รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาระบบการควบคุมและจัดการข้อมูลโดยการใช้กระบวนการ DevOps
Data Control and Management by DevOps Processes

นายพิจักษณ์ จรรยาวิวัฒน์กุล

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา การพัฒนาระบบการควบคุมและจัดการข้อมูลโดยการใช้กระบวนการ DevOps
ชื่อ-สกุล นักศึกษา นายพิจักษณ์ จรรยาวิวัฒน์กุล **รหัสนักศึกษา** 59010956
คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์
ชื่อ-สกุล อาจารย์นิเทศ อาจารย์ เกียรติณรงค์ ทองประเสริฐ
ชื่อ-สกุล ผู้นิเทศงาน คุณ ณรงค์วิทย์ คลองน้อย
ชื่อสถานประกอบการ บริษัท มาร์โล กรุ๊ป (ประเทศไทย) จำกัด

บทคัดย่อ

ในปัจจุบันระบบการจัดเก็บข้อมูลและสารสนเทศเป็นข้อมูลที่มาจากทั้งภาครัฐและเอกชนที่มีความหลากหลายของการจัดเก็บโครงสร้างของข้อมูล ทำให้ยากต่อการเข้าถึงข้อมูลที่จัดเก็บไว้ ผู้พัฒนาระบบจึงเห็นว่าควรมีการจัดเก็บข้อมูลที่มีโครงสร้างและมาตรฐานเดียวกันเพื่อให้ง่ายและสามารถเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพและรวดเร็วมากยิ่งขึ้น จึงพัฒนาบริการ API สำหรับระบบการควบคุมและจัดการข้อมูลขึ้นมาด้วย Spring Boot Framework และใช้ CKAN ซึ่งเป็นระบบการจัดการข้อมูลที่ได้รับความนิยมในต่างประเทศ และมีการใช้ KeyCloak เข้ามาช่วยจัดการในเรื่องของสมาชิกในระบบและความปลอดภัยสำหรับ API Services ที่พัฒนาขึ้นจะทำงานอยู่บน OpenShift Container Platform ข้อมูลที่ทำงานบน Container ต่างๆจะถูกเก็บอยู่ใน GlusterFS การเข้าถึง API ทุกตัวจะต้องผ่าน 3Scale เพื่อให้การเข้าถึง API ให้เป็นไปตามข้อกำหนด และในการตรวจสอบระบบที่ทำงานอยู่บน OpenShift จะถูกนำออกมาเก็บไว้ที่ Elastic Search และใช้ Kibana ในการตรวจสอบการทำงาน ทำให้สามารถบริหารจัดการผู้ใช้งานและจัดเก็บข้อมูลให้อยู่ในโครงสร้างมาตรฐานเดียวกัน

คำสำคัญ : Data Control and Management; Microservices; CKAN; DevOps; OpenShift; 3Scale; Open Data; PostgreSQL; KeyCloak; OAuth; JWT; Jenkins; Pipeline; SonarQube; JFrog Artifactory; Harbor Image Registry; Docker; Selenium; Robot Framework; Apache JMeter; Kubernetes; CI/CD;

Co-operative Title: Data Control and Management by DevOps Processes
Student Intern Name: Mr. Phijak Chanyawiwatkul **Student ID:** 59010956
Faculty: Engineering **Department:** Computer Engineering
Advisor Name: Prof. Kiatnarong Tongprasert
Mentor Name: Mr. Narongwit Clongnoi
Company: Marlo Group (Thailand) Limited

ABSTRACT

At present, the data and information storage management systems are from both public and private sectors. Which has multiple storage structures. It makes difficult to access to the stored data. The system developers believe that data storage should have the same structure and standards for access to data faster and more efficient, therefore developing the API Services for data control and management by using spring boot framework and used CKAN system which is a data management system that is popular in foreign countries and uses KeyCloak to help to manage the members of the system and security. For developed API Services running on the OpenShift Container Platform, data running on various containers is stored in GlusterFS. All API will access via 3Scale, used to manage API access. And the system checks running OpenShift will be stored in Elastic Search and use Kibana as a GUI to check the operation. In finally The Data Control and Management can manage users and store data in the same standard structure.

Keywords: Data Control and Management; Microservices; CKAN; DevOps; OpenShift; 3Scale; Open Data; PostgreSQL; KeyCloak; OAuth; JWT; Jenkins; Pipeline; SonarQube; JFrog Artifactory; Harbor Image Registry; Docker; Selenium; Robot Framework; Apache JMeter; Kubernetes; CI/CD;

กิตติกรรมประกาศ

จากการที่ข้าพเจ้าได้มาปฏิบัติงานสหกิจศึกษา ณ บริษัท มาร์โล กรุ๊ป (ประเทศไทย) จำกัด ตั้งแต่วันที่ 5 เดือนสิงหาคม พ.ศ. 2562 ถึง วันที่ 29 เดือนพฤศจิกายน พ.ศ. 2562 ส่งผลให้ข้าพเจ้าได้รับความรู้และประสบการณ์ต่างๆ ที่มีค่ามากมาย สำหรับรายงานวิชาสหกิจศึกษาฉบับนี้ สำเร็จลงได้ด้วยดีจากความร่วมมือและสนับสนุนจาก คุณณรงค์วิทย์ คลองน้อย หัวหน้าแผนกปฏิบัติการ รวมถึงบุคคลท่านอื่นๆที่ไม่ได้กล่าวนามทุกท่านที่ได้ให้คำแนะนำช่วยเหลือในการจัดทำรายงาน

ข้าพเจ้าขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่าน ที่มีส่วนร่วมในการให้ข้อมูล เป็นที่ปรึกษาให้คำแนะนำในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความรู้ความเข้าใจเกี่ยวกับชีวิตของการทำงานจริง รวมถึงอาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ได้ให้ความรู้ตลอดมา ข้าพเจ้าขอขอบคุณ ไว้ ณ ที่นี้

สุดท้ายนี้ ขอขอบพระคุณบิดามารดาและครอบครัวที่คอยสนับสนุนและให้กำลังใจตลอดการดำเนินงานจนสำเร็จลุล่วงด้วยดี

พิจักษณ์ จรรยาวิวัฒน์กุล

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากในปัจจุบันเป็นยุคที่ถูกขับเคลื่อนด้วยข้อมูลและสารสนเทศ ซึ่งหน่วยงานต่างๆมีการนำเทคโนโลยีเข้ามาช่วยในการเก็บข้อมูลและประมวลผลข้อมูล ซึ่งข้อมูลนี้จะเป็นประโยชน์ในหลายๆด้านเช่น ด้านสังคมและสวัสดิการ คมนาคมและโลจิสติกส์ เศรษฐกิจ การเงินและอุตสาหกรรม เป็นต้น ทำให้สามารถนำไปใช้งานได้หลากหลายทั้งวางแผนงาน ทำคอนเทนต์ พยากรณ์หาแนวโน้ม นำข้อมูลมาใช้วิเคราะห์ประกอบการตัดสินใจ การจัดทำบริการออกมาให้ข้อมูลต่างๆ โดยจะเป็นข้อมูลที่อยู่ในรูปแบบมาตรฐานเดียวกันเพื่อที่จะนำมาใช้งานได้ง่ายและสะดวกยิ่งขึ้น แต่เนื่องจากขณะนี้การแบ่งปันข้อมูลถูกให้บริการและเผยแพร่เฉพาะภายในหน่วยงานของตนเท่านั้น ทำให้การเข้าถึงข้อมูลมีขั้นตอนที่มีความซับซ้อนก่อให้เกิดความล่าช้าในการใช้ข้อมูล รวมถึงมีรูปแบบการจัดเก็บและโครงสร้างที่แตกต่างกัน ทำให้ยากต่อการนำข้อมูลมาใช้ จากปัญหาดังกล่าวจึงมีความจำเป็นอย่างมากที่การเก็บข้อมูลจากหน่วยงานต่างๆต้องมีมาตรฐานในการจัดเก็บเพื่อความสะดวกในการเข้าถึงข้อมูลและสามารถนำข้อมูลไปประยุกต์ใช้ต่อได้สะดวกยิ่งขึ้น เพื่อเพิ่มประสิทธิภาพในการให้บริการ จึงพัฒนาระบบการควบคุมและจัดการข้อมูล เพื่อใช้บริหารจัดการข้อมูลและบริหารจัดการผู้ใช้งานในระบบขึ้นมา

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อสร้างบริการต่างๆ ในการนำเสนอข้อมูลขึ้นบนเว็บไซต์ที่เป็นมาตรฐานเดียวกัน

2) เพื่อสร้างบริการสำหรับจัดการผู้ใช้งานและการจัดสิทธิ์ต่างๆ

3) เพื่อให้สามารถควบคุมและจัดการข้อมูลได้อย่างมีประสิทธิภาพ และมีความสามารถในการรับภาระงานที่มากขึ้น

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ข้อมูลถูกจัดเก็บอยู่ในโครงสร้างที่เป็นมาตรฐานเดียวกัน

2) สามารถบริหารจัดการผู้ใช้งานในระบบได้อย่างมีประสิทธิภาพ

3) เพิ่มช่องทางในการค้นหาและเข้าถึงข้อมูลต่างๆในระบบได้สะดวกมากยิ่งขึ้น

1.4 ข้อจำกัดของโครงงาน

1) การเข้าใช้งานระบบจำเป็นต้องผ่านการรับรองจากผู้ดูแลระบบก่อน

2) การเข้าใช้งานข้อมูลของบุคคลสามารถเข้าถึงในระดับต่างกัน

1.5 ขอบเขตของโครงงาน

1) วิเคราะห์ระบบและข้อมูล (System Analysis and Data Analysis) และจัดทำ (System Requirements Specification) รวมทั้งแผนงานตามที่ศึกษาโครงการเพื่อพัฒนาความเหมาะสมในการออกแบบพัฒนาระบบควบคุมและจัดการข้อมูล

2) ออกแบบสถาปัตยกรรมของระบบ (System Architecture) การออกแบบในรายละเอียด (Detail Design) และกรณีทดสอบ (Test Case) โดยการออกแบบทั้งหมดจะต้องครอบคุลมความต้องการตามรายละเอียดที่ระบุไว้ในบทที่ 3

3) พัฒนาระบบควบคุมและจัดการข้อมูลตามที่ออกแบบไว้

3) ทดสอบการทำงานของระบบตามที่ออกแบบไว้

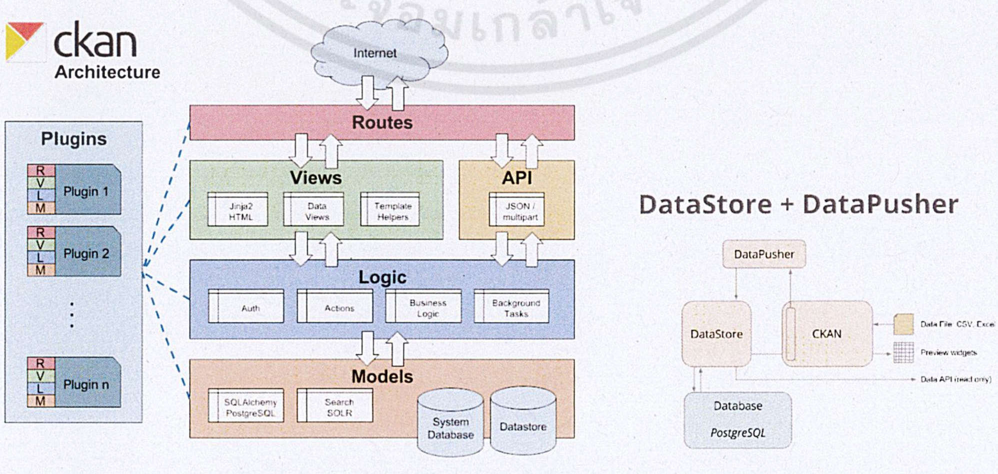
บทที่ 2
เทคโนโลยีที่เกี่ยวข้อง

2.1 CKAN

ภาพที่ 2.1 CKAN Logo

CKAN เป็นระบบ Data Portal Platform สำหรับการเก็บและเผยแพร่ข้อมูลแบบ Open Data ซึ่งมีคุณสมบัติพื้นฐาน เช่น การค้นหาข้อมูล การใช้ข้อมูลผ่าน API การแชร์ข้อมูล ตลอดจนยังสามารถเพิ่มคุณสมบัติใหม่ๆ ได้ผ่าน extension ซึ่งมี extension เหมาะกับโปรเจคที่เกี่ยวกับด้าน Open Data สามารถใช้ CKAN ทำ portal ได้สะดวกและรวดเร็ว สำหรับข้อมูลที่สามารถเผยแพร่บน CKAN ได้มีหลายรูปแบบ ทั้ง CSV, JSON, XML หรือแม้ระทั่ง URL ธรรมดา ทั้งนี้ขึ้นอยู่กับการประยุกต์ใช้

ภาพที่ 2.2 CKAN Feature



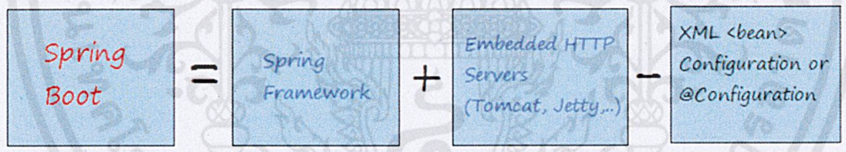
ภาพที่ 2.3 CKAN Architecture และ Extension

2.2 Spring Boot

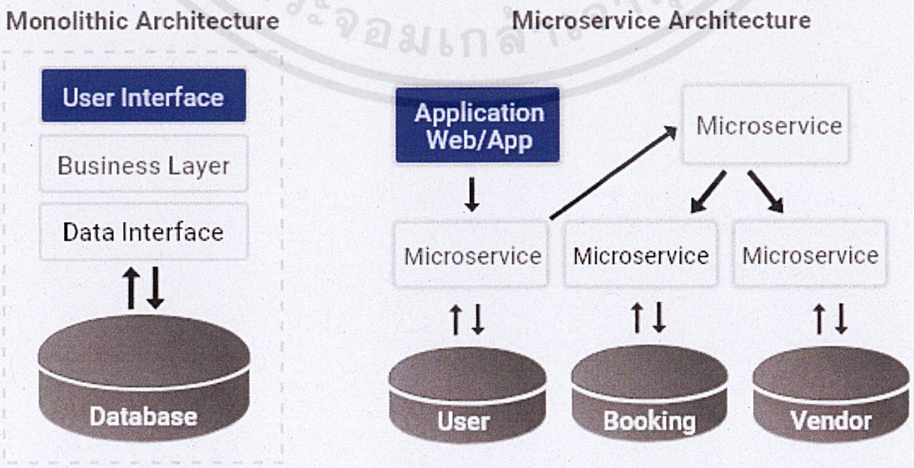
ภาพที่ 2.4 Spring Boot Logo

Spring Boot เป็น Project ของ Spring Framework ที่ช่วยทำให้พัฒนา Application ได้รวดเร็วขึ้นโดยมีการ Config ต่างๆให้อัตโนมัติ และมีการฝังตัว Tomcat Server ไว้ทำให้พร้อมทำงานด้วยตัวเอง

ภาพที่ 2.5 รายละเอียดของ Java EE, Spring Framework และ Spring Boot

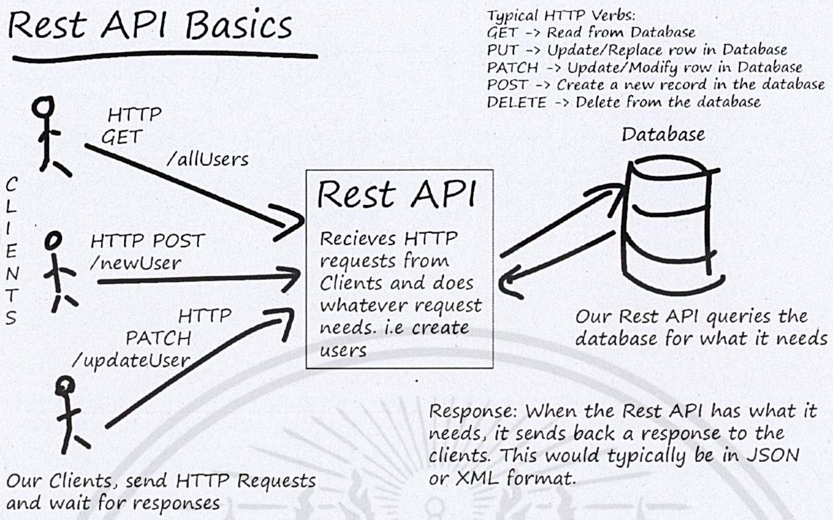


ภาพที่ 2.6 รายละเอียดของ Spring Boot



ภาพที่ 2.7 สถาปัตยกรรม Monolithic และ Microservices

2.3 Rest API และ Restful API



ภาพที่ 2.8 Rest API Basics

REST มาจาก Representational state transfer ส่วน API มาจาก Application Program Interface โดยที่ REST เป็นรูปแบบของสถาปัตยกรรมซอฟต์แวร์ ที่ Roy Fielding Ph.d ได้อธิบายไว้ในวิทยานิพนธ์ ตั้งแต่ปี ค.ศ. 2000, เป็น "รูปแบบสถาปัตยกรรม" ที่ใช้ประโยชน์จากเทคโนโลยี Web protocol เพื่อใช้ในการสร้าง Web Service

Architectural properties ข้อกำหนดของ REST architectural style ทั้ง 7 ประกอบไปด้วย

1) Performance: ประสิทธิภาพในการโต้ตอบ

2) Scalability: มีความสามารถในการปรับขยายได้ทำให้สามารถรองรับส่วนประกอบจำนวนมาก

3) Simplicity: ความเรียบง่ายของอินเตอร์เฟซที่เหมือนกัน

4) Modifiability: ความสามารถในการปรับเปลี่ยนส่วนประกอบเพื่อตอบสนองความต้องการที่เปลี่ยนแปลง (แม้ในขณะที่แอพพลิเคชั่นทำงานอยู่

5) Visibility: การมองเห็นการสื่อสารระหว่าง Component ผ่าน Service agents

6) Portability: ง่ายต่อการโยกย้ายในส่วนของ Program และ Data

7) Reliability: ความน่าเชื่อถือในการต้านทานความล้มเหลวในระบบ

RESTful Web Services (RWS) ที่ใช้ REST architectural style เป็นที่รู้จักกันในชื่อ RESTful web services (RWS). RESTful Web service อนุญาตให้ระบบ Request และเข้าถึง Resource บนเว็บโดยใช้ชุดคำสั่งที่กำหนดเอาไว้ล่วงหน้า โดยที่การโต้ตอบของระบบที่ใช้ REST จะอยู่บนพื้นฐานของ Hypertext Transfer Protocol (HTTP). Request จะส่งคำขอไปยัง URI ที่กำหนด และเอา response กลับมาเป็น Payload ในแบบ HTML, XML, JSON หรือ format อื่น ๆ

RESTful ประกอบไปด้วย Client ผู้ที่จะเข้ามา Request resources และ Server ผู้เป็นผู้ให้บริการ Resources

Architectural constraints of RESTful API มี 6 ข้อกำหนดของ RESTful API ซึ่งถือเป็นสิ่งที่สำคัญในการสร้าง RESTful API ตามมาตรฐานซึ่งทำให้ง่ายต่อการพัฒนา และทำให้เป็นที่ยอมรับ

1) Client-server architecture: Client ไม่จำเป็นต้องรู้อะไรเกี่ยวกับ Business logic ภายใน ไม่มีหน้าที่เกี่ยวกับการจัดเก็บข้อมูล ส่วน Server มีหน้าที่เก็บ Resource และไม่จำเป็นต้องรู้อะไรเกี่ยวกับ UI Frontend หรือสถานะของผู้เรียก

2) Statelessness: ส่ง Request รับ Response จาก Server แล้วเลิก

3) Cache ability: สามารถ cache response ได้ การ Response จะต้องสามารถกำหนดได้ว่าจะ Cache หรือไม่ เพื่อป้องกันไม่ให้ User หรือ Client ได้รับข้อมูลเก่า

4) Layered system: ปกติ Client ไม่รู้ว่าที่ทำการเชื่อมต่อนั้น ได้เชื่อมต่อโดยตรงกับ Server ปลายทาง หรือไปยังตัวกลางอื่น ๆ ระหว่างทาง, Server ตัวกลางควรสามารถปรับปรุงความสามารถในการขยายระบบได้ โดยการใช้งานการทำ Load balance

5) Code on demand (optional): Server สามารถขยายได้ชั่วคราว หรือปรับแต่งการทำงานของ Client ได้ ตัวอย่างเช่น ทำ client-side scripts ใน JavaScript

6) Uniform interface: ถือเป็นข้อสำคัญจะที่แยกระหว่าง REST API และ Non-REST API มันแสดงให้เห็นถึงวิธีการที่จะคุยกับ Server โดยไม่คำนึงถึงประเภทของอุปกรณ์ หรือประเภทของ application Uniform interface ได้แยกออกไปอีก 4 อย่าง

6.1) Resource-Based: เช่น API/users

6.2) Manipulation of Resources Through Representations: เช่นเมื่อ User get user_id หรือ Request list of users แล้วทำการ Delete หรือ Modify user

6.3) Self-descriptive Messages: แต่ละ Message มีข้อมูลเพียงพอที่จะนำมาอธิบายวิธีการ Process message เพื่อให้ Server ทำการวิเคราะห์ได้ง่าย

6.4) Hypermedia as the Engine of Application State (HATEOAS): จำเป็นต้องมี Links สำหรับทุก ๆ Response เพื่อให้ Client สามารถค้นหาได้ง่าย

Standard operations: GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE

สรุปได้ว่า REST เป็นรูปแบบของสถาปัตยกรรมซอฟต์แวร์ ที่ใช้ประโยชน์จากเทคโนโลยี Web protocol เพื่อใช้ในการสร้าง Web Service ขณะที่ RESTful มักใช้เพื่ออ้างถึง Web Service ที่ใช้สถาปัตยกรรม REST

REST เป็นกระบวนทางสถาปัตยกรรม หรือ architectural paradigm.

RESTful เป็นการใช้ หรืออ้างถึงสถาปัตยกรรม หรือ using that paradigm.

2.4 Kubernetes

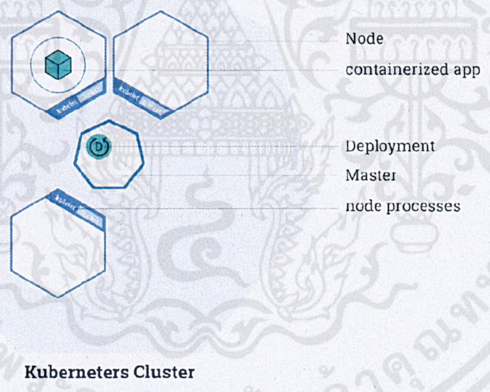
ภาพที่ 2.9 Kubernetes Logo

Kubernetes หรือ k8s เป็นแพลตฟอร์มแบบ Open-Source สำหรับช่วยให้การปฏิบัติงานต่างๆ ที่เกี่ยวข้องกับ Linux Container สามารถทำได้โดยอัตโนมัติ ลดกระบวนการติดตั้งหรือขยายแอพพลิเคชั่นที่รันบน Container หรือกล่าวได้ว่า ช่วยให้นักพัฒนาสามารถทำ Cluster กลุ่มของโฮสต์ที่รัน Linux Container ได้ และ Kubernetes ก็เข้ามาช่วยบริหารจัดการ Cluster เหล่านั้นสำหรับใช้งานบน Public, Private และ Hybrid Cloud ได้อย่างสะดวกรวดเร็วและมีประสิทธิภาพ

Kubernetes ถูกพัฒนาและออกแบบโดยวิศวกรจาก Google ซึ่งเป็นนักพัฒนาทีมแรกๆ ที่เริ่มนำเทคโนโลยี Linux Container มาใช้ ก่อนหน้านี้ Google ใช้แพลตฟอร์ม Borg ในการบริหารจัดการกว่า 2,000 ล้าน Container ที่ถูกสร้างขึ้นในแต่ละสัปดาห์ ก่อนที่จะนำประสบการณ์ที่ได้มาพัฒนาต่อยอดเป็น Kubernetes เพื่อบริหารจัดการระบบ Cloud ทั้งหมดในปัจจุบัน

สำหรับบริษัทที่นำเทคโนโลยี Container มาใช้งานแล้ว จะพบว่าแอพพลิเคชั่นบน Production มักมีการเชื่อมต่อหลายๆ Containers เข้าด้วยกัน ซึ่ง Containers เหล่านั้นมักถูกติดตั้งบนหลายๆ โฮสต์

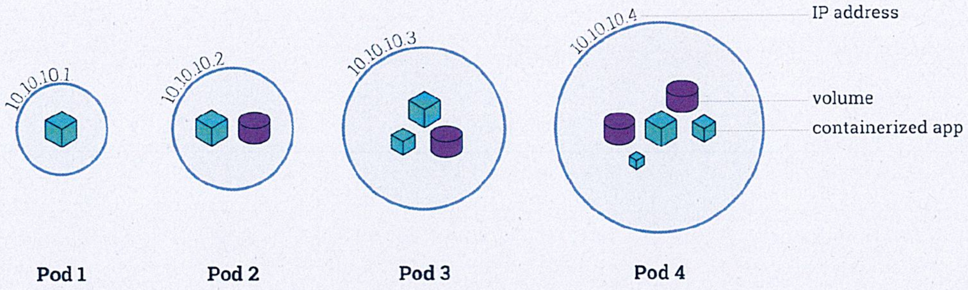
Kubernetes จะเข้ามาช่วยประสานการทำงาน (Orchestration) และบริหารจัดการ Containers ทั้งการติดตั้งและการขยายระบบในอนาคต ส่งผลให้นักพัฒนาสามารถสร้างแอพพลิเคชั่นที่เชื่อมต่อหลายๆ Containers เข้าด้วยกัน กำหนดการทำงานต่างๆ สามารถขยายระบบ และตรวจสอบการทำงานทั้งหมดได้อย่างง่าย ที่สำคัญคือสามารถดำเนินคำสั่งต่างๆ ได้โดยอัตโนมัติ



ภาพที่ 2.10 Kubernetes Cluster

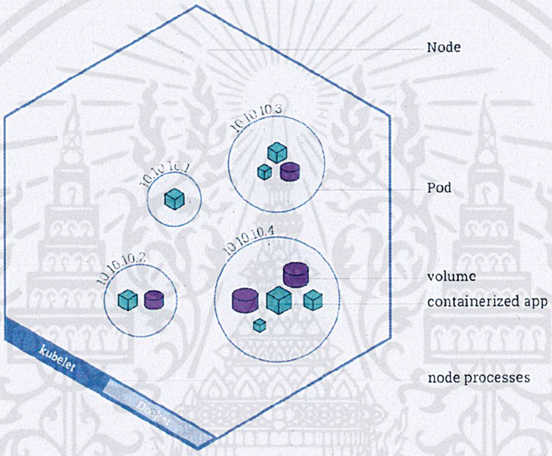
Cluster คือกลุ่มของ server ที่เอาไว้ทำงาน container คือการนำ k8s มาครอบ cluster เอาไว้เสมือนเป็น server ใหญ่ๆ เครื่องเดียว โดยที่ตัว k8s จะจัดการให้เองว่าอะไรทำงานอยู่ที่ไหน ใน cluster จะประกอบด้วย master และ node ปกติ โดย master มีหน้าที่จัดการ node ว่าต้องทำอะไรบ้าง เวลา deploy จะสั่งงานผ่าน master และแต่ละ node มี process ที่ชื่อ kubelet ไว้คุยกับ master และมี docker ในตัวเพื่อรัน container

Deployment เป็น configuration ที่ไว้สร้างหรืออัพเดท app และบอกว่า app จะทำงานอย่างไร (การกำหนด desired state ของ app) โดยที่ k8s จะพยายามดูแลให้เป็นไปตาม config ตลอดเวลา (self-healing) เมื่อสร้าง deployment เพื่อกำหนด config แล้ว k8s จะทำการสร้างและเอาแอพพลิเคชั่นไปทำงานอยู่ในสิ่งที่เรียกว่า Pod



ภาพที่ 2.11 Kubernetes Pod

Pod เป็นหน่วยที่เล็กที่สุดในบรรดา k8s object ซึ่งแทนกลุ่มของ container กับ resource ต่างๆที่จำเป็นสำหรับ container นั้นๆ เช่น Storage, Network ip และยังเก็บ config ต่างๆเกี่ยวกับ container



ภาพที่ 2.12 Kubernetes Kubelet

เมื่อส่ง Deployment ที่มี config ของ pod เข้าไปแล้ว ตัว master ของ k8s ก็จะไปจัดการเอง ว่า pod จะต้องไปอยู่ที่ node อะไร ตัว pod เองก็จะมี Lifecycle โดย Deployment จะมีบริการตัวนึง ที่มาดูแล pod ให้มีจำนวนเท่าที่กำหนดไว้

Replica Set คือ ตอนที่ส่ง deployment เข้าไปใน k8s นอกจากจะสร้าง pod แล้วยังสร้าง Replica Set ขึ้นมาด้วย หน้าที่หลักของ Replica Set คือทำให้ pod มีจำนวนเท่ากับที่บอกไว้ตลอด ตัวอย่างเช่น ต้องการ pod my-app ทั้งหมด 4 replica ถ้ามีเกิน ตัว Replica Set จะทำการลบทิ้ง ถ้ามีน้อยกว่าก็จะสร้างเพิ่มให้ (self-healing) นอกจากจะบอกจำนวนตัว Replica Set ยังทำ Auto Scale ได้ เช่น ถ้า CPU Usage เกิน 80% ให้สร้าง Pod เพิ่มอีก (Horizontal Pod Auto Scalers หรือ HPA)

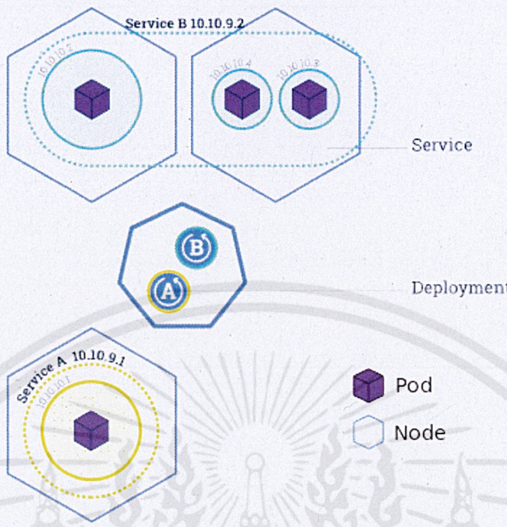
Deployment = Replica Set + Pod

1 Deployment จะมี 1 Replica Set และมี pod ได้ 1 แบบ แต่มีหลายๆ pod ได้จากการ scale

1 Microservices = 1 Deployment

โดยปกติแล้ว Pod จะมี Life cycle ของตนเอง คือ หาก node หยุดทำงาน Replication Controller จะทำการเริ่มทำงาน pod เหล่านั้นขึ้นมาใหม่ โดยจะพยายามทำตาม desired state ที่กำหนดไว้โดยไม่ต้องสนใจว่า pod เหล่านั้นจะอยู่ที่ node ไหน

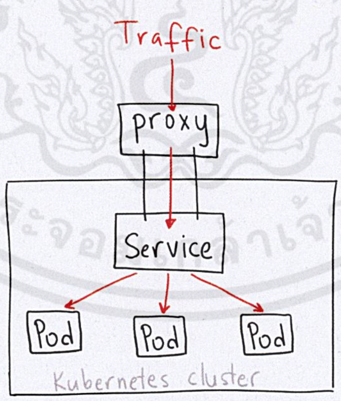
Service เป็นสิ่งที่พยายาม abstract ทุกอย่างเพื่ออนุญาตให้ pod สามารถหยุดทำงานได้โดยไม่ส่งผลกระทบกับ Front-end และหน้าที่ที่สำคัญอีกอย่างคือทำการ routing เส้นทางไปยัง pod ให้ถูกต้อง



ภาพที่ 2.13 Kubernetes Service

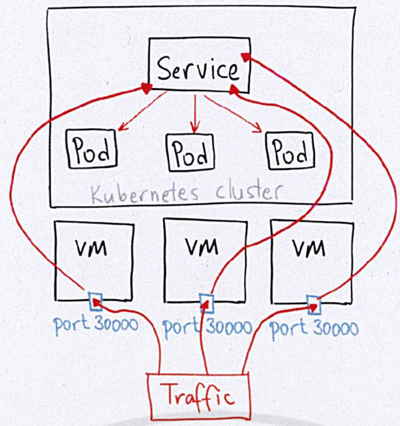
แม้ว่าแต่ละ Pod จะมี Unique IP แต่ IP เหล่านั้นจะใช้กันอยู่แค่ภายใน ซึ่ง Service จะทำหน้าที่ Group หลายๆ endpoint ของแต่ละ Pod ให้มีแค่ endpoint เดียว

Kubernetes มี Service type ให้เลือก 4 ประเภท โดยจะอธิบายถึงความแตกต่างของแต่ละประเภทได้แก่



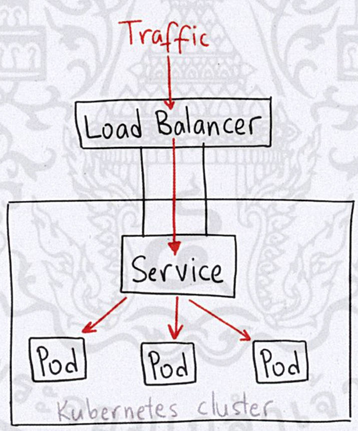
ภาพที่ 2.14 Kubernetes Cluster IP

1) Cluster IP คือต้องการ expose Service เพื่อใช้กันภายใน ไม่สามารถเข้าถึงได้จากภายนอก เช่น ต้องการสร้าง Service ฐานข้อมูล โดย Client ไม่สามารถเรียกฐานข้อมูลโดยตรง จะต้องเรียกผ่าน back-end ก่อน ซึ่ง back-end จะติดต่อกับฐานข้อมูลผ่านทาง Cluster IP ได้ แต่ต้องอยู่ใน network เดียวกัน



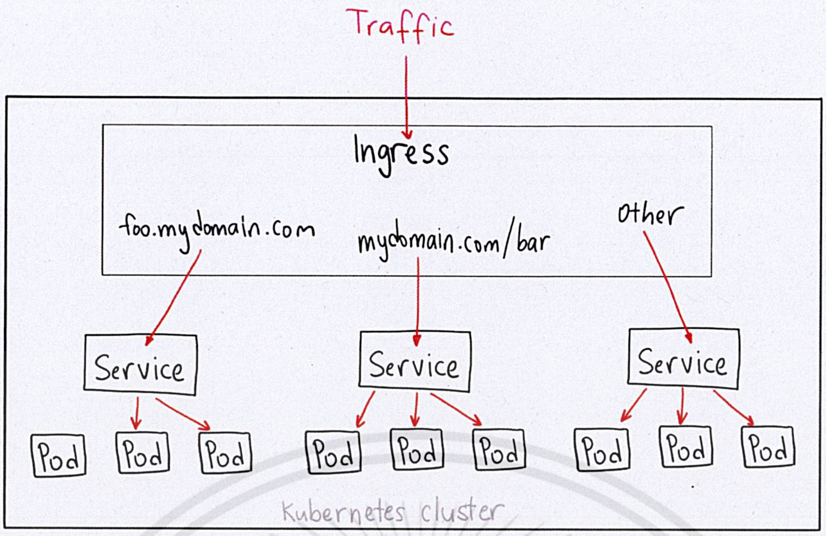
ภาพที่ 2.15 Kubernetes Node Port

2) Node Port คือการ expose service ออกไปสู่โลกภายนอกโดยใช้ public IP ของ Node ที่ทำงาน Cluster นั้นอยู่ แต่จะมีการกำหนด TCP port แบบ random ระหว่าง 30000–32767 (สามารถกำหนดค่าเองได้ แต่ทุก node ต้องมี port ที่ใช้งานได้และต้องอยู่ในช่วง 30000–32767) หากเราทำการ re-deploy service นี้ หมายเลข port อาจถูกเปลี่ยนแปลงได้ ข้อดี คือ สามารถ expose ให้ client ได้เลย ข้อเสีย คือ ถ้า IP ของ Node เปลี่ยน client ก็จะเรียก service ไม่ได้



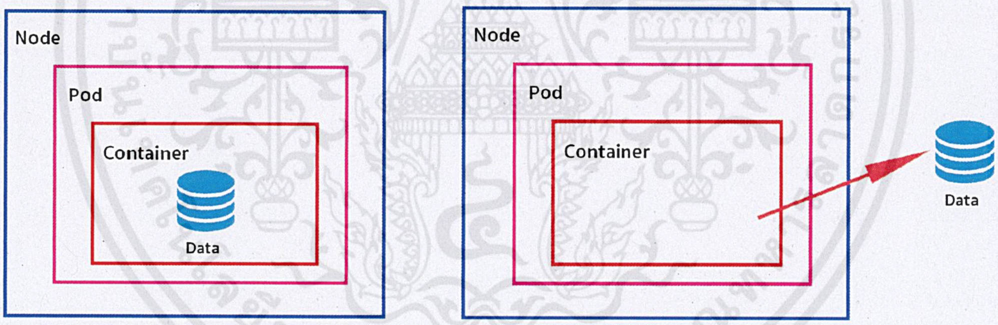
ภาพที่ 2.16 Kubernetes Load Balancer

3) Load Balancer โดย Services ที่กล่าวมาข้างต้นนั้นมีการทำ Load Balancing อยู่แล้ว แต่เป็น Internal Load Balancer หมายความว่าทุกๆ Pod ที่ label ตรงกับ selector ของ Service จะได้รับ load เท่าๆ กัน ซึ่งตัว Service จะเป็นตัวกระจายงาน แต่ถ้าเป็น Type ของ Service ที่ชื่อว่า Load Balancer จะกล่าวถึง External Load Balancer หมายความว่า Load Balancer แต่ละอันจะได้รับการ assign public IP ทำให้ client เข้าถึงได้โดยจะเป็นลักษณะของ Fixed IP ซึ่งเหมาะสำหรับการ expose Service เพื่อให้ client เรียกใช้ได้บนระบบ production ข้อดี คือ สามารถกำหนดเป็น Static IP ได้ รองรับ Protocol ทุกรูปแบบ เช่น HTTP, TCP, UDP, Websockets, gRPC ข้อเสีย คือ Cloud Provider ต้อง Support Layer 4 Load Balancer (e.g. GCP, AWS, Azure) ต้องจ่ายค่า Public IP (ส่วนใหญ่จะให้บริการฟรี



ภาพที่ 2.17 Kubernetes Ingress

4) Ingress คือ ไฟล์ nginx.conf ซึ่งเป็นการทำ Virtual Host (1 IP มีหลาย Domain) จัดการเรื่อง Certificate สำหรับ HTTPS และการทำ Reverse proxy เป็น set of rules ที่ส่งต่อไปให้ controller ซึ่ง Load Balancer ก็พร้อมจะรับ Ingress rules เหล่านั้น



ภาพที่ 2.18 Kubernetes Basic Structure

Volume บน Kubernetes ในส่วนของ Pod สามารถเกิดขึ้นและหยุดทำงานไปตาม life cycle ถ้าใน Container มี data ของ application ที่ต้องใช้ หาก Pod หยุดทำงานไป ข้อมูลก็จะหายไป จึงต้องทำการย้าย data ใน Container ออกมาจาก Pod ซึ่งวิธีนี้จะทำให้ data ไม่ขึ้นอยู่กับ Pod ใดๆ ใน Cluster ซึ่งเรียกว่า Persistent Volume หรือ PV

2.5 3Scale

ภาพที่ 2.19 3Scale Logo

Red Hat 3Scale เป็นโซลูชันหลักทางด้านระบบ API Management ที่มีความสามารถหลากหลายตอบโจทย์ความต้องการทางธุรกิจได้เป็นอย่างดี

1) การทำ Access Control และ Rate Limit ให้กับ API เพื่อเสริมความมั่นคงปลอดภัยและความทนทานให้กับบริการ API สำคัญ

2) เสริมระบบ Security ในการเรียกใช้งาน API เช่น การรองรับ Authentication Pattern และ Credential ที่หลากหลาย

3) มีระบบ API Lifecycle ที่ครบวงจร ตั้งแต่การวางแผน, การออกแบบ, การติดตั้งใช้งาน, การเปิดให้บริการ, การควบคุมการใช้งาน, การตรวจสอบการทำงาน, การวิเคราะห์ข้อมูล, การปรับแต่งประสิทธิภาพการทำงาน และการยกเลิกบริการ API

4) มี Developer Portal เพื่อให้พันธมิตรหรือลูกค้าสามารถศึกษาข้อมูลและวิธีการเรียกใช้งาน API ได้ด้วยตนเอง เพื่อให้เกิดการเร่งใช้งาน API ได้อย่างมีประสิทธิภาพมากขึ้นในอนาคต

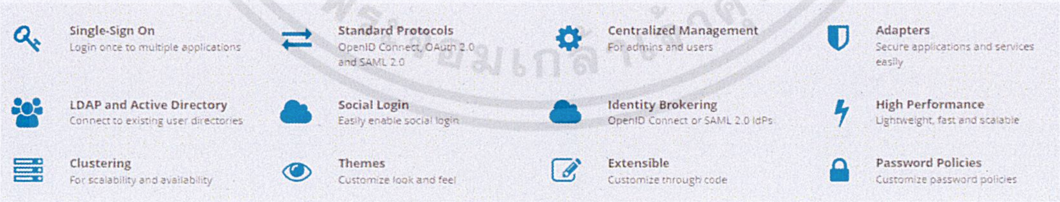
5) มีระบบวิเคราะห์ข้อมูลการใช้งาน API พร้อมระบบแจ้งเตือนเมื่อเกิดเหตุการณ์ผิดปกติ ง่ายต่อการดูแลรักษาและแก้ไขปัญหาต่างๆ ที่เกิดขึ้น

6) มีเครื่องมือในการคิดค่าใช้จ่ายและสร้างรายได้ผ่านทาง API ให้พร้อมใช้งาน ตั้งแต่การตั้งราคา, การออกใบเสร็จ, การเก็บเงิน และการผสานระบบเข้ากับ Payment Gateway ที่ต้องการ

2.6 KeyCloak

ภาพที่ 2.20 KeyCloak Logo

เป็นโปรแกรม Open Source ที่ใช้สำหรับบริหารจัดการการเข้าถึงสำหรับแอพพลิเคชั่นและบริการที่ทันสมัย รวมถึงการเพิ่มการพิสูจน์ตัวจนไปยังแอพพลิเคชั่นและบริการ ช่วยเพิ่มความปลอดภัยการเข้าถึงข้อมูล การจัดการผู้ใช้งาน

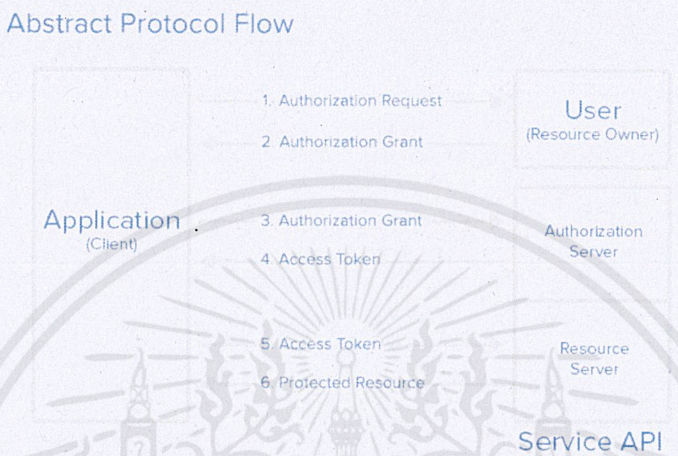


ภาพที่ 2.21 KeyCloak Features

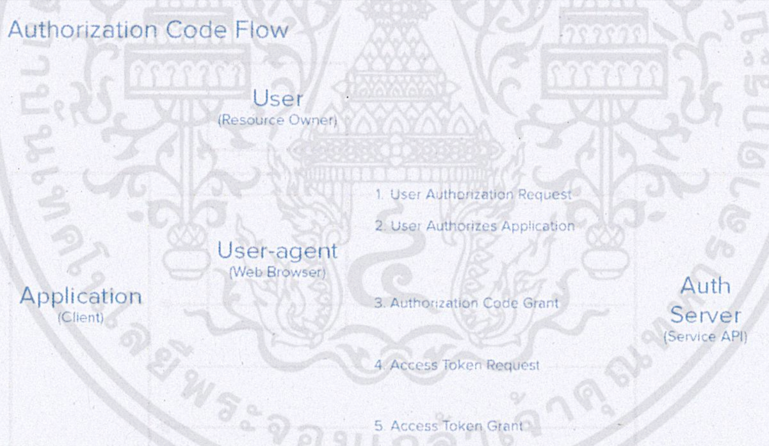
2.7 OAuth 2.0

ภาพที่ 2.22 OAuth 2.0 Logo

OAuth2 คือมาตรฐานหนึ่งของระบบยืนยันตัวตน และจัดการสิทธิ์การเข้าใช้งานระบบต่าง ๆ เป็นมาตรฐาน rfc6747 ที่ใช้สำหรับ Client เชื่อมต่อกับ Server ที่ใช้ในการ Authentication & Authorization เพื่อให้ได้รับสิ่งที่เรียกว่า Access Token เพื่อใช้แทน Username และ Password (สามารถใช้อย่างอื่นเพื่อขอ Token ก็ได้ เพื่อนำไปใช้กับบริการอื่นๆ ทำให้มีความปลอดภัยมากขึ้น รวมถึงบอกว่าทำมีสิทธิ์ทำอะไรได้บ้างกับบริการนั้น ๆ แนะนำต้องใช้คู่กับ https อีกชั้นเพื่อความปลอดภัยสูงสุด



ภาพที่ 2.23 OAuth 2.0 Abstract Code Flow

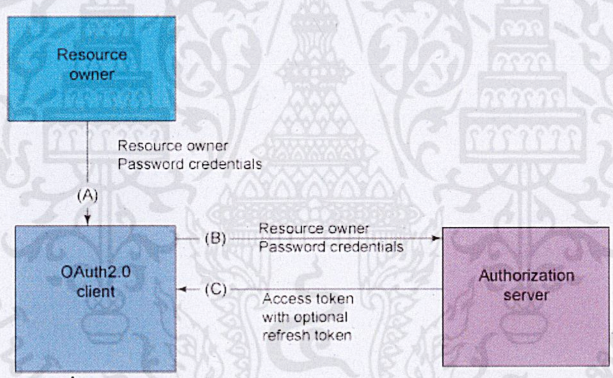


ภาพที่ 2.24 OAuth 2.0 Authorization Code Flow

1) Authorization Code Flow ใช้สำหรับ Web Server ที่ใช้ Code ด้านหลังในการเชื่อมต่อกับ OAuth Server โดยไม่ได้เปิดเผยให้สาธารณะเห็น อธิบายเป็นลักษณะการใช้งานคือ ผู้ใช้งานเข้า Web Site จะมีให้กด Login Facebook, Twitter, Google หรืออื่นๆ เมื่อผู้ใช้กดก็จะขึ้นให้ไป Login ที่ผู้ให้บริการนั้น ๆ ถ้าเคย Login ไว้แล้วก็จะข้ามขั้นตอนนี้ไป ถ้าผู้ให้บริการนั้นๆ เช่น Facebook จะให้กดยอมรับข้ออนุญาต ส่วนมากจะถามเรื่องสิทธิ์ในการเข้าถึงข้อมูลส่วนตัว เมื่อผู้ใช้กดอนุญาต ก็จะกลับมายัง Web Site โดยในเบื้องหลัง Web Site จะได้ authorization code มาเรียบร้อยแล้วจากผู้ให้บริการ จากนั้นทาง Web Site ก็สามารถเข้าถึงข้อมูลของผู้ให้บริการนั้นๆ ได้ตามสิทธิที่อนุญาตไว้

ภาพที่ 2.25 OAuth 2.0 Implicit Code Flow

2) Implicit Code Flow ใช้สำหรับ App ฝั่ง Client ซึ่งไม่จำเป็นต้องมี Web Server เป็นเหมือนการคุยระหว่าง Web Browser Client กับ OAuth Server ตรง ๆ เหมาะกับพวกที่ลงท้ายด้วย JS เช่น ReactJS, AngularJS ที่ต้องการดึงข้อมูลด้วย Browser เลย (เหมาะกับ Mobile เป็นพิเศษ ลักษณะการทำงานคล้าย ๆ กับข้อ 1 แต่จะต่างกันตรงไม่ต้องส่ง Client_Secret เป็นวิธีที่เปิดเผยให้สาธารณะเห็น



ภาพที่ 2.26 OAuth 2.0 Password Credentials Flow

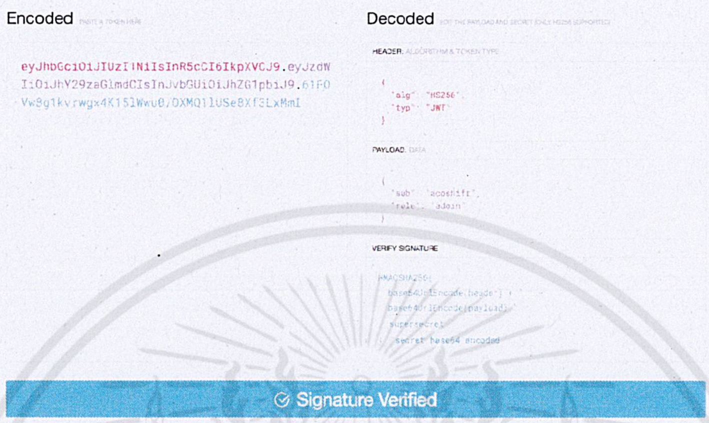
3) Password Credentials Flow ใช้สำหรับ Application ที่มีการจัดการสิทธิเอง แต่ต้องการยืนยันตัวตนเท่านั้น ซึ่งวิธีนี้ไม่ต้อง Redirect ไปที่ผู้ให้บริการอื่น วิธีนี้เหมาะกับการใช้งานที่เป็นบริการของตัวเอง เพราะ username password จะปรากฏในเครื่องที่ส่งขอ token ถ้าไปรันวิธีนี้บน Server อื่นที่ไม่ได้เป็นเจ้าของ แสดงว่า เขาอาจจะดักเอา username password ไปใช้ก็ได้

4) Client Credentials ในกรณีที่เป็นการคุยระหว่าง Application -> Service โดยจะไม่เกี่ยวข้องกับผู้ใช้ ยกตัวอย่างว่าเราอาจจะได้ข้อมูลสักอย่างแต่ต้องการความปลอดภัยว่าต้องเป็นเครื่องที่เราให้สิทธิ์ ก็สามารถส่ง id และ secret ที่ออกให้ Application ส่งมาขอ token เพื่อเข้าถึง Service นั้น ๆ ได้

2.8 JWT

ภาพที่ 2.27 JWT Logo

JSON Web Token เป็น standard ตัวหนึ่ง ใช้เพื่อส่ง claims ที่เป็น JSON ระหว่างระบบ ใน environment ที่ต้องการ URL-Safe เช่น HTTP Header หรือ Query Param ประกอบด้วย 3 ส่วน คือ Header ประกอบด้วยประเภทของ Token และ Algorithm ที่ใช้, Payload คือส่วน claims ที่เป็น JSON, Signature คือลายเซ็นต์เพื่อบอกว่า header และ payload ถูกต้อง



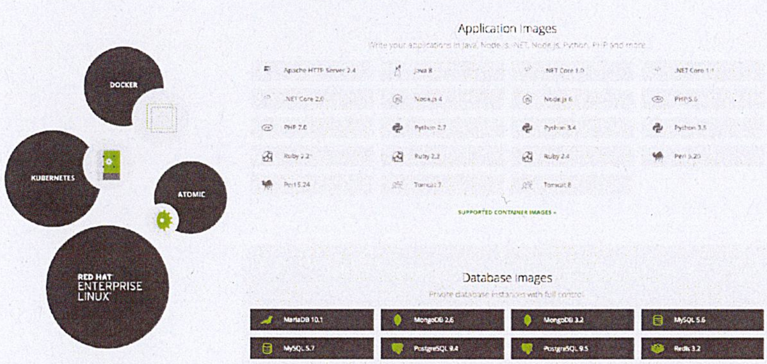
ภาพที่ 2.28 JWT Example

2.9 OpenShift

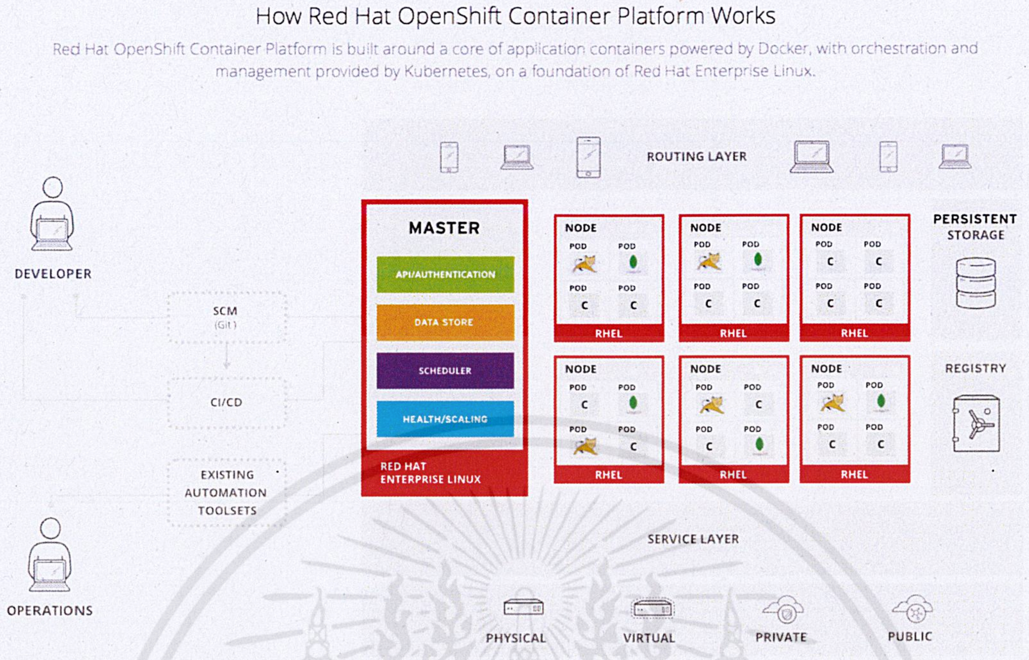
ภาพที่ 2.29 OpenShift Logo

OpenShift เป็นแพลตฟอร์มแอพพลิเคชั่นคอนเทนเนอร์ที่เอา Docker และ Kubernetes มาผนวกเข้ากันและเพิ่มความสามารถต่างๆให้สามารถใช้งานได้ง่ายและรวดเร็วยิ่งขึ้น

ภาพที่ 2.30 OpenShift Feature



ภาพที่ 2.31 OpenShift Image



ภาพที่ 2.32 OpenShift Architecture

2.10 GlusterFS

RED HAT GLUSTER STORAGE

ภาพที่ 2.33 GlusterFS (Red Hat Gluster Storage) Logo

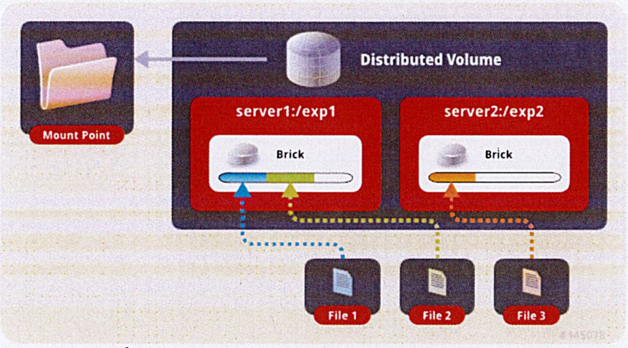
GlusterFS เป็นระบบไฟล์แบบกระจายขยายตัว (scale out) ข้ามเครือข่ายได้ง่าย ระบบไฟล์ตัวนี้พัฒนาขึ้นโดยบริษัท Gluster ที่มีจุดเด่นที่ความทนทานต่อความผิดพลาดของระบบ (high availability/fault tolerant) ชื่ออย่างเป็นทางการในปัจจุบันคือ Red Hat Gluster Storage ทำหน้าที่แชร์พื้นที่ disk ในรูปของ Block device คือ เครื่อง client ที่เรียกใช้จะมองเป็น disk ลูกหนึ่ง สามารถรวม Storage จากหลาย ๆ เครื่อง และทำการกระจายข้อมูลเก็บโดยมีระบบคล้าย ๆ RAID ที่ช่วยป้องกันเวลาเครื่องใดเครื่องหนึ่งพัง โดย Access ได้หลาย Protocol แล้วแต่ว่าการทำไปใช้งานกับอะไร และ XFS คือ filesystem รูปแบบหนึ่ง คล้ายกับพวก NFS FAT32 ซึ่งสามารถจัดการไฟล์ได้รวดเร็วมาก

สำหรับ Gluster FS มีโครงสร้างระบบไฟล์ 2 ส่วน

1) Brick คือ Directory ที่อยู่บนแต่ละเครื่อง โดยจะสร้าง Partition เพิ่มจาก Partition ปกติ (จะเป็น Disk ใหม่ Disk เดิมก็ได้ หรือจะใช้เป็น Directory ที่อยู่บน Partition เดิมก็ได้ โดย recommend เวลาสร้าง partition ให้เป็น xfs

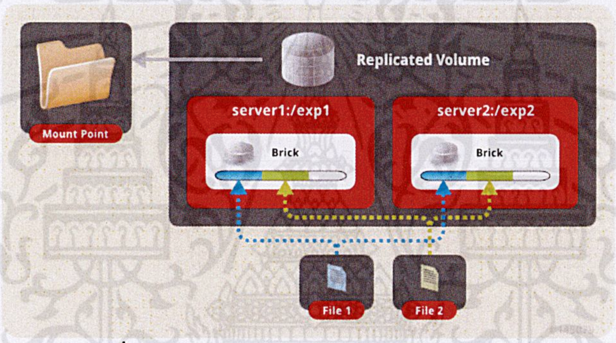
2) Volume คือชุดของข้อมูลที่ให้บริการ ๆ คือ การเอา Brick จากแต่ละเครื่องมาให้บริการภายใต้ Volume นั้น ๆ (แนวคิดเดียวกับ RAID ที่แบ่งข้อมูลเป็นส่วน ๆ ลง HDD เช่น RAID5 RAID10)

ประเภทของ Volume ของตัว GlusterFS



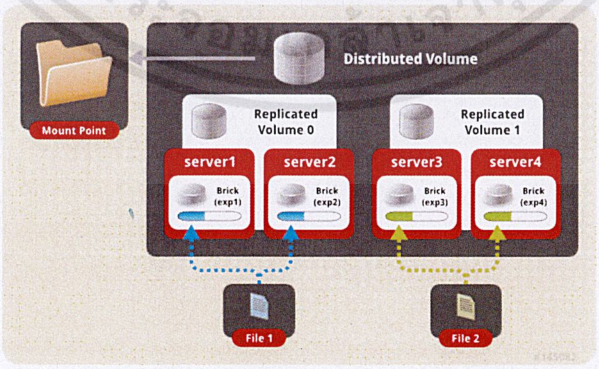
ภาพที่ 2.34 Distributed GlusterFS Volume

1) Distributed GlusterFS Volume เป็น default volume ของ glusterfs คือการเมื่อสร้างไฟล์ลงใน volume นั้น ๆ ก็จะกระจายไปเก็บยัง Brick ต่าง ๆ ถ้าเครื่องไหน Brick พังไฟล์ในเครื่องนั้นก็จะพังไปด้วยเพราะไม่มีสำรอง ข้อดีที่คือประหยัดพื้นที่ ข้อเสียคือการป้องกันไฟล์เสียหายขึ้นอยู่กับ Hardware RAID เครื่องอย่างเดียว ง่าย ๆ คือเน้น ขยายพื้นที่เก็บเอาให้ โตขึ้นเรื่อย ๆ ได้แต่ไม่เน้นสำรอง



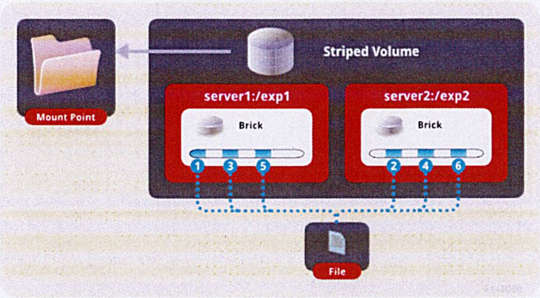
ภาพที่ 2.35 Replicated GlusterFS Volume

2) Replicated GlusterFS Volume รูปแบบนี้จะมีการทำสำเนาข้อมูลไฟล์นั้น ๆ ไปยังอีกเครื่องซึ่งจะตั้งให้เก็บกี่สำเนาก็ได้แล้วแต่จะกำหนด ถ้า Brick ใดเสียก็สามารถ Access ไฟล์นั้นจากอีก Brick แทน เน้นสำรองข้อมูล แต่พื้นที่ก็ น้อยลงไป



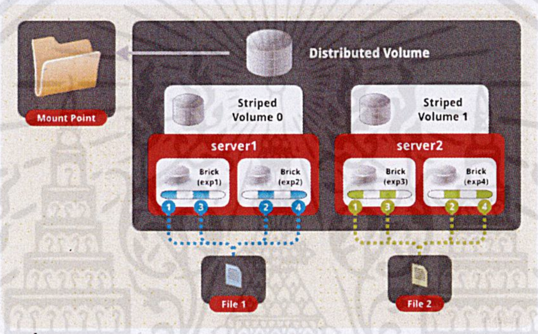
ภาพที่ 2.36 Distributed Replicated GlusterFS Volume

3) Distributed Replicated GlusterFS Volume รูปแบบนี้จะซับซ้อนขึ้น ไฟล์หนึ่งจะกำหนดให้เก็บกี่สำเนา ส่วนจะไปอยู่เครื่องใดบ้าง ระบบจัดการให้เอง แต่จำนวนสำเนาต้องสอดคล้องกับจำนวน Brick คือเอาหลักการ 1+2 มาใช้เพื่อให้ สำรองขึ้นและได้พื้นที่ใช้งานมากยิ่งขึ้น



ภาพที่ 2.37 Striped GlusterFS Volume

4) Striped GlusterFS Volume คือการใช้หลักการในข้อ 1 เพิ่มเติมคือแยกไฟล์ย่อยและเก็บกระจายในแต่ละเครื่อง ช่วยเพิ่มความเร็วในการอ่าน เพราะ หมายความว่าจะอ่านไฟล์หนึ่งจะต้องมีเซิร์ฟเวอร์สองตัวคอยช่วยให้อ่านได้เร็วขึ้น



ภาพที่ 2.38 Distributed Striped GlusterFS Volume

5) Distributed Striped GlusterFS Volume คือการใช้หลักการในข้อ 3 เพิ่มเติมคือแยกไฟล์ย่อย ๆ และเก็บกระจายเช่นเดียวกัน

2.11 PostgreSQL

ภาพที่ 2.39 PostgreSQL Logo

PostgreSQL เป็นระบบการจัดการฐานข้อมูลเชิงวัตถุ-สัมพันธ์ (object-relational) แบบ ORDBMS โดยสามารถใช้รูปแบบคำสั่งของภาษา SQL ได้เกือบทั้งหมด นอกจากนี้ยังเป็นระบบฐานข้อมูลประเภท Open Source

2.12 Jenkins

ภาพที่ 2.40 Jenkins Logo

Jenkins เป็น Open Source Tools ที่ถูกเขียนด้วยภาษา Java โดย Jenkins ถูกสร้างขึ้นมาเพื่อตอบสนอง Concept 2 อย่างหลักๆ คือ Continuous integration และ Continuous Delivery เพื่อใช้กำหนดคำสั่งแต่ละ Stage เพื่อให้ทำงานตามคำสั่ง Flow ของ CI/CD

2.13 Git

ภาพที่ 2.41 Git Logo

Git คือ Version Control แบบ Distributed ตัวหนึ่ง เป็นระบบที่ใช้จัดเก็บและควบคุมการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ชนิดใดก็ได้ ไม่ว่าจะเป็น Text File หรือ Binary File หรือเรียกว่า Source Code โดยสามารถอธิบายประโยชน์ของ Git ได้ดังนี้

1) สามารถ Track version ของ Source Code ย้อนกลับได้ เมื่อจัดเก็บไฟล์เข้าไปในระบบของ Git จะเรียกว่า Git Repository ซึ่งเก็บสำรองข้อมูลและการเปลี่ยนแปลงของ Source Code ทำให้สามารถย้อนกลับไปที่เวอร์ชั่นใดๆ ก่อนหน้า และดูรายละเอียดการเปลี่ยนแปลงของแต่ละเวอร์ชั่นได้

2) ช่วยในการพัฒนาซอฟต์แวร์เป็นทีมสามารถเก็บบันทึกการเปลี่ยนแปลงของ Source Code เวอร์ชั่นล่าสุดไว้ที่ Local Repository ซึ่งสามารถทำงานได้โดยที่ไม่ต้องต่อกับอินเตอร์เน็ต และเมื่อต้องการเปลี่ยนแปลงของ Source Code เวอร์ชั่นล่าสุดให้กับคนในทีมก็สามารถที่จะนำ Source Code Push ขึ้นไปเก็บที่ Remote Repository (Git Hosting) และคนในทีมก็สามารถ Pull เวอร์ชั่นล่าสุดนั้นมารวม (Auto Merge) ที่เครื่องตนเอง ทำให้ Source Code ที่พัฒนาร่วมกันกับคนภายในทีมเป็นเวอร์ชั่นล่าสุดเสมอ ซึ่งปัจจุบันมี Git Hosting หลายตัวเช่น GitLab, GitHub, Bitbucket, Azure DevOps

2.14 GitLab

ภาพที่ 2.42 GitLab Logo

GitLab คือ Open Source Software ที่ถูกพัฒนาขึ้นจาก Git ซึ่งทำ 2 หน้าที่หลักด้วยกัน นั่นคือ เข้ามาช่วยจัดการเก็บ Source Code ของแต่ละโปรเจค (Git Repository) และจัดการ CI/CD (Continuous Integration and Continuous Delivery) โดย GitLab สามารถใช้งานได้ 2 รูปแบบ คือ 1) ใช้งานผ่าน www.gitlab.com 2) ติดตั้งบนเครื่องเซิร์ฟเวอร์ (Self-Hosted)

2.15 JFrog Artifactory

ภาพที่ 2.43 JFrog Logo

ใช้บริหารจัดการตัว Artifact ซึ่งถูก Compile เป็นไฟล์ Binary หรืออาจจะเป็น Source Code ที่ถูกติดตั้ง Dependency ต่างๆที่จำเป็นเข้ามาแล้ว โดยสามารถนำ Artifact ดังกล่าวไปใช้งานได้ทันที โดยหน้าที่หลักของ JFrog เป็นตัวกลางที่เข้ามาจัดการ Artifact เข้ามาเก็บและดาวน์โหลดไปใช้งาน โดยมีการเก็บบันทึกเวอร์ชั่น ไปใช้งานกับโปรเจคอื่นๆได้ โดยจะเก็บ Library ต่างๆจากภายนอกมา Cache เก็บไว้เพื่อใช้งาน

2.16 SonarQube

ภาพที่ 2.44 SonarQube Logo

SonarQube เป็นเครื่องมือที่ช่วยให้เราตรวจสอบคุณภาพ Source Code และแนะนำการแก้ไขให้คุณภาพดียิ่งขึ้น (Code Review) เป็น Open Source โดยสามารถใช้ตรวจสอบ Code ได้มากกว่า 25 ภาษา ใช้ทำการวิเคราะห์ Code ที่เราเขียนด้วยเกณฑ์พื้นฐานต่างๆ เช่น Duplicate Code (Code ที่ซ้ำๆกันในหลายไฟล์ , Bugs & Vulnerabilities (ช่องโหว่หรือสิ่งที่อาจจะทำให้เกิดปัญหา , Code Smells (Code ที่ไม่ได้ทำงานผิดพลาดแต่สามารถปรับปรุงให้ดีขึ้นได้

2.17 Selenium Robot Framework

ภาพที่ 2.45 Selenium Robot Framework Logo

Robot Framework เป็น Automated Test Framework ที่ออกแบบมาเพื่อสำหรับใช้ทำ Acceptance Test Driven Development (ATDD) โดยใช้ลักษณะการเขียน Test case เป็นแบบ Keyword Driven approach คือเขียนเป็นภาษามนุษย์มากขึ้น

2.18 Apache JMeter

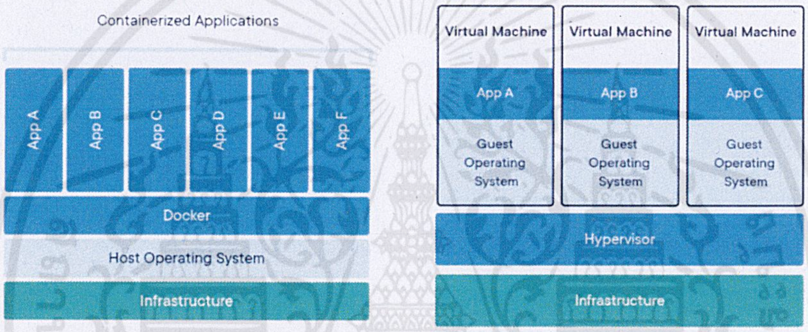
ภาพที่ 2.46 Apache JMeter Logo

Apache JMeter คือ Opensource ที่เขียนด้วย Java ถูกออกแบบมาให้ทำ Performance Test และ ช่วยวัดผลของการ Test ให้สามารถวิเคราะห์ได้ง่ายขึ้น โดยสามารถสร้าง Test Plan สำหรับทำการ Test ด้วยจำนวน Thread, จำนวน Request ที่กำหนด

2.19 Docker

ภาพที่ 2.47 Docker Logo

Docker คือ engine ตัวหนึ่งที่มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการ Run service ที่ต้องการ มีการทำงานคล้ายคลึงกับ Virtual Machine เช่น VMWare, VirtualBox, XEN, KVM แต่ข้อแตกต่างที่ชัดเจนคือ Virtual Machine ที่รู้จักกันก่อนหน้านี้นั้น เป็นการจำลองทั้ง OS เพื่อใช้งานและหากต้องการใช้งาน service ใดๆ จึงทำการติดตั้งเพิ่มเติมบน OS นั้นๆ แต่สำหรับ docker แล้วจะใช้ container ในการจำลองสภาพแวดล้อมขึ้นมา เพื่อใช้งานสำหรับ 1 service ที่ต้องการใช้งานเท่านั้น โดยไม่ต้องมีส่วนของ OS เข้าไปเกี่ยวข้องเหมือน Virtual Machines อื่นๆ



ภาพที่ 2.48 Docker Architecture

2.20 Harbor Registry

ภาพที่ 2.49 Harbor Logo

Docker image ที่สร้างขึ้นนั้นสามารถเก็บไว้ในสิ่งที่เรียกว่า Docker registry ซึ่งแบ่งได้เป็น 2 อย่างคือ 1) Public registry อยู่ที่ hub.docker.com เราจะต้องลงทะเบียนเพื่อขอมีบัญชีผู้ใช้จึงจะสามารถอัพโหลด image ไปเก็บไว้ได้ 2) Private (Local) registry จะเป็นการสร้างที่เก็บส่วนตัว ที่เก็บของหน่วยงาน ซึ่งสามารถติดตั้งบนเครื่องเซิร์ฟเวอร์เองได้ (Self-Hosted)

2.21 Rocket Chat

ภาพที่ 2.50 Rocket Chat Logo

Rocket Chat ระบบแชทที่สร้างด้วย JavaScript รองรับ Video/Voice Conference

บทที่ 3
การออกแบบและพัฒนา

ในการพัฒนาระบบการควบคุมและจัดการข้อมูล จะนำกระบวนการและเครื่องมือของ DevOps มาประยุกต์ใช้ร่วมกับระบบดังกล่าวเพื่อให้การพัฒนาซอฟต์แวร์มีประสิทธิภาพและรวดเร็ว

3.1 ขั้นตอนการดำเนินงาน

1) ศึกษาความต้องการตามเอกสาร TOR (Term of Reference)

2) วิเคราะห์และออกแบบ API รวมถึงขั้นตอนการทำงานของระบบ ศึกษาเทคโนโลยีสำหรับการพัฒนาโครงงาน และวางแผนการทำงานในการพัฒนาโครงงานให้สำเร็จภายในระยะเวลาที่กำหนด

3) พัฒนาระบบตามที่ออกแบบไว้ บนเครื่อง Local โดยการใช้ Docker จำลองสภาพแวดล้อม

4) ทำการ Deploy API ไปยัง OpenShift

5) ทดสอบระบบตามเอกสาร Test Case ที่ออกแบบไว้ให้ทำงานได้ถูกต้อง

6) จัดทำเอกสารอธิบายสถาปัตยกรรม กระบวนการทำงาน และโครงสร้างของระบบ

3.2 ความต้องการของระบบ

ระบบการควบคุมและจัดการข้อมูลจะต้องครอบคลุมตามความสามารถตามรายละเอียดความต้องการ (Requirement) ดังต่อไปนี้

