รายงานฉบับสมบูรณ์

โครงการขอรับการส่งเสริมและสนับสนุนจากเงินกองทุนวิจัยและพัฒนากิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคม เพื่อประโยชน์สาธารณะ

โครงการพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล

บริษัท นิภา เทคโนโลยี จำกัด

มกราคม 2568

กองทุนวิจัยและพัฒนากิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคม เพื่อประโยชน์สาธารณะ (สำนักงาน กสทช.)

แบบ กทปส. ME-003

รายงานฉบับสมบูรณ์

ทุนส่งเสริมและสนับสนุนการวิจัยและพัฒนา

สัญญารับทุนเลขที่ A63-1-(2)-019

โครงการพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล

Development of Cloud & Edge Cloud Computing Technology for Thai Industry

(คณะ) นักวิจัย

1.	ดร.อภิศักดิ์ จุลยา	นักวิจัยหัวหน้าโครงการ
2.	นายฐิติวัชร์ นันทนิตติ	นักวิจัยร่วม
3.	นางสาวณัชชา พูนศรีธนากูล	นักวิจัยร่วม
4.	นายชาญศิลป์ ชิ้นประเสิรฐ	นักวิจัยร่วม
5.	นายชยพล ลิมานนท์	นักวิจัยร่วม
6.	นายพิพิธพนธ์ นวลงาม	นักวิจัยร่วม
7.	นายคมกฤษ เวียงวิเศษ	นักวิจัยร่วม
8.	นายประทิน บุญรอด	นักวิจัยร่วม
9.	นายปัณณวิชญ์ บุรินทร์ทรัพย์ฤดี	นักวิจัยร่วม
10.	นายจีรพัฒน์ ศุภอภินันต์	นักวิจัยร่วม
11.	นายณัฐพร พูลสุขเสริม	นักวิจัยร่วม
12.	นางสาวเปมิกา โรจนรัตน์	นักวิจัยร่วม
13.	นายบุณยทัต อินสว่าง	นักวิจัยร่วม
14.	นายอภิวัฒน์ เปลี่ยนจิตรดี	นักวิจัยร่วม
15.	นายจำเริญ สีด้วง	นักวิจัยร่วม
16.	นายตรีภพ ชุณหะสันต์	นักวิจัยร่วม
17.	นายสรสิช ศณีเอี่ยมสะอาด	นักวิจัยร่วม
18.	นายพันเทพ ไทยประทุม	นักวิจัยร่วม

ได้รับทุนอุดหนุนจาก

กองทุนวิจัยและพัฒนากิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคม เพื่อประโยชน์สาธารณะ

(สำนักงาน กสทช.)

มกราคม 2568

บทสรุปผู้บริหาร

โครงการพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล

มกราคม 2568

โครงการวิจัยพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล มีวัตถุประสงค์เพื่อศึกษาและพัฒนาเทคโนโลยี Cloud Technology และ Edge Cloud Computing ให้รองรับการเข้าสู่ยุคอุตสาหกรรม 4.0 ที่มีความต้องการใช้งานโครงสร้างพื้นฐาน Cloud และ Edge Computing สำหรับการดำเนินธุรกิจหรืองานด้านบริการลูกค้า ระบบที่พัฒนานี้มีเป้าหมายเพื่อรองรับการนำเทคโนโลยี AI และ IoT มาผสานกับ Edge Computing ให้เกิดประสิทธิภาพสูงสุด

Edge จะมีบทบาทสำคัญในการเสริมความปลอดภัยของข้อมูล (Data Security) และเพิ่มความเป็นส่วนตัว (Privacy) ของผู้ใช้งาน อีกทั้งยังช่วยประมวลผลข้อมูลได้อย่างรวดเร็วและแบบเรียลไทม์ ซึ่งคุณสมบัตินี้เหมาะสมสำหรับการพัฒนาระบบอัตโนมัติ เช่น การควบคุมเครื่องจักรในโรงงาน และยานพาหนะอัตโนมัติสำหรับการขนส่งในกระบวนการผลิต โครงการนี้ประกอบด้วยการศึกษาวิจัยและพัฒนาใน 3 ส่วนหลัก เพื่อให้เทคโนโลยีที่พัฒนาขึ้นสามารถตอบโจทย์กับความต้องการของอุตสาหกรรมไทยได้อย่างมีประสิทธิภาพ

1. การศึกษาและพัฒนาระบบ Edge Computing, IoT Platform และ IoT Edge โดยการวิจัยจะมุ่งเน้นในการนำ Open Source Technology มาพัฒนาต่อยอด ซึ่งเป็นการศึกษาวิจัยในเชิงลึก เนื่องจากมีความซับซ้อนของการการทำงานร่วมกันของแต่ละระบบทั้ง Cloud Platform, Software Define Network (SDN) และ IoT Platform โดยได้ผลของการวิจัยดังต่อไปนี้

1.1. Edge Computing จะเป็นการทำงานร่วมกัน 3 Software คือ

- OpenStack คือ software สำหรับการจัดการ Virtualization เพื่อใช้งาน Resource ทั้ง CPU, RAM, Disk
- CEPH คือ software สำหรับจัดการ Storage ทั้ง Block, Object และ File
- OpenSDN คือ software สำหรับจัดการ Networking เช่น VPC (subnet), IP address, Routing, Security Group เป็นต้น

1.2. IoT Platform และ IoT Edge ได้นำ Software Thingsboard ที่เป็น Open Source มาพัฒนาและให้บริการตลอดโครงการ ซึ่งเป็น Software ที่มีการใช้งานที่ยืดหยุ่น หลากหลาย สอดคล้องกับโครงการ

2. การพัฒนาโครงสร้างของ Edge Computing เพื่อให้รองรับการใช้งานที่หลากหลายและสอดคล้องกับภาคธุรกิจจึงพัฒนา Edge Computing เป็น 2 รูปแบบคือ

2.1. Full Edge Computing คือเครื่องแม่ข่าย (Server) ขนาดใหญ่ที่สามารถให้บริการสำหรับภาคธุรกิจขนาดกลางและขนาดใหญ่ที่มีความต้องการใช้งานที่หลากหลายทั้ง Application, Data Process และ AI เป็นต้น โดยได้ทำการติดตั้ง Full Edge ไว้

2.2. IoT Edge คือเครื่องแม่ข่ายขนาดเล็กหรืออุปกรณ์ IoT ที่ทำงานเฉพาะ โดยเก็บข้อมูลได้ไม่มาก ประมวลผลข้อมูลที่มีขนาดเล็ก หรือการทำการจัดการข้อมูลการเก็บหรือส่งข้อมูลต่อได้

3. การพัฒนา IoT platform และ IoT Edge โดยการพัฒนาระบบ IoT Platform เพื่อให้บริการสำหรับผู้ที่สนใจทั้งภาครัฐและภาคเอกชน และพัฒนาระบบ IoT Edge ให้ทำงานร่วมกับ Edge Computing โดยให้มีการเก็บข้อมูลประมวลผลข้อมูลใกล้กับพื้นที่หรือผู้ใช้งานข้อมูลนั้นมากที่สุด

โดยโครงการวิจัยนี้ได้ดำเนินการพัฒนาทั้งส่วนของ Edge Computing และ IoT platform/IoT Edge ตามรายละเอียดข้างต้นโดยระบบมีความพร้อมในการให้บริการเป็นไปตามเป้าหมายและวัตถุประสงค์ของโครงการ และได้ประชาสัมพันธ์ผ่านช่องทางต่างๆ เช่น https://nipa.cloud, https://facebook.com และงาน Industrial IoT Solution Expo จัดโดยสภาอุตสาหกรรม เพื่อส่งเสริมให้ผู้สนใจทั้งหน่วยงานภาครัฐและภาคเอกชนได้เข้ามาทดลองเข้าใช้งานระบบ Thingsboard IoT Platform โดยมีหน่วยงานที่ประสงค์เข้ามาใช้งานระบบค่อนข้างน้อย ซึ่งปัญหาหลักเกิดจากความพร้อมทั้งบุคคลากรในการใช้งานและระบบเดิมที่ใช้งานไม่รองรับทั้งฮาร์แวร์และซอฟแวร์ อีกทั้งการนำไปประยุกต์ใช้งานยังไม่มีความคุ้มค่าต่อการลงทุน

ผลจากโครงการวิจัยนี้ คณะผู้วิจัยจะนำไปศึกษาและพัฒนาในเชิงพาณิชย์ทั้งส่วนของ Edge Computing และ IoT Platform/IoT Edge ต่อ ซึ่งจะเป็นส่วนหนึ่งในการผลักดันและขับเคลื่อนประเทศไทยให้ก้าวสู่อุตสาหกรรม 4.0 และลดการพึ่งพาเทคโนโลยีจากต่างประเทศ ลดการขาดดุลด้านการค้า สืบต่อไป

โครงการพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล

ดร.อภิศักดิ์ จุลยา

มกราคม 2568

โครงการพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล เป็นโครงการที่มีเป้าหมายสำคัญในการเสริมสร้างความก้าวหน้าด้านเทคโนโลยีดิจิทัลเพื่อสนับสนุนการพัฒนาเพิ่มขีดความสามารถของอุตสาหกรรมของไทยให้เข้าสู่ยุคอุตสาหกรรม 4.0 ได้อย่างเต็มรูปแบบ โดยการนำเทคโนโลยี Open Source ที่มีความทันสมัย ตัวอย่างเช่น OpenStack, OpenSDN และ Thingsboard IoT Platform มาใช้ในการพัฒนาระบบที่มีความยืดหยุ่นและตอบสนองต่อความต้องการของอุตสาหกรรมได้ในหลากหลายมิติ โครงการนี้ประกอบด้วยการวิจัยและพัฒนาที่ครอบคลุม 3 ด้านหลัก ได้แก่ การพัฒนาโครงสร้างพื้นฐานของ Edge Computing ที่ล้ำสมัย การสร้าง IoT Platform ที่รองรับการประยุกต์ใช้งานในรูปแบบต่าง ๆ และการพัฒนาระบบ IoT Edge ที่ช่วยเพิ่มศักยภาพในการประมวลผลข้อมูลใกล้กับแหล่งกำเนิดข้อมูลได้อย่างมีประสิทธิภาพสูงสุด

ผลการดำเนินงานของโครงการนำไปสู่การสร้างระบบที่สามารถลดต้นทุนในด้านการประมวลผลและการถ่ายโอนข้อมูลได้อย่างมีนัยสำคัญ โดยยังสามารถเพิ่มระดับความปลอดภัยและความเป็นส่วนตัวของข้อมูลตลอดจนช่วยลดเวลาการตอบสนองของระบบ ซึ่งทำให้มีเหมาะสำหรับการใช้งานในภาคอุตสาหกรรมที่มีความต้องการในการประมวลข้อมูลผลจากอุปกรณ์ IoT ได้แบบเรียลไทม์ นอกจากนี้ โครงการ Cloud & Edge Cloud Computing และยังมีบทบาทสำคัญในการลดการพึ่งพาการใช้งานเทคโนโลยีจากต่างประเทศ พร้อมทั้งเพิ่มขีดความสามารถของบุคลากรภายในประเทศผ่านการพัฒนาทักษะด้านเทคโนโลยีดิจิทัลที่เกี่ยวข้องอย่างยั่งยืน

ระบบที่พัฒนาขึ้นได้รับการออกแบบให้ตอบสนองต่อความต้องการของหน่วยงานภาครัฐและภาคเอกชน โดยเฉพาะ SME ที่สามารถใช้งานระบบได้โดยไม่มีค่าใช้จ่ายในช่วงระยะเวลาทดลองใช้ 2 ปี ซึ่งจะเป็นการสร้างโอกาสที่ดีในการเริ่มต้นใช้เทคโนโลยีขั้นสูง โดยระบบนี้ยังสนับสนุนการขยายตัวของภาคอุตสาหกรรมดิจิทัลภายในประเทศ และเป็นรากฐานที่สำคัญในการพัฒนาประเทศไทยไปสู่ Smart Thailand 4.0 ได้อย่างยั่งยืนในอนาคต ทั้งนี้โครงการยังช่วยสร้างความเชื่อมั่นในศักยภาพของประเทศในการรองรับการลงทุนจากต่างชาติ

พร้อมทั้งส่งเสริมการสร้างเครือข่ายความร่วมมือระหว่างหน่วยงานต่าง ๆ เพื่อยกระดับเทคโนโลยีให้สอดคล้องกับความต้องการของตลาดและอุตสาหกรรมในยุคดิจิทัล

Development of Cloud & Edge Cloud Computing Technology for Thai Industry

Dr. Abhisak Chulya

January 2025

The "Development of Cloud & Edge Cloud Computing Technology for Thai Industry" project is a key initiative aimed at advancing digital technology to enhance the capabilities of Thailand's industries in transitioning to Industry 4.0. By leveraging modern open-source technologies such as OpenStack and Thingsboard IoT Platform, the project develops highly flexible systems designed to meet the diverse demands of various industrial sectors. The initiative focuses on three main areas: developing advanced Edge Computing infrastructure, creating an IoT Platform capable of supporting various applications, and building IoT Edge systems that maximize data processing efficiency near the source.

The project results in systems that significantly reduce processing and data transfer costs while enhancing data security and privacy. These systems also minimize response times, making them ideal for real-time industrial applications. Moreover, the project plays a vital role in reducing reliance on foreign technologies and strengthening domestic expertise in relevant digital technology fields sustainably.

Designed to meet the needs of government agencies and private SMEs, the developed systems are offered free of charge during a two-year trial period. This initiative provides a valuable opportunity for organizations to adopt advanced technologies, supporting the expansion of the country's digital industry. The project serves as a critical foundation for Thailand’s journey toward Smart Thailand 4.0, fostering confidence in the nation's ability to attract foreign investment and encouraging collaborative networks among various sectors to elevate technology in line with market and industrial demands in the digital era.

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1) ที่มาและความสำคัญ

ปัจจุบันประเทศไทยกำลังเข้าสู่ยุคอุตสาหกรรม 4.0 อย่างเต็มตัวซึ่งภาคอุตสาหกรรมต่าง ๆ ได้มีการปรับตัวเข้าสู่ยุคดิจิทัล ทั้งในด้านการวางแผนปรับปรุงหรือปรับเปลี่ยนระบบที่สำคัญต่าง ๆ เพื่อให้การดำเนินธุรกิจเป็นไปอย่างมีประสิทธิภาพ ระบบ Edge Cloud Computing จึงกลายเป็น Workload หนึ่งที่เริ่มเข้ามามีบทบาทมากขึ้นในยุคปัจจุบันของอุตสาหกรรมภายในประเทศ เพื่อตอบรับต่อเทรนด์ทางด้านโซลูชั่นที่มีความต้องการเร่งการประมวลผลทางด้านข้อมูล เพื่อตอบกลับไปยังผู้ใช้งานได้อย่างรวดเร็ว รวมถึงเทคโนโลยี 5G มีความจำเป็นที่จะต้องใช้ Edge Cloud Computing ในการประมวลผลเพื่อให้เกิดประสิทธิภาพสูงที่สุด

บริษัท นิภา เทคโนโลยี จำกัด มีแนวคิดในการพัฒนานวัตกรรม Edge Cloud Computing เพื่อตอบสนองภาคอุตสาหกรรมในประเทศ โดยเฉพาะการปรับตัวเข้าสู่ยุคดิจิทัล ระบบ Edge Cloud Computing จะเข้ามามีบทบาทในทุกธุรกิจ เพราะมีความรวดเร็วทั้งในส่วนของ Respond Time และ Transfer Rate มีความปลอดภัยและความถูกต้องของข้อมูลขณะประมวลผลสูง ช่วยลดต้นทุนในการสื่อสาร และเพิ่มความสะดวกรวดเร็วทำให้งานด้านบริการง่ายขึ้นทั้งในส่วนของผู้ให้และผู้รับบริการ หลังจากการทำ Research พบว่าในต่างประเทศมีการนำซอฟต์แวร์ StarlingX และ OpenStack ซึ่งเป็น Open Source Software มาให้บริการ Edge Cloud Computing เพื่อตอบโจทย์ความต้องการของธุรกิจที่ก้าวกระโดด และมีความยืดหยุ่นสูง ทำให้ต้นทุนในการพัฒนา Edge Cloud Computing ต่ำกว่าการใช้เทคโนโลยีอื่น อีกทั้งภายในประเทศไทยยังไม่มีผู้ให้บริการที่พัฒนา Edge Cloud Computing มาจาก OpenStack เนื่องจากมีความยุ่งยากและซับซ้อนทั้งในด้านการติดตั้งและการพัฒนา จำเป็นต้องใช้บุคลากรที่มีความเชี่ยวชาญสูง

โครงการนี้เล็งเห็นคุณประโยชน์ของการใช้งาน Open Source software (OpenStack) ที่มีความโดดเด่นในเรื่องของประสิทธิภาพ และสามารถ Optimize ให้เป็นไปตามความต้องการของผู้ใช้งาน และยังเป็นรากฐานที่สำคัญในการพัฒนาเทคโนโลยี Edge Cloud Computing ภายในประเทศ อีกทั้งบริษัทมีความพร้อมทั้งในด้านของทรัพยากรและบุคลากร ที่จะสามารถดำเนินโครงการให้แล้วเสร็จได้ตามเป้าหมายที่กำหนดไว้เพื่อรองรับยุทธศาสตร์ประเทศไทย 4.0 และระเบียงเขตเศรษฐกิจตะวันออก (EEC) ทำให้ประเทศไทยเป็นตลาดที่ดึงดูดความสนใจและมีความมั่นคงในการมาลงทุน โดยเฉพาะโครงการขนาดใหญ่ต่าง ๆ ที่เป็นเทคโนโลยี เช่น IoT, AI หรือ Smart Places ที่จะต้องการการวิเคราะห์ข้อมูลแบบเรียลไทม์ถือว่ามีบทบาทสำคัญในการขับเคลื่อนธุรกิจและการลงทุนในอนาคตของประเทศ ปัจจุบันประเทศไทยยังต้องพึ่งพาเทคโนโลยีและซอฟต์แวร์จากต่างประเทศ ดังนั้นโครงการนี้เป็นโครงการพัฒนาเพื่อทดแทนเทคโนโลยีต่างประเทศ ส่งผลให้ธุรกิจไทยและธุรกิจข้ามชาติสามารถเข้าถึงและใช้งานได้จริง สร้างรายได้เข้าสู่ภายในประเทศ ลดปัญหาการขาดดุลทางเศรษฐกิจและสังคมในอนาคตต่อไป

1.2) วัตถุประสงค์ และขอบเขตของโครงการ

โครงการจะดำเนินกิจกรรมตามวัตถุประสงค์ในการส่งเสริมและสนับสนุนการพัฒนาทรัพยากรสื่อสาร การวิจัย และพัฒนากิจการกระจายเสียง เทคโนโลยีสารสนเทศ ในอุตสาหกรรมต่อเนื่อง ในด้านการวิจัยและพัฒนาเชิงนวัตกรรม เทคโนโลยีสารสนเทศและโทรคมนาคมด้านอุตสาหกรรม 4.0 ตรงตามเป้าหมายของการจัดสรรเงินตามมาตรา 52 โดยมีวัตถุประสงค์ของโครงการ ดังนี้

1. เพื่อพัฒนาเทคโนโลยี Edge Cloud Computing ด้วย OpenStack Cloud และ StarlingX เพื่อสนับสนุนการยกระดับตามกรอบอุตสาหกรรม 4.0 (Industry 4.0)

2. เพื่อสร้างนวัตกรรมใหม่ ด้าน Edge Cloud Computing และ IoT Platform ที่เหมาะสม กับอุตสาหกรรมทุกภาคส่วนทั้งขนาดกลาง ขนาดใหญ่ และภาครัฐของประเทศไทย

3. เพื่อการนำเทคโนโลยี Edge Cloud Computing ที่เป็น Open source มาพัฒนาต่อยอด ทำให้ไม่ต้องพึ่งพาเทคโนโลยีจากต่างประเทศ อีกทั้งช่วยพัฒนาบุคลากรภายในประเทศให้มีความรู้ความสามารถมากขึ้น เป็นการเพิ่มขีดความสามารถของประเทศไทย และช่วยลดภาวะการขาดดุลของประเทศได้ในระยะยาว

4. เพื่อรองรับการประมวลผลสำหรับ IoT Platform ที่จะสามารถรองรับ IoT applications ได้หลากหลาย สามารถรวบรวมข้อมูล ประมวลผลข้อมูล และส่งกลับของข้อมูลได้ง่ายสะดวกรวดเร็ว อย่างมีประสิทธิภาพ และมีความปลอดภัย ช่วยให้องค์กรต่างๆ สามารถนำไปใช้ในการพัฒนา IoT Solutions ที่เหมาะสมเพื่อให้บริการแก่ลูกค้าได้อย่างเต็มประสิทธิภาพ

1.3) ประโยชน์ที่คาดว่าจะได้รับ

โครงการจะช่วยแก้ไขปัญหาภาคธุรกิจและอุตสาหกรรมที่เกี่ยวโยงกับเทคโนโลยี IoT และ AI ด้วยเทคโนโลยี Edge Cloud Computing ภายในประเทศ เพื่อเพิ่มประสิทธิภาพของการใช้ IoT และ AI ทำให้เกิดความรวดเร็วในการประมวลผล ลดคอขวดในการ Transfer ข้อมูล ลด Latency และลดการร่วมศูนย์ (Decentralize) ของข้อมูลให้การประมวลผลอยู่ใกล้ผู้ใช้ให้มากที่สุด อีกทั้งยังยกระดับอุตสาหกรรม 4.0 ให้สามารถเข้าถึงและรับคำปรึกษาจากผู้เชี่ยวชาญ ในการนำ Edge Cloud Computing ไปใช้ให้เกิดประโยชน์สูงสุดในภาคอุตสาหกรรม

Edge Cloud Computing จะช่วยขับเคลื่อนการเติบโตของธุรกิจ เพื่อให้บริหารจัดการ Application ได้อย่างง่ายดาย และสามารถโยกย้ายข้อมูล (Data) ไปมาได้ทั้ง Public Cloud, Private Cloud และ Edge Cloud สามารถรองรับ Application ได้ทุกขนาด และทุกรูปแบบด้วยต้นทุนรวมที่ลดลงอย่างมาก ส่งผลให้องค์กรสามารถให้บริการ IoT Solutions ได้อย่างรวดเร็ว

