**เว็บแอปพลิเคชันสำหรับการจัดการทรัพยากรและค่าใช้จ่ายใน AWS**

**(WEB APPLICATION FOR RESOURCE AND COST MONITORING IN AWS)**

**โดย**

**นายชนายุส หนูเอียด**

**CHANAYUS NUIAD**

**นายภูชิต รุ่งระวิ**

**POOHCID RUGRAWI**

**อาจารย์ที่ปรึกษา**

**ผู้ช่วยศาสตราจารย์ ดร.ลภัส ประดิษฐ์ทัศนีย์**

**ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต**

**สาขาวิชาเทคโนโลยีสารสนเทศ**

**คณะเทคโนโลยีสารสนเทศ**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ภาคเรียนที่ 1 ปีการศึกษา 2564**

**WEB APPLICATION FOR RESOURCE AND COST MONITORING IN AWS**

**CHANAYUS NUIAD**

**POOHCID RUGRAWI**

**A PROJECT SUBMITTED IN PARTIAL FULFILLMENT**

**OF THE REQUIREMENT FOR THE DEGREE OF**

**BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY**

**FACULTY OF INFORMATION TECHNOLOGY**

**KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG**

**1/2021**

**COPYRIGHT 2021**

**FACULTY OF INFORMATION TECHNOLOGY**

**KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG**

**ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2564**

**คณะเทคโนโลยีสารสนเทศ**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**เรื่อง เว็บแอปพลิเคชันสำหรับการจัดการทรัพยากรและค่าใช้จ่ายใน AWS**

**WEB APPLICATION FOR RESOURCE AND COST MONITORING IN AWS**

**ผู้จัดทำ**

**1. นายชนายุส หนูเอียด รหัสนักศึกษา 61070034**

**2. นายภูชิต รุ่งระวิ รหัสนักศึกษา 61070166**

**.......................................อาจารย์ที่ปรึกษา**

**(ผู้ช่วยศาสตราจารย์ ดร.ลภัส ประดิษฐ์ทัศนีย์)**

**ใบรับรองโครงงาน (PROJECT)**

**เรื่อง**

**เว็บแอปพลิเคชันสำหรับการจัดการทรัพยากรและค่าใช้จ่ายใน AWS**

**WEB APPLICATION FOR RESOURCE AND COST MONITORING IN AWS**

**นายชนายุส หนูเอียด รหัสนักศึกษา 61070034**

**นายภูชิต รุ่งระวิ รหัสนักศึกษา 61070166**

**ขอรับรองว่ารายงานฉบับนี้ ข้าพเจ้าไม่ได้คัดลอกมาจากที่ใด**

**รายงานฉบับนี้ได้รับการตรวจสอบและอนุมัติให้เป็นส่วนหนึ่งของ**

**การศึกษาวิชาโครงงาน หลักสูตรวิทยาศาสตรบัณฑิต (เทคโนโลยีสารสนเทศ)**

**ภาคเรียนที่ 1 ปีการศึกษา 2564**

**………………………………..**

**(นายชนายุส หนูเอียด)**

**………………………………..**

**(นายภูชิต รุ่งระวิ)**

**หัวข้อโครงงาน**       เว็บแอปพลิเคชันสำหรับการจัดการทรัพยากรและค่าใช้จ่ายใน AWS

**นักศึกษา** นายชนายุส หนูเอียด รหัสนักศึกษา 61070034

      นายภูชิต รุ่งระวิ รหัสนักศึกษา 61070166

**ปริญญา**       วิทยาศาสตร์บัณฑิต

**สาขาวิชา** เทคโนโลยีสารสนเทศ

**ปีการศึกษา** 2562

**อาจารย์ที่ปรึกษา** ผู้ช่วยศาสตราจารย์ ดร.ลภัส ประดิษฐ์ทัศนีย์

**บทคัดย่อ**

การประมวลผลระบบคลาวด์เป็นเทคโนโลยีที่มีความสำคัญต่อการติดต่อสื่อสารผ่านอินเทอร์เน็ตในการกระจายข้อมูลไปยังส่วนต่าง ๆ ของโลกโดยที่มีการจัดการที่ง่ายและขยายตัวตามความต้องการได้รวดเร็ว ซึ่งจะมีผู้ให้บริการติดตั้งศูนย์กลางข้อมูลกระจายไปทั่วโลก การเรียนรู้โครงสร้างพื้นฐานของระบบสารสนเทศในรูปแบบคลาวด์เป็นการเรียนรู้ที่ง่าย ประหยัด และสามารถไปต่อยอดไปใช้งานในธุรกิจจริงได้ง่าย

Amazon Web Services เป็นหนึ่งในผู้ให้บริการคลาวด์และเป็นแพลตฟอร์มคลาวด์ที่นำมาใช้ในวิชาการพัฒนาคลาวด์แอปพลิเคชันระดับองค์กรและวิชาเทคโนโลยีกลุ่มเมฆเบื้องต้นของคณะเทคโนโลยีสารสนเทศ สจล. โดยมี ผศ. ดร.ลภัส ประดิษฐ์ทัศนีย์ เป็นอาจารย์ประจำวิชาที่ได้พบเจอปัญหาในการสอนวิชาคลาวด์ในส่วนที่เป็นข้อจำกัดของ AWS Management Console ที่ให้การใช้งาน IAM User ให้กับกลุ่มโปรเจคนักศึกษาไปใช้งาน โดยปัญหาหลัก ๆ ที่พบเจอคือการเข้าถึงและจัดการทรัพยากรหรือการบริการต่างๆที่นักศึกษาสร้างไว้ทั้งหมดค่อนข้างใช้เวลานานและทรัพยากรที่สร้างขึ้นไม่มีการระบุว่าเป็นของ IAM User คนไหน จึงเกิดเป็นโปรเจคเว็บแอปพลิเคชันสำหรับการจัดการทรัพยากรที่จะมีการวางระบบเบื้องหลังในการติดตามการกระทำของ IAM User มีการระบุเจ้าของทรัพยากรที่สร้างขึ้น การลบทรัพยากรที่ง่าย และระบบเสริมที่มาชดเชยในส่วนที่ AWS Management Console ไม่มีที่จะช่วยอาจารย์ประจำวิชาและนักศึกษาในการลดปัญหาในการทำงานได้ดียิ่งขึ้น

**Project Title**      Web Application for Resource and Cost Monitoring in AWS

**Students** Mr. Chanayus Nuiad Student ID 61070034

      Mr. Poohcid Rugravi Student ID 61070166

**Degree**       Bachelor of Science

**Program**  Information Technology

**Academic Year** 2021

**Advisor** Assoc. Prof. Dr. Lapat Praditthasanee

**ABSTRACT**

Cloud computing is a critical technology for internet communication and information distribution around the world. Cloud computing allows easy management and rapid expansion on demand, which has service providers located around the world. The infrastructure of cloud-based information systems is simple to learn, economical, and easily extended to use in business.

Amazon Web Services (AWS) is a cloud provider and cloud platform used in enterprise cloud application development, which is an introductory cloud technology course offered by the Faculty of Information Technology at KMITL. by Asst. Prof. Dr. Lapat Praditthasanee is an instructor who has struggled to teach cloud subjects due to the limitations of the AWS Management Console. The main problem encountered is that accessing and managing all student-created resources or services takes a long time and resources to create. It does not specify which IAM user it belongs to. Therefore, we developed a web application for resource management in AWS that includes a background event tracking system for IAM users. The owner of the created resource is specified. Easy resource removal and a complementary system that compensates for parts that the AWS Management Console does not have. to help course instructors and students reduce problems with work.

**กิตติกรรมประกาศ**

ปริญญานิพนธ์ฉบับนี้จะประสบผลสำเร็จได้เป็นอย่างดี ด้วยความกรุณาของอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.ลภัส ประดิษฐ์ทัศนีย์ ที่ให้ข้อมูลอันสำคัญในการวิจัย ทรัพยากร และคำชี้แนะต่าง ๆ อีกทั้ง คอยติดตามความก้าวหน้าของโครงงานอยู่เป็นระยะ ๆ เพื่อให้ปริญญานิพนธ์ฉบับนี้เกิดประสบผลสำเร็จที่จะมาซึ่งในการแก้ไขปัญหาที่มีอยู่และสร้างเป็นผลิตภัณฑ์ที่จะถูกนำไปใช้งานจริงในการเรียนการสอนในโอกาสถัดไป

ขอขอบคุณคณาจารย์และเจ้าหน้าที่ในคณะเทคโนโลยีสารสนเทศ ที่เป็น กรรมการผู้ช่วยตรวจสอบปริญญานิพนธ์ฉบับนี้ให้ประสบผลสำเร็จ ตลอดจนให้ทรัพยากรในการวิจัยและสร้างผลิตภัณฑ์ขึ้นมา ก่อให้เกิดประโยชน์แก่คณะเทคโนโลยีสารสนเทศ อีกทั้งนำไปประยุกต์ใช้งานในการทำงานในอนาคตได้