3.2.1 การจัดการผู้ใช้งาน

1) การเข้าสู่ระบบ

2) การออกจากระบบ

3) การลงทะเบียนผู้ใช้งาน

4) การยืนยันการลงทะเบียนโดยผู้ใช้งานผ่านอีเมล

6) การรีเซ็ตรหัสผ่านและการตั้งค่ารหัสผ่านใหม่ของผู้ใช้งาน

7) การค้นหาข้อมูลผู้ใช้งาน

8) การเพิ่ม ปรับปรุง ลบข้อมูล ผู้ใช้งาน

9) การอนุมัติหรือไม่อนุมัติ ผู้ใช้งาน

10) การปิดบัญชีของผู้ใช้งาน

11) การอนุมัติ เพื่อปิดบัญชีของผู้ใช้งาน

12) การกู้คืนผู้ใช้งาน

3.2.2 การจัดการสิทธิ์ของผู้ใช้งาน

1) สามารถกำหนดสิทธิการเข้าถึงและการแสดงผลชุดข้อมูล

2) สามารถกำหนดสิทธิการจัดการความคิดเห็น

3) สามารถกำหนดสิทธิการจัดการการคำถาม-คำตอบ

4) สามารถกำหนดสิทธิการแสดงประวัติการใช้หรือทำรายการข้อมูล

5) การเพิ่ม แก้ไข ลบ สิทธิการใช้งานองค์ประกอบ

6) การเพิ่ม แก้ไข ลบ บทบาทการใช้งานของผู้ใช้งาน

3.2.3 การจัดการชุดข้อมูล

1) การแสดงรายการและรายละเอียดของชุดข้อมูล

2) การลบชุดข้อมูล

3) การดาวน์โหลดชุดข้อมูล

4) การนำเข้าชุดข้อมูลโดยใช้แฟ้มข้อมูลที่มีรูปแบบ CSV, JSON, TXT, GeoJSON, XML

5) การแสดงเมตาดาต้า (Show Meta Data)

6) การเพิ่ม ปรับปรุง ลบเมตาดาต้า (Add, Update, Delete Meta Data)

3.2.4 การจัดการหน่วยงาน

1) การเพิ่ม ปรับปรุง ลบ หน่วยงาน

2) การค้นหาข้อมูลหน่วยงาน (Search Department) โดยสามารถแสดงรายการหน่วยงานเป็นไปตามเงื่อนไขการค้นหา

3.2.5 การจัดการหมวดข้อมูล หรือกลุ่มข้อมูล

1) สามารถเพิ่ม ปรับปรุง ลบ ค้นหาหมวดข้อมูล หรือหมวดหมู่ย่อย

3.2.6 ระบบแสดงประวัติการใช้หรือการทำรายการข้อมูล

1) การเพิ่มประวัติการใช้หรือการทำรายการข้อมูลในหน่วยงาน

2) การแสดงประวัติการใช้หรือการทำรายการข้อมูลในหน่วยงาน โดยแสดงตามกลุ่มผู้ใช้งาน ตามหน่วยงานผู้ใช้งาน

3.2.7 ระบบความคิดเห็น

1) การเพิ่ม ปรับปรุง ลบ ความคิดเห็น

3.2.8 ระบบถามตอบ

1) การเพิ่มข้อความ โดยมีประเภทข้อความ 2 ประเภท คือ คำถามและคำตอบ

2) การปรับปรุงข้อความ

3) การลบข้อความแบ่งออกเป็น 2 แบบ คือ 1 กรณีลบข้อความประเภทคำถาม ให้ลบข้อความประเภทคำตอบที่เกี่ยวข้องภายใต้คำถามนั้นออกทั้งหมด 2 กรณีลบข้อความประเภทคำตอบ ให้ลบข้อความประเภทคำตอบรายนั้นเพียงรายการเดียว

4) การแสดงข้อความ สามารถแสดงทุกข้อความประเภทคำตอบของคำถามนั้น

3.2.9 ระบบการค้นหาข้อมูล

1) สืบค้นข้อมูลจากการกำหนดเงื่อนไขใน เช่น ชื่อชุดข้อมูล ชื่อแฟ้มข้อมูล ตามช่วงเวลา

2) สืบค้นข้อมูลจากเนื้อหาข้อมูลภายในคำอธิบายข้อมูล

3.2.10 ประเภทกลุ่มผู้ใช้งานในระบบ

1) กลุ่มผู้ใช้งานที่ดูรายงานอย่างเดียว (View only Users) คือ กลุ่มผู้ใช้งานที่เข้ามาดูรายงานโดยเฉพาะ จะไม่สามารถสร้างชุดข้อมูล หรือทำการวิเคราะห์ข้อมูลได้ แต่จะสามารถดูคำถาม-คำตอบ รายงาน หรือ เนื้อหาตามสิทธิ์ที่ได้รับอนุญาตเท่านั้น

2) กลุ่มผู้ใช้งานที่วิเคราะห์ข้อมูลและสร้างรายงาน (Analytic Users) คือ กลุ่มผู้ใช้งานที่สามารถสร้างชุดข้อมูล โดยการนำเข้า (Upload) ชุดข้อมูลได้เอง และสามารถวิเคราะห์และจัดทำข้อมูลเชิงภาพ (Visualize) ทั้งจากชุดข้อมูลของตนเองและชุดข้อมูลของคนอื่นที่อนุญาตได้

3) กลุ่มผู้ดูแลจัดการบทความ (Content Administrators) คือ กลุ่มผู้ใช้งานที่สามารถจัดการเกี่ยวกับคำถาม-คำตอบ เพื่อเผยแพร่ให้กับผู้ใช้งานหรือหน่วยงานต่างๆ

4) กลุ่มผู้ดูแลจัดการบัญชีผู้ใช้งาน (User Administrators) คือ กลุ่มผู้ใช้งานที่ทำหน้าที่จัดการเกี่ยวกับบัญชีผู้ใช้งาน การอนุมัติในการสร้างบัญชีผู้ใช้งาน รวมถึงจัดการเกี่ยวกับหน่วยงานด้วย

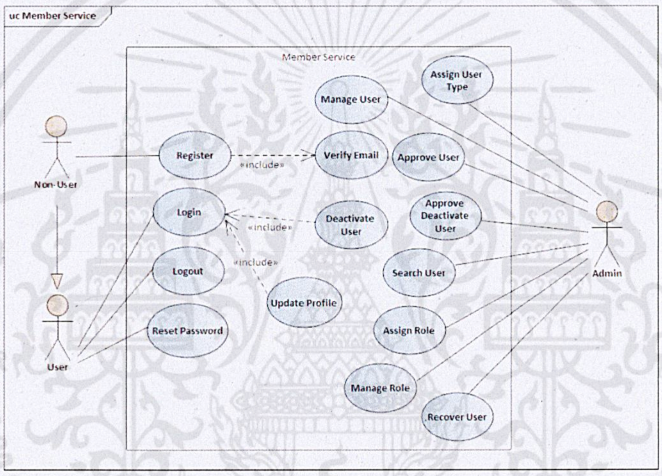
5) กลุ่มผู้ดูแลสูงสุด (Super Administrators) คือ กลุ่มผู้ใช้งานที่มีสิทธิ์ ในการกำหนดค่าของระบบ เช่น จัดการประเภทและหมวดหมู่ของบทความ การจัดการหน้าเว็บหลัก (Portal) และการกำหนดนโยบาย (Policy) โดยรวม เป็นต้น

3.3 แผนภาพอธิบายโครงสร้างและการทำงานของระบบ

3.3.1 แผนภาพ Use Case Diagram

โครงสร้างของระบบควบคุมและจัดการข้อมูล สามารถแสดงอยู่ในรูปของแผนภาพ Use Case ที่มาจากความต้องการของระบบ โดยถูกแบ่งเป็น Service ดังต่อไปนี้

3.3.1.1 Member Service เซอร์วิสสมาชิก



ภาพที่ 3.1 Member Service

1) Use Case Login การเข้าสู่ระบบ โดยผู้ที่ลงทะเบียนเป็นสมาชิกของระบบแล้ว จะต้องใช้ชื่อผู้ใช้งานและรหัสผ่าน เพื่อใช้สำหรับเข้าสู่ระบบ เมื่อระบบตรวจสอบว่าข้อมูลถูกต้อง และมีข้อมูลอยู่ในระบบ จะทำการส่ง Bearer Token เพื่อใช้เป็น Access Token ในการใช้งาน Service

2) Use Case Logout การออกจากระบบ โดยผู้ที่เข้าสู่ระบบแล้ว จะต้องทำการใช้ Access Token ที่ได้รับไปเมื่อตอนเข้าสู่ระบบส่งเข้ามาเพื่อนำ Session นี้ออก

3) Use Case Register การลงทะเบียนผู้ใช้งาน โดยผู้ลงทะเบียนยังไม่ได้เป็นสมาชิกของระบบ จะต้องทำการส่งข้อมูลที่ใช้ลงทะเบียนผู้ใช้งานเข้ามา เมื่อระบบตรวจสอบว่าข้อมูลถูกต้อง และไม่ซ้ำกับข้อมูลที่มีอยู่ในระบบ จะทำการลงทะเบียนให้ผู้ใช้งาน

4) Use Case Verify E-mail การยืนยันตัวตนหลังจากลงทะเบียน เมื่อผู้ใช้ได้รับอีเมลจากการลงทะเบียนแล้ว จะต้องเปิดใช้งานจากลิงค์ที่ส่งไปพร้อมกับอีเมล เพื่อทำการยืนยันตัวตน เมื่อยืนยันตัวตนสำเร็จจะต้องรอการอนุมัติจากผู้ดูแลระบบต่อไป

5) Use Case Reset Password ผู้ใช้งานจำรหัสผ่านไม่ได้ และต้องการตั้งค่ารหัสผ่านใหม่ ต้องส่งชื่อผู้ใช้งานที่ต้องการตั้งค่ารหัสผ่านใหม่เข้ามา ระบบจะส่งอีเมลพร้อมลิงค์เพื่อใช้ตั้งค่ารหัสใหม่ไปยังผู้ใช้งาน

6) Use Case Set New Password การตั้งค่ารหัสผ่านใหม่ ต้องทำการเข้าสู่ระบบและใช้ Access Token ในการส่งรหัสผ่านใหม่เข้ามาเพื่อแก้ไขในระบบ

7) Use Case Deactivate User ผู้ใช้งานผ่านการอนุมัติจากผู้ดูแลระบบแล้ว และต้องการยกเลิกการใช้งานบัญชีเพื่อปิดการใช้งานหลังจากทำงานนี้แล้วจะเปลี่ยนสถานะของบัญชีเป็นรออนุมัติจากผู้ดูแลระบบในการปิดใช้งาน

8) Use Case Search User ผู้ดูแลต้องค้นหาผู้ใช้งานต่างๆในระบบ

9) Use Case Manage User ผู้ดูแลต้องการจัดการกับผู้ใช้งาน เพิ่ม ลบ แก้ไข

10) Use Case Update Profile ผู้ใช้งานต้องการเปลี่ยนข้อมูลส่วนตัว จะต้องระบุไอดีของผู้ใช้งาน, ข้อมูลส่วนตัวที่ต้องการแก้ไข เพื่อใช้เปลี่ยนข้อมูลส่วนตัว

11) Use Case Assign User Type ผู้ดูแลต้องการกำหนดประเภทของผู้ใช้งาน ต้องระบุไอดีของผู้ใช้งาน, ประเภทของผู้ใช้งาน เพื่อใช้เปลี่ยนประเภทของผู้ใช้งาน

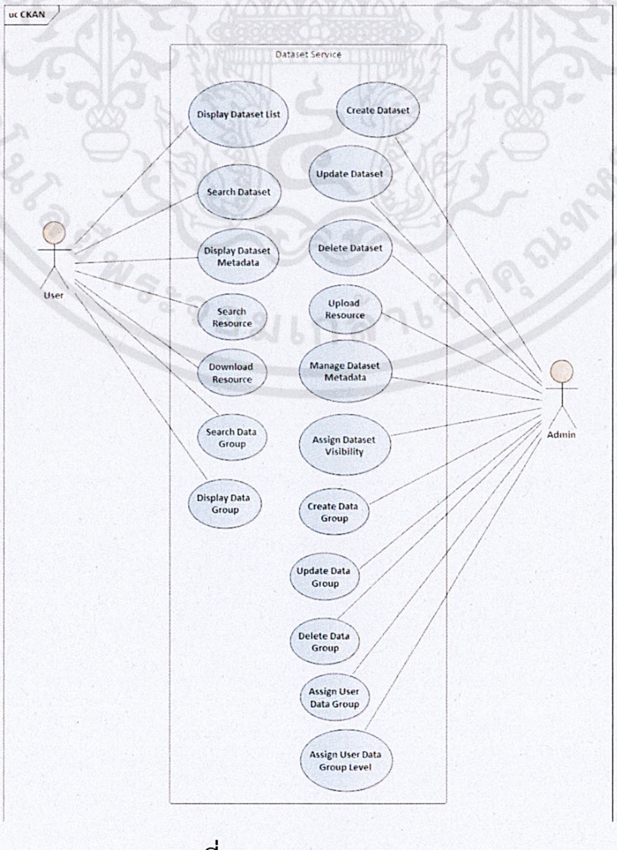
12) Use Case Approve Deactivate User ผู้ดูแลอนุมัติการปิดใช้งานของบัญชี หลังจากทำแล้วสถานะของบัญชีจะเปลี่ยนเป็นปิดบัญชีแล้ว

13) Use Case Assign Role ผู้ดูแลต้องการกำหนดสิทธิ์ให้ผู้ใช้งาน

14) Use Case Manage Role ผู้ดูแลจัดการสิทธิ์ของผู้ใช้งาน เพิ่ม ลบ แก้ไข

15) Use Case Recover User ผู้ดูแลต้องการกู้คืนบัญชีผู้ใช้งาน เมื่อทำงานสำเร็จจะเปลี่ยนสถานะบัญชีกลับมาเป็น เปิดใช้งานแล้ว

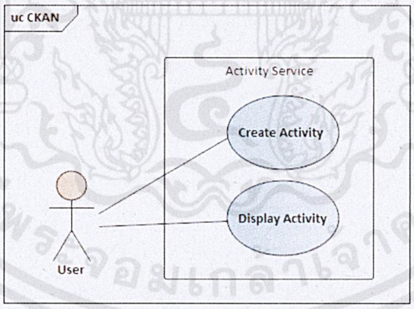
3.3.1.2 Dataset Service เซอร์วิสชุดข้อมูล



ภาพที่ 3.2 Dataset Service

1) Use Case Display Dataset List ผู้ใช้งานสามารถแสดงรายการ Dataset
2) Use Case Search Dataset คือ ผู้ใช้งานสามารถค้นหา Dataset
3) Use Case Display Dataset Metadata ผู้ใช้งานสามารถแสดง Metadata
4) Use Case Search Resource ผู้ใช้งานค้นหา Resource ของ Dataset
5) Use Case Download Resource ผู้ใช้งานสามารถดาวน์โหลด Resource
6) Use Case Search Data Group ผู้ใช้งานสามารถค้นหากลุ่มของข้อมูล
7) Use Case Display Data Group ผู้ใช้งานสามารถแสดงกลุ่มของข้อมูล
8) Use Case Create Dataset ผู้ดูแลสามารถเพิ่มชุดข้อมูลใหม่
9) Use Case Update Dataset ผู้ดูแลสามารถแก้ไขชุดข้อมูล
10) Use Case Delete Dataset ผู้ดูแลสามารถลบชุดข้อมูลได้
11) Use Case Upload Resource ผู้ดูแลเพิ่มแฟ้มข้อมูลเข้ามาที่ชุดข้อมูล
12) Use Case Manage Dataset Metadata ผู้ดูแลจัดการรายละเอียดชุดข้อมูล
13) Use Case Assign Dataset Visibility ผู้ดูแลกำหนดประเภทชุดข้อมูล
14) Use Case Create Data Group ผู้ดูแลเพิ่มกลุ่มของชุดข้อมูล
15) Use Case Update Data Group ผู้ดูแแก้ไขกลุ่มของชุดข้อมูล
16) Use Case Delete Data Group ผู้ดูแลลบกลุ่มของชุดข้อมูล
17) Use Case Assign User Data Group ผู้ดูแลเพิ่มสมาชิกในกลุ่มของชุดข้อมูล
18) Use Case Upload Assign User Data Group Level ผู้ดูแลกำหนดระดับสมาชิกในกลุ่มของข้อมูล

3.3.1.3 Activity Service เซอร์วิสกิจกรรม

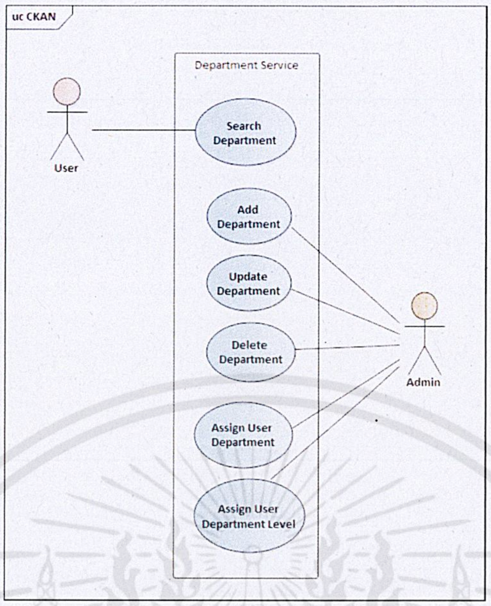


ภาพที่ 3.3 Activity Service

1) Use Case Create Activity ผู้ใช้งานใช้งานระบบ จะมีการบันทึกรายการการทำกิจกรรมทั้งหมด ลงในฐานข้อมูล

2) Use Case Display Activity ผู้ใช้งานสามารถดูประวัติการทำรายการทั้งหมดบนระบบได้

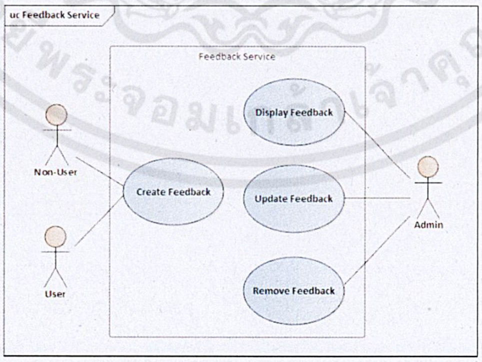
3.3.1.4 Department Service เซอร์วิสหน่วยงาน



ภาพที่ 3.4 Department Service

1) Use Case Search Department ผู้ใช้งานสามารถค้นหาหน่วยงาน
2) Use Case Add Department ผู้ดูแลสามารถเพิ่มหน่วยงาน
3) Use Case Update Department ผู้ดูแลสามารถแก้ไขหน่วยงาน
4) Use Case Delete Department ผู้ดูแลสามารถลบหน่วยงาน
5) Use Case Assign User Department ผู้ดูแลเพิ่มสมาชิกของหน่วยงาน
6) Assign User Department Level ผู้ดูแลกำหนดระดับของสมาชิกในหน่วยงาน

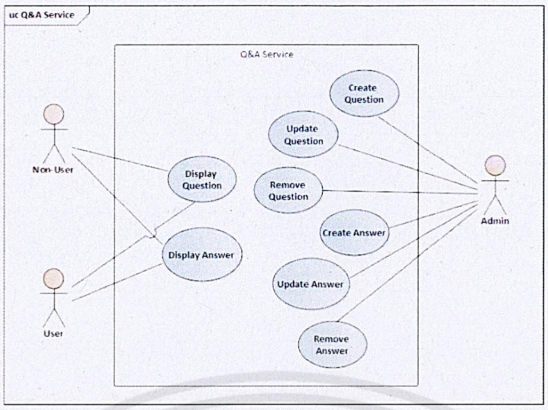
3.3.1.5 Feedback Service เซอร์วิสความเห็น



ภาพที่ 3.5 Feedback Service

1) Use Case Create Feedback คือ ผู้ใช้งานสามารถสร้างความเห็นใหม่
2) Use Case Display Feedback คือ ผู้ใช้งานสามารถดูความเห็นได้
3) Use Case Update Feedback คือ ผู้ใช้งานสามารถแก้ไขความเห็นได้
4) Use Case Remove Feedback คือ ผู้ใช้งานสามารถลบความเห็นได้

3.3.1.6 Q&A Service เซอร์วิสคำถาม-คำตอบ



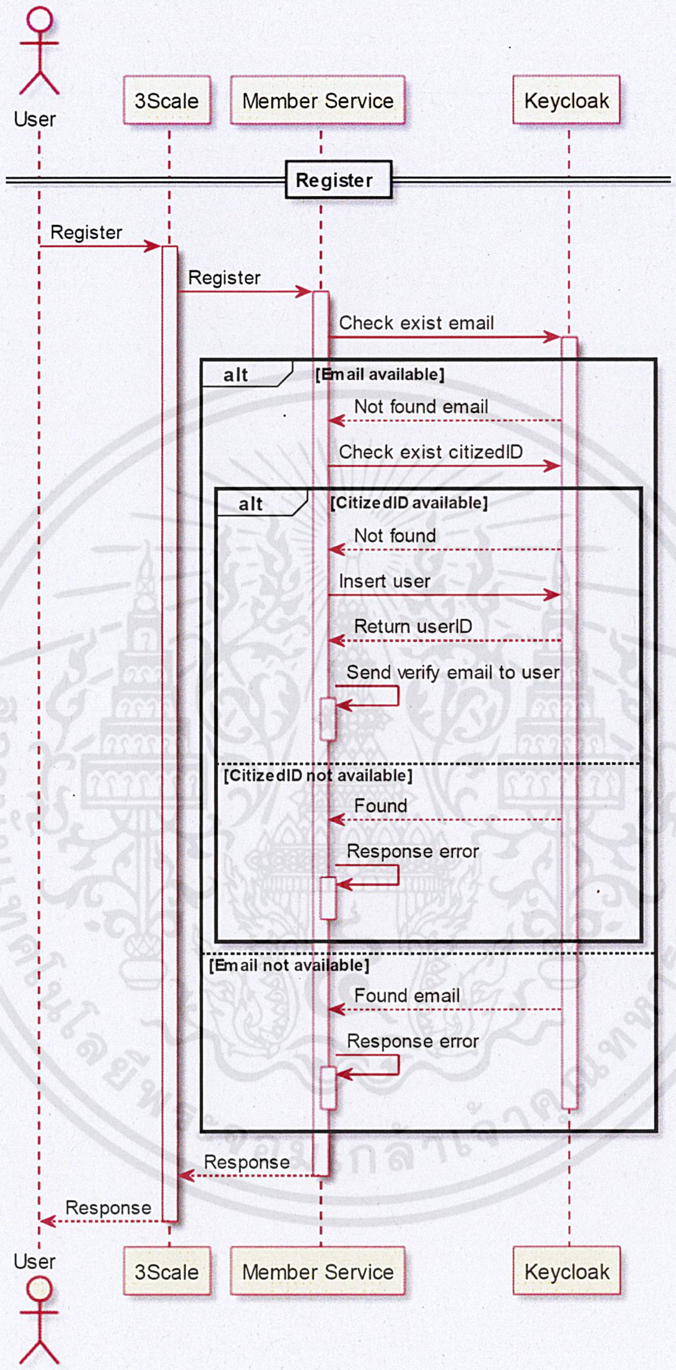
ภาพที่ 3.6 Q&A Service

1) Use Case Display Question คือ ผู้ใช้งานสามารถดูคำถาม
2) Use Case Display Answer คือ ผู้ใช้งานสามารถดูคำตอบของคำถาม
3) Use Case Create Question คือ ผู้ใช้งานสามารถสร้างคำถามใหม่
4) Use Case Update Question คือ ผู้ใช้งานสามารถแก้ไขคำถาม
5) Use Case Remove Question คือ ผู้ใช้งานสามารถลบคำถาม
6) Use Case Create Answer คือ ผู้ใช้งานสามารถสร้างคำตอบของคำถามที่ระบุ
8) Use Case Remove Answer คือ ผู้ใช้งานสามารถลบคำตอบของคำถามที่ระบุ

3.3.2 แผนภาพ Sequence Diagram

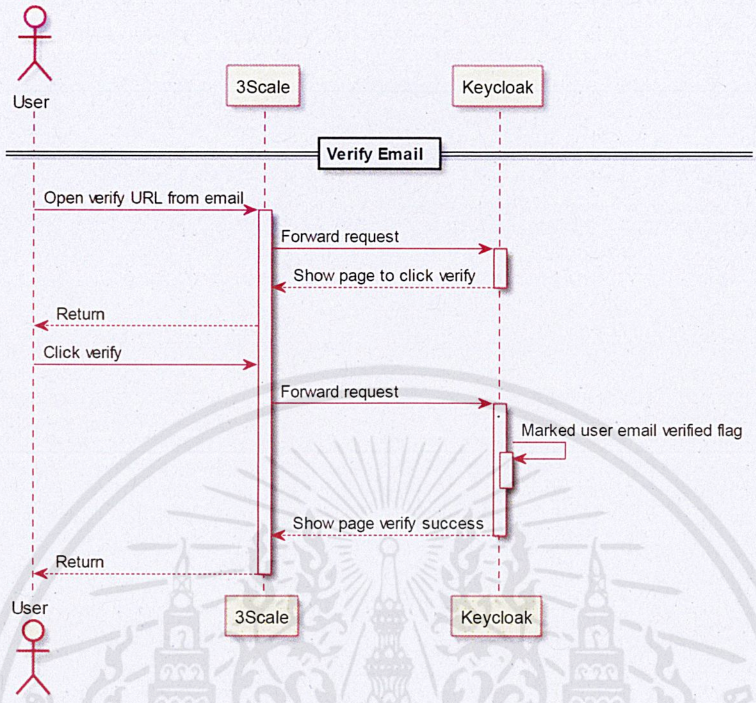
เนื่องจากระบบการควบคุมและจัดการข้อมูลมีขั้นตอนการทำงานหลากหลายขั้นตอน และบางขั้นตอนมีความซับซ้อน ดังนั้นผู้จัดทำจึงออกแบบการทำงานของการทำงานแต่ละขั้นตอนออกมาในรูปแบบของ Sequence Diagram ที่สามารถแสดงให้เห็นถึงลำดับการทำงานและการปฏิสัมพันธ์ (Sequence of Interactions) ตามลำดับของเวลาจากก่อนไปหลัง

3.3.2.1 Function Register



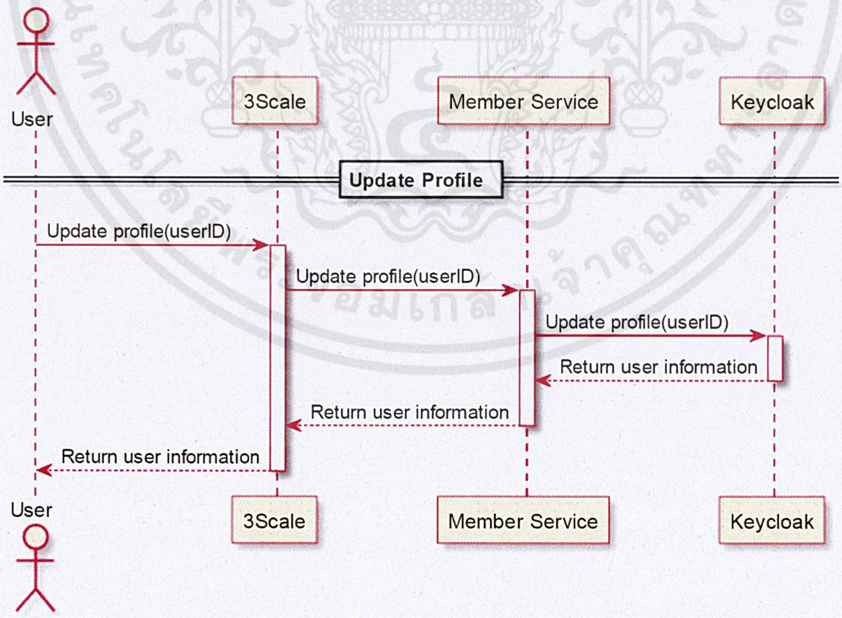
ภาพที่ 3.7 Sequence Diagram ของ Function Register

3.3.2.2 Function Verify Email



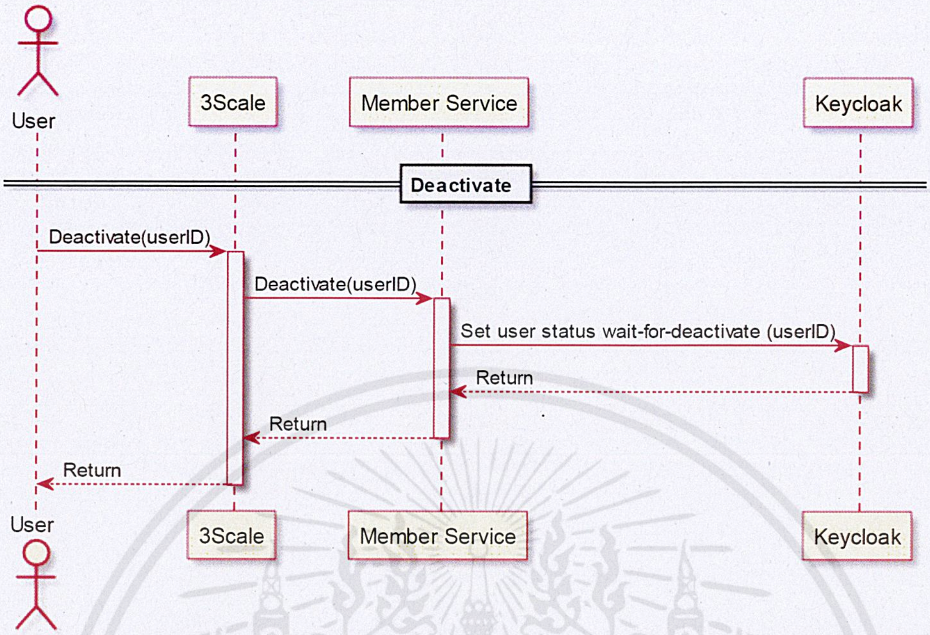
ภาพที่ 3.8 Sequence Diagram ของ Function Verify Email

3.3.2.3 Function Update Profile



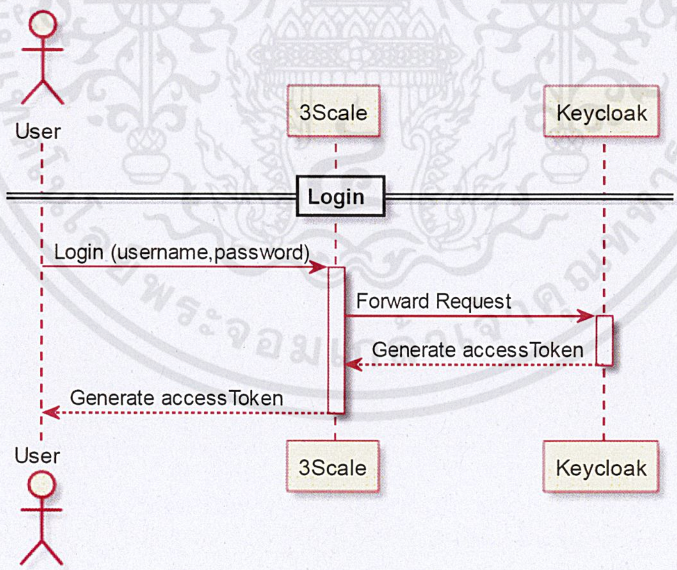
ภาพที่ 3.9 Sequence Diagram ของ Function Update Profile

3.3.2.4 Function Deactivate



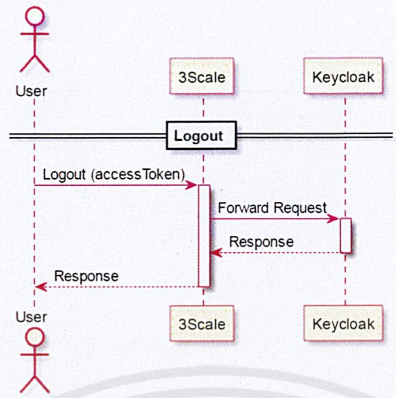
ภาพที่ 3.10 Sequence Diagram ของ Function Deactivate

3.3.2.5 Function Login



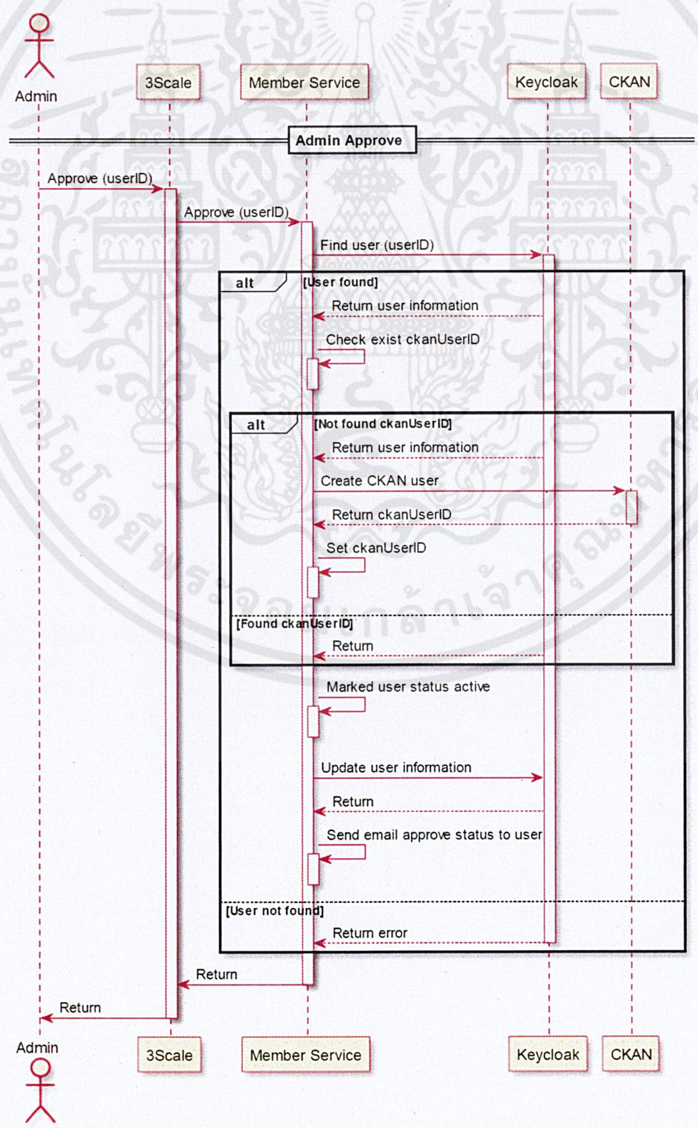
ภาพที่ 3.11 Sequence Diagram ของ Function Login

3.3.2.6 Function Logout



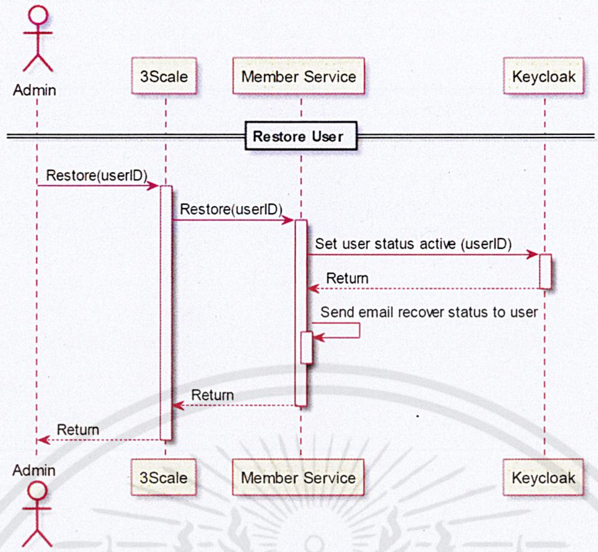
ภาพที่ 3.12 Sequence Diagram ของ Function Logout

3.3.2.7 Function Admin Approve



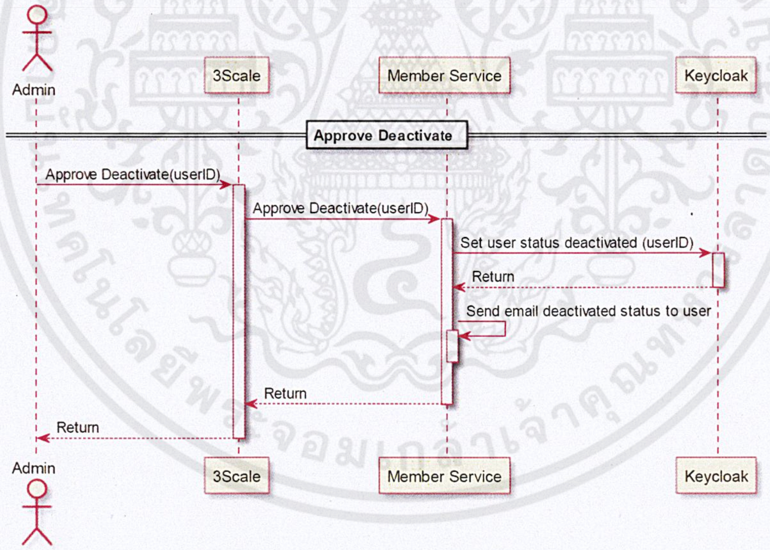
ภาพที่ 3.13 Sequence Diagram Function ของ Admin Approve

3.3.2.8 Function Restore User



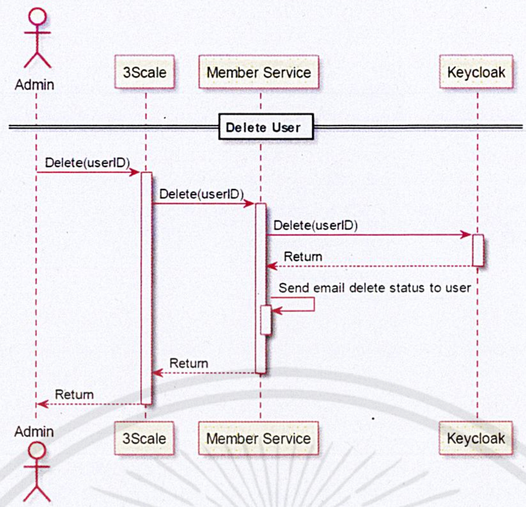
ภาพที่ 3.14 Sequence Diagram ของ Function Restore User

3.3.2.9 Function Approve Deactivate



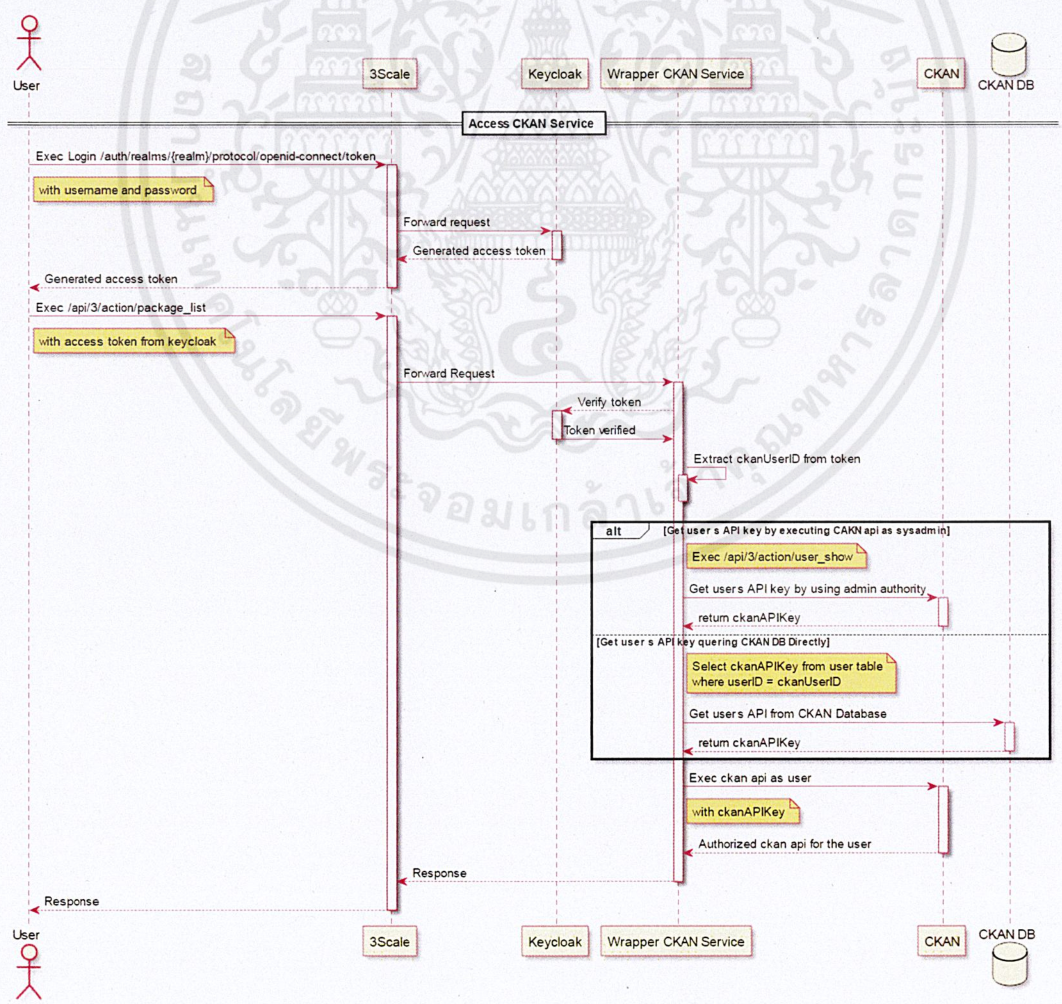
ภาพที่ 3.15 Sequence Diagram ของ Function Approve Deactivate

3.3.2.10 Function Delete User



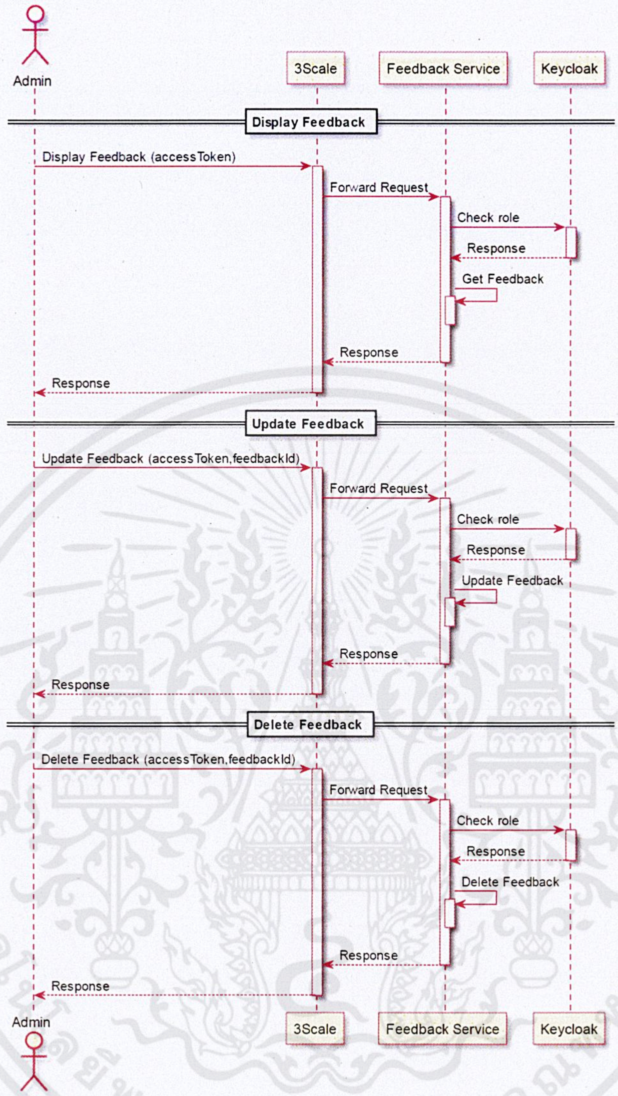
ภาพที่ 3.16 Sequence Diagram ของ Function Delete User

3.3.2.11 Function Access CKAN Service



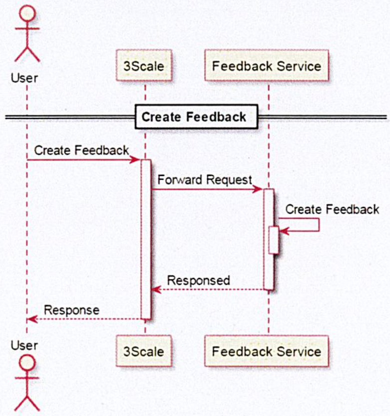
ภาพที่ 3.17 Sequence Diagram ของ Function Access CKAN Service

3.3.2.12 Function Manage Feedback



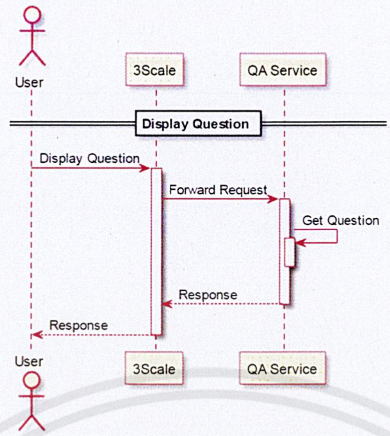
ภาพที่ 3.18 Sequence Diagram ของ Function Manage Feedback

3.3.2.13 Function Create Feedback



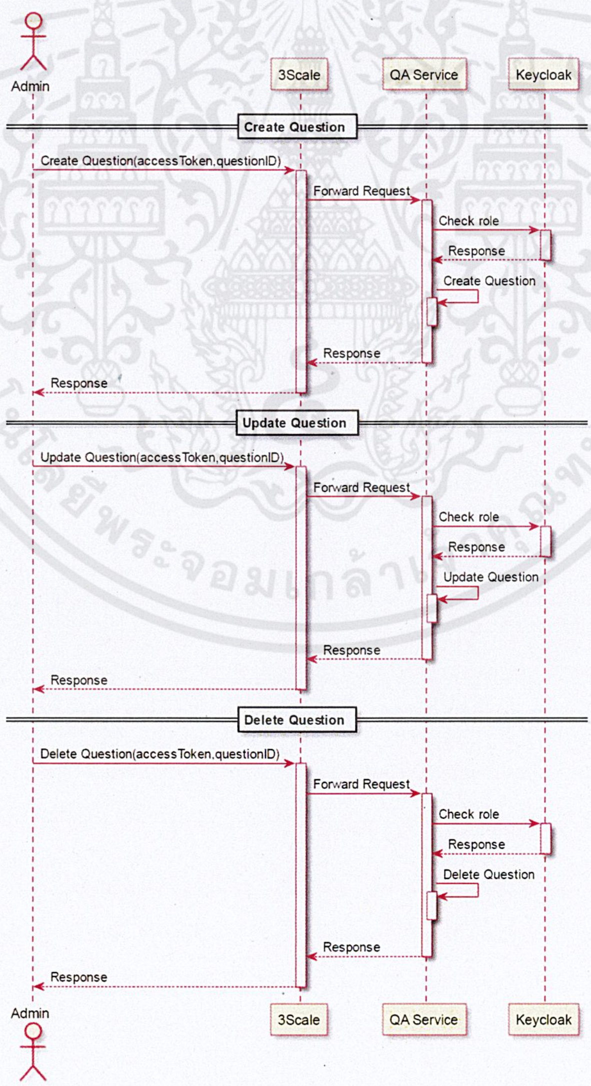
ภาพที่ 3. 19 Sequence Diagram ของ Function Create Feedback

3.3.2.14 Function Display Question



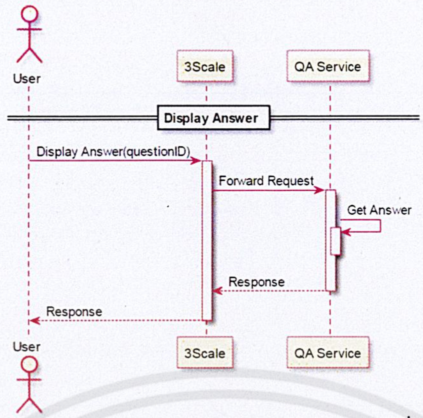
ภาพที่ 3.20 Sequence Diagram ของ Function Display Question

3.3.2.15 Function Manage Question



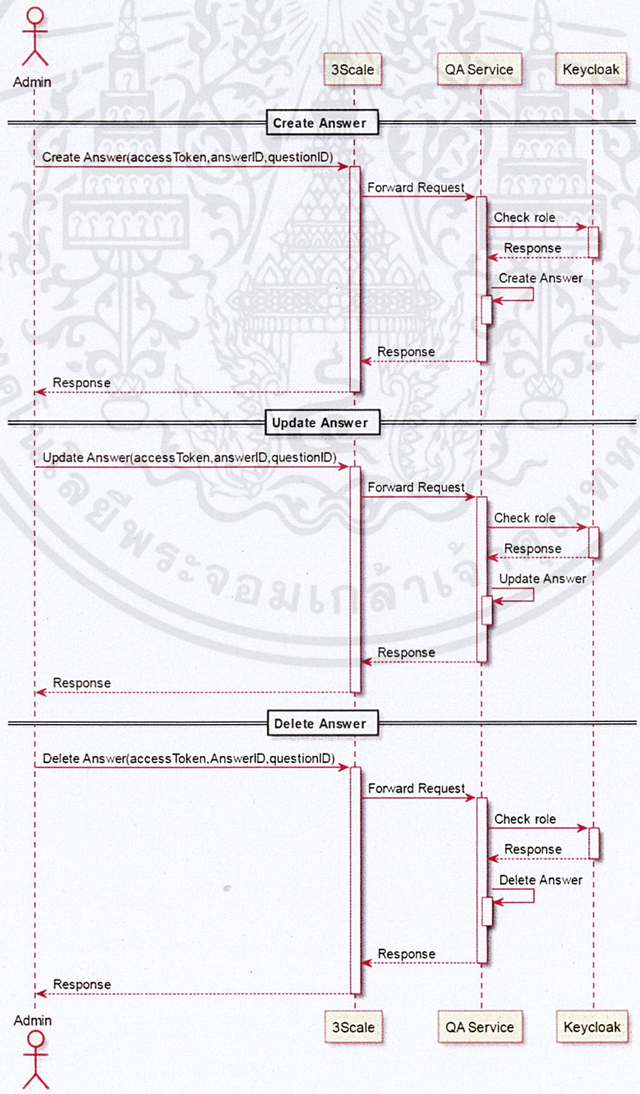
ภาพที่ 3.21 Sequence Diagram ของ Function Manage Question

3.3.2.16 Function Display Answer



ภาพที่ 3.22 Sequence Diagram ของ Function Display Answer

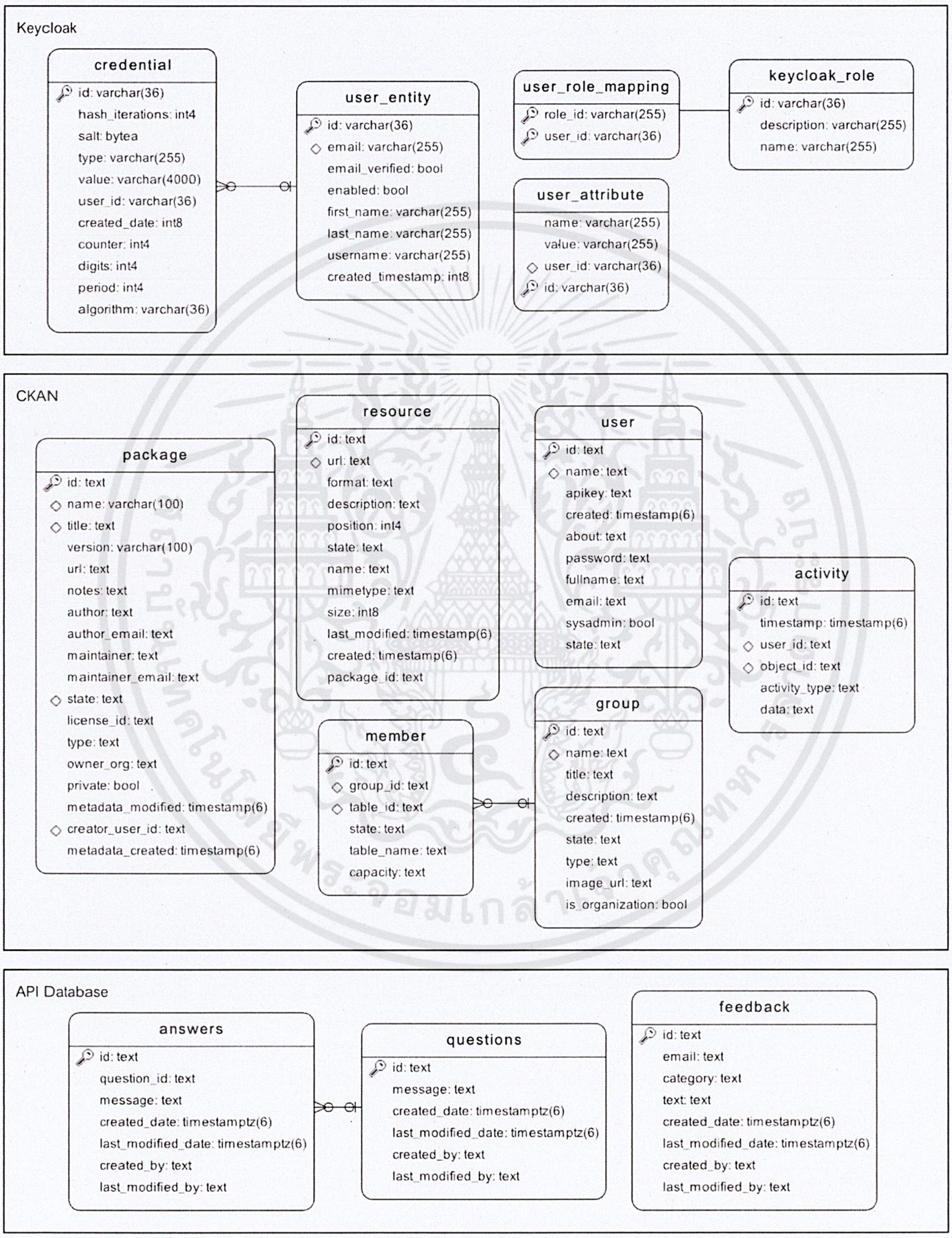
3.3.2.17 Function Manage Answer



ภาพที่ 3.23 Sequence Diagram ของ Function Manage Answer

3.3.3 Database Schema Diagram

ในส่วนของการออกแบบรายละเอียดภาพรวมทั้งหมดของโครงสร้างฐานข้อมูล ผู้จัดทำได้นำเอา Schema Diagram มาใช้ในการออกแบบและอธิบายโครงร่างของฐานข้อมูลทั้งหมดสำหรับฐานข้อมูลของ KeyCloak และ CKAN จะแสดงเฉพาะในส่วนที่มีความเกี่ยวข้องกับโครงงานเท่านั้น



ภาพที่ 3.24 Database Schema Diagram

3.3.3.1 ตารางในฐานข้อมูล CKAN

ตารางที่ 3.1 ตาราง user ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของผู้ใช้งาน CKAN
name		text	ชื่อผู้ใช้งาน
apiKey		bool	apiKey สำหรับใช้งาน CKAN
created		timestamp(6)	วันที่และเวลาที่สร้างผู้ใช้งาน
about		text	รายละเอียดเกี่ยวกับผู้ใช้งาน
password		text	รหัสผ่านผู้ใช้งาน
fullname		text	ชื่อเต็มผู้ใช้งาน
email		text	อีเมลของผู้ใช้งาน
sysadmin		bool	ประเภทผู้ใช้งาน (t = Admin , f = User)
state		text	สถานะ (active , deleted)

ตารางที่ 3.2 ตาราง group ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของกลุ่มข้อมูล
name		text	ชื่อของกลุ่มข้อมูล
title		text	ชื่อเต็มของกลุ่มข้อมูล
description		text	รายละเอียดของกลุ่มข้อมูล
create		timestamp(6)	วันที่และเวลาที่สร้างกลุ่มข้อมูล
state		text	สถานะ (active , deleted)
type		text	ประเภท (group , organization)
image_url		text	ตำแหน่งที่เก็บรูปภาพ
is_organization		bool	เป็นองค์กร (t = เป็น , f = ไม่เป็น)

ตารางที่ 3.3 ตาราง activity ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของกิจกรรม
user_id		text	เลขรหัสของผู้ใช้งาน CKAN
timestamp		timestamp(6)	วันที่และเวลาที่สร้างรายการกิจกรรม
object_id		text	เลขรหัสของกิจกรรมที่เกี่ยวข้อง
activity_type		text	ประเภทของกิจกรรม
data		text	ข้อมูลของกิจกรรม

ตารางที่ 3.4 ตาราง member ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของสมาชิกในกลุ่มข้อมูล
group_id		text	เลขรหัสของกลุ่มข้อมูล
table_id		text	เลขรหัสของ user , package
table_name		text	ชื่อตาราง user , package
state		text	สถานะ (active , deleted)
capacity		text	Group (admin, member) Organization (admin, editor, member) Dataset in Group (public) Dataset in Organization (organization)

ตารางที่ 3.5 ตาราง package ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของ Package
name		varchar (100)	ชื่อของ Package
title		text	ชื่อเต็มของ Package
version		varchar (100)	เวอร์ชั่นของข้อมูล
url		text	ที่มาของข้อมูล
notes		text	รายละเอียดชุดข้อมูล
author		text	ชื่อเจ้าของข้อมูล
author_email		text	อีเมลของเจ้าของข้อมูล
maintainer		text	ชื่อผู้ดูแล
maintainer_email		text	อีเมลของผู้ดูแลชุดข้อมูล
state		text	สถานะ (active , deleted)
license		text	ประเภท License ของชุดข้อมูล
type		text	dataset
owner_org		text	เลขรหัสของ Group Organization
private		bool	ส่วนตัว (t = เป็น , f = ไม่เป็น)
metadata_modified		timestamp(6)	วันที่และเวลาที่แก้ไขชุดข้อมูล
metadata_created		timestamp(6)	วันที่และเวลาที่สร้างชุดข้อมูล
creator_user_id		text	เลขรหัสของผู้ใช้งาน CKAN

ตารางที่ 3.6 ตาราง resource ใน CKAN

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของแฟ้มข้อมูล
url		text	ชื่อของแฟ้มข้อมูล
format		text	ประเภทของแฟ้มข้อมูล
description		text	รายละเอียดของแฟ้มข้อมูล
position		Int4	ลำดับของแฟ้มข้อมูลในชุดข้อมูล
state		text	สถานะ (active , deleted)
name		text	ชื่อของแฟ้มข้อมูล
mimetype		text	มาตรฐานการสื่อสารของเอกสาร
size		int8	ขนาดของแฟ้มข้อมูล
metadata_modified		timestamp(6)	วันที่และเวลาที่แก้ไขชุดข้อมูล
metadata_created		timestamp(6)	วันที่และเวลาที่สร้างชุดข้อมูล
title		text	ชื่อเต็มของ Package

3.3.3.2 ตารางในฐานข้อมูล KeyCloak

ตารางที่ 3.7 ตาราง user_entity ใน KeyCloak

Field	Key	Datatype	Description
id	PK	varchar (36)	เลขรหัสของผู้ใช้งาน KeyCloak
email		varchar (255)	อีเมลของผู้ใช้งาน
email verified		bool	การยืนยันอีเมล
enabled		bool	การเปิดใช้งานของบัญชี
first_name		varchar (255)	ชื่อจริงผู้ใช้งาน
last_name		varchar (255)	นามสกุลผู้ใช้งาน
username		varchar (255)	ชื่อผู้ใช้งาน
created_timestamp		varchar (255)	วันที่และเวลาที่สร้างผู้ใช้งาน

ตารางที่ 3.8 ตาราง credential ใน KeyCloak

Field	Key	Datatype	Description
id	PK	varchar (36)	เลขรหัสของ Credential
user_id		varchar (36)	เลขรหัสของผู้ใช้งาน KeyCloak
salt		byteA	Random Salt ใช้เพื่อเข้ารหัสของรหัสผ่าน
type		varchar (255)	ประเภทของ Credential
value		varchar (4000)	ค่าของ Credential
created_date		int8	วันที่และเวลาที่สร้าง Credential
algorithm		varchar (36)	อัลกอริทึมที่เข้ารหัส ใช้ pbkdf2-sha256

ตารางที่ 3.9 ตาราง user_attribute ใน KeyCloak

Field	Key	Datatype	Description
id	PK	varchar (36)	เลขรหัสของผู้ใช้งาน Attribute
user_id		varchar (36)	เลขรหัสของผู้ใช้งาน KeyCloak
name		varchar (255)	ชื่อ Key ของ Attribute
value		varchar (255)	ชื่อ Value ของ Attribute

ตารางที่ 3.10 ตาราง user_role_mapping ใน KeyCloak

Field	Key	Datatype	Description
role_id	PK	varchar (255)	เลขรหัสของ Role
user_id	PK	varchar (36)	เลขรหัสของผู้ใช้งาน

ตารางที่ 3.11 ตาราง keycloak_role ใน KeyCloak

Field	Key	Datatype	Description
id	PK	varchar (36)	เลขรหัสของ Role
description		varchar (255)	รายละเอียดของ Role
name		varchar (255)	รายละเอียดของ Role

3.3.3.3 ตารางในฐานข้อมูล API

ตารางที่ 3.12 ตาราง feedback ใน API Database

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของความเห็น
email		text	อีเมลของผู้ให้ความเห็น
category		text	หมวดหมู่ของความเห็น
text		text	ข้อความของความเห็น
created_date		timestamptz	วันที่และเวลาเมื่อสร้างความเห็น
created_by		text	ผู้สร้างความเห็น
last_modified_date		timestamptz	วันที่และเวลาเมื่อแก้ไขความเห็น
last_modified_by		text	ผู้แก้ไขความเห็น

ตารางที่ 3.13 ตาราง answers ใน API Database

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของคำตอบ
question_id	FK	text	เลขรหัสของคำถาม
message		text	ข้อความของคำตอบ
created_date		timestamptz	วันที่และเวลาเมื่อสร้างคำตอบ
created_by		text	ผู้สร้างคำตอบ
last_modified_date		timestamptz	วันที่และเวลาเมื่อแก้ไขคำตอบ
last_modified_by		text	ผู้แก้ไขคำตอบ

ตารางที่ 3.14 ตาราง questions ใน API Database

Field	Key	Datatype	Description
id	PK	text	เลขรหัสของคำถาม
message		text	ข้อความของคำถาม
created_date		timestamptz	วันที่และเวลาเมื่อสร้างคำถาม
created_by		text	ผู้สร้างคำถาม
last_modified_date		timestamptz	วันที่และเวลาเมื่อแก้ไขคำถาม
last_modified_by		text	ผู้แก้ไขคำถาม

3.3.4 API Design Document

ตารางที่ 3.15 API การเข้าสู่ระบบ

HTTP Request Method	POST
URL	/auth/realms/{realm}/protocol/openid-connect/token
Parameters	grant_type - ประเภทของการ login เช่น authorization_code หรือ password client_id - ID ของ application ที่ต้องการใช้งาน client_secret - secret ของ application ที่ต้องการใช้งาน username - username ของผู้ใช้งาน password - password ของผู้ใช้งาน scope - scope ของผู้ใช้งานที่ application ต้องการเข้าถึง
Path Parameter	realm - ชื่อ Realm
Response	access_token - ใช้สำหรับเข้าใช้งาน Service อื่นๆในระบบ

ตารางที่ 3.16 API การออกจากระบบ

HTTP Request Method	POST
URL	/auth/realms/{realm}/protocol/openid-connect/logout
Request Header	Authorization - access_token ของผู้ใช้งานที่ต้องการ logout
Path Parameter	realm - ชื่อ Realm
Response	success

ตารางที่ 3.17 API การรีเซ็ตรหัสผ่าน

HTTP Request Method	PUT
URL	/auth/realms/{realm}/{id}/reset-password
Path Parameter	realm - ชื่อ Realm id - ID ของผู้ใช้งานที่ต้องการรีเซ็ต
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://www.KeyCloak.org/docs-api/6.0/rest-api/index.html#_credentialrepresentation
Response	success

ตารางที่ 3.18 API การลงทะเบียนผู้ใช้งาน

HTTP Request Method	POST
URL	/api/users/register
Request Body	fullName - ชื่อเต็มของผู้ใช้งาน firstName - ชื่อของผู้ใช้งาน lastName - นามสกุลของผู้ใช้งาน email - email ของผู้ใช้งาน password - password ของผู้ใช้งาน citizenId - รหัสบัตรประชาชน telephoneNumber - หมายเลขโทรศัพท์ของผู้ใช้งาน organization - หน่วยงานของผู้ใช้งาน department – แผนกของผู้ใช้งาน
Response	userId - ID ของผู้ใช้งาน

ตารางที่ 3.19 API การตั้งค่ารหัสผ่านของผู้ใช้งาน

HTTP Request Method	PUT
URL	/auth/realms/{realm}/users/{id}/reset-password
Path Parameter	realm - ชื่อ Realm id - ID ของผู้ใช้งานที่ต้องการรีเซ็ต
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://www.KeyCloak.org/docs-api/6.0/rest-api/index.html#_credentialrepresentation

ตารางที่ 3.20 API การค้นหาข้อมูลผู้ใช้งาน

HTTP Request Method	GET
URL	/auth/{realm}/dga/users
Request Parameters	search – ข้อความที่ต้องการค้นหาใน username, ชื่อ, นามสกุล หรือ email
Request Parameters	client_id – ID ของ application redirect_uri – URI ที่ต้องการให้ redirect

ตารางที่ 3.21 API การเพิ่มข้อมูลผู้ใช้งาน

HTTP Request Method	POST
URL	/auth/realms/dga/users
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://www.KeyCloak.org/docs-api/6.0/rest-api/index.html#_userrepresentation

ตารางที่ 3.22 API การเพิ่มข้อมูลผู้ใช้งาน

HTTP Request Method	PUT
URL	/auth/realms/dga/users/{id}
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการปรับปรุง
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://www.KeyCloak.org/docs-api/6.0/rest-api/index.html#_userrepresentation

ตารางที่ 3.23 API การลบข้อมูลผู้ใช้งาน

HTTP Request Method	DELETE
URL	/auth/realms/dga/users/{id}
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการลบ

ตารางที่ 3.24 API การอนุมัติผู้ใช้งาน

HTTP Request Method	PUT
URL	/api/users/{id}/approve
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการอนุมัติการลงทะเบียน

ตารางที่ 3.25 API การ Deactivate ผู้ใช้งาน

HTTP Request Method	PUT
URL	/api/users/{id}/deactivate
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการ deactivate

ตารางที่ 3.26 API การอนุมัติ Deactivate ผู้ใช้งาน

HTTP Request Method	PUT
URL	/api/users/{id}/approve-deactivate
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการอนุมัติการ deactivate

ตารางที่ 3.27 API การกู้คืนผู้ใช้งาน

HTTP Request Method	PUT
URL	/api/users/{id}/recover
Path Parameter	id – ID ของผู้ใช้งานที่ต้องการกู้คืน

ตารางที่ 3.28 API การกำหนดกลุ่มผู้ใช้งานได้

HTTP Request Method	POST
URL	/api/3/action/group_member_create
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.group_member_create

ตารางที่ 3.29 API การกำหนดหน่วยงานผู้ใช้งาน

HTTP Request Method	POST
URL	/api/3/action/organization_member_create
Request Body	โครงสร้างของ request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.organization_member_create

ตารางที่ 3.30 API การจัดการมองเห็นของชุดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/package_patch
Request Body	id – ID ของ dataset ที่ต้องการกำหนดสิทธิ์ private – True เมื่อต้องการให้ dataset เป็น private, False เมื่อต้องการให้ dataset เป็น public allowed_users – ชื่อผู้ใช้งานที่ต้องการเข้าถึง dataset นี้

ตารางที่ 3.31 API การแสดงรายการชุดข้อมูล

HTTP Request Method	GET
URL	/api/3/action/package_list
Request Parameter	Request parameter อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.package_list

ตารางที่ 3.32 API การการแสดงรายละเอียดของชุดข้อมูล

HTTP Request Method	GET
URL	/api/3/action/package_show
Request Parameter	Request parameter อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.package_show

ตารางที่ 3.33 API การการลบชุดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/package_delete
Parameters	id - ชื่อ หรือ ID ของ dataset ที่ต้องการลบ

ตารางที่ 3.34 API การดาวน์โหลดชุดข้อมูล

HTTP Request Method	GET
URL	/dataset/{dataset-id}/resource/{resource-id}/download
Path Parameter	dataset-id – ID ของ dataset resource-id – ID ของ resource

ตารางที่ 3.35 API การสร้างชุดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/package_create
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.package_create

ตารางที่ 3.36 API การนำเข้าชุดข้อมูลโดยใช้แฟ้มข้อมูล

HTTP Request Method	POST
URL	/api/3/action/resource_create
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.resource_create

ตารางที่ 3.37 API การแสดงเมตาดาต้า

HTTP Request Method	GET
URL	/api/dataset/{id}/metadata
Path Parameters	id – ID ของ dataset ที่ต้องการแสดงข้อมูล metadata

ตารางที่ 3.38 API การปรับปรุงเมตาดาต้า

HTTP Request Method	DELETE
URL	/api/dataset/{id}/metadata
Path Parameters	id – ID ของ dataset ที่ต้องการปรับปรุง metadata
Request Parameter	file – file ที่มีเนื้อหา metadata

ตารางที่ 3.39 API การการเพิ่มข้อมูลหน่วยงาน

HTTP Request Method	POST
URL	/api/3/action/organization_create
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.organization_create

ตารางที่ 3.40 API การปรับปรุงข้อมูลหน่วยงาน

HTTP Request Method	POST
URL	/api/3/action/organization_update
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.update.organization_update

ตารางที่ 3.41 API การลบข้อมูลหน่วยงาน

HTTP Request Method	POST
URL	/api/3/action/organization_delete
Parameters	id - ชื่อ หรือ ID ของหน่วยงานที่ต้องการลบ

ตารางที่ 3.42 API การค้นหาข้อมูลหน่วยงาน

HTTP Request Method	GET
URL	/api/3/action/organization_list
Parameters	Parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.organization_list

ตารางที่ 3.43 API การเพิ่มหมวดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/group_create
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.create.group_create

ตารางที่ 3.44 API การปรับปรุงหมวดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/group_update
Request Body	Request body อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.update.group_update

ตารางที่ 3.45 API การลบหมวดข้อมูล

HTTP Request Method	POST
URL	/api/3/action/group_delete
Parameters	**id** – ID ของหมวดหมู่ที่ต้องการลบ

ตารางที่ 3.46 API การค้นหาหมวดข้อมูล

HTTP Request Method	GET
URL	/api/3/action/group_list
Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.group_list

ตารางที่ 3.47 API การแสดงประวัติตามหน่วยงาน

HTTP Request Method	GET
URL	/api/3/action/organization_activity_list
Request Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.organization_activity_list

ตารางที่ 3.48 API การแสดงประวัติตามหมวดหมู่

HTTP Request Method	GET
URL	/api/3/action/group_activity_list
Request Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.group_activity_list

ตารางที่ 3.49 API การแสดงประวัติตาม Dataset

HTTP Request Method	GET
URL	/api/3/action/package_activity_list
Request Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.package_activity_list

ตารางที่ 3.50 API การแสดงประวัติตามผู้ใช้งาน

HTTP Request Method	GET
URL	/api/3/action/user_activity_list
Request Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.user_activity_list

ตารางที่ 3.51 API การเพิ่มความคิดเห็น

HTTP Request Method	POST
URL	/api/feedback
Request Body	email – email ที่ต้องการให้ติดต่อกลับ category – หมวดหมู่ของความคิดเห็น text - ความคิดเห็นที่ต้องการเพิ่ม

ตารางที่ 3.52 API การปรับปรุงความคิดเห็น

HTTP Request Method	PUT
URL	/api/feedback/{id}
Path Parameter	id – ID ของความคิดเห็นที่ต้องการปรับปรุง
Request Body	email – email ที่ต้องการให้ติดต่อกลับ category – หมวดหมู่ของความคิดเห็น text - ความคิดเห็นที่ต้องการเพิ่ม

ตารางที่ 3.53 API การลบความคิดเห็น

HTTP Request Method	DELETE
URL	/api/feedback/{id}
Path Parameter	id – ID ของความคิดเห็นที่ต้องการลบ

ตารางที่ 3.54 API การเพิ่มคำถาม

HTTP Request Method	POST
URL	/api/questions
Request Body	message - ข้อความของคำถาม

ตารางที่ 3.55 API การเพิ่มคำตอบ

HTTP Request Method	POST
URL	/api/questions/{id}/answers
Path Parameter	id – ID ของคำถามที่ต้องการเพิ่มคำตอบ
Request Body	message – ข้อความของคำตอบ

ตารางที่ 3.56 API การแก้ไขคำถาม

HTTP Request Method	PUT
URL	/api/questions/{id}
Path Parameter	id – ID ของคำถามที่ต้องการปรับปรุง
Request Body	message – ข้อความของคำถามที่ต้องการปรับปรุง