โครงการนี้ไม่ใช่แค่ทำเป็นต้นแบบ (Proof of Concept) แต่เมื่อสิ้นสุดโครงการ จะมีระบบ Edge Cloud Computing และ Mobile Edge Computing (MEC) Platform-as-a-Service ที่ส่งผลอย่างต่อเนื่องในการพัฒนา Smart Thailand 4.0 ยกระดับอุตสาหกรรมและเทคโนโลยีของประเทศ ลดต้นทุนในการพัฒนา IoT Solutions และการลงทุนด้านอุปกรณ์ฮาร์ดแวร์และซอฟต์แวร์ เพิ่มอัตราการลงทุนจากต่างประเทศ สร้างความเข้มแข็งให้กับองค์กรต่างๆ และช่วยเพิ่มประสิทธิภาพในการดำเนินการด้านต่างๆ เช่น การขนส่งและโลจิสติกส์ อุตสาหกรรมการผลิต อุตสาหกรรมการแพทย์และ

สุขภาพ ความปลอดภัย โดยจะเปิดให้บริการเต็มรูปแบบต่อสาธารณะที่เป็นทั้งหน่วยงานภาครัฐและองค์กรเอกชน SME ภายในประเทศ โดยไม่คิดค่าใช้จ่ายเป็นระยะเวลา 2 ปี

ทางบริษัทคาดการณ์ว่าจะมีหน่วยงานภาครัฐเป็นอย่างน้อย 2 หน่วยงานที่จะให้ความร่วมมือในการนำระบบมาเชื่อมต่อกับ Edge Cloud Computing เพื่อช่วยในการมอนิเตอร์และควบคุมอุปกรณ์เพื่อให้บริการอันเป็นประโยชน์กับสาธารณชน เมื่อประสบความสำเร็จจะมีประโยชน์กับประชาชนที่ได้รับบริการจากภาครัฐหน่วยงานนั้นๆ มากกว่า 1 ล้านคน

ในส่วนภาคเอกชน ในช่วงของการพัฒนาระบบ ผู้ใช้จะได้ประโยชน์จากการทำงานร่วมกันระหว่างบริษัทที่มีการใช้ระบบ Edge Cloud Computing อยู่แล้ว และบริษัทที่มีการวางแผนจะใช้ระบบ Edge Cloud Computing ซึ่งจะทำให้ Platform ของโครงการ มีฟังก์ชันการใช้งานที่ตรงตามความต้องการของภาครัฐ และเอกชนในปัจจุบัน ระบบที่ติดตั้งสามารถรองรับการใช้งานได้สูงสุดถึง 500 หน่วยงาน โดยบริษัทตั้งเป้าการใช้งานจากภาคเอกชนอย่างน้อย 28 หน่วยงาน

1.4) ผลผลิตสำคัญ

ตารางที่ 1 ผลผลิตสำคัญของโครงการ

ลำดับ	ผลผลิต	หน่วยวัด	ตัวชี้วัด (เชิงคุณภาพ)
1	Edge Cloud Computing Cluster	ระบบ	Edge Cloud Computing Cluster ติดตั้งทั้งหมด 5 ภูมิภาค
2	IoT Platform	Platform	IoT Platform จำนวน 1 platform

1.5) แผนปฏิบัติการโครงการ/แผนการดำเนินงาน

ตารางที่ 2 แผนปฏิบัติการโครงการ/แผนการดำเนินงาน (1)

ลำดับ	กิจกรรมที่สำคัญ	ระยะเวลาการดำเนินกิจกรรม (1)								น้ำหนัก (%)
		ประจำปี 2564				ประจำปี 2565				
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
1	Researchเทคโนโลยี Open source ด้าน Edge Cloud Computing และ StarlingX ในเรื่องซอฟต์แวร์, features, ระบบ Network, แนวทางการพัฒนาซอฟต์แวร์ และการนำไปใช้งาน									13
2	กำหนด Fearture ที่จำเป็นสำหรับ Edge Cloud Computing และ Mobile Edge Computing (MEC) Platform-as-a-Service									2
3	ดำเนินการซื้อฮาร์ดแวร์สำหรับการ POC									2
4	วิเคราะห์เพื่อวางแผนการพัฒนา Features ต่างๆ									2
5	จัดทำรายงานแผนการดำเนินงาน และจัดทำรายงานเบื้องต้น									1
6	ออกแบบ Edge Cloud Computing Network สำหรับใช้ในการพัฒนา Edge Cloud Computing Cluster									5
7	ดำเนินการซื้อฮาร์ดแวร์สำหรับ Production									2
8	หาข้อมูล Data Center เพื่อวาง Co-location									1
9	ดำเนินการติดต่อและหาความร่วมมือกับผู้ประกอบการตัวอย่างเพื่อนำมาทดสอบระบบ									5

ลำดับ	กิจกรรมที่สำคัญ	ระยะเวลาการดำเนินกิจกรรม (1)								น้ำหนัก (%)
		ประจำปี 2564				ประจำปี 2565				
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
10	ดำเนินการติดตั้งฮาร์ดแวร์และซอฟต์แวร์ บนระบบ Edge Cloud Computing (POC)									7
11	ออกแบบหลักเกณฑ์ในการทดสอบซอฟต์แวร์									1
12	ติดตามประเมินผลความก้าวหน้าของโครงการ									0.5
13	จัดทำรายงานความก้าวหน้าของโครงการฉบับที่ 1 งวดที่ 2									0.5
14	ดำเนินการติดตั้งฮาร์ดแวร์และซอฟต์แวร์ บนระบบ Edge Cloud Computing (Production)									8
15	ทดสอบ, ปรับปรุง, และแก้ไขระบบให้เกิดความสมบูรณ์พร้อมใช้งาน									5
16	เก็บ requirements จากผู้ประกอบการตัวอย่างเพื่อนำมาออกแบบระบบให้เหมาะสมกับผู้ใช้งานมากขึ้น									1
17	ติดตาม ประเมินผล ความก้าวหน้าของโครงการ									0.5

ลำดับ	กิจกรรมที่สำคัญ	ระยะเวลาการดำเนินกิจกรรม (1)								น้ำหนัก (%)
		ประจำปี 2564				ประจำปี 2565				
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
18	จัดทำรายงานความก้าวหน้าของโครงการฉบับที่ 2 งวดที่ 3									0.5
19	ออกแบบและพัฒนา UX/UI ของ IoT Platform เพื่อให้ง่ายต่อการใช้งานของ User									2
20	ดำเนินการติดตั้ง IoT Platform ทั้ง Front End และ Back End บน production									10
21	ดำเนินการคัดเลือกผู้ประกอบการจากหลากหลายกลุ่มอุตสาหกรรม									1.5
22	ประกาศผลการคัดเลือกผู้ประกอบการทั้ง 30 หน่วยงาน									0.5
23	ติดตาม ประเมินผล ความก้าวหน้าของโครงการ									0.5
24	จัดทำรายงานความก้าวหน้าของโครงการฉบับที่ 3 งวดที่ 4									0.5
25	นำระบบของผู้ประกอบการเชื่อมต่อกับ Edge Cloud Computing									4
26	เก็บ Feedback ของผู้ประกอบการทั้ง 30 หน่วยงาน เพื่อนำมาพัฒนา UX/UI ของ IoT Platform									1.5
27	ทดสอบ, ปรับปรุง, และแก้ไขระบบให้เกิดความสมบูรณ์พร้อมใช้งาน									2

ตารางที่ 3 แผนปฏิบัติการโครงการ/แผนการดำเนินงาน (2)

ลำดับ	กิจกรรมที่สำคัญ	ระยะเวลาการดำเนินกิจกรรม (2)									
		ประจำปี 2566				ประจำปี 2567				2568	น้ำหนัก (%)
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	
28	เปิดระบบ IoT Platform ให้หน่วยงานภาครัฐและองค์กรเอกชน SME ภายในประเทศเข้าใช้บริการ										0.5
29	ติดตาม ประเมินผล ความก้าวหน้าของโครงการ										0.5
30	จัดทำรายงานความก้าวหน้าของโครงการฉบับที่ 4 งวดที่ 5										0.5
31	ดำเนินการติดตั้งระบบ Full Edge Computing ของภาคตะวันออกเฉียงเหนือ จังหวัดขอนแก่น										2
32	ดำเนินการติดตั้งระบบ IoT Edge 5 ภูมิภาค										2
33	ดูแลและบำรุงรักษาระบบ IoT Platform ให้สามารถใช้งานได้										10
34	จัดอบรม Workshop “โครงการการเรียนรู้และเสริมศักยภาพอุตสาหกรรมไทยด้วยระบบ IoT”										1.5
35	รายงานความก้าวหน้าประจำปี 67										0.5
36	จัดทำรายงานของโครงการฉบับสมบูรณ์										2

บทที่ 2
ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1) แนวคิดในการพัฒนาโครงการ

บทนี้มีจุดมุ่งหมายสำคัญเพื่อแสดงกรอบแนวคิดที่เป็นรากฐานในการพัฒนาโครงการโดยคำนึงถึงสภาพแวดล้อมทางเทคโนโลยีสารสนเทศและการสื่อสารในปัจจุบันที่มีการขยายตัวอย่างรวดเร็วของข้อมูลและอุปกรณ์ IoT การนำแนวคิด Edge Cloud Computing มาใช้ถือเป็นหนึ่งในกลยุทธ์สำคัญ เนื่องจากสามารถกระจายทรัพยากรการประมวลผลและการจัดเก็บข้อมูลมายังบริเวณใกล้กับแหล่งกำเนิดข้อมูลได้มากขึ้น ลดระยะทางการส่งข้อมูลสู่ระบบศูนย์กลาง และลดความหน่วงเวลาในการตอบสนอง พร้อมทั้งสามารถบรรเทาภาระการประมวลผลบนศูนย์ข้อมูลหลักได้อย่างมีประสิทธิภาพ

ภายใต้กรอบดังกล่าว การเลือกใช้ IoT Platform ที่เหมาะสมยิ่งมีความสำคัญ เนื่องจากแพลตฟอร์มดังกล่าวทำหน้าที่เป็นศูนย์กลางในการเชื่อมโยงอุปกรณ์หลากหลายรูปแบบ จัดการข้อมูลที่เกิดขึ้นมหาศาล และนำเสนอโครงสร้างสำหรับการบริหารจัดการอุปกรณ์ ตลอดจนการนำข้อมูลไปวิเคราะห์เพื่อสนับสนุนการตัดสินใจ ทั้งนี้ IoT Platform ที่เลือกใช้งานควรมีความสามารถในการรองรับโปรโตคอลหลากหลาย, จัดการข้อมูลแบบเรียลไทม์, แสดงผลข้อมูลในรูปแบบที่เข้าใจง่าย รวมถึงมีความยืดหยุ่นในการปรับใช้ร่วมกับสถาปัตยกรรม Edge Cloud Computing เพื่อให้กระบวนการประมวลผลเบื้องต้นเกิดขึ้นใกล้กับอุปกรณ์ยิ่งขึ้น

ในส่วนของแนวคิดและเทคโนโลยีที่เกี่ยวข้อง มีการพิจารณา MEC (Multi-Access Edge Computing) ซึ่งเป็นสถาปัตยกรรมที่ผลักทรัพยากรการประมวลผลเข้าสู่ชั้นการเข้าถึงเครือข่าย (Access Layer) ช่วยเพิ่มประสิทธิภาพการให้บริการเรียลไทม์ นอกจากนี้ ยังพิจารณาเครื่องมือและแพลตฟอร์มสำหรับโครงสร้างพื้นฐาน Cloud เช่น OpenStack และ StarlingX ซึ่งเป็นระบบ Open Source ที่ปรับแต่งได้ตามข้อกำหนด เพื่อรองรับการทำงานร่วมกันระหว่างศูนย์ข้อมูลหลักและ Edge Node ส่วน OpenSDN (Software Defined Networking) ช่วยเสริมศักยภาพในการบริหารจัดการเครือข่ายให้มีความยืดหยุ่นและสามารถปรับเปลี่ยนตามสภาพโหลดได้อย่างมีประสิทธิภาพ

การเปรียบเทียบ IoT Platform หลายรูปแบบ ทั้ง Open Source และเชิงพาณิชย์ ยังเป็นส่วนสำคัญในการเลือกใช้เทคโนโลยีที่เหมาะสมที่สุด โดยพิจารณาจากเกณฑ์ต่าง ๆ เช่น ความง่ายในการปรับตั้งค่า ความสามารถในการรองรับปริมาณอุปกรณ์มหาศาล ความมั่นคงปลอดภัย และศักยภาพในการทำงานร่วมกับโครงสร้างพื้นฐาน Edge Cloud Computing ด้วยเหตุนี้ การคัดเลือก IoT Platform ที่เหมาะสมจะช่วยให้ระบบตอบสนองต่อความต้องการในทางปฏิบัติได้อย่างมีประสิทธิผล

กล่าวโดยสรุป แนวคิดในการพัฒนาโครงการนี้ครอบคลุมถึงการวางสถาปัตยกรรม Edge Cloud Computing ที่ผนวกเข้ากับ IoT Platform รวมถึงการอ้างอิงแนวคิดและเทคโนโลยีที่เกี่ยวข้อง เช่น MEC, OpenStack, StarlingX และ OpenSDN ตลอดจนการเปรียบเทียบทางเลือกด้าน IoT Platform ที่หลากหลายเพื่อให้ได้โซลูชันที่เหมาะสมที่สุด แนวทางดังกล่าวจะส่งเสริมการประมวลผลข้อมูล IoT ให้เกิดขึ้นใกล้แหล่งกำเนิด สร้างระบบที่มีความยืดหยุ่น ปรับขยายได้ง่าย ตอบสนองรวดเร็ว และมีความมั่นคงปลอดภัย ตลอดจนพร้อมรองรับการเปลี่ยนแปลงทางเทคโนโลยีและความต้องการในอนาคตได้อย่างสมบูรณ์ยิ่งขึ้น

2.1.1) Internet of Things (IoT)

Internet of Things (IoT) คือการเปลี่ยนแปลงครั้งสำคัญในโลกของเทคโนโลยีดิจิทัล ซึ่งทำให้อุปกรณ์ต่าง ๆ สามารถเชื่อมต่อกันผ่านอินเทอร์เน็ตเพื่อแลกเปลี่ยนข้อมูลได้ ระบบ IoT เริ่มต้นด้วยการใช้การสื่อสารแบบ Device-to-System Communication ที่อุปกรณ์จะส่งข้อมูลไปยังระบบเป้าหมายโดยตรง แม้ว่าโมเดลนี้จะช่วยให้เกิดการพัฒนาขั้นต้นในหลายด้าน แต่กลับเผชิญกับความท้าทายที่สำคัญ โดยเฉพาะเรื่องการทำงานร่วมกันของระบบ (Interoperability) ที่เกิดจากการใช้โปรโตคอลที่แตกต่างกันโดยผู้ผลิตแต่ละราย ทำให้เกิดปัญหาในการรวมข้อมูลเพื่อวิเคราะห์เชิงลึก นอกจากนี้ การพึ่งพาศูนย์ข้อมูลกลาง (Centralized Data Center) ยังทำให้เกิดความล่าช้า (Latency) ในการประมวลผลข้อมูล โดยเฉพาะเมื่อแหล่งข้อมูลอยู่ห่างไกลจากศูนย์กลาง ส่งผลให้ระบบไม่สามารถตอบสนองในแบบเรียลไทม์ (Real-time Responsiveness) ได้อย่างเต็มประสิทธิภาพ

ในยุคปัจจุบัน การพัฒนา IoT ได้เปลี่ยนไปสู่โมเดลที่กระจายตัวมากขึ้น เช่น การใช้ Edge Computing และ Fog Computing เพื่อลดความล่าช้าในการประมวลผล การพัฒนามาตรฐานกลาง เช่น MQTT และ CoAP ยังช่วยเพิ่มความสามารถในการทำงานร่วมกันของอุปกรณ์จากผู้ผลิตหลากหลายราย นอกจากนี้ ระบบ IoT ยังสามารถขยายการใช้งานในระดับใหญ่ขึ้น เช่น การสร้างเมืองอัจฉริยะ (Smart City) หรือระบบอุตสาหกรรมที่ต้องการควบคุมอุปกรณ์แบบเรียลไทม์

IoT Architecture

IoT Architecture เป็นโครงสร้างพื้นฐานสำคัญที่ช่วยให้อุปกรณ์ IoT สามารถเชื่อมต่อและทำงานร่วมกันได้อย่างมีประสิทธิภาพในระบบที่ซับซ้อน โดยครอบคลุมกระบวนการต่าง ๆ ตั้งแต่การเก็บรวบรวมข้อมูล การประมวลผล การวิเคราะห์เชิงลึก ไปจนถึงการนำเสนอข้อมูลในรูปแบบที่สามารถใช้งานได้จริง โครงสร้างนี้แบ่งออกเป็น 4 ชั้นหลัก ซึ่งแต่ละชั้นมีบทบาทที่สำคัญและเชื่อมโยงกันเพื่อให้การทำงานของระบบ IoT เป็นไปอย่างราบรื่นและมีประสิทธิภาพสูงสุด บทบาทของแต่ละชั้นรวมถึงการจัดการข้อมูลในรูปแบบที่ปรับเปลี่ยนได้ การรองรับการเชื่อมต่อแบบไร้รอยต่อ และการสนับสนุนเทคโนโลยีใหม่ เช่น Machine Learning เพื่อเพิ่มศักยภาพในการประมวลผลและวิเคราะห์ข้อมูลในเชิงลึก ตลอดจนการพัฒนาการเชื่อมโยงระหว่างระบบและผู้ใช้งานอย่างยั่งยืน

- **Perception Layer**

Perception Layer คือส่วนแรกของระบบ IoT ซึ่งทำหน้าที่รวบรวมข้อมูลจากเซ็นเซอร์และอุปกรณ์ IoT ต่าง ๆ ที่ติดตั้งในพื้นที่หลากหลาย ไม่ว่าจะเป็นการวัดอุณหภูมิ ความชื้น การตรวจจับการเคลื่อนไหว หรือแรงดัน ข้อมูลเหล่านี้จะถูกแปลงเป็นสัญญาณดิจิทัลและส่งต่อไปยังส่วนถัดไปของระบบอย่างรวดเร็วและมีประสิทธิภาพ

ชั้นนี้อุปกรณ์จะได้รับการออกแบบให้สามารถทำงานได้ในสภาพแวดล้อมที่หลากหลาย เช่น พื้นที่ที่มีสัญญาณรบกวนสูง หรือพื้นที่ห่างไกลที่มีข้อจำกัดด้านพลังงานและการเชื่อมต่อ นอกจากนี้ เซ็นเซอร์ยังมีความสามารถในการประหยัดพลังงานเพื่อยืดอายุการใช้งาน รวมถึงการส่งข้อมูลผ่านเครือข่ายไร้สายที่มีความมั่นคงและน่าเชื่อถือ เพื่อให้มั่นใจว่าข้อมูลที่ส่งออกไปมีความแม่นยำและครบถ้วน

Perception Layer ยังรองรับการประมวลผลเบื้องต้นในตัวอุปกรณ์ เช่น การกรองข้อมูลที่ไม่จำเป็นหรือการคำนวณค่าพื้นฐานเพื่อลดปริมาณข้อมูลที่ต้องส่งต่อไปยังชั้นอื่น การทำงานนี้ช่วยลดภาระของระบบประมวลผลกลาง และเพิ่มประสิทธิภาพของระบบ IoT โดยรวม นอกจากนี้ เทคโนโลยีที่พัฒนาต่อเนื่อง เช่น เซ็นเซอร์ที่มีความไวสูงและการใช้โปรโตคอลการสื่อสารมาตรฐาน ยังช่วยให้ Perception Layer เป็นรากฐานสำคัญที่สนับสนุนการพัฒนาระบบ IoT ที่ทันสมัยและยืดหยุ่นต่อความต้องการในอนาคต

- **Network Layer**

Network Layer ทำหน้าที่สำคัญในการส่งข้อมูลจาก Perception Layer ไปยัง Processing Layer โดยกระบวนการนี้ใช้เทคโนโลยีการเชื่อมต่อหลากหลายรูปแบบเพื่อให้การส่งข้อมูลมีประสิทธิภาพสูงสุด เทคโนโลยีเหล่านี้ช่วยรับประกันว่าข้อมูลสามารถส่งผ่านได้อย่างราบรื่นและรวดเร็ว ไม่ว่าจะเป็นในสภาพแวดล้อมที่มีข้อจำกัดด้านทรัพยากรหรือในระบบที่ต้องการความเสถียรสูง เช่น:

WiFi: รองรับการเชื่อมต่อระยะใกล้ที่ให้ความเร็วสูงมากในการส่งข้อมูล เหมาะสำหรับการใช้งานในพื้นที่ที่มีความต้องการส่งข้อมูลแบบเรียลไทม์และมีความหนาแน่นของอุปกรณ์สูง

LoRaWAN: ออกแบบมาสำหรับการเชื่อมต่อในระยะทางไกลโดยมีการใช้พลังงานต่ำเพื่อช่วยยืดอายุการใช้งานของอุปกรณ์ IoT ซึ่งเหมาะสมสำหรับการใช้งานในพื้นที่ที่ห่างไกลหรือระบบที่ไม่ต้องการการส่งข้อมูลแบบเรียลไทม์

Bluetooth Low Energy (BLE): เทคโนโลยีที่เน้นการใช้พลังงานต่ำเหมาะสำหรับอุปกรณ์ IoT ขนาดเล็ก เช่น อุปกรณ์สวมใส่ และเซ็นเซอร์ในระบบ Smart Home ที่ต้องการประหยัดพลังงานและใช้งานได้ยาวนาน BLE ช่วยลดความถี่ในการชาร์จแบตเตอรี่และเพิ่มความสะดวกในการเชื่อมต่อข้อมูลสำหรับการใช้งานในชีวิตประจำวัน

Zigbee: เทคโนโลยีการสื่อสารแบบไร้สายที่ออกแบบมาสำหรับอุปกรณ์ IoT ที่ใช้พลังงานต่ำและต้องการการเชื่อมต่อแบบเครือข่าย Mesh Network เพื่อเพิ่มความครอบคลุมและเสถียรภาพในพื้นที่ใหญ่ ระบบสมาร์ทโฮมและเซ็นเซอร์ในอุตสาหกรรมชั้นนี้ยังสนับสนุนการสื่อสารแบบ Machine-to-Machine (M2M) เพื่อการเชื่อมต่อที่รวดเร็วและปลอดภัย นอกจากนี้ เทคโนโลยียังรองรับการอัปเดตซอฟต์แวร์แบบ Over-The-Air (OTA) ซึ่งช่วยลดความยุ่งยากในการบำรุงรักษาและเพิ่มความสะดวกในการจัดการอุปกรณ์ในระบบที่ซับซ้อน

