอบคลุมของการทำงาน การทดสอบช่องโหว่ของซอฟต์แวร์ การทดสอบช่องโหว่ของ image ที่สร้างขึ้นจากการบีบอัดซอฟต์แวร์ด้วย ขั้นตอนต่อมาคือ Package คือการแพคซอฟต์แวร์และติดป้ายระบุเวอร์ชั่น ซึ่งขั้นตอนนี้ในปัจจุบันหากใช้ Docker image ในขั้นตอนของการ Build จะทำขั้นตอนนี้ไปแล้วด้วยเทียบได้กับการติด tag ของ image ถัดมาในขั้นตอนของการ Release หรือการปล่อยซอฟต์แวร์ หรือการส่งออกซอฟต์แวร์จะเป็นขั้นตอนที่นำ images จากการ build มาสร้างเป็น Container โดยสามารถนำไปทำงานได้ในหลายสภาพแวดล้อมขึ้นอยู่กับความต้องการของผู้พัฒนา โดยสามารถกำหนดความต้องการในการส่งออกได้ที่ขั้นตอน Configure เพื่อให้การส่งออกเป็นไปตามสภาพแวดล้อมที่ผู้พัฒนากำหนด สุดท้ายคือขั้นตอนหลังจากที่ส่งออกซอฟต์แวร์ไปเรียบร้อยแล้วจะเป็นขั้นตอนการติดตามผลการทำงานของซอฟต์แวร์ หรือขั้นตอน Monitor ซึ่งจะเป็นการสอดส่องดูผลลัพธ์ของซอฟต์แวร์หลังจากที่ส่งออกไปว่ามีการทำงานปกติหรือไม่ ในปัจจุบันใช้ CI/CD เพื่อทำให้การดำเนินการในขั้นตอนต่างๆเป็นโดยอัตโนมัติ กล่าวคือ DevSecOps มี CI/CD เป็นขั้นตอนย่อย โดยที่กระบวนการตั้งแต่ Build ไปจนถึง Configure เป็นกระบวนการที่อยู่ในขั้นตอนของ CI/CD ในการนำเอาเครื่องมืออัตโนมัติมาช่วยในการทำงาน เช่น การนำเอา kaniko มาช่วยในขั้นตอนของการ Build และ Package การนำเอา SonarQube มาช่วยในเรื่องของการตรวจสอบช่องโหว่ของซอร์สโค้ดดังที่แสดงในรูปที่ 4



รูป 4 CI/CD Scope

2.2 เครื่องมือที่ใช้ในการพัฒนาไปป์ไลน์เทมเพลต

มีเครื่องมือต่าง ๆ มากมายที่ใช้ในการพัฒนา DevSecOps ดังที่แสดงในรูปที่ 5 แต่เครื่องมือที่เลือกใช้ในการพัฒนาไปป์ไลน์เทมเพลตมีดังนี้



รูป 5 DevSecOpsและเครื่องมือ

2.2.1 SonarQube

SonarQube เป็นเครื่องมือประกันคุณภาพโค้ดที่รวบรวมและวิเคราะห์ซอร์สโค้ดและจัดทำรายงานสำหรับคุณภาพโค้ดของโครงการ โดยจะรวมเครื่องมือการวิเคราะห์แบบคงที่และแบบไดนามิกเข้าด้วยกัน และช่วยให้วัดคุณภาพได้อย่างต่อเนื่องตลอดเวลา

2.2.2 Kaniko

Kaniko เป็นเครื่องมือในการสร้างอิมเมจคอนเทนเนอร์จาก Dockerfile ภายในคอนเทนเนอร์หรือคลัสเตอร์ Kubernetes

2.2.3 Harbor

เป็นรีจีสทรีแบบโอเพ่นซอร์สที่รักษาความปลอดภัยของอาร์ติแฟกต์ด้วยนโยบายและการควบคุมการเข้าถึงตามบทบาท ทำให้มั่นใจว่ารูปภาพจะถูกสแกนและปราศจากช่องโหว่และเซ็นอิมเมจว่าเชื่อถือได้

2.2.4 Helm

Helm เป็นเครื่องมือที่ทำให้การสร้าง การบรรจุ การกำหนดค่า และการปรับใช้แอปพลิเคชัน Kubernetes เป็นไปโดยอัตโนมัติโดยการรวมไฟล์การกำหนดค่าของคุณไว้ในแพ็คเกจเดียวที่ใช้ซ้ำได้

2.2.5 Gitlab CI

เป็นเครื่องมือที่ใช้ในการพัฒนาซอฟแวร์แบบขั้นตอน CI/CD โดยทำให้เป็นไปป์ไลน์ ซึ่งมีส่วนประกอบหลักๆดังนี้

2.2.5.1 Jobs

เป็นหน่วยการทำงานย่อยที่สุดสำหรับ Pipeline ใช้กำหนดค่าการทำงานต่างๆ โดยปกติหนึ่ง Jobs จะทำเพียง 1 งาน เช่น Jobs สำหรับการ build image จะทำงานโดยการสร้างอิมเมจของซอฟต์แวร์ขึ้นมาจากการบีบอัดซอร์สโค้ดของซอฟต์แวร์โดยในแต่ละ Jobs เราสามารถกำหนดเงื่อนไขการทำงาน ได้มากมาย ดังที่แสดงไว้ในตาราง 1

สิ่งที่ต้องระบุทุก Jobs

ชื่อของเงื่อนไข	หน้าที่
name	ใช้กำหนดชื่อของ Jobs
script	คำสั่งในการทำงานของ Jobs

ตาราง 1 ส่วนประกอบของที่ต้องระบุทุก Jobs

ตัวอย่างการกำหนดค่าเพิ่มเติมที่สามารถทำได้สามารถดูได้จากตาราง 2

ชื่อของเงื่อนไข	หน้าที่
stage	ใช้ระบุ stage ของ Jobs
variables	ค่าตัวแปรที่กำหนดขึ้นมาเพื่อใช้งานใน Jobs
rules	ใช้กำหนดเงื่อนไขในการทำงานของ Jobs
dependencies	ใช้กำหนดความสัมพันธ์ของ jobs กับ Jobsอื่น ๆ
image	ชื่อของ image ที่ต้องการใช้แทนที่ Default image

ตาราง 2 ส่วนประกอบที่สามารถใส่เพิ่มเติมได้ของ Jobs

2.2.5.2 Stages

เป็นหน่วยที่ใช้ในการกำหนดลำดับการทำงานของ Jobs ใน Pipeline โดยจะเรียงการทำงานเป็นลำดับขั้น Sequence เช่นหากกำหนดค่า stages ดังนี้

Stage1

Jobs1

Stage2

Jobs1.1

Jobs1 ที่อยู่ใน Stage1 ที่มาก่อน จะทำงานก่อน Jobs1.1 ที่อยู่ใน Stage2 ที่มาภายหลัง ในหนึ่ง Stagesสามารถมีได้หลาย Jobs

2.2.5.3 Pipeline

เป็นสิ่งที่รวบรวม Jobs และ Stages ไว้ภายใน กำหนดค่าการทำงานทั้งหมดของกระบวนการ CI/CD

2.2.5.4 Gitlab Runner

เป็นเครื่องมือที่ใช้ในการประมวลผลการทำงานของ Pipeline CI/CD หากไม่มี Gitlab Runner ตัว Pipeline CI/CD จะไม่ทำงาน

2.3 ภาษาที่ต้องการใช้ในไปป์ไลน์เทมเพลต

2.3.1 YAML

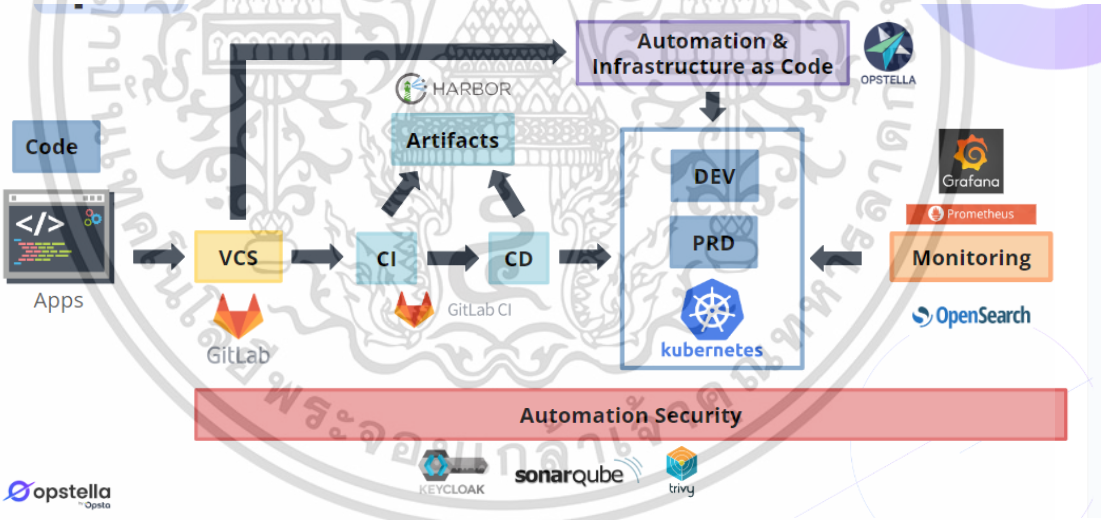
เป็นภาษาโปรแกรมที่ได้รับความนิยมเนื่องจากได้รับการออกแบบมาให้อ่านและเข้าใจง่าย นอกจากนี้ยังสามารถใช้ร่วมกับภาษาโปรแกรมอื่นๆ เนื่องจากความยืดหยุ่นและการเข้าถึง โดยนิยมใช้ในการกำหนดค่าต่าง ๆ

บทที่ 3
วิธีการดำเนินโครงงานสหกิจศึกษา

ในการพัฒนาเทมเพลต CI/CD สำหรับซอฟต์แวร์โดยใช้ GitLab CI/CD ผู้จัดทำได้แบ่งการทำงานเป็น 5 ขั้นตอน ได้แก่ 1) ขั้นตอนการกำหนดสภาพแวดล้อม 2) การตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ด 3) การทดสอบความครอบคลุมของซอร์สโค้ด 4) การบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ด 5) การส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์

3.1 การวิเคราะห์งาน

ดังที่แสดงในรูปที่ 3 DevOps จะไม่ครอบคลุมไปถึงการ Code หรือการพัฒนาซอฟต์แวร์ แต่จะเริ่มตั้งแต่การ Build, Test ไปจนถึงการ Monitor โดยมี Security หรือ Sec ใน DevSecOps อยู่ในทุกขั้นตอน ซึ่งหากนำเอารูปภาพด้านบนไปแจกแจงการทำงานแต่ละส่วน โดยที่ใช้ Opstella platform ในการทำงานจะได้ผลลัพธ์ออกมาเป็นแผนภาพของการทำงานและเครื่องมือที่ใช้ในแต่ละขั้นตอนอย่างที่แสดงในรูปที่ 6