ตารางที่ 3.57 API การแก้ไขคำตอบ

HTTP Request Method	POST
URL	/api/questions/{questionId}/answers/{answerId}
Path Parameter	questionId – ID ของคำถาม answerId – ID ของคำตอบ
Request Body	message – ข้อความของคำตอบที่ต้องการปรับปรุง

ตารางที่ 3.58 API การลบคำถาม

HTTP Request Method	DELETE
URL	/api/questions/{id}
Path Parameter	id – ID ของคำถามที่ต้องการลบ
เพิ่มเติม	ลบคำตอบที่เกี่ยวข้องภายใต้คำถามนั้นออกทั้งหมด

ตารางที่ 3.59 API การแสดงคำถาม

HTTP Request Method	GET
URL	/api/questions

ตารางที่ 3.60 API การลบคำตอบ

HTTP Request Method	DELETE
URL	/api/questions/{questionId}/answers/{answerId}
Path Parameter	questionId – ID ของคำถาม answerId – ID ของคำตอบที่ต้องการลบ

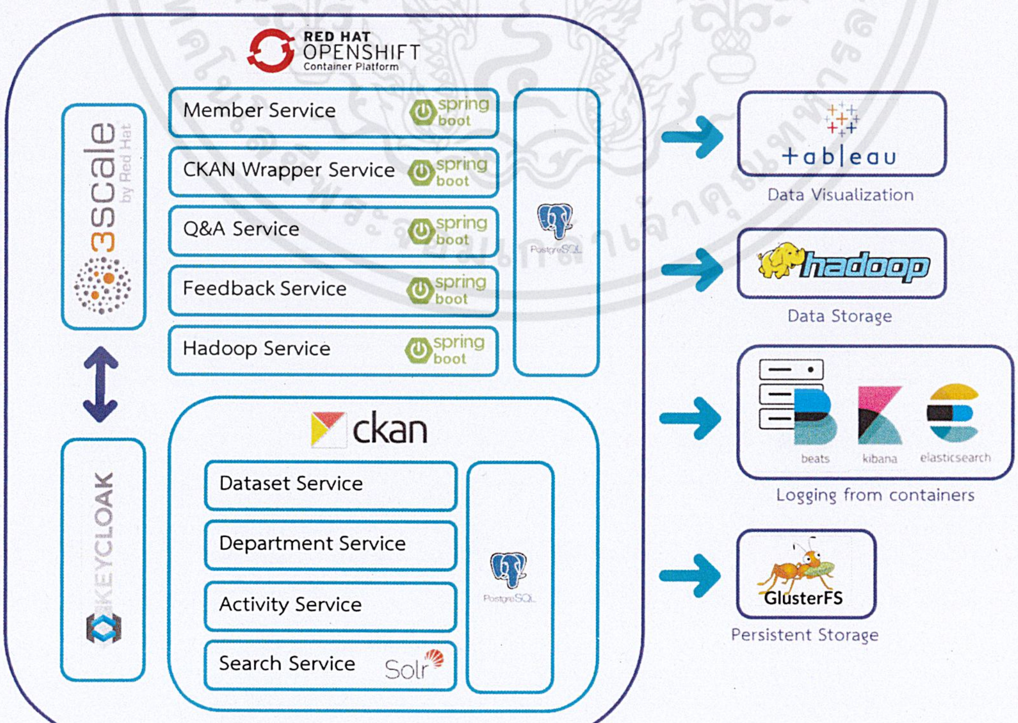
ตารางที่ 3.61 API การแสดงคำตอบ

HTTP Request Method	GET
URL	/api/questions/{id}/answers
Path Parameter	id – ID ของคำถามที่ต้องการแสดงคำตอบ

ตารางที่ 3.62 API การค้นหาข้อมูล ชื่อชุดข้อมูล ชื่อแฟ้มข้อมูล

HTTP Request Method	GET
URL	/api/3/action/package_search
Request Parameters	Request parameters อ้างอิงได้ที่ https://docs.ckan.org/en/2.8/api/index.html# ckan.logic.action.get.package_search

3.4 ภาพรวมของระบบการควบคุมและจัดการข้อมูล



ภาพที่ 3.25 ภาพรวมของระบบการควบคุมและจัดการข้อมูล

ระบบควบคุมและจัดการข้อมูลแบ่งออกเป็น 3 ส่วน คือ ส่วน Front-End ส่วน Back-End และส่วนของระบบการทำรายงานวิเคราะห์ข้อมูลจากชุดข้อมูล ซึ่งทางผู้จัดทำได้รับผิดชอบในส่วนของระบบ Back-End ซึ่งจะประกอบไปด้วยการออกแบบและพัฒนา API ทั้งหมดรวมถึงการวางโครงสร้างระบบ

ระบบจะถูกออกแบบเป็น Microservices ซึ่งจะแยกการทำงานของ API ออกจากกัน โดยสามารถเข้าใช้งาน API ได้ผ่านโปรโตคอล HTTP เมื่อมี Request เข้ามาในระบบจะผ่าน 3Scale API Gateway ซึ่งทำหน้าที่กรอง Traffic ต่างๆที่เข้ามาในระบบ เพิ่มความปลอดภัยในการเข้าถึง API ทั้งหมดด้วย Endpoint เดียว โดยสามารถแบ่งแยกการเข้าถึงแต่ละ Service ผ่าน Context Path โดย NGINX จะทำหน้าที่เป็น Router เลือกเส้นทางไปยัง API ต่างๆที่กำหนดไว้ โดย API ต่างๆจะทำงานอยู่บน OpenShift Container Platform ซึ่งจะมีการควบคุมการทำงานของ Pod ให้เป็นไปตามที่กำหนดตลอดเวลา เมื่อ CPU ถึงระดับที่ตั้งค่าไว้จะมีการ Scale Pod ขึ้นมาตามจำนวนที่ตั้งไว้

API Services เป็นส่วนงานที่ต้องพัฒนาขึ้นมาใหม่เพื่อให้ตรงกับความต้องการ โดยทีการใช้ภาษา Java พัฒนา Software ทั้งหมด โดยมีการใช้ Spring Boot Framework มาเป็น Framework หลักในการพัฒนา เมื่อการพัฒนาเสร็จสิ้น ตัว Application นั้นจะทำงานในรูปแบบของ Container Service ที่ถูกติดตั้งและ Deploy อยู่ใน OpenShift Container Platform

3.4.1 Member Service

ใช้สำหรับจัดการสมาชิกทั้งหมดในระบบเป็นการเชื่อมต่อระบบสมาชิกของ CKAN และ Tableau และใช้ KeyCloak เป็นตัวกลางในการจัดการสมาชิก

3.4.2 CKAN Wrapper Service

ใช้สำหรับเป็นตัวควบคุมความปลอดภัยในการเข้าถึง API ใน CKAN โดยจะนำ User ID ของ CKAN ไปค้นหา API Key ที่อยู่ในฐานข้อมูลของ CKAN หลังจากได้ API Key มาตัว Service จะส่ง Request ต่อไปให้ API ของ CKAN ต่อ

3.4.3 Q&A Service

ใช้สำหรับจัดการเกี่ยวกับคำถาม-คำตอบ

3.4.4 Feedback Service

ใช้สำหรับจัดการความเห็น

3.4.5 CKAN Services

CKAN มี API ที่ครอบคลุมตามความต้องการ จึงนำระบบ CKAN ซึ่งเป็นซอฟต์แวร์ที่มีความสามารถสำหรับด้านการจัดการข้อมูลโดยเฉพาะ มีหลายหน่วยงานในต่างประเทศเลือกใช้ มีความน่าเชื่อถือสูง รวมถึงระบบของ CKAN มีการเปิดให้เข้าใช้งาน API ได้ และตรงกับความต้องการของระบบ โดย CKAN จะถูกติดตั้งบน OpenShift และมี CKAN Wrapper Service มาครอบการทำงานของ CKAN อีกครั้ง เพื่อตรวจสอบความปลอดภัย

3.4.5.1 Dataset Service

ใช้สำหรับจัดการชุดข้อมูลและจัดการ Resource ต่างๆในชุดข้อมูล

3.4.5.2 Department Service

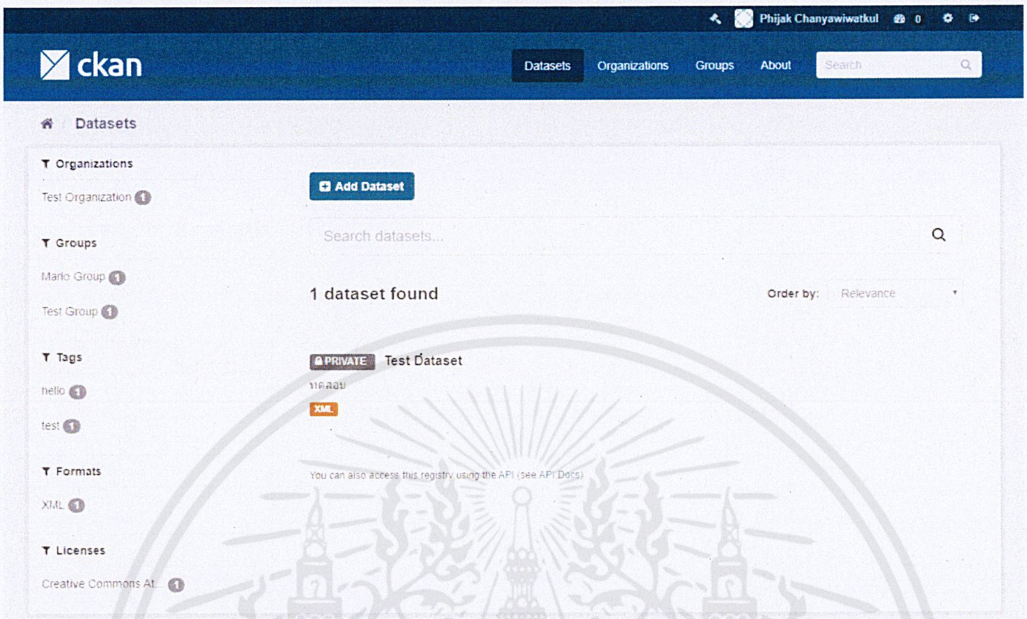
ใช้สำหรับจัดการหน่วยงานและสามารถจัดการกับชุดข้อมูลที่เกี่ยวข้องกับหน่วยงาน

3.4.5.3 Activity Service

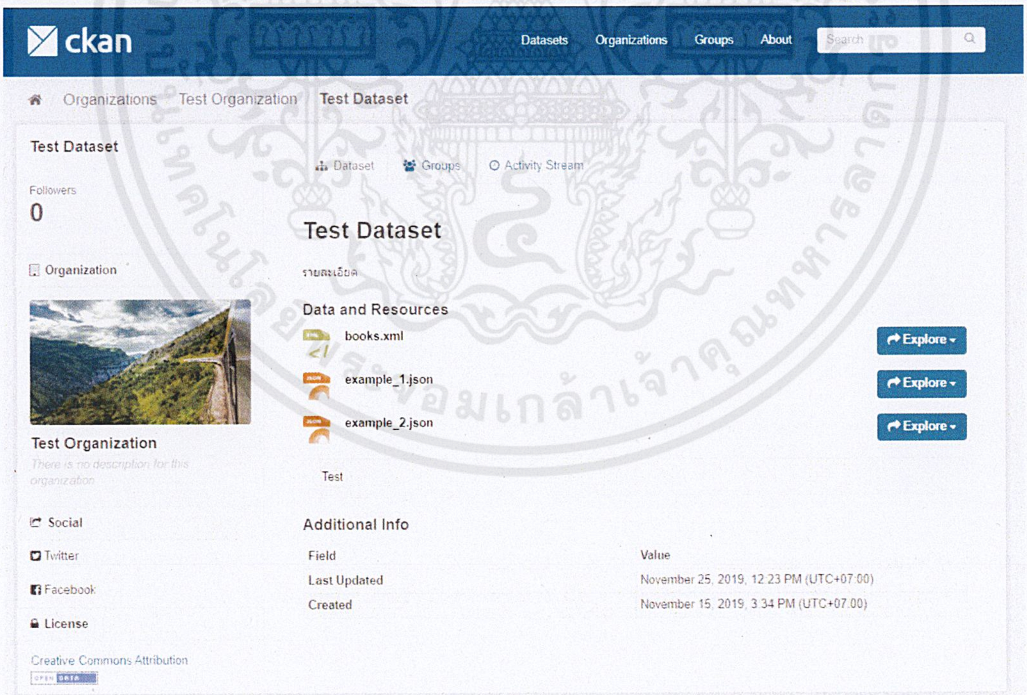
ใช้สำหรับจัดการกิจกรรมที่เกิดขึ้นในระบบต่างๆ

3.4.5.4 Search Service

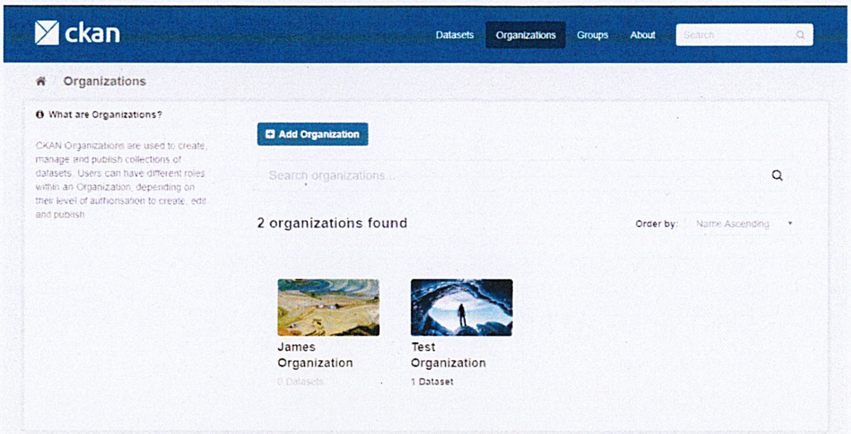
ใช้สำหรับค้นหาข้อมูลต่างๆ เช่น ชุดข้อมูล, Resource ของชุดข้อมูล, ผู้ใช้งาน



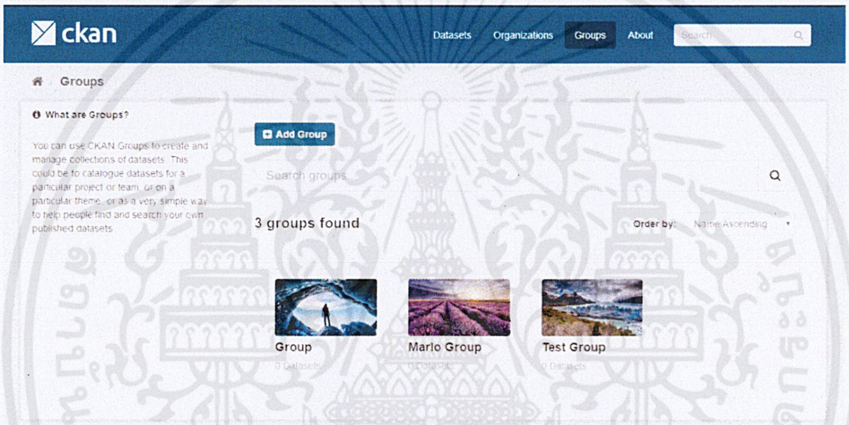
ภาพที่ 3.26 Front-End ของ CKAN ส่วนของ Dataset



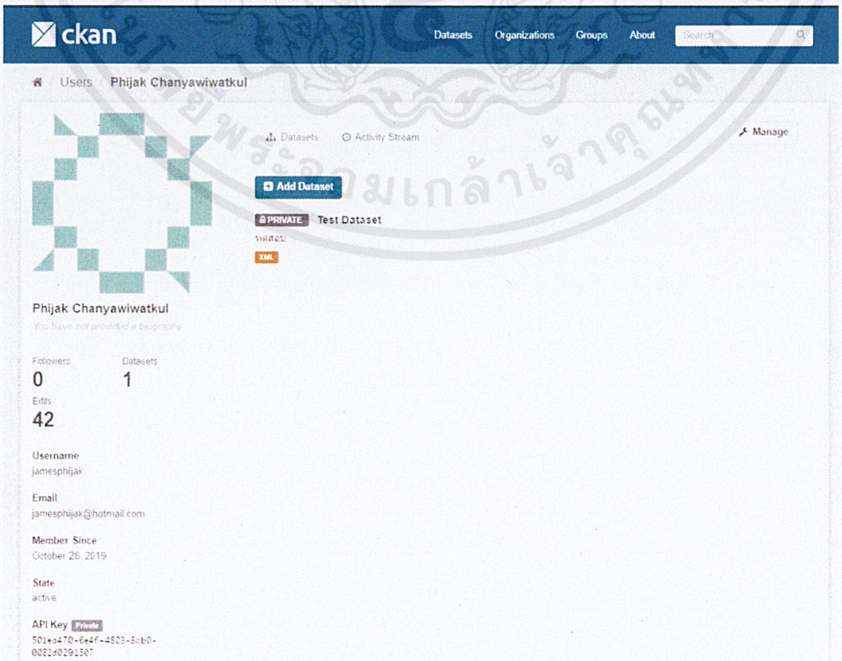
ภาพที่ 3.27 Front-End ของ CKAN ส่วนของ Resource ใน Dataset



ภาพที่ 3.28 Front-End ของ CKAN ส่วนของ Organization



ภาพที่ 3.29 Front-End ของ CKAN ส่วนของ Groups

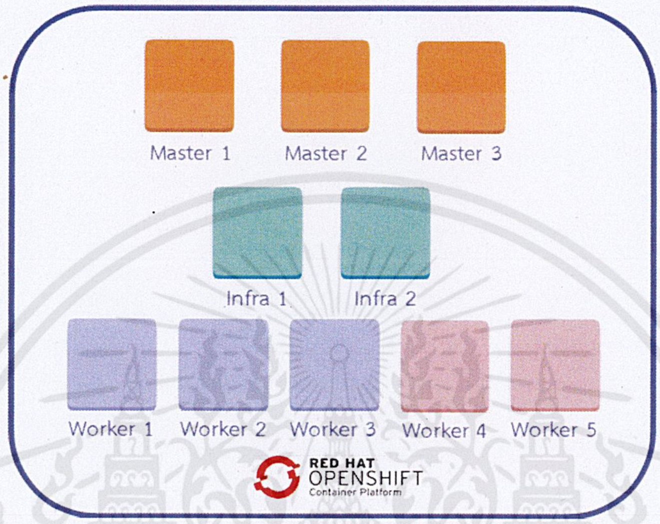


ภาพที่ 3.30 Front-End ของ CKAN ส่วนของ User

3.4.6 OpenShift Container Platform

OpenShift Container Platform ติดตั้งเป็น Version 3.11 โดยที่ OpenShift Container Platform นั้น และถูกติดตั้งบน OS ของ RedHat Enterprise Linux version 7.4

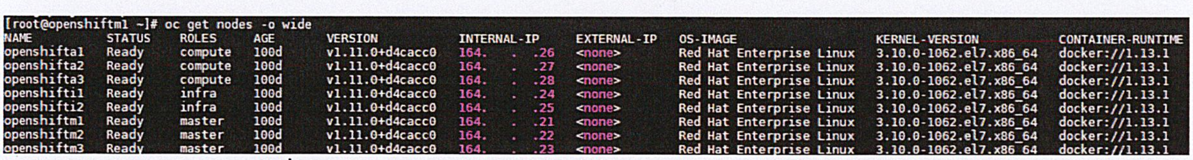
โดยการ Cluster ของ OpenShift นั้นจะติดตั้งทั้งหมด 10 Nodes โดยจะเป็น Master 3 Nodes ส่วนที่เหลือจะเป็น Compute Node (2 Infra Nodes + 5 Worker Nodes)



ภาพที่ 3.31 Cluster Nodes ของ OpenShift

ตารางที่ 3.63 IP Address และ Hostname ที่ติดตั้ง OpenShift

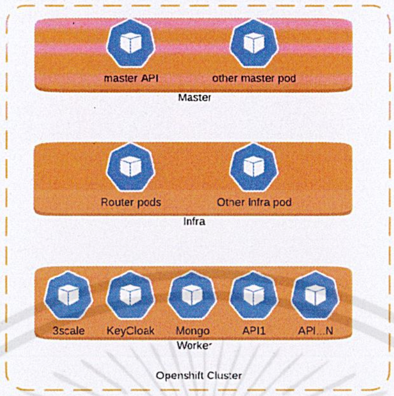
IP Address	Hostname	Node
164.xxx.xx.21	openshiftm1	master
164.xxx.xx.22	openshiftm2	master
164.xxx.xx.23	openshiftm3	master
164.xxx.xx.24	openshifti1	infra
164.xxx.xx.25	openshifti2	infra
164.xxx.xx.26	openshifta1	worker
164.xxx.xx.27	openshifta2	worker
164.xxx.xx.28	openshifta3	worker
Reserved IP	openshifta4	worker
Reserved IP	openshifta5	worker



```
[root@openshiftm1 ~]# oc get nodes -o wide
NAME         STATUS   ROLES     AGE    VERSION            INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                    KERNEL-VERSION             CONTAINER-RUNTIME
openshifta1  Ready    compute   100d   v1.11.0+d4cacc0    164.  .  .26  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshifta2  Ready    compute   100d   v1.11.0+d4cacc0    164.  .  .27  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshifta3  Ready    compute   100d   v1.11.0+d4cacc0    164.  .  .28  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshifti1  Ready    infra     100d   v1.11.0+d4cacc0    164.  .  .24  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshifti2  Ready    infra     100d   v1.11.0+d4cacc0    164.  .  .25  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshiftm1  Ready    master    100d   v1.11.0+d4cacc0    164.  .  .21  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshiftm2  Ready    master    100d   v1.11.0+d4cacc0    164.  .  .22  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
openshiftm3  Ready    master    100d   v1.11.0+d4cacc0    164.  .  .23  <none>        Red Hat Enterprise Linux    3.10.0-1062.el7.x86_64     docker://1.13.1
```

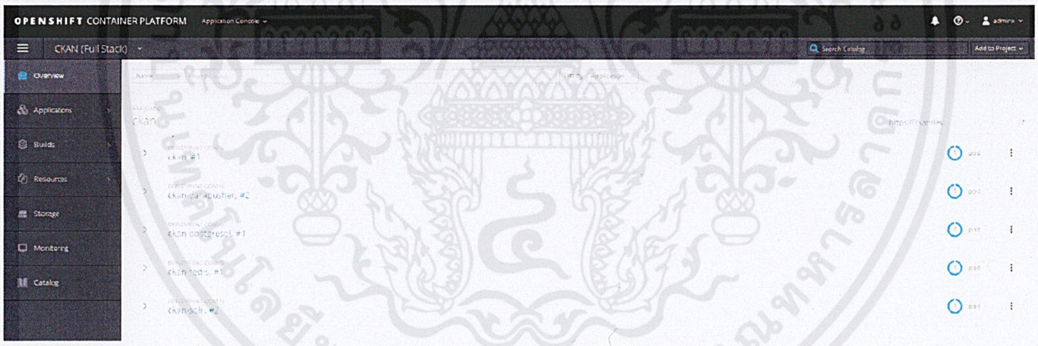
ภาพที่ 3.32 ตรวจสอบสถานะ Cluster Nodes บน OpenShift

ภายใต้ OpenShift Container Platform นั้นการทำงานต่างๆ จะถูกกระจายการทำงานออกไปให้ทุก Node ใน Cluster ซึ่งการกระจายการทำงานนั้นขึ้นอยู่กับหน้าที่ของ Node และ Pod นั้นจะต้องสัมพันธ์กัน

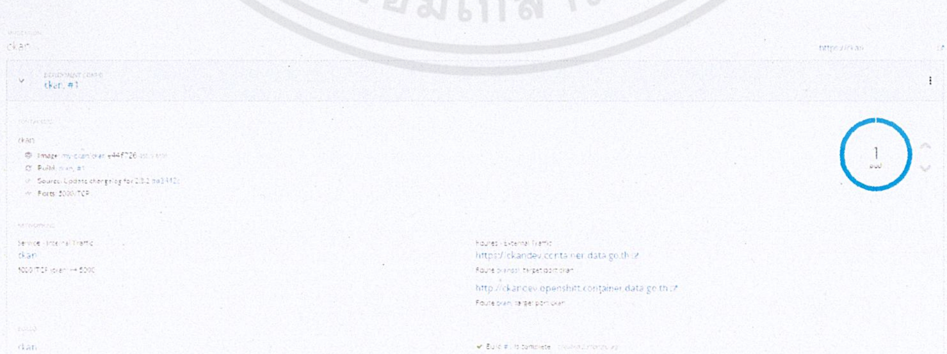


ภาพที่ 3.33 การกระจายตัวของ Pod บน Cluster ของ OpenShift

การติดตั้ง OpenShift Cluster ในรูปแบบ Architecture จะทำให้ OpenShift Cluster นั้นมีความน่าเชื่อถือที่สูงขึ้น เพราะว่าการทำงานของ OpenShift Cluster นั้นสามารถถูกกระจายการทำงานไปยัง Nodes อื่นๆ ได้ ในกรณีที่ Node ที่รองรับการทำงานปัจจุบันนั้น Down ลงไป



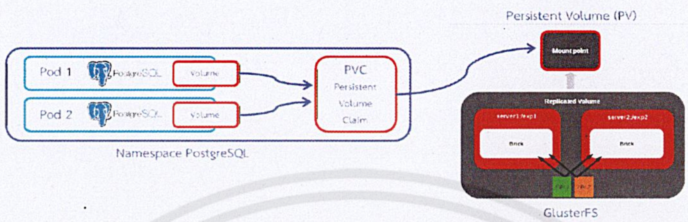
ภาพที่ 3.34 OpenShift Container Platform Overview



ภาพที่ 3.35 OpenShift Container Platform Application Monitor

3.4.7 GlusterFS Software Defined Storage

GlusterFS เป็นส่วนที่ต้องทำงานร่วมกับ OpenShift Container Platform โดยที่ GlusterFS นั้นจะใช้เก็บ Data ทั้งในส่วนของ System Data และ User Data ทั้งหมดของระบบ โดยตัว GlusterFS นั้นถูกติดตั้งใน Mode ของ Replica 2 ซึ่งการทำงานในรูปแบบนี้ System Data และ User Data จะถูกทำคัดลอกไว้ 1 ชุดเสมอ เพื่อให้ข้อมูลมีเสถียรภาพมากขึ้น

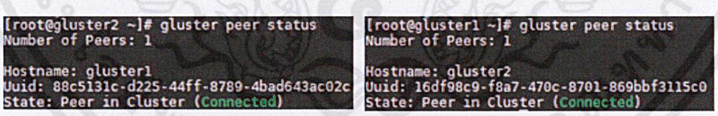


ภาพที่ 3.36 การทำงานของ GlusterFS แบบ Replicated Volume กับ OpenShift

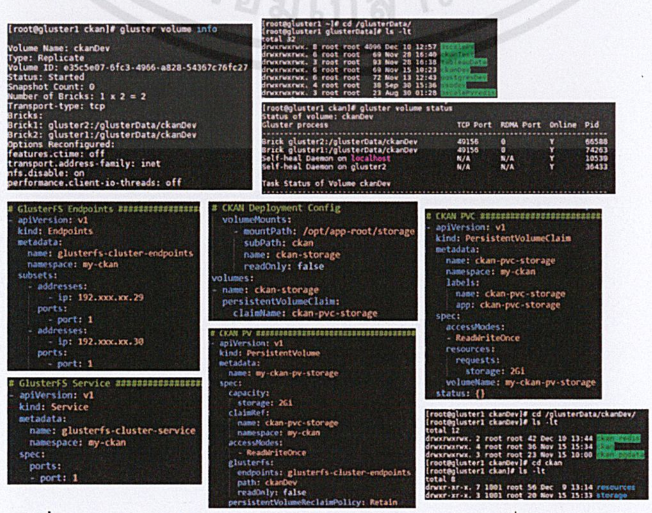
การทำงานร่วมกันกับ OpenShift Container Platform นั้นจะสร้าง TCP Network Connection มาทำการอ่านและเขียนข้อมูลที่อยู่บน GlusterFS เมื่อ GlusterFS ได้รับ Request การเขียนข้อมูลจากทางฝั่ง OpenShift Container Platform แล้วตัว GlusterFS จะทำการเขียนข้อมูลลงไปทั้ง 2 Bricks ซึ่งได้ถูกสร้างไว้ในแต่ละเครื่องของ GlusterFS Cluster

ตารางที่ 3.64 IP Address และ Hostname ที่ติดตั้ง GlusterFS

IP Address	Hostname	Host
164.xxx.xx.29	gluster1	GlusterFS Server 1
164.xxx.xx.30	gluster2	GlusterFS Server 2



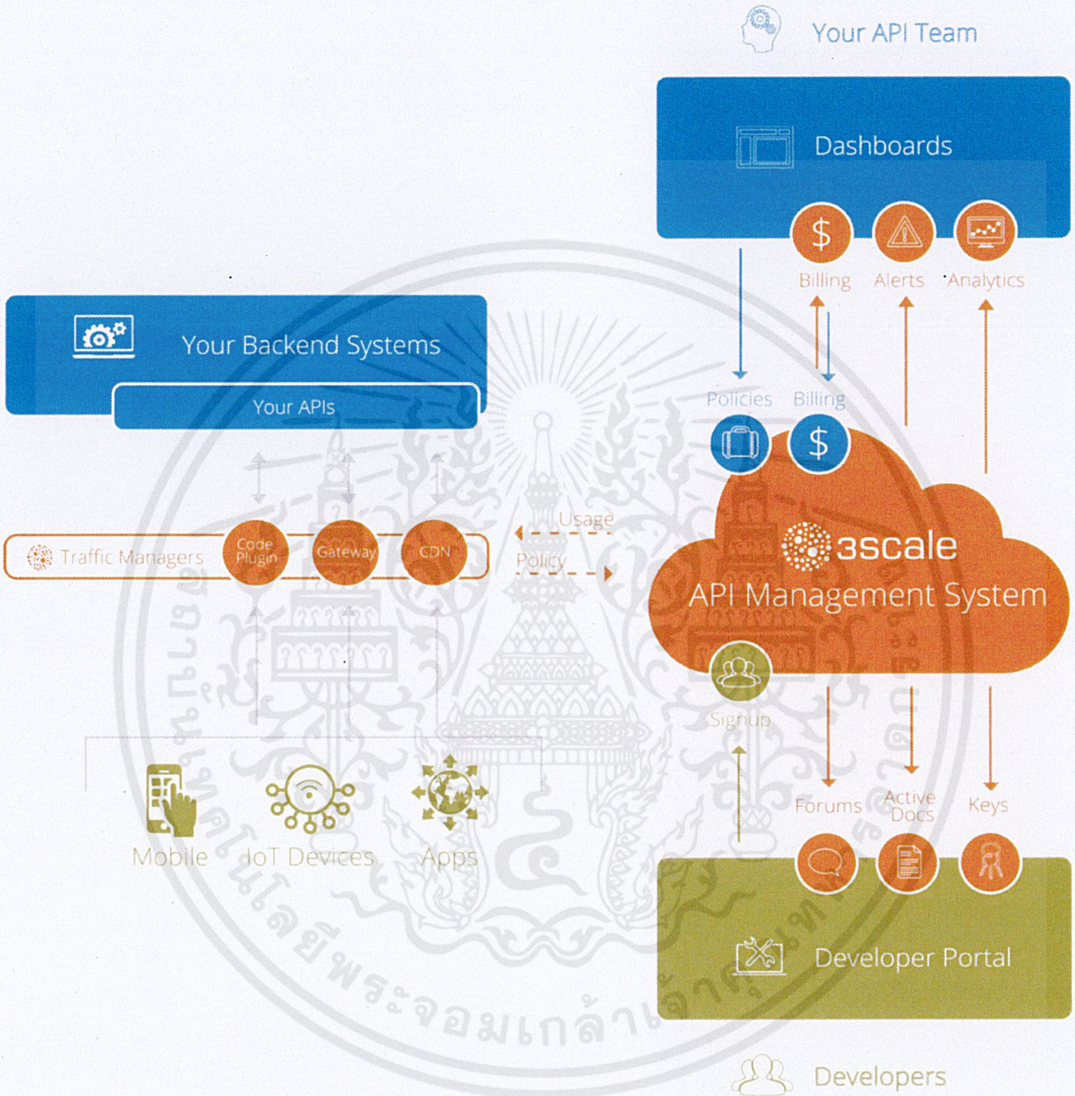
ภาพที่ 3.37 การเชื่อมต่อของ GlusterFS ทั้ง 2 เครื่อง



ภาพที่ 3.38 การสร้างและการใช้ Volume บนเครื่อง GlusterFS

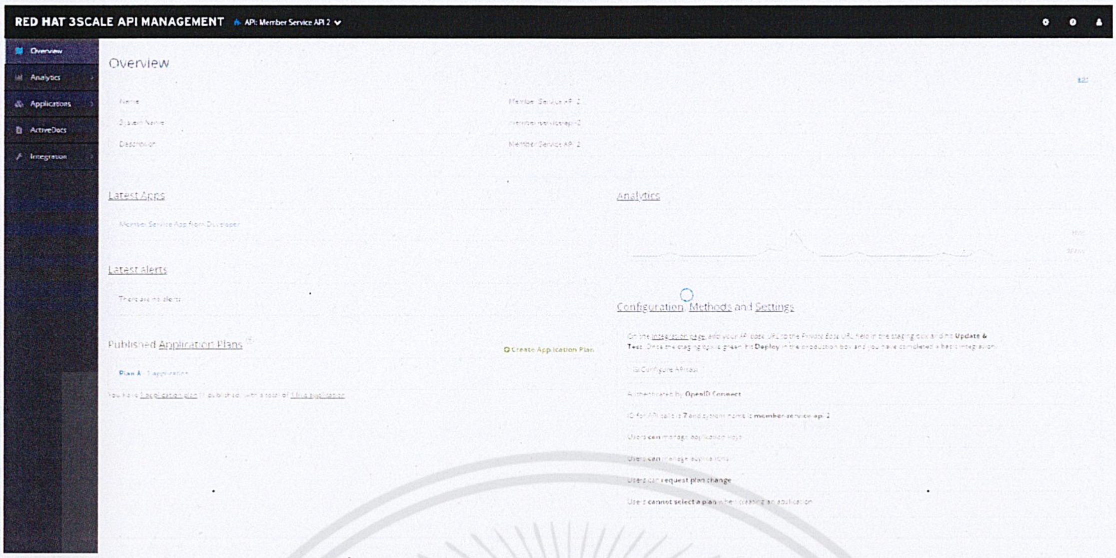
3.4.8 3Scale API Management Platform

3Scale API Management Platform นั้นได้ติดตั้งบน OpenShift Container Platform หน้าที่หลักของ 3scale API Management ทำหน้าที่เป็น Gateway สำหรับการเรียกใช้งาน API ทั้งหมดที่จะได้ถูกพัฒนาขึ้นมาใหม่ และยังทำหน้าที่เป็น Authentication service ด้วย

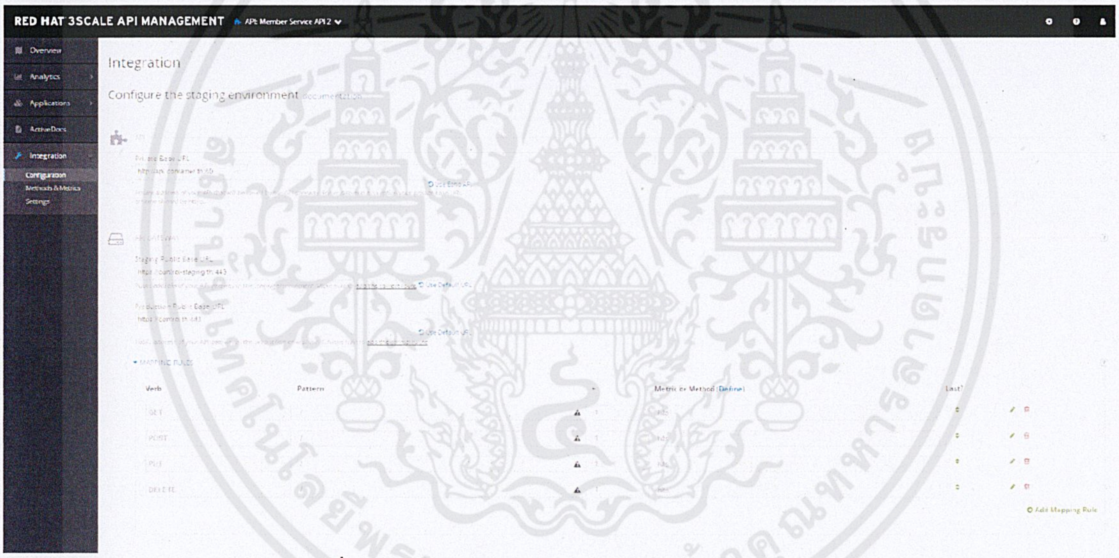


ภาพที่ 3.39 การทำงานของ 3Scale API Management

การทำงานภายใน 3scale ด้านบนนั้น เมื่อ 3scale API Management platform ได้รับ Request จะทำการส่ง API request นั้นไปหา Backend System เพื่อที่จะประมวลผล Request ต่อ



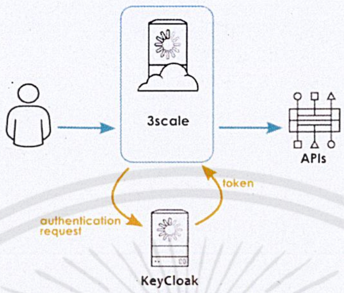
ภาพที่ 3.40 3Scale API Management Overview



ภาพที่ 3.41 3Scale API Management Integration

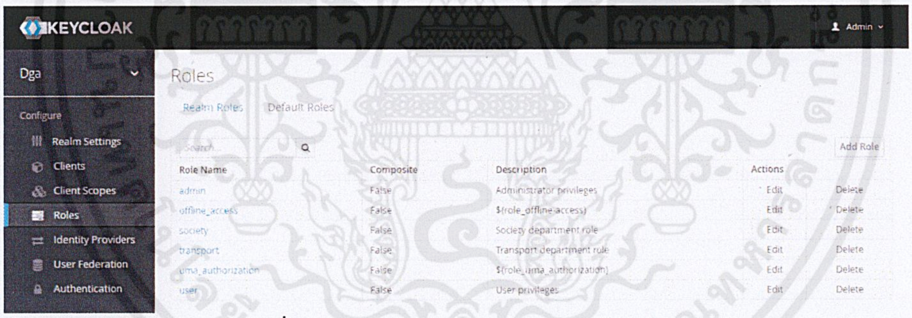
3.4.9 KeyCloak Identity and Access Management

มีหน้าที่รองรับการทำงานแบบ Single-Signed on โดยที่ User Information และ User Credential ทั้งหมด จะถูกเก็บรวบรวมไว้ที่ KeyCloak เมื่อมี Request API เข้ามา KeyCloak จะทำการ Authentication/Authorization และส่ง Role ที่เป็นของ User ที่เข้าสู่ระบบนั้นกลับไปในรูปแบบของ JWT Encrypted Token

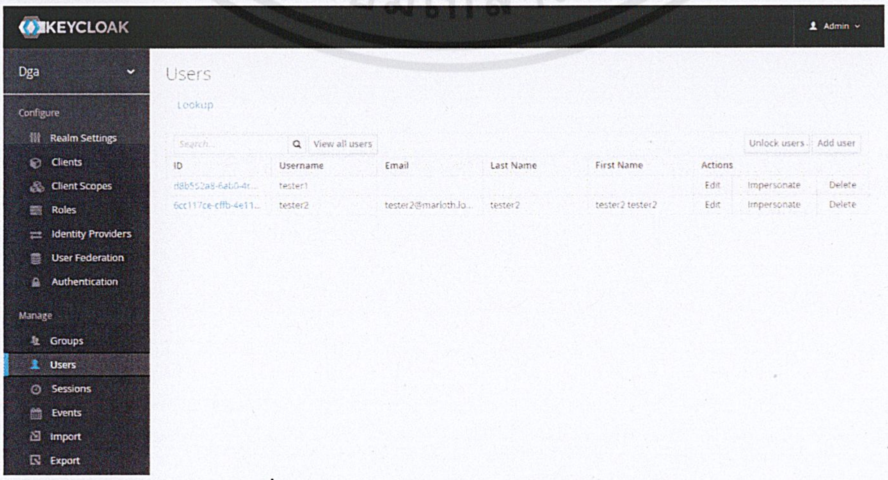


ภาพที่ 3.42 การทำงานของ KeyCloak

การทำงานของ KeyCloak จะ Integrated รวมกับ 3Scale API Management Platform เพื่อทำหน้าที่เป็น Authentication และ Authorization service ที่ built-in มากับ 3Scale โดยที่ KeyCloak นั้นจะถูกติดตั้งในรูปแบบของ Container บน OpenShift Container Platform



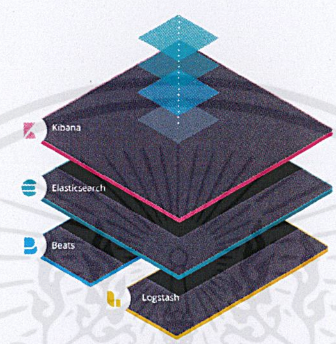
ภาพที่ 3.43 การจัดการ Role ใน KeyCloak



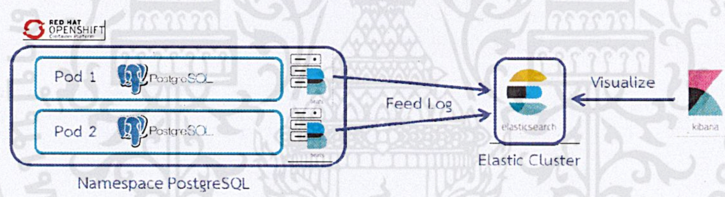
ภาพที่ 3.44 การจัดการ User ใน KeyCloak

3.4.10 Beats, Elastic Search และ Kibana

สำหรับ Elastic Search จะทำหน้าที่เป็น Log Centralization ของทุก API ที่จะถูกพัฒนาขึ้นมาใหม่ในภายในโครงการ โดยที่ Elastic Search นั้นจะถูกติดตั้งแยกออกไปจาก OpenShift Container Platform โดยการติดตั้งนั้นจะติดตั้งในรูปแบบของ Elastic Search Cluster ซึ่งจะรวมการทำงานของ 3 elastic node โดยที่ Log ของทุก API ที่ทำงานอยู่ในรูปแบบของ Container นั้นจะถูก Feed ออกมาจาก OpenShift Container Platform ด้วย Beats และถูกส่งมายัง Elastic search Cluster และทำเป็น Index เพื่อนำ Log นั้นไปหาสาเหตุของปัญหา และทำ Analytic ต่อได้บน Kibana



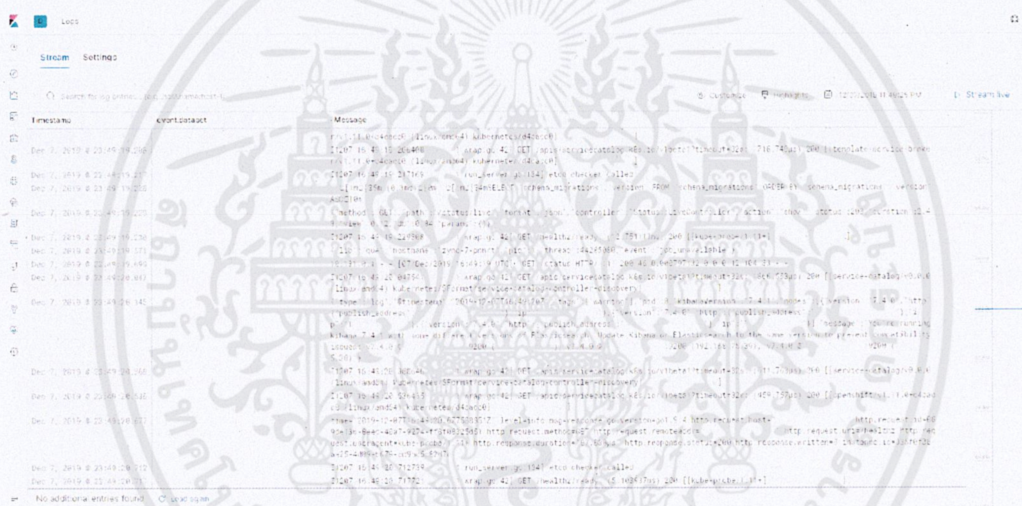
ภาพที่ 3.45 เทคโนโลยีของ Elastic Search



ภาพที่ 3.46 การทำงานของ Elastic Search ร่วมกับ OpenShift

ภาพที่ 3.47 Kibana Discovery Index

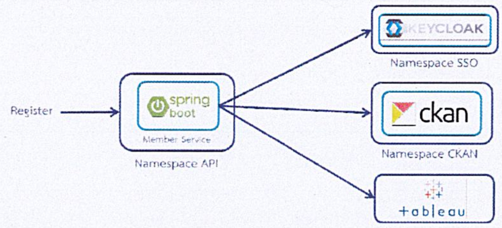
ภาพที่ 3.48 Kibana Dashboard



ภาพที่ 3.49 Kibana Stream Logs

ภาพที่ 3.50 Kibana Infrastructure Dashboard

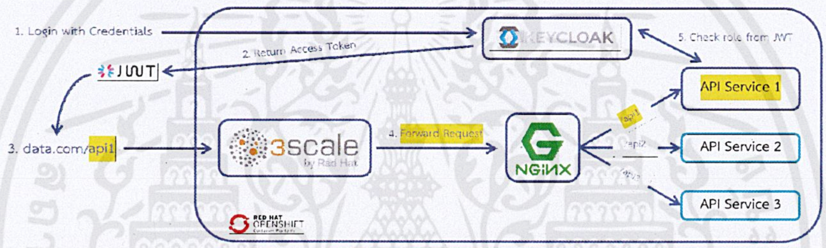
3.4.11 การสมัครสมาชิกด้วย Member Service



ภาพที่ 3.51 การสมัครสมาชิกด้วย Member Service

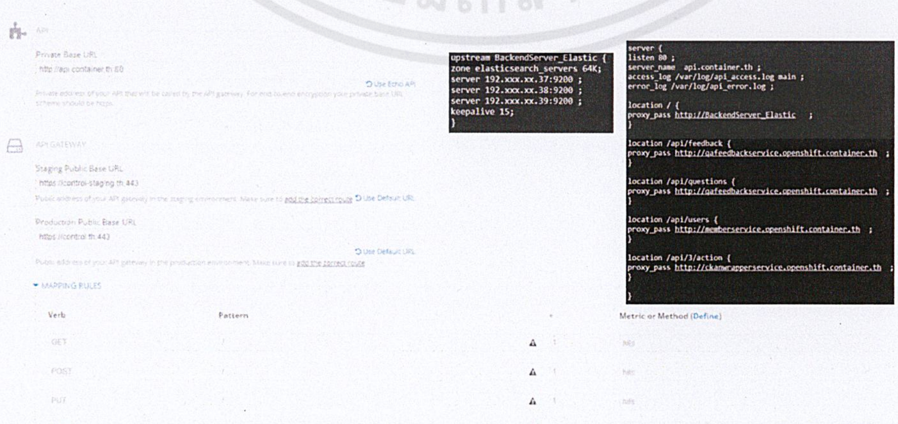
เมื่อมีการสมัครสมาชิกเข้ามาตัว Member Service จะทำการ Register ไปลง 3 ที่ คือ KeyCloak, CKAN และ Tableau โดยบน KeyCloak จะเก็บ CKAN User ID เพื่อนำไปใช้ดึง API Key เพื่อไปใช้งาน CKAN Service และ เก็บ Tableau User ID เพื่อใช้ตอนใช้งาน Tableau

3.4.12 การเข้าใช้งาน Services ต่างๆ



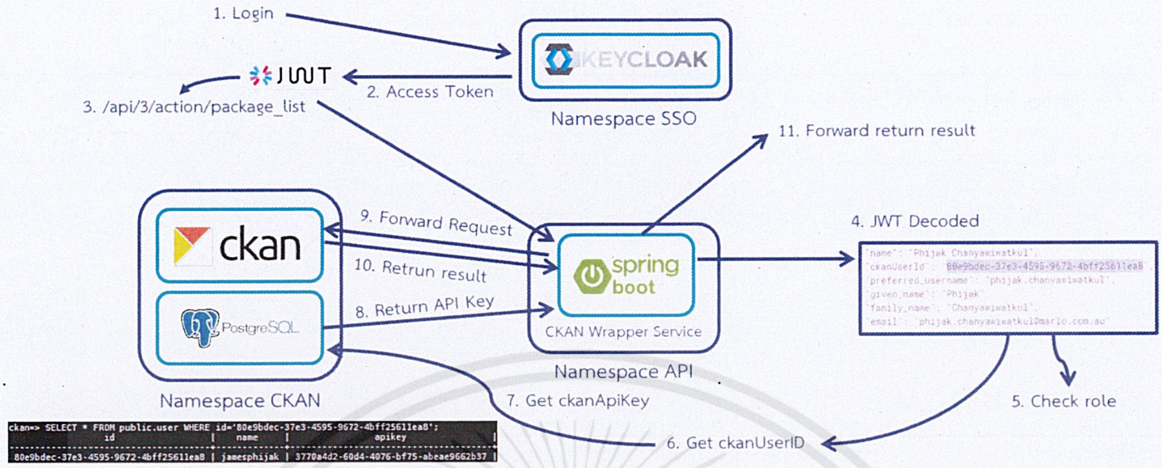
ภาพที่ 3.52 การเข้าใช้งาน API Service ต่างๆ

เมื่อ Account ทำการ Confirm E-mail และ Admin Approve แล้วทำการ Login เข้ามาจะได้ Token กลับไปเพื่อใช้ส่งไปหา Service ตัวอื่นเมื่อเรียกไปที่ URL ตาม Context Path ที่กำหนดผ่าน 3Scale ตัว 3Scale จะ Forward Request ไปยังตัว NGINX ที่ทำงานอยู่บน OpenShift จากนั้นจะทำการ Routing ไปยังเส้นทางที่ระบุไว้บน NGINX เพื่อไปหา Service ที่ต้องการ ตัว Service จะทำการตรวจสอบจาก JWT Token ที่ส่งมาโดยส่งกลับไปตรวจสอบที่ KeyCloak อีกครั้งว่า Token นี้มีจริงในระบบและตรวจสอบ Role จาก Token จากนั้นเช็คในตัว Service เองว่ามีสิทธิ์ใช้งานหรือไม่



ภาพที่ 3.53 การ Config 3Scale และ NGINX

3.4.12 การเข้าใช้งาน CKAN

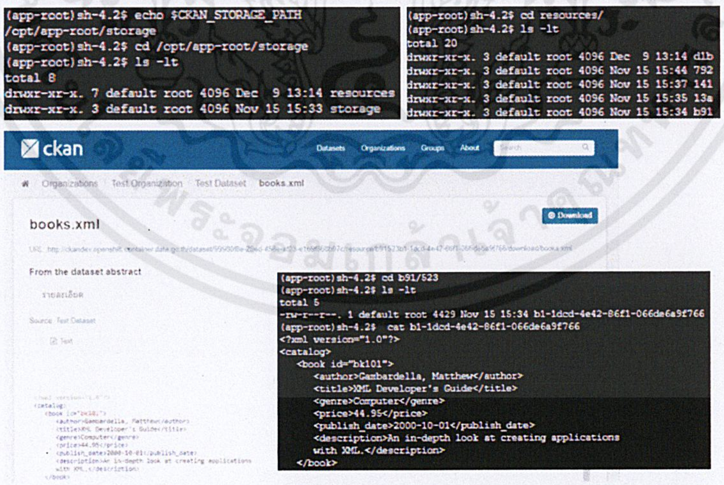


ภาพที่ 3.54 การเข้าใช้งาน CKAN API ผ่าน CKAN Wrapper Service

เมื่อเข้าสู่ระบบเรียบร้อยแล้วและได้ Token จะทำการส่ง Request เข้ามาตัว Service จะทำการตรวจสอบ Role ว่ามีสิทธิ์ใช้ Service ดังกล่าวหรือไม่ ถ้ามีสิทธิ์จะนำ ckanUserID ส่งไปหา CKAN API Key ที่ Database ของ CKAN และนำ API Key ส่ง Request ต่อไปหา CKAN

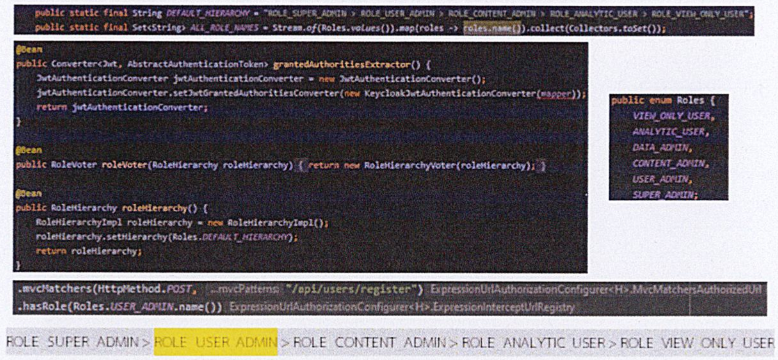
3.4.13 การเก็บ Resources ของ CKAN

CKAN จะอัพโหลดไฟล์ไปเก็บไว้ที่ $CKAN_STORAGE_PATH/resources/<Resource ID 3 ตัวแรก>/<Resource ID 3 ตัวต่อมา>/ตัวไฟล์จะมีชื่อเป็น Resource ID ตัวที่เหลือ



ภาพที่ 3.55 การเก็บไฟล์ Resources ของ CKAN

3.4.14 การตรวจสอบ Role แบบ Role Hierarchy



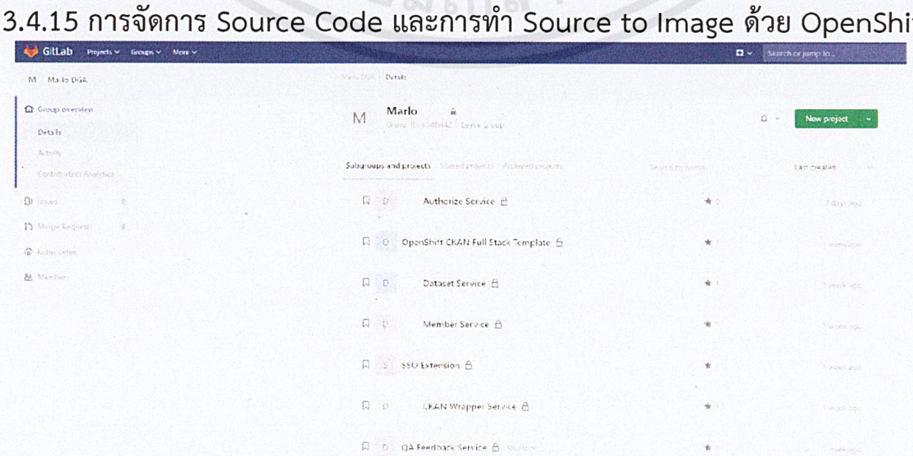
ภาพที่ 3.56 การตรวจสอบ Role แบบ Role Hierarchy

3.4.13 การสร้าง Template สำหรับ Deploy CKAN บน OpenShift

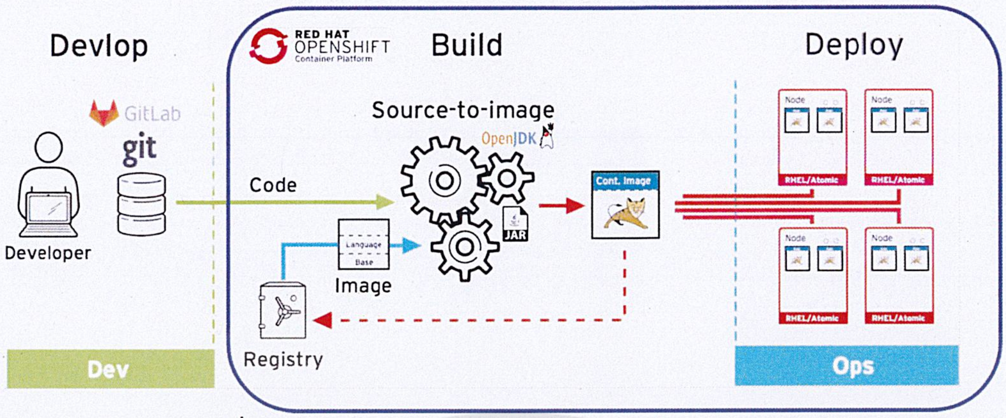
ภาพที่ 3.57 Template สำหรับ Deploy CKAN

สำหรับ Template ที่รวบรวม Deployment, Service, Route สำหรับ Deploy CKAN Stack โดยสามารถเปลี่ยนแปลง Parameter ต่างๆ และนำไป Deploy บน OpenShift เพียงใช้คำสั่ง oc process <ชื่อ Template> <ชื่อไฟล์ Parameter> ก็จะได้ Stack ทั้งหมดของ CKAN ไปทำงานอยู่บน OpenShift รวมถึง Mount Persistent Volume ไปยัง GlusterFS ด้วย

3.4.15 การจัดการ Source Code และการทำ Source to Image ด้วย OpenShift

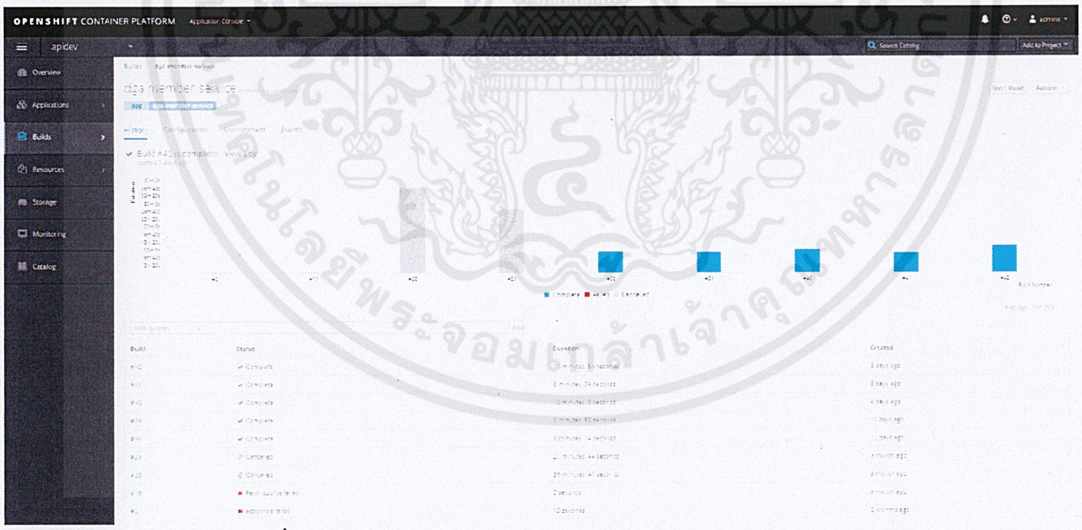


ภาพที่ 3.58 การจัดเก็บ Source Code บน Repository ของ GitLab สำหรับแต่ละ Service

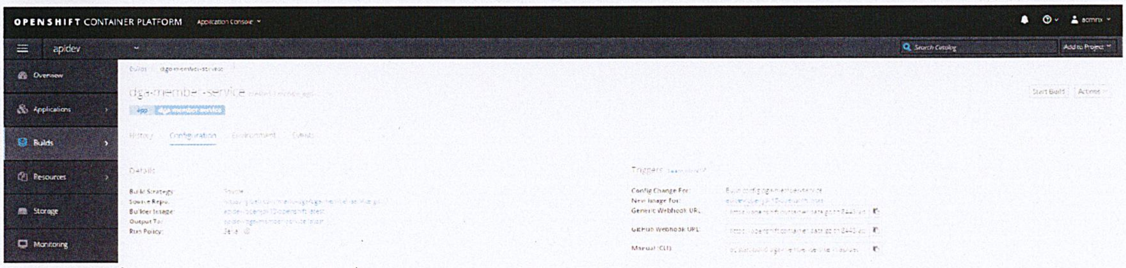


ภาพที่ 3.59 กระบวนการ Source 2 Image ด้วย OpenShift

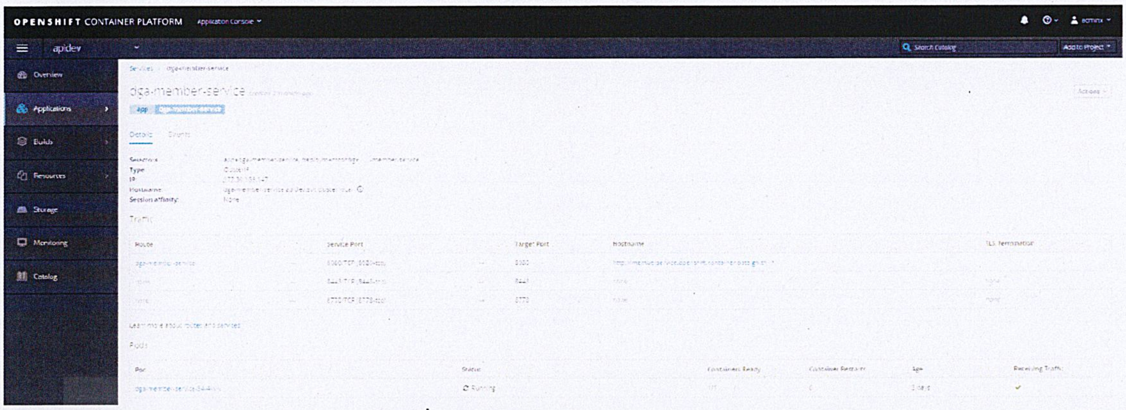
สำหรับ Source Code ที่เก็บไว้บน GitLab สามารถนำมาสร้างบน OpenShift ได้โดยใส่ Git URL และ Secret (กรณีเป็น Private Repository) และเลือกตัว Base Image จากนั้นตัว OpenShift จะนำ Source Code มาทำการ Build (กรณีเป็น Java Maven จะตรวจสอบไฟล์ pom.xml) และทำการ Build จนได้ไฟล์ .jar จากนั้นจะนำไปส่งไว้ใน Image ที่ได้เลือกไว้และเก็บอยู่บนเครื่อง Local ของ OpenShift และ OpenShift จะทำการดึง Image ดังกล่าวไป Deploy บน OpenShift ตามไฟล์ Deployment จากนั้นจะสร้าง Service เพื่อเข้าใช้ Pod ที่ไม่สามารถคาดเดาได้ว่าชื่ออะไรและมีจำนวนเท่าไหร จากนั้นจะทำการสร้าง Route เพื่อเข้ามาหา Service



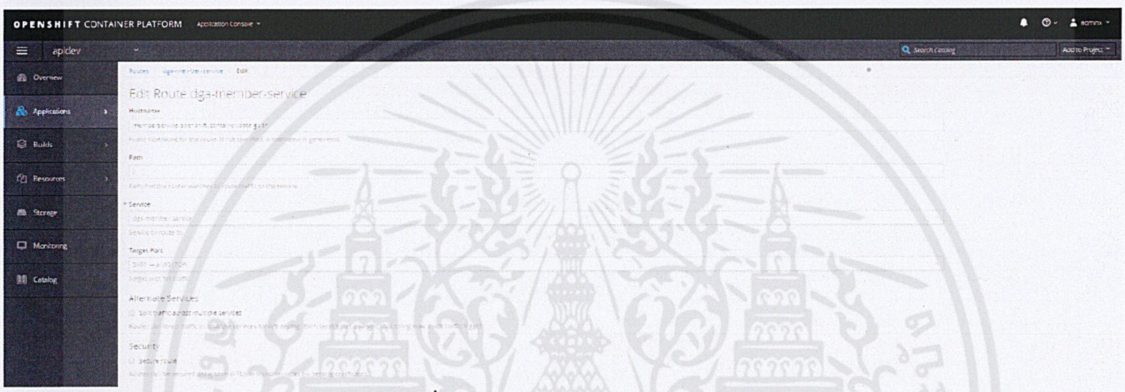
ภาพที่ 3.60 การ Build จาก Source to Image บน OpenShift



ภาพที่ 3.61 Web Hook เมื่อมี Event จาก GitLab สำหรับการ Trigger Build บน OpenShift

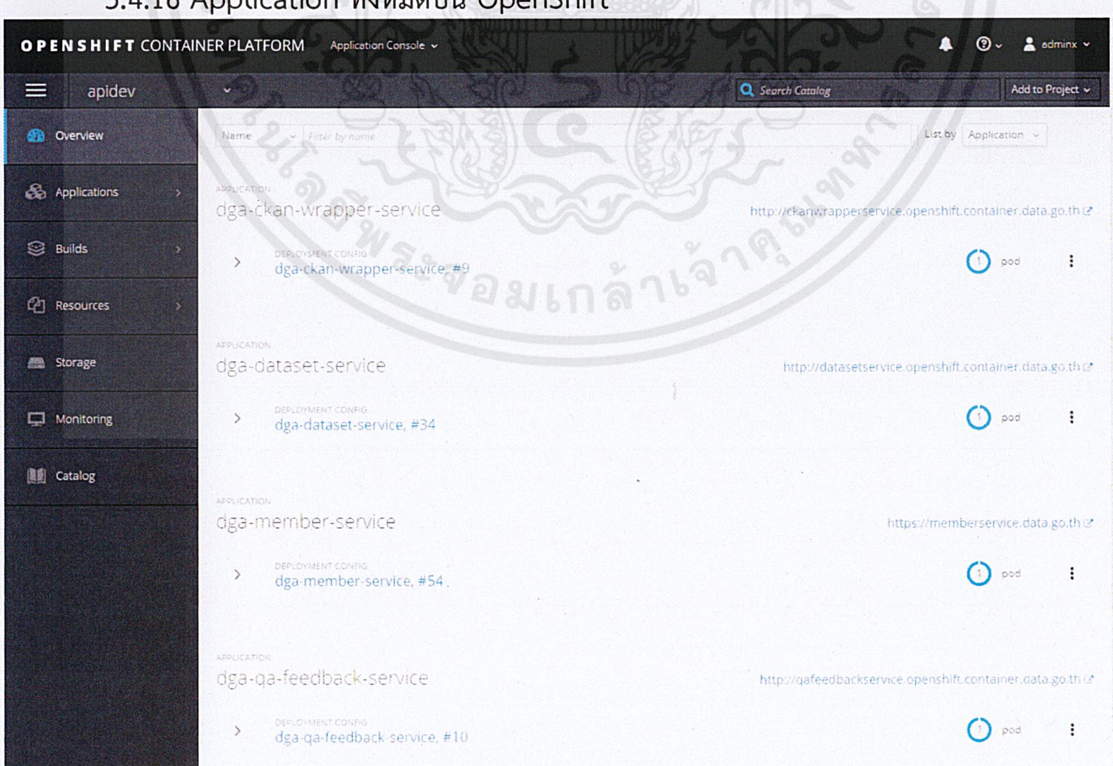


ภาพที่ 3.62 Service บน OpenShift

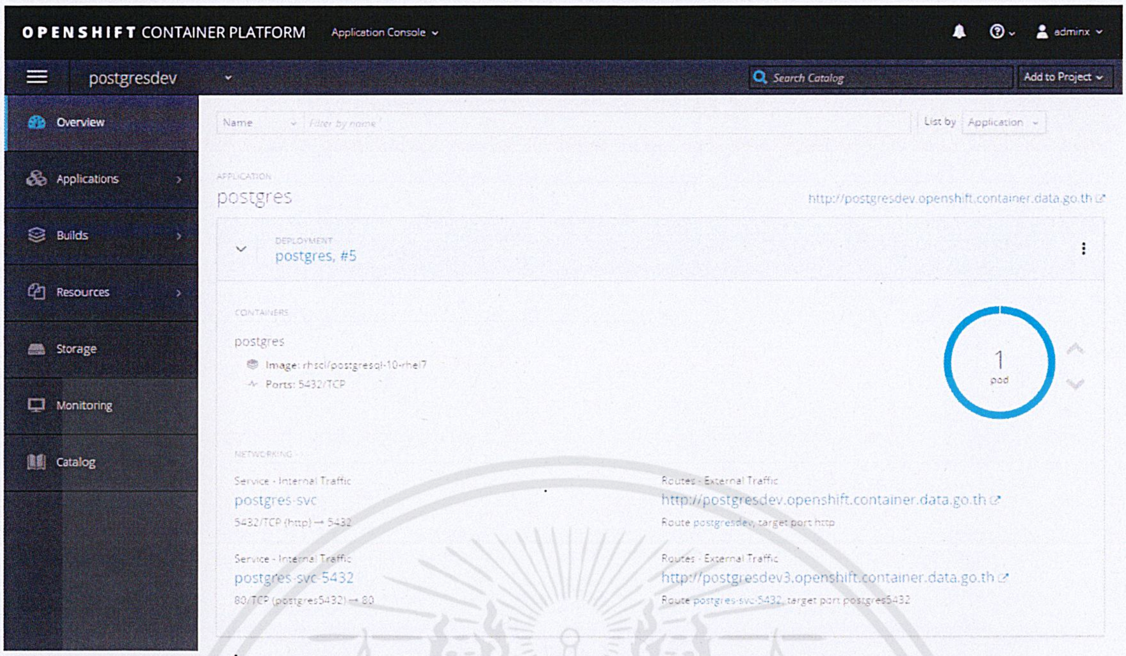


ภาพที่ 3.63 Route บน OpenShift

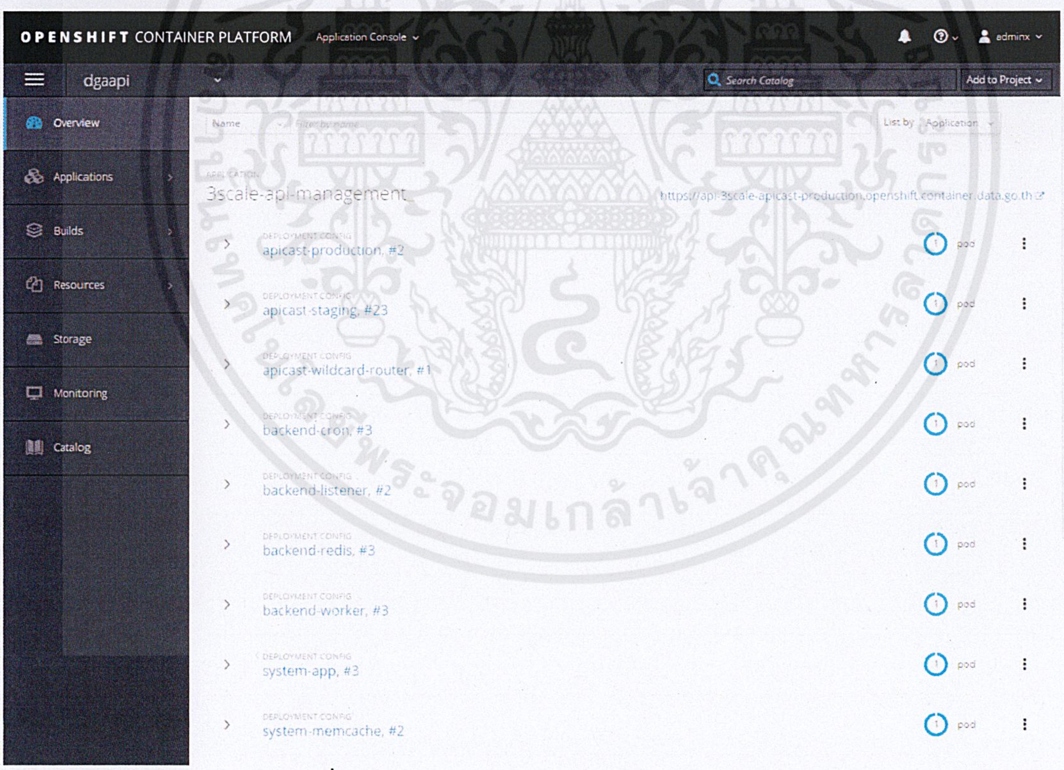
3.4.16 Application ทั้งหมดบน OpenShift



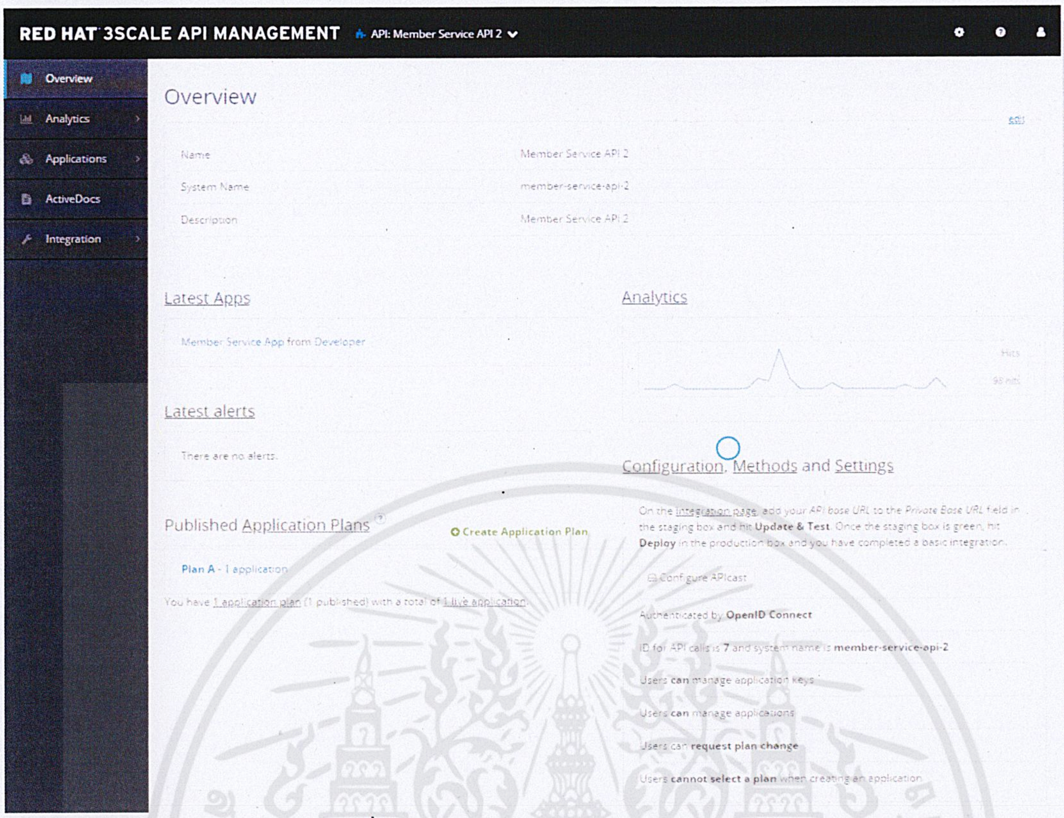
ภาพที่ 3.64 Application ของ Service API บน OpenShift