- **Processing Layer**

Processing Layer เป็นศูนย์กลางในการประมวลผลข้อมูลที่ได้รับ โดยใช้เทคโนโลยีขั้นสูงเพื่อจัดการและแปลงข้อมูลให้กลายเป็นรูปแบบที่สามารถใช้งานได้อย่างมีประสิทธิภาพ ชั้นนี้มีบทบาทสำคัญในการเชื่อมโยงข้อมูลจาก Perception Layer และ Network Layer ก่อนส่งต่อไปยัง Application Layer

- Edge Computing: ช่วยลดความล่าช้า (Latency) ในการประมวลผลข้อมูลโดยเน้นการจัดการข้อมูลที่มีความสำคัญใกล้กับแหล่งกำเนิดข้อมูล ซึ่งส่งผลช่วยให้ลดการพึ่งพาระบบประมวลผลจากศูนย์กลางและปรับปรุงความเร็วในการตอบสนองของระบบ ดังนั้นจึงมีความเหมาะสำหรับการใช้งานที่ต้องการข้อมูลแบบเรียลไทม์และการประมวลผลที่รวดเร็ว เช่น ในระบบ IoT ที่มีการใช้งานอย่างต่อเนื่องและมีความสำคัญสูง

- Cloud Computing: สนับสนุนการประมวลผลและจัดเก็บข้อมูลปริมาณมากในศูนย์ข้อมูลกลาง โดยมีความสามารถในการขยายตัวเพื่อรองรับการประมวลผลในระดับสูง เหมาะสำหรับระบบที่ต้องการการจัดการข้อมูลที่มีโครงสร้างซับซ้อน หรือการวิเคราะห์ข้อมูลเชิงลึกในขนาดใหญ่ เช่น การวิเคราะห์แบบ Big Data การใช้ Machine Learning เพื่อคาดการณ์ หรือการดำเนินการวิเคราะห์ที่เกี่ยวข้องกับข้อมูลเชิงสถิติ นอกจากนี้ Cloud Computing ยังช่วยให้ผู้ใช้งานสามารถเข้าถึงข้อมูลและประมวลผลได้จากทุกที่ผ่านอินเทอร์เน็ต เพิ่มความสะดวกและประสิทธิภาพในการจัดการทรัพยากรคอมพิวเตอร์ในระบบ IoT ที่มีการขยายตัวอย่างต่อเนื่อง

Processing Layer ยังรองรับการใช้งานเทคโนโลยีปัญญาประดิษฐ์ Machine Learning เพื่อช่วยวิเคราะห์ข้อมูลและสร้างการคาดการณ์ที่แม่นยำ นอกจากนี้ยังสามารถรวมข้อมูลจากหลายแหล่งเพื่อลดความซับซ้อนและเพิ่มประสิทธิภาพของระบบโดยรวม การทำงานในชั้นนี้เป็นรากฐานสำคัญที่ช่วยให้ระบบ IoT สามารถตอบสนองต่อความต้องการที่ซับซ้อนและหลากหลายในยุคปัจจุบันได้อย่างเต็มรูปแบบ

- **Application Layer**

Application Layer ในสถาปัตยกรรม IoT ที่มีบทบาทในการเชื่อมโยงระหว่างข้อมูลที่ผ่านการประมวลผลกับการใช้งานจริงของผู้ใช้ โดยเน้นการนำข้อมูลที่ซับซ้อนมานำเสนอในรูปแบบที่เข้าใจง่าย เพื่อให้สอดคล้องกับความต้องการและลักษณะการใช้งานที่หลากหลาย ทั้งนี้ Application Layer ยังสนับสนุนการตัดสินใจในระบบที่ต้องการการตอบสนองแบบเรียลไทม์ การปรับตัวตามสถานการณ์ และการวิเคราะห์ข้อมูลเชิงลึก

การออกแบบในชั้นนี้มุ่งเน้นให้เกิดการทำงานร่วมกันระหว่างส่วนต่าง ๆ ของระบบ IoT ผ่านอินเทอร์เฟซที่ใช้งานง่ายและเชื่อถือได้ รวมถึงการรองรับเทคโนโลยีใหม่ เช่น การบูรณาการกับปัญญาประดิษฐ์ (AI) และระบบการวิเคราะห์ข้อมูลขั้นสูง (Advanced Analytics) เพื่อเพิ่มศักยภาพในการนำเสนอข้อมูลที่มีความหมาย นอกจากนี้ ชั้นนี้ยังช่วยให้ระบบ IoT สามารถสร้างความพึงพอใจและตอบโจทย์การใช้งานของผู้ใช้งานในภาคส่วนต่าง ๆ ได้อย่างมีประสิทธิภาพ

2.1.2) Edge Cloud Computing

Edge Cloud Computing เป็นแนวคิดที่นำการประมวลผลข้อมูลมาไว้ใกล้กับแหล่งข้อมูลหรืออุปกรณ์ IoT มากที่สุด เพื่อเพิ่มประสิทธิภาพและลดความล่าช้า (Latency) ในการประมวลผลข้อมูล โดยไม่จำเป็นต้องพึ่งพาการส่งข้อมูลทั้งหมดไปยังศูนย์ข้อมูลกลาง (Cloud) แนวคิดนี้ช่วยให้สามารถตอบสนองต่อเหตุการณ์ต่าง ๆ ได้อย่างรวดเร็วและทันท่วงที (Real-time Processing) ซึ่งเป็นประโยชน์ในงานที่ต้องการการตอบสนองแบบฉับไว เช่น การควบคุมกระบวนการผลิตในอุตสาหกรรม หรือการให้บริการในภาคสาธารณสุข

นอกจากนี้ Edge Cloud Computing ยังช่วยแก้ปัญหาในด้านการใช้ทรัพยากรเครือข่าย โดยลดความจำเป็นในการส่งข้อมูลจำนวนมากไปยังจุดศูนย์กลางการประมวลผล ทำให้สามารถลดค่าใช้จ่ายด้านแบนด์วิธและปรับปรุงความเสถียรของระบบในสถานการณ์ที่การเชื่อมต่ออินเทอร์เน็ตไม่เสถียร อีกทั้งยังช่วยให้มีความปลอดภัยของข้อมูลเพิ่มขึ้น เนื่องจากข้อมูลสามารถถูกประมวลผลและจัดเก็บในพื้นที่ใกล้กับแหล่งที่มาของข้อมูลโดยตรง ลดความเสี่ยงจากการถูกโจมตีในระหว่างการส่งข้อมูล และทำให้สามารถควบคุมการเข้าถึงข้อมูลได้ดีขึ้น

Edge Cloud Computing จึงถือเป็นแนวทางที่ตอบสนองต่อความต้องการที่หลากหลายในยุคดิจิทัล โดยเฉพาะในอุตสาหกรรมที่ต้องการการประมวลผลที่รวดเร็วและมีความปลอดภัยสูง ทั้งยังช่วยส่งเสริมการพัฒนาโครงสร้างพื้นฐานด้าน IoT ที่มีความยั่งยืนและเหมาะสมกับการใช้งานในอนาคต

คุณสมบัติหลักของ Edge Cloud Computing

1. Scalabiityl: รองรับการขยายตัวของระบบได้ตามความต้องการ และรวดเร็วไม่ว่าจะเป็นการเพิ่มจำนวนอุปกรณ์ IoT หรือการจัดการข้อมูลในระดับอุตสาหกรรมอย่างมีประสิทธิภาพ
2. Network Traffic Optimization: ช่วยลดการใช้แบนด์วิธในการส่งข้อมูลไปยังศูนย์กลาง โดยข้อมูลที่สำคัญจะถูกจัดการในพื้นที่ก่อนจาก ลดความแออัดบนเครือข่าย
3. Ultra-low Latency: ช่วยลดเวลาในการตอบสนองระหว่างเครื่องเซิร์ฟเวอร์และอุปกรณ์ IoT และเพิ่มความเร็วในการประมวลผลแบบเรียลไทม์
4. Edge Security: การรักษาความปลอดภัยที่ขอบเครือข่าย เพื่อป้องกันการโจมตีและการเข้าถึงข้อมูลที่ไม่พึงประสงค์จากอุปกรณ์ IoT

Openstack

OpenStack เป็นแพลตฟอร์มโครงสร้างพื้นฐานระบบคลาวด์แบบโอเพนซอร์ส (Open Source Cloud Infrastructure Platform) ซึ่งเกิดจากความร่วมมือของชุมชนเทคโนโลยีทั่วโลก และได้รับการสนับสนุนจากองค์กรไอทีชั้นนำหลากหลายสาขา แพลตฟอร์มนี้มีจุดมุ่งหมายเพื่อเปิดโอกาสให้ผู้ใช้งานสามารถสร้างระบบคลาวด์ของตนเองได้อย่างยืดหยุ่น ไม่ยึดติดกับผู้ให้บริการเพียงรายเดียว และสามารถปรับขยาย (Scale) หรือปรับแต่งทรัพยากรได้ง่ายตามความต้องการของงานและการเติบโตขององค์กร ความสามารถดังกล่าวทำให้ OpenStack กลายเป็นทางเลือกสำคัญสำหรับองค์กรที่ต้องการสร้างโครงสร้างพื้นฐานคลาวด์ส่วนตัว (Private Cloud) หรือแบบผสม (Hybrid Cloud) ที่สามารถควบคุมได้เองอย่างเต็มที่

จุดเด่นของ OpenStack อยู่ที่ความสามารถในการจัดสรรทรัพยากรคอมพิวติ้ง ที่เก็บข้อมูล (Storage) และระบบเครือข่าย (Networking) โดยอาศัยชุดบริการ (Services) ที่ทำงานร่วมกันผ่านมาตรฐานและ API ที่สอดคล้องกัน ผู้ดูแลระบบสามารถเลือกใช้เฉพาะโมดูลที่ต้องการ เพื่อให้ตอบสนองต่อสภาพแวดล้อมและลักษณะงานที่หลากหลาย นอกจากนี้การเป็นโอเพนซอร์สยังช่วยส่งเสริมให้เกิดนวัตกรรม การสนับสนุนชุมชนที่เข้มแข็ง ตลอดจนการปรับใช้เทคโนโลยีใหม่ ๆ อย่างรวดเร็ว ด้วยเหตุนี้ OpenStack จึงกลายเป็นฐานรากสำคัญที่สามารถรองรับการเติบโตของข้อมูลและการประมวลผลในยุคดิจิทัลได้เป็นอย่างดี

สถาปัตยกรรมและโมดูลหลักของ OpenStack ประกอบด้วยบริการโมดูลหลักหลายส่วน ซึ่งแต่ละส่วนได้รับการออกแบบมาเพื่อรองรับภารกิจเฉพาะ เช่น การบริหารจัดการคอมพิวติ้ง การจัดสรรสตอเรจ และการกำหนดเครือข่ายเสมือน เป็นต้น ผู้ดูแลระบบสามารถเลือกใช้เฉพาะโมดูลที่เหมาะสมกับบริบทการใช้งานได้อย่างยืดหยุ่น ส่งผลให้ระบบคลาวด์ที่สร้างขึ้นมีความเหมาะสม และปรับเปลี่ยนหรือขยายได้ตามความต้องการ

- **Nova (Compute Service): [1]**

Nova เป็นหนึ่งในบริการหลักที่สำคัญที่สุดในสถาปัตยกรรมของ OpenStack ซึ่งจะมีบทบาทหน้าที่เป็นจุดศูนย์กลางในการบริหารจัดการทรัพยากรภายในระบบคลาวด์ โดยสามารถรองรับการสร้างและควบคุมรายละเอียดภายใน instances ที่อาจอยู่ในรูปของ Virtual Machine หรือแม้กระทั่ง Container ได้อย่างมีประสิทธิภาพ ภายใต้การดำเนินงานของ Nova ผู้ให้บริการหรือผู้ดูแลระบบสามารถตั้งค่ากฎในการจัดสรรทรัพยากร เช่น หน่วนประมวลผล, หน่วยความจำ, และพื้นที่เก็บข้อมูล ให้แก่เวิร์กโหลดต่าง ๆ ตามความต้องการใช้งานและความสำคัญของงานได้อย่างยืดหยุ่น

ตารางที่ 4 องค์ประกอบและสถาปัตยกรรมภายในของ Nova

คอมโพเนนต์ (Component)	**บทบาทและหน้าที่ (Role & Responsibility)**
nova-api	ส่วนติดต่อ (Interface) ระหว่างผู้ใช้งานหรือระบบภายนอกกับ Nova ผ่าน RESTful API รองรับคำสั่งสร้าง, ลบ, ปรับขนาดอินสแตนซ์ เชื่อมโยงกับ Keystone เพื่อพิสูจน์ตัวตนและกำหนดสิทธิ์การเข้าถึง
nova-scheduler	กลไกตัดสินใจ (Decision Maker) ที่เลือกโฮสต์ (Compute Node) ที่เหมาะสมสำหรับการสร้างอินสแตนซ์ โดยอิงตามข้อมูลทรัพยากรที่ว่างอยู่ (Resource Availability) และนโยบายการกรองต่าง ๆ
nova-conductor	บริการตัวกลาง (Mediator) สำหรับประสานงานระหว่างส่วนประกอบของ Nova ลดการสื่อสารตรงระหว่าง nova-scheduler และ nova-compute และช่วยอัปเดตสถานะอินสแตนซ์ในฐานข้อมูล ทำให้ระบบยืดหยุ่นและปรับขยายง่าย
nova-compute	ส่วนประมวลผลหลัก (Compute Engine) บนแต่ละ Compute Node ทำหน้าที่รับคำสั่งจาก nova-scheduler เพื่อลงมือสร้าง, ลบ, หยุด หรือปรับขนาดอินสแตนซ์ จัดการวงจรการใช้งาน VM/Container ผ่าน Hypervisor หรือ Container Runtime
nova-placement	ระบบจัดสรรทรัพยากร (Resource Placement) ที่ติดตามการใช้ทรัพยากร เช่น CPU, RAM, Disk รวมถึงทรัพยากรเฉพาะทาง (GPU, FPGA) ให้ข้อมูลแก่ nova-scheduler เพื่อการตัดสินใจที่แม่นยำในการวางอินสแตนซ์ลงบนโฮสต์ที่เหมาะสม

สรุปแล้ว Nova จะเป็นหัวใจสำคัญใน OpenStack สำหรับการสร้างและบริหารจัดการทรัพยากรในรูปแบบ Virtual Machine หรือ Container มีความยืดหยุ่นในการขยายขนาดและปรับปรุงทรัพยากรตามความต้องการผันแปรของผู้ใช้งาน การสนับสนุนวงจรของอินสแตนซ์อย่างครบวงจร รวมถึงการทำงานร่วมกับบริการอื่น ๆ ภายใน OpenStack ช่วยให้ระบบคลาวด์มีความสมบูรณ์ ปรับใช้งานได้หลากหลาย และมีประสิทธิภาพทั้งด้านเทคนิคและเชิงธุรกิจ Nova ไม่เพียงตอบโจทย์ความสามารถในการสร้าง Compute Resource แบบ On-demand แต่ยังเปิดโอกาสให้ผู้ใช้งานนำเทคโนโลยีคลาวด์ไปประยุกต์กับเวิร์กโหลดทุกประเภท ขยายหรือย่อขนาดตามความจำเป็น ด้วยความมั่นคง ปลอดภัย และคล่องตัวที่เหนือกว่าการจัดการทรัพยากรแบบดั้งเดิม ส่งผลให้ Nova เป็นกลไกหลักที่ทำให้ OpenStack สามารถเป็นพื้นฐานในการสร้างสภาพแวดล้อมคลาวด์สมัยใหม่ รองรับการเจริญเติบโตขององค์กรและความต้องการการประมวลผลในโลกดิจิทัลได้อย่างแท้จริง

- **Neutron (Networking Service) [2]:**

Neutron เป็นบริการจัดการเครือข่ายภายใน OpenStack ที่มุ่งเน้นการสร้างและควบคุมเครือข่ายเสมือน (Virtual Networks) โดยไม่ต้องพึ่งพาอุปกรณ์เครือข่ายทางกายภาพโดยตรง ช่วยให้ผู้ใช้งานสามารถสร้างสภาวะแวดล้อมทางเครือข่ายที่เหมาะสมกับภาระงานและข้อกำหนดทางความปลอดภัยของตนเองได้อย่างยืดหยุ่น Neutron ช่วยสร้างรูปแบบเครือข่ายแยกอิสระสำหรับแต่ละโครงการ (Project) ซึ่งทำให้เกิดความเป็นส่วนตัวและความปลอดภัยของข้อมูล

ตารางที่ 5 คุณสมบัติของ Neutron

คุณสมบัติ (Feature)	**คำอธิบาย (Description)**
VM, Subnet	ผู้ใช้งานสามารถสร้างเครือข่าย (Network) และเครือข่ายย่อย (Subnet) สำหรับแต่ละโครงการได้ตามต้องการ โดยไม่ขึ้นกับโครงสร้างทางกายภาพ การกำหนด IP Address, DNS, และ DHCP จะถูกจัดการผ่าน Neutron ลดภาระงานการตั้งค่าแบบดั้งเดิม
External Network, NAT	การใช้งานเราเตอร์เสมือน (Virtual Router) ช่วยให้เชื่อมโยงเครือข่ายภายในโครงการกับเครือข่ายภายนอกหรืออินเทอร์เน็ตได้ง่าย รองรับการทำ NAT ทำให้อินสแตนซ์ภายในสามารถเข้าถึงภายนอกได้โดยไม่ต้องเปิดเผยโครงสร้างภายในทั้งหมด
Floating IP	ผู้ใช้งานสามารถมอบหมาย Floating IP ให้อินสแตนซ์ภายใน เพื่อให้เข้าถึงจากภายนอกได้อย่างยืดหยุ่น โดยไม่ต้องปรับเปลี่ยนการตั้งค่าเครือข่ายภายในใหม่เมื่อมีการย้ายอินสแตนซ์หรือปรับเปลี่ยนทรัพยากร
Security Groups	สามารถกำหนดกฎไฟร์วอลล์ระดับอินสแตนซ์ (Instance-level) เพื่อควบคุมการรับ-ส่งข้อมูลเข้าออกตามนโยบายความปลอดภัย กฎเหล่านี้ปรับเปลี่ยนได้ตามความต้องการ และรองรับการตั้งค่าเงื่อนไขเพื่อควบคุม Traffic ได้อย่างละเอียด
Project Isolation	แต่ละโครงการมีเครือข่ายของตนเอง ทำให้ข้อมูลและ Traffic ไม่ปะปนระหว่างโครงการอื่น เพิ่มความมั่นใจในความเป็นส่วนตัวและความปลอดภัยของข้อมูล

Neutron เป็นโครงสร้างพื้นฐานด้านเครือข่ายที่สำคัญใน OpenStack ทำให้การสร้าง ปรับเปลี่ยน และบริหารจัดการเครือข่ายเสมือนภายในคลาวด์เกิดขึ้นได้อย่างง่ายดาย ด้วยการผนวกรวมกับแนวคิดการแยกเครือข่ายตามโครงการ การจัดสรร Floating IP และการกำหนดนโยบายความปลอดภัยที่ยืดหยุ่น ผู้ใช้งานสามารถออกแบบสภาพแวดล้อมเครือข่ายให้สอดคล้องกับความต้องการของตนได้อย่างมีประสิทธิภาพ ตามที่อธิบายไว้ในเอกสารของ HPC Cambridge ผู้ดูแลระบบและนักวิจัยสามารถใช้ Neutron ในการสร้างระบบเครือข่ายที่มั่นคง ปลอดภัย และตอบสนองต่อการเติบโตและการเปลี่ยนแปลงของงานในสภาพแวดล้อมคลาวด์ได้เป็นอย่างดี

- **Cinder (Block Storage Service):**

Cinder Zเป็นบริการบริหารจัดการพื้นที่จัดเก็บข้อมูลแบบบล็อก (Block Storage) ภายใน OpenStack โดยมีบทบาทหน้าที่หลักในการสร้างจัดการและนำเสนอ Volume ให้กับอินสแตนซ์ ซึ่งทำหน้าที่เสมือนฮาร์ดดิสก์เสมือน (Virtual Hard Disk) ที่สามารถแนบเข้ากับอินสแตนซ์ได้ตามความต้องการ จุดเด่นของ Cinder คือการมอบพื้นที่เก็บข้อมูลที่มีความคงทนถาวร (Persistent) ต่ออายุการใช้งานของอินสแตนซ์ กล่าวคือแม้ภายหลังจากที่อินสแตนซ์ถูกลบไปแล้ว ข้อมูลที่เก็บไว้บน Volume ยังคงอยู่และสามารถนำไปใช้งานกับอินสแตนซ์อื่นหรือดำเนินการจัดเก็บเพื่ออ้างอิงภายหลังได้

ตารางที่ 6 คุณสมบัติของ Cinder

คุณสมบัติ (Feature)	**คำอธิบาย (Description)**
Persistent Storage	มอบพื้นที่เก็บข้อมูลแบบถาวรที่ยังคงอยู่แม้อินสแตนซ์ถูกลบ ช่วยให้ข้อมูลสำคัญไม่สูญหาย
Volume Attachment/Detachment	ผู้ใช้สามารถแนบ (Attach) หรือถอด (Detach) Volume เข้ากับอินสแตนซ์ได้อย่างยืดหยุ่นตามความต้องการของงาน
Snapshot & Backup	รองรับการสร้างสแนปช็อต (Snapshot) เพื่อบันทึกสภาพข้อมูลปัจจุบัน และสามารถสำรองข้อมูล (Backup) ไปยังที่จัดเก็บภายนอกเพื่อรองรับการกู้คืนข้อมูล
Volume Resizin	สามารถปรับขนาด Volume เพื่อรองรับปริมาณข้อมูลที่เพิ่มขึ้นโดยไม่จำเป็นต้องสร้าง Volume ใหม่
Integration with Other Services	ทำงานร่วมกับบริการอื่นใน OpenStack เช่น Nova และ Glance ทำให้สามารถสร้าง Volume จากอิมเมจระบบปฏิบัติการและแนบเข้าสู่อินสแตนซ์เพื่อใช้งานได้ทันที