**สารบัญ**

**หน้า**

บทคัดย่อภาษาไทย……………………………………………………………………………………..I

บทคัดย่อภาษาอังกฤษ………………………………………………………………………………....II

กิตติกรรมประกาศ…………………………………………………...……………………………….III

สารบัญ…………………………………………………………………………………...…………...IV

สารบัญตาราง……………………………………………...…………………………………………..V

สารบัญรูป……………………………………………………………………………………………..VI

บทที่

1. บทนำ……………………………………………………………………..………………....1

1.1 ความเป็นมาของปัญหา……………………………………………………………….1

1.2 วัตถุประสงค์………………………………………………………………………….1

1.3 ขอบเขตของระบบ…………………………………………………………...……….1

1.4 ประโยชน์ที่คาดว่าจะได้รับ…………………………………………………………...2

1.5 ศัพท์นิยามเฉพาะที่ใช้ในโครงงาน……………………………………………………2

2. การทบทวนวรรณกรรม……..……………………………………………………………..3

2.1 การบริการและทรัพยากรใน AWS เพื่อใช้ในการสร้างแท็กแบบอัตโนมัติ...………….3

2.2 เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์………………………………………………...5

3. วิธีการดำเนินการวิจัย…………..…………………………………………………………..7

3.1 การวิเคราะห์และออกแบบระบบ…………………………...…………………………7

3.2 แผนดำเนินงาน……………………...………………………………………………....9

3.3 หลักการทำงานของระบบ…………………………..………………………………..11

4. ผลการทดลองเบื้องต้นหรือระบบต้นแบบ....………..……………………………………28

4.1 เว็บแอปพลิเคชัน…….........................……………………...………………………..28

4.2 รายชื่อ Resources ที่ทำ Auto tag ได้…………………………..............................…...29

5. บทสรุป อภิปรายผล และข้อเสนอแนะ....………..………………………….....………….31

5.1 บทสรุป…….........................……………………...………………………………….31

5.2 อภิปรายผล…….........................……………………...……………………………...31

5.3 ข้อเสนอแนะ…….........................……………………...…………………………….32

บรรณานุกรม……………………………..…………………………………………………………...19

**สารบัญตาราง**

**หน้า**

**ตารางที่**

3.1 ปฏิทินแผนดำเนินงาน……………………………….………………………………….9

**สารบัญภาพ**

**รูปที่ หน้า**

2.1 ตัวอย่างรูปแบบข้อมูลของ Event Record 3

2.2 สัญลักษณ์ AWS CloudTrail 3

2.3 สัญลักษณ์ AWS CloudWatch3

2.4 สัญลักษณ์ AWS Lambda4

2.5 สัญลักษณ์ IAM4

2.6 สัญลักษณ์ Boto 35

2.7 สัญลักษณ์ NEXT.JS5

2.8 สัญลักษณ์ MongoDB6

3.1 Use Case Diagram8

3.2 Auto-Tagging Solution Workflow 11

3.3 โค้ด Boto 3 (1)12

3.4 โค้ด Boto 3 (2)12

3.5 อธิบายการเปิดใช้งาน AWS CloudTrail 12

3.6 อธิบายการเพิ่ม Event Rule ใน AWS CloudWatch 13

3.7 อธิบายการเพิ่ม Event ที่ต้องการจะดักผ่าน JSON 13

3.8 อธิบายการ ใส่ Targets Trigger Function ใน CloudWatch14

3.9 ฟังก์ชันโค้ด AWS SDK (boto3)14

3.10 การสร้าง API Gateway เพื่อเรียกใช้งานฟังก์ชัน (1)15

3.11 การสร้าง API Gateway เพื่อเรียกใช้งานฟังก์ชัน (2)15

3.12 หน้า Log in16

3.13 หน้า Dashboard ของอาจารย์ผู้สอน16

3.14 หน้าแสดงผล Resource ที่กำลังใช้งานในแต่ละ Region 17

3.15 หน้าแสดง IAM User ทั้งหมด17

3.16 หน้ารายละเอียดของ Resource 18

3.17 หน้าแสดงผล IAM User ที่ใช้งาน Resource 23

4.1 หน้าหลักที่มีการแสดงผลข้อมูล Resources จริง 27

4.2 หน้าแสดง Resources ที่มีการแสดงผลข้อมูล Resources จริง27

4.3 หน้าหลักที่แสดงผลแบบ Light Mode27

4.4 หน้าหลักที่แสดงผลแบบ Dark Mode27

4.5 หน้าแสดง Resources ที่เลือกการแสดงผลแบบหมวดหมู่29

**บทที่ 1**

**บทนำ**

**1.1 ความเป็นมาของปัญหา**

การที่มีนักศึกษาในการเรียนวิชาที่เกี่ยวข้องกับระบบ Cloud Computing เป็นจำนวนมากจึงส่งผลให้อาจารย์ประจำวิชาเข้าถึงและจัดการ Resource หรือ Service ต่างๆที่นักศึกษาสร้างไว้ทั้งหมดค่อนข้างใช้เวลานานและยากลำบาก ทางกลุ่มของเราจึงต้องการทำโครงงานเกี่ยวกับเว็บแอปพลิเคชันสำหรับจัดการ Resource และ Service ต่างๆ ใน AWS เพื่อให้สามารถเข้าถึงและติดตามได้สะดวกมากยิ่งขึ้น

**1.2 วัตถุประสงค์**

1. **เ**พื่อเพิ่มความรวดเร็วและความสะดวกสะบายให้กับอาจารย์ ในจัดการ Resource ต่างๆ ใน AWS ที่นักศึกษาสร้างขึ้นในวิชาเรียนของคณะเทคโนโลยีสารสนเทศสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2. ลดข้อผิดพลาดในการบริหารจัดการระบบ ยกต้ออย่างเช่น กรณีที่นักศึกษาลืมใส่ Tag ให้กับ Resource เพื่อมั่นใจได้ว่า Resource ที่ถูกสร้างขึ้นทุกที่เป็นของกลุ่มไหนและสามารถลบได้ตามกรอบ Policy ที่กำหนด
3. เพิ่มฟังก์ชันในส่วนที่ AWS Management Console ไม่มี สำหรับอาจารย์ผู้รับผิดชอบวิชา เช่น ฟังก์ชันการดูค่าใช้จ่าย ฟังก์ชันในการติดตั้งและตั้งค่า Resources ที่นอกเหนือในการเรียนในวิชา Cloud

**1.3 ขอบเขตของระบบ**

โปรเจคจัดทำขึ้นเพื่อใช้สำหรับวิชา Cloud-Based Enterprise Application Development หรือวิชาที่เรียนทางด้าน Cloud Computing โดยใช้บริการของ AWS เท่านั้น เพื่อแก้ไขปัญหาของอาจารย์ประจำวิชาและนักศึกษาที่เรียนในวิชานี้

**1.4 ประโยชน์ที่คาดว่าจะได้รับ**

1. อาจารย์ประจำวิชามีความสะดวกสบายในการจัดการ Resources
2. นักศึกษาที่เรียนในวิชา Cloud วางแผนการคำนวณค่าใช้จ่ายได้ง่ายขึ้น
3. อาจารย์ประจำวิชาที่เป็นเจ้าของ Account ไม่ต้องเสียค่าใช้จ่ายส่วนเกินที่ได้รับมา
4. นักศึกษาสามารถทดลองการใช้ Resource ได้เต็มที่เพื่อให้โปรเจคสำเร็จลุร่วง
5. คณะผู้จัดทำได้ความรู้และประสบการณ์ในการทำ Infrastructure as Code ของระบบ AWS

**1.5  ศัพท์นิยามเฉพาะที่ใช้ในโครงงาน**

1. SDK ( Software Development Kit ) คือ ชุดเครื่องมือที่ใช้สำหรับการพัฒนาซอฟต์แวร์ที่ถูกสร้างโดยผู้พัฒนา เพื่อให้ผู้ใช้ซอฟต์แวร์สามารถนำเครื่องมือนี้ไปสร้างหรือพัฒนางานของตนเองได้
2. AWS Resource คือ ทรัพยากรต่างๆที่มีให้บริการใน Amazon Web Service เช่น AWS Lambda, AWS CloudTrail และ AWS S3 เป็นต้น
3. IAM User คือ ผู้ใช้ในส่วนของนักศึกษาที่ใช้งาน AWS Management Console ของ Account

อาจารย์ผู้สอน

**บทที่ 2**

**การทบทวนวรรณกรรม**

**2.1 การบริการและทรัพยากรใน AWS เพื่อใช้ในการสร้างแท็กแบบอัตโนมัติ**

**2.1.1 รูปแบบข้อมูลต่าง ๆ ที่จำเป็นต้องรู้ใน AWS สำหรับการสร้าง Tag**

**2.1.1.1 ResourceARN**

ARN (Amazon Resource Name) เป็นชื่อของ Resource ต่างๆที่ถูกสร้างใน AWS โดย Format ของ ARN ประกอบด้วยดังนี้

arn:partition:service:region:account-id:resource-id

arn:partition:service:region:account-id:resource-type/resource-id

arn:partition:service:region:account-id:resource-type:resource-id

ดังนั้นการที่จะสามารถทำการ Tag Resource ได้นั้นจะต้องทราบ Format ARN ของ Resource ที่ต้องการ Tag