รูป 6 เครื่องมือในแต่ละขั้นตอนของ DevSecOps

จะเห็นได้ว่า ขั้นตอน CI/CD ของเราจะเริ่มตั้งแต่หลังจากขั้นตอน Code นั่นก็คือขั้นตอน Build ไปจนถึงก่อนขั้นตอน Monitor นั่นคือขั้นตอนการ Configure ถ้าหากนำมาแยกตามส่วนของงานที่ได้รับมอบหมาย จะได้ดังนี้

1) ขั้นตอนการกำหนดสภาพแวดล้อม (Configure)

2) การตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ด (Test)

3) การทดสอบความครอบคลุมของซอร์สโค้ด (Test)

4) การบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ด (Build, Package)

5) การส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ (Release)

โดยที่หากนำการทำงานในแต่ละขั้นตอนมาจัดแบ่งออกเป็น Jobs และกำหนดเครื่องมือที่ใช้ในการทำงานแต่ละ Jobs จะได้ออกมาทั้งหมด 7 Jobs ดังที่แสดงในรูปที่ 7 ดังนี้

1) .Init คือขั้นตอนในการกำหนดสภาพแวดล้อมที่จะส่งออกซอฟต์แวร์โดยจะบันทึกไว้ด้วยการใช้เครื่องมือ Artifact ที่อยู่ใน Gitlab ในการบันทึกและส่งต่อสภาพแวดล้อมตามที่กำหนด

2) Code Coverage คือขั้นตอนในการทดสอบความครอบคลุมของซอร์สโค้ดโดยเครื่องมือที่ใช้ในการทดสอบจะขึ้นอยู่กับภาษาที่ใช้ในการพัฒนาซอฟต์แวร์

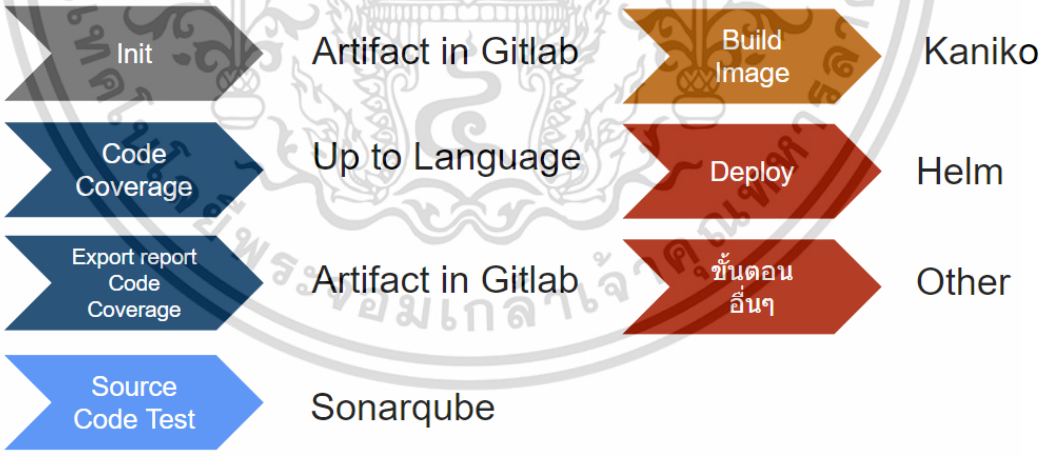
3) Export report Code Coverage คือขั้นตอนในการบันทึกผลของการทดสอบความครอบคลุมของซอร์สโค้ดด้วยการใช้เครื่องมือ Artifact ที่อยู่ใน Gitlab

4) Source Code Test คือขั้นตอนในการตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ดด้วยการใช้เครื่องมือ SonarQube ในการทดสอบ

5) Build Image คือขั้นตอนในการการบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ดด้วยการใช้เครื่องมือ Kaniko ในการทำงาน

6) Deploy คือขั้นตอนในการส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ด้วยการใช้เครื่องมือ Helm ในการส่งออก

7) Jobs สำหรับขั้นตอนอื่น ๆ ตาม Requirements



รูป 7 การทำงานแบบแยกตาม Jobs

ซึ่งหากนำ Jobs มาจัดเรียงกันโดยทำเป็น Stage ไปแบ่งเป็นแต่ละ Stage จะได้ 6 ดังที่แสดงในรูปที่ 8 ดังนี้

1) .Init Stage จะประกอบไปด้วย Jobs Init

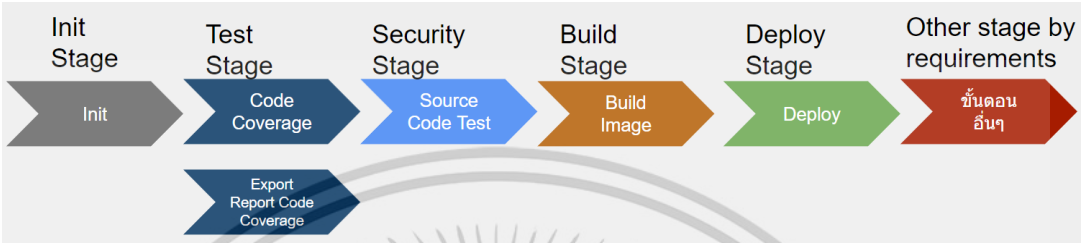
2) Test Stage จะประกอบไปด้วย Jobs Code Coverage และ Jobs Export report Code Coveage

3) Security Stage จะประกอบไปด้วย Jobs Sonarqube Code Test

4) Build Stage จะประกอบไปด้วย Jobs Build Image

5) Deploy Stage จะประกอบไปด้วย Jobs Deploy

6) Other stage by requirements จะประกอบไปด้วย Jobs สำหรับขั้นตอนอื่น ๆ ตาม requirements



รูป 8 การทำงานแบบแยกตาม Stage

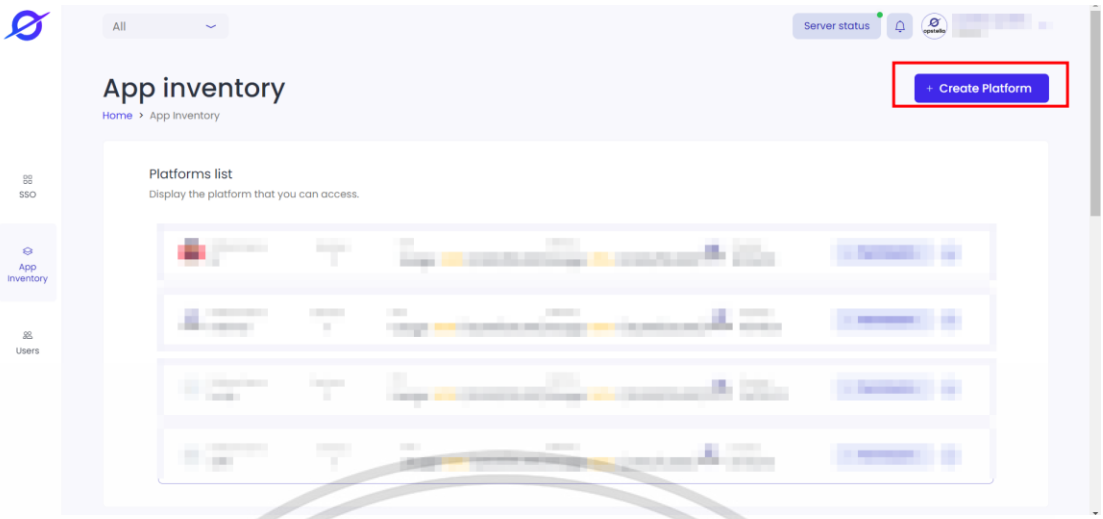
3.2 ขั้นตอนการดำเนินงาน

3.2.1 เตรียมความพร้อมก่อนเริ่มทำเทมเพลต CI/CD

ก่อนจะเริ่มทำเทมเพลต CI/CD จำเป็นต้องเตรียมเครื่องมือที่จำเป็นไว้ก่อนโดยการใช้งาน Opstella platform ดังนี้

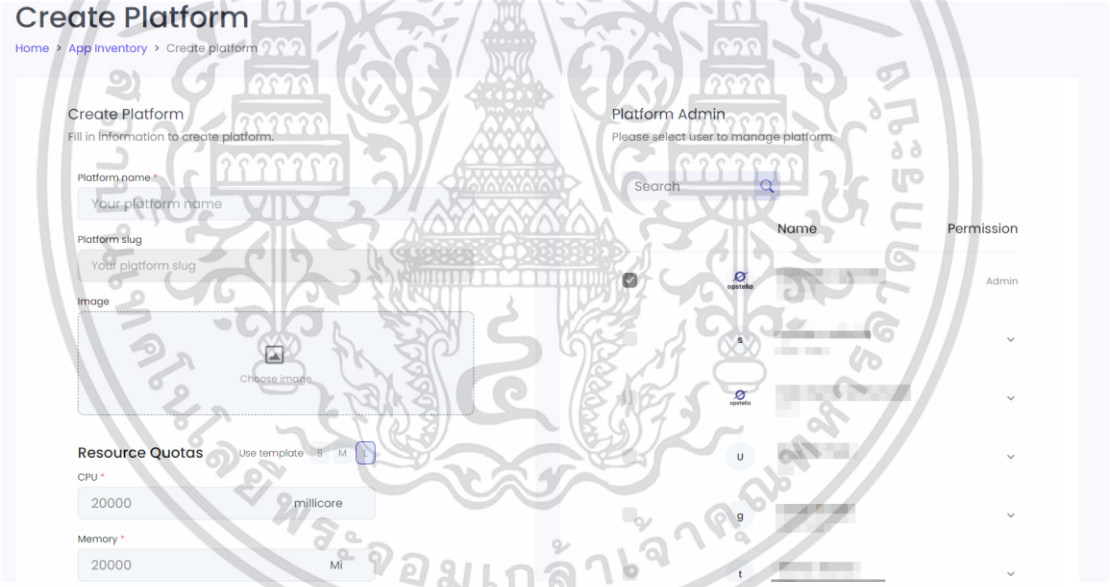
- ล็อคอินเข้าไปที่ Opstella platformด้วยการใส่ Username และ Password ดังที่แสดงในภาพที่ 9 จากนั้นกดปุ่ม Login
- เมื่อLogin สำเร็จจะพบหน้าแรกให้ทำการสร้าง Platform สำหรับโปรเจค โดยคลิกไปที่ปุ่ม Create Platformที่อยู่มุมขวาทางด้านบนดังที่แสดงในภาพที่ 10