Cinder มีบทบาทสำคัญในระบบ OpenStack ด้านการจัดเก็บข้อมูล ช่วยให้ผู้ใช้งานสามารถควบคุม รักษาความสอดคล้องในการบริหารทรัพยากรที่เกี่ยวข้องกับข้อมูลได้อย่างสะดวกด้วยการรองรับการสร้าง ปรับขนาดและสำรองข้อมูลบน Volume ผู้ดูแลระบบและนักวิจัยสามารถบริหารจัดเก็บข้อมูลภายในคลาวด์ให้เกิดประโยชน์สูงสุดและรองรับการขยายตัวของงานในอนาคตได้อย่างมีประสิทธิภาพ

- **Swift (Object Storage Service): [3]**

Swift เป็นบริการเก็บข้อมูลในรูปแบบ Object Storage ของ OpenStack ซึ่งแตกต่างจาก Block Storage (Cinder) ตรงที่ Swift ไม่ได้ให้ข้อมูลเป็นบล็อก แต่จะเก็บข้อมูลในรูปแบบวัตถุ (Object) และจัดการด้วยคีย์ (Key) หรือชื่อไฟล์ แทนที่จะเข้าถึงข้อมูลด้วยที่อยู่บล็อกหรือโครงสร้างไฟล์ที่ซับซ้อน Swift ถูกออกแบบให้มีความทนทานต่อข้อผิดพลาดและขยายได้ง่าย รองรับการเก็บข้อมูลจำนวนมากโดยไม่สูญเสียประสิทธิภาพ นอกจากนี้ Swift ยังสนับสนุนการกระจายข้อมูลข้ามโหนดและข้ามพื้นที่ทางภูมิศาสตร์ เพื่อเพิ่มความทนทานและความเชื่อถือได้

ตารางที่ 7 คุณสมบัติของ Swift

คุณสมบัติ (Feature)	**คำอธิบาย (Description)**
Object-based Storage	จัดเก็บข้อมูลเป็น Object พร้อม Metadata ไม่ต้องพึ่งพา Directory หรือ Filesystem แบบดั้งเดิม ลดความซับซ้อนในการบริหาร
High Durability & Availability	กระจายข้อมูลและ Replica ข้ามโหนดและโซน เพื่อลดความเสี่ยงในการสูญเสียข้อมูลและรักษาการให้บริการตลอดเวลา
No Centralized Metadata Server	ไม่ใช้เซิร์ฟเวอร์ Metadata ส่วนกลาง ลดความเสี่ยงจากจุดล้มเหลวเดียว และเพิ่มความยืดหยุ่นในการปรับแต่งระบบ
Scalability	เพิ่มทรัพยากร (เช่น Storage Nodes) ได้ง่าย เพื่อรองรับการเติบโตของข้อมูลขนาดใหญ่ โดยไม่รบกวนการทำงานเดิม
Integration with Other Services	ทำงานร่วมกับ Glance และบริการอื่น ๆ ใน OpenStack ช่วยให้การจัดเก็บอิมเมจ การสำรองข้อมูล และการแชร์ข้อมูลภายในคลาวด์เป็นไปอย่างสะดวก

Swift เป็นบริการ Object Storage ที่ให้ความทนทานต่อข้อผิดพลาด ความยืดหยุ่นในการปรับขยายและความสะดวกในการจัดการข้อมูล ทำให้เหมาะสำหรับการเก็บข้อมูลปริมาณมากที่ต้องการความเชื่อถือได้สูง และไม่ต้องการโครงสร้างไฟล์ที่ซับซ้อน การทำงานร่วมกับบริการอื่น ๆ ของ OpenStack ยังช่วยให้ Swift เป็นส่วนหนึ่งของระบบคลาวด์ที่ครอบคลุมทุกด้านของการจัดการทรัพยากรและข้อมูลในสภาพแวดล้อมการประมวลผลสมัยใหม่

- **Glance (Image Service): [4]**

Glance เป็นบริการใน OpenStack สำหรับการบริหารจัดการอิมเมจ (Image) ของระบบปฏิบัติการที่ใช้สร้างอินสแตนซ์ ไม่ว่าจะเป็น Linux, Windows หรือระบบปฏิบัติการเฉพาะทาง อิมเมจเหล่านี้จะถูกเก็บไว้ในที่จัดเก็บซึ่ง Glance รองรับหลายรูปแบบ เช่น Object Storage (Swift) หรือระบบจัดเก็บภายนอกอื่น ๆ เป้าหมายหลักของ Glance คือการมอบกระบวนการมาตรฐานในการเก็บรวบรวมแบ่งปัน และจัดการอิมเมจอย่างมีประสิทธิภาพและปลอดภัย

ตารางที่ 8 คุณสมบัติของ Swift

คุณสมบัติ (Feature)	คำอธิบาย (Description)
Image Registry & Retrieval	จัดเก็บและเรียกใช้อิมเมจจากศูนย์กลาง ช่วยให้ค้นหา อัปโหลด และดึงอิมเมจมาใช้ได้ง่าย
Multiple Formats & Backends	รองรับอิมเมจหลากหลายฟอร์แมตและ Backend จัดเก็บข้อมูลหลายรูปแบบเพื่อความยืดหยุ่นสูงสุด
Image Metadata & Properties	กำหนด Metadata สำหรับอิมเมจเพื่ออธิบายคุณลักษณะพิเศษ ช่วยเลือกใช้อิมเมจที่เหมาะสม
Integration with Other Services	ทำงานร่วมกับ Nova, Cinder และ Horizon ทำให้การสร้างอินสแตนซ์และจัดการอิมเมจเป็นไปอย่างลื่นไหล
Security & Access Control	ควบคุมสิทธิ์การเข้าถึงอิมเมจผ่าน Keystone และสามารถอัปเดตหรือลบอิมเมจที่ไม่จำเป็นเพื่อความปลอดภัย

Glance เป็นส่วนสำคัญที่เสริมสร้างความสมบูรณ์ให้กับ OpenStack ทำให้การเก็บรวบรวมและบริหารอิมเมจระบบปฏิบัติการเป็นไปได้อย่างมีมาตรฐาน ยืดหยุ่น และปรับเปลี่ยนตามความต้องการ ผู้ดูแลระบบและผู้ใช้งานสามารถเชื่อมต่อ Glance เข้ากับบริการอื่น ๆ ใน OpenStack เพื่อสร้างเวิร์กโหลดที่เหมาะสม ช่วยประหยัดเวลา ลดความซับซ้อน และเพิ่มความมั่นคงปลอดภัยในการบริหารทรัพยากรภายในคลาวด์

- **Keystone (Identity Service): [5]**

Keystone เป็นหนึ่งในองค์ประกอบหลักที่สำคัญของสถาปัตยกรรม OpenStack ทำหน้าที่เป็นศูนย์กลางในการบริหารจัดการตัวตน (Identity Management) และการควบคุมการเข้าถึงทรัพยากร (Access Management) ภายในสภาพแวดล้อมคลาวด์ โดยการทำงานของ Keystone เน้นการให้บริการด้านการพิสูจน์ตัวตน (Authentication) การกำหนดสิทธิ์ (Authorization) และการออกโทเคน (Token) ให้แก่ผู้ใช้งาน บริการภายใน และแอปพลิเคชันหลากหลายรูปแบบที่ปฏิบัติการอยู่บน OpenStack

จุดแข็งสำคัญของ Keystone คือความสามารถในการเชื่อมต่อกับระบบ Identity ภายนอก เช่น LDAP, Active Directory หรือระบบ Single Sign-On (SSO) ซึ่งเอื้อให้ผู้ดูแลระบบสามารถนำบัญชีผู้ใช้ที่มีอยู่แล้วมาใช้ใน OpenStack ได้อย่างราบรื่น ลดภาระการบริหารหลายบัญชีผู้ใช้ และเพิ่มความยืดหยุ่นในการปรับใช้งาน นอกจากนี้ Keystone ยังสนับสนุนการกำหนดโดเมน (Domain) และโครงการ (Project/Tenant) เพื่อจัดสรรทรัพยากรและผู้ใช้งานให้เป็นระเบียบและเป็นระบบ สามารถกำหนดบทบาท (Role) เพื่อรองรับการมอบหมายหน้าที่ ความรับผิดชอบ ตลอดจนสิทธิ์การเข้าถึงบริการและทรัพยากรที่เหมาะสมกับแต่ละบุคคลหรือกลุ่มผู้ใช้งาน

ภายใต้กลไก Role-Based Access Control (RBAC) Keystone ทำให้การกำหนดสิทธิ์การเข้าถึงทรัพยากรเป็นเรื่องง่ายขึ้น ผู้ดูแลระบบสามารถกำหนดนโยบายความปลอดภัยและการเข้าถึงที่ชัดเจน สอดคล้องกับข้อกำหนดภายในองค์กรหรือมาตรฐานสากล นอกจากนี้ การใช้งาน Keystone ยังเอื้อให้ผู้ใช้งานและบริการภายใน OpenStack สามารถเรียกใช้ทรัพยากรต่าง ๆ ผ่านโทเคนที่ออกโดย Keystone หลังการพิสูจน์ตัวตน ซึ่งช่วยลดความซับซ้อนในการส่งข้อมูลรับรองซ้ำซ้อน (Credentials) ไปยังบริการอื่น ๆ ในระบบ โดยที่การเรียกใช้ทรัพยากรและบริการจะดำเนินการผ่านAPI ทำให้สามารถพัฒนาเครื่องมืออัตโนมัติหรือสคริปต์เพิ่มเติมได้ตามต้องการ ด้วยเหตุนี้ Keystone จึงเป็นองค์ประกอบพื้นฐานที่ทำให้ OpenStack มีความสามารถในการรองรับผู้ใช้งานและทรัพยากรจำนวนมากได้อย่างมีประสิทธิภาพ มั่นคงปลอดภัย และปรับขนาดได้ตามความต้องการขององค์กร ทำให้การบริหารจัดการคลาวด์ที่มีหลายโดเมน หลายโครงการ และผู้ใช้หลายประเภทเป็นไปได้อย่างง่ายดายและคุ้มค่าในการปฏิบัติงานจริง

ตารางที่ 9 คุณสมบัติของ Keystone

คุณสมบัติ (Feature)	**คำอธิบาย (Description)**
Identity Management	จัดการข้อมูลผู้ใช้ โครงการ (Project/Tenant) และบทบาท (Role) ทำให้การบริหารจัดการผู้ใช้เป็นระบบ
Authentication	พิสูจน์ตัวตนผู้ใช้ด้วยกลไกที่หลากหลาย และออกโทเคนเพื่อใช้เข้าถึงบริการใน OpenStack
Authorization	กำหนดสิทธิ์การเข้าถึงทรัพยากรตามบทบาทและนโยบายที่กำหนด ปรับใช้ได้ตามโครงสร้างองค์กร
Service Catalog	จัดเก็บรายการบริการ (Services) ในระบบ ช่วยให้ผู้ใช้ทราบว่ามีบริการใดบ้างและเข้าถึงได้อย่างไร
Multi-Domain Support	รองรับการบริหารจัดการผู้ใช้และทรัพยากรในหลายโดเมน (Domain) เพื่อแยกการควบคุมให้อยู่ในขอบเขตที่ชัดเจน

Keystone สามารถเชื่อมโยงกับระบบยืนยันตัวตนภายนอก เช่น LDAP หรือ Active Directory และรองรับ Single Sign-On (SSO) ทำให้บัญชีผู้ใช้เดิมใช้งานใน OpenStack ได้ทันที อีกทั้งยังใช้ Role-Based Access Control เพื่อกำหนดสิทธิ์การเข้าถึง เช่น การแยกผู้ดูแลระบบออกจากผู้ใช้งานทั่วไป พร้อมปรับนโยบายให้มีความสอดคล้องกับมาตรฐานความปลอดภัย Keystone เป็นศูนย์กลางในการจัดการตัวตนและการเข้าถึงทรัพยากรใน OpenStack

โดยมีบทบาทในพิสูจน์ตัวตน กำหนดสิทธิ์ และออกโทเคนให้แก่ผู้ใช้งาน อีกทั้งรองรับ Multi-Domain ทำให้การบริหารจัดการผู้ใช้และทรัพยากรมีความยืดหยุ่น ปลอดภัย และตรงตามความต้องการขององค์กร

- **Horizon (Dashboard): [6]**

Horizon เป็นเว็บอินเทอร์เฟซ (Web-based UI) ที่ทำหน้าที่เป็น Front-end สำหรับผู้ใช้งานและผู้ดูแลระบบในการบริหารจัดการทรัพยากรและบริการภายใน OpenStack ช่วยให้การควบคุม การเฝ้าติดตาม และการดำเนินการทางด้านโครงสร้างพื้นฐานคลาวด์เป็นไปได้อย่างสะดวก รวดเร็ว และเป็นมิตรต่อผู้ใช้งาน แม้ผู้ใช้จะไม่มีทักษะทางเทคนิคเชิงลึกก็ตาม

ตารางที่ 10 คุณสมบัติของ Keystone

คุณสมบัติ (Feature)	**คำอธิบาย (Description)**
Web-based Interface	เข้าถึงผ่านเว็บเบราว์เซอร์ ไม่จำเป็นต้องติดตั้งซอฟต์แวร์เพิ่มเติม
Integration with Core Services	ทำงานร่วมกับ Nova, Neutron, Cinder, Swift, Glance และ Keystone ในอินเทอร์เฟซเดียว
RBAC via Keystone	แสดงเมนูและทรัพยากรตามบทบาทและสิทธิ์ของผู้ใช้ที่ผ่านการพิสูจน์ตัวตนด้วย Keystone
Project & Domain Management	สร้างและจัดการโครงการ รวมถึงกำหนด Quota และมอบหมายบทบาทให้ผู้ใช้งานได้ง่าย
Customization & Plugins	ปรับแต่งธีม เพิ่มปลั๊กอิน หรือเชื่อมต่อเครื่องมือเสริมตามความต้องการขององค์กร

Horizon ทำหน้าที่เป็น Dashboard ที่ช่วยให้ผู้ใช้งานและผู้ดูแลระบบสามารถจัดการทรัพยากรและบริการภายใน OpenStack ได้อย่างสะดวกและมีประสิทธิภาพ โดยไม่ต้องอาศัยคำสั่งผ่านบรรทัดคำสั่ง (CLI) หรือ API โดยตรง คุณสมบัติด้านการผสานรวมกับบริการหลัก การควบคุมการเข้าถึงตามบทบาท ความสามารถในการปรับแต่งและขยายฟังก์ชันตลอดจนการบริหารโครงการและโดเมน ล้วนช่วยเพิ่มประสิทธิภาพในการดำเนินงาน และทำให้ Horizon เป็นส่วนสำคัญในการสร้างประสบการณ์ที่ดีแก่ผู้ใช้ OpenStack ในสภาพแวดล้อมคลาวด์สมัยใหม่

จากที่ได้อธิบายในส่วนที่ผ่านมา OpenStack ได้รับการพัฒนาให้เป็นแพลตฟอร์มโครงสร้างพื้นฐานระบบคลาวด์แบบโอเพนซอร์สที่มีความยืดหยุ่นและปรับขยายได้ง่าย องค์ประกอบหลักต่าง ๆ เช่น Nova, Neutron, Cinder, Swift, Glance, Keystone และ Horizon ล้วนทำงานร่วมกันเพื่อสนับสนุนการบริหารจัดการทรัพยากรคอมพิวติ้ง ที่เก็บข้อมูล ระบบเครือข่าย อิมเมจระบบปฏิบัติการ ตลอดจนการจัดการตัวตนและสิทธิ์การเข้าถึง ตลอดจนการอำนวยความสะดวกในการใช้งานผ่านอินเทอร์เฟซแบบเว็บ

ความสามารถในการเลือกใช้เฉพาะโมดูลที่ตรงกับความต้องการ การสนับสนุนมาตรฐานและ API ที่สอดคล้องกัน รวมถึงความเป็นโอเพนซอร์สที่มีชุมชนเข้มแข็งและมีนวัตกรรมต่อเนื่อง ทำให้ OpenStack เป็นโครงสร้างพื้นฐานคลาวด์ที่สามารถปรับให้เข้ากับสภาพแวดล้อมการใช้งานได้หลากหลาย ทั้งในรูปแบบ Private Cloud หรือ Hybrid Cloud องค์กรสามารถควบคุมทรัพยากรและพัฒนาโซลูชันตามความต้องการเฉพาะทาง ลดการพึ่งพาผู้ให้บริการภายนอก และรองรับการเติบโตของข้อมูลและการประมวลผลในยุคดิจิทัลได้อย่างมั่นใจ

ด้วยคุณสมบัติเหล่านี้ OpenStack จึงกลายเป็นทางเลือกสำคัญสำหรับองค์กรและนักวิจัยที่ต้องการสร้าง สร้างสรรค์ และบริหารจัดการโครงสร้างพื้นฐานคลาวด์อย่างคล่องตัว มีประสิทธิภาพ และพร้อมรองรับความเปลี่ยนแปลงของเทคโนโลยีและตลาดในอนาคต ต่อจากนี้ รายงานจะนำเสนอเนื้อหาในมิติเชิงลึกหรือประเด็นที่เกี่ยวข้องเพิ่มเติม เพื่อให้ผู้อ่านสามารถเชื่อมโยงและประยุกต์ใช้ข้อมูลในบริบทที่กว้างขวางยิ่งขึ้น

StarlingX

StarlingX เป็นโครงการซอฟต์แวร์โอเพนซอร์สที่ถูกพัฒนาขึ้นเพื่อตอบสนองความต้องการของระบบ Edge Cloud Computing โดยเฉพาะ ซึ่งระบบ Edge Cloud Computing เป็นหนึ่งในหัวใจสำคัญของการเปลี่ยนแปลงยุคดิจิทัล (Digital Transformation) และเป็นรากฐานสำคัญของเทคโนโลยีในยุคอุตสาหกรรม 4.0 เช่น ระบบ Internet of Things (IoT), ปัญญาประดิษฐ์ (AI), และระบบอัจฉริยะอื่น ๆ ที่ต้องการการตอบสนองข้อมูลแบบเรียลไทม์ StarlingX พัฒนาขึ้นเพื่อนำเสนอโซลูชันที่สามารถประมวลผลและจัดเก็บข้อมูลใกล้กับแหล่งที่มาของข้อมูล เช่น อุปกรณ์ IoT หรือเซ็นเซอร์ในพื้นที่การใช้งานจริง เพื่อลดความล่าช้า (Latency) ในการสื่อสารกับศูนย์ข้อมูล (Data Center) ขนาดใหญ่ที่อยู่ห่างไกล ระบบนี้ช่วยเพิ่มความเร็วในการตอบสนองและลดปริมาณข้อมูลที่ต้องส่งกลับไปยังศูนย์กลาง

ตารางที่ 11 คุณสมบัติหลักของ StarlingX

คุณสมบัติ (Feature)	คำอธิบาย (Description)
Distributed Computing	รองรับการกระจายงานประมวลผลไปยังโหนด (Nodes) หลายตำแหน่ง เพื่อเพิ่มประสิทธิภาพและลดเวลาแฝง
High Availability (HA)	ระบบสำรองและตรวจสอบสถานะที่ช่วยให้มั่นใจว่าการทำงานของระบบต่อเนื่อง แม้เกิดความผิดพลาด
Containerized Workloads	รองรับ Kubernetes และการบริหารจัดการคอนเทนเนอร์สำหรับแอปพลิเคชันที่ต้องการการปรับตัวอย่างรวดเร็ว
Security	มาตรการความปลอดภัยที่ครอบคลุมตั้งแต่การเข้ารหัสข้อมูลไปจนถึงการตรวจสอบตัวตนและป้องกันภัยคุกคาม
Scalability	รองรับการปรับขยายระบบตั้งแต่ระดับเล็กไปจนถึงเครือข่ายขนาดใหญ่
Real-Time Performance	รองรับแอปพลิเคชันที่ต้องการการตอบสนองในระดับมิลลิวินาที
Fault Management	การตรวจสอบและจัดการความผิดพลาดในระบบเพื่อป้องกันปัญหาที่อาจเกิดขึ้น
Multi-Architecture Support	รองรับการใช้งานบนทั้งสถาปัตยกรรม x86 และ ARM

โดยตัวอย่างการใช้งานจริงของ StarlingX มีดังนี้

- **การประมวลผลภาพและการจดจำใบหน้า (AI, Face Recognition) บน Edge Cloud**
  ในงาน StarlingX Deep Dive Meetup (2019) ที่ปักกิ่ง มีการสาธิตการใช้ StarlingX รุ่น 2.0 ซึ่งผสานรวม OpenStack และ Kubernetes เพื่อรองรับการประมวลผล AI บน Edge Cloud โดยทำการฝึกฝนโมเดลที่คลาวด์ส่วนกลางและส่งโมเดลไปยัง Edge เพื่อทำการจดจำใบหน้าแบบเรียลไทม์
- **การรวมงาน Edge สำหรับบริการคลาวด์ที่เชื่อมโยง โดยใช้ ACRN Hypervisor และ StarlingX**
  มีการสาธิตการใช้ ACRN ซึ่งเป็นโอเพนซอร์สไฮเปอร์ไวเซอร์ที่ยืดหยุ่นและน้ำหนักเบา ร่วมกับ StarlingX เพื่อรันเวอร์ชวลแมชชีนที่มีระบบปฏิบัติการต่าง ๆ เช่น Clear Linux, Preempt-RT Yocto หรือ Windows
- **Android ในคอนเทนเนอร์ (AIC) บน StarlingX**
  การสาธิตนี้ใช้ Kata Containers ร่วมกับ StarlingX เพื่อรันระบบปฏิบัติการ Android ภายในคอนเทนเนอร์บนแพลตฟอร์ม StarlingX ซึ่งเหมาะสำหรับการใช้งานใน Edge Computing เช่น การขับขี่อัตโนมัติและอุตสาหกรรม

OpenSDN [7]

OpenSDN เป็น Software Define Network (SDN) สำหรับการใช้งานในระดับ enterprise, telco, cloud service provider เป็น Open Source software ที่พัฒนาเพื่อมาใช้งานร่วมกับ OpenStack และ Kubernetes โดยออกแบบมาเพื่อให้รองรับการใช้งานแบบขยายตัวสูง (highly scalable) และการใช้งานขนาดใหญ่ และติดตั้งบนเครื่องแม่ขายแบบ x86 ได้ทุกยี่ห้อ ทำให้ไม่เกิดการผูกขาด (vendor lock-in) ซึ่งมีความสามารถหลักดังต่อไปนี้