**2.1.1.2 Event record**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติเป็นข้อมูลรายละเอียดของ Event ต่างๆที่ผู้ใช้กระทำต่อระบบของ AWS เช่น การสร้าง หรือลบ Resource โดยรูปแบบของข้อมูล Event record เป็นประเภท JSON

**รูปที่ 2.1** ตัวอย่างรูปแบบข้อมูลของ Event record

**2.1.2 AWS CloudTrail**

**รูปที่ 2.2** สัญลักษณ์ AWS CloudTrail

เป็น Service ที่จะเป็น Log ทุกการกระทำของ IAM user ทั้งหมด โดยการเก็บ Log จะขึ้นอยู่กับ Policy ที่ตั้งไว้ เมื่อ User ใช้งานตาม Policy ที่กำหนดจะมีการเก็บ Log โดย CloudTrail ลง S3 Bucket

**2.1.3 AWS EventBridge**



**รูปที่ 2.3** สัญลักษณ์ AWS EventBridge

เป็น Service ที่จะให้ลงทะเบียนเหตุการณ์ที่เกิดขึ้นต่าง ๆ ใน AWS ได้ เพื่อต้องการทำบางสิ่งบางอย่างได้จากเหตุการณ์ เช่น สั่งใช้งาน Lambda Function หรือ แอปพลิเคชัน SaaS เมื่อเกิดเหตุการณ์บางอย่างขึ้น ซึ่งสามารถใช้ข้อมูล Event ที่ส่งมาได้ด้วย

**2.1.4 AWS Lambda**

**รูปที่ 2.4** สัญลักษณ์ AWS Lambda

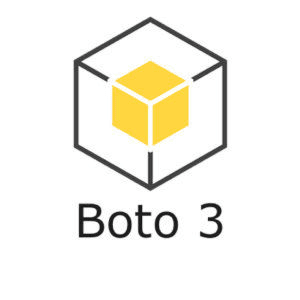
เป็น Service ที่ทำให้ผู้ใช้สามารถรันโค้ดโดยที่ไม่จำเป็นต้องมี Server เป็นของตัวเอง

ซึ่งส่งผลให้ผู้ใช้ไม่ต้องทำการตั้งค่าต่างๆเกี่ยวกับ Server ผู้ใช้เพียงแค่นำโค้ดขึ้นไปที่ AWS Lambda โค้ดจะสามารถทำงานได้ทันที

**2.1.5 IAM**

**รูปที่ 2.5** สัญลักษณ์ AWS IAM

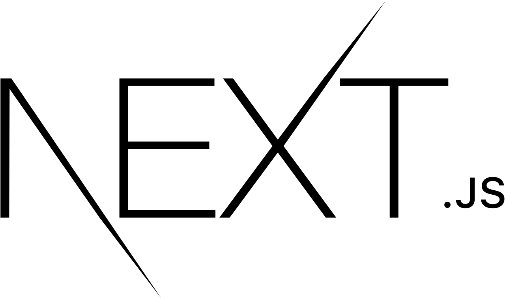
เป็น Service ที่ช่วยจัดการสิทธิ์การเข้าถึงทรัพยากรและบริการของ AWS อย่างปลอดภัย สามารถสร้างและจัดการผู้ใช้และกลุ่ม AWS ได้โดยใช้ IAM และใช้สิทธิ์เพื่ออนุมัติหรือปฏิเสธไม่ให้ผู้ใช้เข้าถึงทรัพยากร AWS ได้

**2.1.6 Boto 3**

**รูปที่ 2.6** สัญลักษณ์ Boto 3

Boto 3 เป็น AWS SDK ที่ช่วยให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ให้ทำงานร่วมกับ AWS ผ่านการเขียนโปรแกรมภาษา Python

**2.2 เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์**

 **2.2.1 เครื่องมือสำหรับการพัฒนาเว็บแอปพลิเคชัน**

**รูปที่ 2.7** สัญลักษณ์ NEXT.JS

**2.2.1.1 Next JS**

Next Js เป็น React Framework พัฒนาโดย Vercel ที่ใช้สำหรับการพัฒนาเว็บแอปพลิเคชันทั้งในฝั่ง Front-end โดยจุดเด่นหลักๆของ Next JS คือ ทาง Vercel ได้นำเครื่องมือต่างๆที่จำเป็นมาไว้ใน Next.JS ให้แล้วเช่น การทำ Routing, การทำ API และมีการ Optimize ทำให้มี Performance ที่ดียิ่งขึ้น สิ่งสำคัญคือหน้าเว็บจะได้รับการประมวลผลจากฝั่งเซิร์ฟเวอร์ซึ่งจะเอื้ออำนวยในการทำ SEO เป็นข้อดีในการค้นหาหน้าเว็บจาก Google ได้ไวยิ่งขึ้น

รูปภาพประกอบด้วย ข้อความ, สัญลักษณ์

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 2.8** สัญลักษณ์ MongoDB

**2.2.1.2 MongoDB**

MongoDB เป็น Database โดยมีพื้นฐานเป็น NoSQL จึงไม่มีความสัมพันธ์ของข้อมูลเหมือนกับ Database แบบ SQL ซึ่ง MongoDB จะบันทึกข้อมูลลงในไฟล์รูปแบบ JSON ซึ่งต่างจาก Database SQL ที่เก็บข้อมูลในรูปแบบตาราง

**บทที่ 3**

**วิธีการดำเนินการวิจัย**

**3.1 การวิเคราะห์และออกแบบระบบ**

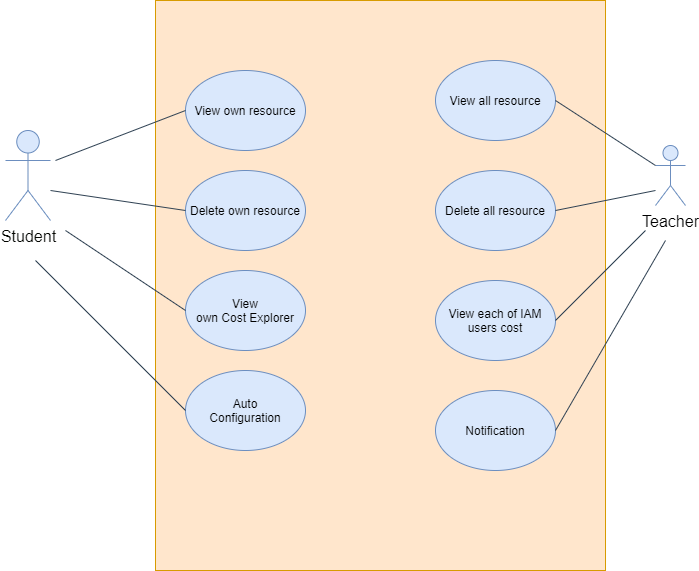
**3.1.1 วิเคราะห์ปัญหา**

* ทรัพยากรที่ถูกสร้างขึ้นจะไม่มีการระบุว่าเป็นของผู้ใช้งานคนไหน
* การดูแลทรัพยากรทุกอย่างใน AWS เป็นเรื่องที่ยุ่งยาก
* การจะลบทรัพยากรต้องใช้ระยะเวลานานถึงจะลบหมด
* IAM User ไม่มีหน้าการคำนวนค่าใช้จ่ายบน AWS Management Console

**3.1.2 ออกแบบความต้องการของระบบ**

* วางระบบสร้างแท็กอัตโนมัติเมื่อมีการสร้างทรัพยากรนั้น ๆ บน AWS ที่จะมีการระบุชื่อของกลุ่มที่สร้าง
* สร้างเว็บแอปพลิเคชันที่มีฟังก์ชันในการดูข้อมูลและลบทรัพยากรให้สะดวกมากยิ่งขึ้น และมีการคำนวนค่าใช้จ่ายของ IAM User นักศึกษา
* อาจารย์สามารถดูทรัพยากรของทุกกลุ่มที่สร้างขึ้นได้โดยจะแสดงผลให้เข้าใจได้ง่ายและสามารถสั่งลบทรัพยากรที่สร้างขึ้นจาก IAM User นักศึกษาได้ทั้งหมดได้ในขั้นตอนเดียว
* เว็บแอปพลิเคชันมีระบบสนับสนุนเพิ่มเติมจาก AWS Management Console เช่น การแจ้งเตือนการสร้าง VPC เกินจำนวน และ Internet Gateway เกินจำนวน เป็นต้น