รูป 9 Opstella Login Page



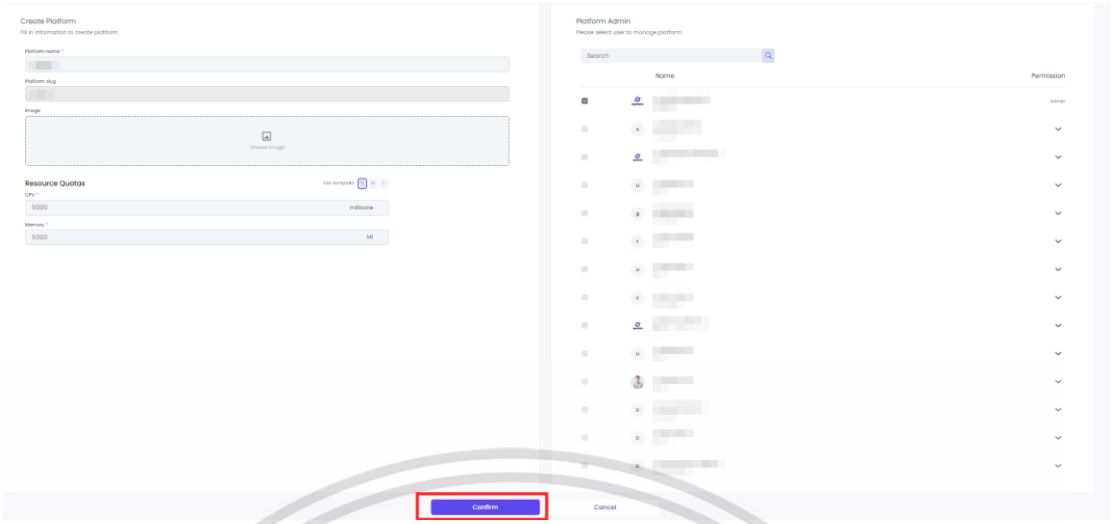
รูป 10 Opstella Default Page

- จะเข้าสู่หน้า Create Platform ให้ทำการกำหนดค่าต่าง ๆ ของ Platform ตามที่ต้องการเช่น Platform name, CPU&Memory Quotas ดังที่แสดงในรูปที่ 11

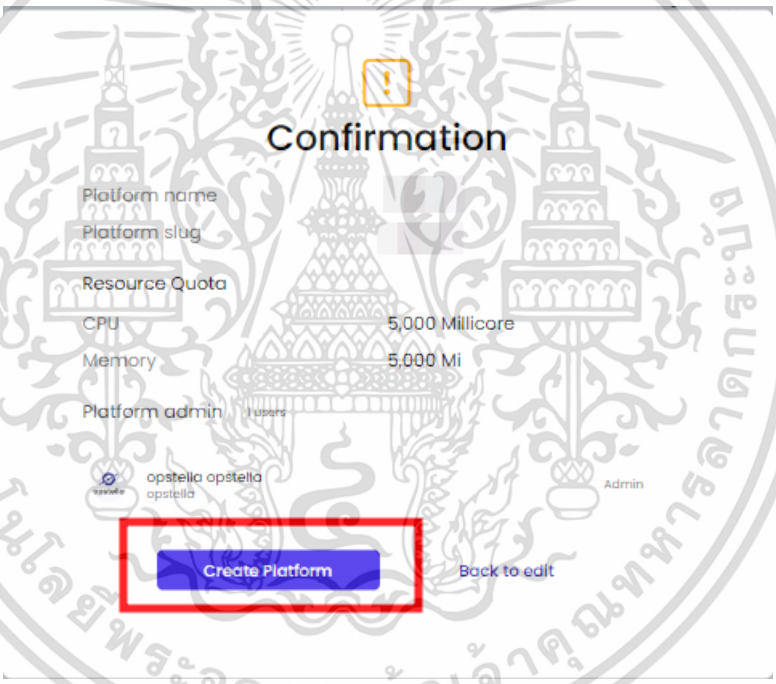


รูป 11 Opstella Create Platform Page

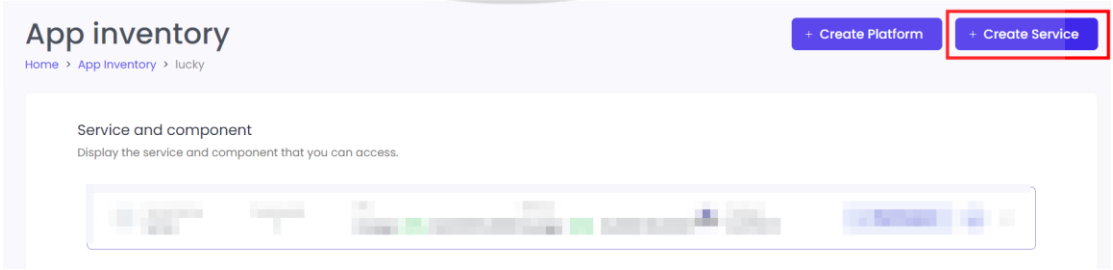
- กดปุ่ม Confirm ที่อยู่ด้านล่างเพื่อยืนยันการสร้างPlatformดังที่แสดงในรูปที่ 12
- จะมีหน้าต่างแจ้งเตือนและสรุปรายละเอียดในสิ่งที่ได้ทำการตั้งค่าอีกครั้งเพื่อยืนยันการสร้าง ให้กดปุ่ม Create Platform เพื่อยืนยัน
- จะมีหน้าต่างแจ้งเตือนและสรุปรายละเอียดอีกครั้งเพื่อยืนยันการสร้าง ให้กดปุ่ม Create Platform เพื่อยืนยันการสร้าง Platform ดังที่แสดงในรูปที่ 13
- จะพบกับหน้า Opstella Platform Page ให้ทำการสร้าง Service ด้วยการกดไปที่ปุ่ม Create Service ที่อยู่มุมบนขวาดังที่แสดงในรูปที่ 14



รูป 12 Confirm button in Opstella Create Platform Page

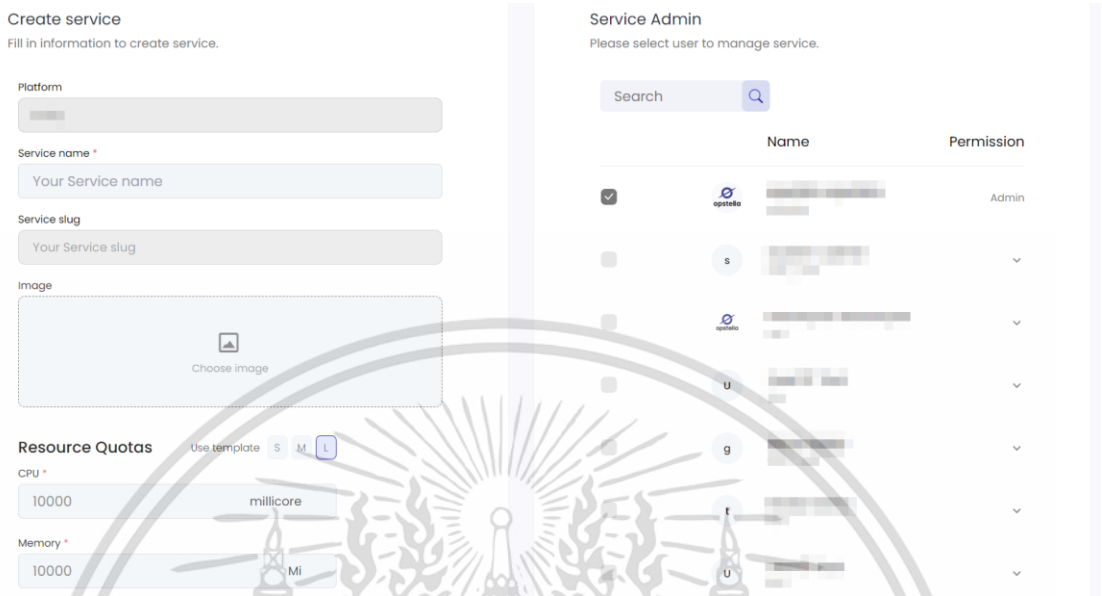


รูป 13 Create Platform button in Confirmation Create Platform



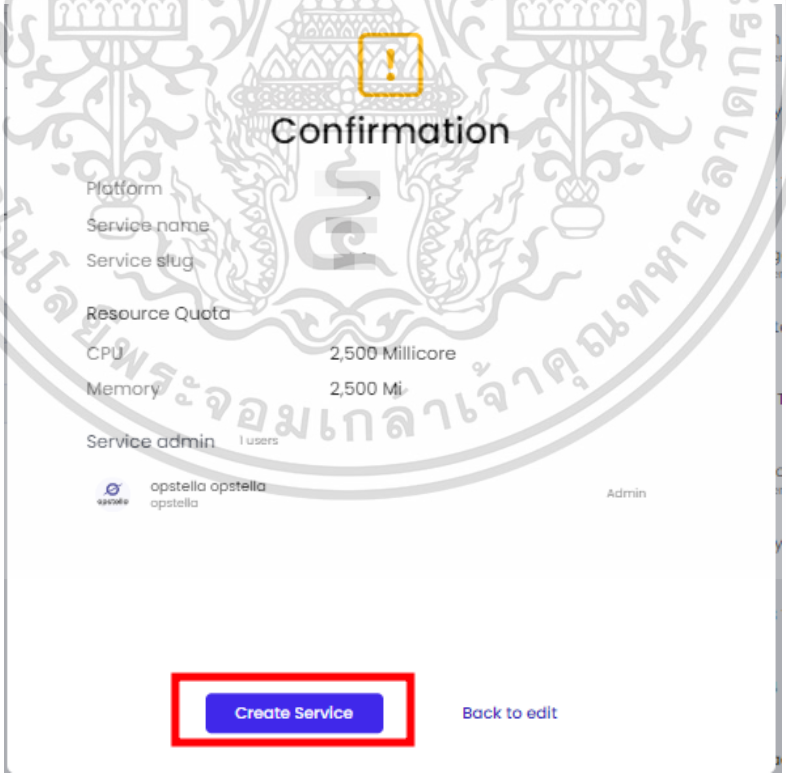
รูป 14 Step to Create Service in Opstella Plarfrom Page

- จะพบกับหน้า Opstella Create Service Page ให้ทำการกำหนดค่าของ Service ตามที่ต้องการดังที่แสดงในรูปที่ 15