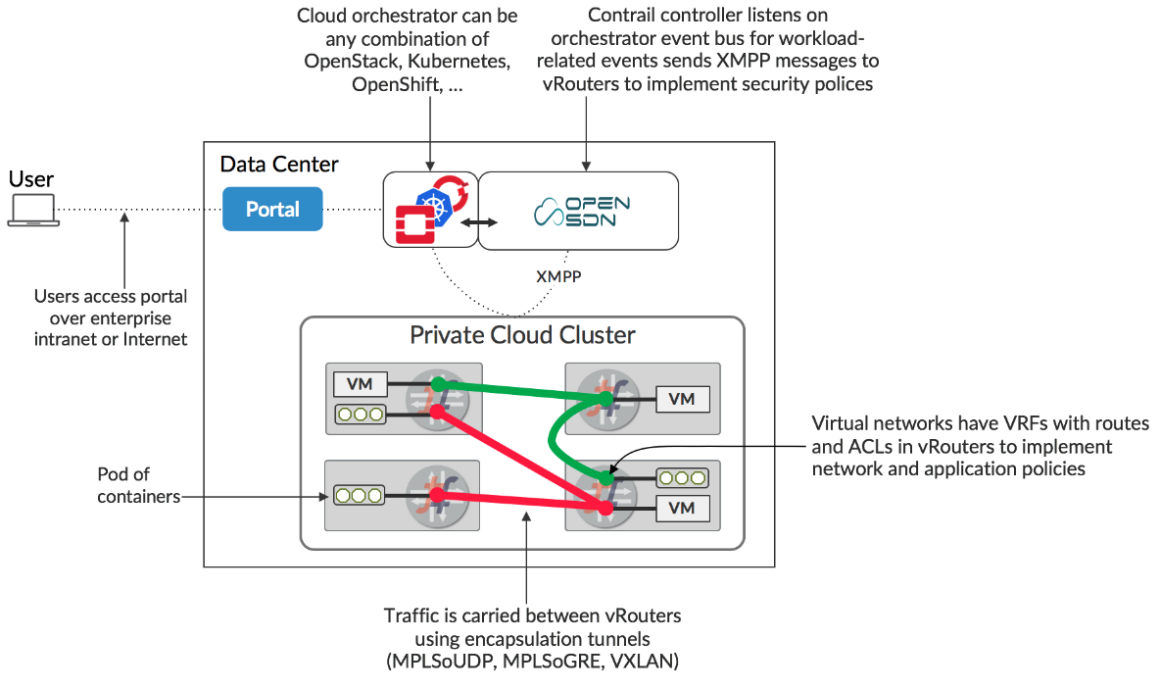
- Highly scalable, multi-tenant networking
- Multi-tenant IP address management
- DHCP, ARP proxies to avoid flooding into networks
- Efficient edge replication for broadcast and multicast traffic
- Local, per-tenant DNS resolution
- Distributed firewall with access control lists
- Application-based security policies
- Distributed load balancing across hosts
- Network address translation (1:1 floating IPs and distributed SNAT)
- Service chaining with virtual network functions
- Dual stack IPv4 and IPv6
- BGP peering with gateway routers
- BGP as a Service (BGPaaS) for distribution of routes between privately managed customer networks and service provider networks

การทำงานของ OpenSDN

OpenSDN จะทำงานแค่ตัวเองไม่ได้จะต้องมีการใช้งานร่วมกับ OpenStack หรือ Kubernetes ซึ่งต้องเชื่อมต่อ controller เข้ากับ controller ของ OpenStack และ Kubernetes โดย OpenSDN จะทำหน้าที่จัดการส่วนของเครือข่าย เมื่อมีการสร้างเครื่องแม่ข่ายเสมือน (Virtual Machain) แล้ว OpenSDN จะทำหน้าที่กำหนด IP address และเชื่อมต่อเครื่องแม่ข่ายเสมือนให้สามารถสื่อสารกันได้รวมถึงกำหนดเรื่อความปลอดภัย (security policy) สำหรับการสื่อสารระหว่างเครื่องแม่ข่ายเสมือนด้วย โดยจะมีการทำงานหลัก 2 ส่วนคือ

- OpenSDN controller คือตัวควบคุมและจัดการเครือข่ายเสมือน (virtual network), Routing, และการควบคุมการสื่อสาร (network policy) ให้กับ VM

- OpenSDN vRouter คือตัวจัดการสำหรับการส่งข้อมูล (packet forwarding) สำหรับ VM โดยทีจะทำงานในระดับของ compute node (Hypervisor) และจะค่อยรับข้อมูลการจัดการต่างๆ จาก OpenSDN controller เพื่อตัดสินใจในการส่งข้อมูล (packet) ต่างๆ

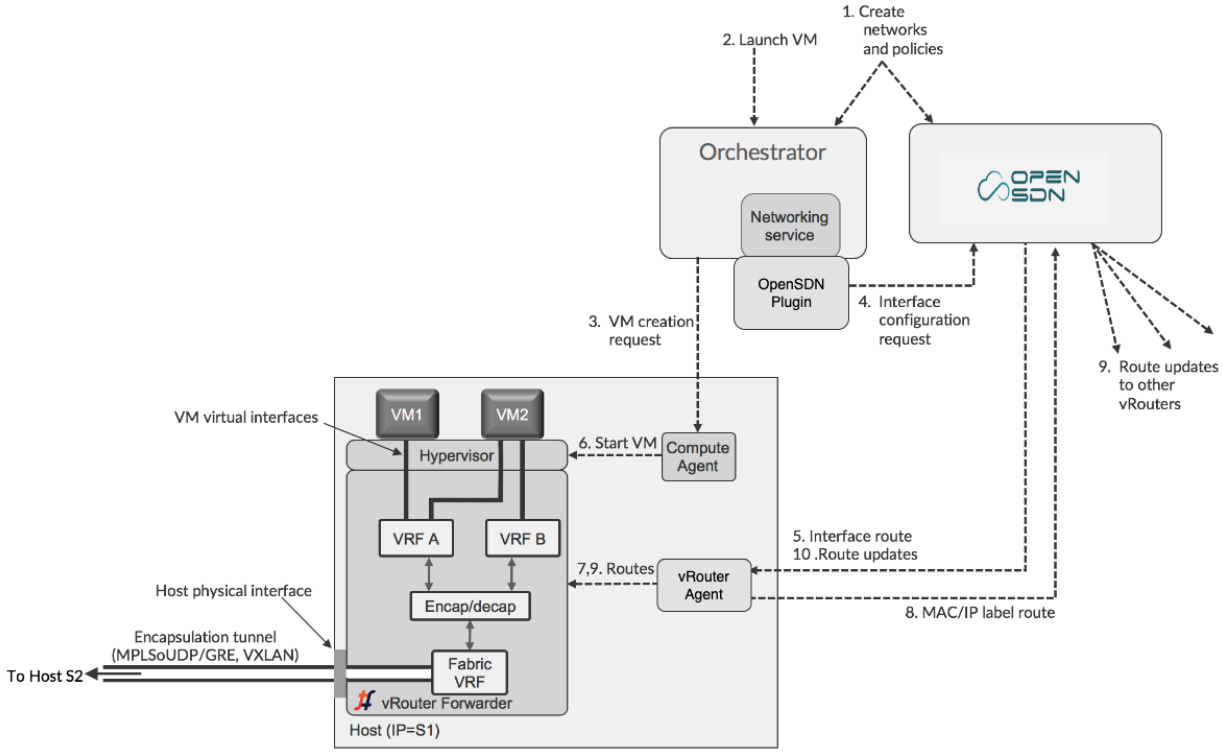


รูปที่ 1 โครงสร้างของ OpenSDN

OpenSDN controller เมื่อทำงานร่วมกับ OpenStack จะมีการเชื่อมต่อการทำงานกับ Neutron API ที่ทำหน้าที่จัดการเครือข่ายเสมือน และหากทำงานร่วมกับ Kubernetes จะมีการทำงานร่วมกับ kube-network-manager และ container network interface (CNI) โดยจะทำงานผ่าน Kubernetes K8S API

OpenSDN vRouter เมื่อทำงานร่วมกับ OpenStack จะเป็นการทำงานแทนที่ Linux bridge และ IP tables หรือ Open vSwtich ในแต่ละ compute node ซึ่ง OpenSDN controller จะเป็นคนกำหนดค่าต่างๆ ให้กับเครือข่ายร่วมกับควบคุมการสื่อสาร (security policies) ด้วย

เมื่อมีการส่งข้อมูล (packet) จาก VM ที่อยู่ compute node หนึ่งไปยังอีก VM ที่อยู่ compute node หนึ่ง vRouter จะทำการเข้าห่อหุ้มข้อมูล (encapsulation) ให้อยู่ในรูปแบบของ MPLS over UDP/GRE หรือ VXLAN โดยการห่อหุ้มข้อมูลจะต้องกำหนด IP address ปลายทางของ vRouter (compute node) ที่จะส่งข้อมูลไป โดย OpenSDN controller จะทำการกำหนด virtual router forwarding (VRF) ให้กับแต่ละ virtual network (VN) ในแต่ละ vRouter (compute node) ในกรณีที่ VM อยู่ใน virtual network เดียวกัน โดยการ OpenSDN controller จะกำหนดค่าต่างๆ ให้กับ vRouter ผ่าน Extensible Messaging and Presence Protocol (XMPP)

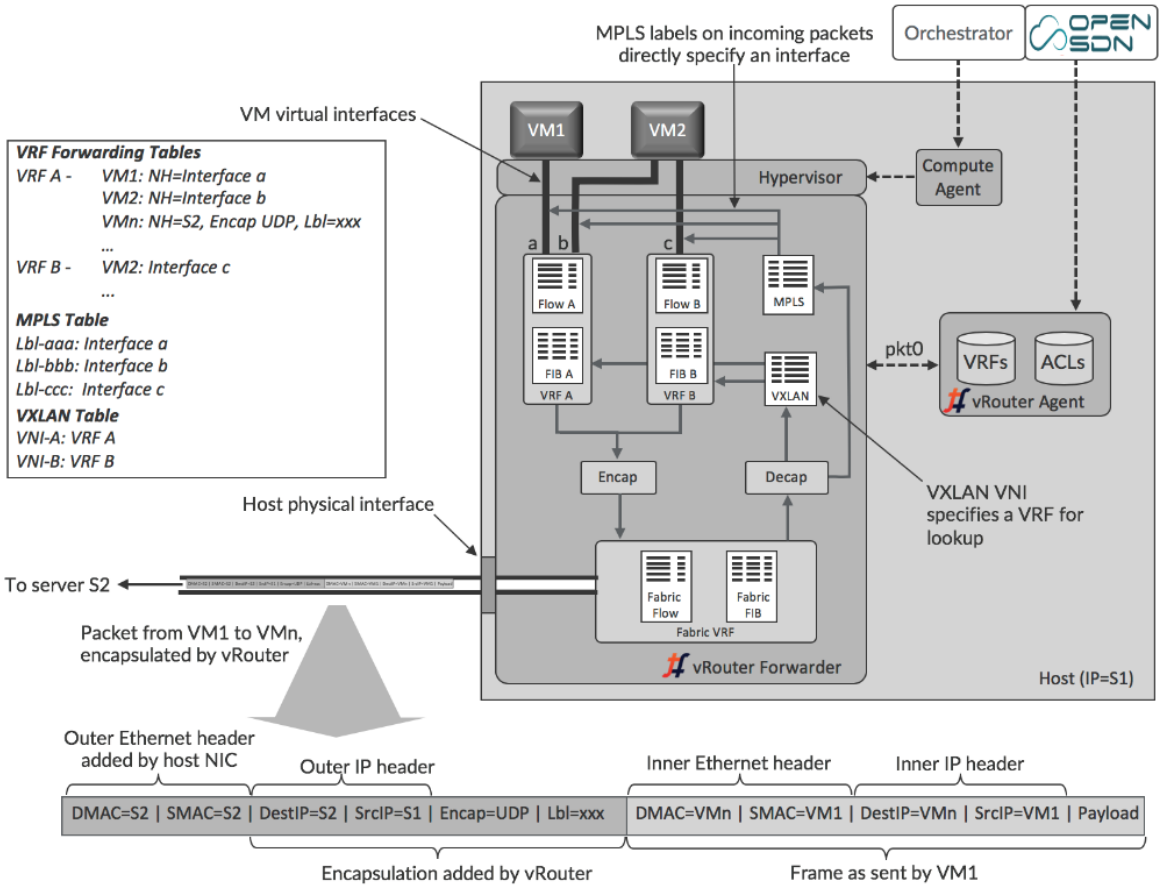


รูปที่ 2 การสื่อสารระหว่าง OpenSDN กับหน้าบริหารจัดการ (Orchestrator)

จากรูปจะมีรายละเอียดการทำงานดังนี้

1. การกำหนดหรือสร้าง virtual networks และ network policies ผ่าน Web User Interface (Web UI) จะเป็นการกำหนดให้มีการสร้าง IP address ที่จะกำหนดให้กับ VM สำหรับการใช้งาน
2. ทำการสร้าง VM และกำหนดให้อยู่ใน virtual network ที่ผู้ใช้สร้างมาในขั้นตอนก่อนหน้า
3. เมื่อมีการสร้าง VM OpenStack Nova จะเป็นผู้ทำหน้าที่โดยจะเลือก compute node ที่มีทรัพยากรเพียงพอต่อการสร้าง VM และจะทำการสร้าง VM จาก image ที่ผู้ใช้เลือก
4. OpenSDN plugin จะได้รับข้อมูลว่ามีการสร้าง VM ใหม่เกิดขึ้น และทำการแปลงคำสั่งให้อยู่ในรูปแบบ REST API และส่งคำสั่ง ไปยัง OpenSDN Controller
5. OpenSDN controller ทำการกำหนด VRF ให้กับ virtual network สำหรับ VM ที่จะทำการสร้างใน compute node เครื่องดังกล่าวผ่าน vRouter Agent สำหรับในกรณีที่ virtual network นั้นมีการใช้งานครั้งแรกกับ VM และเชื่อมต่อกับ virtual machain interface (VMI)
6. VRouter Agent ทำการเริ่มการทำงานของ VM ซึ่งจะมีการร้องขอ IP address ผ่าน DHCP ของ virtual network ดังกล่าว ซึ่ง vRouter ทำหน้าที่เสมือน DHCP proxies (คนกลาง) ในการทำงาน โดยจะกำหนด IP address, default gateway และ DNS server ให้กับ VM
7. ในครั้งแรกที่ Interface ของ VM ทำงาน vRouter จะทำการเพิ่ม ip address ของ VM ใน route table ของ virtual network และทำการกำหนด next-hop (ค่าสำหรับการส่งข้อมูล) เป็นค่าของ VM virtual interface (VMI)

8. vRouter ทำการกำหนดค่า label ให้กับ interface ของ VM และเพิ่ม label ดังกล่าวใน MPLS route table และทำการส่งข้อมูลดังกล่าวไปยัง OpenSDN controller ผ่าน XMPP ซึ่งทำให้ contoller รู้ว่าหากต้องการส่งข้อมูลไปยัง VM เครื่องดังกล่าวจะส่งไปยัง vRouter (compute node) ใด และ label ของ MPLS เป็นค่าอะไร
9. OpenSDN controller ทำการประกาศ route ว่ามี VM ใหม่ถูกสร้างขึ้นใน vRouter (compute node) เครื่องดังกล่าวไปยังทุกๆ vRouter เครื่องอื่นที่มี VM ทำงานอยู่ใน virtual network เดียวกันและต่าง virtual network กัน
10. OpenSDN controller ทำการประกาศ route ให้กับ VM อื่นๆ ที่กำหนดให้สื่อสารกันได้ยัง vRouter ของ VM ที่ถูกสร้างใหม่



รูปที่ 3 โครงสร้างของ vRouter

จากรูปคือการแสดงรายละเอียดการทำงานของ vRouter ใน compute node โดยใน vRouter จะมี vRouter agent ที่ทำงานอยู่ใน user space ของ operating system (host) และมี vRouter forwarder ที่ทำงานอยู่ใน kernel space

โดยที่ vRouter agent ค่อยทำงานคู่กันกับ controller และจะรับส่งข้อมูล เช่น VRFs, Route และ access control lists (ACLs) ที่จำเป็น โดยที่จะเก็บข้อมูลต่าง ๆ ไว้ในฐานข้อมูลของตัวเอง และจะใช้ข้อมูลดังกล่าวในการตัดสินใจสำหรับการส่งข้อมูล (packet) ของ interface ที่เชื่อมต่ออยู่กับ VRF และ forwarding information base (FIB) ในแต่ละ VRF ด้วย

ในแต่ละ VRF จะข้อมูลของ forwarding และ flow table ซึ่งจะมี MPLS และ VXLAN table ที่เป็นข้อมูลในระดับ global ของ vRouter โดยที่ forwarding table จะมีข้อมูล route ทั้ง IP address และ MAC adddress ของปลายทาง และมี IP-to-MAC ที่ใช้สำหรับทำหน้าที่เป็น proxy ARP ด้วย ส่วนของ MPLS จะมีการเลือกและกำหนดค่า label เมื่อมีการสร้าง VM ใหม่เกิดขึ้นอ้างอิงจาก interface ของ VM ซึ่งจะมีค่าเป็นของแต่ละ vRouter เท่านั้น และอีกส่วนคือ VXLAN จะมีการกำหนด VXLAN Network Identifier (VNI) จะทำการกำหนดค่าให้เหมือนกันกับทุกๆ VRFs ที่อยู่ใน virtual network เดียวกัน ในทุกๆ vRouter

ในแต่ละ virtual network จะมี default gateway address ที่ถูกำหนดไว้ และทุกๆ interface ของ VM จะได้รับ IP address จาก DHCP ในแต่ละ virtual network นั้นๆ เมื่อ VM ต้องการส่งข้อมูล (packet) ออกไปยังนอก virtual network (ต่างเครือข่าย) ก็จะทำการส่ง ARP ไปถามที่ default gateway IP และ vRouter ก็จะทำการตอบกลับมาด้วย MAC address ของ vRouter ซึ่งการทำงานในลักษณะนี้คือการทำ distributed default gateway ซึ่งจะทำในลักษณะแบบเดียวกันกับทุกๆ virtual network

Mobile Edge Computing (MEC) Platform-as-a-Service

Mobile Edge Computing หรือ Multi-access edge computing (MEC) เป็นแนวคิดสถาปัตยกรรมเครือข่ายที่กำหนดโดย ETSI ซึ่งช่วยเปิดความสามารถในการประมวลผลบนคลาวด์ และการบริการด้านไอทีที่ขอบของ Cellular network หรือขอบของเครือข่ายอื่นใดๆ โดยแนวคิดพื้นฐานที่อยู่เบื้องหลัง MEC คือการเรียกใช้แอปพลิเคชัน และดำเนินการประมวลผลที่เกี่ยวข้องให้ใกล้กับอุปกรณ์ของผู้ใช้มากขึ้น ซึ่งจะทำให้ความแออัดของเครือข่ายลดลง และแอปพลิเคชันต่างๆ ทำงานได้มีประสิทธิภาพมากขึ้น

หลักการที่สำคัญของ Mobile Edge Computing (MEC) มีดังนี้

1. NFV Alignment: MEC ใช้แพลทฟอร์มการจำลองเครือข่ายเสมือน (Network Virtualisation Platform) สำหรับการรันแอพพลิเคชันที่ Mobile Network Edge ซึ่งแพลทฟอร์มดังกล่าวถูกจัดเตรียมโดย Network Function Virtualization (NFV)
2. Mobility support: MEC ต้องออกแบบให้รองรับการเคลื่อนที่ เพราะอุปกรณ์ส่วนใหญ่ที่เชื่อมต่อกับเครือข่าย เป็นอุปกรณ์ที่เคลื่อนที่ได้ภายใต้เครือข่ายมือถือ เมื่ออุกรณ์มีการเปลี่ยนที่ไปยัง host อื่น MEC application บน host จะต้องดูแลอุปกรณ์นั้นๆต่อ โดยจะต้องมุ่งเน้นในด้านความต่อเนื่องของบริการ, ความคล่องตัวของ MEC Application หรือ VM และความคล่องตัวของข้อมูลที่เกี่ยวข้องกับผู้ใช้เฉพาะแอปพลิเคชัน
3. Deployment independence: ด้วยเหตุผลด้านประสิทธิภาพ ต้นทุน ความสามารถในการเพิ่มจำนวน ความต้องการของผู้ปฏิบัติงาน ฯลฯ การติดตั้ง MEC จำเป็นต้องรองรับสถานการณ์การใช้งานที่แตกต่างกันตามสถานที่และสถาณการณ์ต่างๆ เช่น การติดตั้งที่โหนดวิทยุ, การติดตั้งที่จุดร่วม, การติดตั้งที่ขอบของเครือข่ายหลัก (edge of the Core

Network) เป็นต้น ซึ่งจะต้องครอบคลุมถึงการป้องกันการเข้าถึงอย่างผิดกฎหมาย การตรวจสอบสิทธิ์ และดูแลความปลอดภัยในการสื่อสารระหว่าง Radio node(s) และ MEC service ด้วย