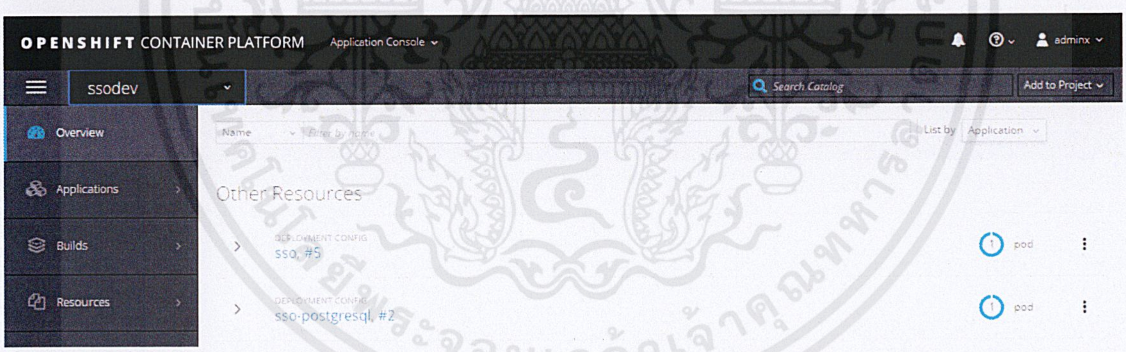
ภาพที่ 3.65 Application ของ PostgreSQL สำหรับ API บน OpenShift



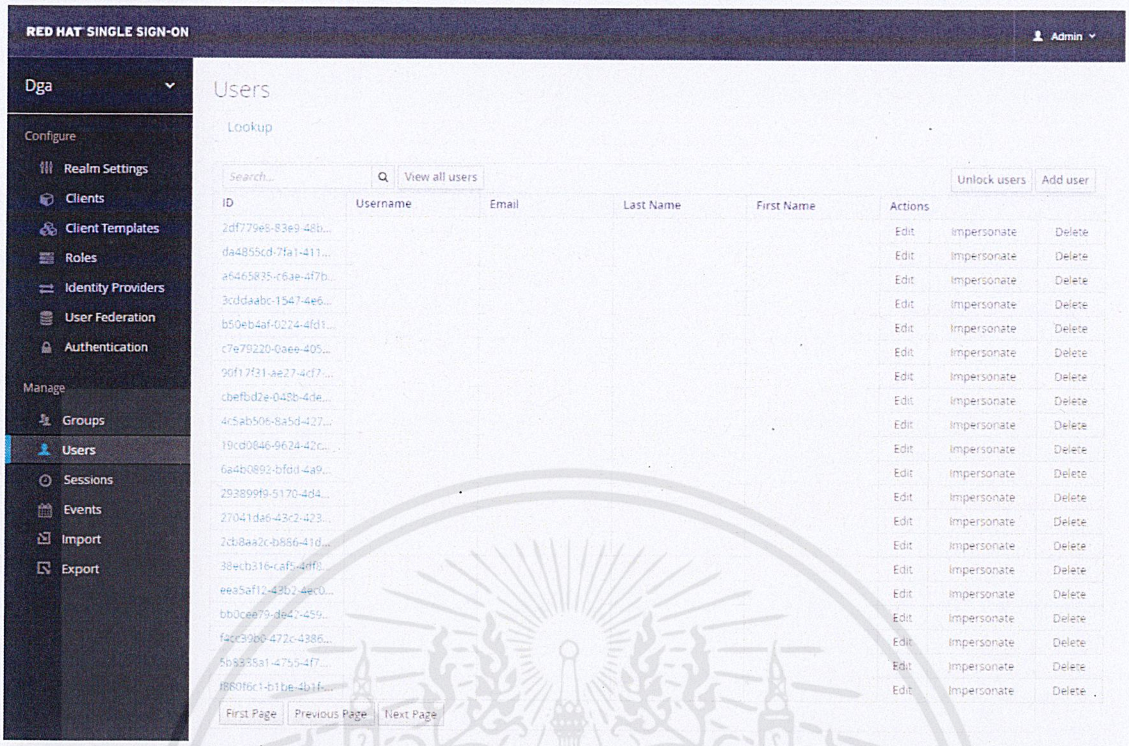
ภาพที่ 3.66 Application ของ 3Scale บน OpenShift



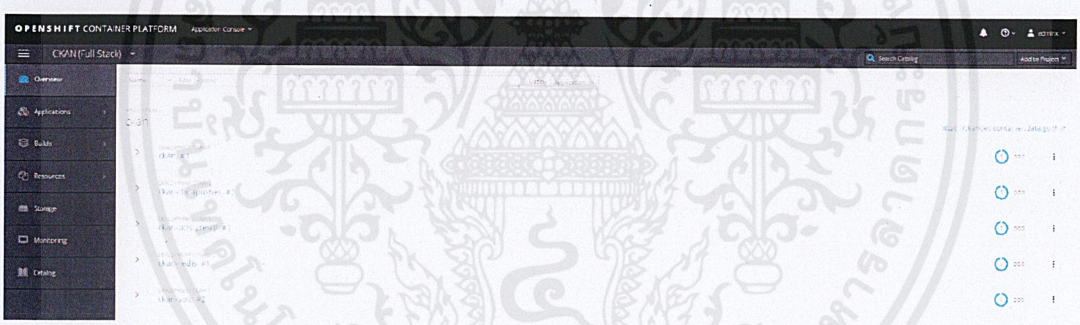
ภาพที่ 3.67 ตัวอย่าง Application ของ 3Scale



ภาพที่ 3.68 Application ของ KeyCloak หรือ Single Sign-On บน OpenShift

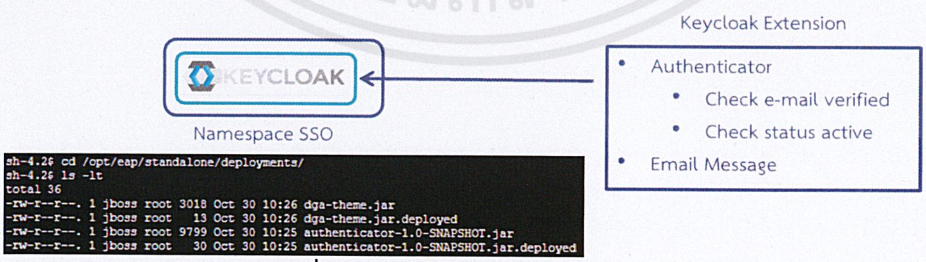


ภาพที่ 3.69 ตัวอย่าง Application KeyCloak หรือ Single Sign-On



ภาพที่ 3.70 Application ของ CKAN บน OpenShift

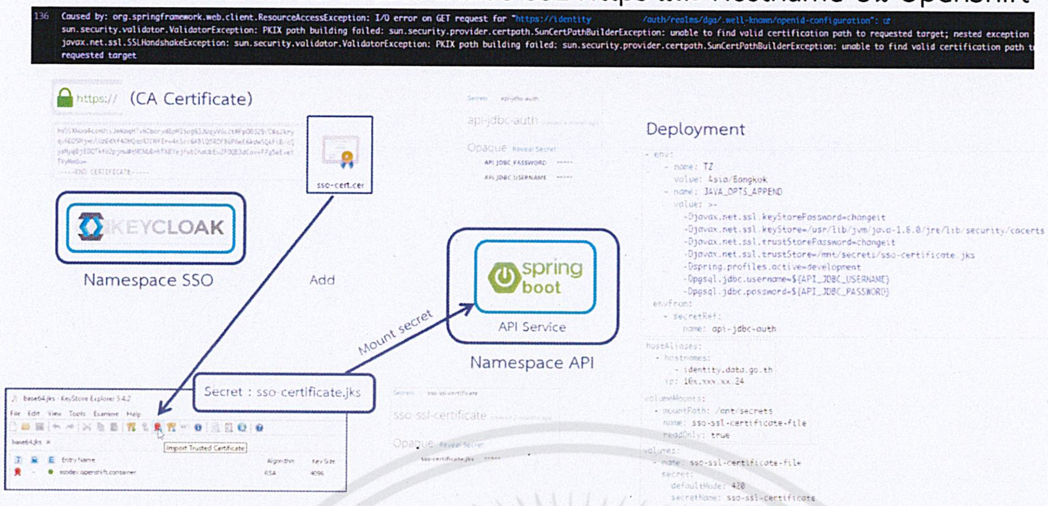
3.4.17 การเพิ่ม Extension สำหรับ KeyCloak



ภาพที่ 3.71 Extension บน KeyCloak

เนื่องจาก KeyCloak สามารถเข้าสู่ระบบโดยไม่ต้องรอยืนยัน E-mail และไม่ต้องรอสถานะของ Status ให้เป็น Activate ตัว Extension จะเข้าไปแก้ไขให้สามารถตรวจสอบตอน Login ได้ และ Extension อีกตัวคือการแก้ไขข้อความที่ส่งไปยังอีเมลของผู้ใช้งาน

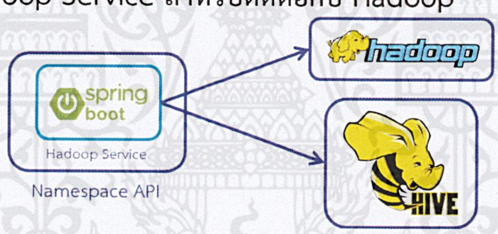
3.4.18 การเพิ่ม Certificate สำหรับ SSL Https และ Hostname บน OpenShift



ภาพที่ 3.72 การเพิ่ม Certificate สำหรับ Java บน OpenShift

เนื่องจาก Service มีการเชื่อมต่อไปยังอีก Service หนึ่งที่ใช้ Https ซึ่งต้องใช้ Certificate ในการเข้าใช้งาน จึงต้องมีการ Import Certificate ลงไปในตัว JAVA JRE เพื่อให้สามารถติดต่อใช้งานได้

3.4.19 การใช้ Hadoop Service สำหรับติดต่อกับ Hadoop



ภาพที่ 3.73 การเชื่อมต่อ Hadoop Service กับ Hadoop และ Hive

Hadoop Service จะช่วยติดต่อไปยัง Hadoop เพื่อใช้ Upload และจัดการไฟล์ต่างๆ ส่วน Hive จะใช้เมื่อ Tableau ต้องการเข้ามาอ่านข้อมูลบน Hive โดยสามารถ Query เป็น Row ได้

3.5 การประยุกต์ใช้กระบวนการและเครื่องมือของ DevOps มาใช้ร่วมกับการพัฒนาซอฟต์แวร์

เนื่องจากปัจจุบันเทคโนโลยีการพัฒนา และสร้างซอฟต์แวร์มีการเปลี่ยนแปลงไปอย่างรวดเร็ว รวมถึงมีการเพิ่มเทคโนโลยีใหม่ๆ มากมาย ทำให้นักพัฒนามีความจำเป็นที่ต้องเรียนรู้ ศึกษา และต้องสามารถนำคุณสมบัติใหม่ๆ มาปรับใช้กับการพัฒนาซอฟต์แวร์ขององค์กรได้ เทคโนโลยีใหม่นี้ จะทำให้การสร้าง และพัฒนาซอฟต์แวร์เป็นไปได้ง่ายขึ้น สามารถส่งมอบซอฟต์แวร์ได้รวดเร็วกว่าเดิม มีประสิทธิภาพ เหมาะกับการทำงานเป็นทีม และมีความปลอดภัยสูง

3.5.1.1 โปรแกรม CI/CD (Continuous Integration/Continuous Delivery)

1) สามารถเชื่อมต่อกับ GitLab ซึ่งเป็นที่จัดเก็บของ Source Code

2) สามารถตรวจสอบ GitLab ได้ว่ามีผู้ใช้งาน Commit Source Code ซึ่งโปรแกรมควบคุมการสร้างและทดสอบ (CI) นี้จะต้องสามารถนำเอารหัสโปรแกรมดังกล่าวมาทำการแปลงภาษา (Compile) และสร้างบรรจุภัณฑ์ (Package) เพื่อใช้ในการติดตั้ง และจัดเก็บ Package ดังกล่าวไว้บนที่จัดเก็บ (Artifactory) โดยจะกำหนดหมายเลข Build Number ให้กับ Package ดังกล่าวได้

3) สามารถดาวน์โหลด Package จากที่เก็บ Artifactory โดยระบุหมายเลขการสร้าง (Build Number) เมื่อได้ Package ที่กำหนด จะถูกนำไปไว้ในสภาพแวดล้อมสำหรับการสร้าง Image เพื่อนำไปใช้ Deploy บน Container Platform

4) สามารถเชื่อมต่อกับโปรแกรมทดสอบระบบ (Testing Tool) เพื่อทำการทดสอบแบบอัตโนมัติ (Automated Test) ตามขั้นตอนตาม Script ของ Test Framework ที่ระบุไว้

5) สามารถรายงานผลการทดสอบ ทั้งผ่านและไม่ผ่าน ไปยังระบบประสานงานอัตโนมัติสำหรับนักพัฒนาโปรแกรมและผู้ดูแลระบบ

6) สามารถสร้างโครงสร้าง (Model หรือ Pipeline) เพื่อทำการส่งมอบโปรแกรม (Software Deliver) แบบอัตโนมัติ โดยที่โครงสร้างดดังกล่าวต้องครอบคลุมการพัฒนาโปรแกรม และการทดสอบโปรแกรม และการติดตั้งบนระบบ Production

7) สามารถกำหนดให้มีผู้อนุมัติ (Approver) เพื่อนำโปรแกรมไปติดตั้งบนระบบ Production แบบอัตโนมัติ

8) สามารถแสดงรายงานและหน้าจอที่แสดงสถานะของการนำโปรแกรมขึ้น Production (Release Dashboard) เช่น Pipeline ที่ทำสำเร็จ และสามารถแสดงขั้นตอนที่กำลังทำงานอยู่หรือมีความผิดพลาด (Error)

3.5.1.2 โปรแกรมบริหารจัดการ Source Code (Source Code Management)

1) สามารถสร้างโครงงาน (Project) เพื่อเก็บรวบรวมที่จัดเก็บ (Repository) สำหรับจัดเก็บ Source Code โดยที่มีการควบคุม Version (Version Control)

2) ต้องสามารถสร้าง เพิ่ม และลบเอกสารในที่จัดเก็บ (Repository)

3) ต้องสามารถยืนยันการเปลี่ยนแปลง (Commit) ของ Source Code ได้

3.5.1.3 โปรแกรมบริหารจัดการ Artifact

1) ต้องทำหน้าที่เป็นที่จัดเก็บ Artifact (Repository)

2) สามารถจัดการ เพิ่ม ลบ แก้ไข Artifact ได้

3.5.1.4 โปรแกรมทดสอบระบบ (Testing Tools)

1) สามารถทำงานผ่าน Driver ของ Google Chrome ได้

2) สามารทำงานบนระบบปฏิบัติการ (OS) Linux ได้

3.5.1.5 โปรแกรมทดสอบระบบแบบเน้นปริมาณ (Load Testing)

3.5.1.6 โปรแกรมควบคุมคุณภาพ Source Code

1) สามารถให้นักพัฒนาตรวจสอบคุณภาพของ Source Code ได้

2) สามารถแสดงรายละเอียดคุณภาพของโปรเจค รวบถึงเก็บสถิติต่างๆ เช่น ข้อผิดพลาดที่เกิดขึ้น รหัสโปรแกรมที่ไม่ได้ใช้งาน (Unused Code)

3.5.1.7 โปรแกรมจัดเก็บ Docker Image (Docker Registry)

1) สามารถตรวจสอบรายละเอียดของ Docker image โดยใช้ GUI

2) สามารถลบ Docker image ที่ไม่ต้องการได้

3.5.1.8 ซอฟต์แวร์เพื่อความร่วมมือ (Collaborative Software)

1) สามารถสร้างกลุ่มสำหรับการสนทนาได้และกำหนดเป็นกลุ่มส่วนตัว

2) สามารถอัพโหลดไฟล์ได้

3) สามารถใช้งานได้ทั้ง PC และ Mobile Phone

4) สามารถค้นหารายชื่อผู้ใช้งานได้

5) สามารถปักหมุด (Pinned Message) ข้อความได้

6) สามารถแจ้งเตือนข้อความที่ผู้ใช้ถูกกล่าวถึงและข้อความที่ยังไม่ได้อ่านได้

7) สามารถจัดเก็บข้อมูลสนทนาได้

3.5.2 ขั้นตอนการดำเนินงาน

1) เลือกใช้เทคโนโลยีสำหรับพัฒนาโครงงานให้เหมาะสมกับความต้องการ

2) ศึกษาเทคโนโลยีที่ต้องใช้สำหรับพัฒนาโครงงาน

3) ออกแบบและติดตั้งโปรแกรมบนเครื่องเซิร์ฟเวอร์เพื่อใช้สำหรับพัฒนาโครงงาน

4) ออกแบบขั้นตอนการทำงานของกระบวนการ CI/CD สำหรับ 6 ภาษา คือ PHP, Java, Node.js, Angular, Go, Python

5) ทดสอบและปรับปรุงการทำงานให้ขั้นตอนการทำงานถูกต้อง

6) สรุปผลและจัดทำเอกสารอธิบายกระบวนการทำงานและโครงสร้างของโครงงาน

7) นำเสนอผลงานและส่งงานให้ลูกค้าตรวจสอบรวมถึงแนะนำวิธีการใช้งาน

3.5.3 การวิเคราะห์และออกแบบโครงสร้าง Pipeline

3.5.3.1 เทคโนโลยีที่เลือกใช้สำหรับเป็นเครื่องมือช่วยในกระบวนการของ DevOps

ตารางที่ 3.65 เทคโนโลยีที่เลือกใช้สำหรับเป็นเครื่องมือในกระบวนการของ DevOps

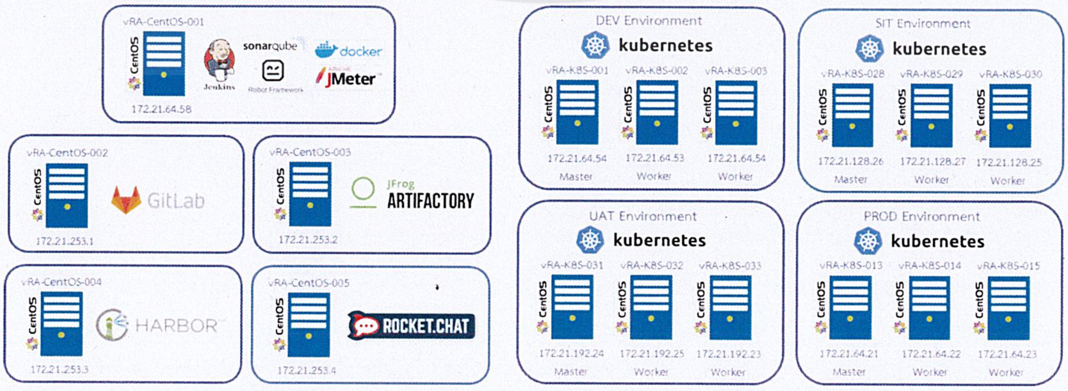
ประเภทของโปรแกรม	เทคโนโลยีที่เลือกใช้
โปรแกรมเพื่อใช้สำหรับการควบคุม CI/CD	Jenkins
โปรแกรมบริหารจัดการ Source Code	GitLab
โปรแกรมบริหารจัดการ Artifact	JFrog Artifactory
โปรแกรมทดสอบระบบ	Robot Framework
โปรแกรมทดสอบระบบแบบเน้นปริมาณ	Apache JMeter
โปรแกรมควบคุมคุณภาพ Source Code	SonarQube
โปรแกรมจัดเก็บ Docker Image	Harbor Registry
ซอฟต์แวร์เพื่อความร่วมมือ	Rocket Chat
Platform สำหรับ Deploy Container	Kubernetes

3.5.3.2 เทคโนโลยีที่เลือกใช้สำหรับจัดการ Dependency ในแต่ละภาษา

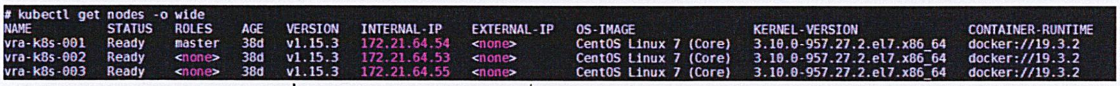
ตารางที่ 3.66 เทคโนโลยีที่เลือกใช้สำหรับจัดการ Dependency ในแต่ละภาษา

ภาษา	เทคโนโลยีที่เลือกใช้
PHP	Composer
Java Spring Boot	Maven
JavaScript Node.js	NPM
TypeScript Angular	NPM / Angular CLI
Go	Go
Python	PIP

3.5.3.3 ภาพรวมของเครื่องเซิร์ฟเวอร์ที่ใช้ในการพัฒนา

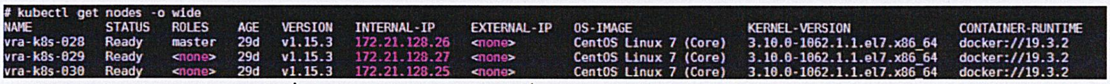


ภาพที่ 3.74 การติดตั้งเทคโนโลยีต่าง ๆ ลงบนเครื่องเซิฟเวอร์



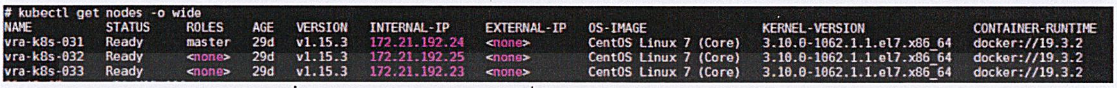
```
# kubectl get nodes -o wide
NAME         STATUS   ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-RUNTIME
vra-k8s-001  Ready    master   38d   v1.15.3   172.21.64.54   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.2
vra-k8s-002  Ready    <none>   38d   v1.15.3   172.21.64.53   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.2
vra-k8s-003  Ready    <none>   38d   v1.15.3   172.21.64.55   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.2
```

ภาพที่ 3.75 รายละเอียดเครื่องเซิร์ฟเวอร์ Environment DEV



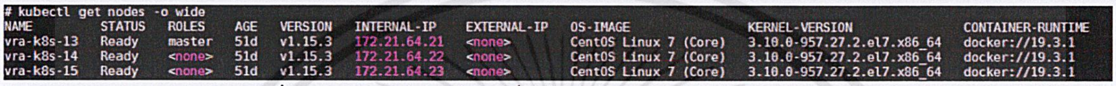
```
# kubectl get nodes -o wide
NAME         STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-RUNTIME
vra-k8s-028  Ready    master   29d   v1.15.3   172.21.128.26   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
vra-k8s-029  Ready    <none>   29d   v1.15.3   172.21.128.27   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
vra-k8s-030  Ready    <none>   29d   v1.15.3   172.21.128.25   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
```

ภาพที่ 3.76 รายละเอียดเครื่องเซิร์ฟเวอร์ Environment SIT



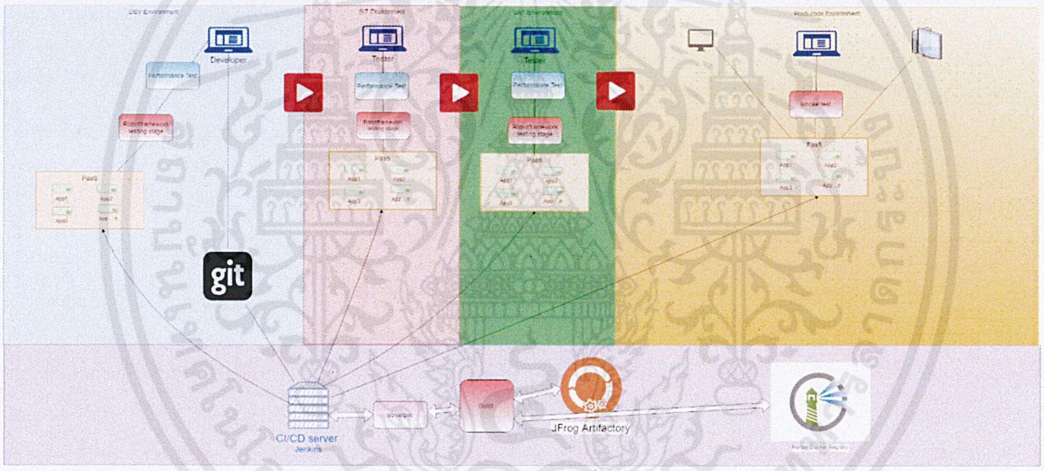
```
# kubectl get nodes -o wide
NAME         STATUS   ROLES    AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-RUNTIME
vra-k8s-031  Ready    master   29d   v1.15.3   172.21.192.24   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
vra-k8s-032  Ready    <none>   29d   v1.15.3   172.21.192.25   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
vra-k8s-033  Ready    <none>   29d   v1.15.3   172.21.192.23   <none>        CentOS Linux 7 (Core)   3.10.0-1062.1.1.el7.x86_64   docker://19.3.2
```

ภาพที่ 3.77 รายละเอียดเครื่องเซิร์ฟเวอร์ Environment UAT

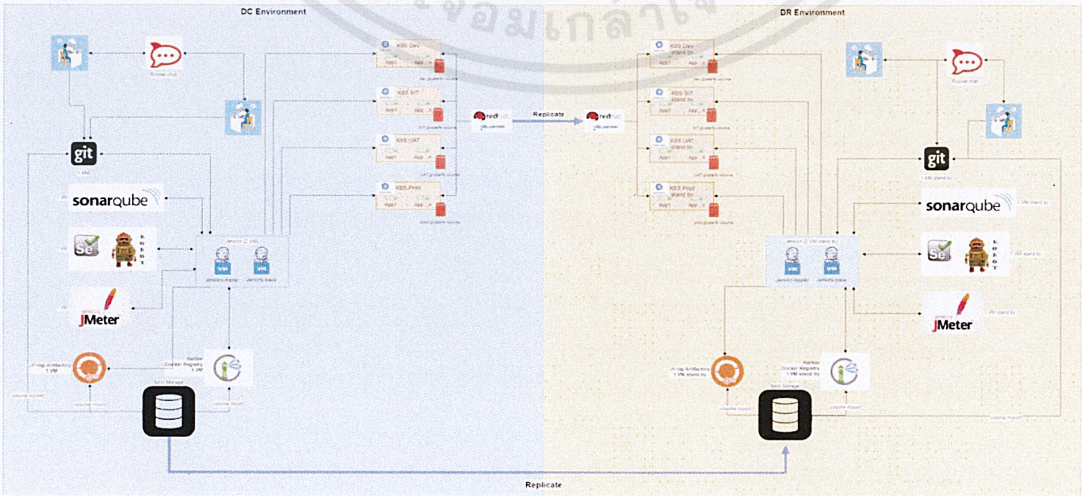


```
# kubectl get nodes -o wide
NAME        STATUS   ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-RUNTIME
vra-k8s-13  Ready    master   51d   v1.15.3   172.21.64.21   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.1
vra-k8s-14  Ready    <none>   51d   v1.15.3   172.21.64.22   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.1
vra-k8s-15  Ready    <none>   51d   v1.15.3   172.21.64.23   <none>        CentOS Linux 7 (Core)   3.10.0-957.27.2.el7.x86_64   docker://19.3.1
```

ภาพที่ 3.78 รายละเอียดเครื่องเซิร์ฟเวอร์ Environment PROD

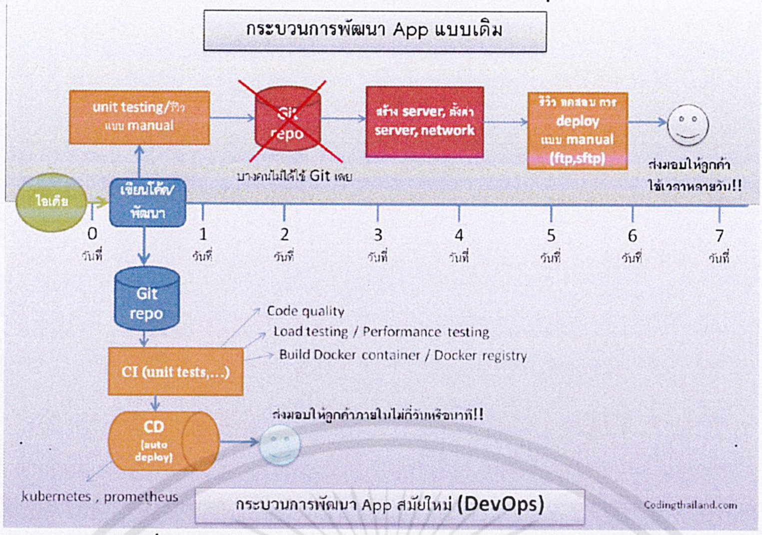


ภาพที่ 3.79 แผนผัง Automate Pipeline

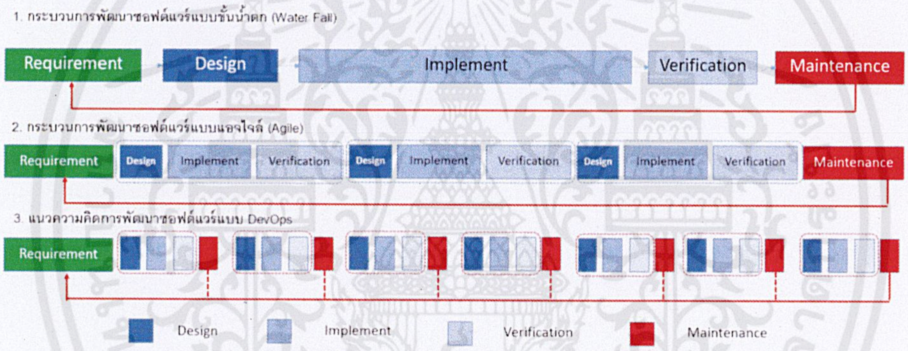


ภาพที่ 3.80 รายละเอียดแผนผังของ Data Center และ Disaster Recovery

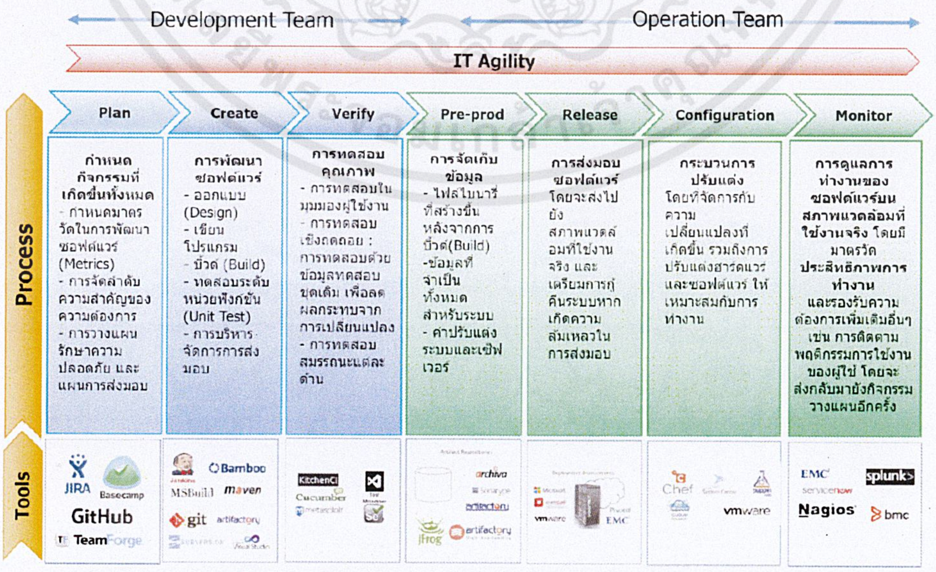
3.5.3.4 ขั้นตอนของกระบวนการทำงานแบบ DevOps



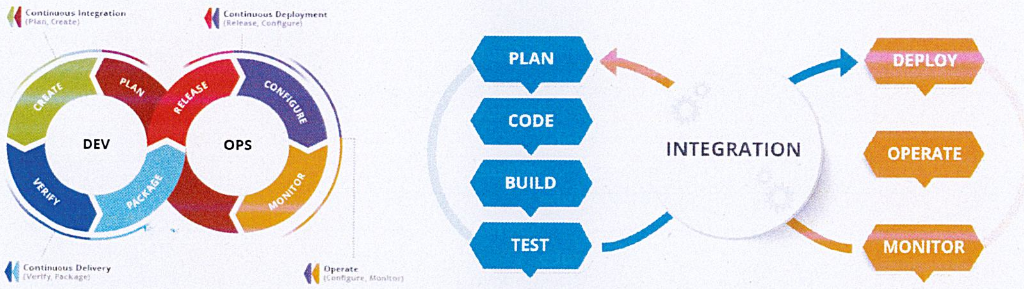
ภาพที่ 3.81 เปรียบเทียบรูปแบบเดิมกับแบบ DevOps



ภาพที่ 3.82 เปรียบเทียบรูปแบบ Waterfall, Agile, DevOps



ภาพที่ 3.83 อธิบายกระบวนการทำงานของ DevOps

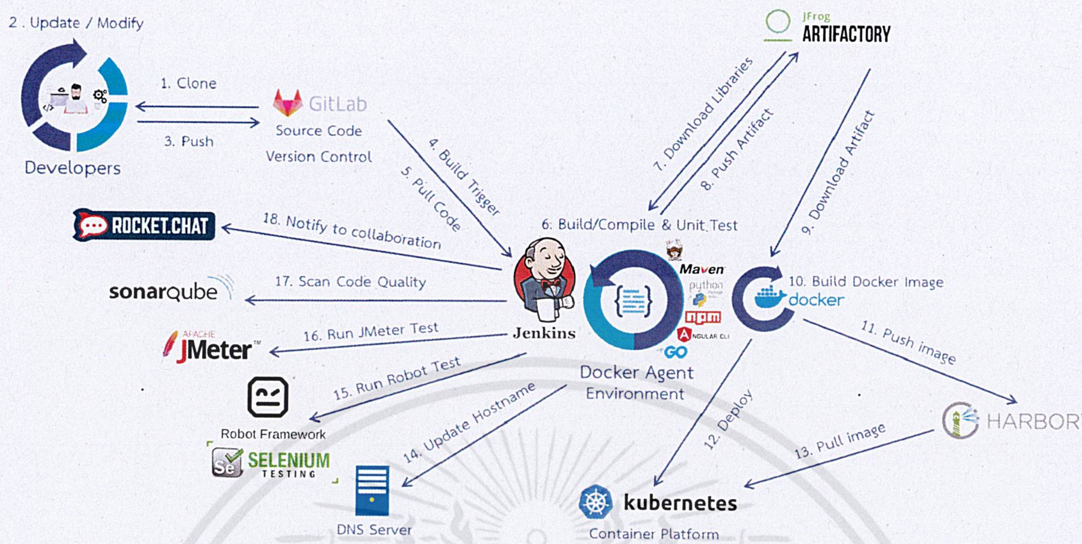


ภาพที่ 3.84 ขั้นตอนกระบวนการทำงานของ DevOps

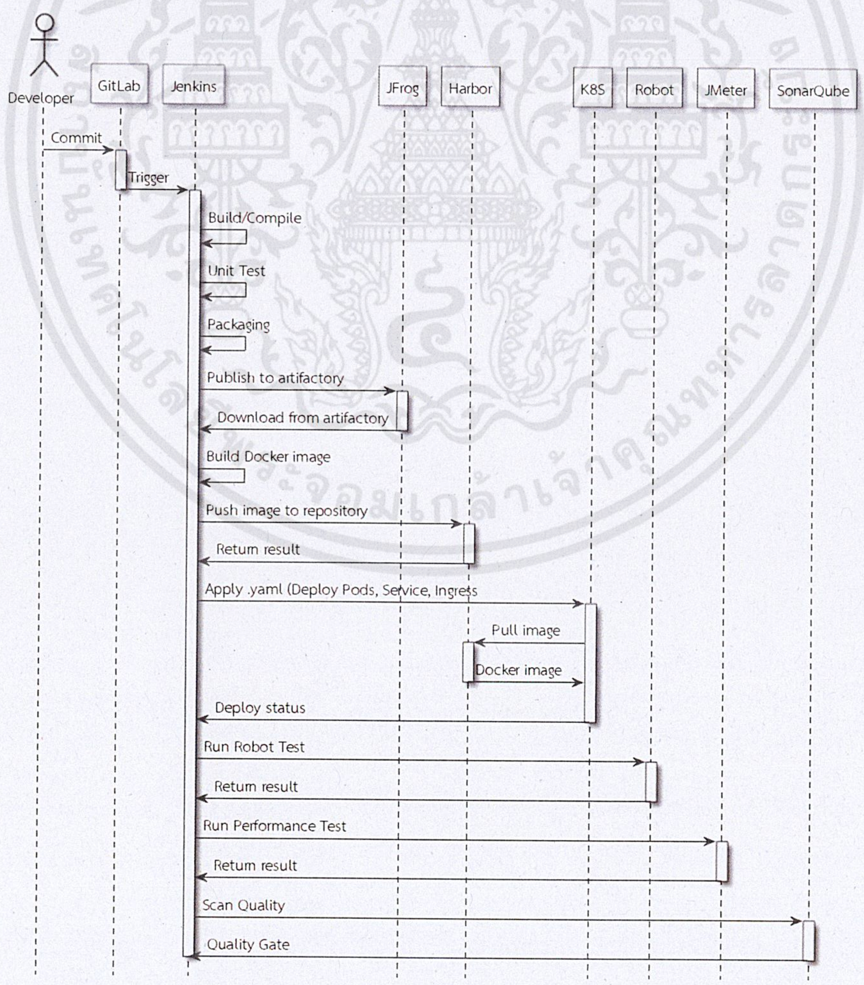
ตารางที่ 3.67 กระบวนการทำงานของ DevOps เบื้องต้น

กระบวนการ	อธิบาย
Plan	การวางแผนการพัฒนาโปรแกรมจากทุกฝ่ายที่เกี่ยวข้อง เช่น โปรแกรมจะทำอะไร, พัฒนาด้วยภาษาอะไร, ทำงานด้วย Platform อะไร, เก็บข้อมูลอย่างไร, กระบวนการทำงาน
Code	การสร้างและพัฒนาโปรแกรม โดยนักพัฒนาจะพัฒนาโปรแกรมบน Environment เดียวกันไม่ว่าจะพัฒนาบน OS ใดก็ตามด้วย Docker และมีการเก็บควบคุมเวอร์ชั่นของซอร์สโค้ดบน GitLab
Package	การนำซอร์สโค้ดที่เก็บอยู่บน GitLab มาทำการ Build หรือ Compile (ในบางภาษาจะติดตั้ง Library ที่จำเป็นเท่านั้น ไม่มีการ Build หรือ Compile) ให้โปรแกรมสามารถทำงานได้ และทำการตรวจสอบว่าโปรแกรมถูกต้องตามที่ต้องการ รวมถึงการทำ Unit Test เมื่อได้ Artifact ที่ถูกต้องและพร้อมใช้งานจะถูกนำไปเก็บบน JFrog Artifactory Server และนำ Artifact ดังกล่าวมาใส่ใน Docker image รวมถึงจัดเก็บไว้บน Harbor Image Registry เพื่อพร้อมนำไป Deploy โดยมีการระบุเลขในการ Build แต่ละครั้งไว้ในไฟล์ของ Artifact รวมถึงระบุไว้ใน Tag ของ Image ด้วย
Deploy	การเขียนไฟล์ Deploy เพื่อนำ Image ที่อยู่บน Harbor Registry ไป Deploy บน Kubernetes
Test	การเขียนไฟล์ Test UI/API และใช้ Robot Framework ทำงาน รวมถึงการเขียนไฟล์ Test Performance และใช้ JMeter ทดสอบประสิทธิภาพ
Operate	Kubernetes สามารถรองรับภาระของโหลดด้วยการสร้าง Pods มารองรับ Traffic ที่เกิดขึ้นจำนวนมากได้แบบอัตโนมัติ (Horizontal Pod Auto Scaler)
Monitor	การนำ Logs จาก Container ที่รันอยู่บน Kubernetes ทั้งหมดส่งไปให้ Elasticsearch เพื่อทำ Index ด้วยการลง Fluentd ไว้ที่แต่ละเครื่องของ Kubernetes เพื่อส่ง Log ของ Container ออกมา และสามารถเข้าใช้งานผ่านหน้า GUI ด้วย Kibana

3.5.3.5 การออกแบบกระบวนการทำงานในรูปแบบของ DevOps



ภาพที่ 3.85 ภาพรวมขั้นตอนการทำงานของ Pipeline



ภาพที่ 3.86 แผนภาพ Sequence Diagram ขั้นตอนการทำงานของ Pipeline

เมื่อนักพัฒนา (Developer) ได้รับมอบหมายงานมา จะทำการพัฒนาโปรแกรมให้ตรงตามที่ได้ออกแบบไว้ จากนั้นนักพัฒนาจะนำ Source Code ทั้งหมดขึ้นไปเก็บไว้บน Repository ที่อยู่บน GitLab เพื่อให้สามารถจัดการและควบคุมเวอร์ชั่นของ Source Code ได้ง่าย เมื่อ GitLab ทราบว่ามีการเปลี่ยนแปลงของ Source Code หรือมีการ Push Commit เข้ามาใหม่ GitLab จะไปสั่งให้ Jenkins เริ่มทำงาน ซึ่งตัว Jenkins จะไป Clone Source Code ที่อยู่บน GitLab ลงมามาเก็บไว้ที่เครื่อง Local ตาม Project ที่ได้สร้างไว้แล้ว เนื่องจากใน Project ถูกตั้งค่าไว้ให้อ่านกระบวนการทำงานทั้งหมดจาก Jenkinsfile ซึ่งมีอยู่ใน Source Code ที่ Clone มา ซึ่งถูกเขียนขึ้นมาด้วยภาษา Groovy มีการแบ่งการทำงานออกเป็น Stage สามารถตั้งชื่อ Stage และรวบรวมคำสั่งต่าง ๆ ที่เกี่ยวข้องไว้ใน Stage นั้นๆ ในที่นี่สามารถเรียกกระบวนการทำงานทั้งหมดว่า Pipeline

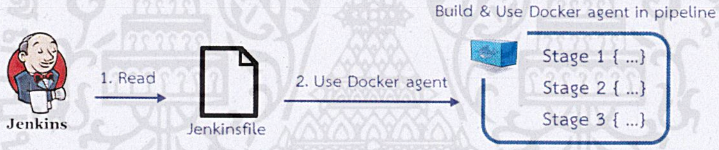
ในการพัฒนา Pipeline ของแต่ละภาษาจะมีความคล้ายคลึงกัน ซึ่งจะแตกต่างในบางขั้นตอนเท่านั้น เช่น การ Build, การทำ Unit Test เราจะต้องทราบว่าในแต่ละภาษานั้นทำงานอย่างไร ถึงจะเขียนคำสั่งลงไปให้ Jenkins ทำงานได้ถูกต้อง ซึ่งกระบวนการทำงานอื่น ๆ ใน Pipeline นั้นจะถูกออกแบบตามการทำงานของทีมในบริษัทว่าทำงานกันอย่างไร ในบางครั้งอาจจะไม่ได้ทำเป็นแบบอัตโนมัติ (Automate) ทั้งหมด แต่อาจนำ Jenkins เข้ามาช่วยทำงานในบางส่วนเท่านั้น ในการทำแม่แบบ (Template) Pipeline ครั้งนี้สามารถนำไปประยุกต์ใช้กับโปรเจคอื่น ๆ ที่ใช้ภาษาเดียวกันได้ อาจจะต้องเปลี่ยนคำสั่งบางอย่างให้เหมาะสมกับโปรเจคนั้น ๆ เพื่อให้สามารถทำงานได้อย่างถูกต้อง ซึ่งแม่แบบของภาษาที่จะพัฒนามีอยู่ทั้งหมด 6 ภาษา คือ 1) PHP 2) Java 3) Node.js 4) Angular 5) Go และ 6) Python

เมื่อ Jenkins เริ่มอ่าน Jenkinsfile ในส่วนเริ่มต้นของ Pipeline จะเป็นการใช้ Docker เป็น Agent ซึ่งทำหน้าที่มาครอบการทำงานในส่วนของ Pipeline อีกทีหนึ่ง โดยปกติตัว Jenkinsfile จะใช้สภาพแวดล้อมที่อยู่บนเครื่องตัวเอง เช่น เมื่อติดตั้งโปรแกรมใด ๆ บนเครื่อง Jenkins ก็สามารถใช้คำสั่งนั้นบน Pipeline ได้ทันที เหตุผลในการเลือกใช้ Docker Agent เข้ามาใช้ เพราะว่าสามารถแก้ไขเวอร์ชั่นหรือสภาพแวดล้อมที่ใช้ในการทำงานในแต่ละภาษาหรือในแต่ละโปรเจคได้ง่าย ไม่จำเป็นต้องติดตั้งลงบนเครื่อง Jenkins เช่น เมื่อต้องการใช้ Java เวอร์ชั่น 8 สำหรับโปรเจคที่ 1 แต่โปรเจคที่ 2 ต้องการใช้ Java เวอร์ชั่น 11 ทำให้เกิดปัญหาขึ้นได้ แต่ถ้าแยกเป็น Dockerfile ซึ่งสามารถแก้ไขเวอร์ชั่นหรือต้องการปรับแต่งอื่นๆเพิ่มเติมก็สามารถแก้ไขได้ทันที โดยไม่ต้องไปแก้ไขบนเครื่อง Jenkins ในขณะที่บนเครื่อง Jenkins จะลงโปรแกรมไว้เฉพาะเท่าที่จำเป็นเท่านั้น ในที่นี้ตัว Jenkins สามารถ Clone เป็น Image เพื่อนำไปสร้างเป็นเครื่องอื่นๆ ได้ด้วยโปรแกรมของ VMWare

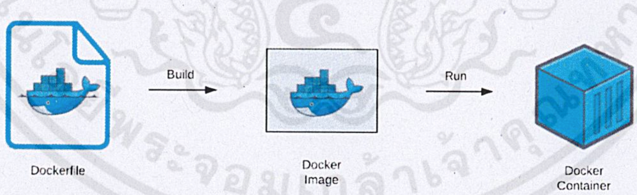
ในไฟล์ชื่อ Dockerfile ที่ใช้สำหรับเป็น Build Environment เมื่อสร้างเป็น Image ออกมาแล้ว เมื่อเริ่มใช้งานจะสามารถใช้สภาพแวดล้อมภายในทั้งหมดได้ทันที ในการออกแบบให้อ้างอิงตามภาษาของโปรเจคที่จะใช้ โดยสามารถออกแบบ Dockerfile สำหรับทั้ง 6 ภาษาได้ดังนี้

ตารางที่ 3.68 การเตรียม Dockerfile สำหรับ Build Environment ทั้ง 6 ภาษา

ภาษาที่ใช้	สิ่งที่ต้องติดตั้งใน Dockerfile เพื่อใช้เป็น Build Environment
PHP	Base image: PHP:7.3-apache Composer เวอร์ชั่นล่าสุด
JAVA	Base image: java:8-jdk-alpine Maven เวอร์ชั่น 3.6.1 (กำหนดได้)
Node.js	Base image: node:8
Angular	Base image: node:10 AngularCLI เวอร์ชั่น 8.1.2 (กำหนดได้)
Go	Base image: golang:1.12.6
Python	Base image: python:3.7.3-stretch
ทุกภาษาจำเป็นต้องมี	Java: ใช้สำหรับแสกน SonarQube JFrog CLI: ใช้สำหรับสั่งงานและเชื่อมต่อกับ JFrog Artifactory Docker: ใช้สำหรับ Build image และเชื่อมต่อกับ Docker บน Jenkins Kubectl: ใช้สำหรับสั่งงานไปยัง Kubernetes

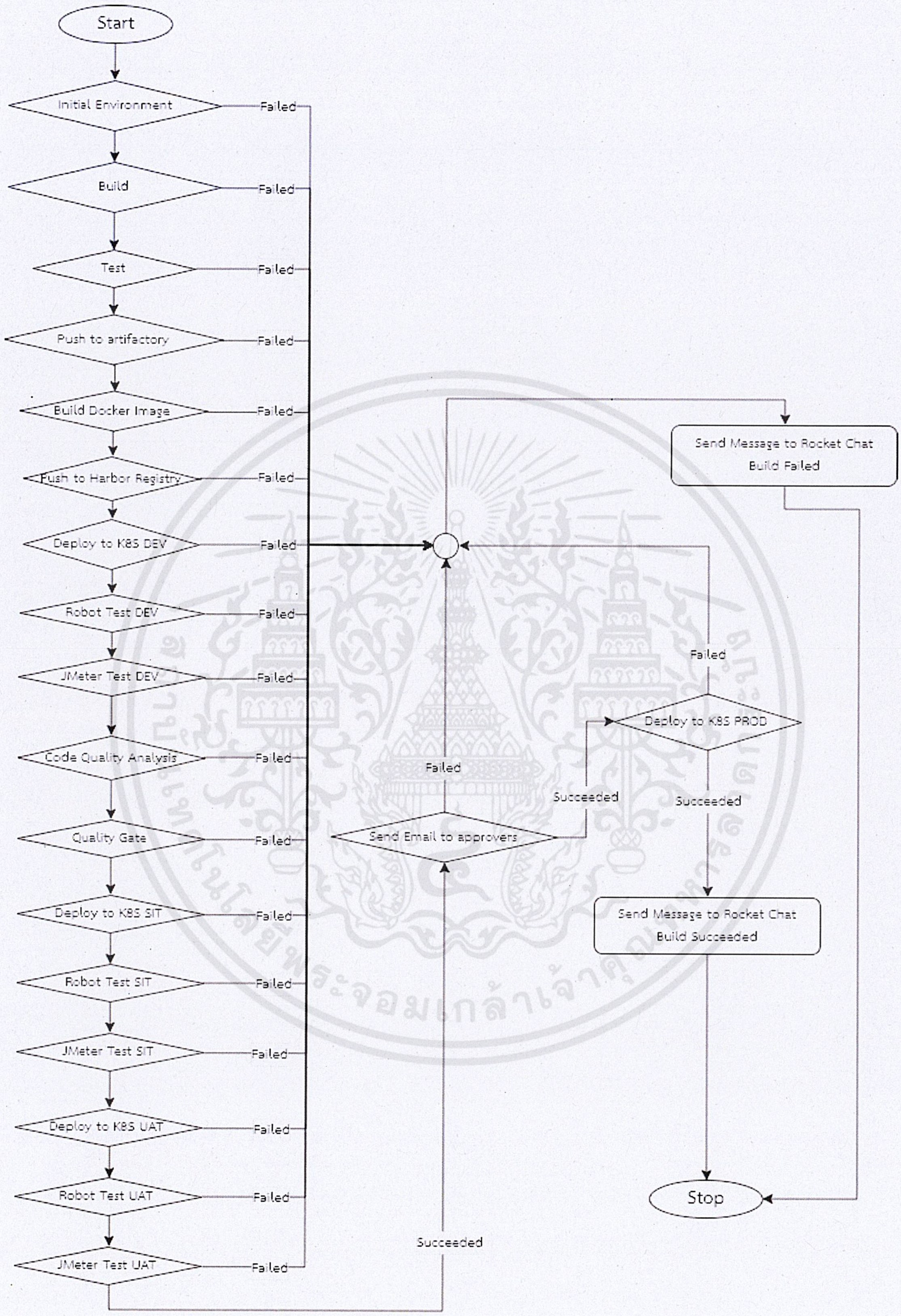


ภาพที่ 3.87 การทำงานของ Docker Agent บน Jenkins



ภาพที่ 3.88 การทำงานของ Docker เบื้องต้น

เมื่อ Jenkins เข้ามาทำงานใน Agent เรียบร้อยแล้ว Jenkins จะเริ่มทำงานตั้งแต่ Stage แรกไปจนถึง Stage สุดท้ายหากไม่มีข้อผิดพลาดในระหว่างการทำงาน



ภาพที่ 3.89 แผนภาพ Flowchart การทำงานของ Pipeline

3.5.4 อธิบาย Pipeline Stage

1) Stage Initial Environment: เป็นการอ่านค่าตัวแปรของ Environment ซึ่งอยู่ในไฟล์ .env มาเป็นตัวแปรที่ใช้ในสภาพแวดล้อมของ Container โดยตัวแปรที่มีในไฟล์ .env จะเป็นการตั้งค่า URL เซิร์ฟเวอร์ปลายทาง, ชื่อ Credentials ต่างๆ

2) Stage Code Quality Analysis: การให้ SonarQube เข้ามาแสกนโปรเจค เพื่อทดสอบคุณภาพของโค้ด

3) Stage Quality Gate: ตรวจสอบว่าผ่านเกณฑ์ตามที่กำหนดหรือไม่ด้วย Quality Gate

4) Stage Build: เป็นการนำ Source Code ของโปรเจคมาทำการติดตั้ง Package ที่จำเป็นโดยจะทำการตั้งค่าให้ดาวน์โหลดผ่าน JFrog Proxy ในแต่ละภาษาตั้งค่าไม่เหมือนกัน เพื่อให้ Source Code สมบูรณ์ก่อนทำการ Build หรือ Compile เพื่อนำไฟล์ที่ผ่านการ Build หรือ Compile สำเร็จแล้วไปใช้งานต่อ

5) Stage Test: เป็นการทำ Unit Test จากไฟล์เทสที่เขียนไว้ล่วงหน้าแล้ว

6) Stage Push to Artifactory: เป็นการนำไฟล์ที่ผ่านการ Build หรือ Compile แล้วนำไปเก็บไว้บน JFrog ซึ่งบางภาษาไม่ได้ถูก Compile เป็น Binary จึงต้องทำการเก็บรวบรวมเป็นไฟล์ที่ถูกบีบอัดแล้วจึงส่งขึ้นไปเก็บด้วย JFrog CLI

7) Stage Build Docker image & Push to Harbor registry: เป็นการดาวน์โหลดไฟล์ Artifact กลับลงมาและทำการ Build image จาก Dockerfile ซึ่งจะนำไฟล์หรือโฟลเดอร์ดังกล่าวไปไว้ใน Image หลังจากสร้าง Image แล้วจะนำ Image ไปเก็บไว้บน Harbor ซึ่งเป็น Image Registry
3.1.3.5.7 Stage Deploy to Kubernetes (DEV, SIT, UAT, PROD)

8) เป็นการนำ Application ที่ทำเป็น Image แล้ว Deploy ไปยัง Container Platform หรือ Kubernetes ในสภาพแวดล้อมของ DEV, SIT, UAT, PROD หลังจาก Apply ไฟล์ Deployment แล้วจะทำการอัพเดท Dynamic DNS ให้สามารถวิ่งเข้ามาหาเครื่อง Kubernetes ได้ถูกต้อง

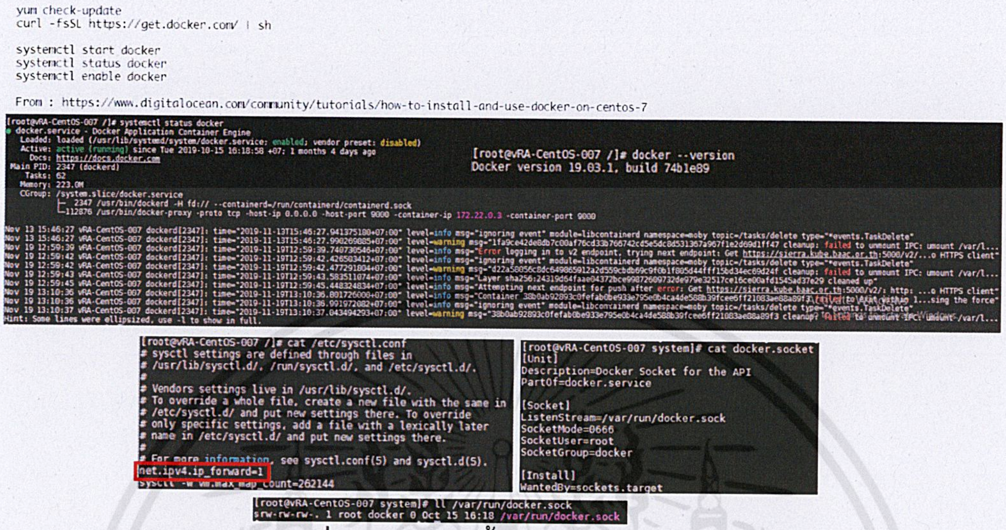
9) Stage Robot Selenium Test (DEV, SIT, UAT, PROD): เป็นการนำไฟล์ Test script ที่ถูกเขียนไว้ล่วงหน้ามาทำงาน ว่าได้ผลตามที่คาดหวังหรือไม่

10) Stage JMeter Test (DEV, SIT, UAT, PROD): เป็นการนำไฟล์ Performance Test Script ที่ถูกเขียนไว้ล่วงหน้ามาทำงานว่าสามารถรับโหลดได้ตามที่คาดหวังหรือไม่

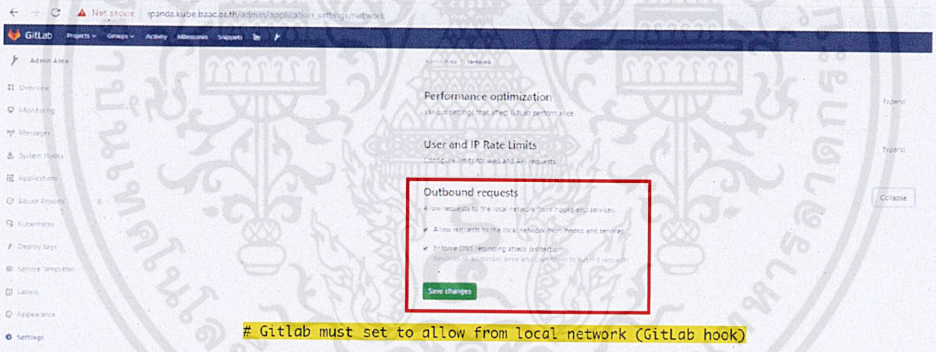
11) Approve for Production: เป็นการส่งอีเมลไปหาผู้ที่เกี่ยวข้องและสามารถอนุมัติให้ดำเนินการ Deploy ไปยัง Production environment ได้

3.5.5 อธิบายการทำงาน Pipeline ภาษา Go

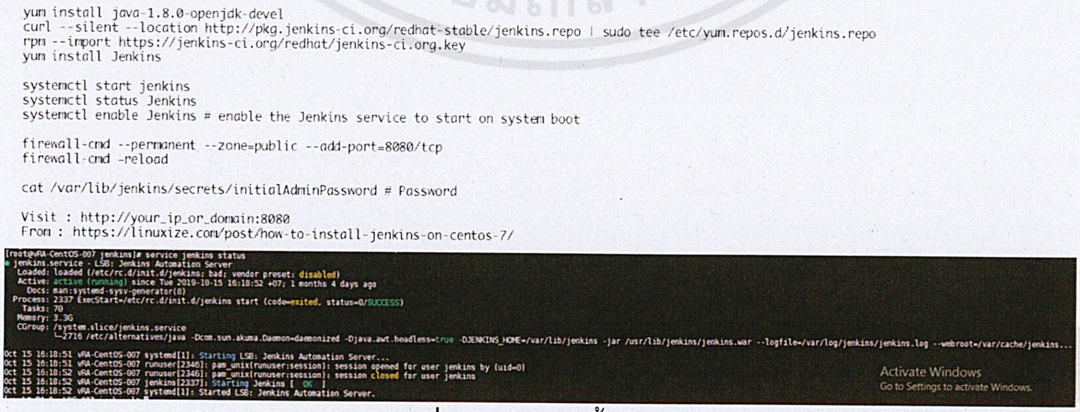
เนื่องจาก Pipeline ของทุกภาษาจะมีหลักการทำงานที่คล้ายกัน มีความแตกต่างกันในบาง Stage ในหัวข้อนี้จะอธิบายการทำงานของ Pipeline ตั้งแต่เริ่มต้นถึงสิ้นสุดสำหรับภาษา Go



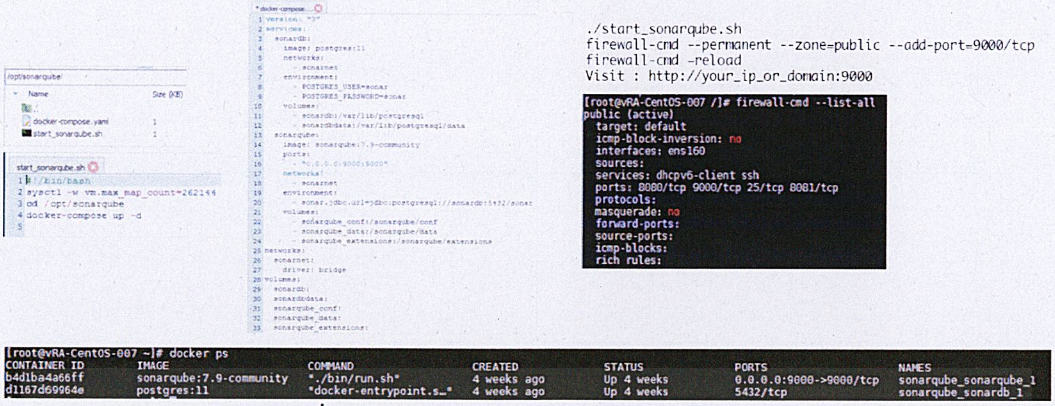
ภาพที่ 3.90 การติดตั้ง Docker บน Jenkins



ภาพที่ 3.91 การตั้งค่า GitLab Hook

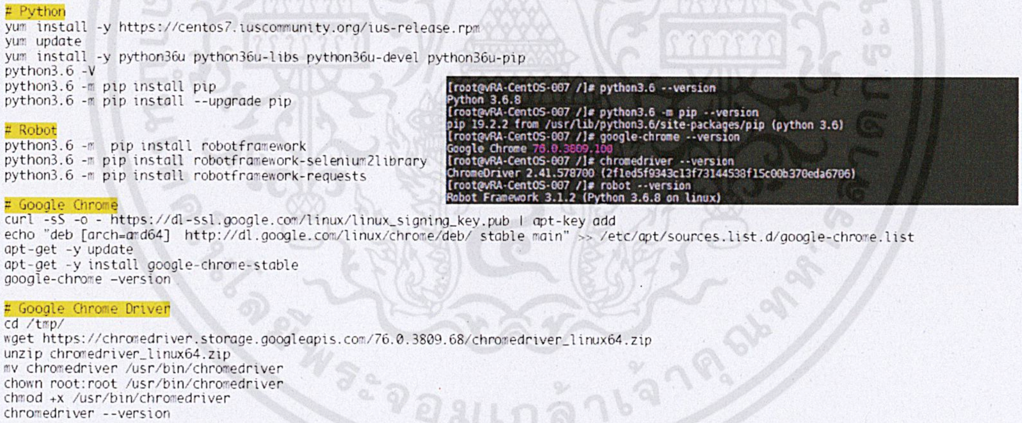


ภาพที่ 3.92 การติดตั้ง Jenkins

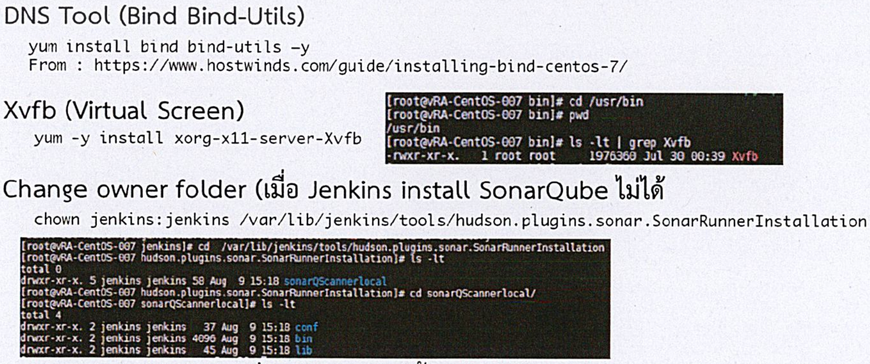


ภาพที่ 3.93 การใช้งาน SonarQube ด้วย Docker

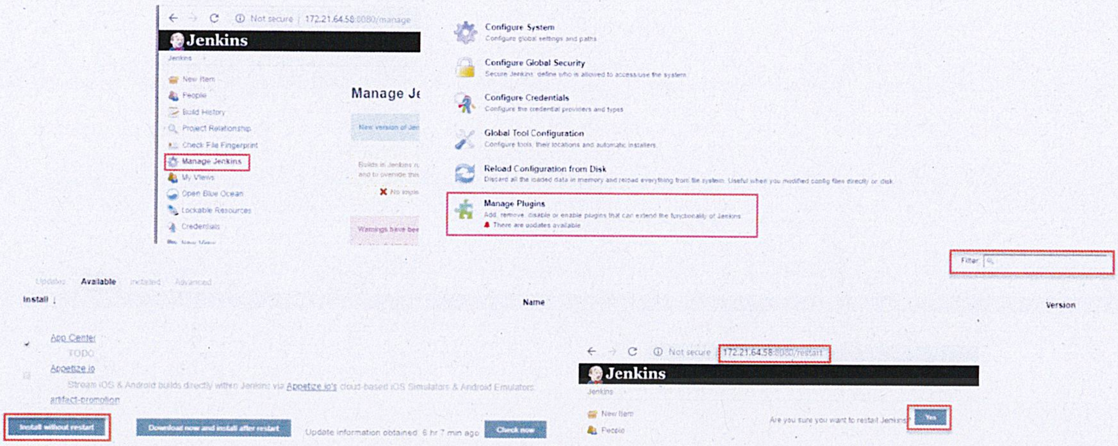
ภาพที่ 3.94 การติดตั้ง Apache JMeter



ภาพที่ 3.95 การติดตั้ง Robot Framework

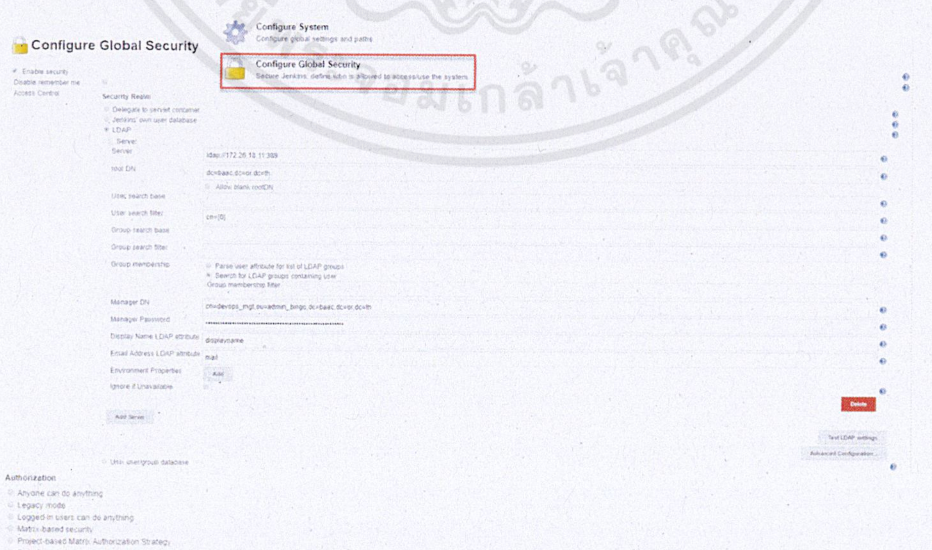


ภาพที่ 3.96 การติดตั้ง DNS Tool และ Xvfb

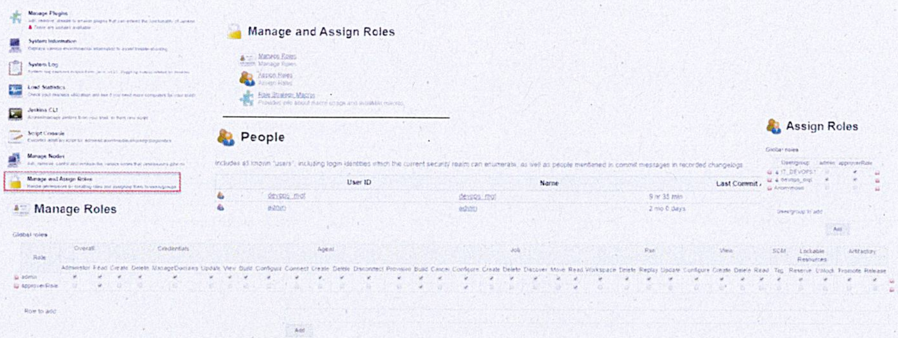


ภาพที่ 3.97 การติดตั้ง Plugin บน Jenkins

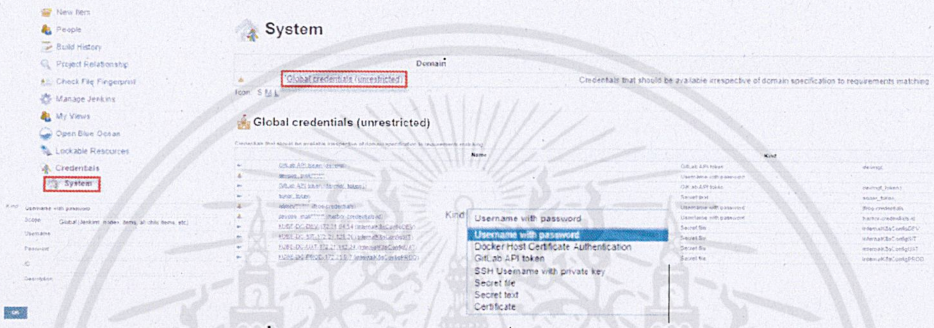
ภาพที่ 3.98 Plugins จำเป็นที่ใช้บน Jenkins



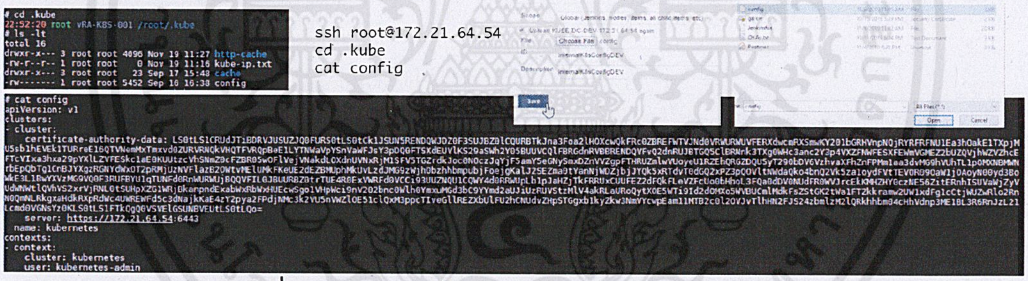
ภาพที่ 3.99 การ Config Security บน Jenkins



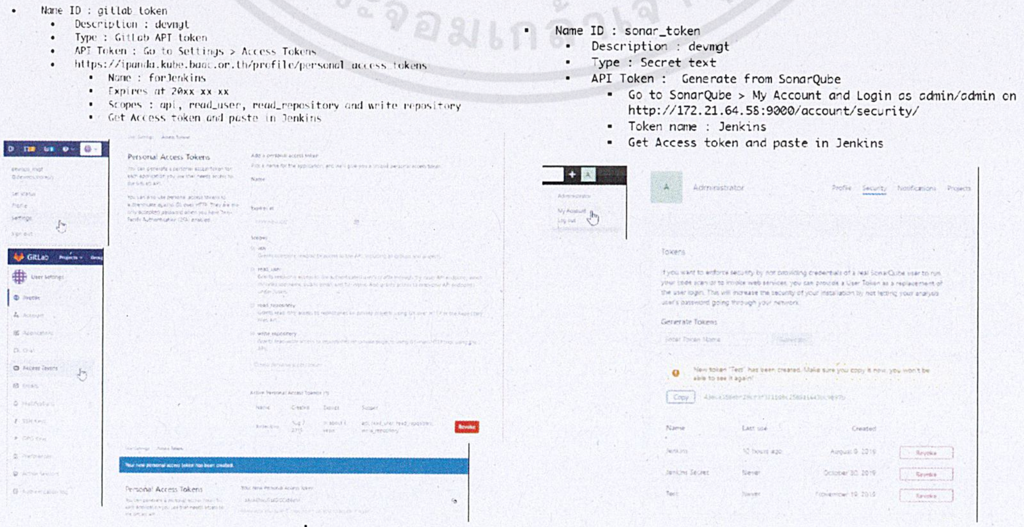
ภาพที่ 3.100 การเพิ่ม Role ให้ User บน Jenkins