**3.1.3 Use Case Diagram**

****

**รูปที่ 3.1** Use Case Diagram

**3.2 แผนดำเนินงาน**

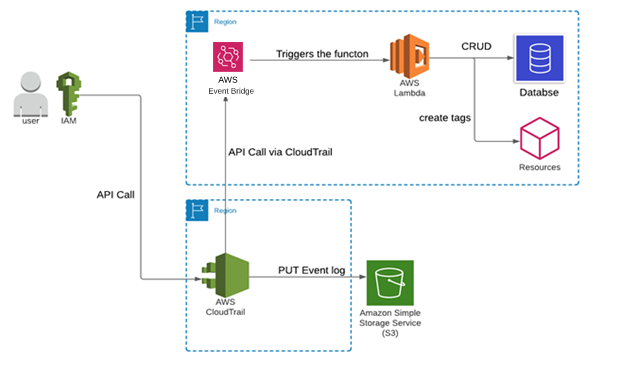
**ตารางที่ 3.1 ปฏิทินแผนดำเนินงาน**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ขั้นตอนการดำเนินงาน | ระยะเวลาการดำเนินงาน | | | | | | | | | | | | | | | |
| สิงหาคม 2564 | | | | กันยายน 2564 | | | | ตุลาคม 2564 | | | | พฤศจิกายน 2564 | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1.ศึกษาและทดลองทำ auto-tagging สำหรับ AWS Resource |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. ศึกษาและทดลองการทำ AWS SDK (ใส่ส่วนของการเรียกดูข้อมูล s3 ec2 และลบ instance) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. ออกแบบ UI หรือ prototype ของเว็บ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. สร้าง API สำหรับการดึงข้อมูล Resource และลบ Resource |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. เริ่มพัฒนาเว็บแอป |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6. เริ่มทำระบบแจ้ง เตือน เมื่อมีการสร้าง Resource ที่เกินความจำเป็น |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7. เริ่มศึกษาเรื่องการคำนวนค่าใช้จ่ายใน กรณีที่แยก Resource ตาม IAM User |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**หมายถึง เริ่มต้นทำในเวลาหนึ่งจนสิ้นสุดในอีกเวลาหนึ่ง**

**หมายถึง เริ่มต้นทำในเวลาหนึ่งจนถึงวันสิ้นสุดโครงการ**

**3.3 แนวทางการออกแบบระบบ Auto Tag**



**รูปที่ 3.2** Auto-Tagging Solution Workflow

เมื่อ IAM User ทำการสร้าง Resource หรือ ใช้งาน Service ต่างๆ ระบบจะทำการเรียกใช้งาน AWS CloudTrail เพื่อทำการตรวจสอบ Event ที่ IAM User ได้กระทำต่อระบบของ AWS  และเก็บข้อมูล Event นั้นลงใน AWS S3 และจากนั้น AWS CloudTrail จะทำการเรียกใช้งาน AWS EventBridge สำหรับ Event ที่ได้ทำการลงทะเบียนไว้ใน EventBridge rules จากนั้น EventBridge จะเรียกใช้งาน Lambda Function Auto Tagging ที่เขียน Script ในการสั่งการสร้าง Tags ให้กับ Resources โดยอ้างอิงข้อมูลจาก Event record ที่ส่งมา

**3.4 ลำดับการวางระบบ Auto Tag และหลักการเบื้องต้น**

**3.4.1 สร้าง Lambda Function**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.3** อธิบายการสร้าง Lambda Function Auto Tagging

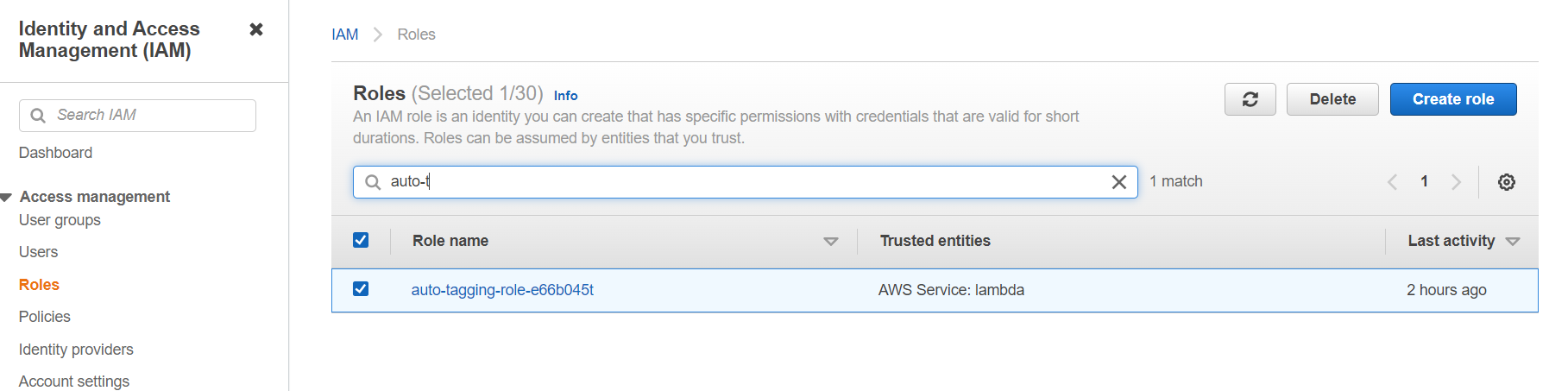
รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.4** อธิบายการสร้าง Lambda Function Auto Tagging ส่วนสิทธิ์การใช้งาน

ทำการสร้าง Lambda Function โดยในที่นี้เลือกภาษาโปรแกรมมิ่งเป็น Python 3.9 จากนั้นตรงตั้งค่าสิทธิ์การใช้งานให้เลือกสร้าง Role ขึ้นมาใหม่ โดย Role ในที่นี้คือ User ของ Lambda Function ที่ลักษณะการกำหนด Policies เหมือนกับ IAM User จากนั้นกดปุ่ม Create Function

**3.4.2 กำหนด Policies and permissions ให้กับ Role User**



**รูปที่ 3.5** การค้นหา Auto Tagging Role ในหน้า IAM User

เปิดหน้า IAM User ในหมวดหมู่ Role User ค้นหาชื่อ Role ตามชื่อที่ตั้งใน Lambda Function เมื่อพบแล้วให้กดที่ชื่อ Role ในตางรางตรงคอลัม Role-name เพื่อเข้าสู่หน้าตั้งค่า Policies

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.6** ตั้งค่า Policies ให้กับ Role User

กด Attach policies แล้วเพิ่ม Policies ที่จำเป็นต่อการใช้งาน AWS SDK ในส่วนของการสร้าง Tag โดยในตัวอย่างรูปที่ 3.6 ได้กำหนด policy สำหรับการสร้าง Tag ทรัพยากรประเภท EC2

**รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ3.4.3 เปิดใช้งาน AWS CloudTrail**

**รูปที่ 3.7** เปิดใช้งาน AWS CloudTrail

**3.4.4 เพิ่มโค้ด Boto3 ลงใน AWS Lambda Function**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติรูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.9** โค้ด Boto 3

**รูปที่ 3.8** โค้ด Boto 3

เขียน AWS SDK โดยในตัวอย่างใช้ Boto3 ในการเขียน ภายใน Code ได้มีการนำข้อมูล Event record ของ CloudTrail มาหาข้อมูลสำคัญก็คือ resource id, User id, ชื่อ User และเวลาที่สร้างเพื่อระบุเจาะจงไปที่ Resource ที่ต้องการจะเพิ่ม Tags ได้

ส่วนสำคัญและความยุ่งยากในการที่จะ Tags ให้กับ Resource คือจะต้องอ่านคู่มือของ boto3 ในส่วนวิธีการเพิ่ม Tags ของ Resource นั้น ๆ และหาข้อมูลสำคัญใน Event record ซึ่งแต่ละ Event จะให้ข้อมูลแตกต่างกัน ส่งผลในแต่ละ Event มีวิธีการดึง resource id ไม่เหมือนกันและให้ในบางครั้งจำเป็นต้องดัดแปลงข้อมูลก่อนเข้า Function Create Tag ตามคู่มือของ Boto3

**3.4.5 เพิ่ม Event Rule ใน AWS** **EventBridge**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.10** อธิบายการเพิ่ม Event Rule ใน AWS EventBridge

เลือก Service ที่จะดัก Event เช่น EC2 และเลือก Event Type เป็น AWS API Call via CloudTrail หมายความว่าดัก Event เมื่อ User กระทำต่อ Service ผ่านทาง SDK หรือ AWS Management Console เช่น การสร้าง แก้ไข และ ลบ

ส่วนที่เป็น eventName จะเป็นการอ้างอิงถึงชื่อ Event ใน CloudTrail พร้อมกับต้องเป็น Event ที่เป็นประเภทตรงกันกับ Event record ในตัวอย่างได้ทำการลงทะเบียน Event ในส่วนของการสร้าง EC2 Instance โดยข้อมูลไฟล์ JSON จะต้องใส่ Event source ที่ชื่อ aws.ec2, ec2.amazonaws.com ตามในรูปที่ 3.6 และใส่ Event ที่ชื่อว่า RunInstance