4. Simple and controllable APIs: เพื่อให้เกิดระบบ MEC ที่มีประสิทธิภาพ การพัฒนา API ให้เรียบง่ายที่สุดเท่าที่จะเป็นไปได้ และตอบสนองความต้องการของแอปพลิเคชันได้โดยตรงจึงเป็นสิ่งสำคัญมาก ซึ่งในขอบเขตที่เป็นไปได้ MEC จำเป็นจะต้อง reuse APIs ตัวเดิมเพื่อให้ตรงตามข้อกำหนด และในบางครั้งผู้ให้บริการจำเป็นต้องพร้อมเสมอในการเข้าไปควบคุมการเข้าถึง API บางตัว กรณีที่ radio node หรือ MEC host มีโหลดที่สูงเกินไป หรือไม่สามารถให้บริการได้
5. Smart application location: MEC application มีข้อกำหนดหลายประการ ในแง่ของการประมวลผล การจัดเก็บข้อมูล และทรัพยากรเครือข่าย และที่สำคัญกว่านั้น บางแอปพลิเคชันอาจมีข้อกำหนดในแง่ของเวลาแฝง (latency) ซึ่งรวมถึงความเป็นธรรมในการตอบสนองด้วย MEC application บางประเภท เมื่อเวลาผ่านไปต้องการให้ระบบ MEC เปลี่ยนตำแหน่งของแอปพลิเคชัน เนื่องจากอุปกรณ์ของผู้ใช้งานกำลังย้ายจากโหนดหนึ่งไปอีกโหนดหนึ่ง รวมถึงในบางสถานที่อาจมีต้นทุนที่ไม่เท่ากัน ทำให้ ‘best location’ อาจไม่ใช่ตัวเลือกที่ดีที่สุดเสมอในการเรียกใช้ MEC application ด้วยเหตุผลเหล่านี้ MEC application จึงจำเป็นต้องทำงานให้ ถูกที่ ถูกเวลา และออกแบบให้รองรับกับการจัดการ life-cycle ของแอปพลิเคชันทั่วทั้งระบบ
6. Application mobility to/from an external system: เพื่อความต่อเนื่องของการให้บริการ เมื่อมีการย้ายอุปกรณ์ของผู้ใช้งาน หรือ application instance ระบบจะต้องสามารถย้าย mobile edge application ที่ทำงานในสภาพแวดล้อมของคลาวด์ภายนอก มายัง MEC host ที่ตรงตามข้อกำหนด และสามารถย้าย MEC application จาก MEC host ไปยังคลาวด์ภายนอกระบบ MEC ได้ ซึ่งสิ่งที่ต้องคำนึงถึง ได้แก่ วิธีการถ่ายโอนไฟล์ที่ทำงานในระบบคลาวด์ภายนอกไปยังโฮสต์ MEC ต้นทาง และวิธีย้ายอินสแตนซ์ของแอปพลิเคชัน รวมถึงบริบทของผู้ใช้ไปยังโฮสต์ MEC เป้าหมาย

Mobile Edge Computing (MEC) ประกอบด้วยฟีเจอร์สำคัญ 3 อย่าง ได้แก่

1. User Applications Support แพลทฟอร์มจะต้องรองรับ Applications ของผู้ใช้งาน (MEC Application) โดยมีเงื่อนไข ดังนี้
    - แพลทฟอร์มจะต้องสนับสนุนการสร้าง VM ของ MEC Application ต่างๆ
    - แพลทฟอร์มจะต้องสนับสนุนการสร้างการเชื่อมต่อระหว่างอุปกรณ์ของผู้ใช้ และอินสแตนซ์ของ MEC Application
    - แพลทฟอร์มจะต้องสนับสนุนการ On-boarding ของ MEC Application
    - แพลทฟอร์มจะต้องอนุญาตให้สร้างการเชื่อมต่อระหว่างอุปกรณ์ของผู้ใช้ และอินสแตนซ์เฉพาะของ MEC Application เท่านั้น
    - แพลทฟอร์มจะต้องสามารถยุติอินสแตนซ์ของ MEC Application ได้เมื่อไม่มีอุปกรณ์ของผู้ใช้ เชื่อมต่อกับมันอีกต่อไป
2. Smart Relocation แพลทฟอร์มจะต้องรองรับการย้ายที่ของอุปกรณ์ของผู้ใช้งาน โดยมีเงื่อนไข ดังนี้
    - การจัดการ MEC จะต้องรองรับการย้ายตำแหน่งของอินสแตนซ์แอปพลิเคชัน MEC จากโฮสต์ MEC หนึ่งไปยังโฮสต์อื่นภายในระบบ

- โฮสต์ MEC จะต้องรองรับการย้ายตำแหน่งของอินสแตนซ์แอปพลิเคชัน MEC จากโฮสต์อื่น (ภายในระบบ) ไปยังโฮสต์เฉพาะเจาะจง และจากโฮสต์เฉพาะเจาะจงไปยังโฮสต์อื่น (ภายในระบบ) ได้
- แพลทฟอร์มจะต้องสามารถย้าย MEC application instances ระหว่างโฮสต์ MEC เพื่อให้เป็นไปตามข้อกำหนดของแอปพลิเคชัน MEC เช่น latency, ข้อกำหนดด้านพลังงาน, ข้อกำหนดด้านการใช้ทรัพยากร ฯลฯ ได้
- แพลทฟอร์มจะต้องสามารถย้ายแอปพลิเคชัน MEC ที่ทำงานในสภาพแวดล้อมคลาวด์ไปยังโฮสต์ MEC ที่ตรงตามข้อกำหนดของแอปพลิเคชัน MEC และย้ายแอปพลิเคชัน MEC จากโฮสต์ MEC ไปยังสภาพแวดล้อมระบบคลาวด์นอกระบบ MEC ได้
- แพลทฟอร์มใช้ในการติดต่อกับลูกค้า อาจต้องมีการรองรับการรับไฟล์แอปพลิเคชันจากระบบคลาวด์ภายนอก และรองรับการแปลงไฟล์ดังกล่าวเป็นอิมเมจของแอปพลิเคชันที่สามารถสร้างอินสแตนซ์ในแพลทฟอร์ม MEC ได้

3. Radio Network Information แพลทฟอร์มจะต้องรองรับการใช้งานเครือข่ายวิทยุ โดยมีเงื่อนไข ดังนี้
    - จะต้องมีบริการ MEC ที่จะเปิดเผยข้อมูลเครือข่ายวิทยุที่เป็นปัจจุบันเกี่ยวกับสภาพเครือข่ายวิทยุในปัจจุบัน
    - จะต้องมีบริการ MEC ที่ให้ข้อมูลเครือข่ายวิทยุที่ทันสมัยและเหมาะสม ซึ่งอาจเป็นข้อมูลจากแหล่งภายนอก และ/หรือสร้างขึ้นเองก็ได้
    - ข้อมูลเครือข่ายวิทยุจะต้องให้รายละเอียดที่เกี่ยวข้อง เช่น ข้อมูลต่ออุปกรณ์ของผู้ใช้ (UE) หรือต่อเซลล์ ต่อช่วงเวลา เป็นต้น
    - ข้อมูลเครือข่ายวิทยุที่ให้มาจะต้องรวมถึงข้อมูลการวัด และสถิติที่เกี่ยวข้องกับผู้ใช้ ข้อมูลนี้จะขึ้นอยู่กับข้อมูลที่กำหนดโดยข้อกำหนด 3GPP
    - ข้อมูลเครือข่ายวิทยุที่ให้มาจะรวมถึงข้อมูลที่เกี่ยวข้องกับ User Equipment (UE) ที่เชื่อมต่อกับโหนดวิทยุที่เกี่ยวข้องกับโฮสต์ MEC, บริบท UE ของพวกเขา และผู้ถือสิทธิ์การเข้าถึงที่เกี่ยวข้อง
    - ข้อมูลเครือข่ายวิทยุที่ให้มาจะรวมถึงข้อมูล QoS ในการเชื่อมต่อเฉพาะ, ปริมาณการใช้งานจริง, คำแนะนำ bitrate และ throughput ที่มีอยู่สำหรับการเชื่อมต่อที่กำหนด
    - แพลทฟอร์ม MEC จะสามารถถ่ายทอดข้อมูลสถานะวิทยุและเครือข่ายไปยังเซิร์ฟเวอร์ DANE ที่อยู่ในโดเมนบุคคลที่สามภายนอกเครือข่ายผู้ให้บริการได้ตามที่กำหนด

2.1.3) IoT Platform

ก่อนการมี IoT Platform ระบบ IoT ทำงานโดยการส่งข้อมูลจากอุปกรณ์เซ็นเซอร์หรืออุปกรณ์ IoT ตรงไปยังระบบที่เกี่ยวข้องในรูปแบบที่เรียกว่า Device-to-System Communication โดยไม่มีโครงสร้างที่เป็นศูนย์กลาง ข้อมูลเหล่านี้ถูกส่งผ่านโปรโตคอลเฉพาะที่แตกต่างกันตามผู้ผลิต ทำให้เกิดข้อจำกัดในด้านการเชื่อมต่อและการบูรณาการข้อมูล นอกจากนี้ การประมวลผลข้อมูลส่วนใหญ่ต้องพึ่งพาศูนย์ข้อมูลกลาง (Centralized Data Center) ซึ่งตั้งอยู่ห่างไกลจากแหล่งที่มาของข้อมูล ส่งผลให้เกิดความล่าช้าในการประมวลผลข้อมูลและลดความสามารถในการตอบสนองแบบเรียลไทม์

ระบบในยุคก่อนหน้า IoT Platform ยังขาดระบบการจัดการข้อมูลที่มีมาตรฐานและปลอดภัย ข้อมูลถูกจัดเก็บในหลายแหล่งโดยไม่มีการเชื่อมโยงร่วมกัน การวิเคราะห์ข้อมูลต้องเผชิญกับความยุ่งยากในการรวมข้อมูลที่มีโครงสร้างต่างกัน นอกจากนี้ การส่งข้อมูลปริมาณมากไปยังศูนย์กลางยังทำให้เกิดค่าใช้จ่ายด้านแบนด์วิธที่สูง และความซับซ้อนในการดำเนินงาน

ซึ่งในปัจจุบัน IoT Platform เป็นโครงสร้างสำคัญที่ช่วยเชื่อมโยง จัดการ และประมวลผลข้อมูลจากอุปกรณ์ในระบบ Internet of Things (IoT) ซึ่งกลายเป็นหัวใจหลักในการขับเคลื่อนโลกดิจิทัลในปัจจุบัน ด้วยข้อมูลจำนวนมหาศาลที่ถูกสร้างขึ้นทุกวินาที IoT Platform ช่วยให้ข้อมูลเหล่านี้ถูกจัดการอย่างมีประสิทธิภาพ ลดความซับซ้อน และนำไปใช้ในภาคอุตสาหกรรมต่าง ๆ เช่น การผลิต การแพทย์ และการขนส่ง นอกจากนี้ IoT Platform ยังช่วยเพิ่มความสามารถในการวิเคราะห์ข้อมูลและการตัดสินใจที่แม่นยำ ทำให้สามารถตอบสนองต่อความต้องการขององค์กรในยุคดิจิทัลได้อย่างรวดเร็วและยืดหยุ่น IoT Platform ยังสร้างมาตรฐานในการจัดการข้อมูลและอุปกรณ์ IoT ซึ่งช่วยให้ระบบสามารถทำงานร่วมกันได้อย่างราบรื่น โดยเฉพาะในกรณีที่มีการใช้งานอุปกรณ์จากผู้ผลิตหลายราย การสร้างมาตรฐานดังกล่าวช่วยลดปัญหาด้านความเข้ากันไม่ได้ของโปรโตคอล และเพิ่มประสิทธิภาพในการเชื่อมโยงข้อมูลระหว่างอุปกรณ์และแพลตฟอร์มต่าง ๆ

นิยามและหน้าที่ของ IoT Platform คือการเป็นระบบศูนย์กลางที่รวบรวม วิเคราะห์ และจัดการข้อมูลจากอุปกรณ์ IoT โดยช่วยให้ข้อมูลเหล่านั้นสามารถนำไปใช้ในรูปแบบที่มีประสิทธิภาพและเหมาะสมกับบริบทต่าง ๆ

IoT platform compare

ในโลกของเทคโนโลยี IoT Platform มีตัวเลือกที่หลากหลายและได้รับการพัฒนาจากผู้ให้บริการชั้นนำและองค์กรเทคโนโลยีต่าง ๆ การเลือก IoT Platform ที่เหมาะสมจึงเป็นขั้นตอนสำคัญที่ช่วยให้องค์กรสามารถตอบสนองต่อความต้องการของระบบ IoT ได้อย่างมีประสิทธิภาพ การเปรียบเทียบแพลตฟอร์มเหล่านี้จะช่วยให้องค์กรมีมุมมองที่ครอบคลุมทั้งจุดเด่น จุดด้อย และความสามารถในการปรับตัวให้สอดคล้องกับบริบทเฉพาะของแต่ละองค์กร

เกณฑ์ในการเปรียบเทียบ IoT Platform

1. ความสามารถในการปรับตัว (Scalability)

IoT Platform ควรมีศักยภาพในการรองรับอุปกรณ์ IoT และข้อมูลที่เพิ่มขึ้นในอนาคต โดยไม่ส่งผลต่อประสิทธิภาพของระบบ ตัวอย่างเช่น AWS IoT Core และ Microsoft Azure IoT Hub ซึ่งมีโครงสร้างพื้นฐานระดับโลกที่สามารถรองรับการขยายตัวอย่างไร้ขีดจำกัด และตอบโจทย์ความต้องการขององค์กรที่มีปริมาณอุปกรณ์ IoT มาก

2. รองรับโปรโตคอลมาตรฐาน

ความสามารถในการรองรับโปรโตคอลต่าง ๆ เช่น MQTT, CoAP และ HTTP ช่วยให้ IoT Platform ทำงานร่วมกับอุปกรณ์จากผู้ผลิตหลากหลายรายได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น Google Cloud IoT Core ที่สนับสนุนโปรโตคอลมาตรฐานเหล่านี้ ทำให้การเชื่อมต่ออุปกรณ์เป็นไปอย่างราบรื่นและมีเสถียรภาพ

3. เครื่องมือวิเคราะห์ข้อมูล

แพลตฟอร์ม IoT ที่ดีควรมีเครื่องมือวิเคราะห์ข้อมูลที่ทรงพลัง เช่น Predictive Analytics, AI และ Machine Learning เพื่อสนับสนุนการตัดสินใจ ตัวอย่างเช่น IBM Watson IoT Platform ที่มีจุดเด่นในการวิเคราะห์ข้อมูลเชิงลึกและการใช้งาน AI เพื่อเพิ่มประสิทธิภาพการดำเนินงานของระบบ IoT

4. ความปลอดภัยของข้อมูล

การปกป้องข้อมูลถือเป็นหัวใจสำคัญของ IoT Platform เช่น การเข้ารหัสข้อมูล การตรวจสอบสิทธิ์ และการควบคุมการเข้าถึง ตัวอย่างเช่น AWS IoT Core ที่มีระบบความปลอดภัยที่ครอบคลุมตั้งแต่การเชื่อมต่อไปจนถึงการจัดเก็บข้อมูล ทำให้ผู้ใช้งานมั่นใจในความปลอดภัยของระบบ

5. การใช้งานและการพัฒนา (Ease of Use)

IoT Platform ควรมีอินเทอร์เฟซที่ใช้งานง่าย พร้อมทั้งเครื่องมือสนับสนุน เช่น SDK และ API ที่ช่วยให้นักพัฒนาสามารถสร้างและจัดการระบบได้อย่างรวดเร็ว ตัวอย่างเช่น Microsoft Azure IoT Hub ซึ่งมอบความสะดวกในการพัฒนาและการจัดการอุปกรณ์ IoT อย่างครบวงจร

6. ความยืดหยุ่นและโอเพ่นซอร์ส (Flexibility & Open Source)

แพลตฟอร์มโอเพ่นซอร์ส เช่น ThingsBoard ช่วยให้องค์กรสามารถปรับแต่งระบบได้ตามความต้องการ โดยเฉพาะในโครงการที่มีทรัพยากรจำกัด หรือที่ต้องการการปรับเปลี่ยนเพื่อรองรับการพัฒนาที่เฉพาะเจาะจง

ตัวอย่าง IoT Platform ยอดนิยม

- AWS IoT Core เป็นแพลตฟอร์มที่ได้รับความนิยมในด้านการรองรับปริมาณข้อมูลและอุปกรณ์ IoT จำนวนมหาศาล อีกทั้งยังสามารถเชื่อมโยงกับบริการใน AWS Ecosystem ได้อย่างลึกซึ้ง เช่น การจัดเก็บข้อมูลใน Amazon S3 และการวิเคราะห์ข้อมูลแบบเรียลไทม์ผ่าน Amazon QuickSight
- Microsoft Azure IoT Hub มีความโดดเด่นในด้านการพัฒนาและการจัดการอุปกรณ์ IoT โดยเฉพาะในด้านการเชื่อมต่อที่รองรับอุปกรณ์หลากหลายประเภท อีกทั้งยังสามารถเชื่อมโยงกับ Azure Machine Learning เพื่อสนับสนุนการตัดสินใจที่มีประสิทธิภาพ
- Google Cloud IoT Core มีจุดเด่นในด้านการเชื่อมต่อกับบริการ AI และ Machine Learning ใน Google Cloud ซึ่งช่วยให้องค์กรสามารถวิเคราะห์ข้อมูลและคาดการณ์แนวโน้มได้อย่างแม่นยำ
- IBM Watson IoT Platform มีจุดแข็งในด้านการใช้ AI เพื่อวิเคราะห์ข้อมูลเชิงลึก และสนับสนุนการดำเนินงานในภาคอุตสาหกรรมที่ต้องการการประมวลผลข้อมูลที่ซับซ้อน
- ThingsBoard เป็นแพลตฟอร์มโอเพ่นซอร์สที่เหมาะสำหรับการศึกษาและการพัฒนาต้นแบบ รองรับโปรโตคอลมาตรฐาน เช่น MQTT, CoAP และ HTTP และสามารถปรับแต่งแดชบอร์ดได้อย่างยืดหยุ่น เหมาะสำหรับโครงการที่ต้องการการพัฒนาเฉพาะด้าน

IoT Platform เป็นองค์ประกอบสำคัญในการสร้างระบบ IoT ที่มีประสิทธิภาพ ความปลอดภัย และความยืดหยุ่นสูง การเลือกแพลตฟอร์มที่เหมาะสมจะช่วยให้องค์กรสามารถรองรับการเปลี่ยนแปลงในยุคดิจิทัล และเพิ่มขีดความสามารถในการแข่งขันในระยะยาว การลงทุนใน IoT Platform ที่เหมาะสมจึงเป็นกุญแจสำคัญสู่ความสำเร็จในยุคที่เทคโนโลยี IoT มีบทบาทสำคัญมากขึ้นในทุกภาคส่วน

2.2 ผลงานวิจัยที่เกี่ยวข้อง

2.2.1 Multi-access Edge Computing by ETSI [8]

ETSI หรือ European Telecommunications Standards Institute ได้มีการวิจัยที่มุ่งเน้นการพัฒนานา MEC ให้เป็นกลไกสำคัญที่ช่วยปรับปรุงความสามารถในการประมวลผลข้อมูลในระดับ Edge โดยลด Latency ให้เหลือน้อยที่สุด พร้อมทั้งเพิ่มความสามารถในการรองรับข้อมูลที่ต้องการการประมวลผลแบบเรียลไทม์และการตอบสนองในสถานการณ์ที่ต้องการความแม่นยำสูง การศึกษาได้ให้ความสำคัญกับการผสาน MEC เข้ากับเครือข่าย 5G ซึ่งเปิดโอกาสให้ระบบสามารถขยายตัวอย่างยั่งยืน รองรับการประมวลผลข้อมูลขนาดใหญ่และหลากหลายจากอุปกรณ์ IoT ได้อย่างเสถียร

นอกจากนี้ยังชี้ให้เห็นถึงการออกแบบสถาปัตยกรรมเครือข่ายที่สามารถปรับเปลี่ยนได้ตามปริมาณงานและความต้องการในแต่ละช่วงเวลา การใช้งาน MEC ในการจัดการโครงสร้างข้อมูลในเมืองอัจฉริยะ ระบบขนส่ง และการผลิตอุตสาหกรรมได้รับการวิเคราะห์ในเชิงลึกผ่านกรณีศึกษาต่างๆ เช่น การติดตามสถานะผลิตภัณฑ์ในสายการผลิต การควบคุมปริมาณการจราจรแบบไดนามิก และการสนับสนุนระบบสุขภาพที่ต้องการการวิเคราะห์ข้อมูลแบบทันที

งานวิจัยนี้ยังสำรวจถึงโอกาสและความท้าทายในการพัฒนา MEC ให้กลายเป็นส่วนหนึ่งที่สำคัญของการจัดการโครงสร้าง Edge Cloud ในอนาคต ซึ่งครอบคลุมถึงการใช้เทคโนโลยี AI และการวิเคราะห์ข้อมูลแบบกระจายเพื่อเพิ่มประสิทธิภาพในการตัดสินใจเชิงกลยุทธ์ในระดับเครือข่ายอุตสาหกรรมและเมืองดิจิทัล

2.2.2 Edge Computing: Classification, Applications, and Challenges [9]

งานวิจัยนี้นำเสนอการจำแนกประเภทของการประมวลผลแบบเอดจ์ (Edge Computing) พร้อมทั้งสำรวจการประยุกต์ใช้งานในด้านต่าง ๆ และความท้าทายที่เกี่ยวข้อง การประมวลผลแบบเอดจ์เป็นแนวคิดที่เน้นการประมวลผลข้อมูลใกล้กับแหล่งกำเนิดข้อมูลมากที่สุด เพื่อลดความหน่วงเวลา (latency) และเพิ่มประสิทธิภาพในการตอบสนองของระบบ

การศึกษานี้ได้สรุปเทคโนโลยีการประมวลผลแบบเอดจ์ที่สำคัญ เช่น Mobile Edge Computing, Cloudlet, และ Fog Computing ซึ่งช่วยให้นักวิจัยและผู้ปฏิบัติงานเข้าใจถึงโซลูชันปัจจุบันและแนวทางในอนาคต นอกจากนี้ ยังได้วิเคราะห์การประยุกต์ใช้การประมวลผลแบบเอดจ์ในสถานการณ์ต่าง ๆ เช่น การวิเคราะห์ข้อมูลแบบเรียลไทม์ในระบบ IoT การสนับสนุนการทำงานของอุปกรณ์ที่มีข้อจำกัดด้านพลังงาน และการเพิ่มความปลอดภัยของข้อมูลอย่างไรก็ตาม งานวิจัยยังชี้ให้เห็นถึงความท้าทายที่ต้องเผชิญในการนำการประมวลผลแบบเอดจ์มาใช้ เช่น การจัดการทรัพยากรที่จำกัด การรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูล และการบูรณาการกับระบบคลาวด์ที่มีอยู่เดิม

2.2.3 Data Visualization for Wireless Sensor Networks Using ThingsBoard [10]

งานวิจัยนี้มุ่งเน้นการใช้ ThingsBoard ซึ่งเป็นแพลตฟอร์ม IoT โอเพนซอร์สที่มีความยืดหยุ่นสูง ในการสร้างระบบที่สามารถแสดงผลข้อมูลจาก Wireless Sensor Networks (WSNs) ได้อย่างมีประสิทธิภาพและครอบคลุมความต้องการใน