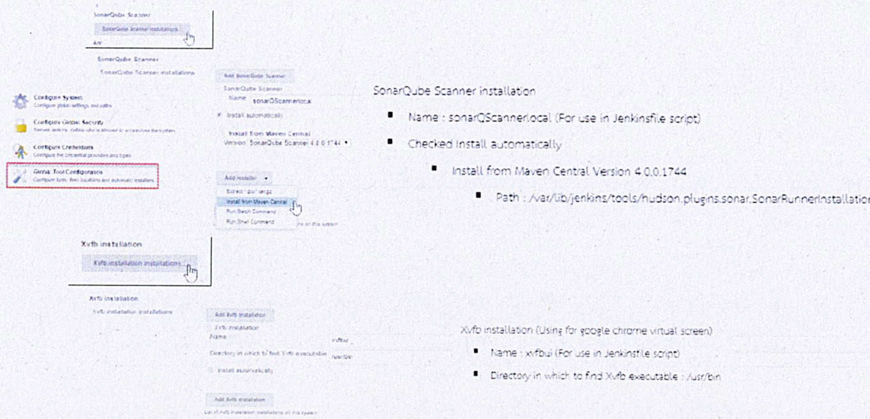
ภาพที่ 3.101 Credential ที่จำเป็น บน Jenkins



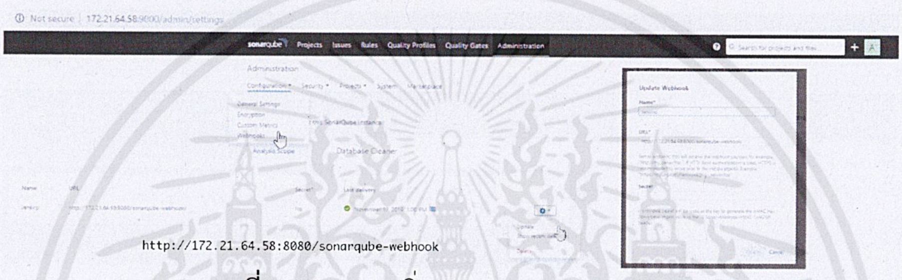
ภาพที่ 3.102 การแก้ไข Config Key ของ Kubernetes



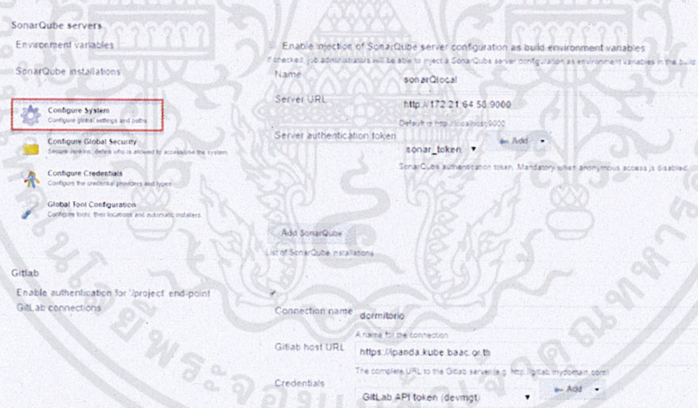
ภาพที่ 3.103 GitLab Token และ Sonar Token



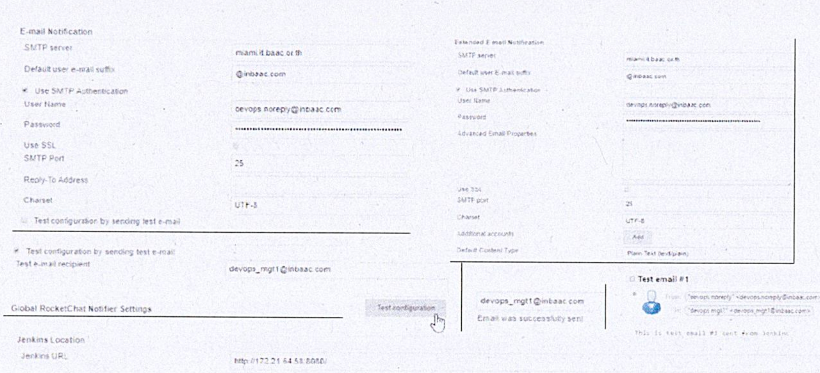
ภาพที่ 3.104 SonarQube & Xvfb Tools บน Jenkins



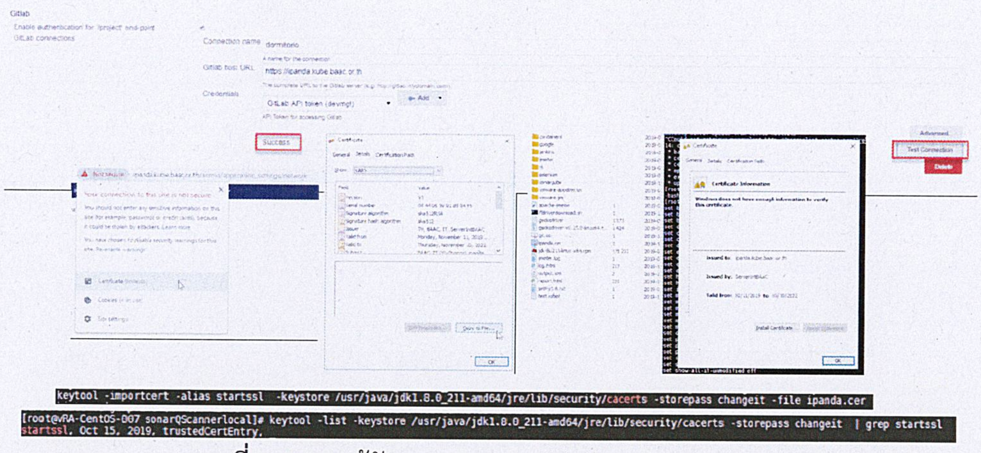
ภาพที่ 3.105 การเพิ่ม SonarQube Webhook



ภาพที่ 3.106 การตั้งค่า Server ของ SonarQune และ GitLab



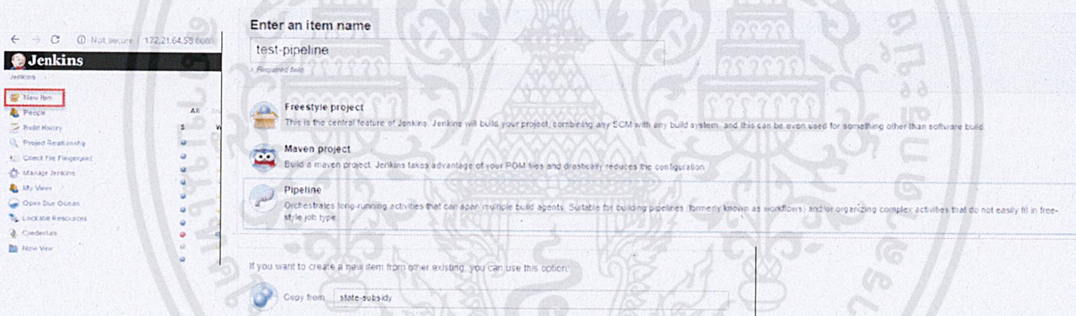
ภาพที่ 3.107 การตั้งค่า E-mail และการทดสอบการส่งบน Jenkins



ภาพที่ 3.108 แก้ปัญหา GitLab Certificate บน Jenkins

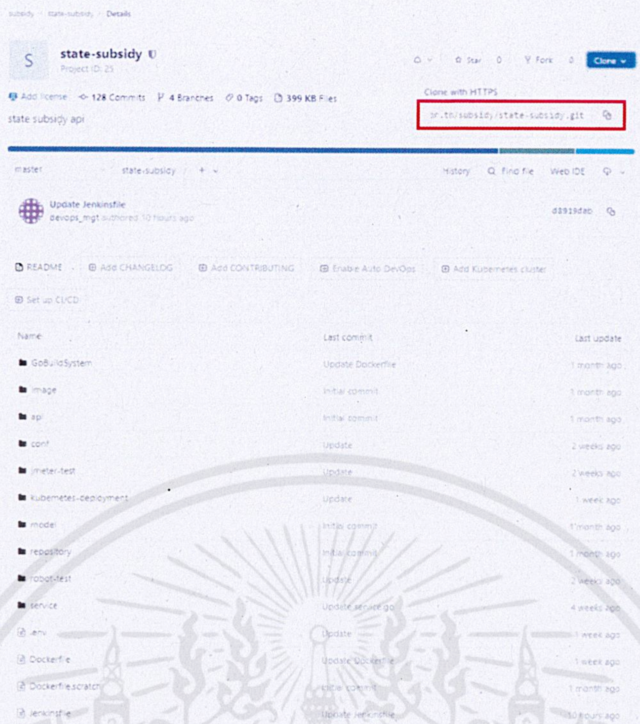
ภาพที่ 3.109 กระบวนการ Pipeline บน Jenkins

การทำงานของ Pipeline ประกอบไปด้วยหลาย Stage และจะอธิบายตามกรอบสีน้ำเงิน



ภาพที่ 3.110 การสร้าง Job บนเครื่อง Jenkins

สำหรับการสร้าง Job บน Jenkins สามารถกด New Item และตั้งชื่อ Job โดยสามารถคัดลอกการตั้งค่าของ Job อื่นๆมาใช้กับ Job ใหม่ที่สร้างขึ้นมาได้



ภาพที่ 3.111 โปรเจค Go บน GitLab

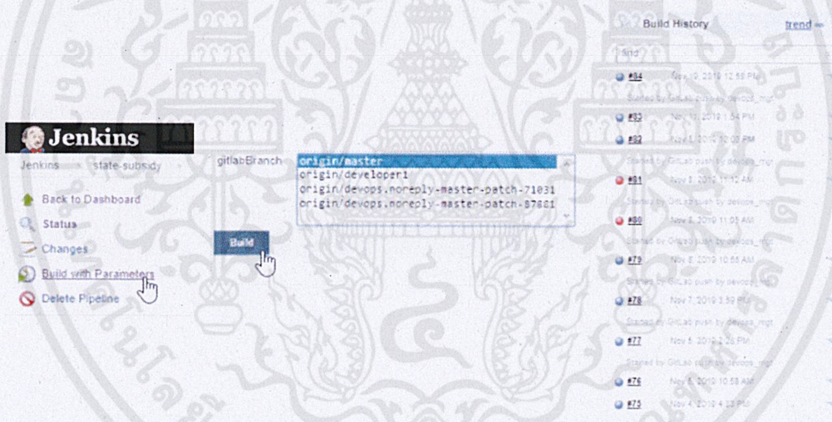
Source Code ของโปรเจค Go (API) จะถูกเก็บอยู่บน GitLab โดยสามารถ Clone มาทำงานบนเครื่องนักพัฒนาและ Clone ไปทำงานบนเครื่อง Jenkins ได้

ภาพที่ 3.112 การกำหนดชื่อของ Git Parameter บน Jenkins

การตั้งค่า Job บนเครื่อง Jenkins สามารถใช้ Plugin ที่ชื่อ Git Parameter โดยจะตั้งชื่อตัวแปรสำหรับเลือก Branch ที่จะนำมา Build โดยใช้ชื่อตัวแปรว่า gitlabBranch ซึ่งเป็นตัวแปรเดียวกับตอนที่ GitLab ส่ง Hook มาให้ Jenkins เมื่อเกิด Event การ Push

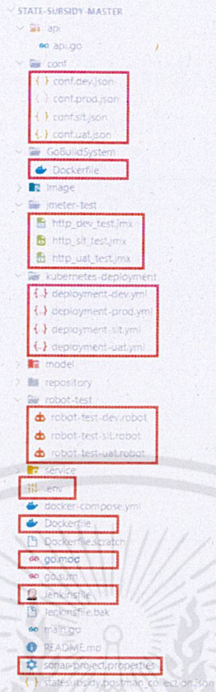
ภาพที่ 3.113 รายละเอียดของ Job บนเครื่อง Jenkins

เมื่อใส่ Git URL ของโปรเจคที่ต้องการและใส่ Branch ที่จะถูก Build โดยใช้ชื่อตัวแปร ${gitlabBranch} เรียบร้อยแล้ว ให้นำเครื่องหมายออกจากตรง Lightweight checkout เนื่องจากจะทำให้ไม่สามารถอ่านค่าตัวแปรได้และทำให้เกิด Error



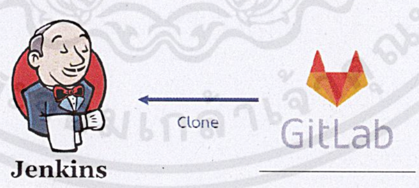
ภาพที่ 3.114 การ Build Job บน Jenkins

เมื่อตั้งค่าเรียบร้อย และทดลองกด Build with Parameters ตัว Plugin จะดึงรายชื่อ Branch ทั้งหมดบน Repository ของ URL ที่เราใส่เข้ามา ให้เราเลือกว่าต้องการ Build จาก Branch อะไร โดยบนเครื่อง Jenkins จะบอกว่าการ Build ครั้งนี้มาจากการกด Build Manual หรือเป็นการ Trigger มาจากบน GitLab



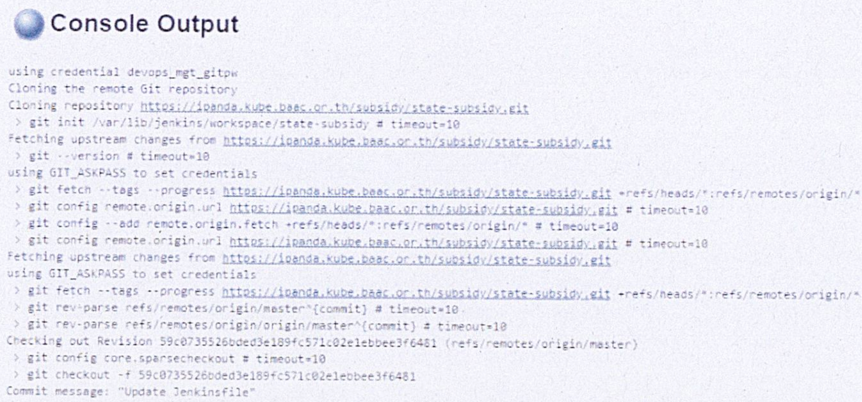
ภาพที่ 3.115 โครงสร้างโปรเจคและไฟล์ต่างๆใน Project Go

โครงสร้างโปรเจคของภาษา Go จะประกอบไปด้วยไฟล์ต่างๆที่จำเป็นในการทำงานของ Pipeline บน Jenkins เช่น ในโฟลเดอร์ conf จะเป็นไฟล์ตั้งค่าฐานข้อมูลสำหรับแต่ละ Environment, โฟลเดอร์ GoBuildSystem จะเป็น Dockerfile ที่ใช้สำหรับเป็น Agent ครอบการทำงาน, โฟลเดอร์ jmeter-test จะเก็บไฟล์ Test ของ JMeter, โฟลเดอร์ kubernetes-deployment จะเป็นไฟล์ที่ใช้สำหรับ Deploy ไปยัง Environment ต่างๆ, โฟลเดอร์ robot-test เป็นไฟล์ Robot ที่ใช้สำหรับ Test แต่ละ Environment, ไฟล์ .env ใช้สำหรับเก็บค่าตัวแปรและค่า Config ต่างๆที่ใช้ใน Jenkins Pipeline, ไฟล์ Jenkinsfile ใช้เก็บ Pipeline Code Jenkins จะเข้ามาอ่านที่ไฟล์นี้

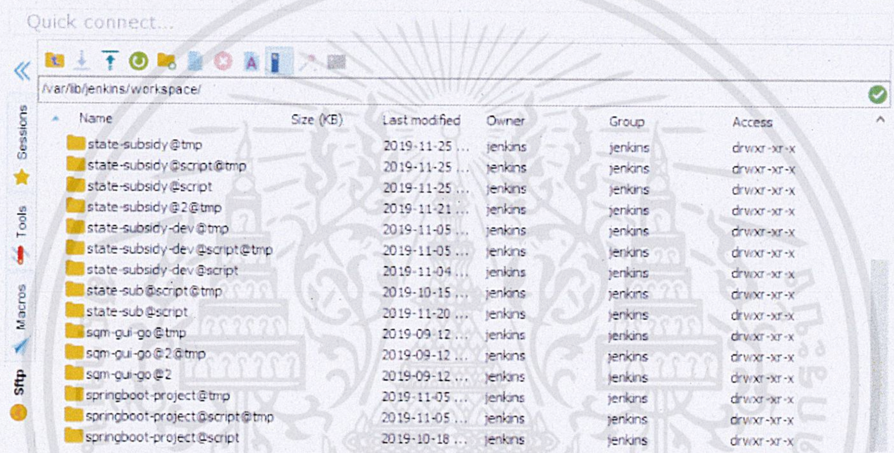


ภาพที่ 3.116 การ Clone Source Code จาก GitLab มายัง Jenkins Workspace

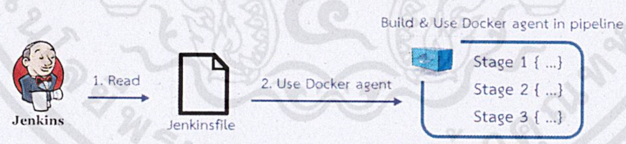
Jenkins เมื่อได้รับคำสั่งให้เริ่ม Build จะทำการไปดึง Source Code ตาม Branch ที่กำหนดมาเก็บไว้ที่ Workspace บนเครื่อง Jenkins



ภาพที่ 3.117 ผลลัพธ์การ Clone Code มายัง Jenkins Workspace

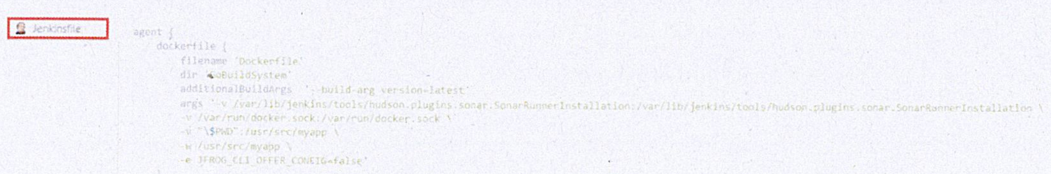


ภาพที่ 3.118 ผลลัพธ์การสร้างโฟลเดอร์ใน Jenkins Workspace

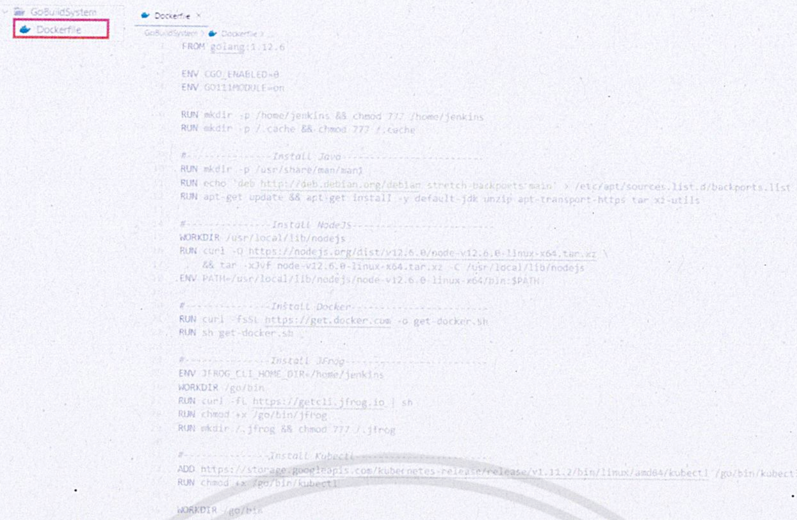


ภาพที่ 3.119 การอ่าน Dockerfile ของ Jenkins เพื่อใช้เป็น Docker Agent

เมื่อ Jenkins ทำการ Clone Source Code มาเรียบร้อยแล้วจะทำการอ่าน Jenkinsfile ที่ Clone มาโดยใน Jenkinsfile ด้านบนจะบอกว่า Pipeline นี้จะใช้ Docker Agent เป็นสภาพแวดล้อม

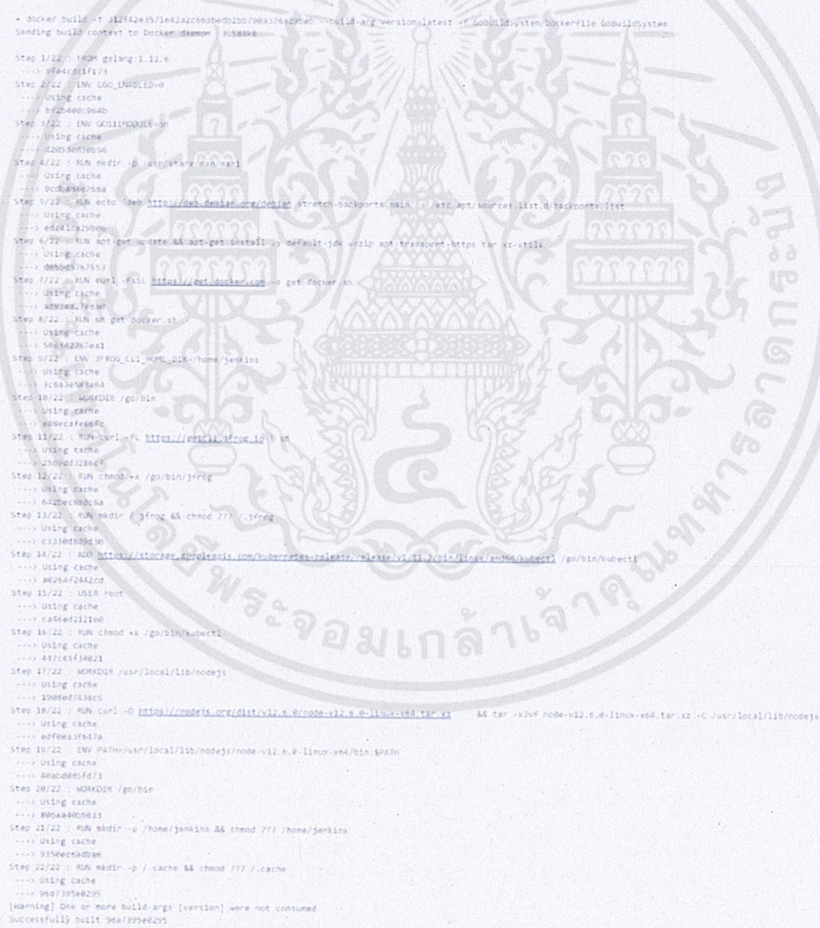


ภาพที่ 3.120 Stage ใน Pipeline การใช้ Docker Agent

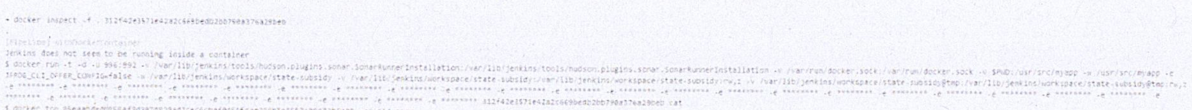


```
FROM golang:1.12.6

ENV CGO_ENABLED=0
ENV GO111MODULE=on

RUN mkdir -p /home/jenkins && chmod 777 /home/jenkins
RUN mkdir -p /.cache && chmod 777 /.cache

#----------------Install Java-----------------------
RUN mkdir -p /usr/share/man/man1
RUN echo 'deb http://deb.debian.org/debian stretch-backports main' > /etc/apt/sources.list.d/backports.list
RUN apt-get update && apt-get install -y default-jdk unzip apt-transport-https tar xz-utils

#----------------Install NodeJS-----------------------
WORKDIR /usr/local/lib/nodejs
RUN curl -O https://nodejs.org/dist/v12.6.0/node-v12.6.0-linux-x64.tar.xz \
    && tar -xJvf node-v12.6.0-linux-x64.tar.xz -C /usr/local/lib/nodejs
ENV PATH=/usr/local/lib/nodejs/node-v12.6.0-linux-x64/bin:$PATH

#----------------Install Docker-----------------------
RUN curl -fsSL https://get.docker.com -o get-docker.sh
RUN sh get-docker.sh

#----------------Install JFrog-----------------------
ENV JFROG_CLI_HOME_DIR=/home/jenkins
WORKDIR /go/bin
RUN curl -fL https://getcli.jfrog.io | sh
RUN chmod +x /go/bin/jfrog
RUN mkdir /.jfrog && chmod 777 /.jfrog

#----------------Install Kubectl-----------------------
ADD https://storage.googleapis.com/kubernetes-release/release/v1.11.2/bin/linux/amd64/kubectl /go/bin/kubectl
RUN chmod +x /go/bin/kubectl

WORKDIR /go/bin
```

ภาพที่ 3.121 Dockerfile ที่ใช้เป็น Docker Agent ในโปรเจคภาษา Go



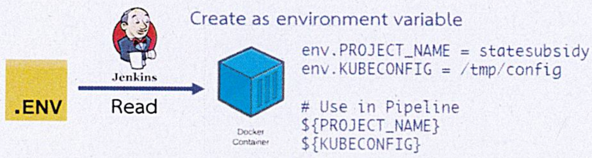
ภาพที่ 3.122 ผลลัพธ์การ Build Image ของ Docker Agent



ภาพที่ 3.123 ผลลัพธ์การทำงานของ Docker Agent

3.5.5.1 Stage Initial Environment

เมื่อเข้ามาทำงาน Stage นี้ Jenkins จะอ่านค่าในไฟล์ .env ที่มีลักษณะเป็น Key-Value มาสร้างเป็นตัวแปรโดยนำเอา Key มาเป็นชื่อตัวแปรและนำเอา Value มาเป็นค่าในตัวแปร จากนั้นสามารถอ้างอิงถึงโดยใช้ ${ชื่อตัวแปร}

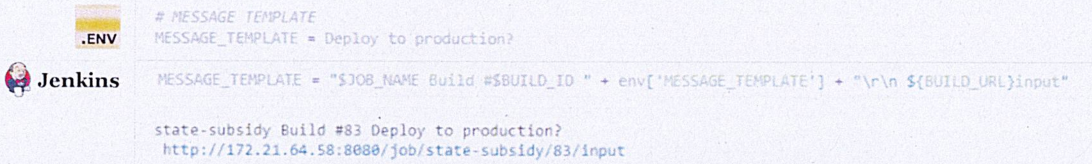



```
def prop = readProperties  file:'.env'
// prop.each{ k,v -> println "$k = $v" }
prop.each{ k,v -> env."${k}" = "${v}" }
```

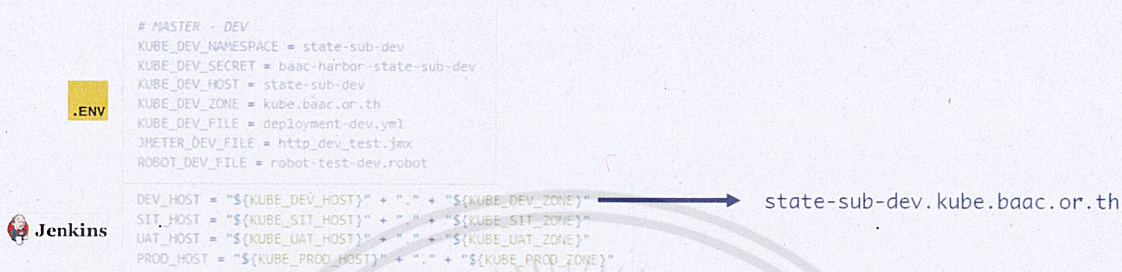
ภาพที่ 3.124 การดึงค่าจากไฟล์ .env มาใช้งานใน Pipeline

```
.env
PROJECT_NAME = statesubsidy
PROJECT_VERSION = 0.2.0

KUBECONFIG = /tmp/config

KUBE_APP_NAME = state-sub-app
KUBE_CONTAINER_PORT = 9016
KUBE_SERVICE_PORT = 9016

# KUBE CREDENTIAL NAME
KUBECONFIG_DEV = internalK8sConfigDEV
KUBECONFIG_SIT = internalK8sConfigSIT
KUBECONFIG_UAT = internalK8sConfigUAT
KUBECONFIG_PROD = internalK8sConfigPROD

# MASTER - DEV
KUBE_DEV_NAMESPACE = state-sub-dev
KUBE_DEV_SECRET = baac-harbor-state-sub-dev
KUBE_DEV_HOST = state-sub-dev
KUBE_DEV_ZONE = kube.baac.or.th
KUBE_DEV_FILE = deployment-dev.yml
JMETER_DEV_FILE = http_dev_test.jmx
ROBOT_DEV_FILE = robot-test-dev.robot

# DEVELOPER - DEV
KUBE_DEV_BRANCH_NAMESPACE = state-sub-dev-developer1
KUBE_DEV_BRANCH_SECRET = baac-harbor-state-sub-dev-developer1
KUBE_DEV_BRANCH_HOST = state-sub-dev-developer1
KUBE_DEV_BRANCH_ZONE = kube.baac.or.th

# PIPELINE NAME
DNS_PIPELINE = dynamic_dns_pipeline
JMETER_PIPELINE = jmeter-pipeline
ROBOT_PIPELINE = robot-pipeline

# MESSAGE TEMPLATE
MESSAGE_TEMPLATE = Deploy to production?

# ROCKET CHAT
ROCKET_URL = https://amigo.kube.baac.or.th/
ROCKET_TOKEN = https://amigo.kube.baac.or.th/hooks/nshDgCq2NdzEf57f5/fbuLYfbX9SdgQSakfPiHrid3jywtKwDwxiXPx56ej2aWduHX

# APPROVAL
APPROVE_1 = 8340221
APPROVE_2 = 8340221,devops_mgt

# JFROG CONFIG
JFROG_ARTIFACTORY_URL = http://rio.kube.baac.or.th:8081/artifactory
JFROG_LOCAL_REPO = go-local
JFROG_VIRTUAL_REPO = go-virtual
JFROG_CREDENTIALS = jfrog-credentials

# HARBOR CONFIG
DOCKER_REGISTRY_PROTOCAL = https://
DOCKER_REGISTRY_URL = sierra.kube.baac.or.th:5000
DOCKER_REGISTRY_CREDENTIALS = harbor-credentials-id
DOCKER_REGISTRY_BASE = baacimage

# MASTER BRANCH NAME
BRANCH_MASTER_NAME = origin/master
```

ภาพที่ 3.125 ตัวอย่างไฟล์ .env ของภาษา Go

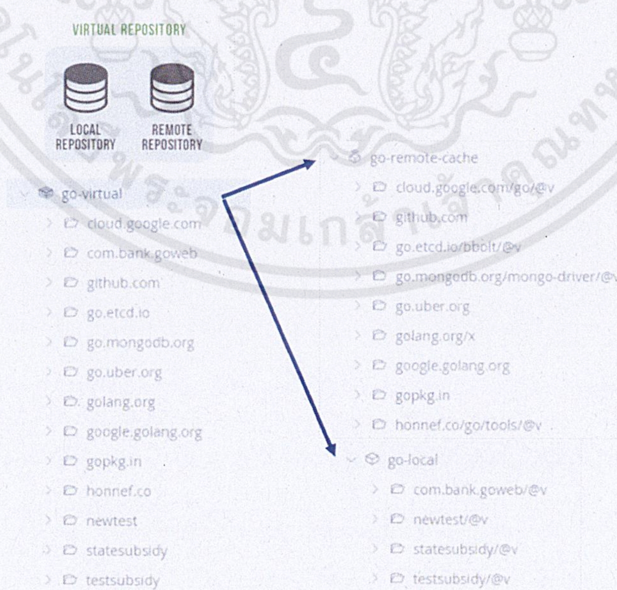


ภาพที่ 3.126 การตั้งค่า Message ของ Email ใน Pipeline

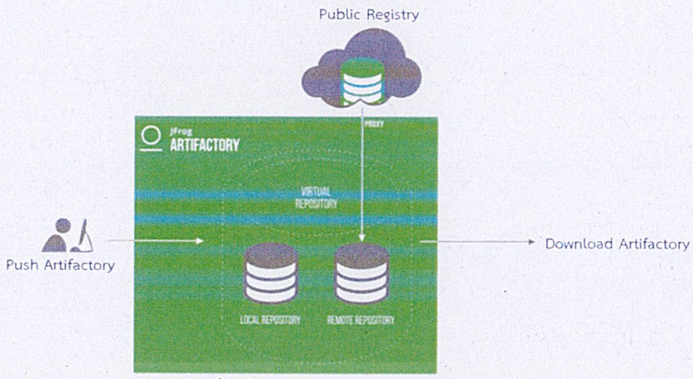


ภาพที่ 3.127 การนำเอา Host และ Zone มาตั้งเป็นตัวแปรใหม่

ภาพที่ 3.128 การกำหนดชื่อโปรเจคและเวอร์ชั่นสำหรับการทำงาน



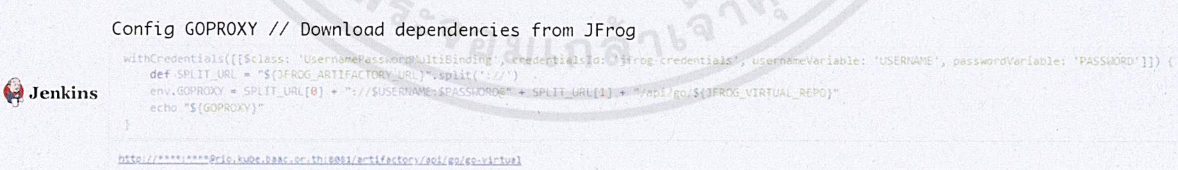
ภาพที่ 3.129 การทำงานของ Virtual Repository บน JFrog



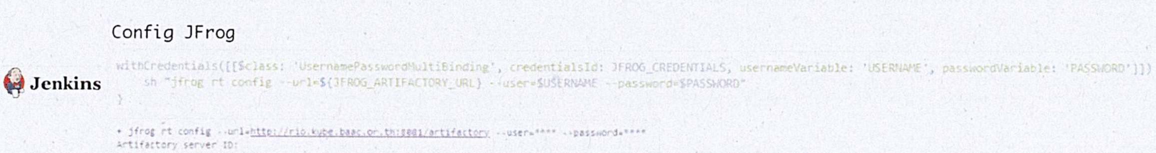
ภาพที่ 3.130 การทำงานของ JFrog

ภาพที่ 3.131 การสร้าง Repository บน JFrog

ภาพที่ 3.132 การตั้งค่า JFrog ใน Pipeline



ภาพที่ 3.133 การตั้งค่า Go Proxy เพื่อดาวน์โหลด Library จาก JFrog



ภาพที่ 3.134 การตั้งค่า Credential ของ JFrog

3.5.5.2 Stage Test

Jenkins

```
stage('Test') {
    steps {
        sh "go test ./service -v -coverprofile=coverage.out"
```

ภาพที่ 3.135 การทำ Unit Test ของภาษา Go

```
service
    service_test.go
    service.go
```

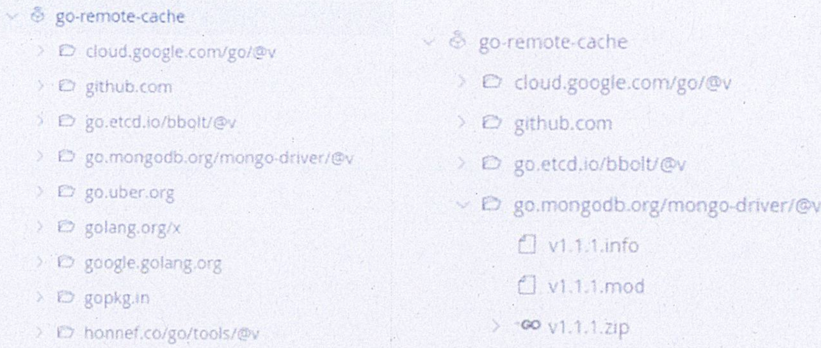
```go
func Mapstatus(status string) string {

    var statuscode = map[string]string{
        "สำเร็จ": "01",
        "ไม่สำเร็จ/เงินในบัญชีไม่พอ":                    "02",
        "ไม่สำเร็จ/บัญชีถูกอายัด":                       "03",
        "ไม่สำเร็จ/บัญชีปิด":                           "04",
        "ไม่สำเร็จ/บัญชีไม่เคลื่อนไหว/ติดข้อจำกัดบัญชี": "05",
        "ไม่สำเร็จ/เลขบัญชีผิด":                         "06",
    }

    return statuscode[status]

}
```

```go
service > service_test.go
package service_test

import (
    "statesubsidy/model"
    "statesubsidy/service"
    "testing"

    "github.com/stretchr/testify/assert"
)

func Test_Mapstatus_input_success_shouldbe_01(t *testing.T) {
    input := "สำเร็จ"
    expected := "01"
    actual := service.Mapstatus(input)
    assert.Equal(t, expected, actual, "actual but it got expected")
}

func Test_Mapstatus_input_notsuccess_because_not_enough_money_in_account_shouldbe_02(t *testing.T) {
    input := "ไม่สำเร็จ/เงินในบัญชีไม่พอ"
    expected := "02"
    actual := service.Mapstatus(input)
    assert.Equal(t, expected, actual, "actual but it got expected")
}

func Test_Mapstatus_input_notsuccess_because_Account_freeze_shouldbe_03(t *testing.T) {
    input := "ไม่สำเร็จ/บัญชีถูกอายัด"
    expected := "03"
    actual := service.Mapstatus(input)
    assert.Equal(t, expected, actual, "actual but it got expected")
}

func Test_Mapstatus_input_notsuccess_because_Account_closed_shouldbe_04(t *testing.T) {
    input := "ไม่สำเร็จ/บัญชีปิด"
    expected := "04"
    actual := service.Mapstatus(input)
    assert.Equal(t, expected, actual, "actual but it got expected")
}

func Test_Mapstatus_input_notsuccess_because_Account_is_inactive_shouldbe_05(t *testing.T) {
    input := "ไม่สำเร็จ/บัญชีไม่เคลื่อนไหว/ติดข้อจำกัดบัญชี"
    expected := "05"
    actual := service.Mapstatus(input)
    assert.Equal(t, expected, actual, "actual but it got expected")
}
```

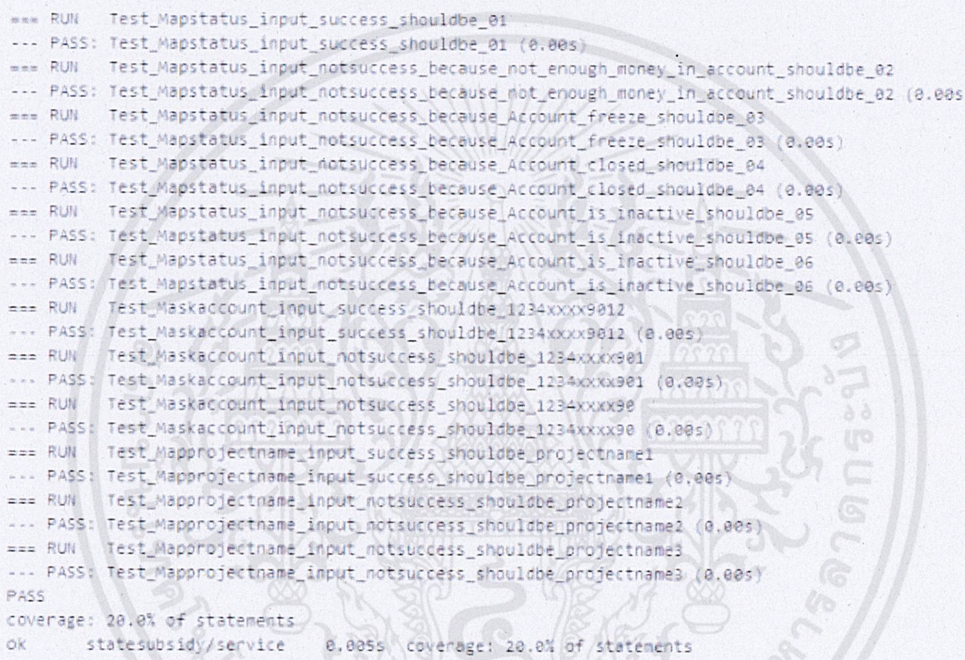
ภาพที่ 3.136 ฟังก์ชั่น Unit Test ในภาษา Go

```
+ go test ./service -v
go: finding github.com/xdg/stringprep v1.0.0
go: finding go.mongodb.org/mongo-driver v1.1.1
go: finding github.com/gorilla/mux v1.7.3
go: finding gopkg.in/mgo.v2 v2.0.0-20190816093944-a6b53ec6cb22
go: finding github.com/rs/cors v1.7.0
go: finding github.com/golang/snappy v0.0.1
go: finding github.com/xdg/scram v0.0.0-20180814205039-7eeb5667e42c
go: finding github.com/spf13/viper v1.4.0
go: finding github.com/gorilla/handlers v1.4.2
go: finding github.com/stretchr/testify v1.2.2
go: finding github.com/hashicorp/hcl v1.0.0
go: finding github.com/prometheus/client_golang v0.9.3
go: finding github.com/gorilla/websocket v1.4.0
go: finding github.com/coreos/go-semver v0.2.0
go: finding golang.org/x/net v0.0.0-20190522155817-f3200d17e092
go: finding github.com/jonboulle/clockwork v0.1.0
go: finding github.com/spf13/jwalterweatherman v1.0.0
go: finding github.com/spf13/afero v1.1.2
go: finding github.com/mitchellh/mapstructure v1.1.2
go: finding go.etcd.io/bbolt v1.3.2
go: extracting github.com/xdg/stringprep v1.0.0
go: extracting github.com/go-stack/stack v1.8.0
go: extracting golang.org/x/sync v0.0.0-20181221193216-37e7f081c4d4
go: extracting github.com/xdg/scram v0.0.0-20180814205039-7eeb5667e42c
go: extracting github.com/golang/snappy v0.0.1
go: downloading golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2
go: extracting golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2
go: extracting golang.org/x/text v0.3.0
```

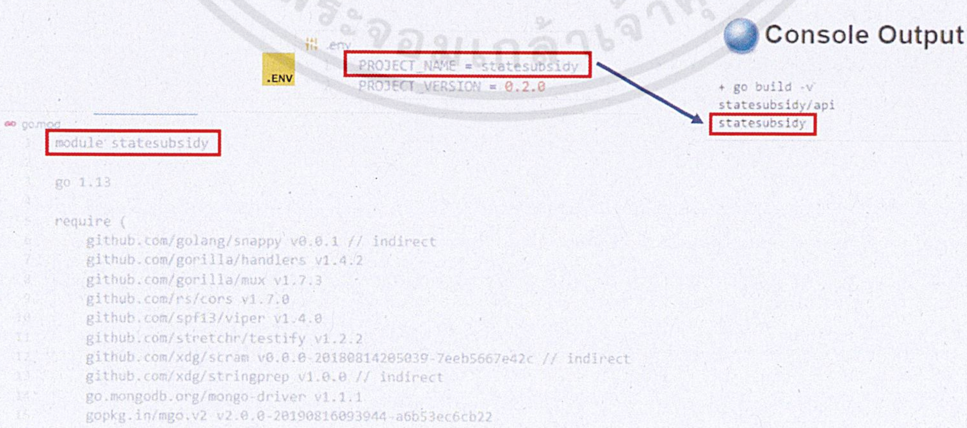
ภาพที่ 3.137 ผลลัพธ์การโหลด Library ต่างๆจาก JFrog



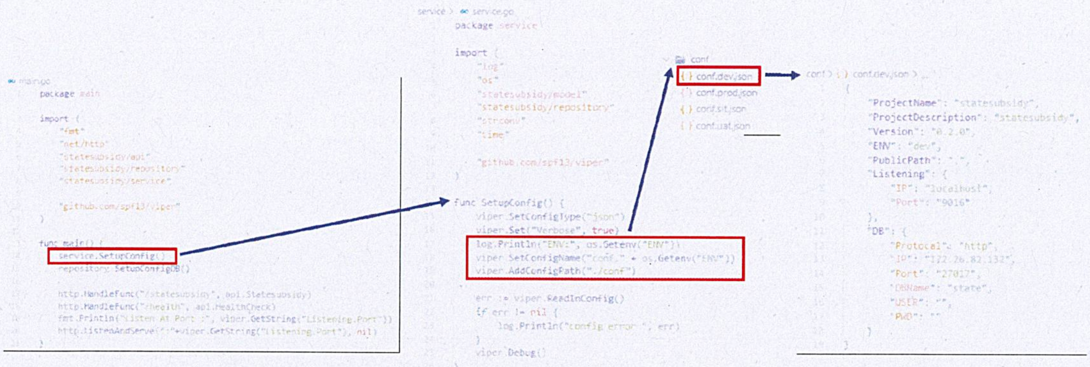
ภาพที่ 3.138 ผลลัพธ์การโหลด Library ต่างๆมา Cache บน JFrog



ภาพที่ 3.139 ผลลัพธ์การทดสอบ Unit Test



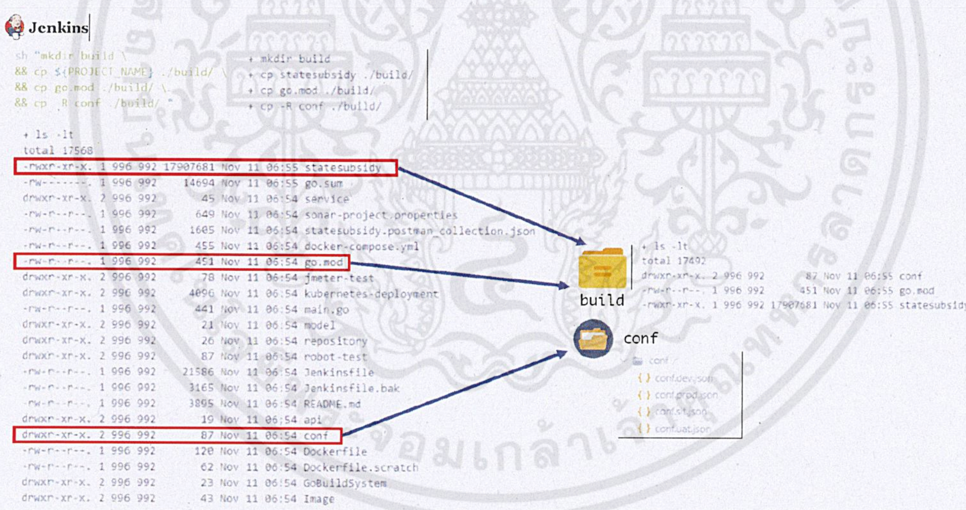
ภาพที่ 3.140 การดูชื่อ Artifact เมื่อ Build สำเร็จจากไฟล์ go.mod



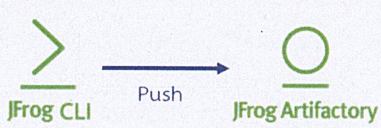
ภาพที่ 3.141 การอ่าน Environment Config ของโปรเจค Go

3.5.5.3 Stage Build

ภาพที่ 3.142 การ Build ของภาษา Go

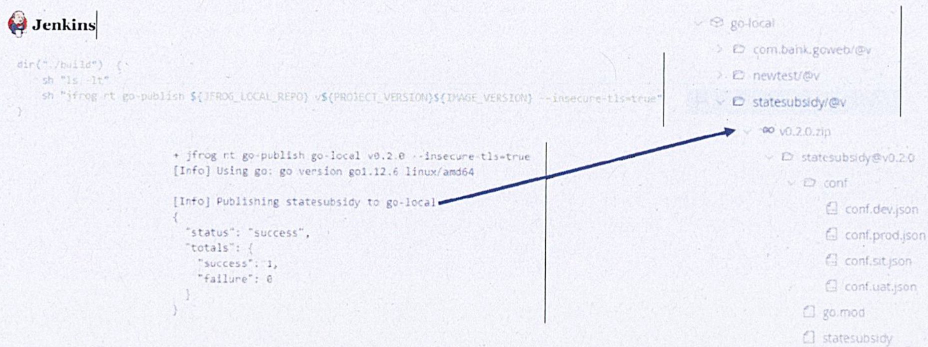


ภาพที่ 3.143 การนำไฟล์ที่ Build เข้าโฟลเดอร์เพื่อพร้อมสำหรับทำ Artifact



ภาพที่ 3.144 การใช้ JFrog CLI Push ไปเก็บบน JFrog Artifactory

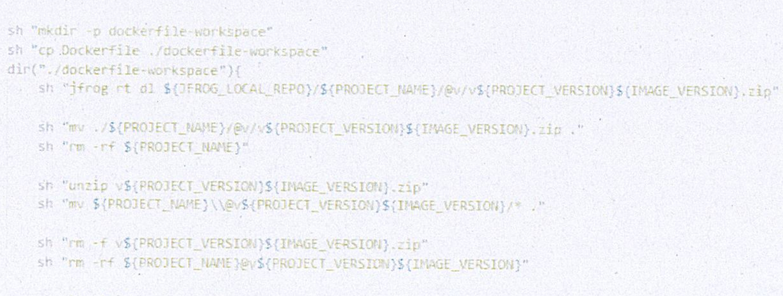
3.5.5.4 Stage Push to Artifactory



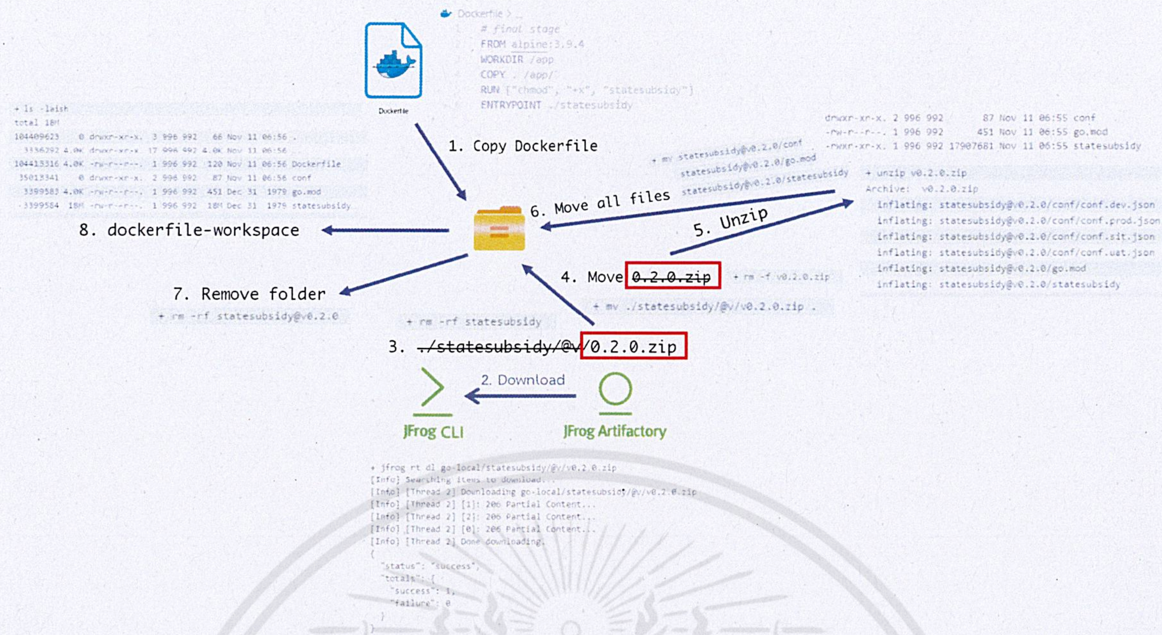
ภาพที่ 3.145 ผลลัพธ์การ Push ไปยัง JFrog

ภาพที่ 3.146 ผลลัพธ์ Artifact บน JFrog

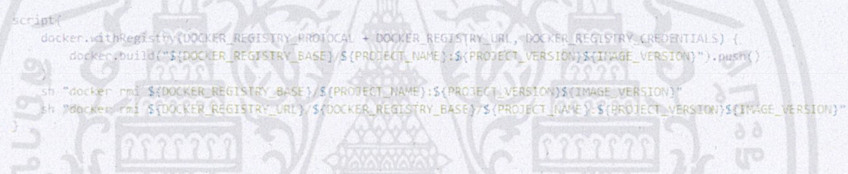
3.5.5.5 Stage Create Image & Push to Harbor Image Registry



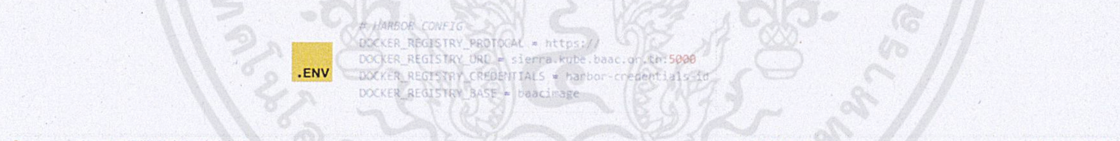
ภาพที่ 3.147 การคัดลอกไฟล์เพื่อใช้สำหรับสร้าง Image



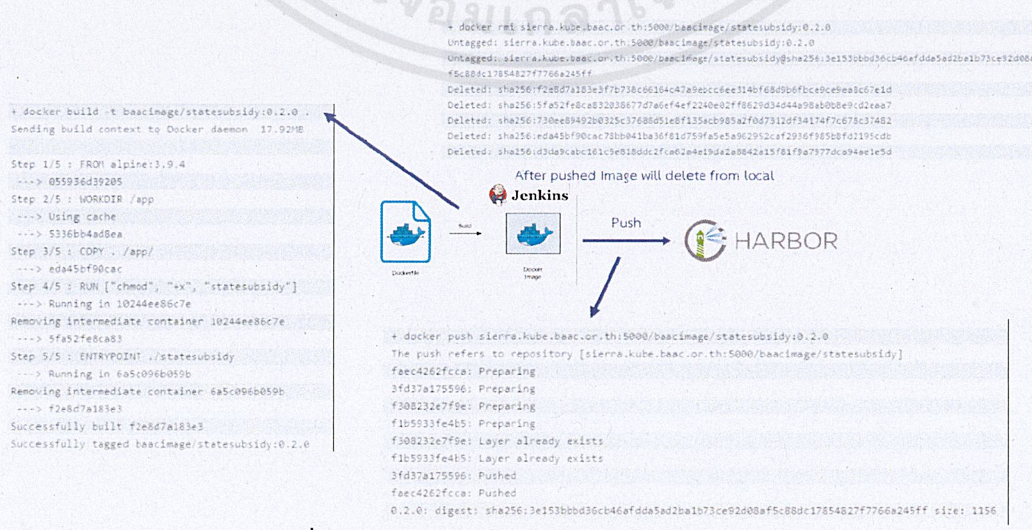
ภาพที่ 3.148 ลำดับการทำงานของการ Build Image



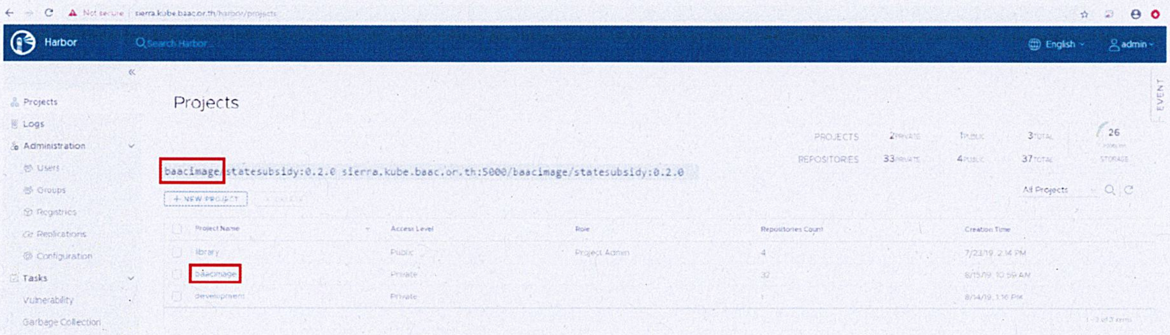
ภาพที่ 3.149 การ Push Image ไปเก็บบน Harbor Registry



ภาพที่ 3.150 ตัวแปรที่ใช้สำหรับการเชื่อมต่อไปยัง Harbor Registry



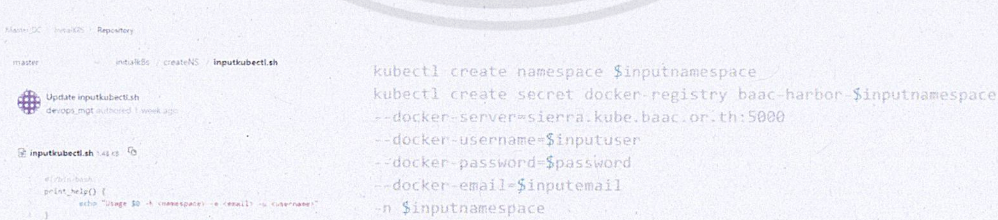
ภาพที่ 3.151 ผลลัพธ์การเก็บ Image ไปยัง Harbor



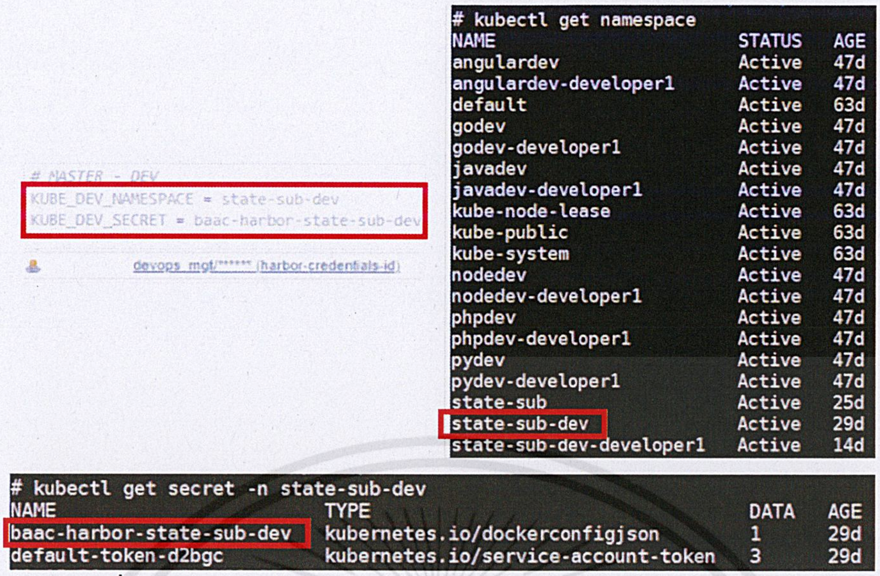
ภาพที่ 3.152 Project บน Harbor

ภาพที่ 3.153 Repository บน Harbor

ภาพที่ 3.154 Tag ของ Image บน Harbor



ภาพที่ 3.155 สคริปต์สำหรับสร้าง Namespace บน Kubernetes

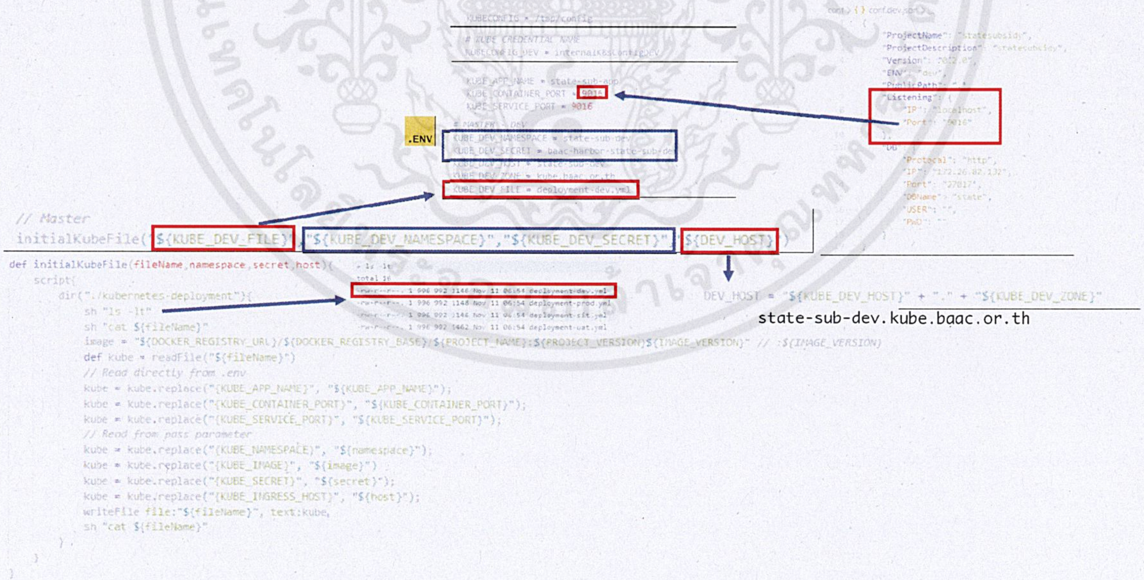


ภาพที่ 3.156 ผลการสร้าง Namespace และ Secret บน Kubernetes

3.5.5.6 Deploy to Kubernetes (DEV)

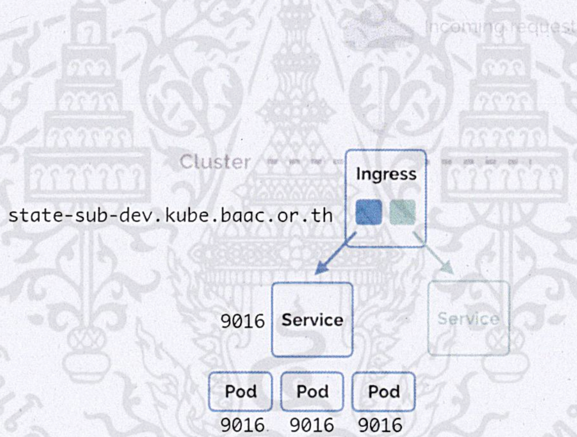
```
initialKubeFile("${KUBE_DEV_FILE}","${KUBE_DEV_NAMESPACE}","${KUBE_DEV_SECRET}","${DEV_HOST}")
deployToKubernetes("${KUBE_DEV_FILE}","${KUBECONFIG_DEV}")
dnsUpdate("${KUBE_DEV_HOST}","${KUBE_DEV_ZONE}","${KUBE_IP}")
```

ภาพที่ 3.157 คำสั่งใช้สำหรับการ Deploy ไปยัง Kubernetes

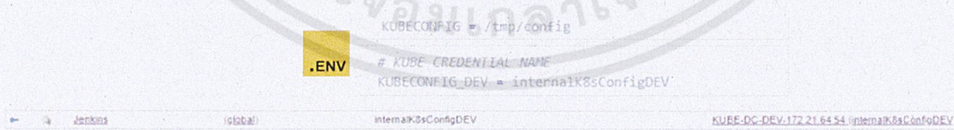


ภาพที่ 3.158 ฟังก์ชั่นแก้ไขไฟล์ Deployment ให้เป็นตัวแปรที่กำหนด

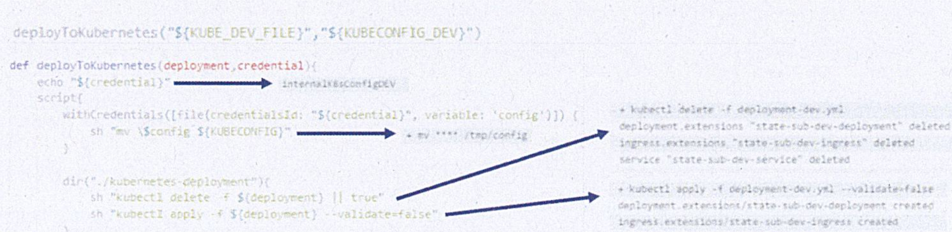
ภาพที่ 3.159 ผลลัพธ์การแก้ไขไฟล์ Deployment



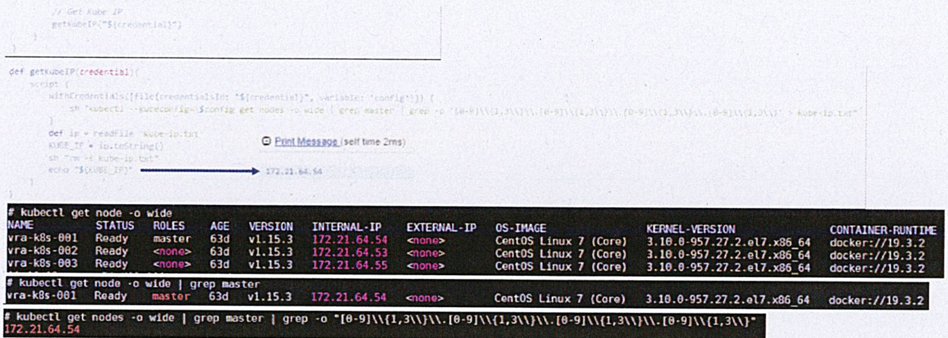
ภาพที่ 3.160 การทำงานของ Kubernetes เบื้องต้น



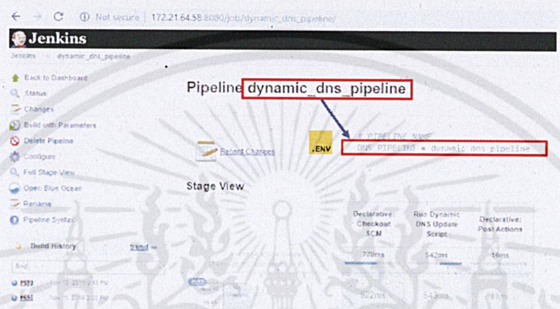
ภาพที่ 3.161 Config ที่จำเป็นสำหรับการ Deploy ไปยัง Kubernetes



ภาพที่ 3.162 ฟังก์ชั่นสำหรับการ Apply ไฟล์ Deployment

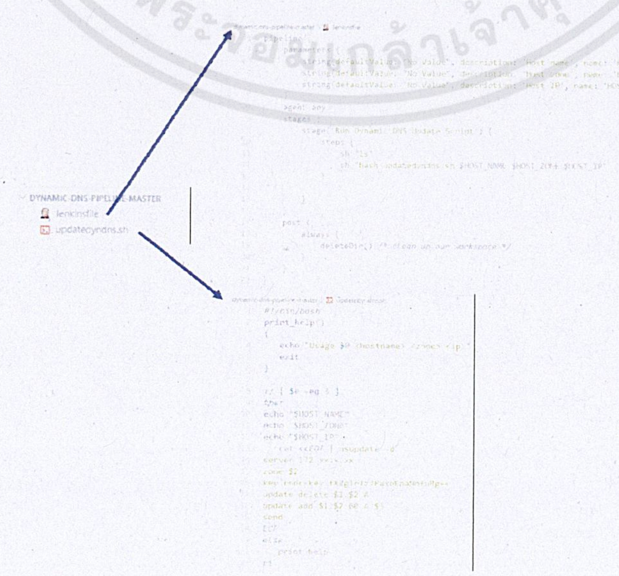


ภาพที่ 3.163 ฟังก์ชั่นการหา IP จากไฟล์ Config

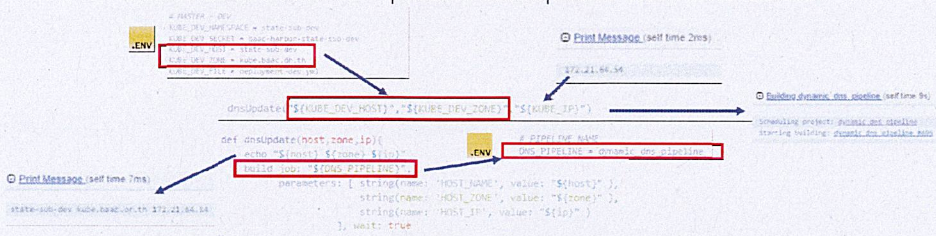


ภาพที่ 3.164 Pipeline สำหรับ Update DNS

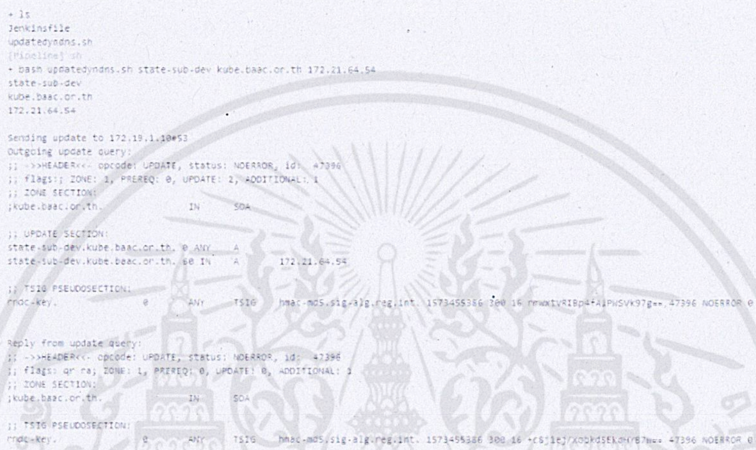
ภาพที่ 3.165 การตั้งค่า Job สำหรับ Pipeline Update DNS



ภาพที่ 3.166 คำสั่งต่างๆที่ใช้สำหรับ Update ไปยัง DNS Server

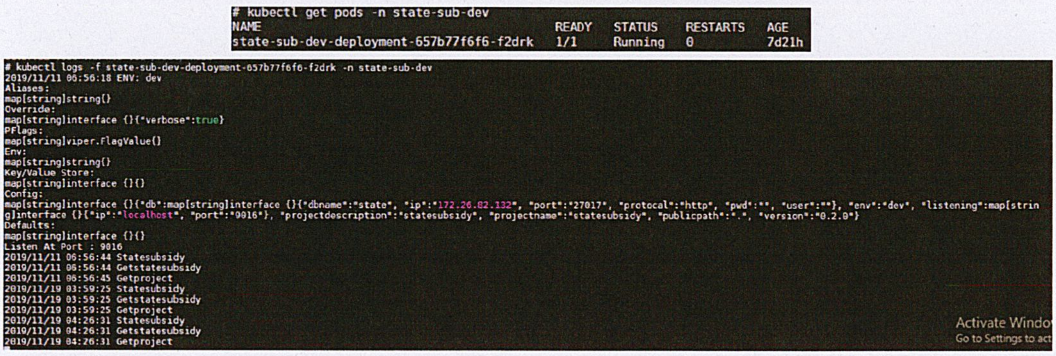


ภาพที่ 3.167 ฟังก์ชั่นใน Pipeline สำหรับการสั่ง Trigger Pipeline DNS



ภาพที่ 3.168 ผลลัพธ์การอัพเดท DNS

ภาพที่ 3.169 ทดสอบการใช้ API ด้วย Postman



ภาพที่ 3.170 การใช้คำสั่งเพื่อตรวจสอบ Log ของ Pod บน Kubernetes

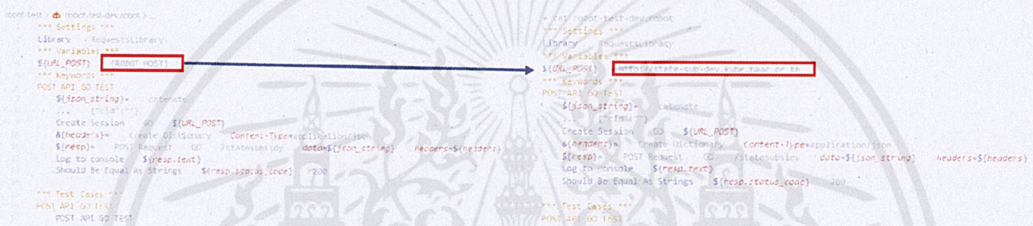
3.5.5.7 Stage Robot Test

```
// Master
initialRobotFile("${ROBOT_DEV_FILE}","${DEV_HOST}")
robotTest("${ROBOT_DEV_FILE}",'DEV')
```

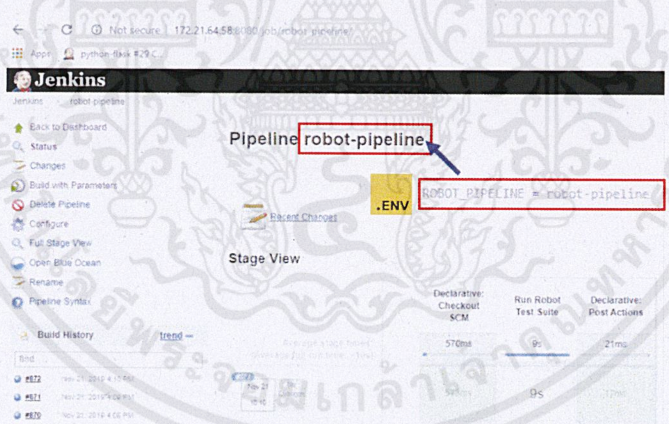
ภาพที่ 3.171 คำสั่งสำหรับการทำงาน Robot

```
        initialRobotFile("${ROBOT_DEV_FILE}","${DEV_HOST}")

def initialRobotFile(fileName,host){
    script{
        dir("./robot-test"){
            def robot = readFile("${fileName}")
            robot = robot.replace("{ROBOT_HOST}", "http://" + "${host}")
            writeFile file:"${fileName}", text:robot
            sh "cat ${fileName}"
        }
    }
}
```