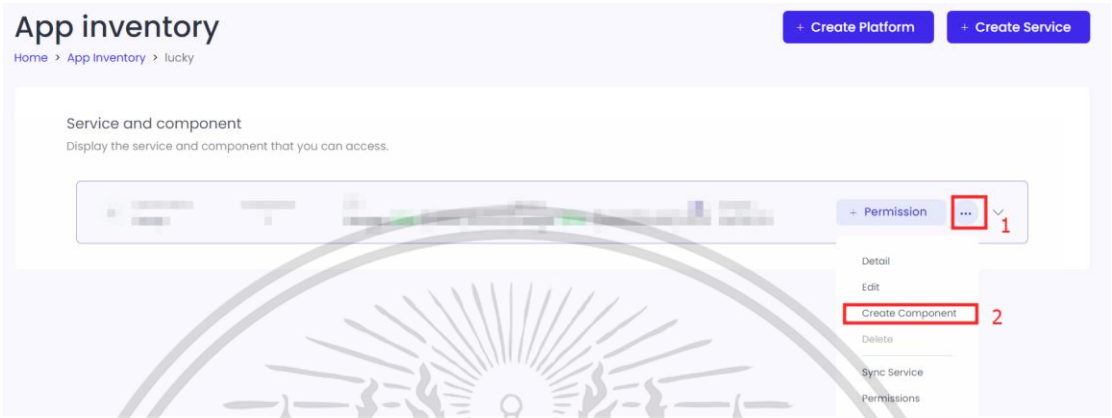
รูป 15 Opstella Create Service Page

- เมื่อกำหนดค่าต่าง ๆ เรียบร้อยแล้วให้กดปุ่ม Confirm เพื่อยืนยันการสร้าง Service



รูป 16 Create Service button in Confirmation Create Service

- จะมีหน้าต่างแจ้งเตือนและสรุปรายละเอียดอีกครั้งเพื่อยืนยันการสร้าง ให้กดปุ่ม Create Service เพื่อยืนยันการสร้าง Service ดังที่แสดงในรูปที่ 16
- จะพบกับหน้า Opstella Service Page ให้คลิกตามภาพที่ 17 เพื่อทำการสร้าง Component หรือโปรเจค

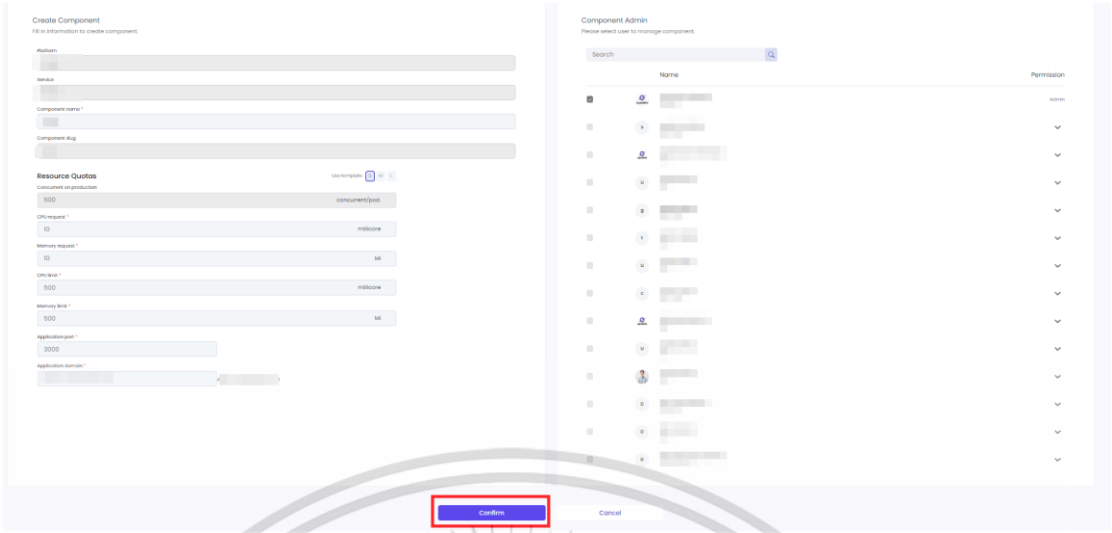


รูป 17 Step to Create Component in Opstella Service Page

- จะพบกับหน้า Opstella Create Component Paage ให้ทำการกำหนดค่าต่าง ๆ สำหรับ Component ดังที่แสดงในภาพที่ 18

รูป 18 Opstella Create Component Page

- จากนั้นกดปุ่ม Confirm เพื่อยืนยันการสร้าง Component ดังที่แสดงในภาพที่ 19

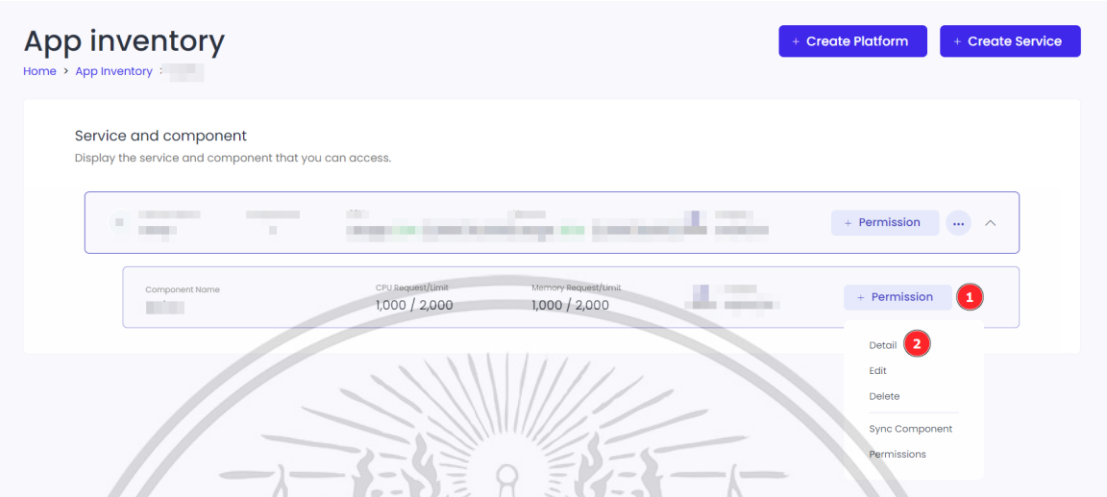


รูป 19 Confirm button in Opstella Create Component Page

- จะมีหน้าต่างแจ้งเตือนและสรุปรายละเอียดอีกครั้งเพื่อยืนยันการสร้าง ให้กดปุ่ม Create Component เพื่อยืนยันการสร้าง Component ดังที่แสดงในภาพที่ 20

รูป 20 Create Component button in Confirmation Create Component

- Opstella จะเตรียมทรัพยากรและตัวแปรที่จำเป็นต่าง ๆ ให้หลังจากสร้าง Component จากนั้นให้ทำการกดเข้าไปที่ Gitlab repository ที่สร้างโดย Opstella platform โดยคลิกตามขั้นตอนในภาพที่ 21



รูป 21 Step to go to Opstella Component Detail Page

- จะเจอกับหน้าOpstella Component Detail

รูป 22 Opstella Component Detail Page

- เลื่อนหน้าจอลงจนเจอกับเครื่องมือต่างๆที่ Opstella Platform เตรียมไว้ 7 อย่าง ดังนี้

1).Gitlab

2) Grafana

3) Harbor

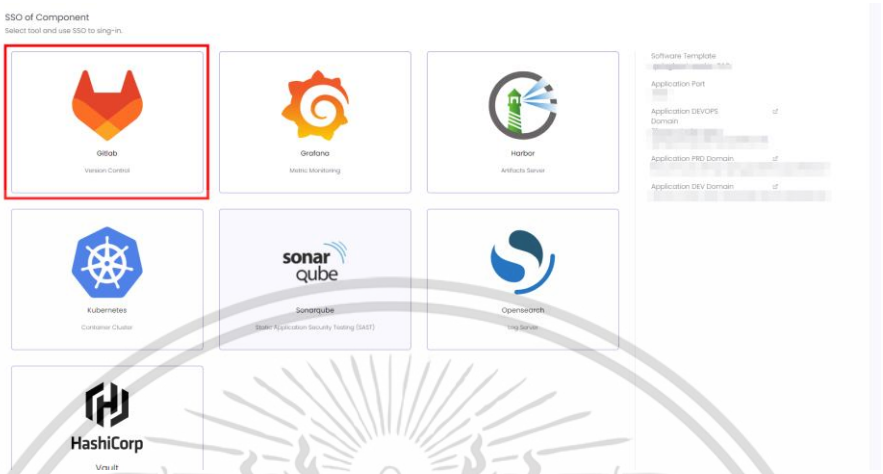
4) Kubernetes

5) SonarQube

6) Opensearch

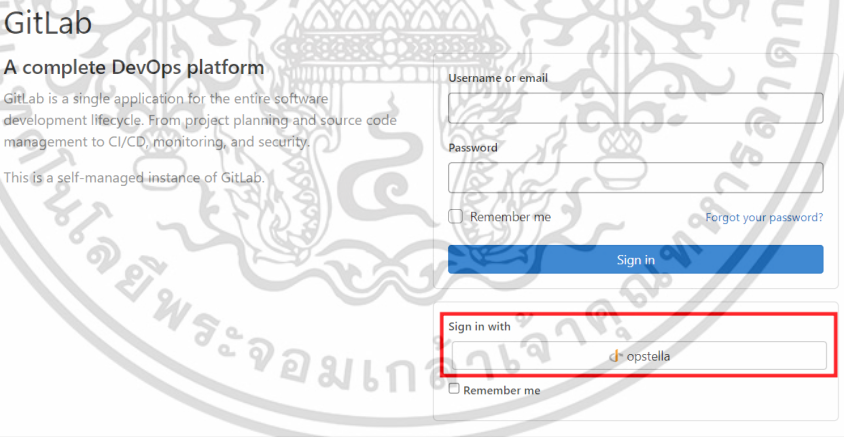
7) HashiCorp Vault

- กดไอคอน Gitlab เพื่อไปยังหน้า Login ของ Gitlab ดังที่แสดงในรูปที่ 23



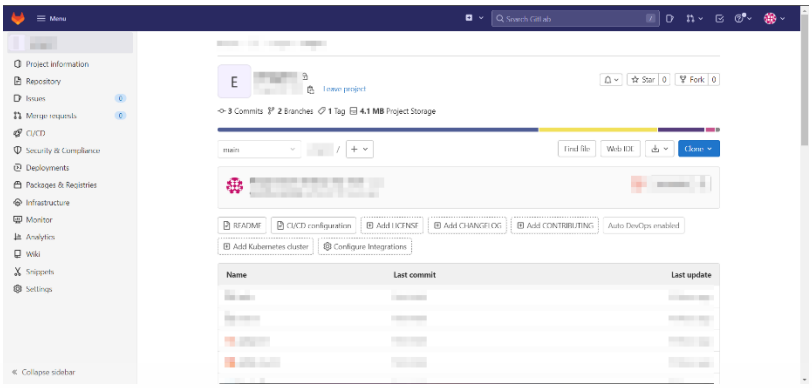
รูป 23 Step to go to Gitlab

- กดปุ่ม opstella ที่อยู่ด้านล่าง Sign in with เพื่อเข้าสู่ Gitlab โดยใช้ Opstella userดังที่แสดงในรูป 24