**3.4.6 ใส่ Targets trigger function ใน AWS EventBridge**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.11** อธิบายการใส่ Targets Trigger Function ใน EventBridge

เลือกประเภท Target เป็น Lambda Function แล้วเลือกชื่อฟังก์ชันที่สร้างในขั้นตอนที่ 3.4.1

**3.5 วิธีการเขียนโค้ด Boto3 สร้าง Tag ลงบนทรัพยากรในแต่ละกรณี**

**3.5.1 กำหนด Method การสร้าง Tag ของ Resource โดยตรง**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.12** ตัวอย่างโค้ดสร้าง Tag จาก Method ของ Resource โดยตรง

ในทุก ๆ Resource object ของ boto3 จะมี Method สร้าง Tag เสมอ โดยจะต้องใส่ข้อมูลและรูปแบบตรงตามคู่มือของ Boto3 จากตัวอย่างได้ใช้ Resource object EC2 แล้วทำการสร้าง Tag จากข้อมูลที่มีอยู่และประเภทข้อมูลตรงตามตัวแปรรับข้อมูล

**3.5.2 กำหนดการสร้าง Tag ของ ResourceGroupsTaggingAPI**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.13** ฟังก์ชัน tag\_by\_grouptaggig

Resource Groups Tagging API เป็น Service ที่รับและส่งข้อมูล Resources ที่จะจัดการเกี่ยวกับ Tag โดยตรง มี Method สร้าง Tag ที่ชื่อ tag\_resources โดยเป็นตัวเลือกอีกทางหนึ่งที่สามารถใส่ Tag ให้ Resources ได้ แต่มีบาง Resources ไม่สามารถใช้ Method นี้สร้าง Tag ได้ มีข้อมูลที่จำเป็นต้องรับได้แก่

1. Tags ที่เป็นข้อมูลประเภท dictionary โดยมี Key เป็น Key ที่จะตั้งใน Tag และ Value เป็น Value ที่จะตั้งใน Tag
2. ResourceARNList จะรับ ResourceARN ได้ทุกชนิดและรับได้หลายค่า

**3.5.3 กรณีที่ Event record ส่ง Identity เป็น ResourceARN**

**3.5.3.1 ใช้ ResourceGroupsTaggingAPI สร้าง Tag**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.14** โค้ดการสร้าง Tag ของ Lambda Funtion

นำ ResourceARN มาใช้ในฟังก์ชัน tag\_by\_grouptaggig ตามในรูปที่ 3.13 และทดสอบแล้วว่าสามารถสร้าง Tag ได้อย่างสมบูรณ์

**3.5.3.1 กรณีที่ไม่สามารถใช้ ResourceGroupsTaggingAPI สร้าง Tag ได้**

รูปภาพประกอบด้วย ข้อความ

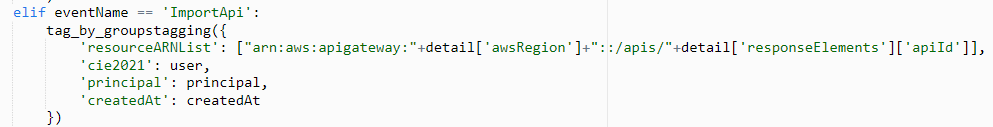
คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.14** โค้ดการสร้าง Tag ของ EFS

Method สร้าง Tag ของ EFS รับ Identity เป็นประเภท Id ฉะนั้นจำเป็นต้องแปลงข้อมูล ResourceARN เป็น Id ของ EFS เสียก่อนจึงจะนำเข้าสู่ฟังก์ชันได้

**3.5.4 กรณีที่ Event record ส่ง Identity เป็น Id และไม่สามารถใช้ค่า Id ได้**

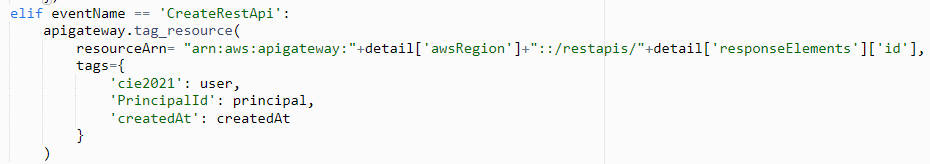
**3.5.4.1 ใช้ ResourceGroupsTaggingAPI สร้าง Tag**



**รูปที่ 3.15** โค้ดการสร้าง Tag ของ API Gateway

แปลงค่า Id ของ API Gateway เป็น ResourceARN ตามรูปแบบ ResourceARN ที่ถูกต้อง จะสามารถสร้าง Tag ด้วยฟังก์ชัน tag\_by\_grouptagging ได้

**3.5.4.2 กรณีที่ Method การสร้าง Tag ของ Resource รับ ResourceARN**



**รูปที่ 3.16** โค้ดการสร้าง Tag ของ REST API Gateway

แปลงค่า Id ของ REST API Gateway เป็น ResourceARN ตามรูปแบบ ResourceARN ที่ถูกต้อง จึงจะใช้ Method การสร้าง Tag ของ API Gateway ได้

**3.5.5 การสร้าง Tag ของ AutoScalingGroup**

**3.5.5.1 โค้ดสร้าง Tag เมื่อสร้าง AutoScalingGroup**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.17** โค้ดการสร้าง Tag ของ AutoScalingGroup

Medthod การสร้าง Tag ของ AutoScalingGroup จะรับค่า Key Value ตามปกติพร้อมจะต้องระบุ ResourceId ของ AutoScalingGroup ด้วยและสร้าง Tag ได้หลาย Tag พร้อมกัน

**3.5.5.2 การสร้าง Tag ให้กับ EC2 Instance ที่สร้างจาก AutoScalingGroup**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.19** Event record RunInstances ในส่วน User Identity

ในกรณีนี้ Event record RunInstances มีการะบุข้อมูล User เป็นประเภท Role ซึ่งเป็น Role User ของ AutoScalingGroup ที่ไม่ใช่ IAM User เจ้าของ Resources

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.20** Event record RunInstances ในส่วน Tag ตั้งต้น

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.18** โค้ดการสร้าง Tag ของ EC2 Instance ที่สร้างจาก AutoScalingGroup

วิธีการที่จะนำข้อมูล IAM User เจ้าของ Resources สามารถทำได้ โดยใน Event record จะมีการส่งชื่อ AutoScalingGroup มาใน Tag ตั้งต้นของ items ใน InstancesSet จากนั้นนำชื่อ AutoScalingGroup มาดึงข้อมูลรายละเอียด ทำให้ได้ข้อมูล IAM User เจ้าของ Resources จากข้อมูล Tag ของ AutoScalingGroup

**3.6 การดึงข้อมูลทรัพยากร**

**3.6.1 ฟังก์ชันโค้ด AWS SDK (Boto3)**

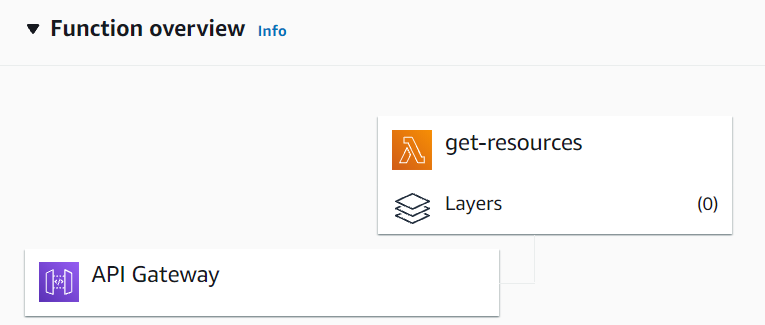
รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.19** ฟังก์ชันโค้ด AWS SDK (boto3)

ทำการสร้าง Lambda Function และให้สิทธิ์ Role User ใช้งาน ResourceGroupsTaggingAPI โดยใช้ Policy ResourceGroupsandTagEditorFullAccess

ResourceGroupsTaggingAPI มี Method ดึงข้อมูล Resources ที่มีการระบุ Tag และไม่ได้ระบุ Tag ในบางตัว โดยจะส่งข้อมูลมาเป็น ResourceARN และ Tags ซึ่งเพียงพอต่อการแสดงผลในหน้าเว็บที่จะต้องระบุชื่อ Resource และ IAM User เจ้าของ Resource

**3.6.2 สร้าง API Gateway เพื่อเรียกใช้งานฟังก์ชัน**

**รูปที่ 3.20** การสร้าง API Gateway เพื่อเรียกใช้งานฟังก์ชัน (1)

**รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ**

**รูปที่ 3.21** การสร้าง API Gateway เพื่อเรียกใช้งานฟังก์ชัน (2)

เปิดใช้งาน API Gateway ประเภท REST API แล้วเชื่อมต่อเข้ากับ Lambda Function get-resources เพื่อนำไปแสดงผลในหน้าเว็บ