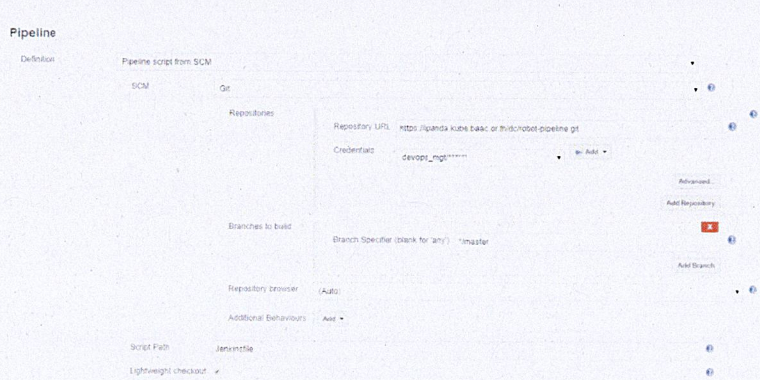
ภาพที่ 3.172 ฟังก์ชั่นสำหรับการแก้ไขไฟล์ Robot



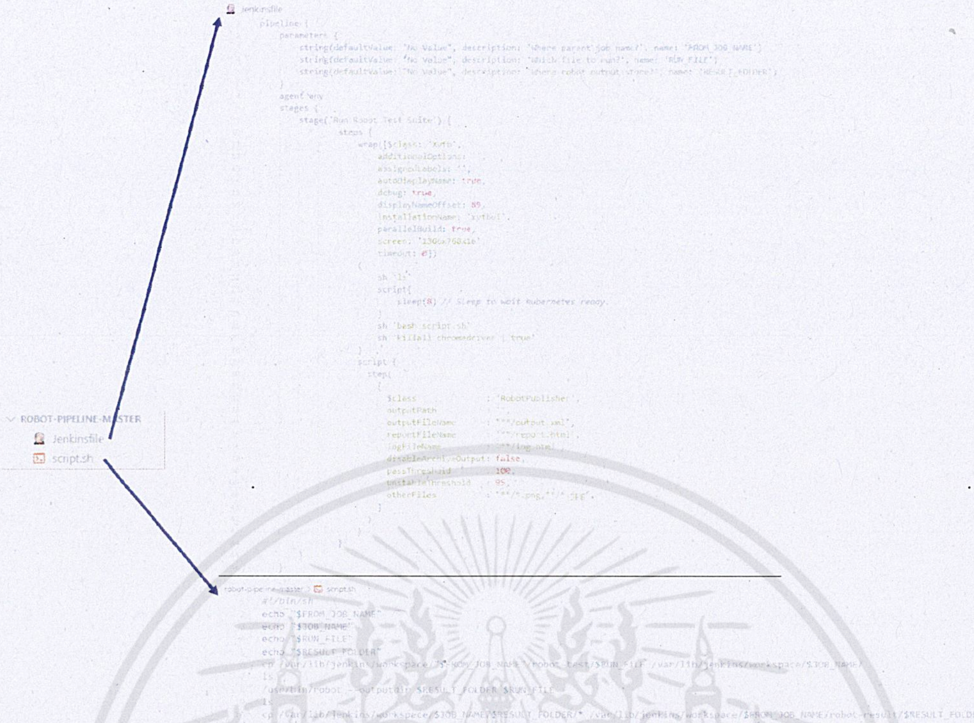
ภาพที่ 3.173 ผลลัพธ์การแก้ไขไฟล์ Robot



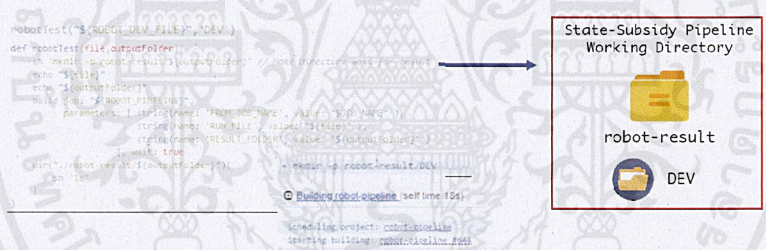
ภาพที่ 3.174 Pipeline Robot



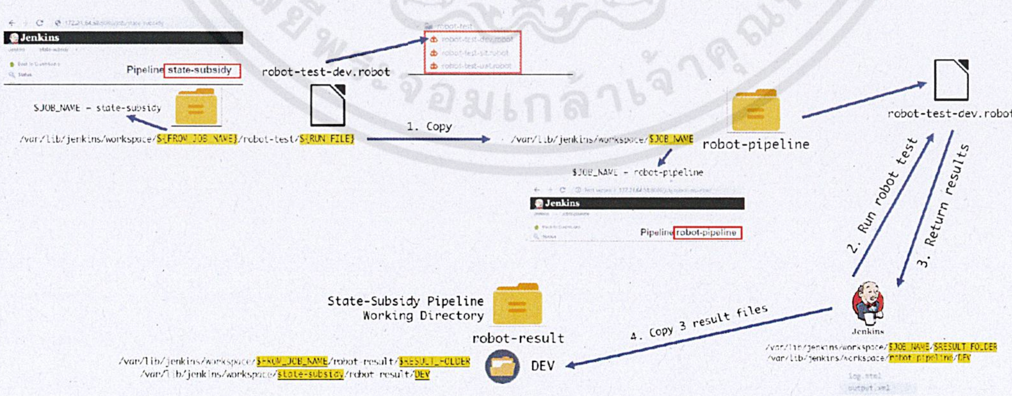
ภาพที่ 3.175 การตั้งค่า Job ของ Robot Pipeline



ภาพที่ 3.176 Pipeline การทำงานของ Robot



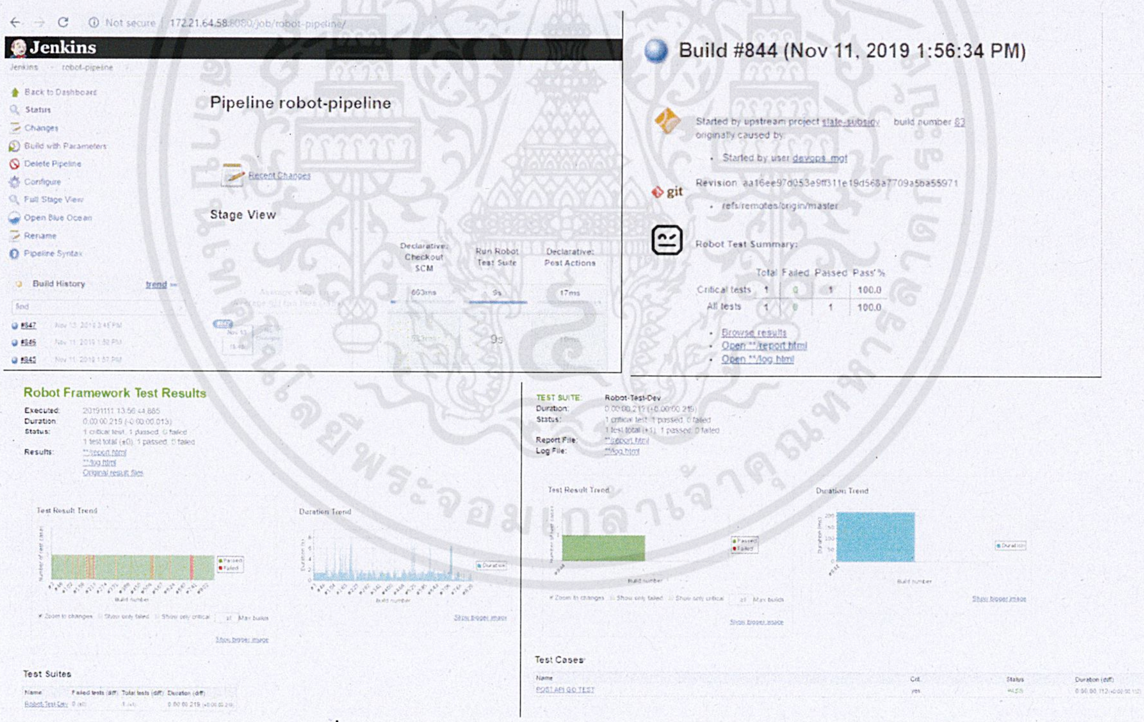
ภาพที่ 3.177 ฟังก์ชั่นสำหรับการ Trigger Pipeline ของ Robot



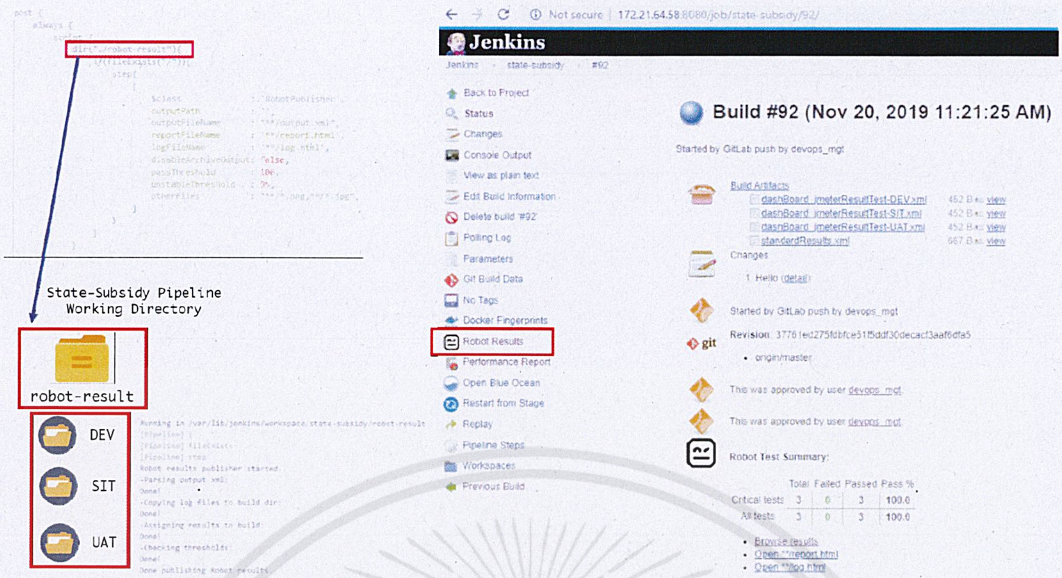
ภาพที่ 3.178 แสดงการทำงานของ Robot Pipeline

```
Xvfb starting$ /usr/bin/Xvfb -displayfd 2 -screen 0 1366x768x16 -fbdir /var/lib/jenkins/xvfb-844-..fbdir737921185424236929?
0
[Pipeline] {
[Pipeline] sh
+ ls
Jenkinsfile
script.sh
[Pipeline] script
[Pipeline] {
[Pipeline] sleep
Sleeping for 8 sec
[Pipeline] }
[Pipeline] // script
[Pipeline] sh
+ bash script.sh
state-subsidy
robot-pipeline
robot-test-dev.robot
DEV
Jenkinsfile
robot-test-dev.robot
script.sh
==============================================================================
Robot-Test-Dev
==============================================================================
POST API GO TEST                                        {"response_data":[{"project_name":"","account_no":"","transfer_amt":"","post_date":"","brn_name":"","status":"07"}]}

| PASS |
------------------------------------------------------------------------------
Robot-Test-Dev                                          | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /var/lib/jenkins/workspace/robot-pipeline/DEV/output.xml
Log:     /var/lib/jenkins/workspace/robot-pipeline/DEV/log.html
Report:  /var/lib/jenkins/workspace/robot-pipeline/DEV/report.html
DEV
Jenkinsfile
robot-test-dev.robot
script.sh
[Pipeline] sh
+ killall chromedriver
+ true
chromedriver: no process found
[Pipeline] }
Xvfb stopping
```

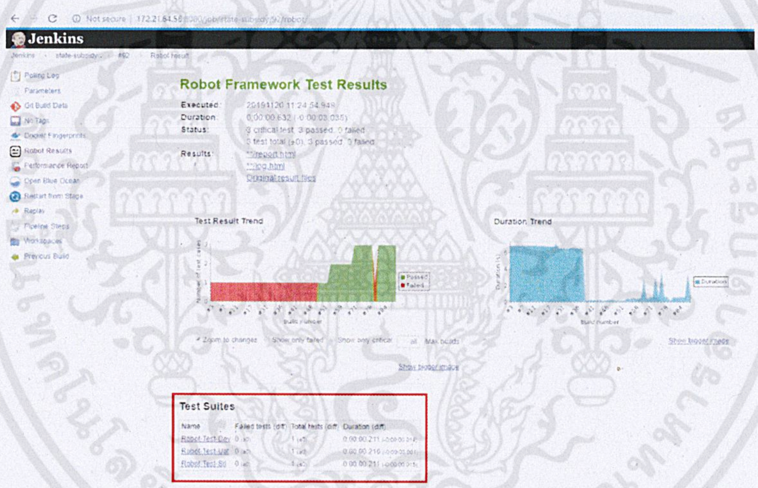
ภาพที่ 3.179 ผลลัพธ์การทำงานของ Robot



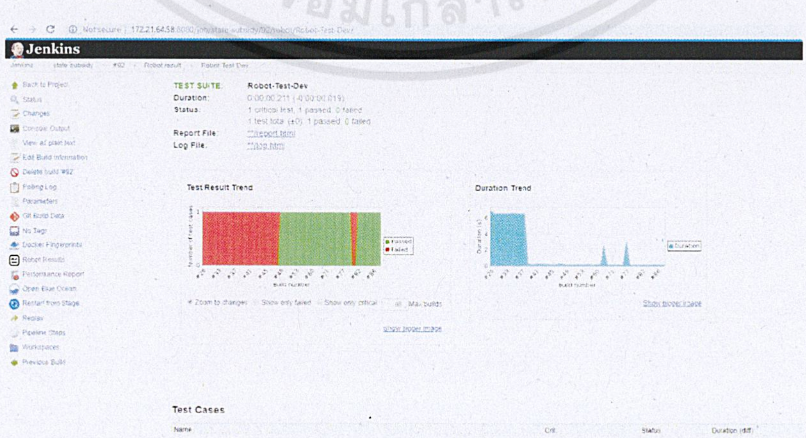
ภาพที่ 3.180 ผลลัพธ์การทำงานของ Pipeline Robot



ภาพที่ 3.181 การอ่านไฟล์ Report ของ Robot ไปแสดงผลบน Jenkin

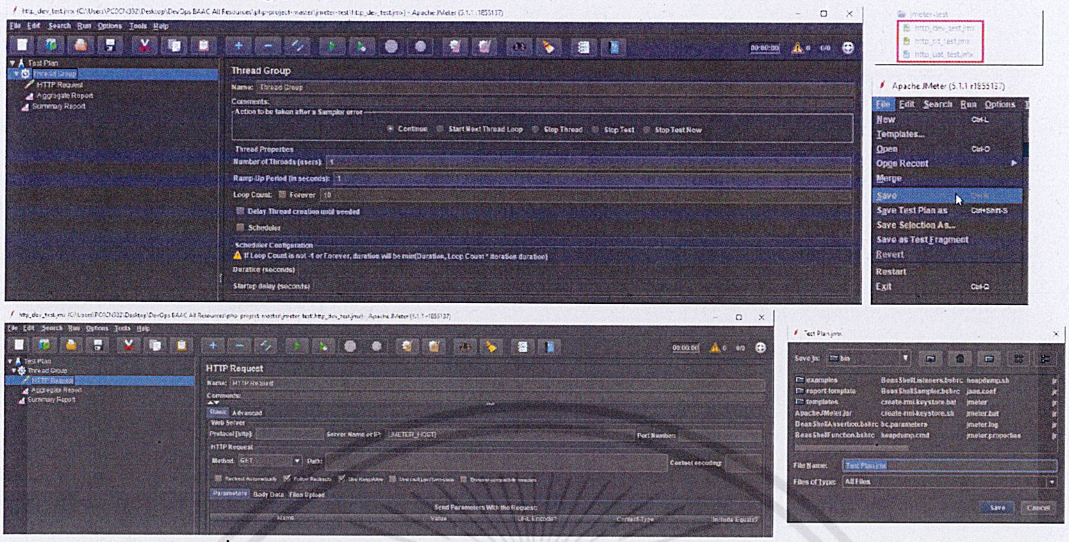


ภาพที่ 3.182 ผลลัพธ์รายงาน Robot บน Jenkins แต่ละ Environment



ภาพที่ 3.183 ผลลัพธ์ของ Test Case Robot บน Jenkins

3.5.5.8 Stage JMeter Test

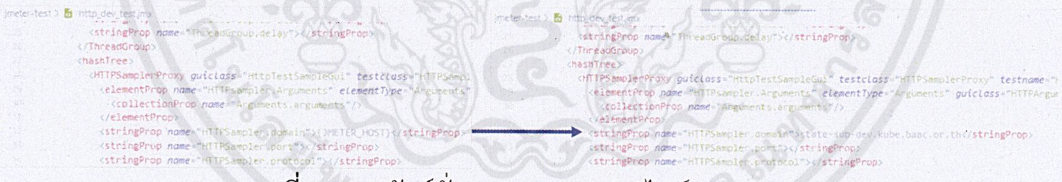


ภาพที่ 3.184 การจัดเตรียมไฟล์ JMeter เพื่อทำ Performance Test

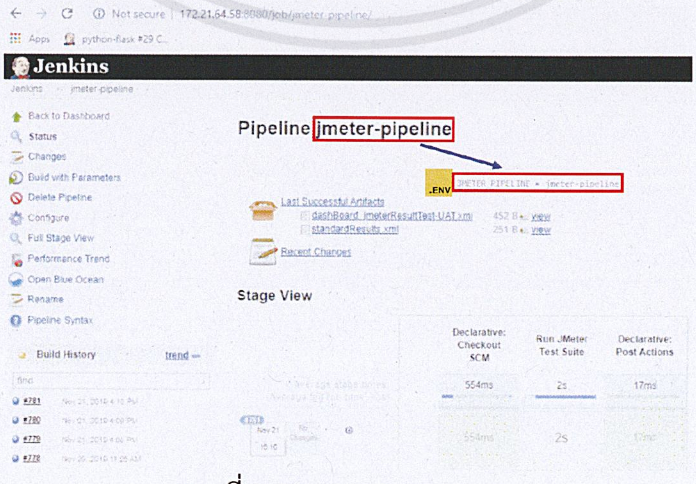
```
// Master
initialJMeterFile("${JMETER_DEV_FILE}","${DEV_HOST}")
jmeterTest("${JMETER_DEV_FILE}",'DEV') // master branch
```

ภาพที่ 3.185 คำสั่งสำหรับการ Test Performance ด้วย JMeter

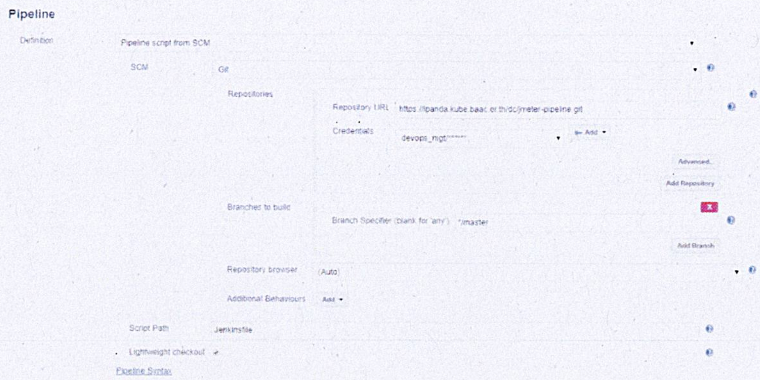
```
def initialJMeterFile(fileName,host){
    script{
        dir("./jmeter-test"){
            def jmeter = readFile("${fileName}")
            jmeter = jmeter.replace("{JMETER_HOST}", "${host}")
            writeFile file:"${fileName}", text:jmeter
        }
    }
}
```



ภาพที่ 3.186 ฟังก์ชั่นและผลการแปลงไฟล์ของ JMeter

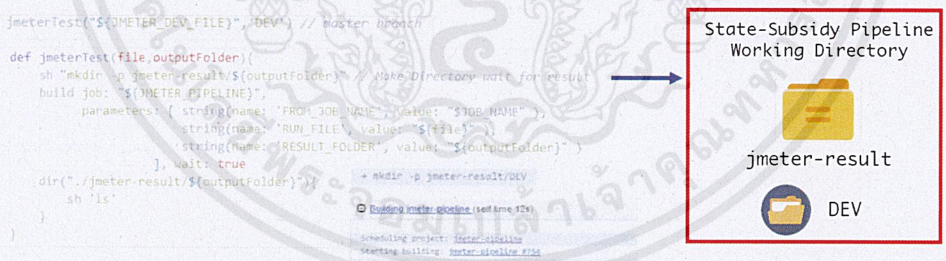


ภาพที่ 3.187 Pipeline JMeter

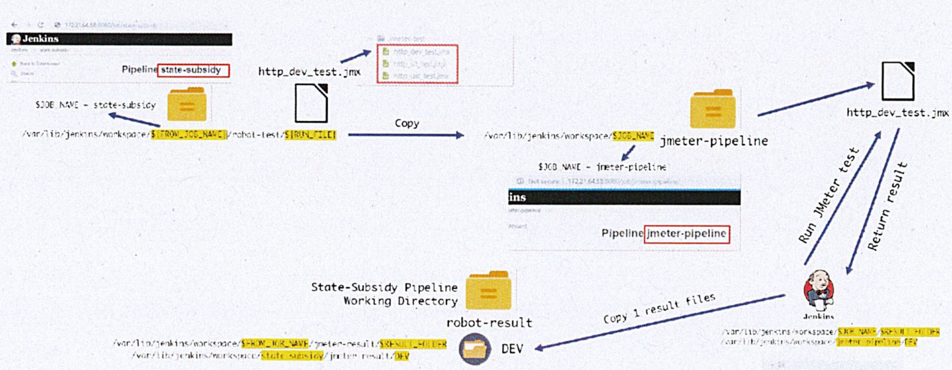


ภาพที่ 3.188 การตั้งค่า Project ของ Pipeline JMeter

ภาพที่ 3.189 Pipeline ของ JMeter



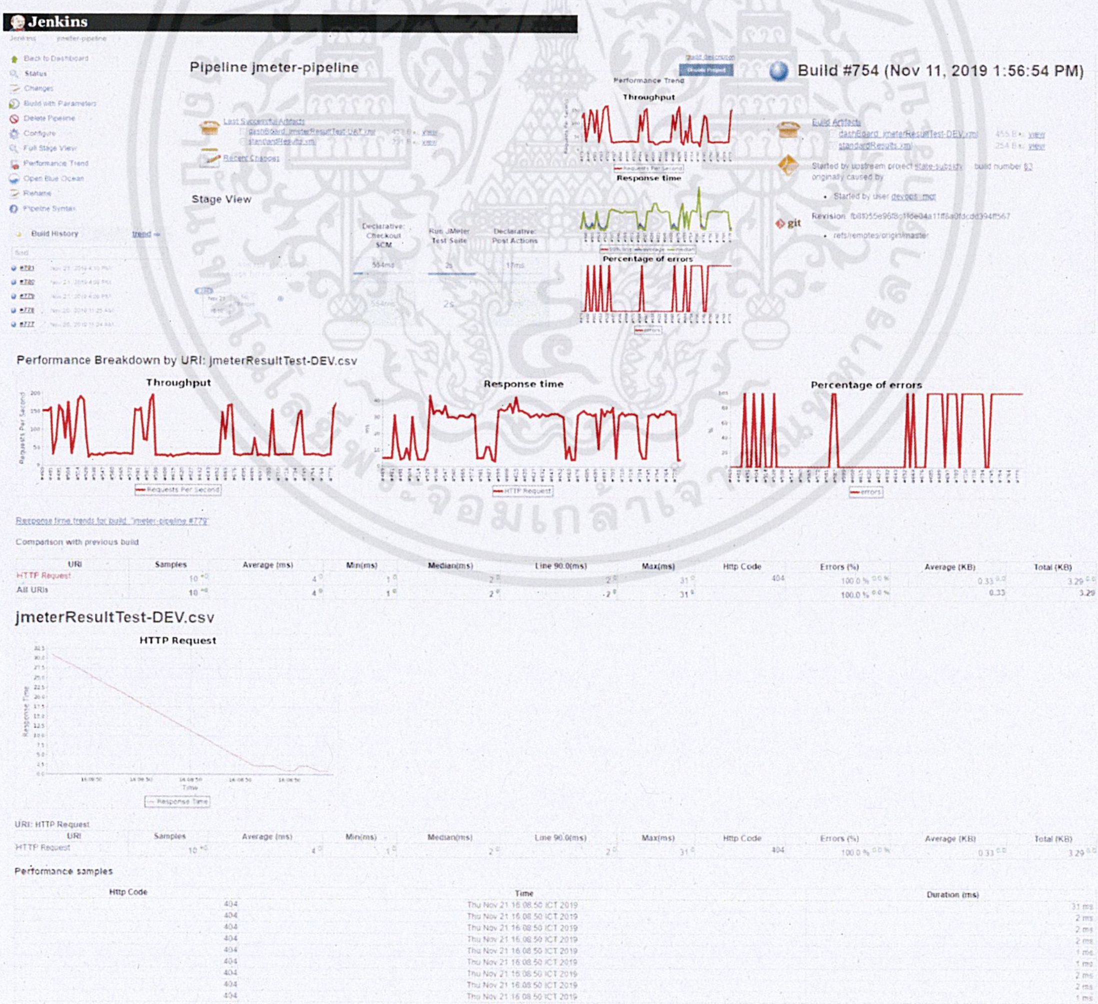
ภาพที่ 3.190 ฟังก์ชั่นการ Trigger Pipeline JMeter



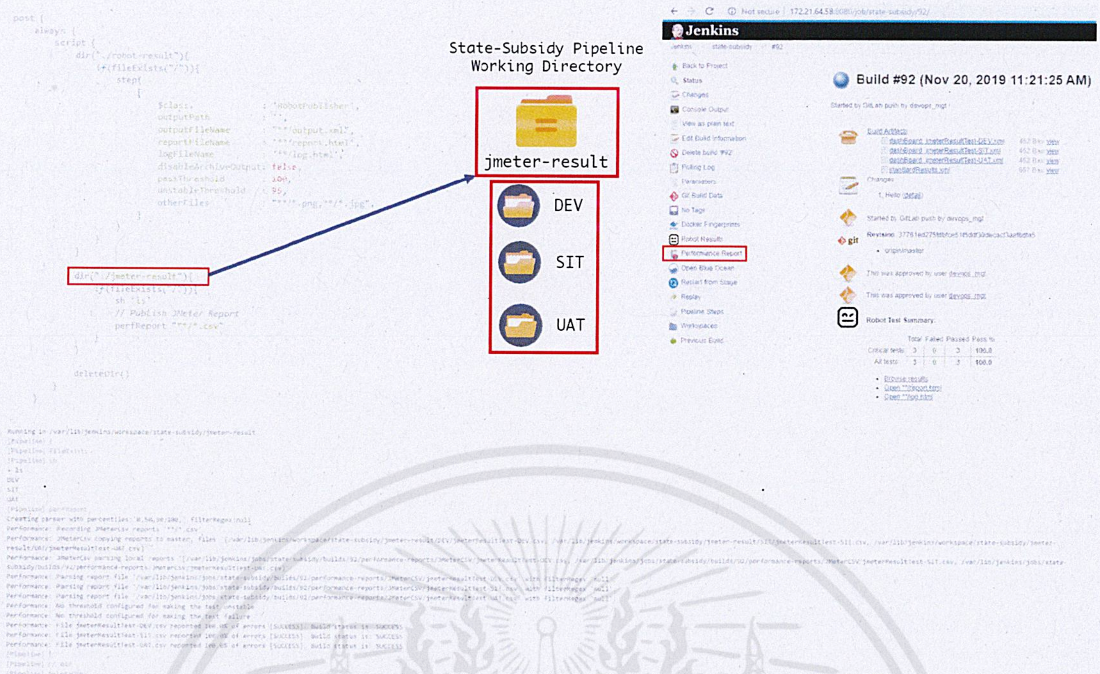
ภาพที่ 3.191 ลำดับการทำงานของ Pipeline JMeter

```
+ ls
Jenkinsfile
script.sh
[Pipeline] sh
+ bash script.sh
state-subsidy
jmeter-pipeline
http_dev_test.jmx
DEV
                                                                5.1.1 r1855137

Copyright (c) 1999-2019 The Apache Software Foundation

http_dev_test.jmx
Jenkinsfile
jmeter.log
script.sh
Creating summariser <summary>
Created the tree successfully using http_dev_test.jmx
Starting the test @ Mon Nov 11 13:56:57 ICT 2019 (1573455417722)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary =      1 in 00:00:00 =    7.8/s Avg:    30 Min:    30 Max:    30 Err:     1 (100.00%)
Tidying up ...    @ Mon Nov 11 13:56:58 ICT 2019 (1573455418109)
... end of run
C:\Kone\projects\gowebForBank\recording.xml
http_dev_test.jmx
Jenkinsfile
jmeter.log
jmeterResultTest-DEV.csv
script.sh
[Pipeline] step
Creating parser with percentiles:'0,50,90,100,' filterRegex:null
Performance: Recording JMeterCsv reports 'jmeterResultTest-DEV.csv'
Performance: JMeterCsv copying reports to master, files '[/var/lib/jenkins/workspace/jmeter-pipeline/jmeterResultTest-DEV.csv]'
Performance: JMeterCsv parsing local reports '[/var/lib/jenkins/jobs/jmeter-pipeline/builds/754/performance-reports/JMeterCSV/jmeterResultTest-DEV.csv]'
Performance: Parsing report file '/var/lib/jenkins/jobs/jmeter-pipeline/builds/754/performance-reports/JMeterCSV/jmeterResultTest-DEV.csv' with filterRegex 'null'.
Performance: No threshold configured for making the test unstable
Performance: No threshold configured for making the test failure
Performance: File jmeterResultTest-DEV.csv reported 100.0% of errors [SUCCESS]. Build status is: SUCCESS
```

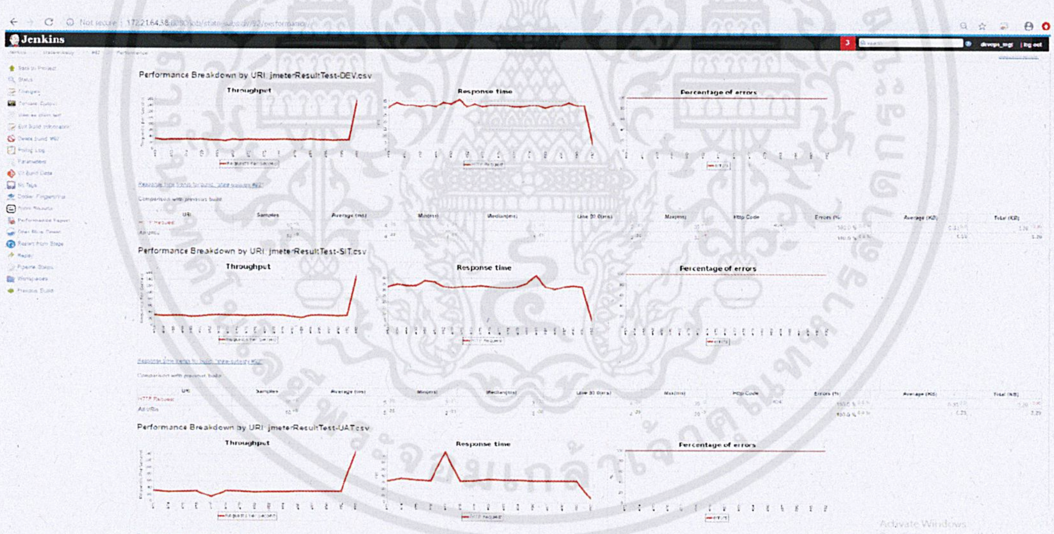
ภาพที่ 3.192 ผลลัพธ์การทำงานของ JMeter



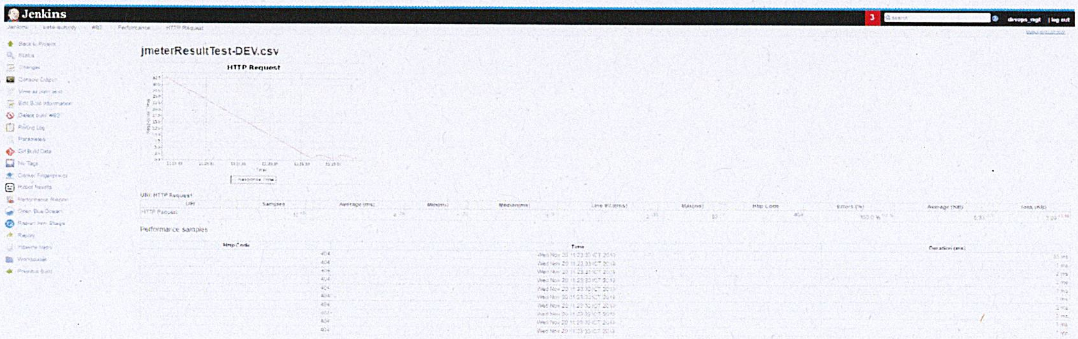
ภาพที่ 3.193 รายงานการทดสอบประสิทธิภาพด้วย JMeter



ภาพที่ 3.194 การอ่านไฟล์ Report ของ JMeter มาแสดงผลบน Jenkins

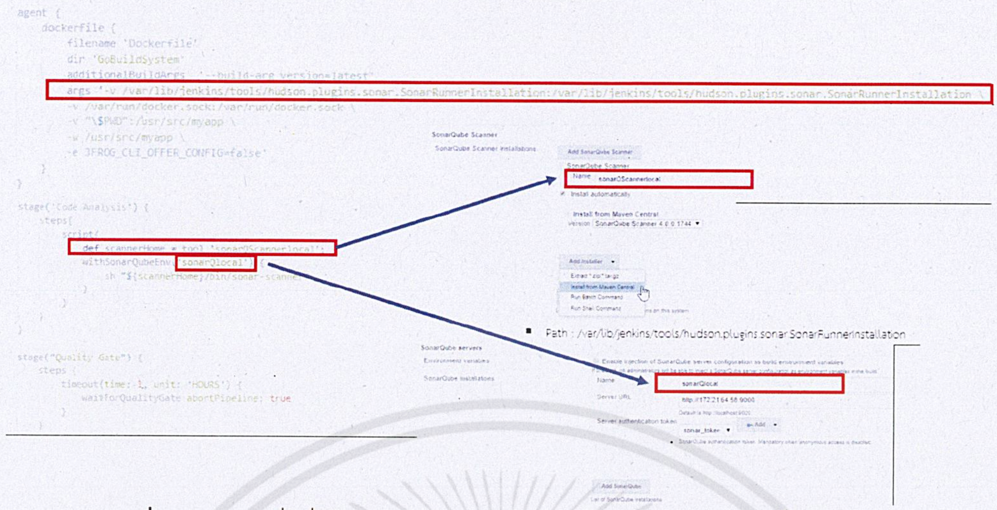


ภาพที่ 3.195 ผลการทดสอบประสิทธิภาพแต่ละ Environment

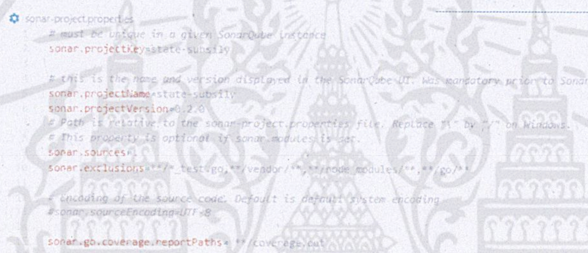


ภาพที่ 3.196 ผลรายงานของ JMeter

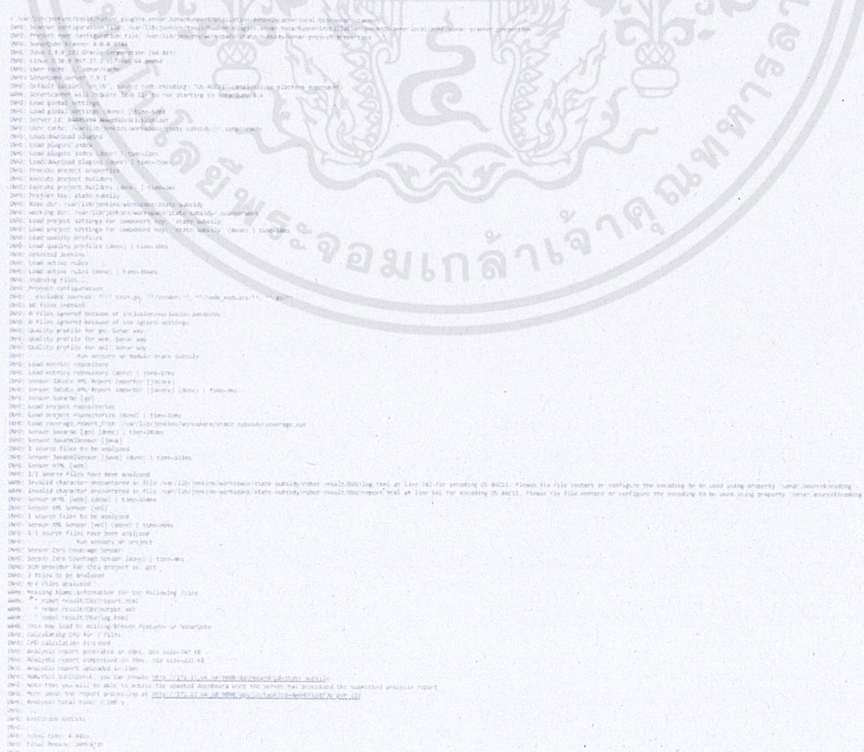
3.5.5.9 Stage Code Analysis และ Quality Gate



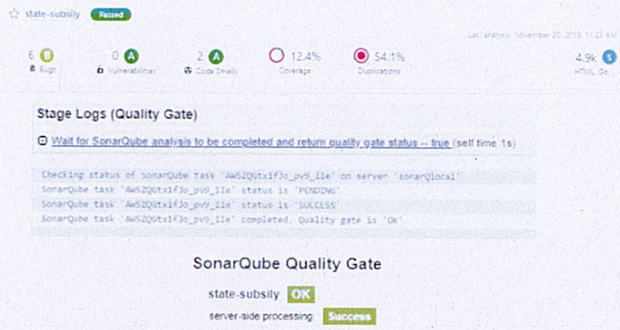
ภาพที่ 3.197 คำสั่งที่ใช้สำหรับการ Mount Sonar Client บนเครื่อง Jenkins



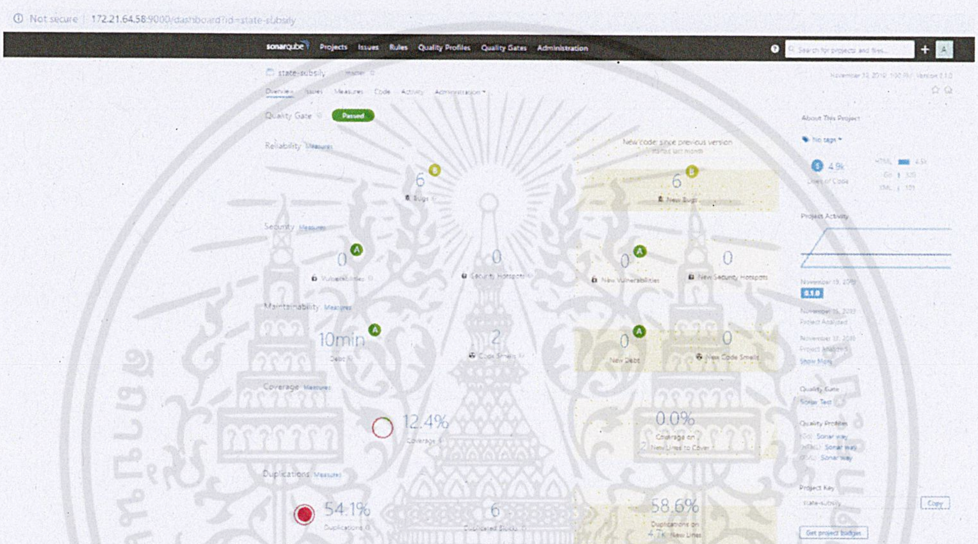
ภาพที่ 3.198 ไฟล์ใช้สำหรับตั้งค่าการแสกนด้วย SonarQube



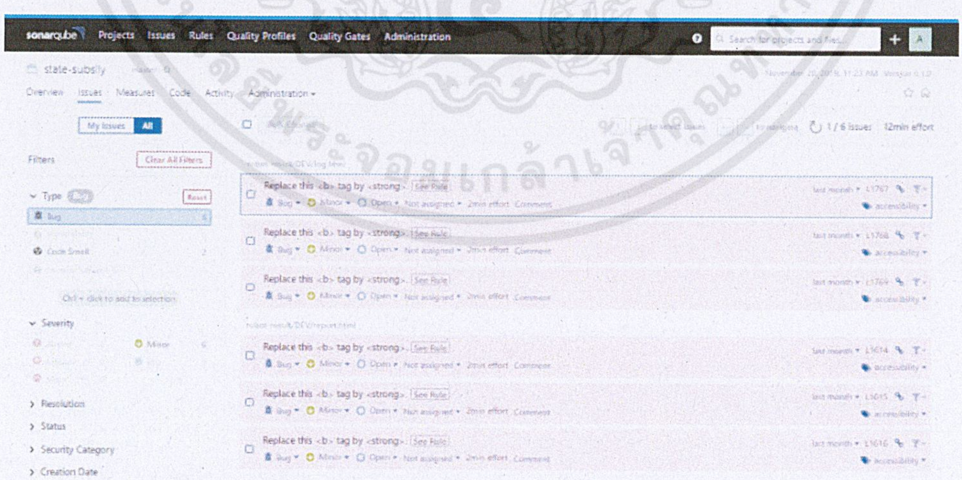
ภาพที่ 3.199 ผลลัพธ์การแสกน Source Code ด้วย Jenkins



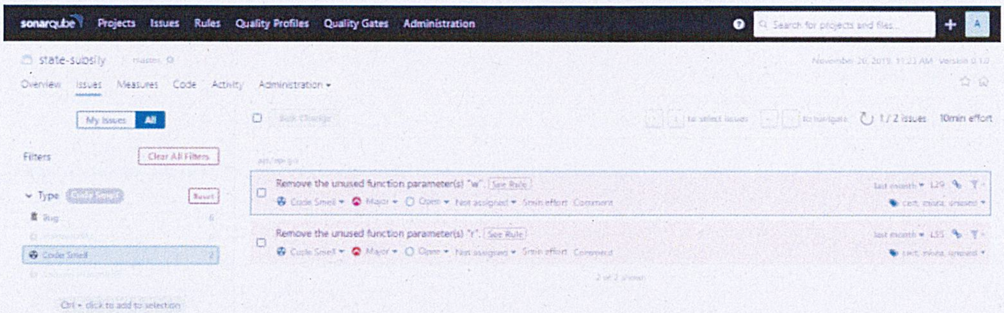
ภาพที่ 3.200 ผลการทำงานของ Quality Gate



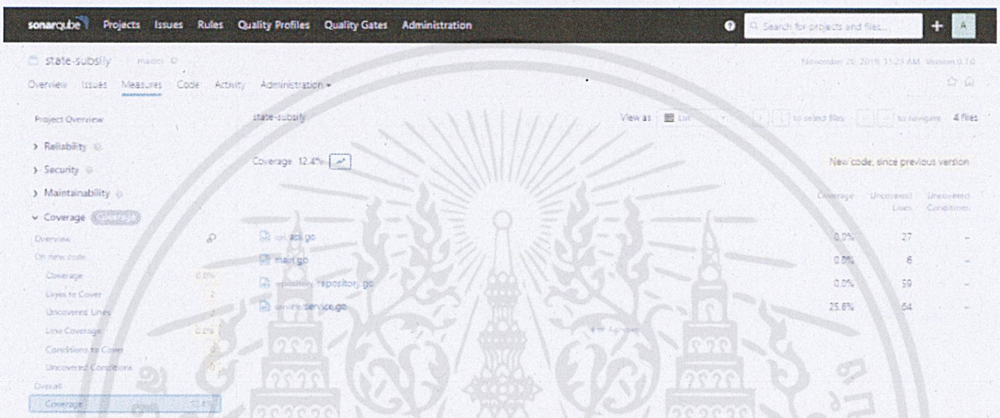
ภาพที่ 3.201 ผลการแสกนของโปรเจคบน SonarQube



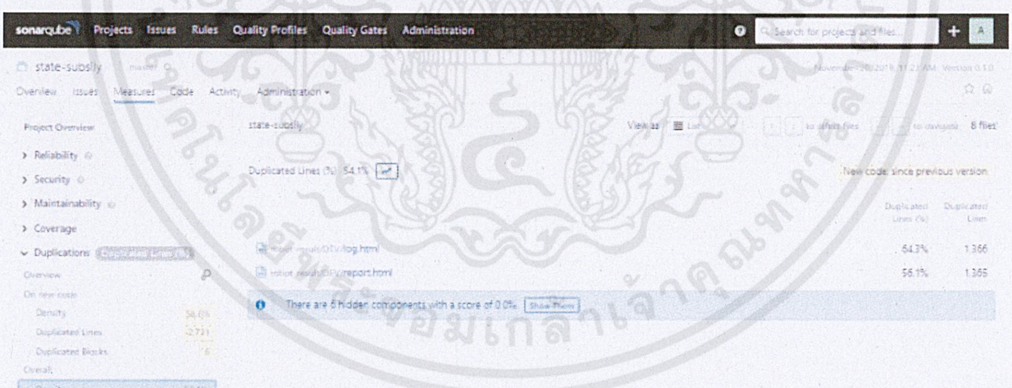
ภาพที่ 3.202 ผลการตรวจสอบ Bug ของ SonarQube



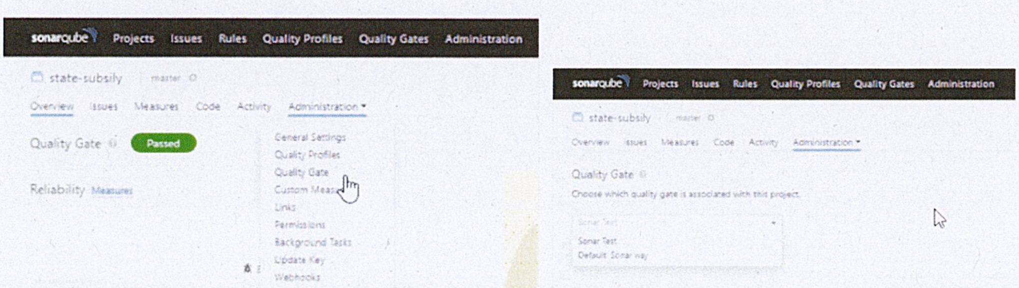
ภาพที่ 3.203 ผลของ Code Smell บน SonarQube



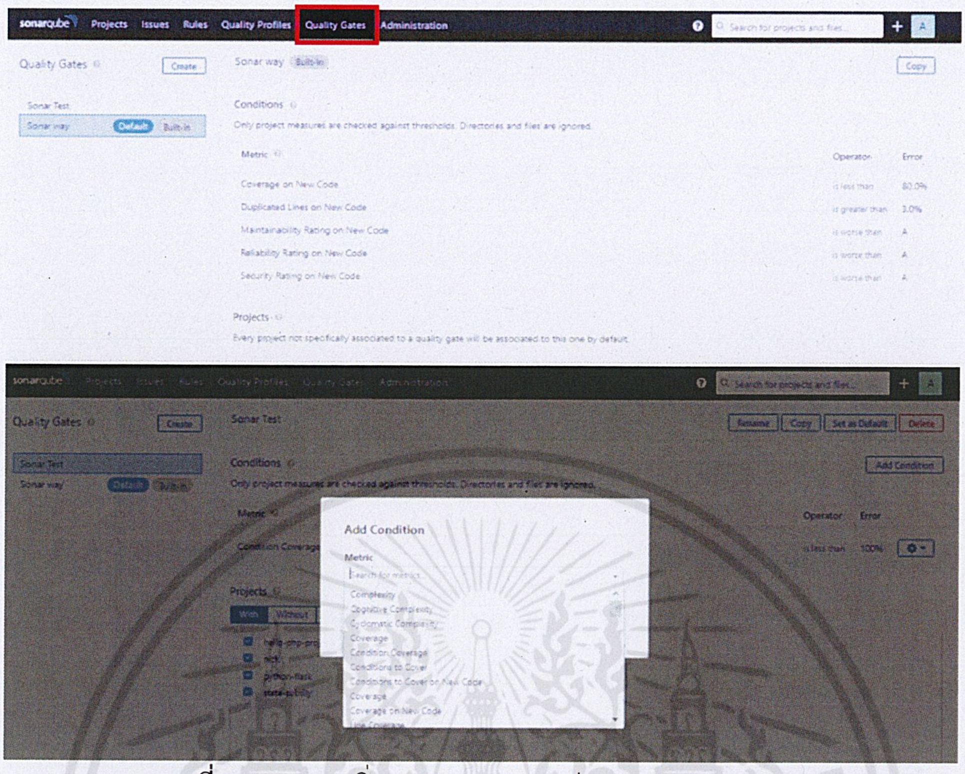
ภาพที่ 3.204 ผลการตรวจสอบ Coverage การทำ Test ของ SonarQube



ภาพที่ 3.205 การตรวจสอบความซ้ำซ้อนของ Code บน SonarQube

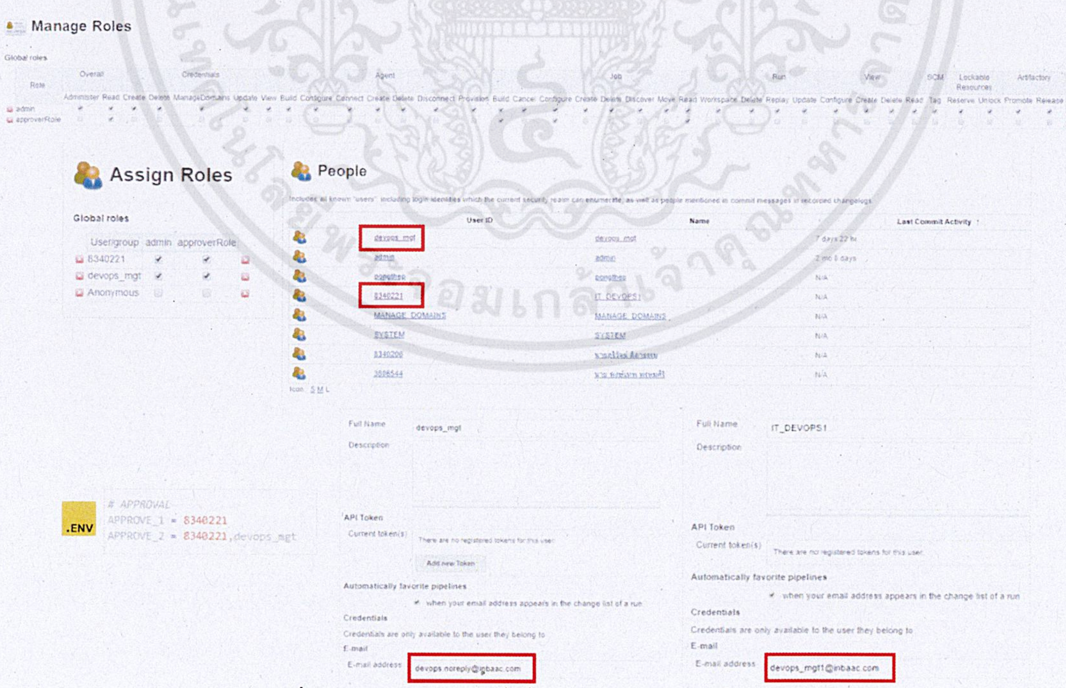


ภาพที่ 3.206 การเลือกโปรไฟล์ Quality Gate

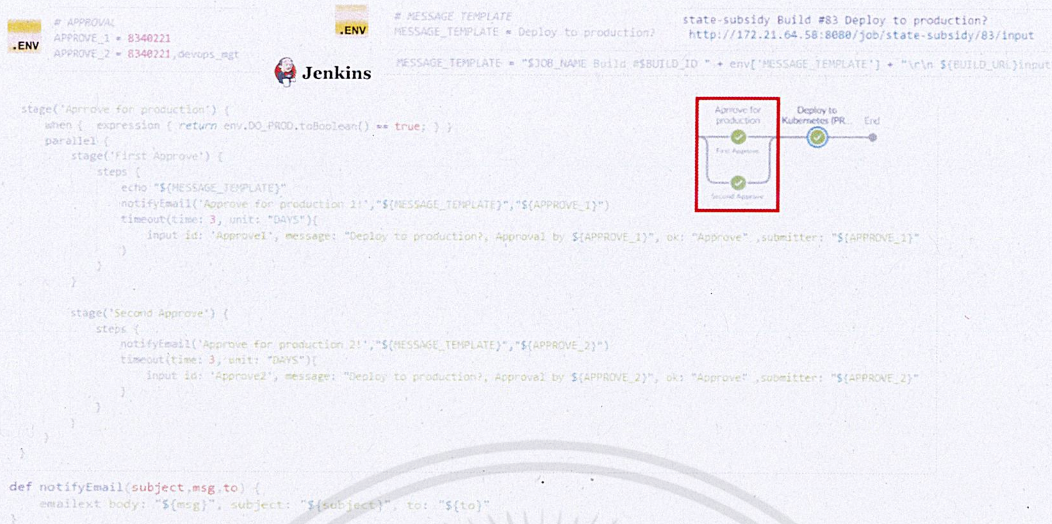


ภาพที่ 3.207 การเพิ่ม Condition การผ่านของ Quality Gate

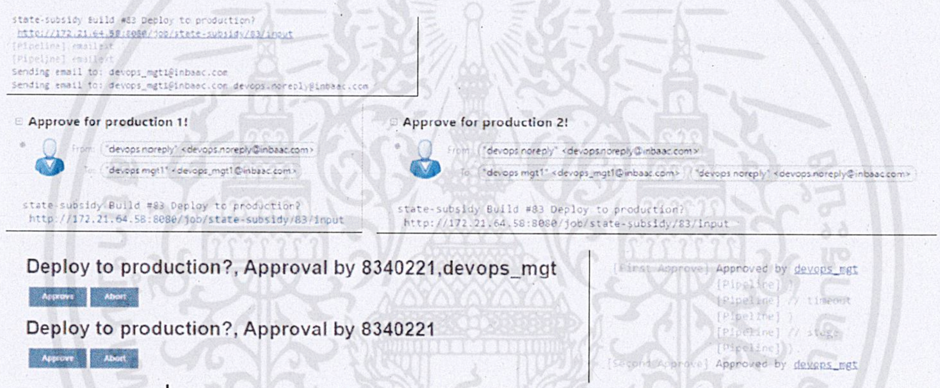
3.5.5.10 Stage Approve ก่อนขึ้น Production



ภาพที่ 3.208 ใช้ User ที่เชื่อมต่อกับ LDAP ที่มี E-mail



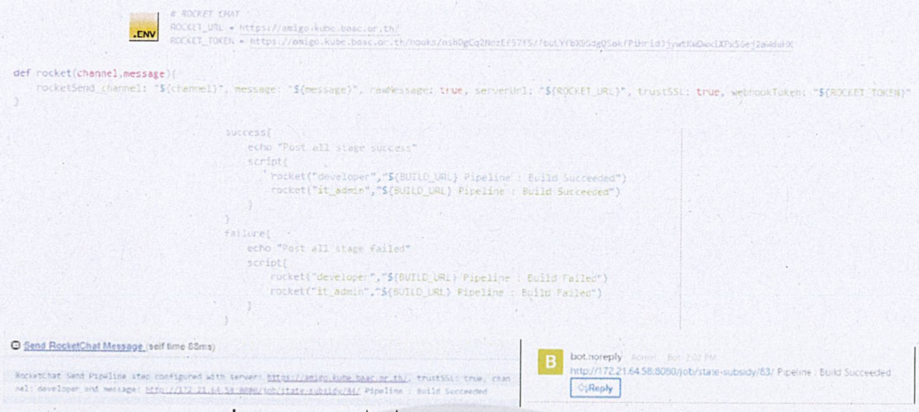
ภาพที่ 3.209 การส่ง URL สำหรับ Approve แจ้งเตือนไปยัง E-mail ของ User



ภาพที่ 3.210 ผลการส่ง E-mail และหน้าจอสำหรับทำการ Approve

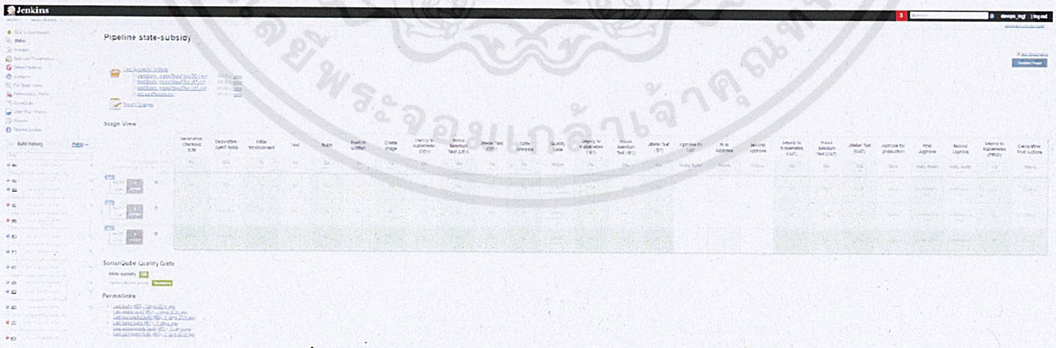
ภาพที่ 3.211 การหา User จากชื่อ Role

3.5.5.11 Rocket Chat



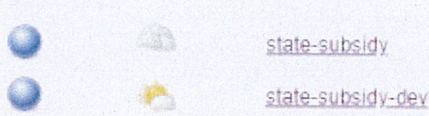
ภาพที่ 3.212 คำสั่งที่ใช้ส่งข้อความไปยัง Rocket Chat

ภาพที่ 3.213 ข้อความที่ถูกส่งมายัง Rocket Chat

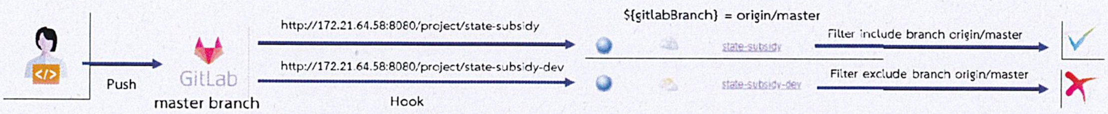


ภาพที่ 3.214 ผลลัพธ์แสดงการทำงานของ Pipeline

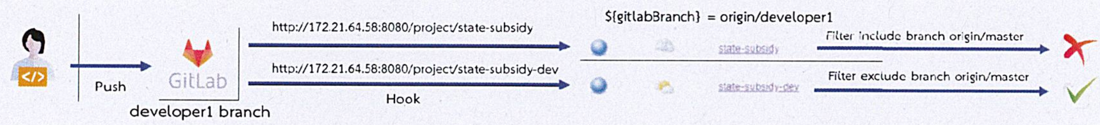
3.5.5.12 การ Build Job Master และ Non-Master แบบอัตโนมัติ



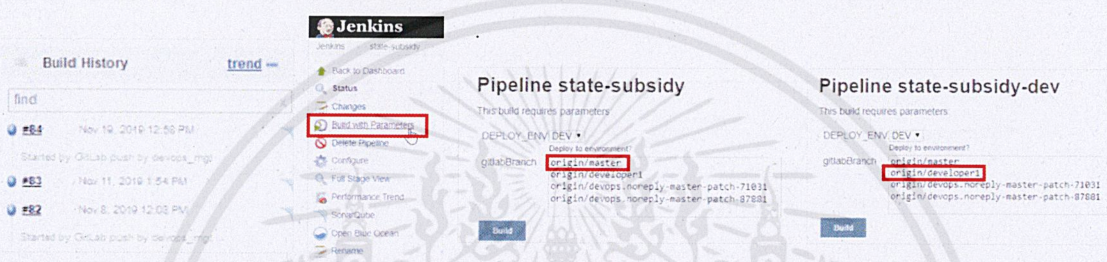
ภาพที่ 3.215 โปรเจคที่ใช้สำหรับการทำ Multi Branch



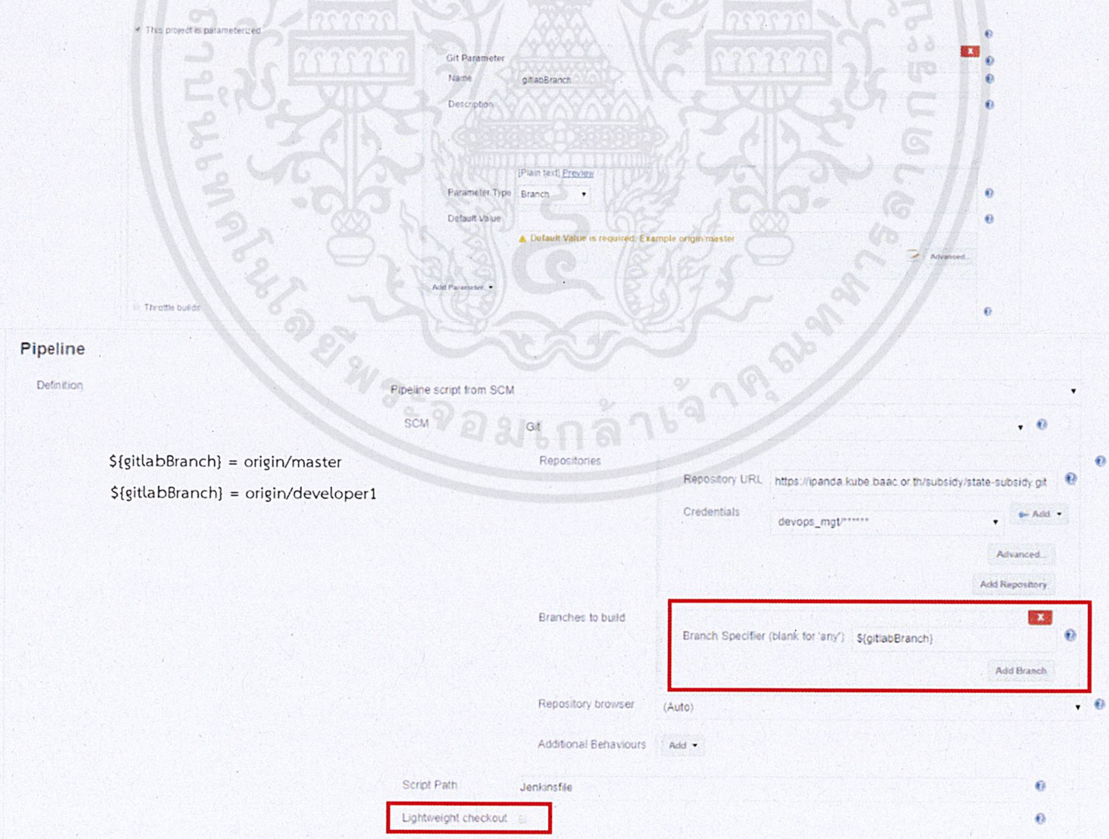
ภาพที่ 3.216 แผนผังการ Hook ของ Branch Master



ภาพที่ 3.217 แผนผังการ Hook ของ Branch ที่ไม่ใช่ Master

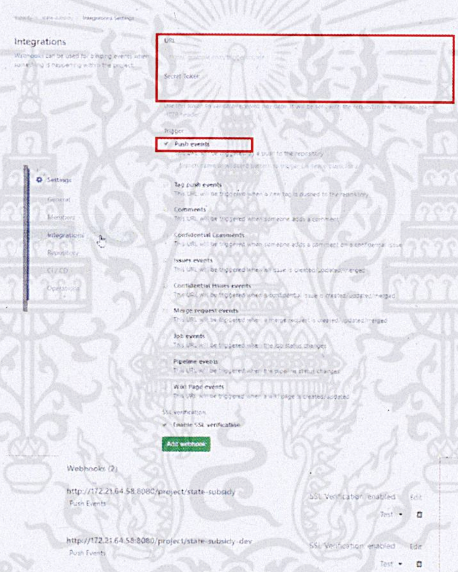


ภาพที่ 3.218 การ Build ของ Job Master และ Non-Master



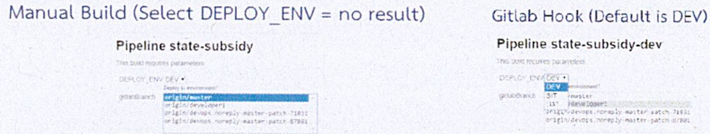
ภาพที่ 3.219 การตั้งค่า Job Master และ Non-Master

ภาพที่ 3.220 การตั้งค่าเกี่ยวกับการ Hook ของ Job Master และ Non-Master

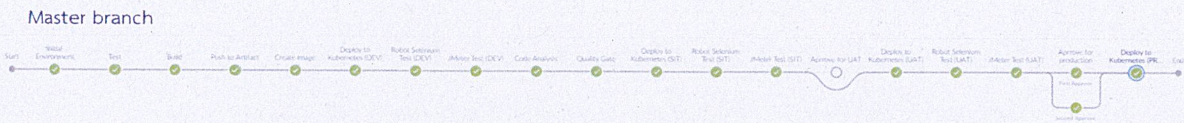


ภาพที่ 3.221 การเพิ่ม Hook URL ของ Jenkins บน GitLab

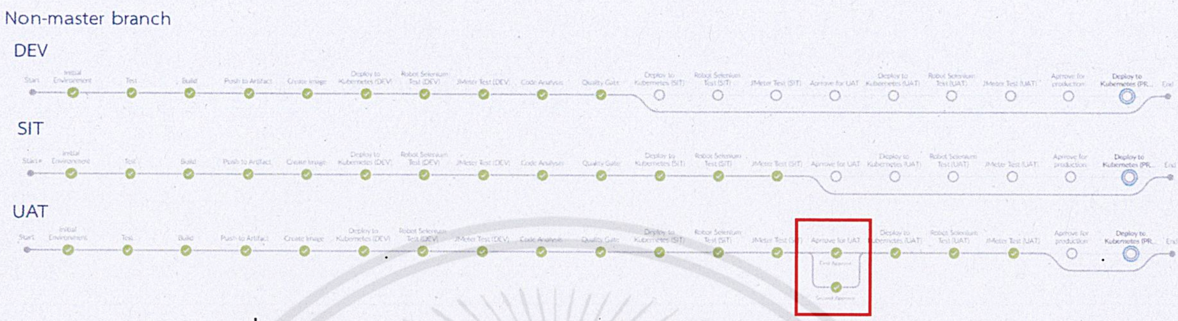
ภาพที่ 3.222 Branch ของ Project Go บน Repository ของ GitLab



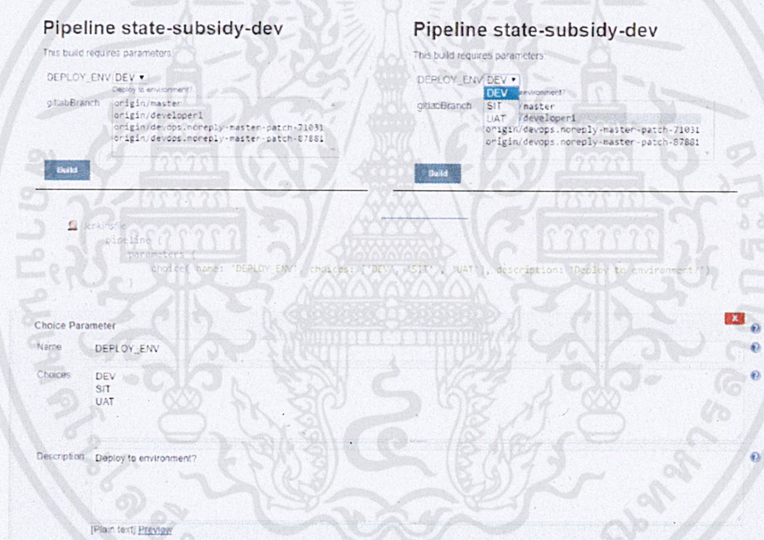
ภาพที่ 3.223 การสั่ง Build จาก Jenkins แบบ Manual



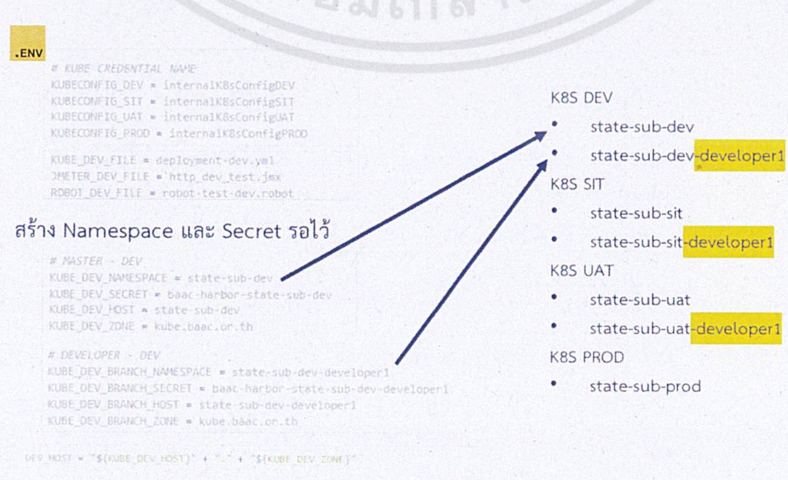
ภาพที่ 3.224 ภาพรวม Stage ของการทำงาน Branch Master



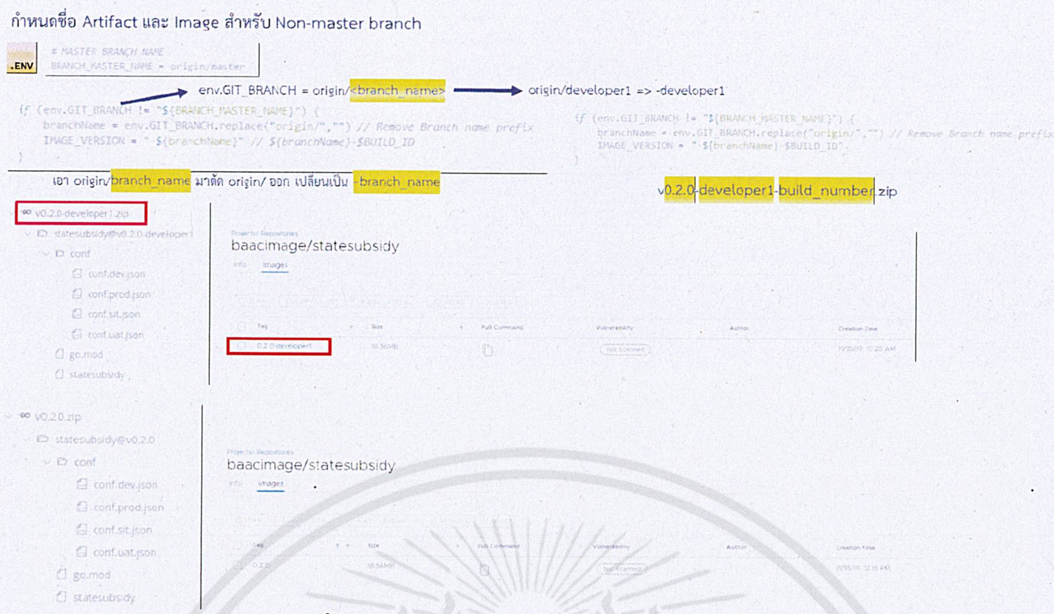
ภาพที่ 3.225 ภาพรวม Stage ของการทำงาน Branch Non-Master



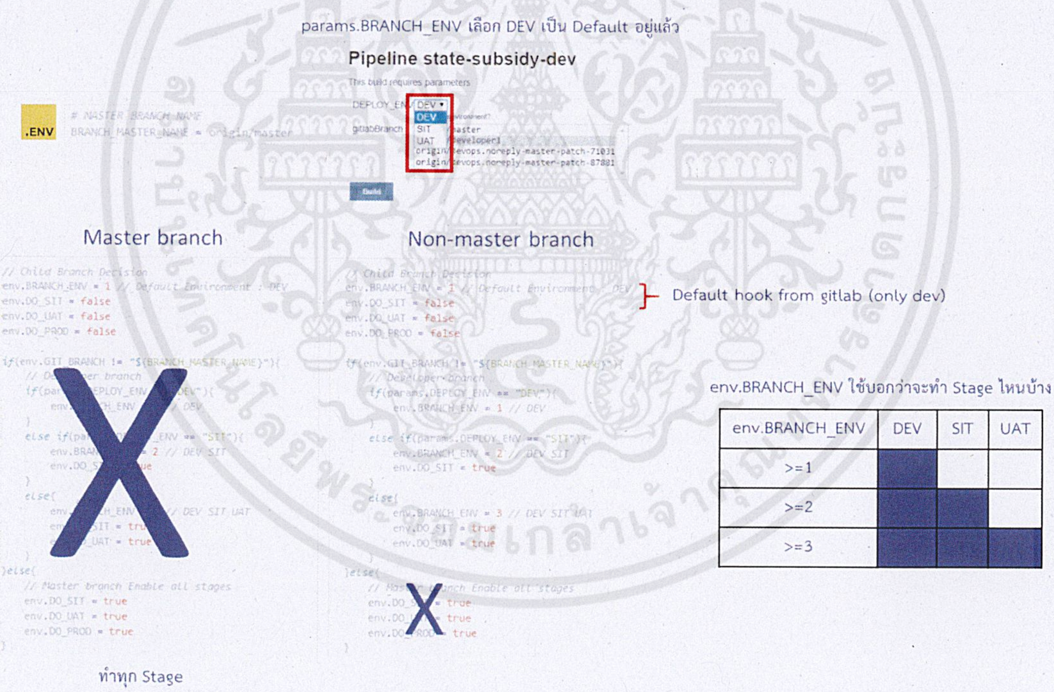
ภาพที่ 3.226 การเลือก Environment สำหรับ Branch Non-Master