รูป 24 Step to Sign in to Gitlab with Opstella

- จะเจอกับหน้า Repository ของ Component ที่เราสร้างไว้ดังที่แสดงในรูป 25



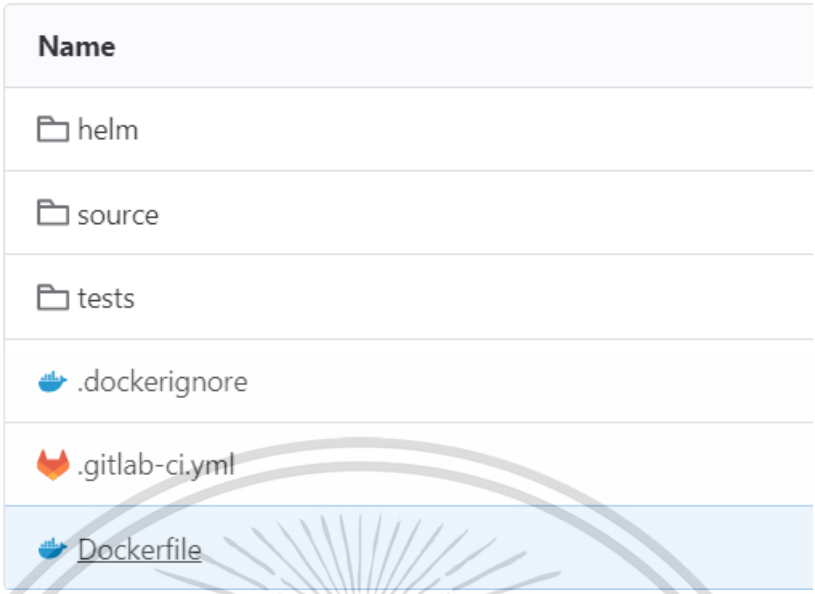
รูป 25 Repository in Gitlap Page

- จากนั้นให้ทำการเตรียมไฟล์ต่าง ๆ ดังนี้
- สร้าง Folder “source” จากนั้นย้ายทุกไฟล์ของซอร์สโค้ดโปรเจคไปไว้ใน source
- สร้าง Dockerfile เก็บไว้ในระดับเดียวกันกับ source
- สร้าง Folder “helm” เก็บไว้ในระดับเดียวกันกับ source
- สร้างไฟล์ helm value ตามแต่ละ ENV (Environment) เข้าไปเก็บไว้ใน helm โดยมีรูปแบบ “values-ชื่อของ ENV.yaml” โปรเจคที่ได้รับใช้ 4 ENV ต้องสร้าง 4 ไฟล์ดังที่แสดงไว้ในตาราง 3

ชื่อ ENV	ชื่อไฟล์ helm values
develop	values-develop.yaml
uat	values-uat.yaml
sit	values-sit.yaml
production	values-production.yaml

ตาราง 3 ตารางการกำหนดชื่อไฟล์ value ของ helm

- นำซอร์สโค้ดทั้งหมดไปเก็บไว้บน Gitlab Repository ของโปรเจค
- เมื่อสร้างเสร็จแล้ว จะได้โครงสร้างของ repository ดังที่แสดงไว้ในรูป 26 ดังนี้



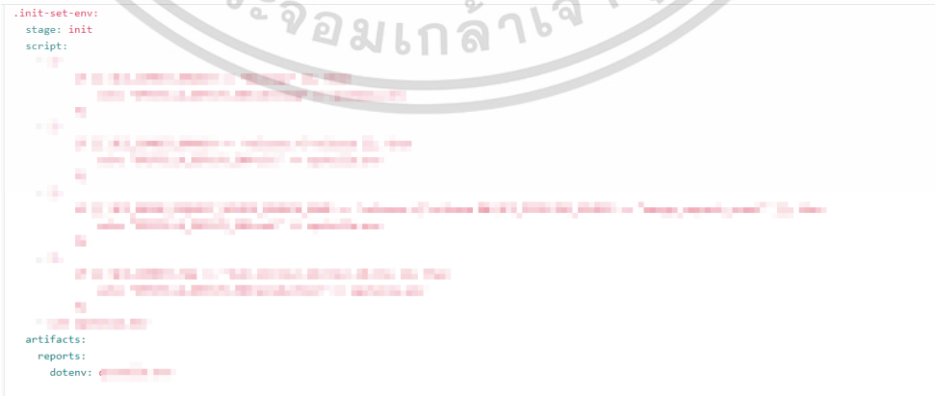
รูป 26 โครงสร้างของ Repository

- สร้าง Repository เพื่อเก็บ Pipeline template และ Jobs บน Gitlab

3.2.2 สร้าง Folder “jobs” ใน Gitlab Repository ของ Pipeline template

เพื่อรวบรวม Jobs ต่าง ๆ ไว้ภายใต้ Folder “jobs” ให้ง่ายต่อการจัดการดูแล

- สร้าง Folder แยกสำหรับแต่ Jobs เพื่อให้สามารถนำ Jobs ไปประกอบเป็น pipeline template ได้อย่างอิสระ จึงแยกแต่ละ Jobs ออกเป็น Folder สำหรับ Jobs ที่เตรียมไว้ใช้ในโปรเจคมีดังนี้
- Jobs init อยู่ใน Stage init ใช้สำหรับขั้นตอนการกำหนดสภาพแวดล้อมให้กับไปป์ไลน์โดยมีการกำหนดเงื่อนไขในการกำหนดสภาพแวดล้อม โดยสคริปต์การทำงานจะทำงานตามเงื่อนไขเพื่อเลือกสภาพแวดล้อมจากนั้นจะใช้ Artifacts ที่อยู่ใน Gitlab เพื่อทำการบันทึกค่าและส่งต่อไปที่ Jobs อื่น ๆ ในไปป์ไลน์ดังที่แสดงไว้ในรูปที่ 27



รูป 27 .init-set-env jobs in Init stage

- Jobs dotnet-unit-test อยู่ใน Stage test ใช้สำหรับขั้นตอนการทดสอบความครอบคลุมของซอร์สโค้ดโดยจะทำการนำซอร์สโค้ดไปทดสอบด้วยคำสั่งเฉพาะของแต่ละภาษาที่ใช้ในการพัฒนาซอฟต์แวร์ ดังที่แสดงในรูป 28 จากนั้นจะใช้ Jobs dotnet-set-report-env ที่ Stage เดียวกันในการบันทึกผลการทดสอบความครอบคลุมของซอร์สโค้ดด้วย Artifact ที่อยู่ใน Gitlab เพื่อทำการส่งต่อผลลัพธ์ไปยัง Jobs sonarqube-scan ดังที่แสดงในรูป 29 โดยมีเงื่อนไขว่าถ้าหากไม่มี Jobs dotnet-unit-test แล้ว Jobs dotnet-set-report-envจะไม่เกิดขึ้น โดยที่ผลลัพธ์การทดสอบจะสามารถส่งต่อขึ้นไปยัง SonarQube Dashboard ได้ผ่านตัวแปรที่อยู่ใน Jobs sonarqube-scan

รูป 28 .dotnet-unit-test jobs in Test stage

รูป 29 .dotnet-set-report-env in Test stage

- Jobs kaniko-execute อยู่ใน Stage build ใช้สำหรับขั้นตอนการบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ดโดยใช้ Dockerfile ที่ได้สร้างเตรียมไว้ใน repository ซึ่งจะมีการกำหนดค่าต่าง ๆ เช่น ชื่อของ image มาจากตัวแปรที่ Opstella platform เตรียมไว้ให้ จากนั้นใช้คำสั่งเฉพาะในการยืนยันตัวตนเพื่อให้มีสิทธิ์ในการนำอิมเมจไปเก็บไว้บน Private Registry ที่ชื่อว่า Harbor ดังที่แสดงในรูป 30 และ รูป 31

รูป 30 .kaniko-execute in Build stage part 1

รูป 31 .kaniko-execute in Build stage part 2

- Jobs sonarqube-scan ใน Stage securityใช้สำหรับขั้นตอนการตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ด โดยใช้คำสั่งเฉพาะในการเรียกใช้ Sonarqube โดยมีการกำหนดค่าต่าง ๆ เป็นตัวแปรไว้ เช่น Path to file ที่จะทำการตรวจสอบช่องโหว่และยังมีการตรวจสอบเงื่อนไขการทำงานที่เชื่อมโยงกับ Stage Test ว่ามีการส่งผ่านบันทึกการทดสอบหรือไม่ ถ้าหากมีก็จะนำบันทึกผลการทดสอบนั้นขึ้นไปบน SonarQube Dashboard ด้วยผ่านตัวแปรที่กำหนดไว้ดังที่แสดงในรูป 32

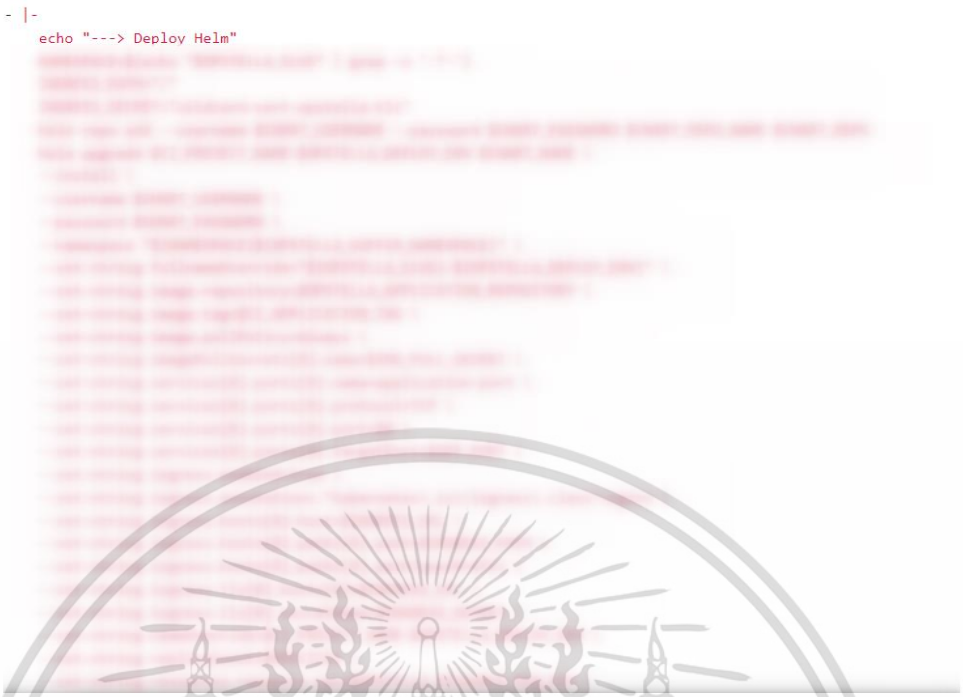
รูป 32 .sonarqube-scan in Security stage

- Jobs helm-deploy ใน Stage deploy ใช้สำหรับการส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์ โดยมีการส่งออกไปตามสภาพแวดล้อมที่กำหนดไว้จาก Stage init และการกำหนดค่าอื่น ๆ ที่ส่งมาด้วยไปป์ไลน์ ดังที่แสดงไว้ในรูป 33 ถึง รูป 37