**3.7 การพัฒนาและการติดตั้งเว็บแอปพลิเคชัน**

**3.7.1 Framework และเทคโนโลยีในการทำเว็บแอปพลิเคชัน**

* + Node.js เป็น platform ในการ Runtime โค้ดภาษา JavaScript โดยเลือกพัฒนาเป็นโปรแกรม Web Server
  + Next.js เป็นชุดพัฒนาอำนวยความสะดวกในการทำ Web Server ใน Node.js โดยนำไปพัฒนาเป็นเว็บแอปพลิเคชัน
  + React เป็นเทคโนโลยีที่อำนวยความสะดวกและเพิ่มประสิทธิภาพในการจัดการหน้าเว็บเพจที่ต้องจัดการองค์ประกอบมากมาย
  + Styled-Components เป็นตัวจัดการในการทำ Style ใน React ที่จะช่วยในเรื่องความเป็นระเบียบและความยืดหยุ่นในการพัฒนา Style

**3.7.2 การติดตั้งเว็บแอปพลิเคชันในในสภาพแวดล้อมทดสอบ**

**3.7.2.1 ทำ version control ประเภท git และนำขึ้น GitHub**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ในอาคาร

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.22** หน้า GitHub AWS-Portal repository

**3.7.2.2 ใช้บริการ Heroku และทำ Continuous Deployment กับ GitHub repository**

รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.23** การตั้งค่าการติดตั้งกับ GitHub Repository ของ Heroku Web Server