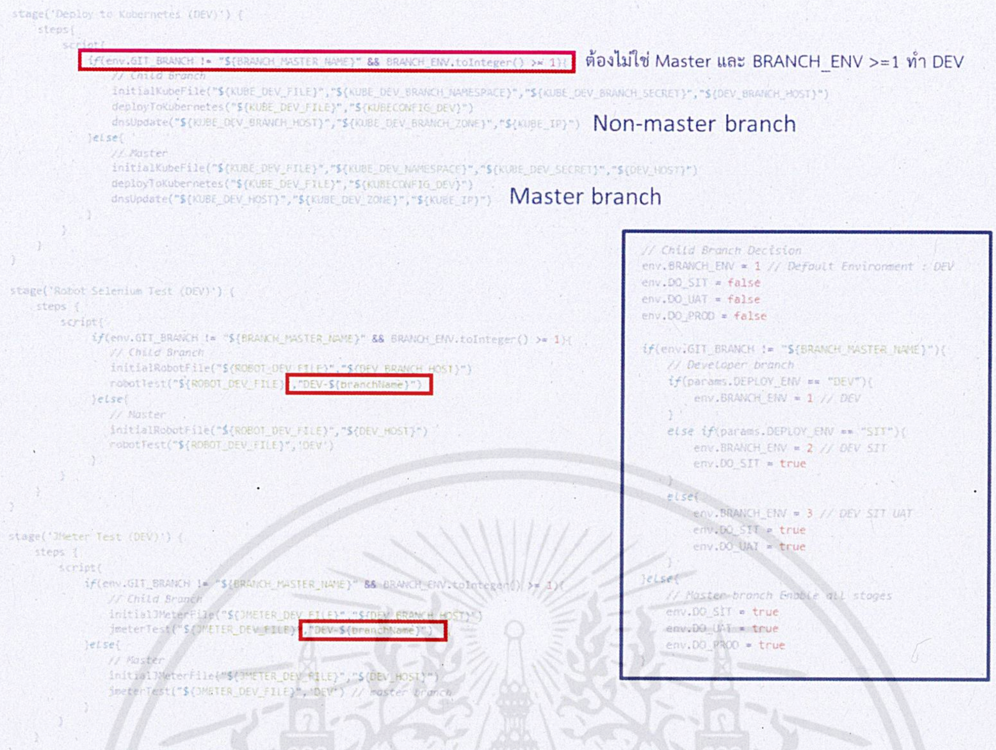
ภาพที่ 3.227 การสร้าง Namespace สำหรับ Branch Non-Master



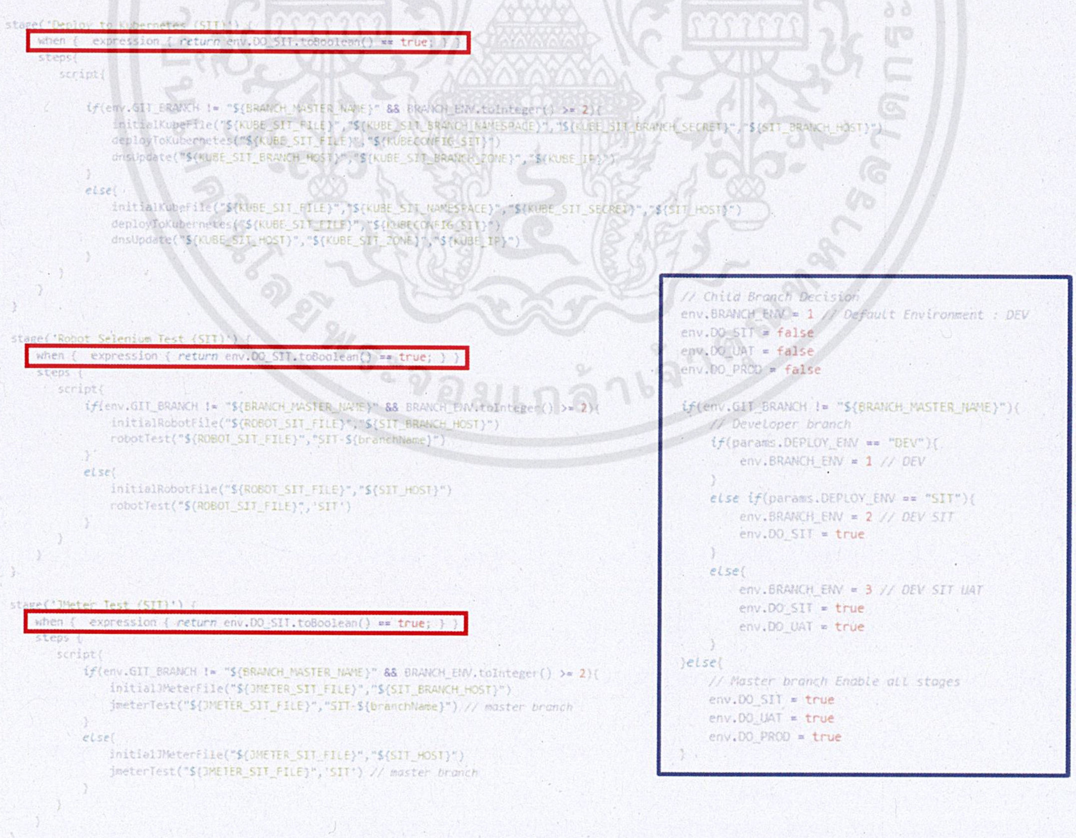
ภาพที่ 3.228 การตั้งชื่อ Artifact และ Image สำหรับ Branch Non-Master



ภาพที่ 3.229 คำสั่งตัดสินใจเลือกทำงาน Deploy จนถึง Environment ต่างๆ

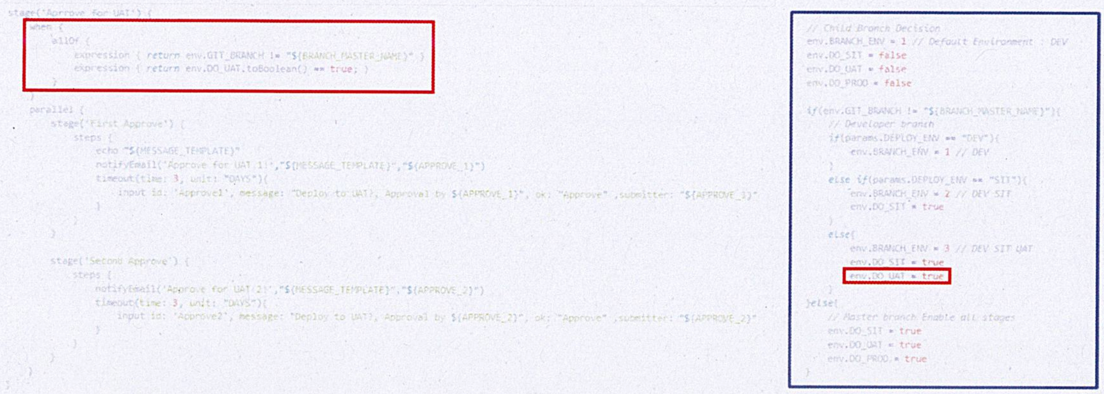


ภาพที่ 3.230 การตรวจสอบการทำงาน Environment DEV

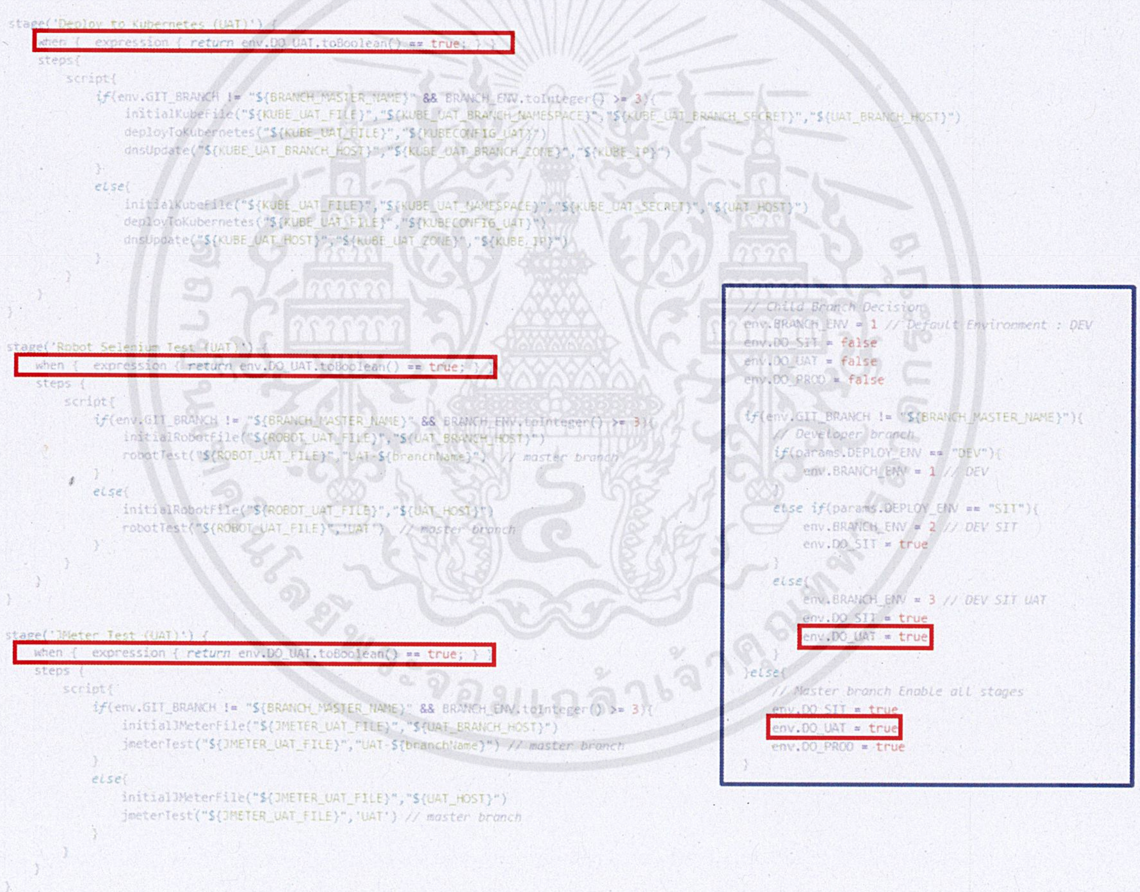


ภาพที่ 3.231 การตรวจสอบการทำงาน Environment SIT

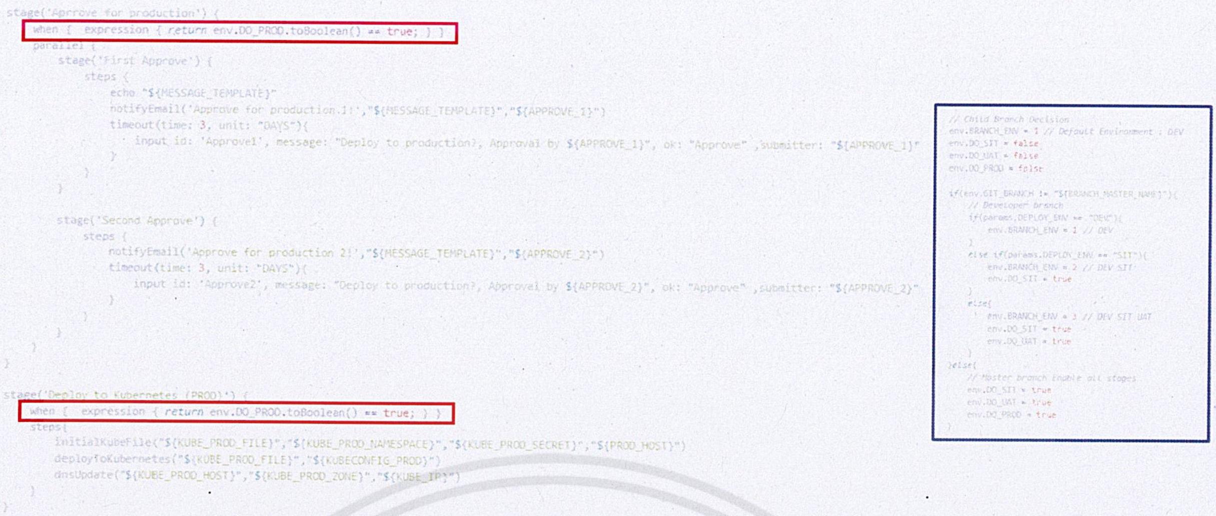
Approve เฉพาะ non-master branch



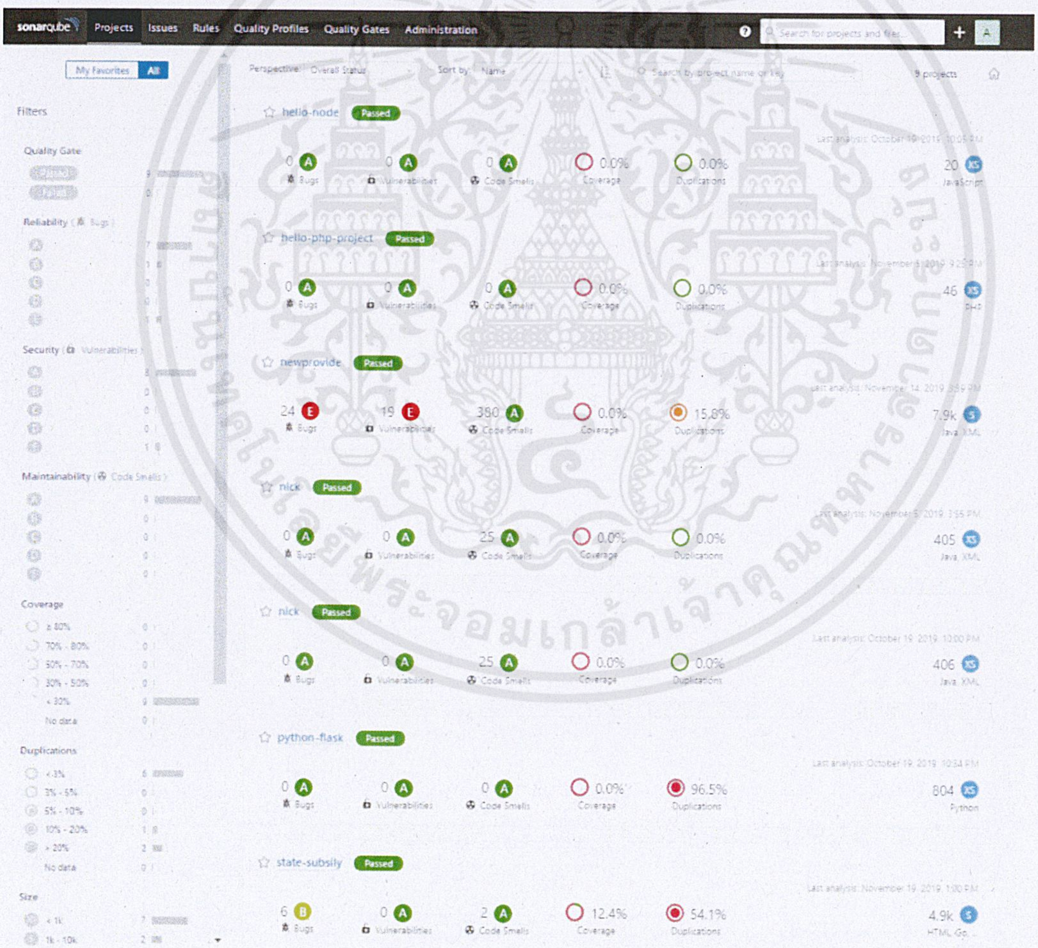
ภาพที่ 3.232 การ Approve สำหรับ Environment UAT บน Branch Non-Master



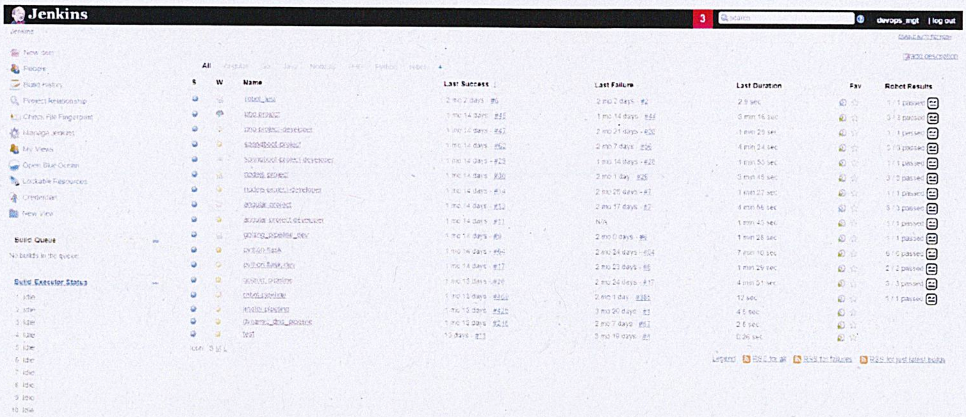
ภาพที่ 3.233 การตรวจสอบการทำงาน Environment UAT



ภาพที่ 3.234 การตรวจสอบการทำงาน Environment PROD



ภาพที่ 3.235 การทดสอบ SonarQube สำหรับทุกภาษา



ภาพที่ 3.236 Job ทั้งหมดที่ทำงานอยู่บนเครื่อง Jenkins

3.5.5.13 การทำ Horizontal Pod Auto Scaler

kubernetes-deployment > { } deployment-uat.yml >

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: {KUBE_NAMESPACE}-hpa
  namespace: {KUBE_NAMESPACE}
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: {KUBE_NAMESPACE}-deployment
  minReplicas: 1
  maxReplicas: 5
  targetCPUUtilizationPercentage: 10
```

ภาพที่ 3.237 การเพิ่มคำสั่งในไฟล์ Deployment ของ Kubernetes

```
# kubectl get hpa -n javauat
NAME          REFERENCE                          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
javauat-hpa   Deployment/hello-java-deployment   2%/10%    1         5         1          27h
```

ภาพที่ 3.238 การตรวจสอบ CPU ของ Pod ขณะเริ่มต้น

```
# kubectl get pod -n javauat
NAME                                    READY   STATUS    RESTARTS   AGE
hello-java-deployment-866b48f68c-x4mmf  1/1     Running   0          27h
```

ภาพที่ 3.239 การตรวจสอบจำนวน Pod ที่ทำงานอยู่

```
[root@vRA-CentOS-007 /]# /opt/apache-jmeter/bin/jmeter -n -t /tmp/http_uat_test.jmx
Creating summariser <summary>
Created the tree successfully using /tmp/http_uat_test.jmx
Starting the test @ Wed Nov 20 15:01:36 ICT 2019 (1574236896049)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary =  10000 in 00:00:04 = 2461.8/s Avg:     3 Min:     1 Max:    31 Err: 10000 (100.00%)
Tidying up ...    @ Wed Nov 20 15:01:40 ICT 2019 (1574236900373)
... end of run
```

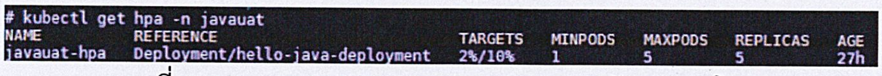
ภาพที่ 3.240 การทดสอบโหลดไปยัง Application

```
# kubectl get hpa -n javauat
NAME          REFERENCE                          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
javauat-hpa   Deployment/hello-java-deployment   322%/10%  1         5         1          27h
```

ภาพที่ 3.241 การตรวจสอบ CPU ของ Pod ขณะทดสอบโหลด

```
# kubectl get pod -n javauat
NAME                                    READY   STATUS    RESTARTS   AGE
hello-java-deployment-866b48f68c-9q6b2  1/1     Running   0          31s
hello-java-deployment-866b48f68c-jdzdk  1/1     Running   0          31s
hello-java-deployment-866b48f68c-vpcmm  1/1     Running   0          16s
hello-java-deployment-866b48f68c-wp26f  1/1     Running   0          31s
hello-java-deployment-866b48f68c-x4mmf  1/1     Running   0          27h
```

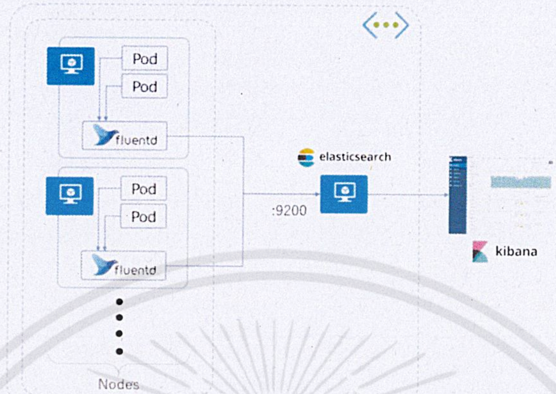
ภาพที่ 3.242 การตรวจสอบจำนวน Pod ทีทำงานอยู่หลังทดสอบโหลด



```
# kubectl get hpa -n javauat
NAME          REFERENCE                              TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
javauat-hpa   Deployment/hello-java-deployment       2%/10%    1         5         5          27h
```

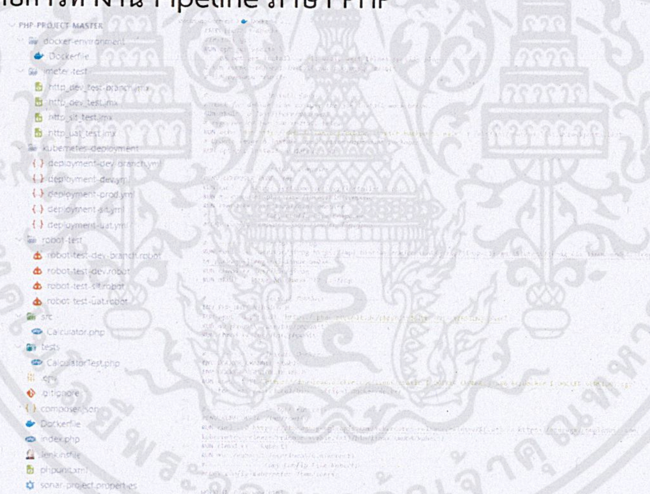
ภาพที่ 3.243 การตรวจสอบ CPU ของ Pod หลังทดสอบโหลด

3.5.5.14 การเก็บ Log ด้วย FluentD

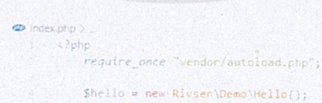


ภาพที่ 3.244 การเก็บ Log จาก Pod ต่างๆบน Kubernetes ด้วย FluentD

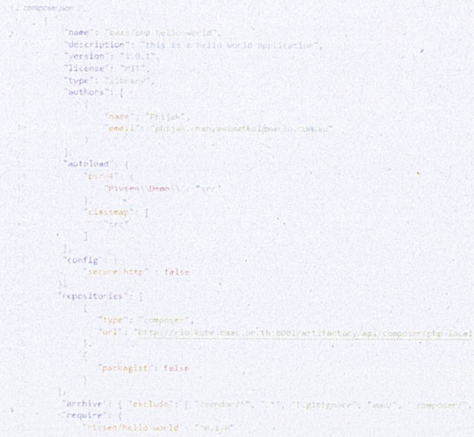
3.5.6 อธิบายการทำงาน Pipeline ภาษา PHP



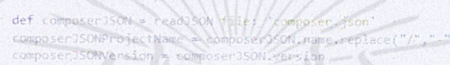
ภาพที่ 3.245 โครงสร้าง Project สำหรับภาษา PHP



ภาพที่ 3.246 รายละเอียดของไฟล์ index.php

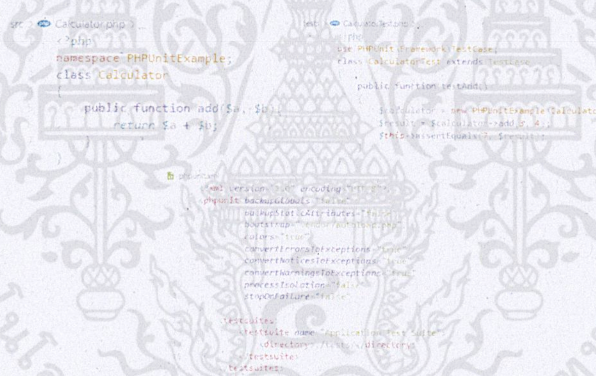


ภาพที่ 3.247 รายละเอียดของไฟล์ composer.json

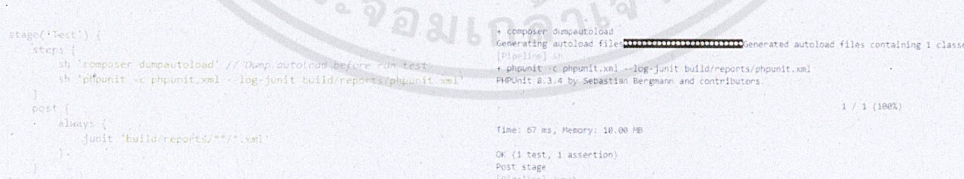


```
def composerJSON = readJSON file: 'composer.json'
composerJSONProjectName = composerJSON.name.replace("/","-")
composerJSONVersion = composerJSON.version
```

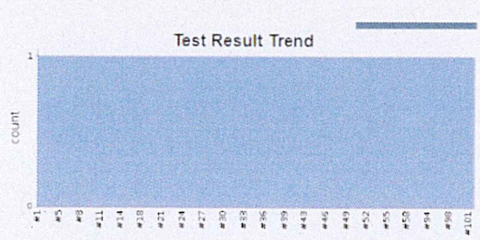
ภาพที่ 3.248 รายละเอียดการแก้ไขไฟล์ composer.json



ภาพที่ 3.249 ไฟล์ Unit Test สำหรับ PHP



ภาพที่ 3.250 คำสั่งและผลลัพธ์การทำ PHP Unit Test



ภาพที่ 3.251 รายงาน PHP Unit Test บน Jenkins

```
def composerJSON = readJSON file: 'composer.json'
composerJSON.repositories[0].url = ''
composerJSON.repositories[0].url = composerJSON.repositories[0].url.toString() + "${JFROG_ARTIFACTORY_URL}/api/composer/${JFROG_VIRTUAL_REPO}"
composerJSON.version = composerJSON.version.toString() + ".$BUILD_ID"
writeFile file:"composer.json", text:composerJSON.toString()

withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: JFROG_CREDENTIALS, usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
    sh "jfrog rt config --url=${JFROG_ARTIFACTORY_URL} --user=$USERNAME --password=$PASSWORD"
}
```

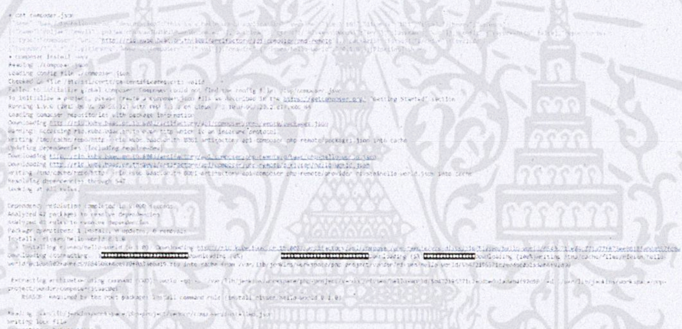
ภาพที่ 3.252 คำสั่งการอ่านค่าในไฟล์ composer.json ของ PHP

```
"require": {
    "rivsen/hello-world": "^0.1.0"
}
```

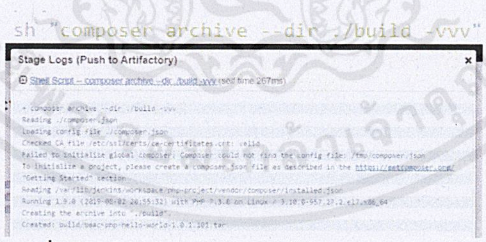
ภาพที่ 3.253 คำสั่ง PHP Unit Test