รูป 33 .helm-deploy in Deploy stage part 1

รูป 34 .helm-deploy in Deploy stage part 2

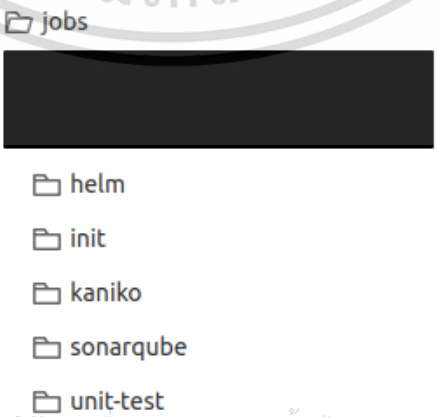
รูป 35 .helm-deploy in Deploy stage part 3



รูป 36 .helm-deploy in Deploy stage part 4

รูป 37 .helm-deploy in Deploy stage part 5

ซึ่งเมื่อดูในภาพรวม jobs จะมีโฟลเดอร์ย่อย ๆ แยกไปตามแต่ละ Jobs ที่ต้องใช้งานดังที่แสดงไว้ในรูป 38 ดังนี้

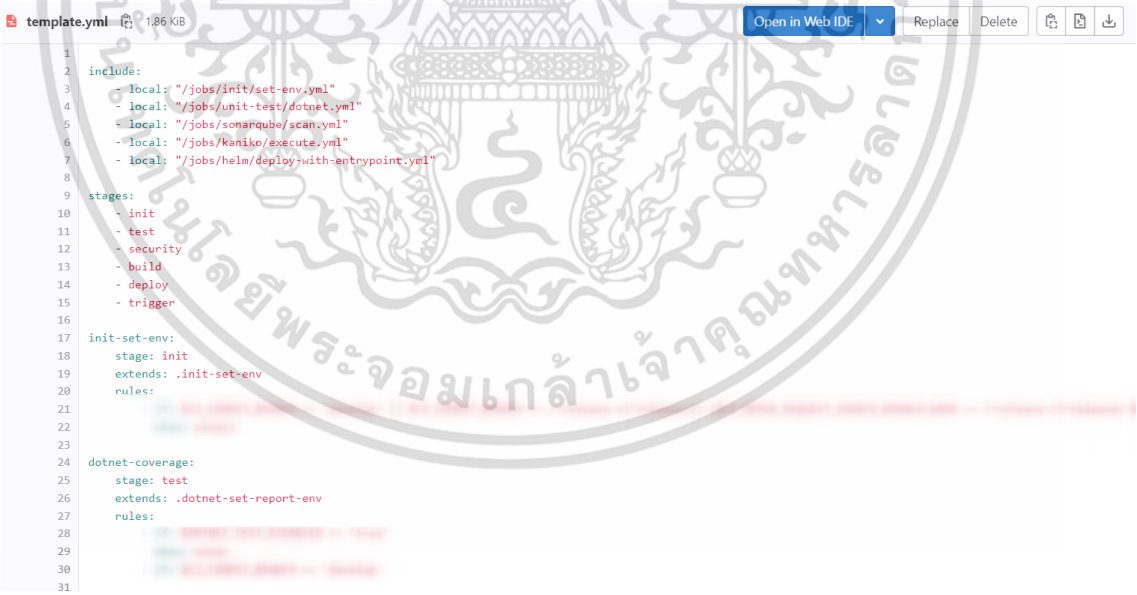


รูป 38 Jobs ทั้งหมดที่มี

3.2.3 สร้าง Folder “pipelines” ใน Gitlab Repository ของ Pipeline template

เพื่อรวบรวม Pipeline ต่าง ๆ ไว้ใน Folder “pipelines” ให้ง่ายต่อการจัดการดูแล

- สร้าง Folder แยกสำหรับแต่ละ Pipeline เพื่อแยก Template สำหรับแต่ละเครื่องมือ ดังนั้นจึงสร้าง Folder ที่ต้องการใช้ในโปรเจคไว้ดังนี้
- Folder 01-dotnet-solution.yml ใช้สำหรับ ซอฟแวร์ที่พัฒนาด้วย .NET โดยแบ่งเป็น template.yml ซึ่งจะมีการนำ Jobs ที่ต้องการใน folder ต่าง ๆ ของ Jobs มาใช้งานผ่านการ include จากนั้นทำการกำหนด Stage ในการทำงานด้วย stages ดังที่แสดงไว้ในรูปที่ 39 ถัดมาเป็นการสร้าง Jobs ขึ้นมาในไฟล์ template.yml โดยจะนำ Jobs ที่ include มาใช้ โดยใช้ extends ในการระบุ Jobs ที่ต้องการมาทำงานร่วมกันกับ Jobs ที่สร้างใน template ดังนี้
- Jobs init-set-env จะทำการ extends Jobs .init-set-env ที่ include มาโดยกำหนดให้อยู่ใน stage init จากนั้นกำหนดเงื่อนไขการทำงานโดยใช้ rules ดังแสดงในรูป 39
- Jobs dotnet-coverage จะทำการ extends Jobs .dotnet-set-report-env ซึ่งทำงานหลังจาก Jobs dotnet-unit-test อีกที ดังนั้นการทำงานใน Jobs dotnet-coverage นี้จะทำทั้ง Jobsที่กล่าวมา โดยกำหนดให้อยู่ใน stage test และกำหนดเงื่อนไขการทำงานผ่าน rules ดังที่แสดงในรูป 39



รูป 39 Pipeline template part 1

- Jobs sonarqube-scan จะทำการ extends Jobs .sonarqube-scan โดยกำหนดให้อยู่ใน stage security และกำหนดเงื่อนไขอื่น ๆ ตาม requirements ดังที่แสดงในรูป 40
- Jobs kaniko-baseimage จะ extends Jobs .kaniko-execute โดยกำหนดให้อยู่ใน stage build และกำหนดเงื่อนไขอื่น ๆ ตาม requirements ดังที่แสดงในรูป 40

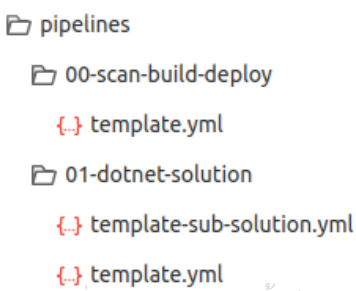
- Jobs kaniko-build จะทำการ extends Jobs .kaniko-execute โดยกำหนด ให้อยู่ใน stage build และกำหนดเงื่อนไขอื่น ๆ ตาม requirements ดังที่แสดงในรูป 40
- Jobs helm-deploy จะทำการ extends Jobs .helm-deploy โดยกำหนด ให้อยู่ใน stage deploy และกำหนดเงื่อนไขอื่น ๆ ตาม requirements ดังที่แสดงในรูป 40

รูป 40 Pipeline template part 2

- ไฟล์ template-sub-solution.yml ซึ่งจะมีการนำเข้า Jobs ผ่าน include และมี Stage และ Jobs เพิ่มเติมตาม requirements ดังที่แสดงในรูป 41

รูป 41 Pipeline sub-template

เมื่อดูภาพรวมของ pipelines จะได้ออกมาเป็นโครงสร้างดังรูป 42 ดังนี้

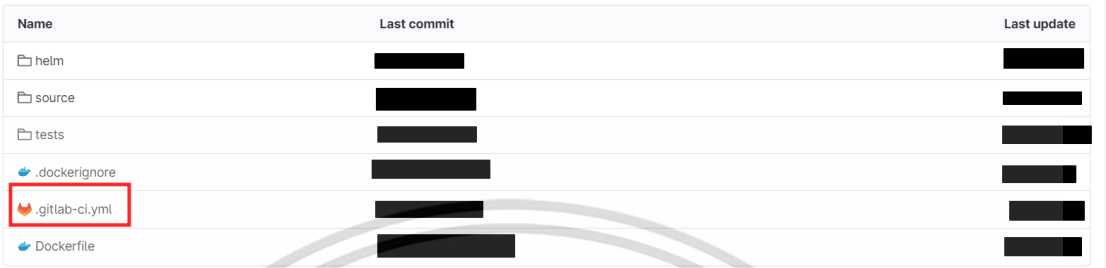


รูป 42 template ทั้งหมดที่มี

3.2.4 นำ Pipeline เทมเพลตมาใช้ใน Gitlab Repository ของโปรเจค

การนำ Pipeline เทมเพลตมาใช้ใน Gitlab Repository ของโปรเจค มีการเตรียมการดังนี้

- กดเข้าไปที่ “.gitlab-ci.yaml” ใน Gitlab Repository ของโปรเจคดังที่แสดงในรูป 43

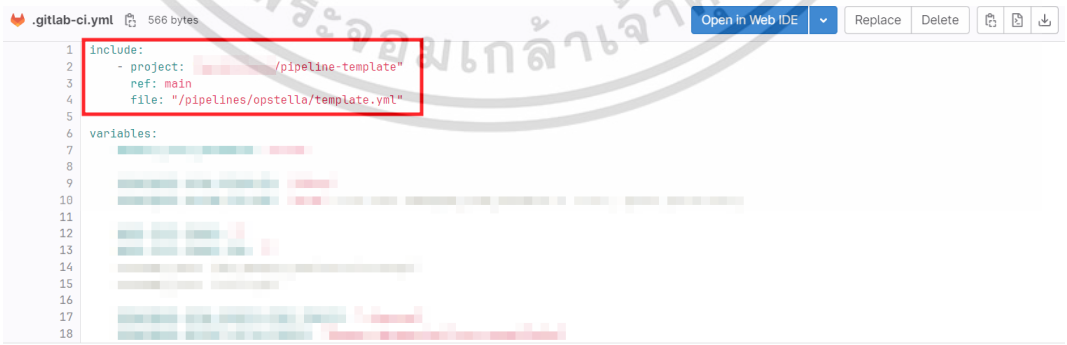


รูป 43 .gitlab-ci.yaml file in Repository

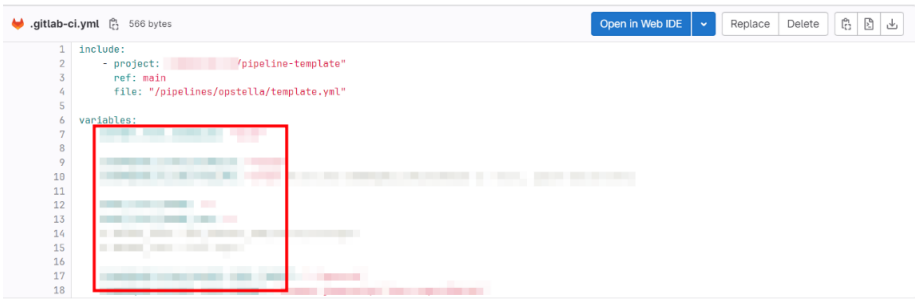
- จะพบกับหน้า .gitlab-ci.yml detail ดังรูป 44

รูป 44 .gitlat-ci.yml detail

- แล้วกำหนดค่าโดยนำเข้าไปป์ไลน์เทมเพลตผ่าน include ดังที่แสดงในรูป 45 จากนั้นกำหนดค่าเงื่อนไขที่ต้องการให้ทำงานต่างๆผ่าน variables ดังที่แสดงไว้ในรูป 46