**3.7.2.3 ทดสอบการติดตั้งจากการอัพเดทโค้ดใน GitHub**

รูปภาพประกอบด้วย ข้อความ

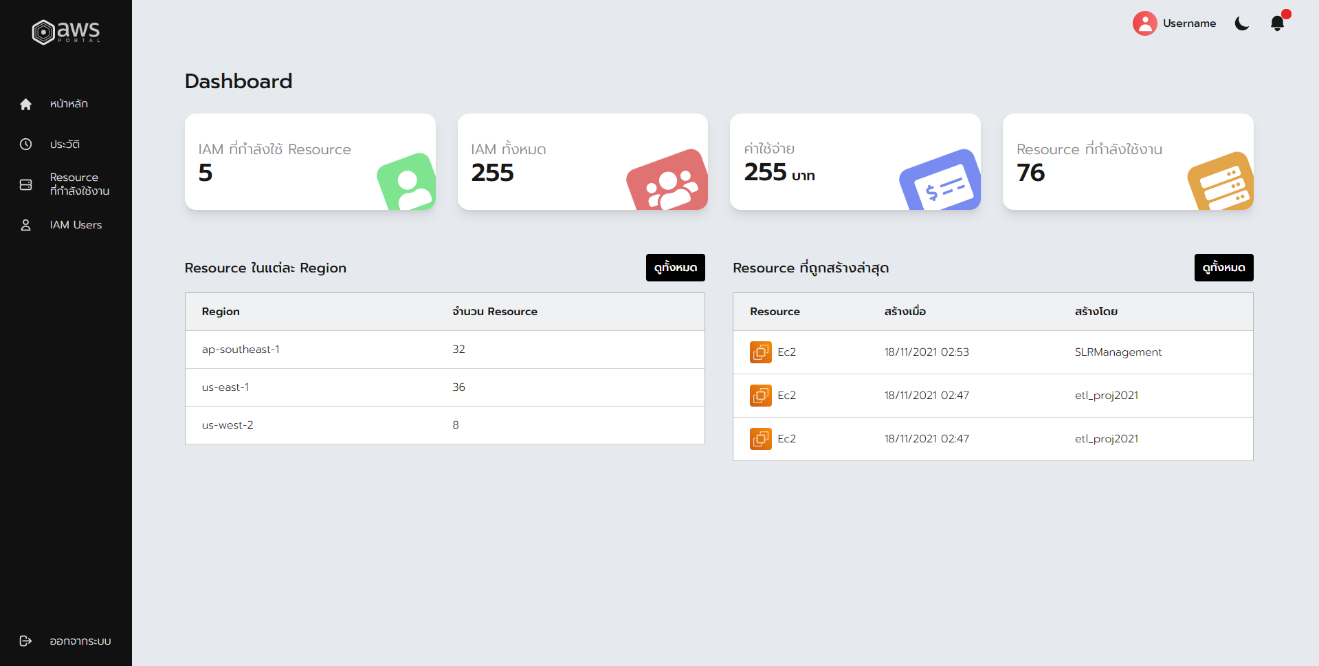
คำอธิบายที่สร้างโดยอัตโนมัติ

**รูปที่ 3.24** Log ของการรันคำสั่งติดตั้ง Web Server ของ node.js

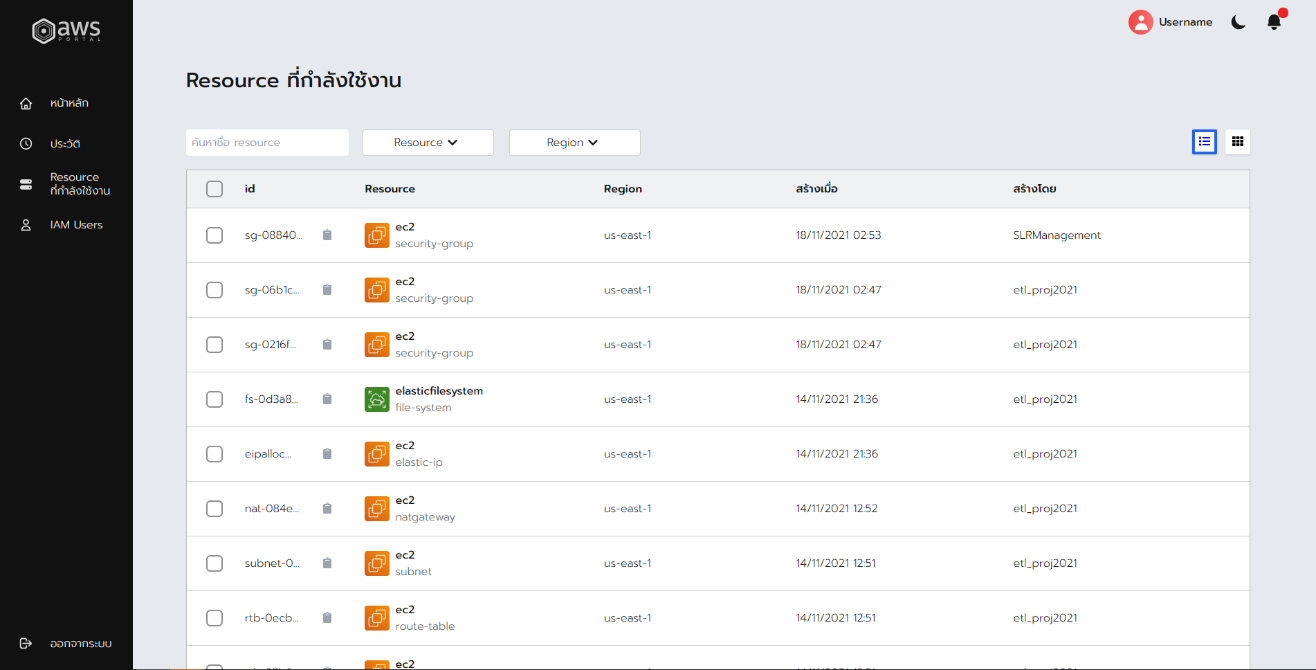
**บทที่ 4**

**ผลการทดลองเบื้องต้นหรือระบบต้นแบบ**

**4.1 เว็บแอพลิเคชัน**

**4.1.1 เว็บแอพลิเคชันสามารถแสดงข้อมูล Resources จริงได้**

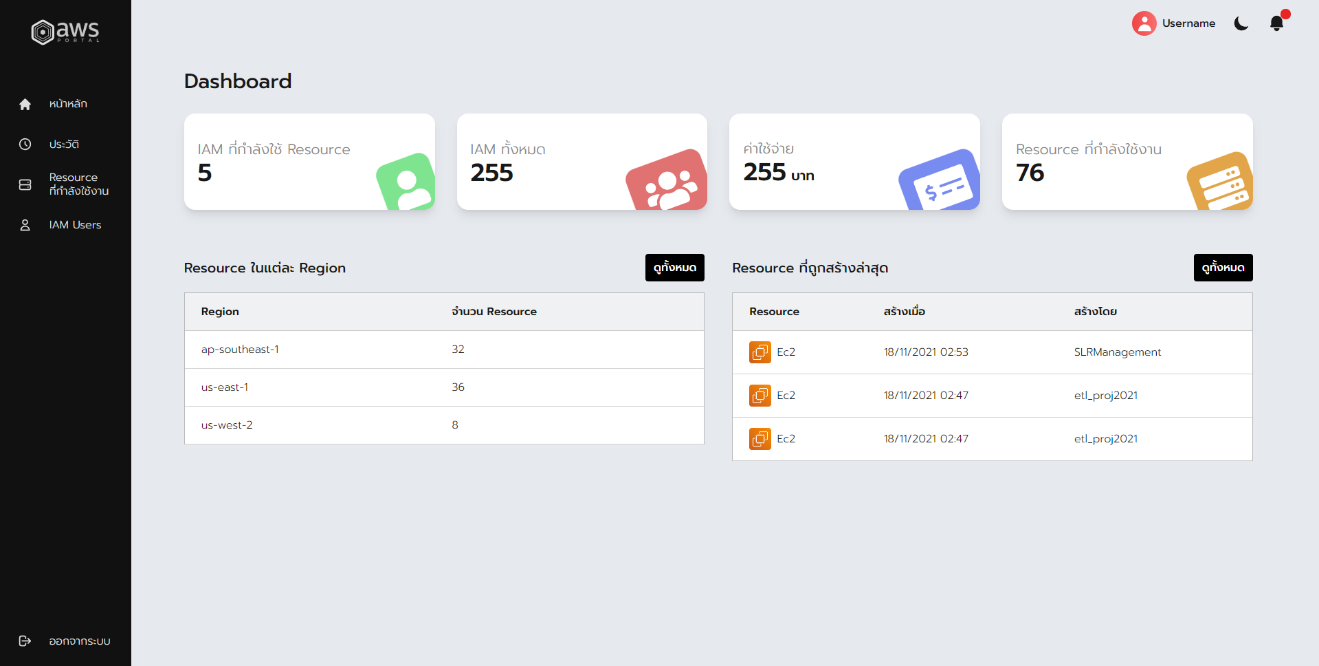
**รูปที่ 4.1 หน้าหลักที่มีการแสดงผลข้อมูล Resources จริง**

****

**รูปที่ 4.2 หน้าแสดง Resources ที่มีการแสดงผลข้อมูล Resources จริง**

เว็บแอพลิเคชันสามารถดูข้อมูล Resource ต่างๆที่ถูกสร้างใน AWS Account

ได้และมีการแสดงข้อมูลผู้สร้าง Resource และเวลาที่ Resource ถูกสร้าง

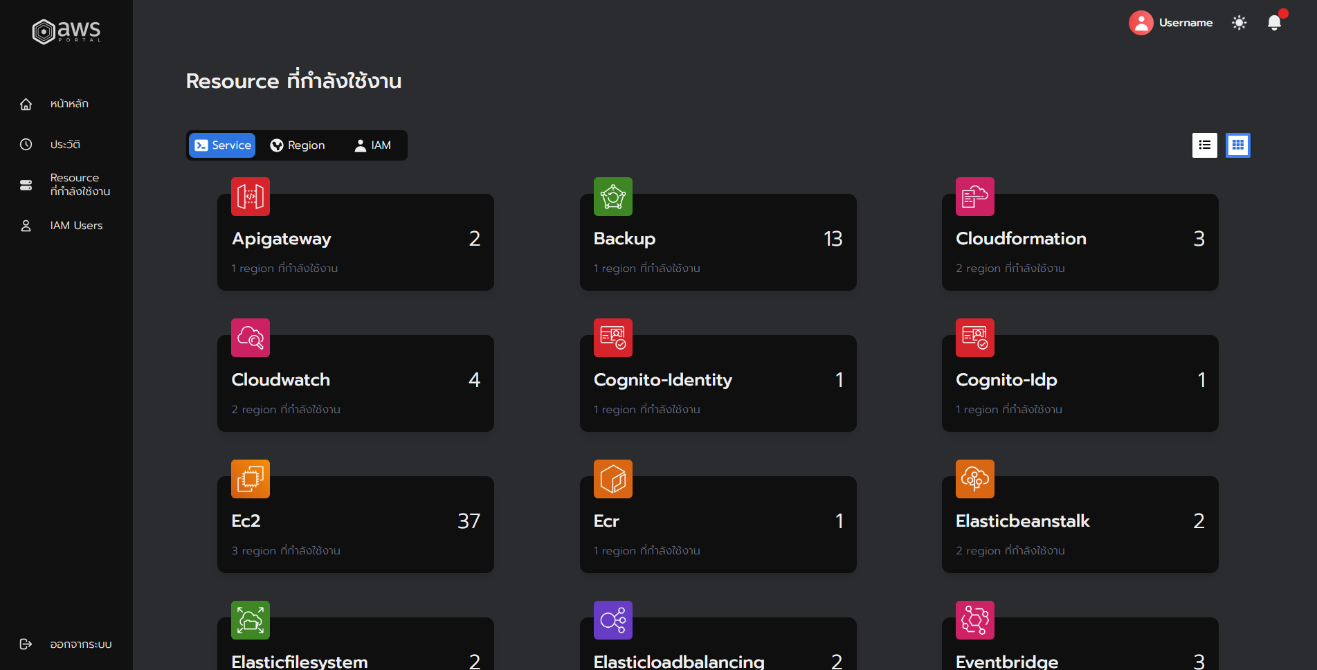
**4.1.2 เว็บแอพลิเคชันรองรับแสดงผลแบบ Dark Mode และ Light Mode**

**รูปที่ 4.3 หน้าหลักที่แสดงผลแบบ Light Mode**

**รูปที่ 4.4 หน้าหลักที่แสดงผลแบบ Dark Mode**

เว็บแอพลิเคชันมีฟังก์ชันรองรับการแสดงผลแบบ Dark Mode (โหมดมืด) และ

Light Mode (โหมดสว่าง) ซึ่งฟังก์ชันนี้ช่วยในการถนอมสายตาในกรณีที่เปิดใช้ Dark Mode

**4.1.3 มีการจัดหมวดหมู่ของ Resources**

**รูปที่ 4.5 หน้าแสดง Resources ที่เลือกการแสดงผลแบบหมวดหมู่**

มีการเพิ่มรูปแบบการแสดงผลแบบหมวดหมู่ของ Resource ว่า Resource ต่างๆที่ได้ทำการ Tag นั้นประกอบไปด้วยประเภทอะไรบ้าง, อยู่ใน Region ใด และ IAM User ใดเป็นผู้สร้าง

**4.2 รายชื่อ Resources ที่ทำ Auto tag ได้**

1. VPC
2. Instance
3. Volume
4. Subnet
5. Security Group
6. Route Table
7. Internet Gateway
8. Network ACL
9. AMI
10. Elastic IP
11. Nat Gateway
12. Lambda
13. Target Group
14. Load Balancer
15. EFS
16. Keypair
17. Launch Configuration
18. Auto Scaling Group
19. DynamoDB Table
20. API Gateway
21. S3 Bucket
22. User Pool
23. Identity Pool
24. Route53

**บทที่ 5**

**บทสรุป อภิปรายผล และข้อเสนอแนะ**

ระบบส่งเสริมใน AWS และเว็บแอปพลิเคชันสำหรับการดูแลจัดการทรัพยากรใน AWS ของวิชา Cloud Computing มีวัตถุประสงค์ 1) เพื่อเพิ่มความรวดเร็วในการติดตามทรัพยากรต่าง ๆ ที่นักศึกษาในวิชาสร้างขึ้น 2) เพื่อเพิ่มความรวดเร็วในการถอนการติดตั้งทรัพยากรให้รวดเร็วยิ่งขึ้น 3) เพื่อเพิ่มระบบสร้างข้อมูลสำคัญในการระบุเจ้าของทรัพยากรอัตโนมัติเมื่อมีการสร้างทรัพยากรเกิดขึ้น 4) เพื่อสร้างระบบการคำนวณการแสดงผลค่าใช้จ่ายให้กับนักศึกษาที่ใช้งาน AWS 5) เพื่อเป็นการเรียนรู้การใช้งาน AWS ของผู้จัดทำโครงงาน 6) เพื่อปรับปรุงการเรียนการสอนวิชา Cloud Computing ที่ใช้ระบบของ AWS ในการศึกษา

ที่มาข้อมูลปัญหาที่ใช้สำหรับในการทำโครงงานครั้งนี้คือ ผศ. ดร.ลภัส ประดิษฐ์ทัศนีย์ เป็นอาจารย์ประจำวิชา Cloud Computing ที่ใช้ระบบ AWS ในการสอน โดยให้ข้อมูลสำคัญที่เป็นวัตถุประสงค์หลักที่นำไปสู่การพัฒนาระบบส่งเสริมการสร้างแท็กอัตโนมัติและเว็บแอปพลิเคชันที่อำนวยความสะดวกในการติดตามทรัพยากรและลบทรัพยากร

**5.1 บทสรุป**

5.1.1 ปัญหาสำคัญที่มาเป็นอันดับแรกคือทรัพยากรที่สร้างขึ้นจะไม่มีการระบุเจ้าของที่สร้าง ทางผู้จัดทำต้องสร้างระบบส่งเสริมขึ้นมาก่อนคือระบบสร้างแท็กอัตโนมัติ โดยเริ่มทำการศึกษาระบบเหตุการณ์ของ AWS ที่เป็นประเภทเหตุการณ์เกี่ยวกับผู้ใช้งานแล้วนำมาผนวกกับการรันโค้ดการทำงานใน AWS ที่ใช้ข้อมูลชุดข้อมูลเหตุการณ์นำมาสร้างแท็กแบบอัตโนมัติได้สำเร็จ โดยเริ่มทำการทดลองทำกับทรัพยากรแต่ละประเภทแล้วทดสอบเสมือนจริงจนสามารถติดตั้งระบบสร้างแท็กอัตโนมัติให้กับทรัพยากรทั้งหมด 24 ประเภท