```
stage('Build') {
    steps {
        script {
            sh 'cat composer.json'
            sh 'composer install -vvv'
        }
    }
}
```

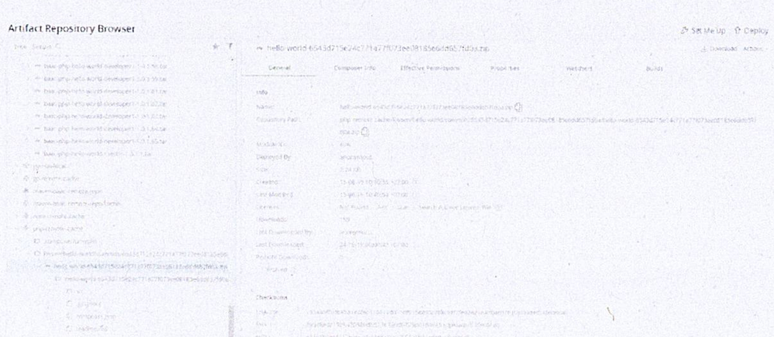
ภาพที่ 3.254 คำสั่งการ Install Library ด้วย Composer



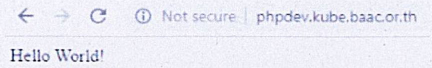
ภาพที่ 3.255 ผลลัพธ์การ Install Library ด้วย Composer



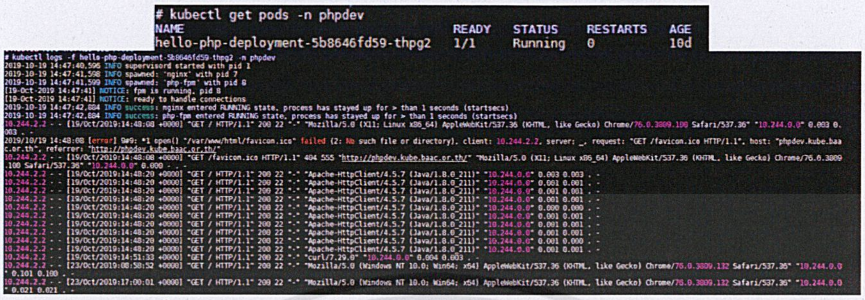
ภาพที่ 3.256 การทำ Archive ด้วย Composer



ภาพที่ 3.257 การเก็บไฟล์ Artifact ภาษา PHP บน JFrog



ภาพที่ 3.258 ผลการทำงานของ PHP

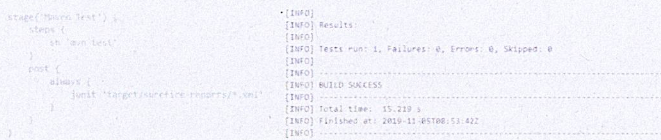


ภาพที่ 3.259 การตรวจสอบ Log ของ PHP บน Kubernetes

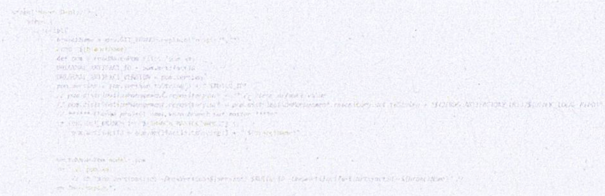
3.5.7 อธิบายการทำงาน Pipeline ภาษา Java (Spring Boot)

ภาพที่ 3.260 โครงสร้าง Project สำหรับภาษา Java

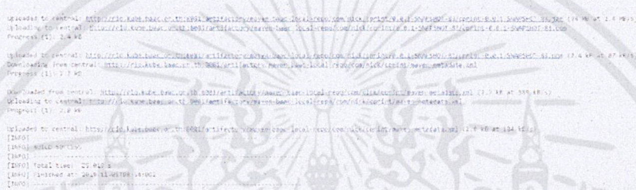
ภาพที่ 3.261 การแก้ไขไฟล์ setting.xml สำหรับ Maven



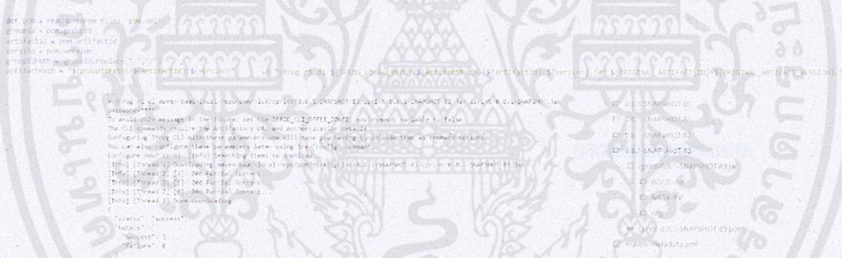
ภาพที่ 3.262 คำสั่งและผลลัพธ์การ Test ด้วย Maven



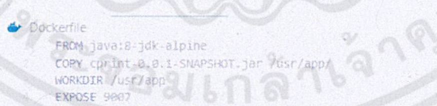
ภาพที่ 3.263 คำสั่งการ Deploy ด้วย Maven



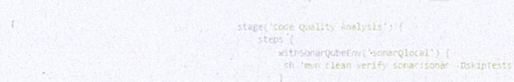
ภาพที่ 3.264 ผลลัพธ์การ Build และ Upload ไป JFrog ด้วย Maven



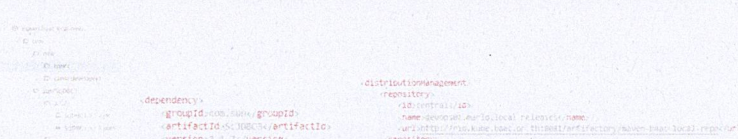
ภาพที่ 3.265 ผลลัพธ์การ Download Java Artifact ด้วย JFrog



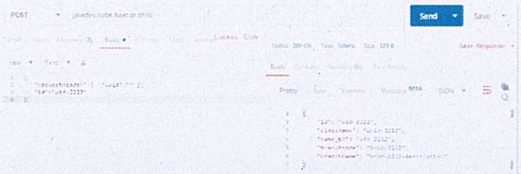
ภาพที่ 3.266 Dockerfile สำหรับ Java



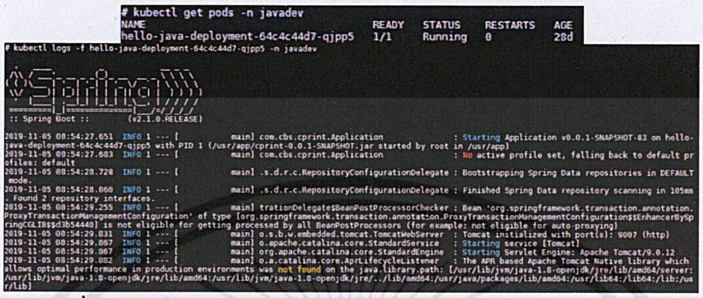
ภาพที่ 3.267 คำสั่งตรวจสอบ Code Quality ด้วย Maven



ภาพที่ 3.268 การใช้ Dependency ที่อยู่บน JFrog

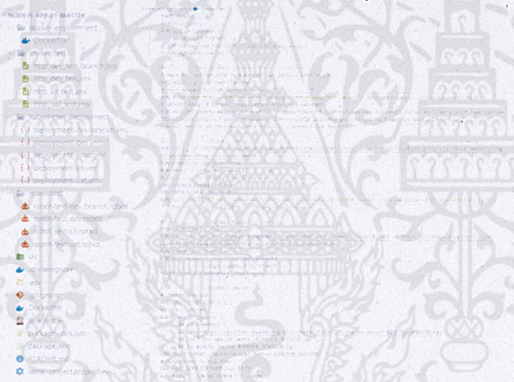


ภาพที่ 3.269 ผลการทำงาน Java (Spring Boot)

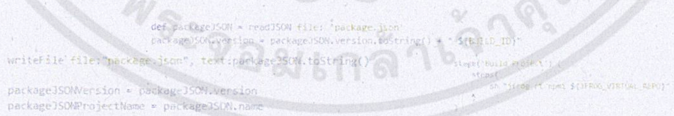


ภาพที่ 3.270 การตรวจสอบ Log ของ Java บน Kubernetes

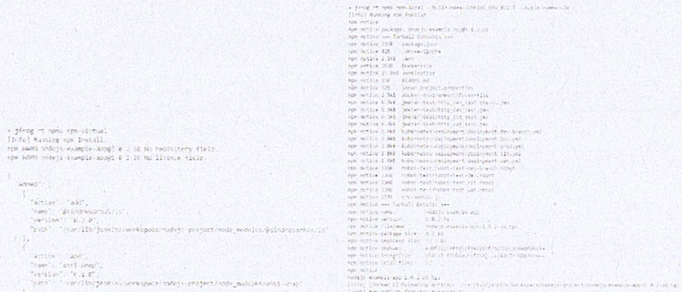
3.5.8 อธิบายการทำงาน Pipeline ภาษา JavaScript (NodeJS)



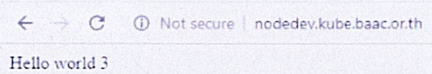
ภาพที่ 3.271 โครงสร้าง Project สำหรับภาษา JavaScript (NodeJS)



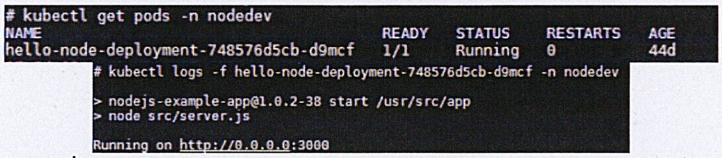
ภาพที่ 3.272 การอ่านไฟล์ package.json และการ Build



ภาพที่ 3.273 ผลลัพธ์การ Build ของ NodeJS

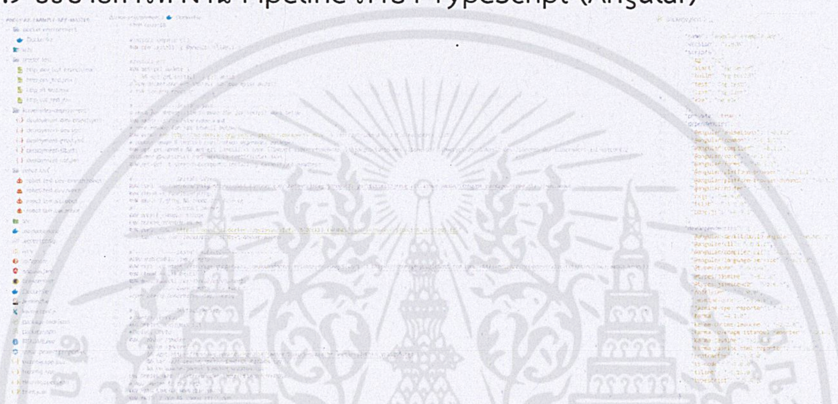


ภาพที่ 3.274 ผลลัพธ์การทำงานของ ของ NodeJS

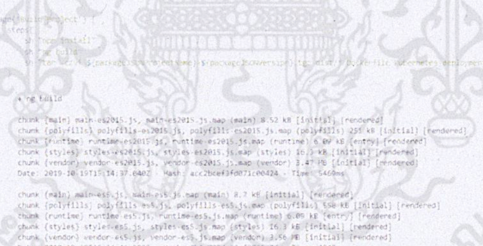


ภาพที่ 3.275 การตรวจสอบ Log ของ NodeJS บน Kubernetes

3.5.9 อธิบายการทำงาน Pipeline ภาษา TypeScript (Angular)

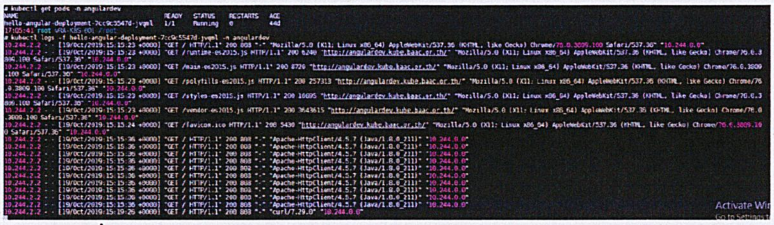


ภาพที่ 3.276 โครงสร้าง Project สำหรับภาษา TypeScript (Angular)



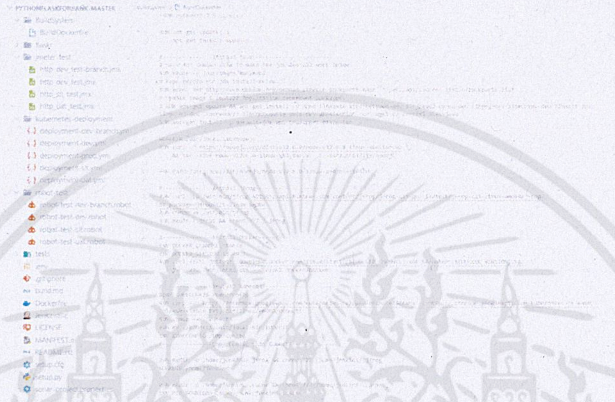
ภาพที่ 3.277 คำสั่งและผลลัพธ์การ Build ด้วย Angular CLI

ภาพที่ 3.278 ผลการทำงานของ Angular

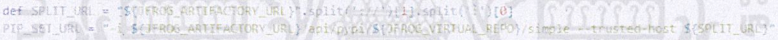


ภาพที่ 3.279 การตรวจสอบ Log ของ Angular บน Kubernetes

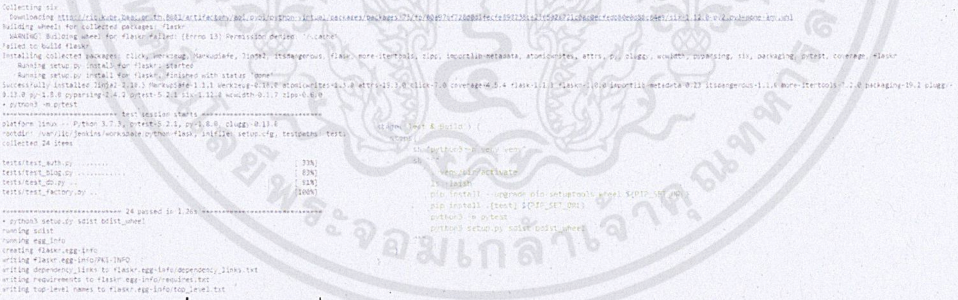
3.5.10 อธิบายการทำงาน Pipeline ภาษา Python



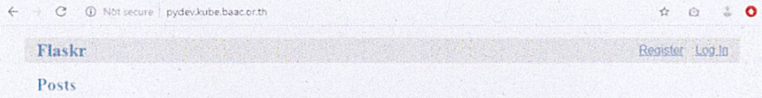
ภาพที่ 3.280 โครงสร้าง Project สำหรับภาษา Python



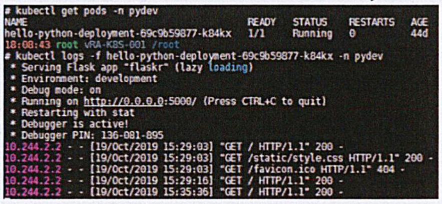
ภาพที่ 3.281 การ Config JForg บนภาษา Python



ภาพที่ 3.282 คำสั่งและผลลัพธ์การ Build และ Test ด้วย Python



ภาพที่ 3.283 ผลการทำงานของ Python



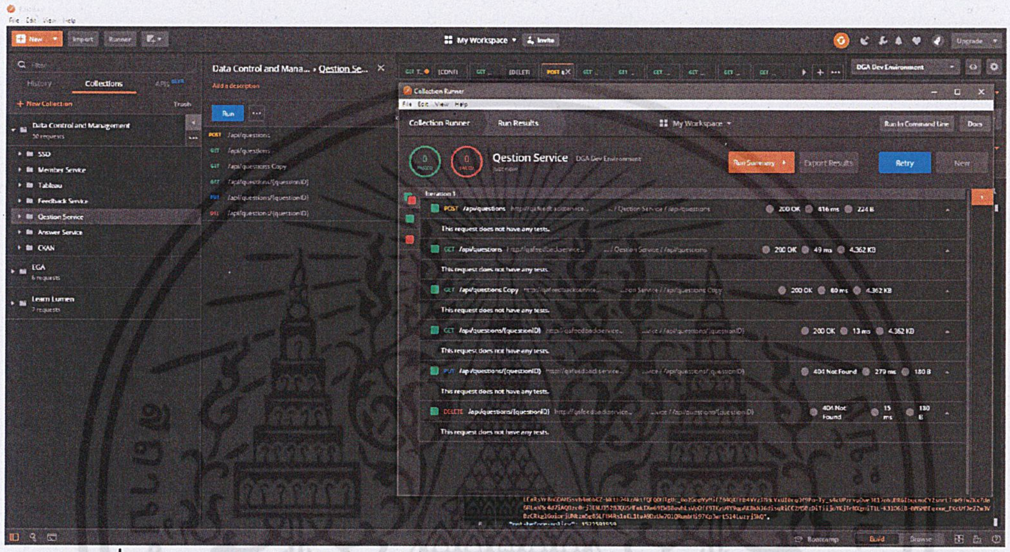
ภาพที่ 3.284 การตรวจสอบ Log ของ Python บน Kubernetes

บทที่ 4
การทดสอบและผลการทดสอบ

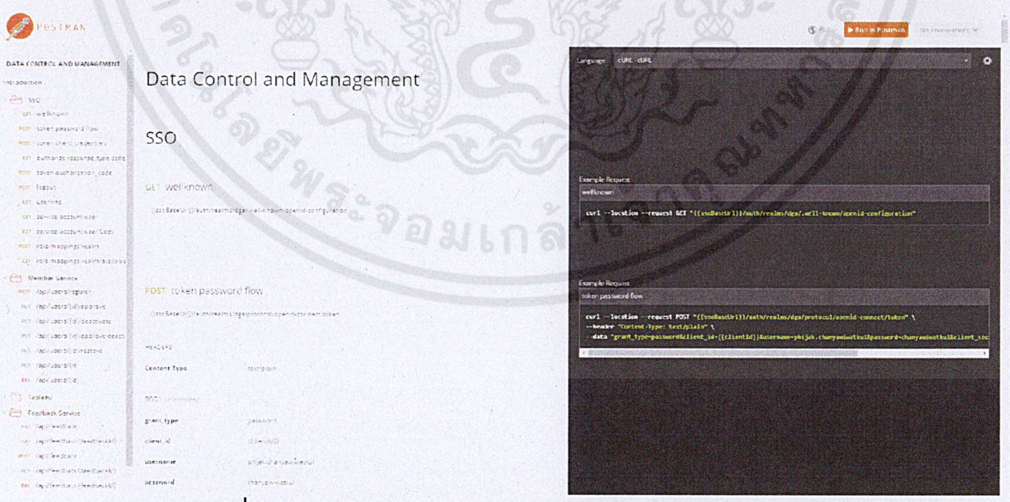
4.1 การทดสอบระบบการควบคุมและจัดการข้อมูล

4.1.1 โปรแกรมที่ใช้ทดสอบระบบ

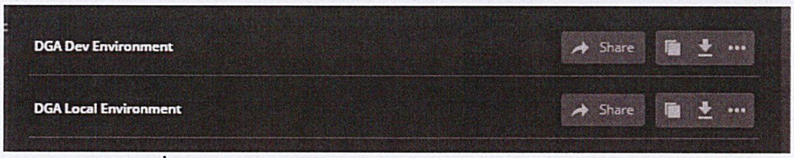
ใช้โปรแกรม Postman ในการทดสอบ API โดยจัดทำเป็น Collections และใช้ Environment เป็นตัวแปรในการแยกทดสอบในส่วนของ Local และการทำงานบน Dev Environment โดยสามารถใช้ Collection ทำ API Automate Test ได้



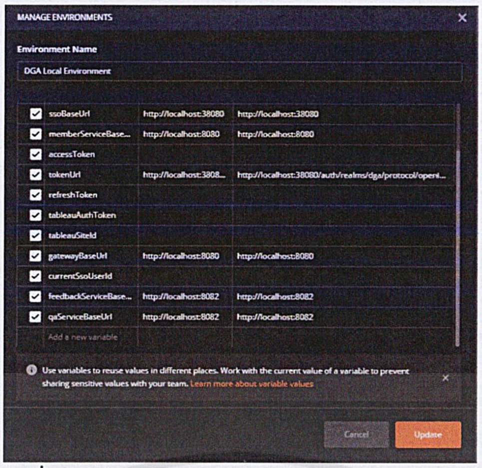
ภาพที่ 4.1 การจัดเก็บ API Request เป็น Collection และการทำ Automate Test



ภาพที่ 4.2 การใช้ Postman จัดทำเอกสาร API Document



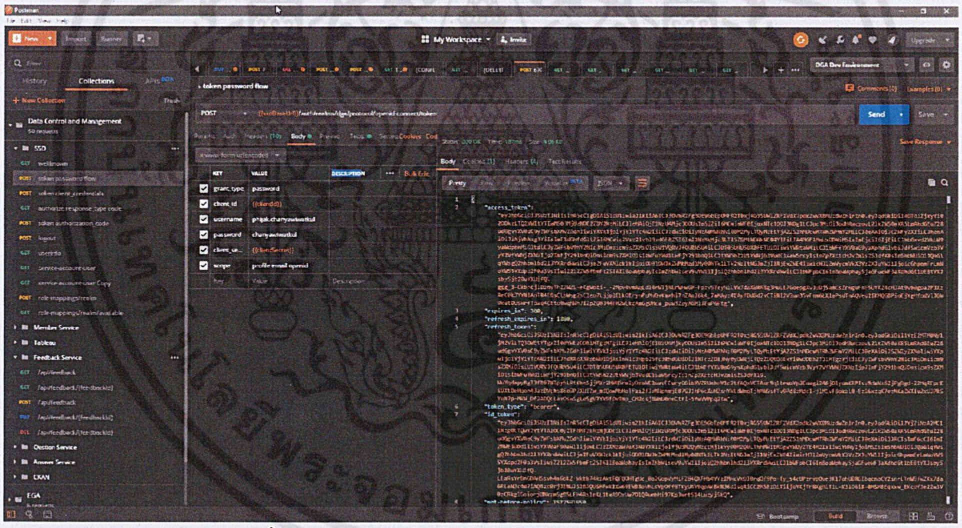
ภาพที่ 4.3 การแยก Environment สำหรับทดสอบ API



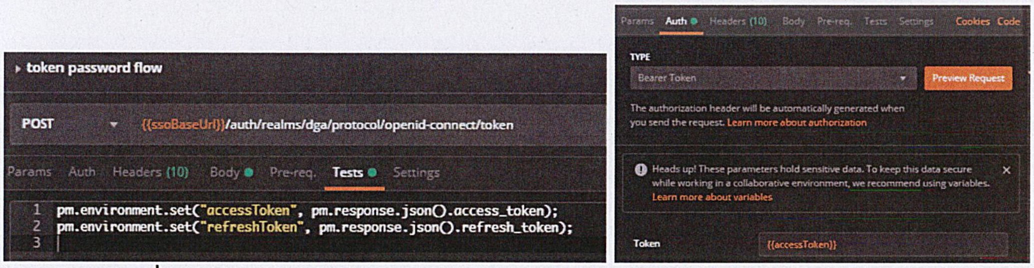
ภาพที่ 4.4 Environment Local ของ Postman

4.1.2 การทดสอบระบบ

ใช้ Postman ทำการส่ง Request ไปยัง API Service ทั้งหมดตาม API Document ที่ได้ออกแบบไว้ในบทที่ 3



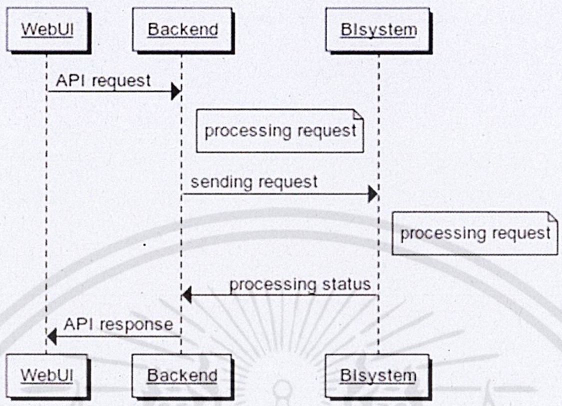
ภาพที่ 4.5 ตัวอย่างการส่ง Request บน Postman



ภาพที่ 4.6 การเก็บและใช้ Access Token ไว้เป็นตัวแปรสำหรับส่งไป Request

4.2 ผลการทดสอบระบบ

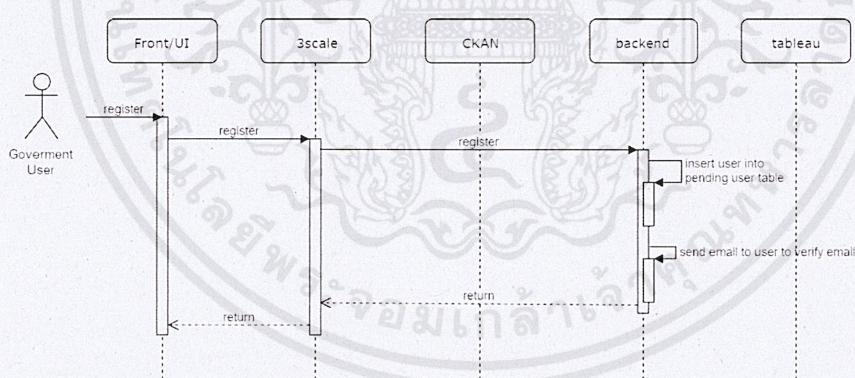
การทดสอบระบบนั้น จะถูกทดสอบในทุกส่วนของ Data Control and Management เริ่มตั้งแต่ Authentication Services ไปสิ้นสุดที่ QA Services การทดสอบระดับ System Integration Test นั้น สามารถเขียนเป็น Sequence Diagram เพื่อเรียงลำดับการทดสอบได้ดัง Sequence Diagram



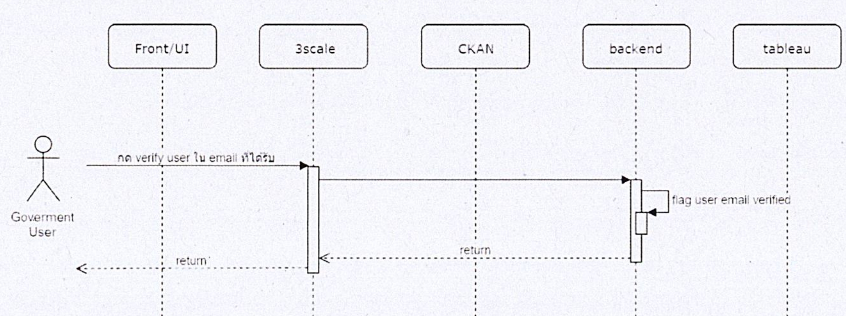
ภาพที่ 4.7 Sequence Diagram เรียงลำดับการทดสอบ System Integration Test

4.2.1 Registration/Login Scenario

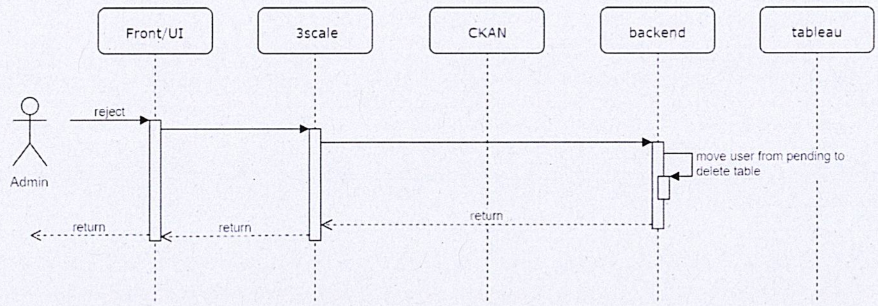
การทำงานของ Registration การ Login และการ Delete User นั้น สามารถเรียงลำดับการทำงานได้จาก sequence diagram



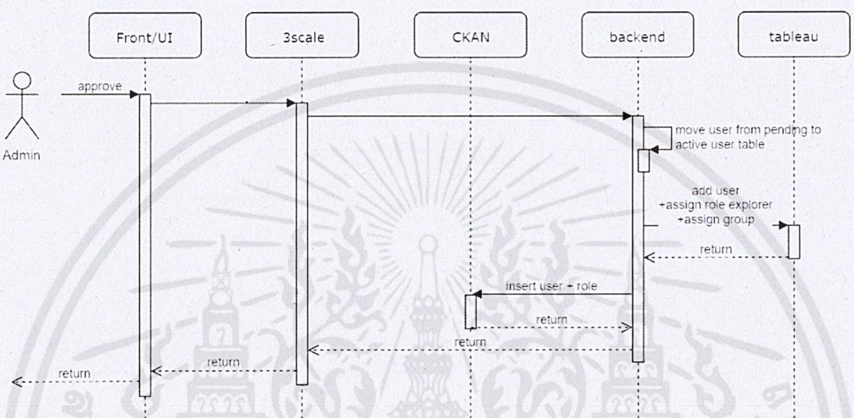
ภาพที่ 4.8 การทำงานรวมของการ Registration ผ่าน Sequence Diagram



ภาพที่ 4.9 การทำงานรวมของการ Approval New User ผ่าน Sequence Diagram



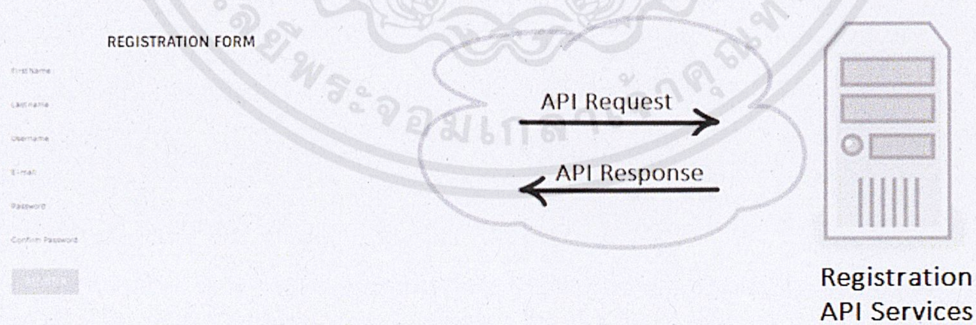
ภาพที่ 4.10 การทำงานรวมของการ Reject New User ผ่าน Sequence Diagram



ภาพที่ 4.11 การทำงานรวมของการ Re-Activate User ผ่าน Sequence Diagram

4.2.1.1 New User Registration

New User Registration นั้น จะเริ่มต้นจาก Frontend-UI โดย Frontend UI จะทำการส่งข้อมูลการ Registration มายัง API เพื่อขอทำการ Registration

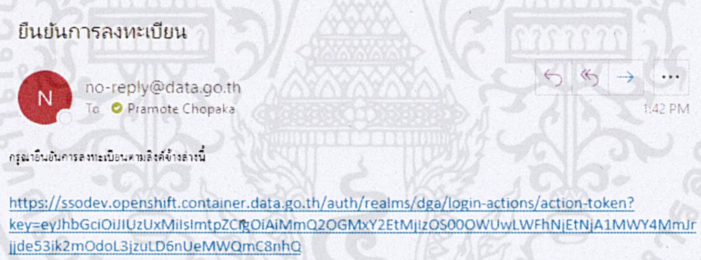


ภาพที่ 4.12 การทำงานของการ Registration

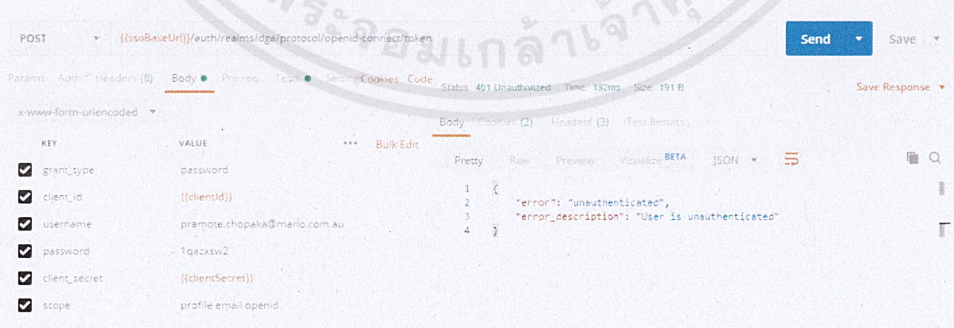
ตารางที่ 4.1 New User Registration

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Registration พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	เรียกใช้งาน API ตามรูปแบบที่ API กำหนด
Expect result	API Registration ทำงานได้อย่างถูกต้อง User ที่สมัครเข้ามาใหม่ อยู่ใน Stage รอการ Approve User ที่สมัครเข้ามาใหม่ ยังไม่สามารถ Login ได้

ภาพที่ 4.13 Test Register

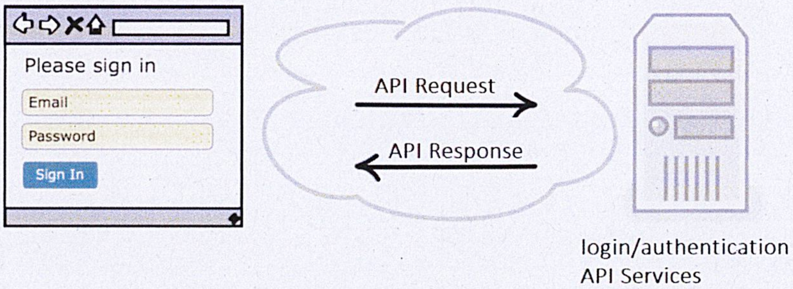


ภาพที่ 4.14 Test Email ยืนยันการลงทะเบียน



ภาพที่ 4.15 Test User ยังไม่สามารถ Login ได้

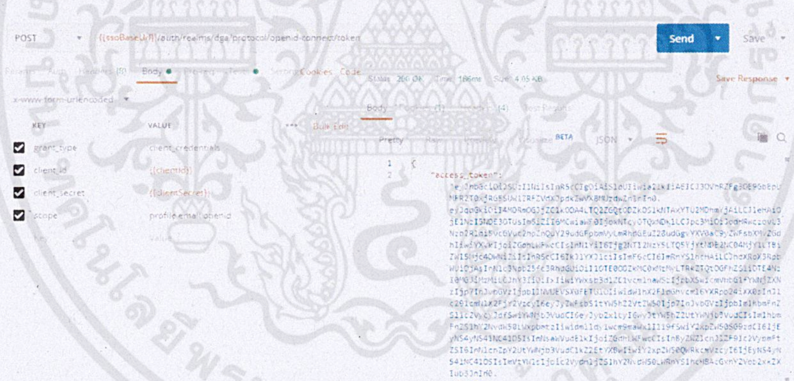
4.2.1.2 Login Authentication and Authorization



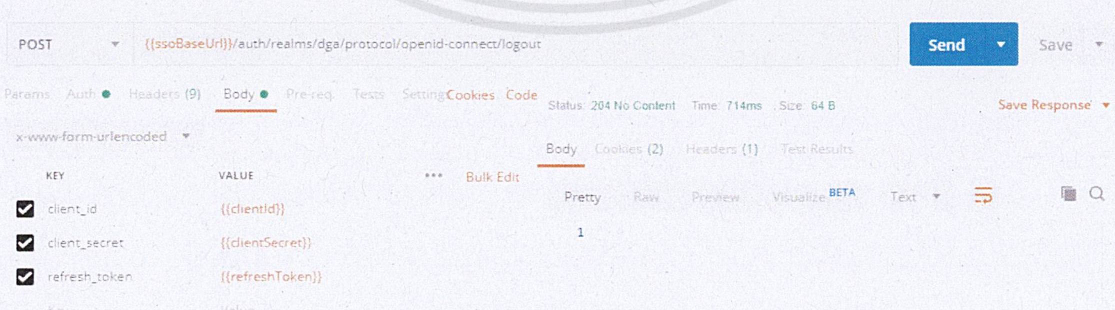
ภาพที่ 4.16 รูปแสดงการทำงานของการ Login

ตารางที่ 4.2 Login Authentication and Authorization

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Login และ Logout พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	Frontend UI เรียกใช้งาน API Login ด้วย API Spec ตามรูปแบบที่ API กำหนด
Expect result	API Login/Logout ทำงานได้อย่างถูกต้อง API Login return token key ตามรูปแบบที่กำหนด



ภาพที่ 4.17 Test Login

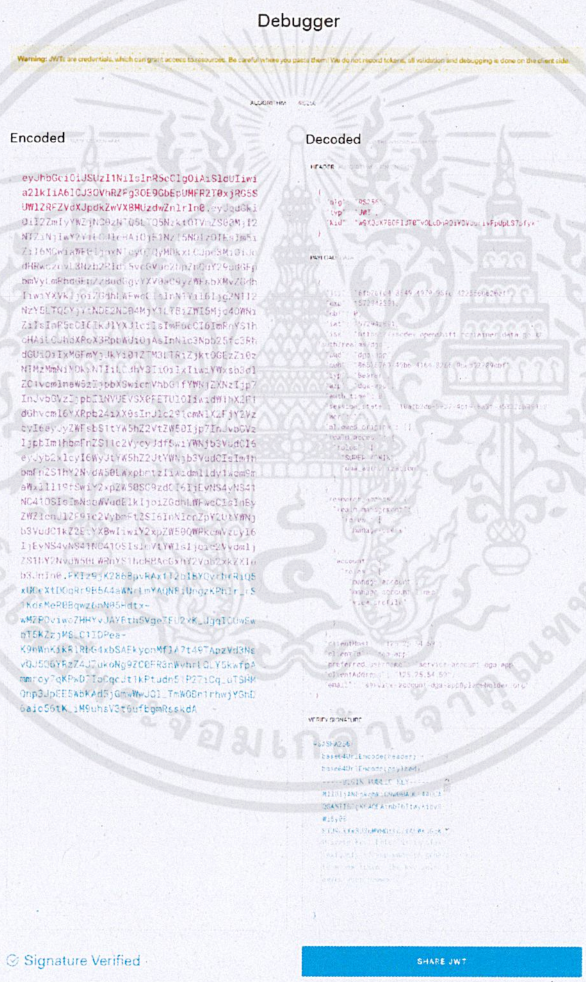


ภาพที่ 4.18 Test Logout

4.2.1.3 Authorization API Return in Token with Information

ตารางที่ 4.3 Authorization API Return in Token with Information

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Login และ Logout พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	Frontend UI เรียกใช้งาน API Login ด้วย API Spec ตามรูปแบบที่ API กำหนด
Expect result	API Login/Logout ทำงานได้อย่างถูกต้อง API Login return token key ตามรูปแบบที่กำหนด System ที่เรียกใช้งาน API ได้รับ Token และสามารถนำไปใช้งานได้ System ปลายทางตรวจสอบ Token ที่ได้รับ ให้ตรงกับ token ที่ได้ส่งออกไป



ภาพที่ 4.19 Test Decoded Token

4.2.1.4 System Admin Manage User

ตารางที่ 4.4 System Admin Manage User

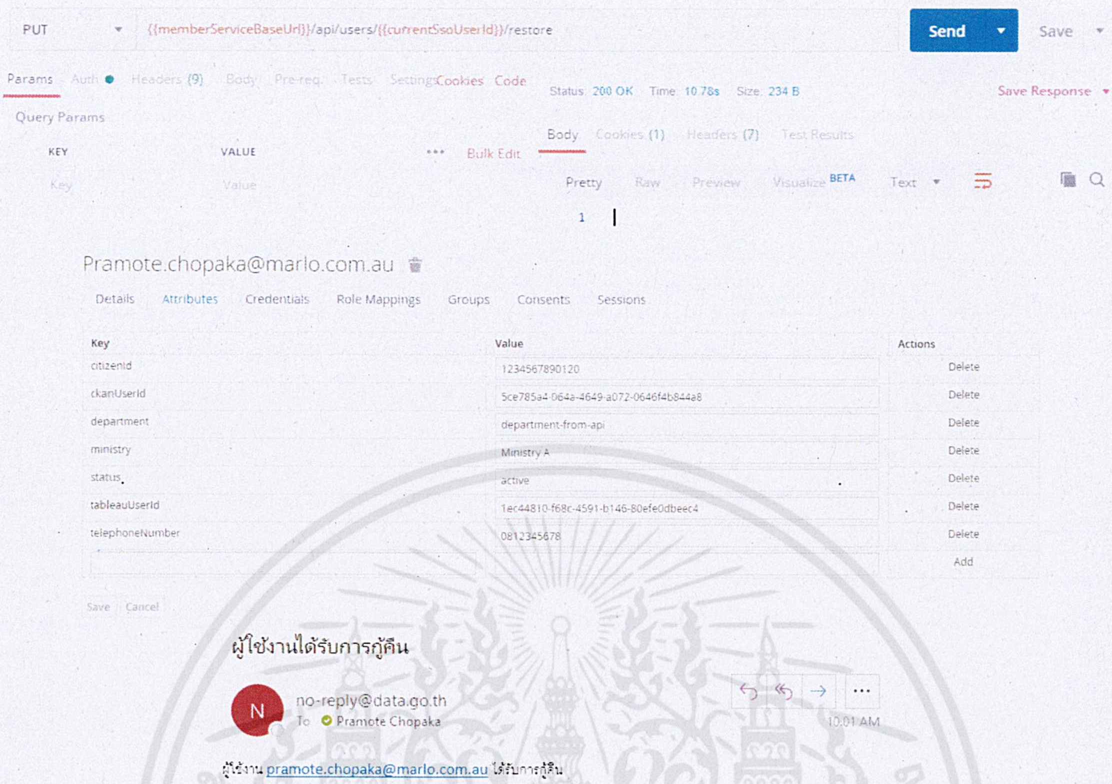
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Approve/Approve Deactivate/Restore/Delete ติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	Frontend UI เรียกใช้งาน API Login ด้วย API Spec ตามรูปแบบที่ API กำหนด
Expect result	API Approve/Approve Deactivate/Restore/Delete ทำงานได้อย่างถูกต้อง API return response การทำงานกลับไปได้อย่างถูกต้อง

ภาพที่ 4.20 Test Approve

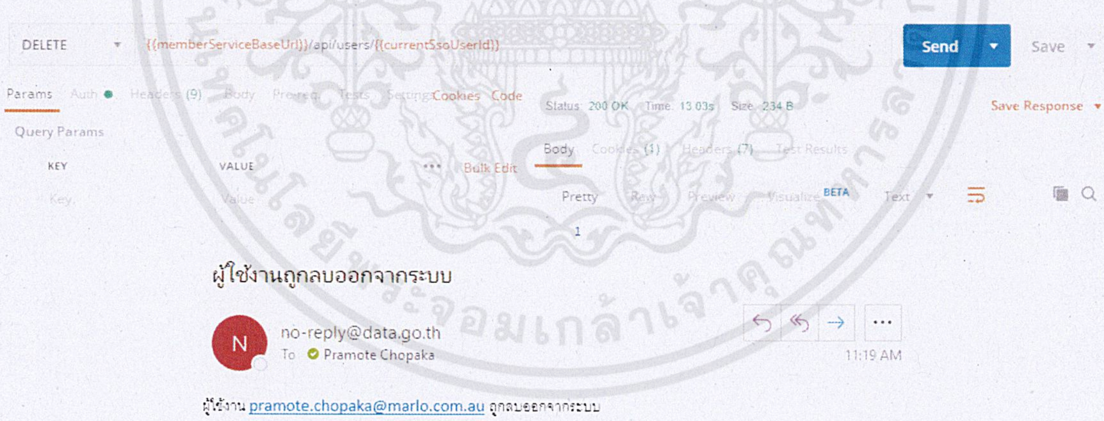
ภาพที่ 4.21 Test Email Approve

ภาพที่ 4.22 Test Deactivate

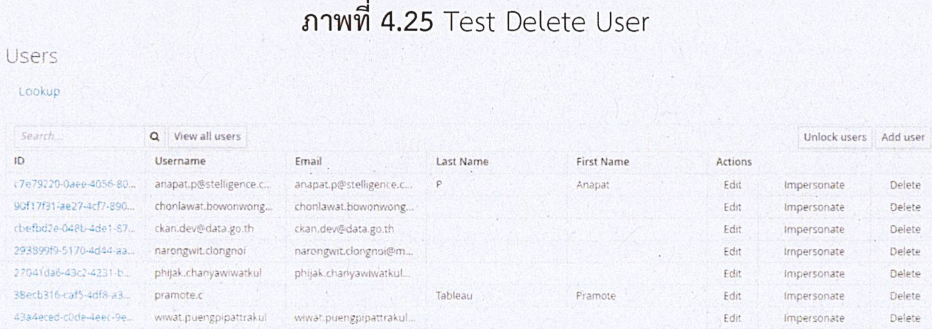
ภาพที่ 4.23 Test Approve Deactivate



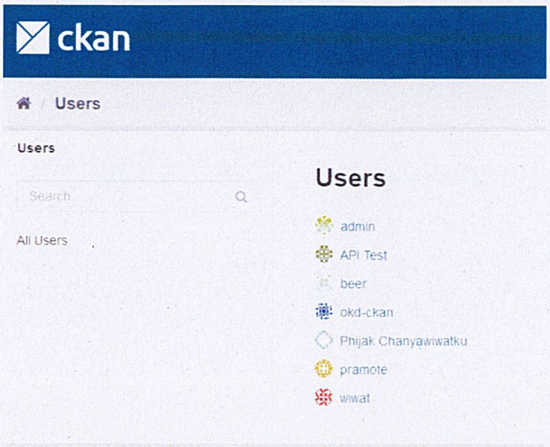
ภาพที่ 4.24 Test Restore



ภาพที่ 4.25 Test Delete User



ภาพที่ 4.26 Test RedHat SSO User



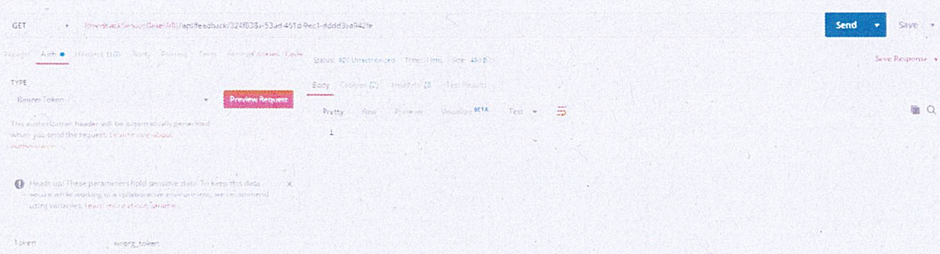
ภาพที่ 4.27 Test CKAN User

ภาพที่ 4.28 Test Tableau User

4.2.1.5 API Unauthorized

ตารางที่ 4.5 API Unauthorized

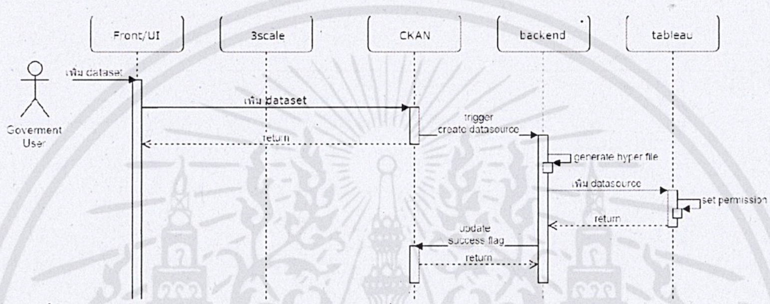
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API เชื่อมต่อไปตรวจสอบ Role ที่ KeyCloak สำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	Frontend UI เรียกใช้งาน API Login ด้วย API Spec ตามรูปแบบที่ API กำหนด
Expect result	API ตรวจสอบ Role และเปิดสิทธิ์การเข้าใช้งานได้อย่างถูกต้อง API return response การทำงานกลับไปได้อย่างถูกต้อง



ภาพที่ 4.29 Test API Unauthorized

4.2.2 Dataset Scenario

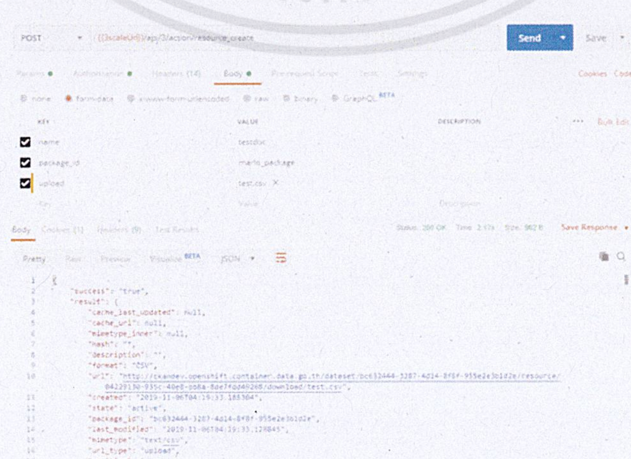
4.2.2.1 New/Upload Dataset



ภาพที่ 4.30 การทำงานรวมของการ เพิ่ม Dataset ผ่าน Sequence Diagram

ตารางที่ 4.6 New/Upload Dataset

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Upload DataSet พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ upload API Frontend UI เรียกใช้งาน API Upload Dataset ตามรูปแบบที่ API กำหนด
Expect result	Dataset ชุดใหม่ถูกสร้างในระบบ Dataset ชุดใหม่ ได้ถูกส่งข้อมูลไปให้ BI system

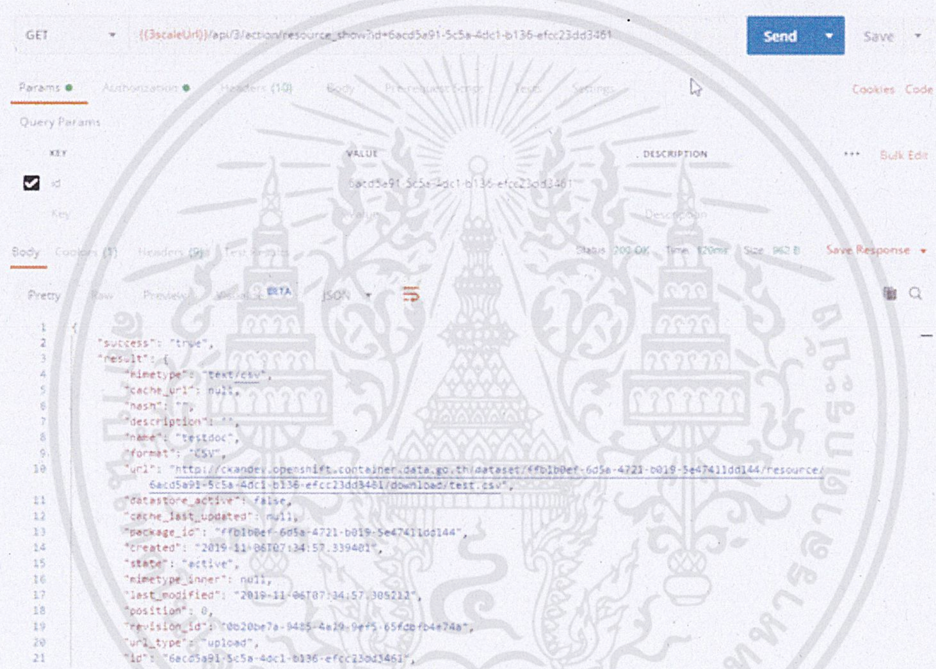


ภาพที่ 4.31 Test Upload Dataset

4.2.2.2 Download Dataset

ตารางที่ 4.7 Download Dataset

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Download Dataset พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Download Dataset API Frontend UI เรียกใช้งาน API Download Dataset ด้วยรูปแบบที่ API กำหนด
Expect result	Dataset ถูก download ผ่านการทำงานในรูปแบบ API

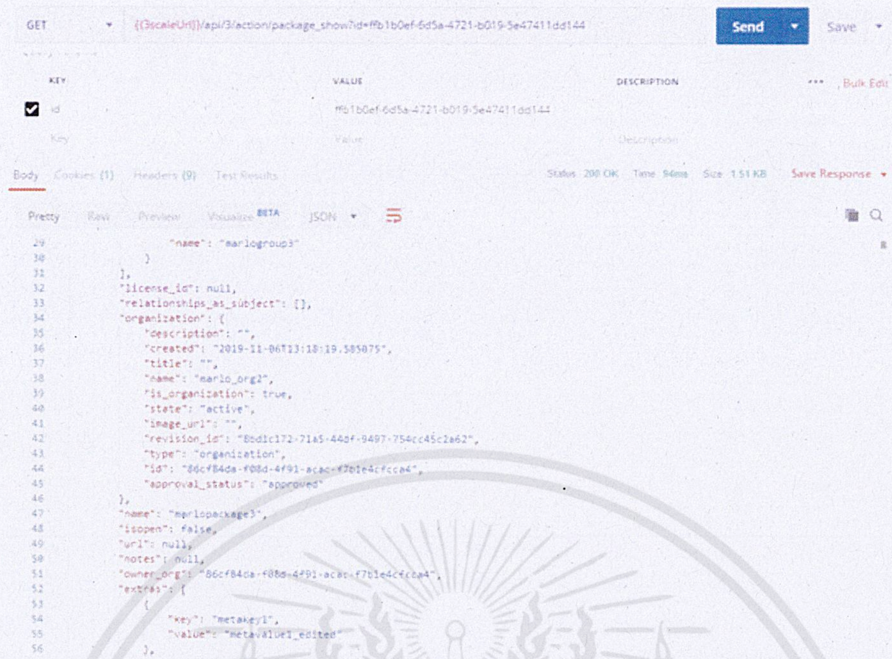


ภาพที่ 4.32 Test Download Dataset

4.2.2.3 Download Dataset Metadata

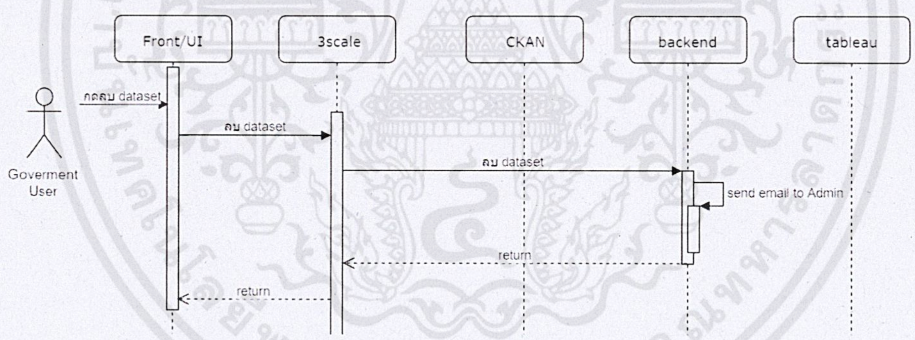
ตารางที่ 4.8 Download Dataset Metadata

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Download DataSet พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ผู้เรียกใช้งานมีสิทธิ์ใช้ Download Dataset Metadata API Frontend UI เรียกใช้งาน API Download Dataset ตามรูปแบบที่ API กำหนด
Expect result	Dataset Metadata ถูก download ผ่านการทำงานในรูปแบบ API



ภาพที่ 4.33 Test Download Dataset Metadata

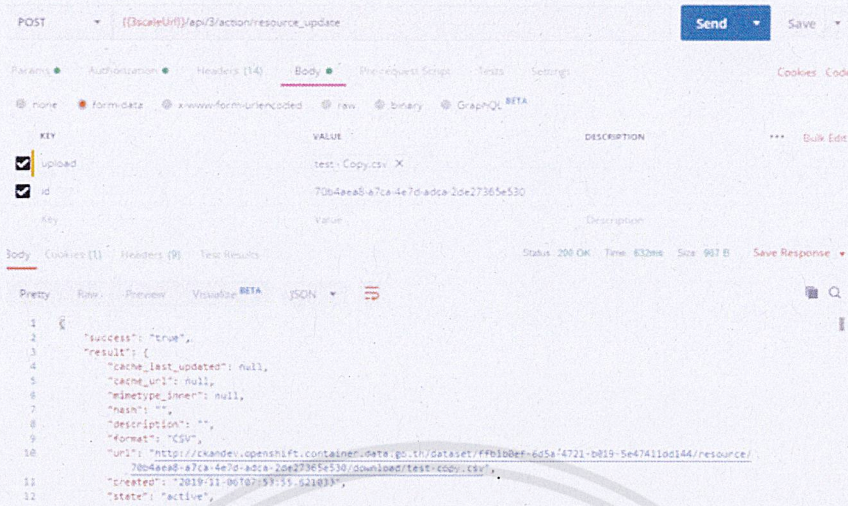
4.2.2.4 Update/Delete Dataset



ภาพที่ 4.34 การทำงานรวมของการลบ Dataset ผ่าน Sequence Diagram

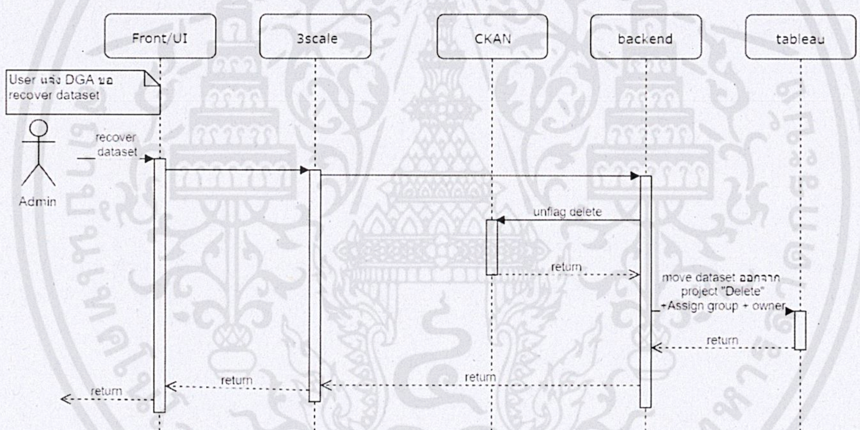
ตารางที่ 4.9 Update/Delete Dataset

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Update/Delete Dataset พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Update/Delete Dataset API Frontend UI เรียกใช้งาน API Upload Dataset ตามรูปแบบที่ API กำหนด
Expect result	Dataset สามารถแก้ไข และลบได้ผ่านการทำงานในรูปแบบ API



ภาพที่ 4.35 Test Update/Delete Dataset

4.2.2.5 Re-Activate Dataset

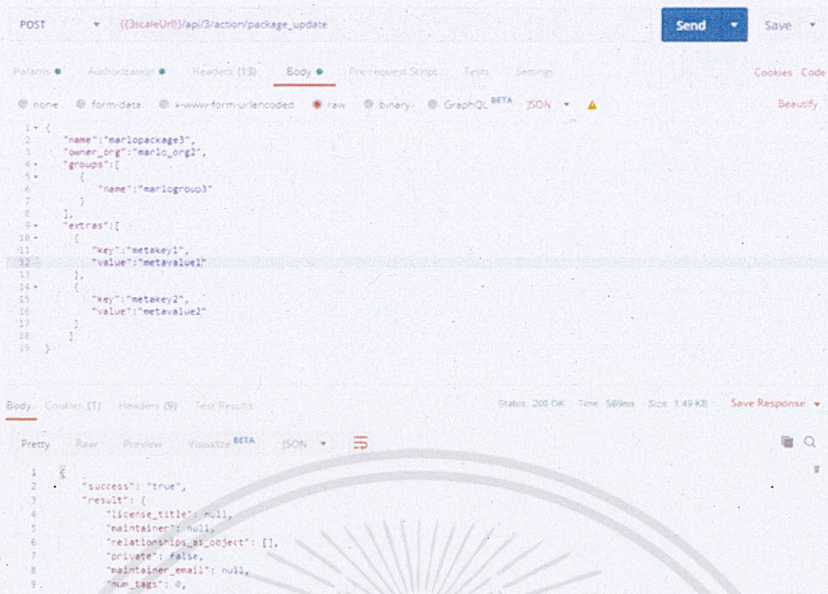


ภาพที่ 4.36 การทำงานรวมของการ Re-Activate Dataset

4.2.2.6 Add Dataset Metadata

ตารางที่ 4.10 Add Dataset Metadata

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Add Metadata พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Add Metadata Dataset API Frontend UI เรียกใช้งาน API Add Dataset metadata ตามรูปแบบที่ API กำหนด
Expect result	Dataset metadata ถูกเพิ่มเข้ามาในระบบ ผ่านการทำงานในรูปแบบ API ได้

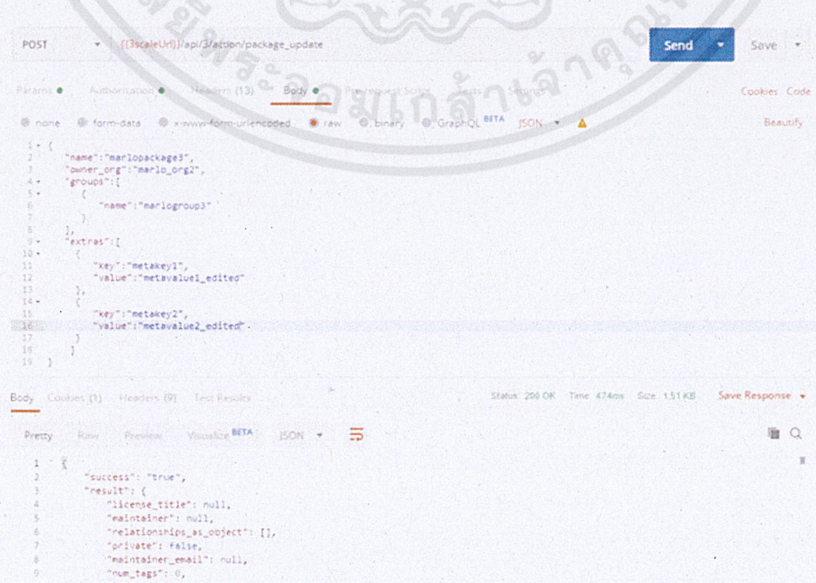


ภาพที่ 4.37 Test Add Dataset Metadata

4.2.2.7 Add/Update/Delete Dataset Metadata

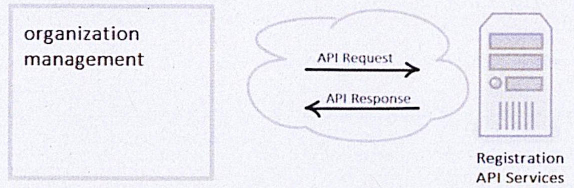
ตารางที่ 4.11 Add/Update/Delete Dataset Metadata

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Update/Delete Dataset Metadata พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ผู้เรียกใช้งาน มีสิทธิ์ในการ Update/Delete Metadata Dataset API Frontend UI เรียกใช้งาน API Dataset metadata ตามรูปแบบที่ API กำหนด
Expect result	Dataset metadata สามารถ download ได้ผ่านการทำงานในรูปแบบ API

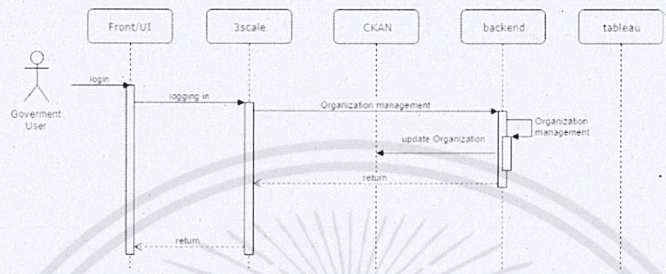


ภาพที่ 4.38 Test Add/Update/Delete Dataset Metadata

4.2.3 Organization Scenario



ภาพที่ 4.39 การทำงานของการจัดการ Organization

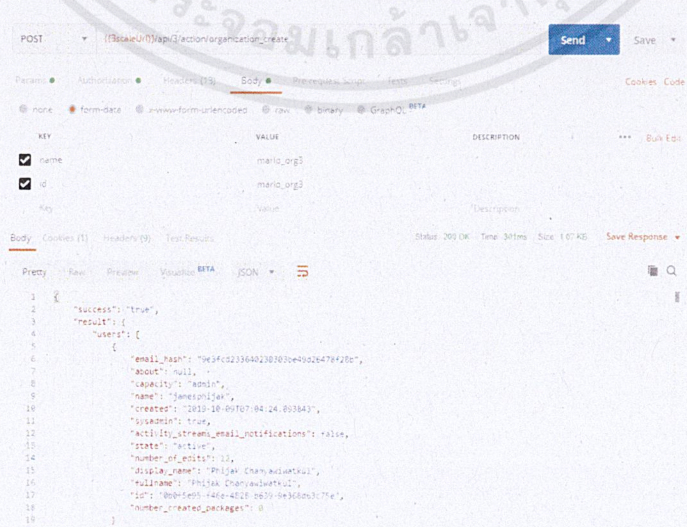


ภาพที่ 4.40 การทำงานของ Organization API

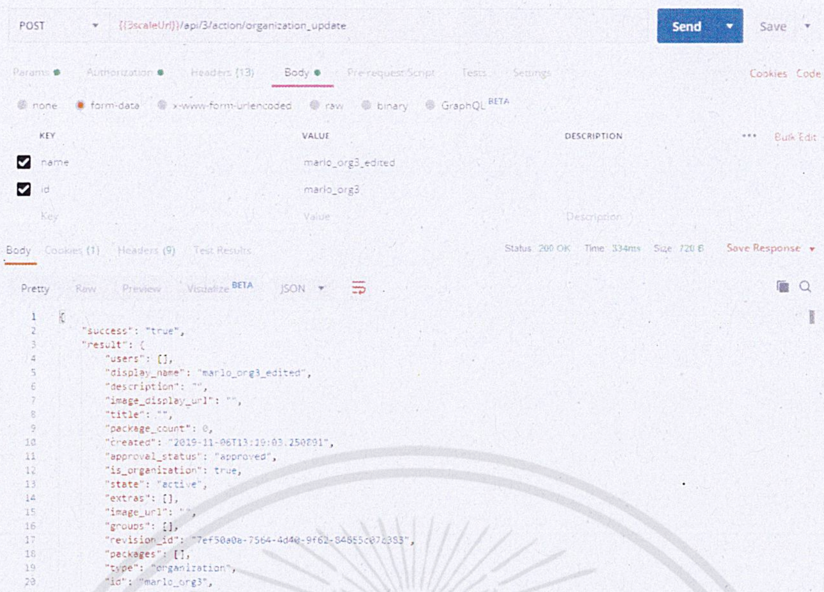
4.2.3.1 Organization Management

ตารางที่ 4.12 Organization Management

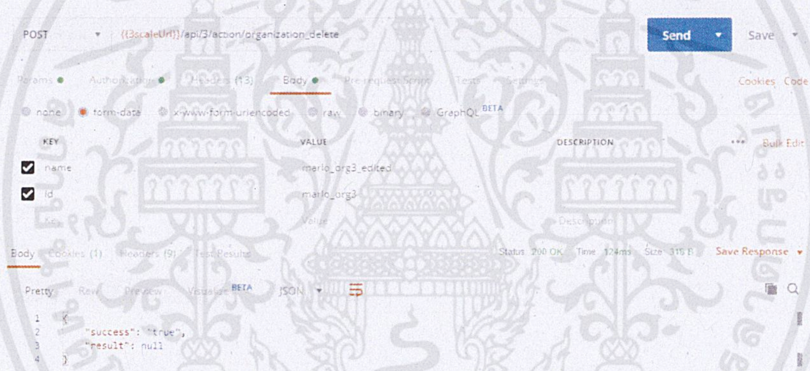
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Organization Management พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ในการ Manage Organization Frontend UI เรียกใช้งาน API Manage Organization ตามรูปแบบที่ API กำหนด
Expect result	API Organization Management สามารถทำงานได้ถูกต้อง ทั้ง Add/update/Delete API Return Response code ได้ถูกต้อง



ภาพที่ 4.41 Test Create Organization



ภาพที่ 4.42 Test Update Organization

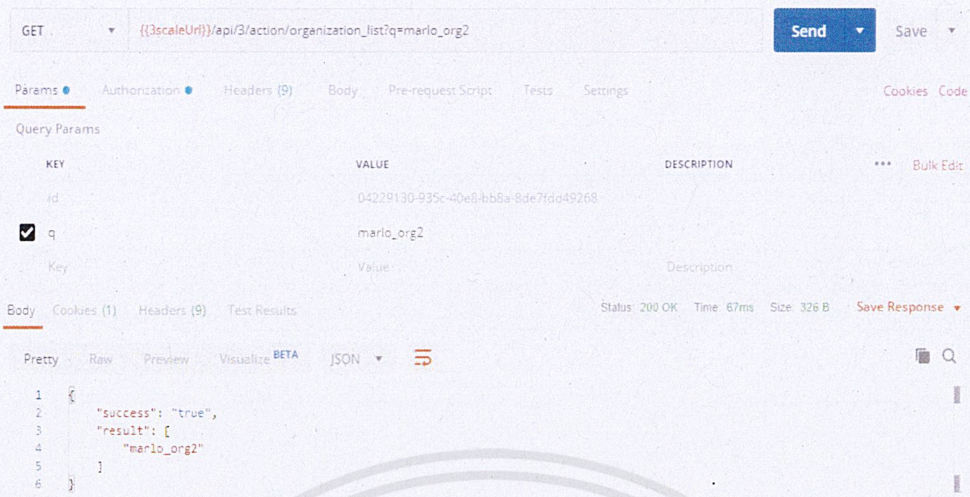


ภาพที่ 4.43 Test Delete Organization

4.2.3.2 Organization Search

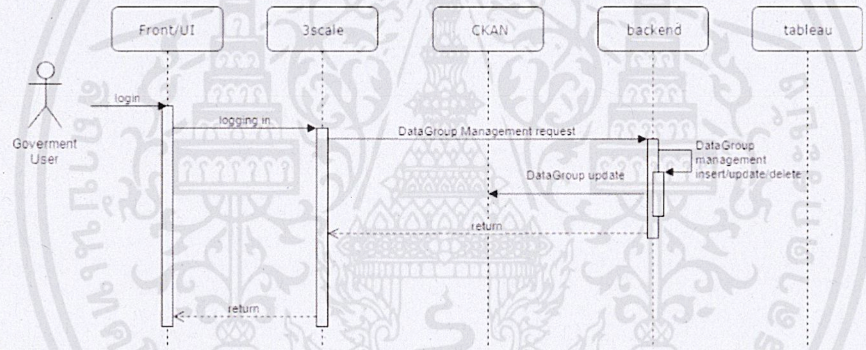
ตารางที่ 4.13 Organization Search

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Organization Searching พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น สำหรับทำการ searching พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ผู้เรียกใช้งาน มีสิทธิ์ในการเรียกใช้งาน Searching Organization Frontend UI เรียกใช้งาน API Searching Organization ตามรูปแบบที่ API กำหนด
Expect result	API Organization Searching สามารถทำการค้นหาข้อมูลได้ถูกต้อง API Return Response code ได้ถูกต้อง



ภาพที่ 4.44 Test Organization Search

4.2.4 Data Group/Catalog Scenario

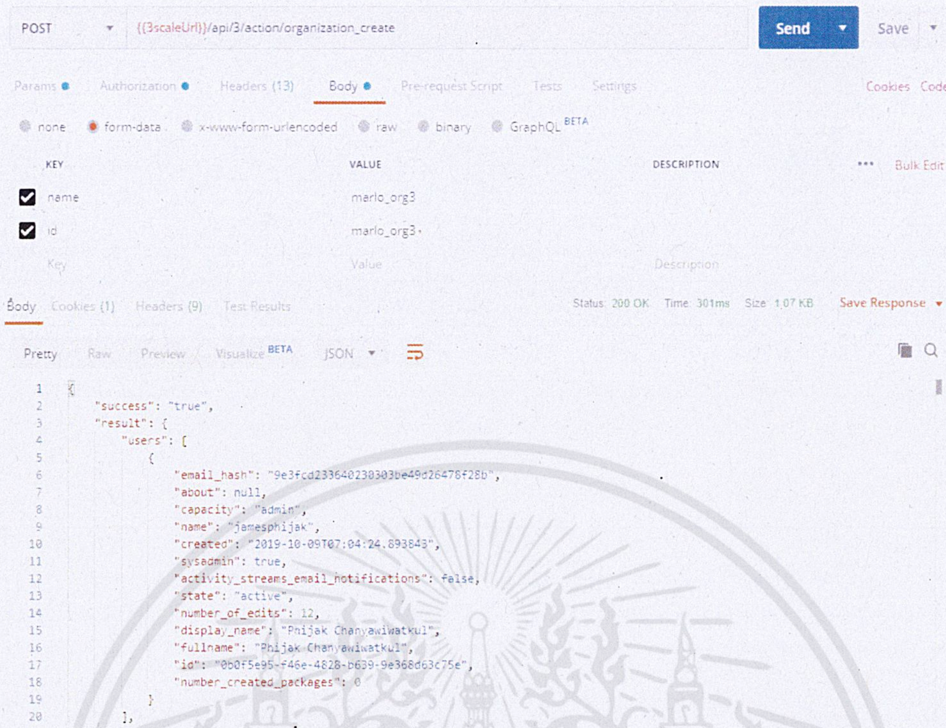


ภาพที่ 4.45 การทำงานของ Data Group และ Data Catalog

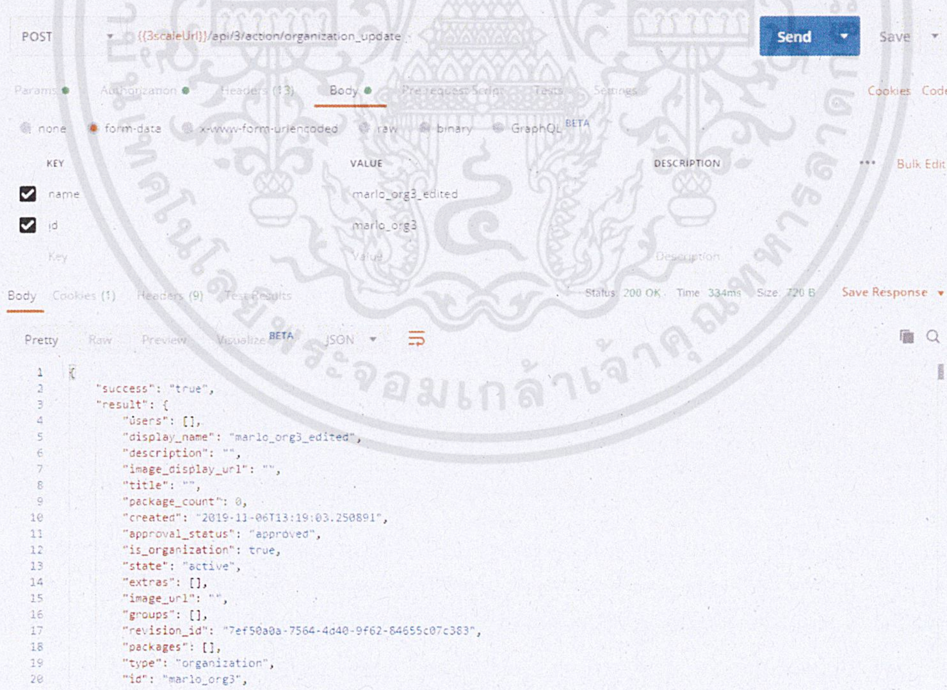
4.2.4.1 จัดการหมดหมู่หลัก

ตารางที่ 4.14 จัดการหมดหมู่หลัก

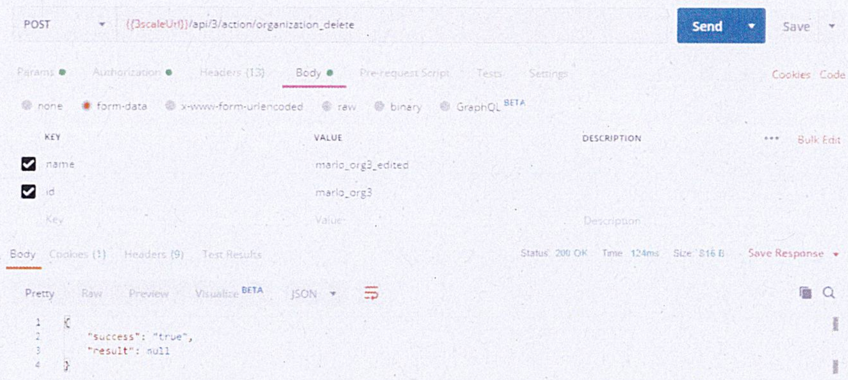
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Data Group and Catalog Management พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ผู้เรียกใช้งาน มีสิทธิ์ในการ Manage Data Group and Catalog Frontend UI เรียกใช้งาน API Manage Data Group and Catalog ตามที่ API กำหนด
Expect result	API Data Group and Catalog Management ทำงานได้ถูกต้อง ทั้ง Add/Update/Delete API Return Response code ได้ถูกต้อง



ภาพที่ 4.46 Test Create Main Data Group



ภาพที่ 4.47 Test Update Main Data Group

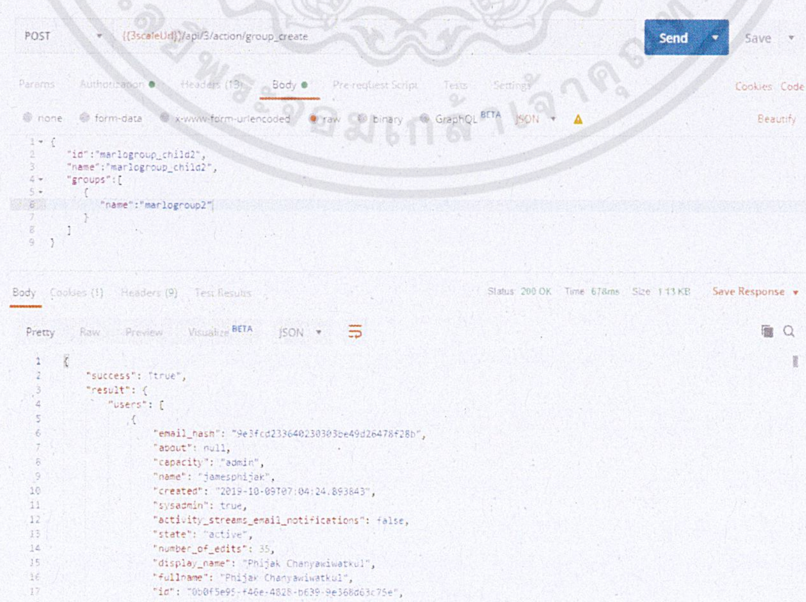


ภาพที่ 4.48 Test Delete Main Data Group

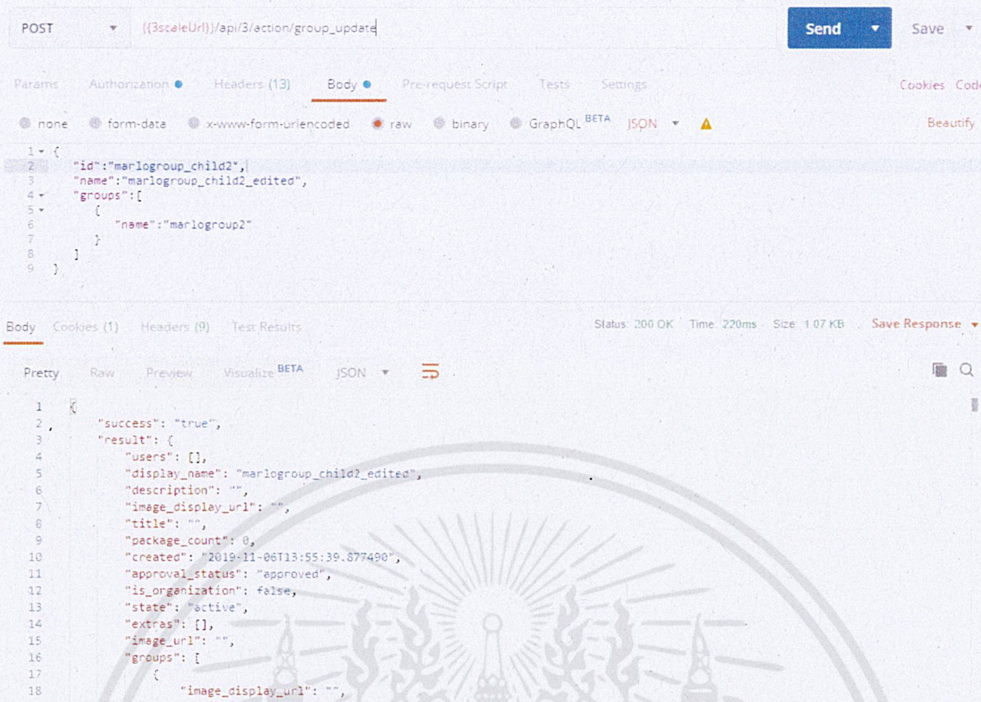
4.2.4.2 จัดการหมวดหมู่ย่อย

ตารางที่ 4.15 จัดการหมวดหมู่ย่อย

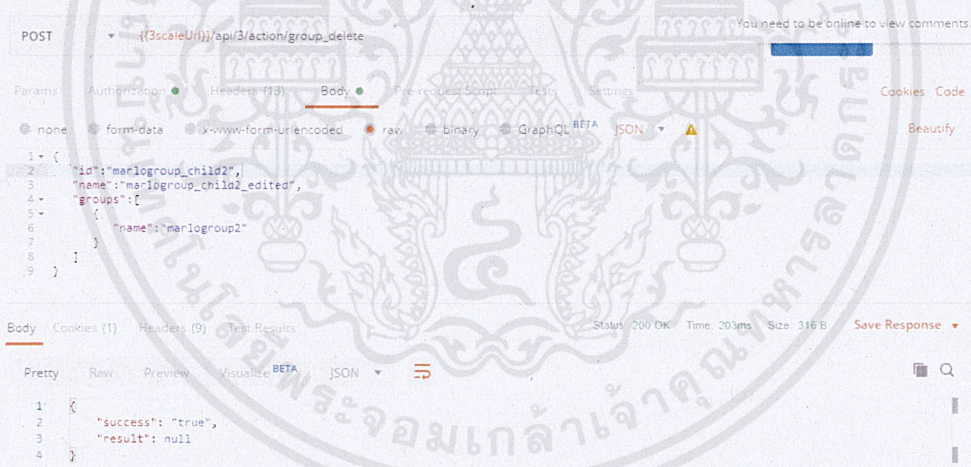
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Data Group and Catalog Management พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานมีสิทธิ์ในการ Manage Data Group and Catalog Frontend UI เรียกใช้งาน API Manage Data Group and Catalog ตามที่ API กำหนด
Expect result	API Data Group and Catalog Management สามารถทำงานได้ถูกต้อง API Return Response code ได้ถูกต้อง



ภาพที่ 4.49 Test Create Sub Data Group

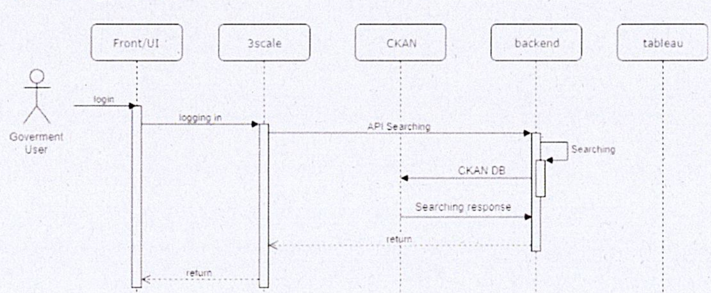


ภาพที่ 4.50 Test Update Sub Data Group



ภาพที่ 4.51 Test Delete Sub Data Group

4.2.5 Searching Scenario

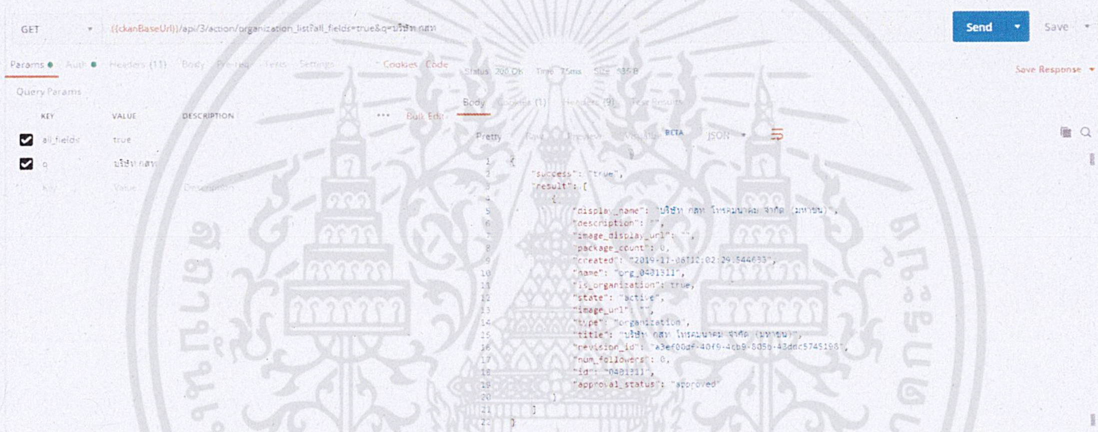


ภาพที่ 4.52 การทำงานของ API Searching

4.2.5.1 ค้นหา Department

ตารางที่ 4.16 ค้นหา Department

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Searching พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ในการ Searching Department Frontend UI เรียกใช้งาน API Searching Department ตามที่ API กำหนด
Expect result	API Searching สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการค้นหาได้ถูกต้อง API Return Response code ได้ถูกต้อง



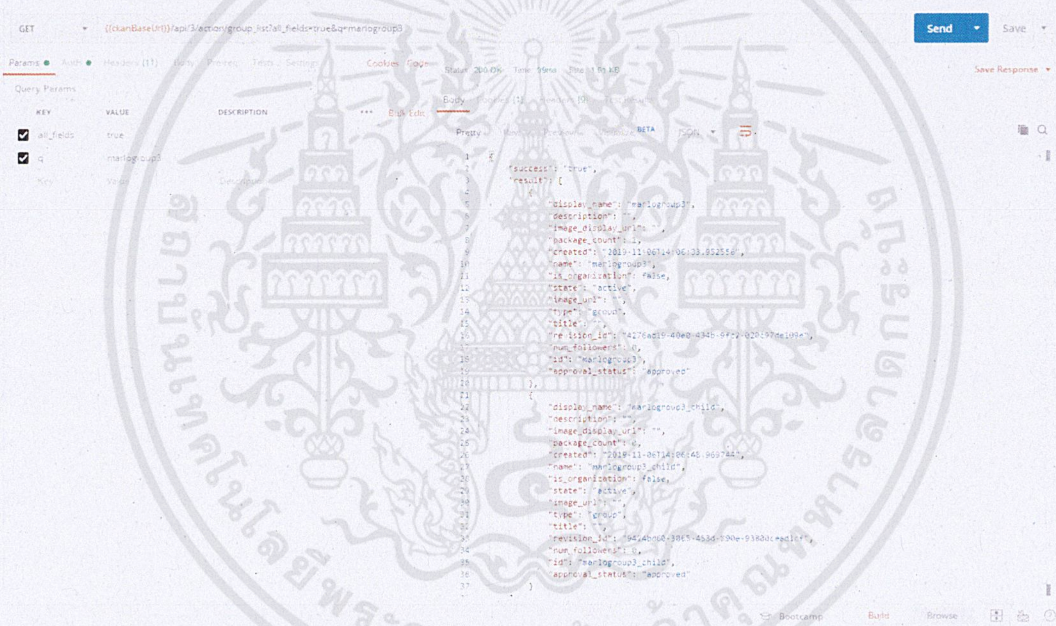
ภาพที่ 4.53 Test Search Organization

ภาพที่ 4.54 Test Search Organization on CKAN Front-end

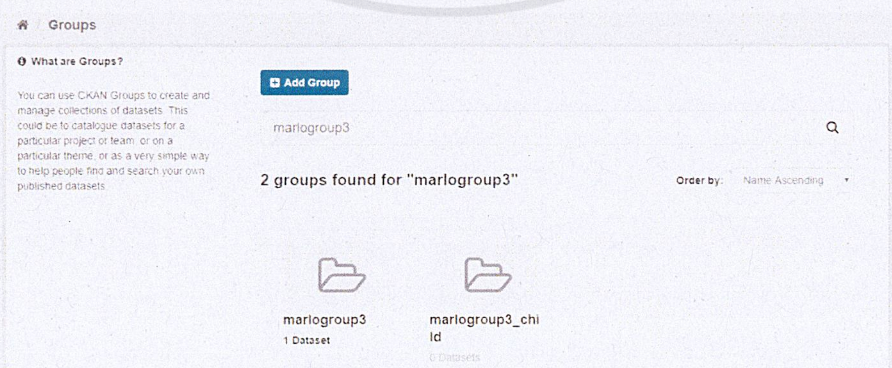
4.2.5.2 ค้นหาหมวดหมู่หรือหมวดหมู่ย่อย

ตารางที่ 4.17 ค้นหาหมวดหมู่หรือหมวดหมู่ย่อย

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Searching พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ในการ Searching Frontend UI เรียกใช้งาน API Searching ตามที่ API กำหนด
Expect result	API Searching สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการค้นหาได้ถูกต้อง API Return Response code ได้ถูกต้อง



ภาพที่ 4.55 Test Search Data Group

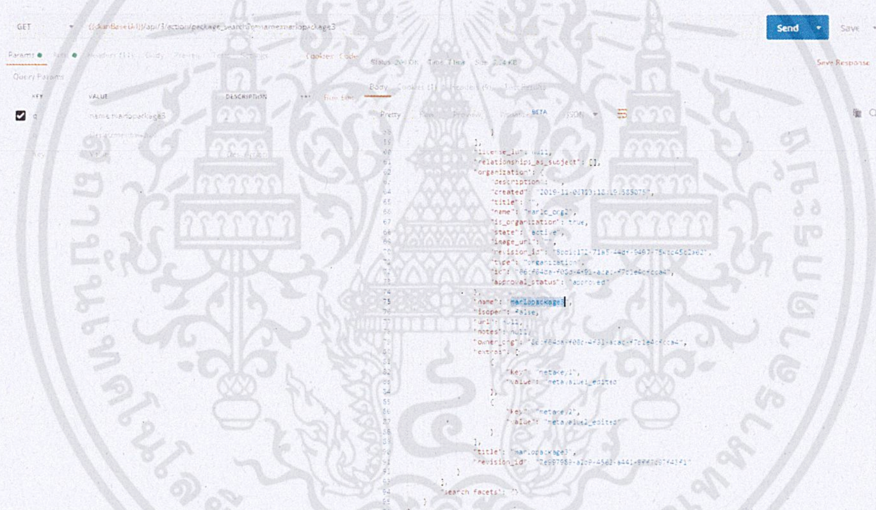


ภาพที่ 4.56 Test Search Data Group on CKAN Front-end

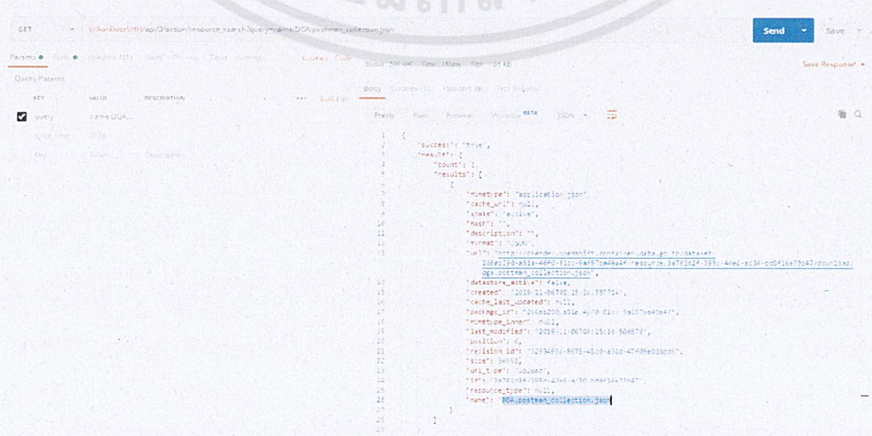
4.2.5.3 ค้นหาข้อมูล ชื่อชุดข้อมูล ชื่อแฟ้มข้อมูล ตามช่วงเวลา

ตารางที่ 4.18 ค้นหาข้อมูล ชื่อชุดข้อมูล ชื่อแฟ้มข้อมูล ตามช่วงเวลา

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Searching พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ในการ Searching Department Frontend UI เรียกใช้งาน API Searching โดยระบุเงื่อนไขการค้นหา ตามที่ API กำหนด
Expect result	API Searching สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการค้นหาได้ถูกต้อง API Return Response code ได้ถูกต้อง



ภาพที่ 4.57 Test ค้นหาชุดข้อมูลจากชื่อของชุดข้อมูล



ภาพที่ 4.58 Test ค้นหาแฟ้มข้อมูลจากชื่อแฟ้มข้อมูล

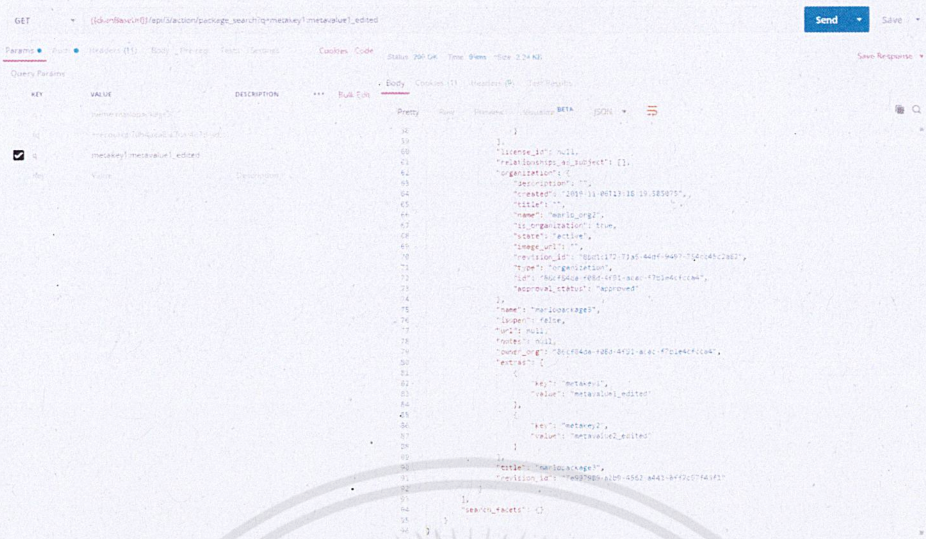
ภาพที่ 4.59 Test ค้นหาชุดข้อมูลตามช่วงเวลา

ภาพที่ 4.60 Test ค้นหาชุดข้อมูลตามช่วงเวลา บน CKAN Front-end

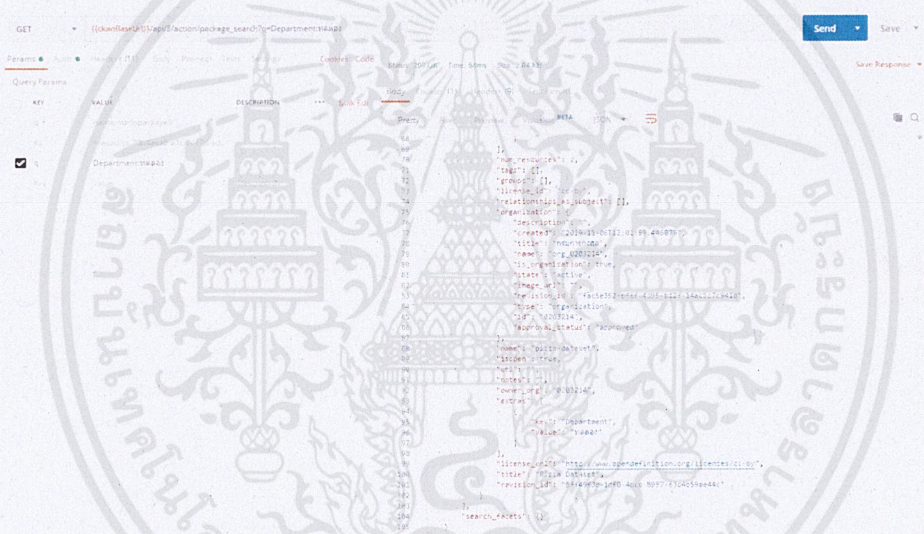
4.2.5.4 ค้นหาข้อมูลจากเนื้อหาข้อมูลภายในคำอธิบายข้อมูล

ตารางที่ 4.19 ค้นหาข้อมูลจากเนื้อหาข้อมูลภายในคำอธิบายข้อมูล

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Searching พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ในการ Searching Department Frontend UI เรียกใช้งาน API Searching โดยระบุการค้นหาด้วยคำอธิบายข้อมูล
Expect result	API Searching สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการค้นหาได้ถูกต้อง API Return Response code ได้ถูกต้อง

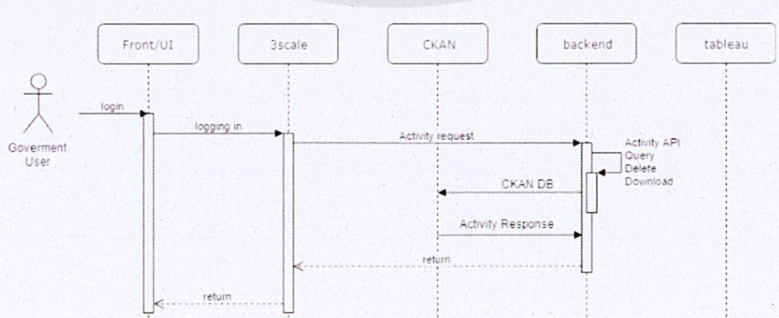


ภาพที่ 4.61 Test ค้นหาชุดข้อมูลจากเนื้อหาข้อมูลภายในคำอธิบายข้อมูล 1



ภาพที่ 4.62 Test ค้นหาชุดข้อมูลจากเนื้อหาข้อมูลภายในคำอธิบายข้อมูล 2

4.2.6 Activity log Scenario

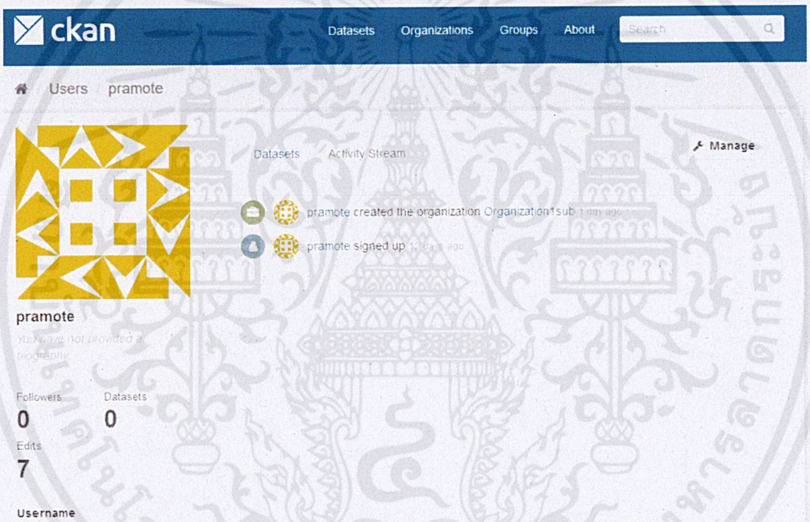


ภาพที่ 4.63 การทำงานของ API Activity

4.2.6.1 แสดงผล Activity Log

ตารางที่ 4.20 แสดงผล Activity Log

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Activity Log พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ใช้งาน Activity API Frontend UI เรียกใช้งาน API Activity Log เพื่อแสดงผล
Expect result	API Activity สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการนำไป display ได้ถูกต้อง API Return Response code ได้ถูกต้อง



ภาพที่ 4.64 Test การแสดงผล Activity ที่เกิดขึ้นบน CKAN Front-end

4.2.6.2 Download Activity Log

ตารางที่ 4.21 Download Activity Log

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ KeyCloak ติดตั้งเสร็จสมบูรณ์ API สำหรับ Activity Log พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ API มีสิทธิ์ใช้งาน Activity API Frontend UI เรียกใช้งาน API Activity Log เพื่อทำการ Download
Expect result	API Activity สามารถทำงานได้ถูกต้อง API return ข้อมูลที่ต้องการ Download ได้ถูกต้อง API Return Response code ได้ถูกต้อง

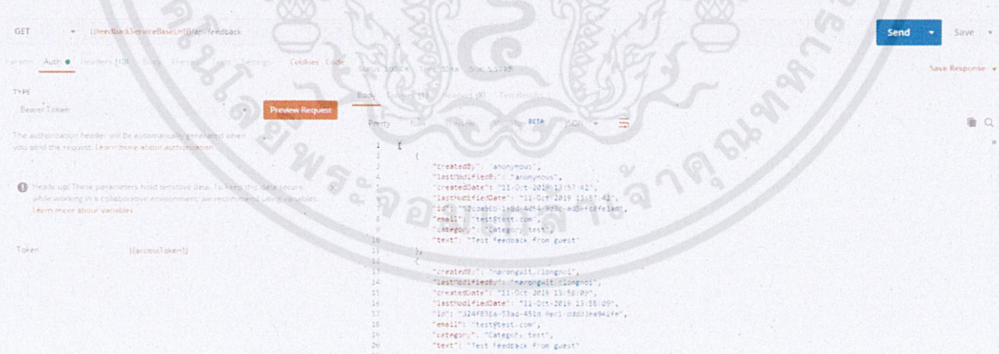
ภาพที่ 4.65 Test การดาวน์โหลด Activity

4.2.7 Feedback Scenario

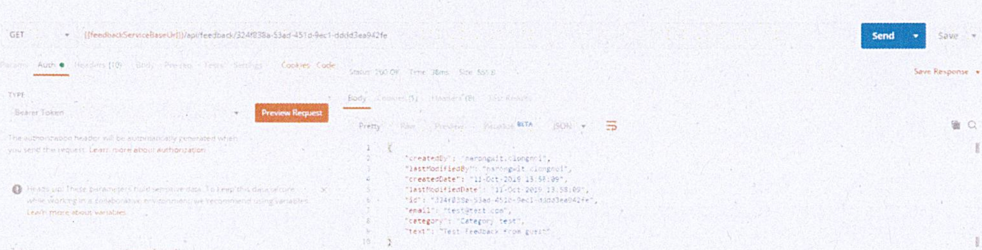
4.2.7.1 View Feedback

ตารางที่ 4.22 New Feedback

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ New feedback พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ View Feedback API Frontend UI เรียกใช้งาน API New feedback
Expect result	Feedback ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามารถเรียกดู Feedback ชุดใหม่ ที่ถูกสร้างขึ้นมาได้



ภาพที่ 4.66 Test View Feedback



ภาพที่ 4.67 Test View Feedback by ID

4.2.7.2 New Feedback

ตารางที่ 4.23 New Feedback

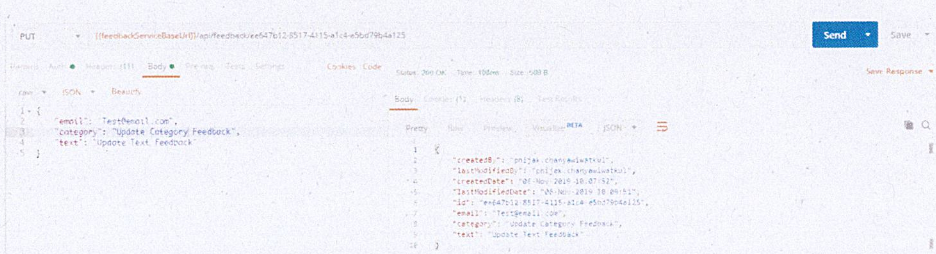
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ New feedback พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ new Feedback API Frontend UI เรียกใช้งาน API New feedback
Expect result	Feedback ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามารถเรียกดู Feedback ชุดใหม่ ที่ถูกสร้างขึ้นมาได้

ภาพที่ 4.68 Test New Feedback

4.2.7.3 Update Feedback

ตารางที่ 4.24 Update Feedback

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Update feedback พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Update Feedback API Frontend UI เรียกใช้งาน API Update feedback
Expect result	Feedback ที่ระบุ ถูกแก้ไขในระบบ API return success code UI Application สามารถเรียกดู feedback ที่ถูก Update ได้



ภาพที่ 4.69 Test Update Feedback

4.2.7.4 Delete Feedback

ตารางที่ 4.25 Delete Feedback

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Delete feedback พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Delete Feedback API Frontend UI เรียกใช้งาน API Delete feedback
Expect result	Feedback ชุดใหม่ ถูกลบออกจากในระบบ API return success code UI Application ไม่สามารถเรียกดู Feedback ที่ถูกลบออกไปได้

ภาพที่ 4.70 Test Delete Feedback

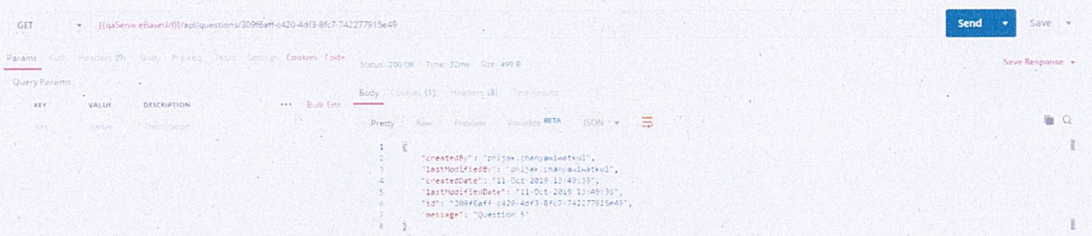
4.2.8 Question and Answer Scenario

4.2.8.1 View Question

ตารางที่ 4.26 View Question

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ View Question พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ View Question API Frontend UI เรียกใช้งาน API View Question
Expect result	Question ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามาเรียกดู Question ชุดใหม่ได้

ภาพที่ 4.71 Test View Question



ภาพที่ 4.72 Test View Question by ID

4.2.8.2 New Question

ตารางที่ 4.27 New Question

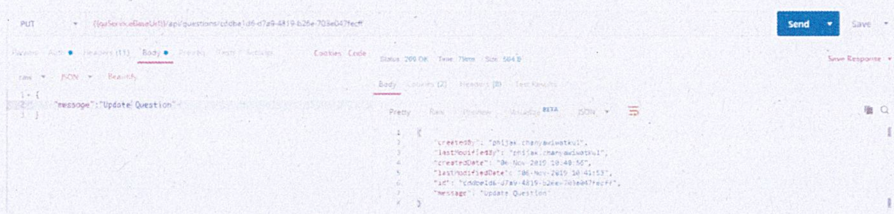
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ New Question พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ new Question API Frontend UI เรียกใช้งาน API New Question
Expect result	Question ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามาเรียกดู Question ชุดใหม่ได้

ภาพที่ 4.73 Test Add Question

4.2.8.3 Update Question

ตารางที่ 4.28 Update Question

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Update question พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Update question API Frontend UI เรียกใช้งาน API update question
Expect result	question ที่ระบุ ถูกแก้ไขในระบบ API return success code UI System สามารถเห็น Question ที่ถูกแก้ไขได้



ภาพที่ 4.74 Test Update Question

4.2.8.4 Delete Question

ตารางที่ 4.29 Delete Question

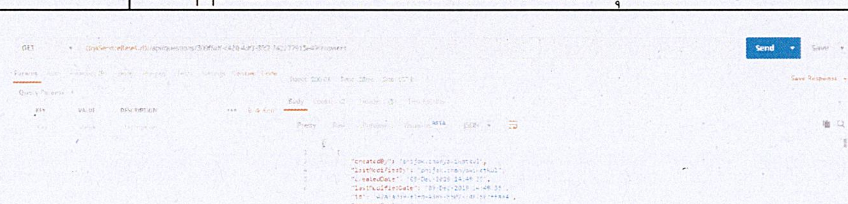
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Delete Question พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Delete Feedback API Frontend UI เรียกใช้งาน API Delete Question
Expect result	Question ถูกลบออกจากในระบบ API return success code UI Application ไม่สามารถเรียกดู Question ที่ถูกลบได้

ภาพที่ 4.75 Test Delete Question

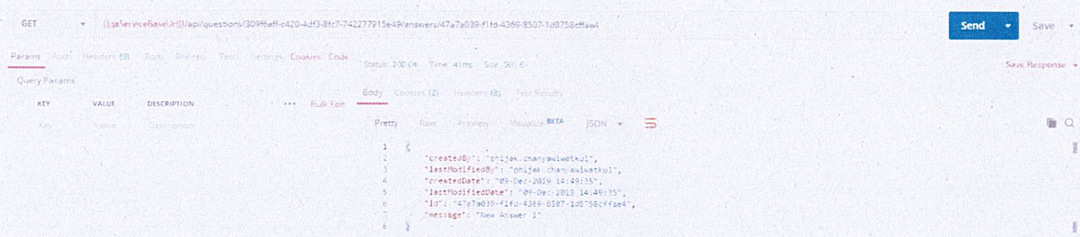
4.2.8.6 View Answer

ตารางที่ 4.30 View Answer

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ View Answer พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ View Answer API Frontend UI เรียกใช้งาน API View Answer ด้วย API Spec
Expect result	Answer ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามารถมองเห็น Answer ชุดใหม่



ภาพที่ 4.76 Test View Answer



ภาพที่ 4.77 Test View Answer by ID

4.2.8.6 New Answer

ตารางที่ 4.31 New Answer

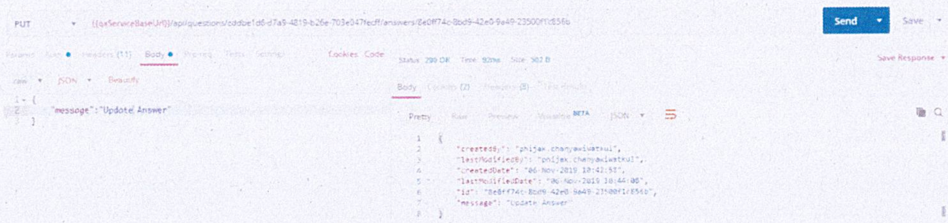
Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ New Answer พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ new Answer API Frontend UI เรียกใช้งาน API New Answer ด้วย API Spec
Expect result	Answer ชุดใหม่ ถูกสร้างในระบบ API return success code UI Application สามารถมองเห็น Answer ชุดใหม่

ภาพที่ 4.78 Test Add Answer

4.2.8.7 Update Answer

ตารางที่ 4.32 Update Answer

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Update Answer พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Update Answer API Frontend UI เรียกใช้งาน API answer
Expect result	answer ที่ระบุ ถูกแก้ไขในระบบ API return success code UI Application สามารถเรียกดู Answer ที่ถูกแก้ไขได้



ภาพที่ 4.79 Test Update Answer

4.2.8.8 Delete Answer

ตารางที่ 4.33 Delete Answer

Pre-requisite	ติดตั้ง 3Scale เสร็จสมบูรณ์ API สำหรับ Delete Answer พัฒนาและติดตั้งสำเร็จ Frontend UI หรือ System อื่น พร้อมใช้งาน
Step	กำหนดสิทธิ์ให้ ผู้เรียกใช้งานสามารถ มีสิทธิ์ในการ Delete Answer API Frontend UI เรียกใช้งาน API Delete Answer ด้วย API Spec ตามรูปแบบที่ API กำหนด
Expect result	Answer ถูกลบออกจากในระบบ API return success code UI ไม่สามารถ เรียกดู Answer ได้

ภาพที่ 4.80 Test Delete Answer

4.3 สรุปผลการทดสอบระบบ

จากผลการทดสอบที่ผ่านมาทำให้สามารถสรุปได้ว่า ระบบการควบคุมและจัดการข้อมูลสามารถทำงานได้ถูกต้องที่ออกแบบ รวมถึงตรงตามความต้องการที่กำหนดไว้

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบการควบคุมและจัดการข้อมูลถูกพัฒนาขึ้นด้วยภาษา Java โดยใช้ Spring Boot Framework พัฒนา API และได้มีการใช้ Core API ของ CKAN ซึ่งเป็นระบบการจัดการข้อมูลที่ได้รับความนิยมในต่างประเทศ มีความเสถียรและปลอดภัยทำให้ลดระยะเวลาการพัฒนาลง และได้ทำงานร่วมกับ Member Service ที่เชื่อมต่อกับ KeyCloak ซึ่งใช้ JWT Token ในการเข้าถึง ซึ่งได้มาเมื่อเข้าสู่ระบบ สำหรับทุก Service จะถูกนำไปทำงานอยู่บน OpenShift Container Platform ในส่วนของ Cluster มีโครงสร้างทั้งหมด 8 Nodes แบ่งเป็น 3 Master Nodes 2 Infra Nodes และ 3 Compute Nodes และตัว OpenShift มีการจัดการและดูแลแต่ละ Service ให้เป็นไปตามที่ตั้งค่าไว้ตลอดเวลา (Self-Healing) ข้อมูลต่างๆที่ทำงานอยู่บน Pod จะถูกนำไปเก็บไว้บน GlusterFS ที่ทำ Replicate แบบ Brick Mode อยู่ 2 เครื่อง ซึ่งก่อนการเข้าถึง API จะผ่านตัว 3Scale ซึ่งใช้จัดการการเข้าถึง API ให้เป็นไปตามข้อกำหนดรวมถึงการ Monitor Traffic ต่างๆ และส่วนสุดท้ายคือการ Monitor Logs ของทุกระบบที่ทำงานอยู่บน OpenShift จะถูก Feed ออกมาด้วย Beats มาเก็บที่ Cluster ของ Elastic Search ทั้ง 3 เครื่องและใช้ Kibana เป็น GUI ในการตรวจสอบการทำงานของ Cluster บน OpenShift รวมถึงจัดทำกราฟ Dashboard เพื่อใช้วิเคราะห์และดูแลระบบให้ทำงานได้ถูกต้องและใช้ตรวจสอบปัญหาเมื่อเกิดข้อผิดพลาดขึ้นในระบบ

5.2 ปัญหาต่างๆที่พบระหว่างการพัฒนาระบบ

ในการพัฒนาระบบการควบคุมและจัดการข้อมูล ผู้พัฒนาต้องทำความเข้าใจพื้นฐานการเขียน API ด้วย Framework Spring Boot ในภาษา Java รวมถึงการ Integrate กับระบบอื่น เช่น CKAN และ Tableau จำเป็นต้องศึกษาและเข้าใจกระบวนการทำงานทั้งหมด และ การนำระบบการควบคุมและจัดการข้อมูลไป Deploy เพื่อทำงานบน OpenShift จำเป็นต้องมีความรู้พื้นฐานของ Kubernetes และทำความเข้าใจเกี่ยวกับการทำ Deployment, Service, Route เพื่อใช้ในการ Deploy Application รวมถึงการจัดการเกี่ยวกับ Services แต่ละตัวให้สามารถเชื่อมต่อและทำงานได้ตามที่ออกแบบไว้ และได้มีการศึกษาการทำงานของการใช้เครื่องมือในกระบวนการของ DevOps ทำให้สามารถนำพื้นฐานดังกล่าวมาใช้ร่วมกับการพัฒนาในครั้งนี้ได้ รวมถึงต้องศึกษาการทำงานและการใช้งานของ GlusterFS เพื่อใช้เป็นที่จัดเก็บข้อมูลสำหรับ Pod แต่ละตัวที่ทำงานอยู่บน OpenShift เนื่องจากข้อมูลของฐานข้อมูลมีความสำคัญมากจึงต้องทำให้แน่ใจว่าการทำงานนั้นถูกต้องสมบูรณ์

5.3 แนวทางการพัฒนาต่อ

ระบบการควบคุมและจัดการข้อมูลถูกพัฒนาขึ้นมาเป็น Back-end API ที่ควบคุมการเข้าถึงฐานข้อมูลและการเชื่อมโยงระบบต่างๆ ซึ่ง API นี้สามารถที่จะนำไปเชื่อมต่อกับแอพพลิเคชั่นที่เป็น Mobile Application หรือ Front-end ต่างๆได้

บรรณานุกรม

การเงินธนาคาร. 2562. เทคโนโลยีการเงิน. (ออนไลน์ แหล่งที่มา : https://www.moneyandbanking.co.th/new/25545/63/ธนาคารท (สืบค้นเมื่อ วันที่ 24 ตุลาคม 2562

จตุพร ชูช่วย . 2562. เรียนรู้เทคโนโลยี OAuth2. (ออนไลน์ แหล่งที่มา : https://sysadmin.psu.ac.th/2019/03/03/what-is-oauth2/ (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

ธันวา จินดารัตนะ. 2561. Kubernetes volume. (ออนไลน์ แหล่งที่มา : https://medium.com/@thanwa/kubernetes-ทำความเข้าใจเรื่อง-กันสักหน่อย-ea6538b50663 (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

ธันวา จินดารัตนะ. 2561. Services. (ออนไลน์ แหล่งที่มา : https://medium.com/@thanwa/kubernetes-ว่าด้วยเรื่อง-services-แต่ละประเภท-25bade6d4725 (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

สกุล มนทา. 2560. OpenShift แบบภาพกว้าง . (ออนไลน์ แหล่งที่มา : https://medium.com/@iamgique/มารู้จัก-openshift-แบบ-overview-กันเถอะ-99cefc444589 (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

สกุล มนทา. 2562. REST กับ RESTful API. (ออนไลน์ แหล่งที่มา : https://medium.com/@iamgique/restful-api-กับ-rest-api-ต่างกันนะรู้ยัง-2c70c42990e3 (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

Acoshift. 2561. JWT. (ออนไลน์ แหล่งที่มา : https://medium.com/acoshift/what-is-jwt-9bad02a30d95 (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

Bennned LD. 2560. เทคโนโลยี Kubernetes. (ออนไลน์ แหล่งที่มา : https://medium.com/sirisoft/getting-start-with-kubernetes-5b4e6b89241e (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

Nut P. 2560. Introduction และ Concept. (ออนไลน์ แหล่งที่มา : https://medium.com/jitta-engineering/kubernetes-s01e01-introduction-concept-6dbd898e7f2d (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562

Red Hat. 2562. กรณีศึกษา: API กับการทำ Open Banking และการนำ API ไปต่อยอดธุรกิจในอุตสาหกรรมอื่นทั่วโลก. (ออนไลน์ แหล่งที่มา : https://www.techtalkthai.com/api-business-case-study-by-red-hat-3scale/ (สืบค้นเมื่อวันที่ 24 ตุลาคม 2562