รูป 45 การนำไปป์ไลน์เทมเพลตเข้ามาใช้



รูป 46 เงื่อนไขการทำงานที่กำหนด

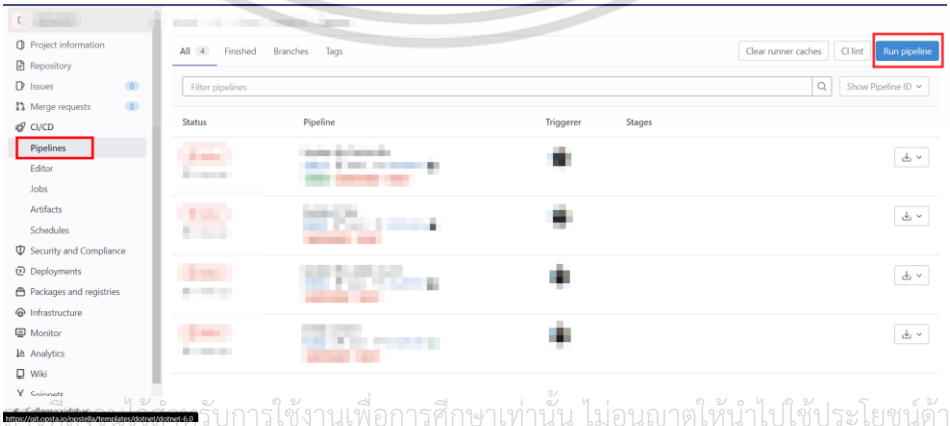
- กดปุ่ม Commit Changes เพื่อบันทึกการกำหนดค่าดังที่แสดงในรูป 47

รูป 47 Commit Change button in .gitlab-ci.yaml in Editor mode

3.2.5 ทำ CI/CD ด้วย Pipeline

การทำ CI/CD จาก Pipeline ที่เรากำหนดค่าไว้ มีหลายวิธีดังต่อไปนี้

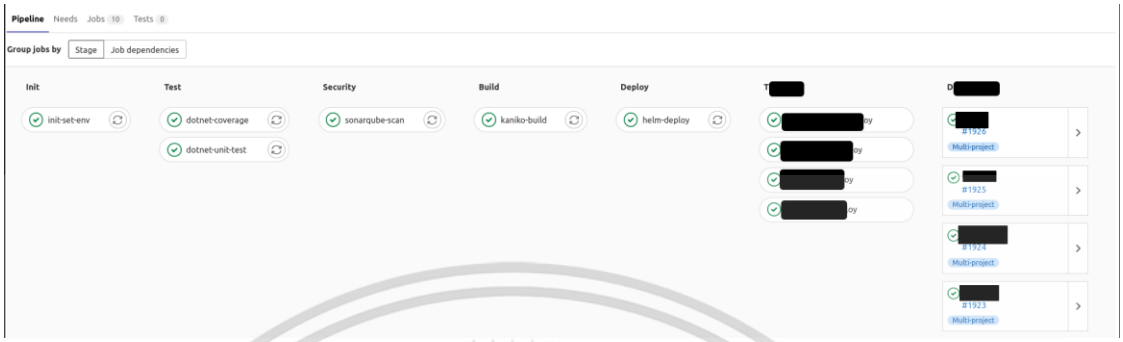
- เมื่อกดปุ่มบันทึกการกำหนดค่าPipelineจะทำงานทุกครั้งที่มีการกดปุ่มบันทึกการกำหนดค่าใน Repository นั้น ๆ
- เมื่อกดปุ่ม “Run Pipeline” เป็นวิธีการแบบ Manual (กำหนดเอง) เพื่อให้ Pipeline ทำงานดังรูป 48



รูป 48 Step to Run pipeline manual

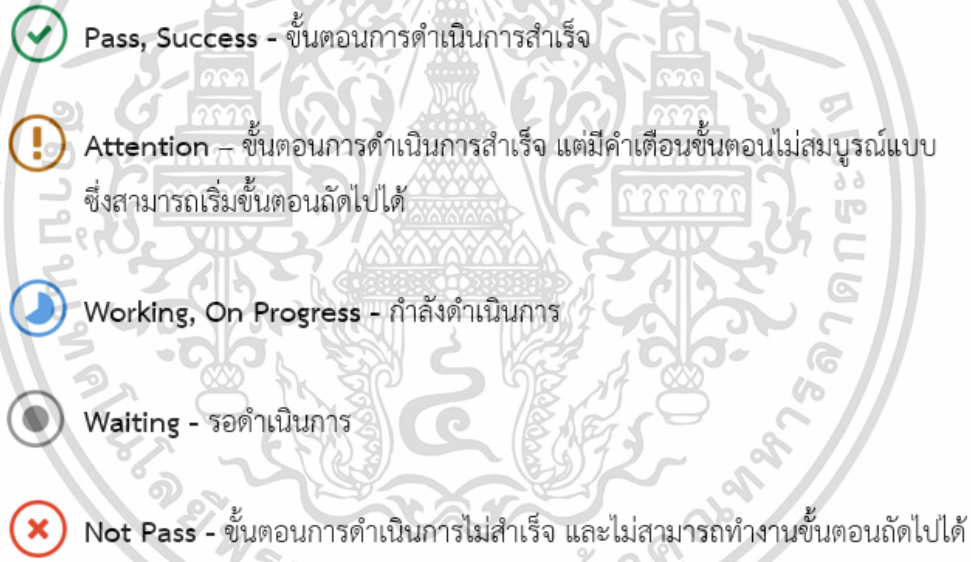
3.2.6 ดูผลลัพธ์การทำงานของ Pipeline

Gitlab CI/CD มีการติดตามการทำงานของ Pipeline และแสดงผลออกมาเป็นบันทึกได้ดังรูป 49 และสามารถดูคำอธิบายความหมายของสถานะของ Pipeline ได้ในรูป 50



รูป 49 ผลลัพธ์การทำงานของไปป์ไลน์ทั้งหมด

คำอธิบายสถานะของ Pipeline



รูป 50 คำอธิบายสถานะของ Pipeline

บทที่ 4
ผลการดำเนินโครงงานสหกิจศึกษา

จากการดำเนินการทดสอบไปป์ไลน์เทมเพลตตามขั้นตอนการทำงานที่กำหนดไว้ตาม Stage ของไปป์ไลน์ ได้ผลลัพธ์การดำเนินการดังนี้

4.1 ผลการทำงานของขั้นตอนการกำหนดสภาพแวดล้อม

ประกอบไปด้วย 1 Jobs นั่นคือ Jobs init-set-env ที่อยู่ในStage Init ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูป 51

รูป 51 ผลลัพธ์การทำงานของ .init-set-ent Jobs

4.2 ผลการทำงานของขั้นตอนการทดสอบความครอบคลุมของซอร์สโค้ด

ประกอบไปด้วย 2 Jobs ดังนี้

Jobs dotnet-coverage

dotnet-unit-test

โดยที่ทั้งสอง Jobs อยู่ในStage Test เช่นเดียวกัน

4.2.1 ขั้นตอนการทดสอบความครอบคลุมของซอร์สโค้ด

Jobs dotnet-coverage ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูปที่ 52

4.2.2 ขั้นตอนการบันทึกผลการทดสอบความครอบคลุมของซอร์สโค้ด

Jobs dotnet-unit-test ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูปที่ 53

รูป 52 ผลลัพธ์การทำงานของ dotnet-coverage Jobs

รูป 53 ผลลัพธ์การทำงานของ dotnet-unit-test Jobs

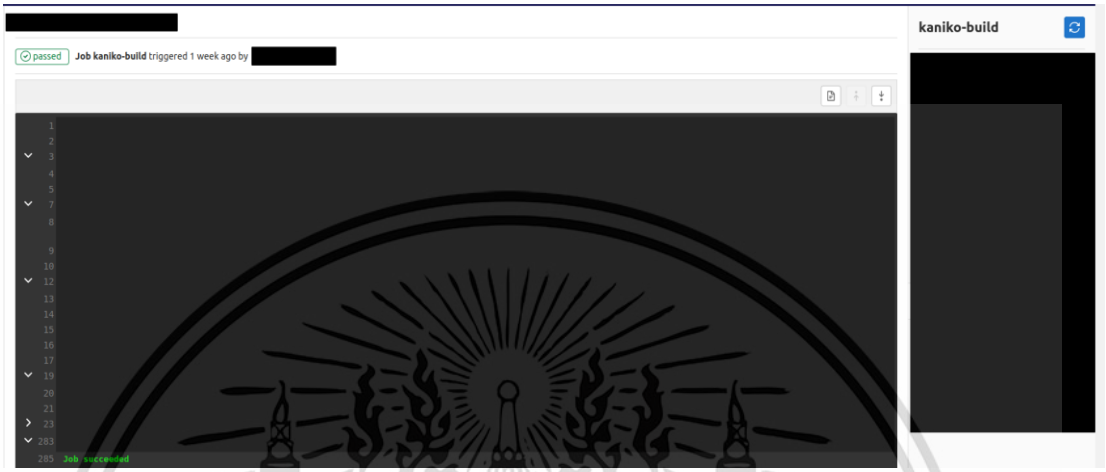
4.3 ผลการทำงานของขั้นตอนการการตรวจสอบช่องโหว่ความปลอดภัยของซอร์สโค้ด

ประกอบไปด้วย 1 Jobs นั่นคือ Jobs sonarqube-scan ที่อยู่ในStage Security ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูป 54

รูป 54 ผลลัพธ์การทำงานของ sonarqube-scan Jobs

4.4 ผลการทำงานของขั้นตอนการการบีบอัดซอร์สโค้ดและการสร้างอิมเมจของซอฟต์แวร์จากซอร์สโค้ด

ประกอบไปด้วย 1 Jobs นั่นคือ Jobs kaniko-build ที่อยู่ในStage build ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูป 55



รูป 55 ผลลัพธ์การทำงานของ kaniko-build Jobs

4.5 ผลการทำงานของขั้นตอนการส่งออกอิมเมจของซอฟต์แวร์ขึ้นไปทำงานบนคลาวด์

ประกอบไปด้วย 1 Jobs นั่นคือ Jobs helm-deploy ที่อยู่ในStage deploy ผลลัพธ์คือขั้นตอนการดำเนินการสำเร็จดังที่แสดงในรูป 56