5.1.2 การพัฒนาของเว็บแอปพลิเคชันสำหรับการติดตามข้อมูลทรัพยากรใน AWS ประกอบไปด้วยฟังก์ชันการทำงานที่ทำสำเร็จไปแล้วได้แก่

1. แสดงผลจำนวนทรัพยากรที่ใช้งานอยู่แบบสรุปในแต่ทวีป โดยจะแสดงผลเฉพาะภูมิภาคเวอร์จิเนียตอนเหนือและสิงคโปร์
2. แบ่งหมวดหมู่การแสดงผลได้ โดยสามารถเลือกเป็น แบ่งตามตามหมวดหมู่ของทรัพยากร, แบ่งตามภูมิภาค และแบ่งตาม IAM User เจ้าของทรัพยากร
3. แสดงรายละเอียดข้อมูลทรัพยากรในรูปแบบตาราง โดยจะระบุ id, ชื่อ IAM User, เวลาที่สร้าง, ภูมิภาค, ประเภททรัพยากร รวมไปถึงเรียงลำดับได้จากคอลัมในตารางแสดงผล

5.1.3 ผลของการทำงานโค้ดการดึงข้อมูลทรัพยากรสามารถดึงข้อมูลทรัพยากรที่มีแท็กได้ทุกประเภทและรวมไปถึงทรัพยากรที่ไม่ได้ติดตั้งการแท็กอัตโนมัติ สำหรับทรัพยากรที่ไม่ได้ติดตั้งการสร้างแท็กอัตโนมัติจะแก้ปัญหาโดยการพิมพ์แท็กเอง

**5.2 อภิปรายผล**

ผลของการพัฒนาระบบ ในช่วงที่ค้นคว้าหาข้อมูลได้ศึกษาวิธีการจากคลิปวิดิโอในยูทูบเป็นส่วนใหญ่และทดลองแบบลองผิดลองถูกต่าง ๆ จนได้ผลลัพธ์ที่ถูกต้องที่สุดกลับมา ส่งผลให้ไม่สามารถพัฒนาระบบทั้งหมดที่วางแผนไว้ได้ จำเป็นต้องพัฒนา 2 ระบบใหญ่ ๆ และยกส่วนที่เหลือไปทำต่อในโครงงาน 2 โดยการพัฒนาระบบสามารถนำมาอภิปรายได้ดังนี้

5.2.1 ระบบการสร้างแท็กอัตโนมัติใช้เวลาในการพัฒนานานที่สุดในโครงงาน 1 เนื่องจากทรัพยากรแต่ละประเภทใช้วิธีในการสร้างแท็กที่แตกต่างกัน ทำให้เกิดข้อผิดพลาดได้บ่อย ๆ และใช้เวลาแก้ไขนานบวกกับมีบางทรัพยากรที่ใช้วิธีสร้างแท็กต่างกับทรัพยากรอื่น ๆ โดยชิ้นเชิง เช่น จะต้องแปลงข้อมูลให้อยู่ในรูป ARN ก่อนถึงจะใช้งานได้ การสร้างแท็กทรัพยากรที่ถูกสร้างโดย Auto Scaling Group ไม่สามารถใช้ข้อมูล User จาก Event record ได้ จะสร้างแท็กโดยที่มีข้อมูล IAM User ได้จะต้องดึงข้อมูลแท็กจาก Auto Scaling Group ที่มีข้อมูล IAM User อยู่ และจำเป็นที่จะต้องเรียนรู้ลักษณะการทำงานของทรัพยากรแบบคร่าว ๆ จึงจะเข้าใจการสร้างแท็กได้

5.2.2 การดึงข้อมูลข้อมูลทรัพยากรเป็นส่วนที่ง่ายที่สุดจากการใช้การบริการ Resource Groups Tagging API ที่สามารถดึงข้อมูลทรัพยากรมาในรูปแบบ ARN และข้อมูลแท็กที่แนบบมา แต่จะใช้ระยะเวลาในการดึงข้อมมูลนานเพราะมีจำนวนทรัพยากรเยอะและดึงข้อมูลใน 2 ภูมิภาคพร้อมกัน

5.2.3 ในส่วนของการแสดงผล อาจารย์ที่ปรึกษาได้เข้ามาทดสอบการใช้งาน โดยสามารถเข้าใจข้อมูลที่แสดงออกมาได้จริงแต่จะต้องเพิ่มในส่วนของการค้นหาที่ดีกว่านี้และมีบางจุดที่ต้องปรับการแสดงผล เช่น การแสดงผล id ทรัพยากรที่ลดรูปแบบย่อจนไม่สามารถทราบถึงข้อมูลตรงนี้ได้ว่าเป็นทรัพยากรอันไหน

**5.3 ข้อเสนอแนะ**

5.3.1 สำหรับในการศึกษา ทดลอง และพัฒนาระบบในครั้งนี้ควรจะสร้างกรอบการทำงานที่จะช่วยเพิ่มความเป็นระเบียบและประสิทธิภาพในการทำงานมากขึ้น เพราะระบบการสร้างแท็กอัตโนมัติจะต้องวางใน 2 ภูมิภาค หมายความว่าโค้ดคำสั่งการทำงานจะแยกออกจากกันส่งผลให้เกิดข้อผิดพลาดได้ง่าย ควรจะมีที่อยู่ของโค้ดการทำงานที่จุดเดียวแล้วสามารถติดตั้งได้หลายภูมิภาคได้พร้อมกันและสามารถตั้งค่าปรับเปลี่ยนการตั้งค่าได้โดยที่ไม่ต้องไปแก้ไขโค้ดการทำงาน

5.3.2 สำหรับในการพัฒนาในโครงงาน 2 จะต้องพัฒนาฟังก์ชันที่เหลือให้แล้วเสร็จ ส่วนของเว็บแอปพลิเคชันมีระบบยืนยันตัวตน การสร้างแท็กทุกครั้งจะมีการบันทึกข้อมูลลงฐานข้อมูลเพื่อเพิ่มประสิทธิภาพในการดึงข้อมูลมาแสดงผล ศึกษาแนวทางในการลบทรัพยากรให้ถูกวิธี ศึกษาการคำนวนค่าใช้จ่าย และทรัพยากรที่ไม่ได้ติดตั้งการสร้างแท็กอัตโนมัติให้ยกไปทำในในส่วนท้ายสุดเพราะมีความเข้าใจในการพัฒนาแล้ว

**บรรณานุกรม**

1. Bill Dry. **“Automatically tag new AWS resources based on identity or role.”** [Online]. Available: https://aws.amazon.com/blogs/mt/auto-tag-aws-resources/ [ ที่เข้าถึง 4 สิงหาคม 2564 ]
2. devahoy. **“getting-started-with-nextjs.”** [Online].

Available: https://devahoy.com/blog/2020/03/getting-started-with-nextjs/

[ ที่เข้าถึง 6 สิงหาคม 2564 ]

1. Next.JS. **“Next.JS Documentation”** [Online].

Available: https://nextjs.org/docs

[ ที่เข้าถึง 8 สิงหาคม 2564 ]

1. Oliver Berger. **“CloudTrail Event Names – A Comprehensive List.”** [Online]. Available: https://www.gorillastack.com/blog/real-time-events/cloudtrail-event-names/ [ ที่เข้าถึง 11 สิงหาคม 2564 ]
2. AWS. **“CloudTrail log file examples.”** [Online]. Available: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file examples.html [ ที่เข้าถึง 11 สิงหาคม 2564 ]
3. Boto3. **“ResourceGroupsTaggingAPI.”** [Online]. Available:

https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/resourcegroup aggingapi.html#ResourceGroupsTaggingAPI.Client.get\_resources

[ ที่เข้าถึง 25 สิงหาคม 2564 ]

1. Boto3. **“EC2.”** [Online]. Available:

https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.ht ml [ ที่เข้าถึง 25 สิงหาคม 2564 ]

1. Twin.macro. **“twin.macro Next.js”** [Online]. Available:

https://github.com/ben-rogerson/twin.examples/tree/master/next-styled-components

[ ที่เข้าถึง 26 สิงหาคม 2564 ]

1. TailwindCSS. **“Tailwind CSS Documentation”** [Online]. Available:

https://tailwindcss.com/docs

[ ที่เข้าถึง 26 สิงหาคม 2564 ]

1. Styled Components. **“Styled** **Components Documentation”** [Online]. Available:

https://styled-components.com/docs

[ ที่เข้าถึง 2 กันยายน 2564 ]

1. Framer Motion. **“Framer Motion Documentation”** [Online]. Available:

https://framer.com/docs

[ ที่เข้าถึง 24 ตุลาคม 2564 ]