รูป 56 ผลลัพธ์การทำงานของ helm-deploy Jobs

โดยผลการดำเนินโครงงานสหกิจศึกษานี้ซึ่งเป็นส่วนหนึ่งของ Opstella CI/CD service ซึ่งครอบคลุมตั้งแต่การนำซอร์สโค้ดขึ้นไปเก็บไว้บนเวอร์ชันคอนโทรลหรือ Gitlab และขั้นตอนการทำงานต่าง ๆ ที่อยู่บนไปป์ไลน์เทมเพลต สามารถทำงานแทนที่วิธีการทำงานโดยใช้ไปป์ไลน์ดั้งเดิมได้โดยมีผลลัพธ์สำเร็จในทุกขั้นตอน และในปัจจุบันโปรเจคทุกโปรเจคที่กำลังพัฒนาหรือพัฒนาเสร็จแล้วที่อยู่บน Opstella Platfrom กำลังทำการปรับเปลี่ยนเพื่อย้ายมาใช้งานไปป์ไลน์เทมเพลตที่ได้ทำการพัฒนาในสหกิจศึกษาครั้งนี้แทนที่การพัฒนารูปแบบเก่าที่ใช้ไปป์ไลน์ดั้งเดิม

บทที่ 5
สรุปผลและข้อเสนอแนะโครงงานสหกิจศึกษา

ไปป์ไลน์เทมเพลต CI/CD เป็นไปป์ไลน์เทมเพลตที่พัฒนาโดยใช้ GitLab CI เทมเพลตนี้จัดทำขึ้นเพื่อลดความซ้ำซ้อนในการทำงานของขั้นตอนต่าง ๆ ให้เป็นไปตามอัตโนมัติ โดยมีความสามารถในการตรวจสอบซอร์สโค้ด การทดสอบซอฟต์แวร์ การบีบอัดซอร์สโค้ดและสร้างเป็นอิมเมจ การจัดการคอนเทนเนอร์หลังจากที่ส่งออกซอฟต์แวร์ไปแล้ว แยกตามสภาพแวดล้อมที่ต้องการ

การพัฒนาไปป์ไลน์เทมเพลต CI/CD ผู้จัดทำได้นำ SonarQube มาใช้ในส่วนของการทดสอบซอร์สโค้ด .NET framework ในส่วนของการทดสอบซอฟต์แวร์ Kaniko ในส่วนของการสร้างอิมเมจและ Helm ในส่วนของการส่งออกซอฟต์แวร์ที่เป็นอิมเมจ Kubernetes ในส่วนของการจัดการคอนเทนเนอร์ของซอฟต์แวร์หลังจากที่ส่งออกไปแล้วซึ่งเป็นเครื่องมือที่ติดตั้งไว้บน Opstella platform ของทางบริษัท ออพซ์ตา (ประเทศไทย) จำกัด

การปฏิบัติสหกิจศึกษาในครั้งนี้ ผู้จัดทำได้เรียนรู้ถึงการพัฒนาไปป์ไลน์เทมเพลต CI/CD สำหรับซอฟต์แวร์ .NET รวมถึงการใช้งาน Opstella platform และเครื่องมือ ต่าง ๆ บนนั้น ทำให้ได้เข้าใจในระบบการทำงานของจริงในบริษัท ที่ต้องการระเบียบ ความตรงต่อเวลาและการแก้ไขปัญหาเฉพาะหน้าเมื่อมีเหตุฉุกเฉิน รวมถึงต้องรอบคอบตั้งแต่การเริ่มวางแผนพัฒนาไปจนถึงขั้นตอนสุดท้าย

5.1 สรุปผลการปฏิบัติงานสหกิจศึกษา

ไปป์ไลน์ CI/CD เทมเพลต แบ่งออกเป็น 5 ส่วน ได้แก่ การกำหนดตัวแปรสภาพแวดล้อม การทดสอบซอฟต์แวร์ การทดสอบซอร์สโค้ด การบีบอัดซอร์สโค้ดและสร้างเป็นอิมเมจและการส่งออกอิมเมจของซอฟร์แวร์ ใช้เวลาการทำทั้งหมด 7 เดือนและสิ่งที่ได้เรียนรู้จากการทำงานครั้งนี้มีดังนี้

1. การใช้ Gitlab CI ในการทำไปป์ไลน์และไปป์ไลน์เทมเพลต
2. การใช้ .NET framwork ในการทดสอบซอฟต์แวร์
3. การใช้ SonaQube ในการทดสอบซอร์สโค้ด
4. การใช้ Kaniko ในการบีบอัดซอร์สโค้ดและสร้างเป็นอิมเมจ
5. การใช้ Helm ในการส่งออกซอฟต์แวร์
6. การใช้ Kubernetes ในการจัดการคอนเทนเนอร์

ปัญหาที่พบจากการทำงานครั้งนี้คือ การต้องเรียนรู้เครื่องมือใหม่ ๆ จำนวนมากทำให้ต้องใช้เวลาส่วนมากในตอนต้นไปกับการทำความเข้าใจและทดลองใช้เครื่องมือ

5.2 ข้อเสนอแนะ

1. เทมเพลตนี้พัฒนาสำหรับใช้กับ Gitlab CI เท่านั้นรวมถึงเครื่องมือต่าง ๆ ที่ใช้เป็นเครื่องมือที่ติดตั้งมาให้พร้อมบน Opstella platform หากต้องการนำเทมเพลตนี้ไปใช้ควรใช้บริการของ Opstella
2. เงื่อนไขในการทำงานต่าง ๆ ของไปป์ไลน์เทมเพลตทำมาตาม requirement เท่านั้น หากต้องการการทำงานอื่น ๆ เพิ่มเติมจำเป็นต้องกำหนดเพิ่มเอง

เอกสารอ้างอิง

VISRUT MANUNPON. 2021. **DevSecOps คืออะไร! ทำไมทุกองค์กรจึงให้ความสำคัญ**. [Online].Available https://www.proen.cloud/th/blog/devsecops

Piyapan, Natchaya. n.d. **มารู้จักกับ CI/CD ตัวช่วยให้งานโปรแกรมเมอร์ง่ายขึ้น**. [Online]. Available https://www.codium.co/blogs/33-CICD

supasiri.s. n.d. **WHAT IS SOFTWARE PERFORMANCE TESTING?.** [Online]. Available https://www.aware.co.th/software-performance-testing/

mk. 2018. **รู้จัก Container มันคืออะไร แตกต่างจาก Virtualization อย่างไร?.** [Online]. Available https://www.blognone.com/node/105928

สายธาร มั่นเกตุวิทย์. 2022. **Containers คืออะไร ?**. [Online]. Available https://www.mindphp.com /คู่มือ/73-คืออะไร/9187-containers.html

Opstella. 2022. **About OPSTELLA**. [Online]. Available https://www.opstella.com/#about

Jacob Schmitt. 2023. **What is Helm? A complete guide**. [Online]. Available https://circleci.com/blog/what-is-helm/

RAJESH KUMAR. 2022. **What is SonarQube and How it works? An Overview and Its Use Cases**. [Online]. Available https://www.devopsschool.com/blog/what-is-sonarqube-and-how-it-works-an-overview-and-its-use-cases/

ETSI. n.d. **Use kaniko to build Docker images (FREE).** [Online]. Available https://forge.etsi.org/rep/help/ci/docker/using_kaniko.md

Microsoft. n.d. **What is .NET?.** [Online]. Available https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

Thanakrit Buranakarn. 2022. **Monorepo กับการจัดการ Repository ในระดับ Mega Project**. [Online]. Available https://blockfint.com/blog/monorepo-for-mega-project

Harbor Authors. 2023. **What is Harbor?**. [Online]. Available https://goharbor.io/

RedHat. 2023. **What is YAML?**. [Online]. Available https://www.redhat.com/en/topics/automation/what-is-yaml

ภาคผนวก

คำรับรองเล่มสหกิจศึกษาโดยสถานประกอบการ

วันที่ ...7.. เดือน .....มิถุนายน..... พ.ศ. ...2566.....

ข้าพเจ้า...นายนันทวัฒน์ ไชยรัตน์... ตำแหน่ง .....Full Stack Leader.....

บริษัท .....Opsta (Thailand) Co.,Ltd..... ขอรับรองว่าทางสถานประกอบการได้ตรวจสอบเล่มสหกิจศึกษา เรื่อง.....การสร้างไปป์ไลน์เทมเพลตสำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI ..... ของนักศึกษาชื่อ .....นายมนัสวี นรบดีศุภกร..... ซึ่งเป็นนักศึกษาภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เรียบร้อยแล้ว และไม่มีส่วนหนึ่งส่วนใดในเล่มสหกิจศึกษานี้ที่มีข้อมูลอ่อนไหว และ/หรือ ข้อมูลอันเป็นความลับอัน จะก่อให้เกิดความเสียหายต่อสถานประกอบการ รวมทั้งอนุญาตให้สามารถเผยแพร่ต่อสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังได้ จึงลงชื่อไว้เป็นหลักฐาน

ลงชื่อ .....

(.....)

ตัวแทนสถานประกอบการ

ข้าพเจ้า อ.สันธนะ อู่อุดมยิ่ง อาจารย์ที่ปรึกษาสหกิจศึกษา ได้ตรวจสอบเล่มสหกิจศึกษาแล้วและรับทราบว่าสถานประกอบการดำเนินการตรวจสอบเล่มสหกิจศึกษาแล้ว จึงลงชื่อไว้เป็นหลักฐาน

ลงชื่อ .....

(.....)

อาจารย์ที่ปรึกษา

งานทะเบียนคณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
คำรับรองเล่มสหกิจศึกษา

วันที่ 21 เดือน มิถุนายน พ.ศ.2566

ข้าพเจ้า นาย มนัสวี นรบดีศุภกร รหัสประจำตัว 62050212 นักศึกษาหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชา วิทยาการคอมพิวเตอร์ ภาควิชา วิทยาการคอมพิวเตอร์ ขอรับรองว่า สหกิจศึกษา เรื่อง

ชื่อภาษาไทย การสร้างไปป์ไลน์เทมเพลต สำหรับพัฒนาซอฟต์แวร์แบบ CI/CD ด้วย Gitlab CI

ชื่อภาษาอังกฤษ Creating a template pipeline for CI/CD software development with Gitlab CI

ปีการศึกษา 2565

เป็นผลงานวิจัยที่มิได้คัดลอกหรือละเมิดลิขสิทธิ์ของผู้อื่นและได้ผ่านการตรวจสอบความซ้ำซ้อนเรียบร้อยแล้ว และได้แนบเอกสารการตรวจสอบการลอกเลียนงานวรรณกรรมที่ตรวจสอบจากเล่มสหกิจศึกษาฉบับสมบูรณ์แล้ว

โปรแกรมอักขราวิสุทธิ์ 0.56 %

ลงชื่อ.....
(นายมนัสวี นรบดีศุภกร)
นักศึกษา

ข้าพเจ้า อ.สันธนะ อู่อุดมยิ่ง อาจารย์ที่ปรึกษาสหกิจศึกษา ได้ตรวจสอบสหกิจศึกษาของนักศึกษาข้างต้นแล้ว ขอรับรองว่าเป็นผลงานวิจัยของนักศึกษาจริงและมีเนื้อหาสมบูรณ์ จึงลงชื่อไว้เป็นหลักฐาน

ลงชื่อ.....
อาจารย์ที่ปรึกษา