หลากหลายบริบท ThingsBoard ถูกออกแบบมาให้รองรับการเชื่อมต่อเซ็นเซอร์หลากหลายประเภท ซึ่งช่วยให้ข้อมูลที่รวบรวมจากเครือข่าย WSNs สามารถนำมาประมวลผลและแสดงผลในรูปแบบที่เข้าใจง่ายผ่านแดชบอร์ดแบบเรียลไทม์ การแจ้งเตือนอัตโนมัติ และการวิเคราะห์ข้อมูลเชิงลึกสำหรับการสนับสนุนการตัดสินใจ งานวิจัยยังให้ความสำคัญกับการพัฒนาแดชบอร์ดที่ปรับแต่งได้ตามความต้องการเฉพาะของผู้ใช้งาน ตัวอย่างเช่น แดชบอร์ดการตรวจสอบคุณภาพอากาศภายในเมืองใหญ่ การบริหารจัดการพลังงานในภาคอุตสาหกรรม และการเฝ้าระวังความปลอดภัยทางสิ่งแวดล้อมในพื้นที่เสี่ยง นอกจากนี้ ThingsBoard ยังมีความสามารถในการนำเสนอข้อมูลแบบโต้ตอบเพื่อสนับสนุนการวิเคราะห์เชิงลึกที่ช่วยให้ผู้ใช้งานสามารถสร้างคำสั่งหรือรูปแบบการควบคุมใหม่ได้อย่างเหมาะสม

งานวิจัยยังเน้นถึงความสามารถของ ThingsBoard ในการเชื่อมต่อกับโปรโตคอลมาตรฐาน เช่น MQTT, HTTP และ CoAP ซึ่งช่วยเพิ่มประสิทธิภาพในการรวบรวมข้อมูลจากหลากหลายแหล่ง นอกจากนี้ ยังมีการศึกษาการออกแบบระบบที่รองรับการขยายตัวของอุตสาหกรรมในอนาคต โดยเฉพาะในบริบทของเมืองอัจฉริยะที่ต้องการการบริหารจัดการทรัพยากรอย่างมีประสิทธิภาพ การเกษตรอัจฉริยะที่เน้นการตรวจสอบและวิเคราะห์ข้อมูลสิ่งแวดล้อม เช่น ความชื้นและอุณหภูมิ ผลลัพธ์จากการศึกษายังแสดงให้เห็นว่า ThingsBoard มีศักยภาพในการช่วยลดต้นทุนการพัฒนาและเพิ่มความคล่องตัวในการจัดการข้อมูลเชิงโครงสร้างสำหรับโครงการขนาดใหญ่

นอกจากนี้ งานวิจัยได้เน้นถึงความสำคัญของการวิเคราะห์ข้อมูลแบบเรียลไทม์ที่สามารถสนับสนุนการตัดสินใจในภาคธุรกิจและอุตสาหกรรมได้อย่างมีประสิทธิภาพ ThingsBoard ช่วยลดช่องว่างระหว่างการรวบรวมข้อมูลและการวิเคราะห์ด้วยการผสานการแสดงผลข้อมูลแบบอินเทอร์แอคทีฟ รวมถึงความสามารถในการคาดการณ์เหตุการณ์ล่วงหน้าจากข้อมูลที่รวบรวมมา ผลลัพธ์ที่ได้สนับสนุนให้แพลตฟอร์มนี้เหมาะสมกับการใช้งานในบริบทของ IoT ที่ต้องการการตอบสนองอย่างรวดเร็วในระบบเครือข่ายไร้สายที่ซับซ้อน

บทที่ 3
ระเบียบวิธีการวิจัยและพัฒนาโครงการ

3.1 ข้อมูลผู้ใช้งานและอุตสาหกรรมเป้าหมาย

ในโครงการนี้ ได้มุ่งเน้นการวิเคราะห์ข้อมูลผู้ใช้งานและอุตสาหกรรมเป้าหมายในภูมิภาคตอย่างละเอียดเพื่อให้สามารถพัฒนาระบบที่ตอบสนองความต้องการได้อย่างตรงจุด โดยประกอบด้วย:

- **ผู้ใช้งานในภาคอุตสาหกรรม:**
    - โรงงานผลิตสินค้า เช่น อุตสาหกรรมอาหารและเครื่องดื่ม
    - อุตสาหกรรมยานยนต์ เช่น โรงงานผลิตชิ้นส่วนและการประกอบรถยนต์
    - อุตสาหกรรมพลังงาน เช่น การจัดการพลังงานแสงอาทิตย์และพลังงานลม
- **ผู้ใช้งานในหน่วยงานภาครัฐและองค์กร:**
    - หน่วยงานด้านการจัดการสาธารณูปโภค เช่น การไฟฟ้าและการประปา
    - โรงพยาบาลและสถานพยาบาลที่ต้องการระบบตรวจสอบเครื่องมือทางการแพทย์
- **ผู้ใช้งานในธุรกิจขนาดกลางและขนาดย่อม (SME):**
    - ผู้ประกอบการรายย่อยในกลุ่ม IoT เช่น ธุรกิจพัฒนาเซ็นเซอร์ติดตามอุปกรณ์

3.1.2 ความต้องการของผู้ใช้งาน

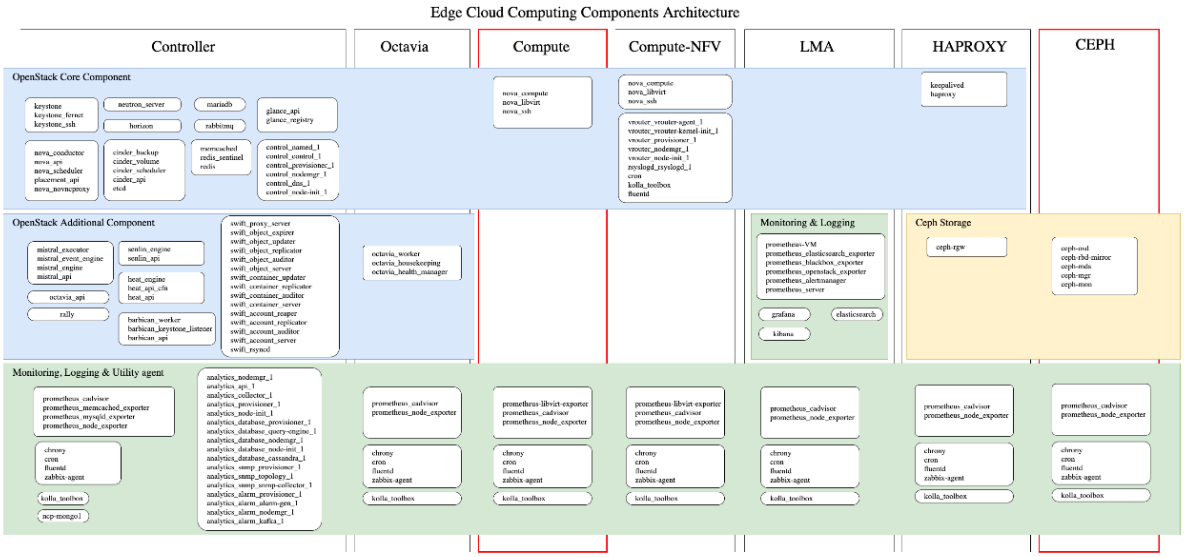
- **ความเร็วในการประมวลผล:** ระบบต้องเร็วพอที่จะตอบสนองทันที เช่น แจ้งเตือนเหตุการณ์สำคัญ ระบบนี้ยังสามารถวิเคราะห์ข้อมูลที่ได้รับแบบเรียลไทม์เพื่อช่วยตัดสินใจในสถานการณ์สำคัญได้อย่างมีประสิทธิภาพ
- **ความยืดหยุ่น:** ใช้ได้กับหลายธุรกิจ เช่น โรงงานหรือร้านค้า นอกจากนี้ยังรองรับการปรับแต่งตามความต้องการเฉพาะของแต่ละธุรกิจ เช่น การเพิ่มฟังก์ชันเฉพาะทาง
- **ความปลอดภัย:** ป้องกันข้อมูลด้วยรหัสผ่านและระบบล็อค รวมถึงการเข้ารหัสข้อมูลก่อนการส่ง และมีระบบตรวจจับการเข้าถึงที่ผิดปกติเพื่อเพิ่มความปลอดภัย
- **มาตรฐาน:** ใช้โปรโตคอลสากล เช่น MQTT สำหรับเชื่อมต่อ พร้อมทั้งรองรับการทำงานร่วมกับอุปกรณ์จากหลายผู้ผลิตที่ใช้มาตรฐานเดียวกัน
- **ลดค่าใช้จ่าย:** ใช้เครือข่ายและอุปกรณ์ให้คุ้มที่สุด โดยการประมวลผลข้อมูลที่ต้นทางเพื่อลดการใช้แบนด์วิธ และลดค่าใช้จ่ายในการจัดเก็บข้อมูลในระบบคลาวด์

3.2 การวิจัยและพัฒนา Edge Cloud Computing

3.2.1 การกำหนดความต้องการของระบบ Edge Cloud Computing

- รองรับการประมวลผลแบบกระจายศูนย์ (Distributed Computing) เพื่อลดภาระของศูนย์ข้อมูลหลัก (Data Center)
- สนับสนุนการทำงานแบบเรียลไทม์ (Real-Time Processing) เพื่อเพิ่มความเร็วในการตอบสนอง
- มีความปลอดภัยสูง รองรับการป้องกันข้อมูลที่ส่งผ่านเครือข่ายด้วยการเข้ารหัส (Encryption)
- รองรับการขยายตัว (Scalability) เพื่อเพิ่มจำนวน Edge Nodes และการประมวลผลได้ตามความต้องการ

3.2.2 การออกแบบโครงสร้างสถาปัตยกรรมภายในของ Full Edge cloud compute



รูปที่ 4 โครงสร้างของระบบ Edge Cloud Computing

- **Node Type**

โครงสร้างของระบบ Edge Cloud Computing ถูกออกแบบโดยใช้แนวคิด Node Type ซึ่งแบ่งบทบาทและหน้าที่ของแต่ละโหนดอย่างชัดเจน เพื่อเพิ่มความสามารถในการประมวลผล ความยืดหยุ่น และความเสถียรของระบบในสภาพแวดล้อมที่มีความซับซ้อน โครงสร้างนี้ช่วยให้การจัดการทรัพยากรในแต่ละ Node เป็นไปอย่างมีประสิทธิภาพ และรองรับการใช้งานหลากหลายลักษณะในภาคธุรกิจและภาครัฐ โดยโครงสร้างหลักประกอบด้วย

- **Controller Node:**

Controller Node เป็นศูนย์กลางในการควบคุมระบบโดยทำหน้าที่บริหารจัดการส่วน Control Plane ของ OpenStack โดยใช้บริการหลัก เช่น Keystone สำหรับการจัดการ Authentication และ Horizon สำหรับการจัดการ Dashboard นอกจากนี้ยังมี MariaDB สำหรับจัดเก็บข้อมูล และ RabbitMQ สำหรับการจัดการ Messaging ระหว่าง Components ต่าง ๆ รวมถึงการใช้งาน Nova Conductor และ Nova API เพื่อควบคุมการทำงานของ Virtual Machines (VMs) Controller Node ยังรองรับการใช้งาน Glance API และ Glance Registry เพื่อจัดการ Images ตัวอย่างการใช้งาน Controller Node ได้แก่ การควบคุมทรัพยากรในระบบที่มีการใช้งานซับซ้อนและต้องการความแม่นยำสูง

- **Network Node:**

Network Node ทำหน้าที่จัดการระบบเครือข่ายเสมือน (Virtual Networking) ใน OpenStack ผ่านการใช้งาน Neutron Server ที่ช่วยให้ Virtual Machines สามารถเชื่อมต่อกับเครือข่ายได้อย่างราบรื่น อีกทั้งยังรองรับการตั้งค่า Load Balancer และ Firewall-as-a-Service (FWaaS) เพื่อเพิ่มความปลอดภัย นอกจากนี้ยังสามารถเชื่อมโยงกับโหนด Compute ผ่าน VXLAN หรือ GRE เพื่อสร้างเครือข่าย Overlay ตัวอย่างการใช้งาน เช่น การตั้งค่าเครือข่ายภายใน Data Center เพื่อให้รองรับการใช้งาน Virtual Machines จำนวนมาก

- **Octavia Node:**

Octavia Node ออกแบบมาเพื่อรองรับการทำงานของ Load Balancer โดยใช้ Amphora VM ในการกระจาย Traffic และตรวจสอบความพร้อมของ Backend Services โดย Octavia Node ทำหน้าที่จัดการส่วน Health Management ผ่าน Octavia Health Manager และรองรับฟังก์ชัน Autoscaling Load Balancer ตัวอย่างการใช้งาน Octavia Node เช่น การกระจายการทำงานในระบบอีคอมเมิร์ซขนาดใหญ่ที่ต้องการความน่าเชื่อถือสูง

- **HAProxy Node:**

HAProxy Node ใช้สำหรับการจัดการ Load Balancing โดยใช้ HAProxy และ Keepalived เพื่อเพิ่มความเสถียรและความปลอดภัยในการกระจาย Traffic Node นี้ยังรองรับ TLS/SSL เพื่อเข้ารหัสข้อมูลในระบบที่ต้องการความปลอดภัย ตัวอย่างการใช้งาน HAProxy Node ได้แก่ การจัดการการจราจรข้อมูลในระบบคลาวด์ที่มีผู้ใช้งานจำนวนมากพร้อมกัน

- **Compute Node:**

Compute Node รองรับการประมวลผลและการทำงานของ Virtual Machines (VMs) โดยใช้ Nova Compute และ Nova Libvirt เพื่อควบคุมการทำงานของ Hypervisor Compute Node ยังสามารถรองรับการใช้งาน GPU สำหรับ Machine Learning หรือ AI และสนับสนุน NUMA เพื่อเพิ่มประสิทธิภาพในการจัดการทรัพยากร ตัวอย่างการใช้งาน Compute Node เช่น การประมวลผลข้อมูล IoT ในอุตสาหกรรมการผลิต

- **Ceph Node:**

Ceph Node ทำหน้าที่เป็น Storage Node โดยรองรับการจัดเก็บข้อมูลผ่าน Ceph OSD, Ceph RBD Mirror, และ Ceph MDS เพื่อรองรับการจัดเก็บ Block Storage และ Object Storage นอกจากนี้ยังรองรับการทำงานแบบ Multi-Site เพื่อการสำรองข้อมูลที่ปลอดภัยและยืดหยุ่น ตัวอย่างการใช้งาน Ceph Node ได้แก่ การจัดการข้อมูลทางการแพทย์ที่ต้องการความเสถียรและความปลอดภัยสูง

- **MAAS Node:**

MAAS (Metal-as-a-Service) Node เป็นส่วนที่ช่วยในการบริหารจัดการ Physical Servers โดยเฉพาะ Node นี้รองรับการติดตั้งระบบปฏิบัติการ (Operating System) บน Physical Machines ผ่านการใช้งาน PXE Boot และ Web Dashboard เพื่อบริหารทรัพยากรและตรวจสอบสถานะเซิร์ฟเวอร์แบบเรียลไทม์ MAAS Node ยังช่วยเพิ่มประสิทธิภาพในการจัดการ Infrastructure-as-a-Service (IaaS) โดยเหมาะสำหรับการจัดการ Bare Metal Servers ตัวอย่างการใช้งาน MAAS Node ได้แก่ การเตรียมระบบสำหรับ High-Performance Computing (HPC) หรือการบริหารเซิร์ฟเวอร์ในระบบคลาวด์ขนาดใหญ่

- **Kolla-Deploy Node:**

Kolla-Deploy Node มีบทบาทสำคัญในการจัดการ Deployment และ Configuration ของ OpenStack ผ่าน Kolla-Ansible Node นี้ช่วยลดความซับซ้อนของการติดตั้งและปรับปรุงระบบ OpenStack โดยสามารถทำงานร่วมกับเครื่องมือ CI/CD เช่น Jenkins เพื่อสนับสนุนการปรับปรุงระบบอย่างต่อเนื่อง ตัวอย่างการใช้งาน Kolla-Deploy Node ได้แก่ การตั้งค่าและอัปเกรด OpenStack Cluster ในสภาพแวดล้อมการผลิต หรือการบริหารจัดการโครงสร้างพื้นฐานด้วยแนวทาง Infrastructure-as-Code (IaC)

- **Monitoring & Logging Nodes:**

Monitoring & Logging Nodes รองรับการติดตามสถานะและการแจ้งเตือนปัญหาของระบบ โดยใช้เครื่องมือ เช่น Prometheus, Grafana, และ Elasticsearch Nodes เหล่านี้ทำหน้าที่รวบรวม Log จากแต่ละ Node และแสดงข้อมูลผ่าน Visualization Dashboards เช่น Kibana ตัวอย่างการใช้งาน Monitoring & Logging Nodes ได้แก่ การวิเคราะห์ปัญหาของระบบและการปรับปรุงประสิทธิภาพการทำงาน

- **Additional Components:**

ในระบบยังมี Additional Components เช่น Mistral Executor สำหรับการจัดการ Workflow และ Senlin Engine สำหรับการจัดการ Cluster นอกจากนี้ยังมี Swift Proxy Server และ Swift Object Server สำหรับการจัดเก็บข้อมูลในรูปแบบ Object Storage ตัวอย่างการใช้งาน Components เหล่านี้ เช่น การจัดการข้อมูลที่มีความซับซ้อนในระบบคลาวด์สำหรับองค์กรขนาดใหญ่

- **Networking layout**

การจัดการและออกแบบเครือข่าย (Networking Layout) เป็นกระบวนการที่เกี่ยวข้องกับการออกแบบและวางโครงสร้างพื้นฐานไอทีในรูปแบบที่รองรับการใช้งานได้อย่างมีประสิทธิภาพในทุกขนาดองค์กร การออกแบบระบบเครือข่ายอย่างเหมาะสมช่วยเสริมสร้างความสามารถในการเชื่อมต่อที่รวดเร็วและมั่นคง รวมถึงการรักษาความปลอดภัยของข้อมูล โดยเฉพาะในยุคที่เทคโนโลยีมีการพัฒนาอย่างรวดเร็ว ความเข้าใจเกี่ยวกับการเพิ่มขนาดของระบบ (Scalability) และการปรับใช้เทคโนโลยีใหม่จึงเป็นหัวใจสำคัญในการออกแบบเครือข่าย

ในการวางแผน Networking Layout มีขั้นตอนสำคัญ เช่น การเลือกประเภทเครือข่ายที่เหมาะสม การจัดการทรัพยากรต่าง ๆ ให้เกิดประสิทธิภาพสูงสุด และการเลือกใช้อุปกรณ์และซอฟต์แวร์ที่มีความสอดคล้องกับความต้องการของระบบ ตัวอย่างเช่น การแบ่งส่วนการทำงานของเครือข่ายออกเป็นเครือข่ายบริหารจัดการ (Management Network) เครือข่ายการสื่อสารของเครื่องเสมือน (VM Traffic) และเครือข่ายสำหรับการใช้งานสาธารณะ (Public Network) ซึ่งทั้งหมดนี้ช่วยให้ระบบมีความมั่นคงและลดความซับซ้อนในการจัดการ โดยเฉพาะอย่างยิ่งในระบบที่ต้องการความยืดหยุ่นสูง เช่น OpenStack

- **Openstack inernal network**

เครือข่ายภายใน OpenStack หรือที่รู้จักกันในชื่อเครือข่ายการจัดการ (Management Network) เป็นส่วนสำคัญของระบบ OpenStack โดยใช้สำหรับจัดการฟังก์ชันต่าง ๆ และการสื่อสารระหว่างบริการภายในเครือข่ายเดียวกัน เครือข่ายนี้ทำหน้าที่เป็นตัวกลางที่สำคัญสำหรับการรับส่งข้อมูล เช่น การเชื่อมต่อระหว่าง OpenStack nova-compute และ OpenStack nova-API เพื่อให้การทำงานของระบบ OpenStack cluster มีความเสถียรและมีประสิทธิภาพสูงสุด นอกจากนี้เครือข่ายนี้ยังรองรับการเชื่อมโยงระหว่าง OpenStack Components เช่น Keystone, Glance, และ Neutron ที่ต้องพึ่งพาการสื่อสารที่มีความเสถียรสูง การกำหนดค่าที่ถูกต้องและการดูแลรักษาเครือข่ายภายใน OpenStack จึงเป็นหัวใจสำคัญในการทำให้ระบบสามารถทำงานร่วมกันได้อย่างราบรื่นในสภาพแวดล้อมที่ซับซ้อนและเปลี่ยนแปลงตลอดเวลา

- **Vm Taffic network**

เครือข่ายการจราจรของ VM (Virtual Machine Traffic Network) เป็นส่วนหนึ่งของระบบเครือข่ายชั้น Layer 2 ซึ่งทำหน้าที่สนับสนุนการสื่อสารระหว่าง VMs ภายในเครือข่ายเดียวกัน โดยเครือข่ายนี้ถูกออกแบบให้แยกจากกันเพื่อความปลอดภัย โดยใช้เทคโนโลยี VLAN หรือ VXLAN ในการสร้างเครือข่ายส่วนตัว (Private Network) VM Traffic Network ยังถูกออกแบบให้แยกจากเครือข่ายภายใน OpenStack และเมื่อต้องการเชื่อมต่อภายนอก ระบบจะส่งผ่าน Layer 3 Agent ซึ่งเป็นตัวกลางที่ช่วยให้ VMs สามารถเชื่อมต่อไปยังเครือข่ายสาธารณะได้อย่างปลอดภัย การจราจรของ VM จะถูกส่งจาก Compute Nodes ไปยัง Network Nodes ซึ่งกระบวนการนี้ช่วยเพิ่มความปลอดภัยและการจัดการที่ง่ายดายมากขึ้น รวมถึงการลดปัญหาคอขวด (bottleneck) ในการส่งข้อมูล โดย VM Traffic Network ยังรองรับการตั้งค่าที่ซับซ้อน เช่น การเชื่อมโยง Multi-tenancy Networks และการจัดการ QoS (Quality of Service) เพื่อเพิ่มประสิทธิภาพการทำงานของ VMs

- **Public network**

เครือข่ายสาธารณะมีบทบาทสำคัญในระบบ OpenStack โดยถูกออกแบบให้เชื่อมต่อกับโลกภายนอก และมีวัตถุประสงค์หลายประการ ได้แก่:

- Public OpenStack Endpoint:

ทำหน้าที่เป็นจุดเชื่อมต่อระหว่างผู้ใช้งานกับบริการ OpenStack โดยจัดการผ่าน HAProxy Nodes เพื่อให้การเข้าถึงบริการมีความปลอดภัยและสามารถรองรับผู้ใช้จำนวนมากได้

- Public IP Network Address:

ใช้ในการกำหนดที่อยู่ IP สาธารณะ (Floating IPs) และสร้าง Private Routers เพื่อช่วยให้ VMs ในเครือข่ายส่วนตัวสามารถส่งข้อมูลออกไปยังเครือข่ายสาธารณะผ่าน Cluster ได้อย่างสะดวก

Public Network ยังรองรับการตั้งค่า High Availability เพื่อให้การเข้าถึงระบบมีความต่อเนื่องในกรณีที่เกิดความล้มเหลวในส่วนของเครือข่าย นอกจากนี้ยังสามารถกำหนด Traffic Routing Policies เพื่อควบคุมการจราจรข้อมูลในแต่ละกลุ่มผู้ใช้งาน การจัดการ Public Network ต้องคำนึงถึงความปลอดภัย การจัดสรรที่อยู่ IP อย่างเหมาะสม และการตั้งค่าที่ช่วยเพิ่มประสิทธิภาพการทำงานในระดับสูงสุด ตัวอย่างเช่น การใช้งาน Load Balancers เพื่อกระจายการจราจรของผู้ใช้งานในระบบที่มีการเข้าถึงสูง

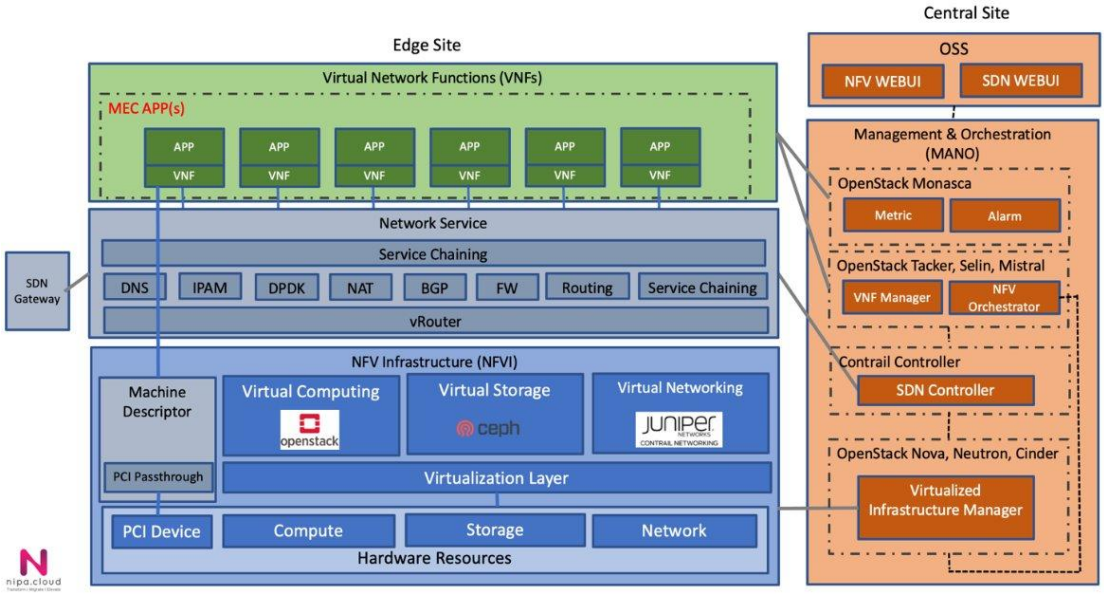
- **Admin and deploying network**

เครือข่ายสำหรับผู้ดูแลและการติดตั้งระบบ (Admin and Deploying Network) ถูกออกแบบมาเพื่อสนับสนุนงานด้านการบริหารจัดการและการติดตั้งระบบ โดยแยกออกจากฟังก์ชันหลักของ OpenStack cluster เพื่อให้การทำงานของระบบไม่ถูกรบกวนในกรณีที่เครือข่ายนี้ไม่พร้อมใช้งาน เครือข่ายนี้มีวัตถุประสงค์หลักดังนี้:

- **Admin Access:** ช่วยให้ผู้ดูแลระบบสามารถเข้าถึงเซิร์ฟเวอร์และโหนดทั้งหมดในระบบผ่าน SSH เพื่อการตรวจสอบและการแก้ไขปัญหาอย่างรวดเร็ว Admin Network ยังรองรับการเข้าถึงที่มีการควบคุมสิทธิ์เพื่อเพิ่มความปลอดภัย
- **PXE Boot for MAAS:** สนับสนุนการติดตั้งระบบปฏิบัติการแบบอัตโนมัติผ่าน PXE Boot ซึ่งช่วยลดระยะเวลาและเพิ่มความแม่นยำในการติดตั้งระบบใหม่ โดยเฉพาะในกรณีที่ต้องจัดการเซิร์ฟเวอร์จำนวนมากในโครงสร้างพื้นฐานขนาดใหญ่

- **Kolla-Ansible Inventory:** เป็นเครื่องมือสำคัญสำหรับการบริหารจัดการโหนดใน Cluster โดยช่วยให้การติดตั้ง การอัปเดต และการดูแลรักษาระบบ OpenStack เป็นไปได้อย่างมีประสิทธิภาพ นอกจากนี้ยังสนับสนุนการทำงานร่วมกับระบบ CI/CD (Continuous Integration/Continuous Deployment) เพื่อเพิ่มความยืดหยุ่นในการพัฒนาและปรับปรุงระบบ

Admin and Deploying Network มีความสำคัญต่อการสร้างความมั่นใจในความเสถียรและการจัดการระบบในสภาพแวดล้อมการใช้งานที่มีความซับซ้อน ทั้งในระยะเริ่มต้นและในระยะการบำรุงรักษา รวมถึงการสนับสนุนการแก้ไขปัญหาเชิงลึกในกรณีที่เกิดเหตุฉุกเฉิน เช่น การกู้คืนระบบหลังจากการล้มเหลว (Disaster Recovery)



รูปที่ 5 ระบบโครงสร้าง MEC Platform

- **Edge Site**
  - **Virtual Network Functions (VNFs)**

ในระบบ Edge Cloud Computing ของโครงการ บริษัท นิภา เทคโนโลยี จำกัด มีการใช้งานฟังก์ชันเครือข่ายเสมือน (VNFs) ซึ่งถือเป็นหัวใจสำคัญในการบริหารจัดการทรัพยากรเครือข่ายเสมือนให้สามารถทำงานได้อย่างมีประสิทธิภาพ โดย VNFs ช่วยลดความล่าช้า (Latency) ของระบบ และช่วยให้การประมวลผลข้อมูลอยู่ใกล้กับแหล่งข้อมูลมากที่สุด VNFs ยังช่วยเพิ่มประสิทธิภาพการทำงานในหลายแง่มุม เช่น การส่งข้อมูลแบบเรียลไทม์ การวิเคราะห์ข้อมูลที่มีความซับซ้อน และการจัดการการประมวลผลในงานที่ต้องการความรวดเร็ว เช่น การควบคุมระบบอัตโนมัติในอุตสาหกรรม การจัดการระบบขนส่งสาธารณะ และการบริการด้านสุขภาพ

- **Network Service**

Network Service ในระบบ Full Edge Cloud Computing ครอบคลุมฟังก์ชันสำคัญต่าง ๆ เช่น DNS, IPAM, Routing, Firewall (FW) และ Service Chaining โดยโครงการนี้มีการนำ OpenStack และ Contrail Controller มาใช้ในการจัดการฟังก์ชันเครือข่ายเหล่านี้ เพื่อให้การทำงานมีความปลอดภัยและเสถียรภาพสูง การบริการเครือข่ายนี้ยังช่วยลดเวลาในการตั้งค่าระบบและเพิ่มความสามารถในการปรับเปลี่ยนตามความต้องการเฉพาะด้านของผู้ใช้งาน เช่น การตั้งค่าเครือข่ายแบบเฉพาะกิจสำหรับกิจกรรมชั่วคราว หรือการเพิ่มประสิทธิภาพการจัดการแบนด์วิธในช่วงเวลาที่มีการใช้งานสูง

- **NFV Infrastructure (NFVI)**

NFVI ในโครงการนี้เป็นโครงสร้างพื้นฐานที่ครอบคลุมถึง Virtual Computing (OpenStack), Virtual Storage (Ceph) และ Virtual Networking (Juniper Contrail Network) ซึ่งช่วยสร้างรากฐานสำหรับการทำงานของฟังก์ชันเครือข่ายเสมือน (VNFs) การออกแบบ NFVI ยังรองรับการขยายตัวในอนาคต เช่น การเพิ่มจำนวนเครื่องแม่ข่าย การปรับปรุงโครงสร้างของศูนย์ข้อมูล และการรองรับการประมวลผลของอุปกรณ์ IoT ในปริมาณมาก โครงสร้างนี้ยังช่วยให้ระบบสามารถทำงานได้อย่างยืดหยุ่นแม้ในสถานการณ์ที่มีข้อจำกัดด้านทรัพยากรหรือความแปรปรวนของการใช้งาน

- **Hardware Resources**

ทรัพยากรฮาร์ดแวร์ เช่น Compute, Storage และ Network เป็นส่วนสำคัญในการติดตั้งระบบ Edge Cloud Computing Cluster ในแต่ละภูมิภาคของประเทศไทย ฮาร์ดแวร์ที่ใช้งานได้รับการคัดเลือกอย่างรอบคอบเพื่อรองรับความต้องการในการประมวลผลข้อมูลที่ซับซ้อนและมีปริมาณมาก ตัวอย่างเช่น การใช้เซิร์ฟเวอร์ที่มีหน่วยประมวลผลที่มีความเร็วสูง พื้นที่จัดเก็บข้อมูลขนาดใหญ่ และการเชื่อมต่อเครือข่ายที่มีความเร็วสูง เพื่อให้สามารถรองรับการทำงานของแอปพลิเคชัน IoT ที่หลากหลายได้อย่างมีประสิทธิภาพ

- **Central Site**
  - **Operations Support System (OSS)**

OSS ในโครงการนี้มีบทบาทสำคัญในการควบคุม วิเคราะห์ และตรวจสอบการทำงานของระบบเครือข่ายทั้งหมด โดยใช้ OpenStack Monasca สำหรับการ Monitoring และการแจ้งเตือนเกี่ยวกับสถานะของระบบ นอกจากนี้ OSS ยังมีความสามารถในการวิเคราะห์ข้อมูลเชิงลึกเกี่ยวกับการใช้งานเครือข่าย เช่น การตรวจจับความผิดปกติในระบบ การวิเคราะห์แนวโน้มของปริมาณการใช้งาน และการตรวจสอบประสิทธิภาพของทรัพยากรเครือข่ายทั้งหมด Contrail Controller ถูกนำมาใช้เพื่อช่วยบริหารจัดการการเชื่อมโยงเครือข่ายให้เป็นไปอย่างมีประสิทธิภาพ ช่วยลดเวลาในการแก้ไขปัญหาและเพิ่มความสามารถในการบริหารจัดการทรัพยากรได้แบบเรียลไทม์

  - **Management & Orchestration (MANO)**

ระบบ MANO ประกอบด้วยเครื่องมือสำคัญหลายส่วน ได้แก่ OpenStack Tacker, Senlin, และ Mistral ที่ช่วยในกระบวนการบริหารจัดการ Lifecycle ของ VNFs เช่น การติดตั้ง การปรับแต่ง และการอัปเดต VNFs ให้เป็นปัจจุบัน นอกจากนี้ OpenStack Nova, Neutron, และ Cinder ยังช่วยในกระบวนการจัดการทรัพยากรเสมือน (Virtual Resources) เช่น การจัดสรรทรัพยากรคอมพิวติ้ง พื้นที่จัดเก็บข้อมูล และการเชื่อมโยงเครือข่ายภายในระบบ

คลาวด์ ระบบ MANO ช่วยลดความซับซ้อนในการบริหารจัดการทรัพยากร และช่วยเพิ่มความคล่องตัวในการขยายขีดความสามารถของระบบให้รองรับความต้องการในอนาคตได้อย่างรวดเร็ว

ความเชื่อมโยงระหว่าง Edge Site และ Central Site ถูกออกแบบให้มีการทำงานร่วมกันอย่างเป็นระบบผ่านเครือข่าย SDN (Software-Defined Networking) Gateway ซึ่งช่วยให้การจัดการทรัพยากรและการประมวลผลเป็นไปอย่างยืดหยุ่นและมีประสิทธิภาพสูงสุด เครือข่าย SDN นี้ยังช่วยเพิ่มประสิทธิภาพของการรับส่งข้อมูล ลดความหน่วงเวลา และช่วยเพิ่มความปลอดภัยของข้อมูลโดยการควบคุมการเข้าถึงจากจุดเดียว Edge Site สามารถดำเนินการประมวลผลข้อมูลที่ต้องการความรวดเร็วได้ในทันที ขณะที่ Central Site ทำหน้าที่บริหารจัดการทรัพยากรทั้งหมดได้จากส่วนกลาง การเชื่อมต่อแบบบูรณาการนี้ช่วยให้ระบบสามารถตอบสนองต่อความต้องการที่หลากหลายของผู้ใช้งานได้อย่างมีประสิทธิภาพและยืดหยุ่นสูงสุด

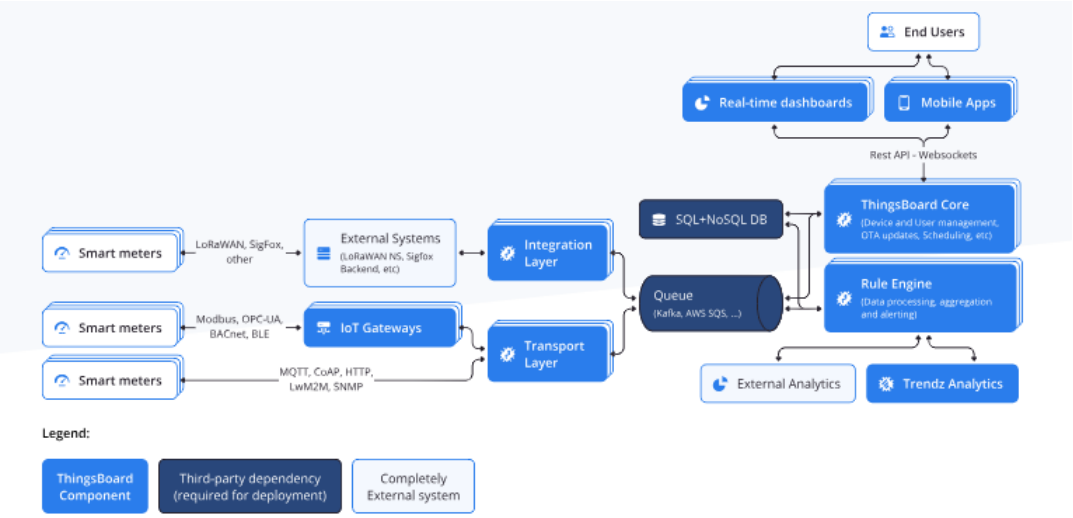
- **Thingsboard IoT Platform**

ในยุคที่เทคโนโลยี IoT ขยายตัวอย่างรวดเร็ว การเลือกใช้แพลตฟอร์ม IoT ที่เหมาะสมกับโครงสร้าง Full Edge Cloud Computing มีความสำคัญอย่างยิ่ง เนื่องจากเป็นหัวใจหลักในการจัดการข้อมูลที่มีความซับซ้อน ThingsBoard ซึ่งเป็นแพลตฟอร์มโอเพนซอร์สที่ถูกพัฒนาขึ้นเพื่อรองรับการเชื่อมต่อและประมวลผลข้อมูลจากอุปกรณ์ IoT ได้อย่างมีประสิทธิภาพ ถือเป็นเครื่องมือสำคัญที่ช่วยสร้างความยืดหยุ่นและความสามารถในการปรับตัวสำหรับโครงสร้างพื้นฐานด้าน Edge Cloud Computing

คุณสมบัติเด่นของ ThingsBoard

ด้วยการสนับสนุนโปรโตคอลที่หลากหลาย เช่น MQTT, CoAP และ HTTP พร้อมความสามารถในการรองรับอุปกรณ์ IoT หลากหลายประเภท ThingsBoard จึงตอบโจทย์การจัดการข้อมูลในระบบที่ต้องการความสามารถในการขยายตัวอย่างต่อเนื่อง นอกจากนี้ แพลตฟอร์มนี้ยังรองรับการประมวลผลข้อมูลแบบเรียลไทม์ ระบบควบคุมการเข้าถึงข้อมูลที่สามารถปรับแต่งได้ และการเพิ่มความปลอดภัยในระดับสูงสุด

รูปที่ 6 ตัวอย่าง Dashboard ระบบการจัดการพลังงานอัจฉริยะ



รูปที่ 7 แผนภาพสถาปัตยกรรมระบบจัดการพลังงานอัจฉริยะ

แผนภาพด้านบนแสดงถึงการไหลของข้อมูลและจุดบูรณาการที่มีความซับซ้อนและเชื่อมโยงกันอย่างมีประสิทธิภาพในโซลูชันพลังงานอัจฉริยะทั่วไป ระบบนี้ใช้ประโยชน์จากแพลตฟอร์ม ThingsBoard เพื่อทำหน้าที่รวบรวมและวิเคราะห์ข้อมูลพลังงานจากสมาร์ทมิเตอร์ที่ติดตั้งอยู่ในจุดต่าง ๆ ภายในเครือข่าย ทั้งนี้ ข้อมูลที่ถูกรวบรวมจะถูกส่งต่อและประมวลผลผ่านขั้นตอนหลายระดับเพื่อให้สามารถวิเคราะห์และสร้างการตัดสินใจที่มีประสิทธิภาพสูงสุด ระบบยังสามารถรองรับการบูรณาการเข้ากับแหล่งข้อมูลอื่น ๆ เช่น อุปกรณ์ IoT และเซ็นเซอร์ต่าง ๆ เพื่อเสริมสร้างความสามารถในการตรวจสอบและจัดการพลังงานได้อย่างครอบคลุมมากยิ่งขึ้น นอกจากนี้ การเชื่อมต่อข้อมูลในระบบยังช่วยให้สามารถคาดการณ์และเพิ่มประสิทธิภาพการใช้พลังงานได้ในระดับที่สูงกว่าเดิม รวมถึงสร้างความยั่งยืนให้กับโครงสร้างพลังงานในระยะยาว

Addons: เพิ่มขีดความสามารถของ ThingsBoard

1. **Trendz Analytics Platform (AI) : [11]** Addon ด้านปัญญาประดิษฐ์ที่ช่วยในงานวิเคราะห์และการคาดการณ์เชิงลึก เพิ่มศักยภาพให้ระบบตัดสินใจได้แม่นยำยิ่งขึ้น
2. **ThingsBoard Edge : [12]** เพิ่มความสามารถในการประมวลผลข้อมูล ณ จุดต้นทาง (Edge) โดยช่วยลดเวลาในการส่งข้อมูลไปยังเซิร์ฟเวอร์กลาง ทำให้ระบบตอบสนองได้รวดเร็วขึ้น
3. **IoT Gateway : [13]** รองรับการเชื่อมต่อกับอุปกรณ์ IoT หลากหลายประเภท ทำหน้าที่เป็นเกตเวย์กลางในการรวบรวมข้อมูลจากอุปกรณ์ต่าง ๆ ให้เป็นระบบและมีประสิทธิภาพ

ThingsBoard เป็นแพลตฟอร์ม IoT ที่มีศักยภาพสูงในการขับเคลื่อน Edge Cloud Computing ด้วยความสามารถที่ยืดหยุ่นและรองรับการขยายตัวได้ในอนาคต สิ่งนี้ช่วยให้องค์กรในภาคอุตสาหกรรมสามารถรับมือกับความท้าทายด้านข้อมูลในยุค IoT ได้อย่างมีประสิทธิภาพและปลอดภัย.

- **ThingsBoard Architecture : [14]**

ThingsBoard IoT Platform มีสถาปัตยกรรมที่ถูกออกแบบมาเพื่อรองรับการทำงานของระบบ Edge Cloud Computing อย่างเต็มรูปแบบ โดยแบ่งเป็นโครงสร้างที่สำคัญดังนี้:

- Device Connectivity Layer: รองรับการเชื่อมต่อจากอุปกรณ์ IoT ผ่านโปรโตคอลมาตรฐาน เช่น MQTT, CoAP และ HTTP ช่วยให้สามารถรวมข้อมูลจากอุปกรณ์หลากหลายประเภทเข้าด้วยกันได้อย่างมีประสิทธิภาพ รองรับการทำงานของอุปกรณ์รุ่นใหม่และรุ่นเก่าได้อย่างสมบูรณ์

- Data Processing Layer: ใช้ Rule Engine เพื่อจัดการการประมวลผลข้อมูล รองรับการวิเคราะห์ข้อมูลที่มีความซับซ้อน การสร้างเงื่อนไขและกฎเกณฑ์เพื่อการตัดสินใจอัตโนมัติ และการแจ้งเตือนที่ตรงต่อความต้องการ

- Data Storage Layer: ใช้ฐานข้อมูล PostgreSQL สำหรับการจัดเก็บข้อมูลขนาดใหญ่ที่ต้องการความรวดเร็วและปลอดภัย PostgreSQL มีความยืดหยุ่นสูงในการจัดการข้อมูลเชิงสัมพันธ์และรองรับการเพิ่มปริมาณข้อมูลได้อย่างต่อเนื่อง ซึ่งเหมาะสมสำหรับการใช้งานในระบบ IoT ที่ต้องการประสิทธิภาพระดับสูง

- Visualization and Integration Layer: รองรับการสร้างแดชบอร์ดและการแสดงผลข้อมูลในรูปแบบกราฟิกที่สามารถปรับแต่งได้ตามความต้องการของผู้ใช้งาน อีกทั้งยังรองรับการบูรณาการกับระบบภายนอก เช่น AI และ Machine Learning เพื่อเสริมความสามารถในการวิเคราะห์ข้อมูลเชิงลึก

- **บทบาทของ ThingsBoard ใน Edge Cloud Compute**

1.1 การจัดการอุปกรณ์ (Device Management):

- รองรับการเชื่อมต่ออุปกรณ์ IoT ผ่านโปรโตคอลมาตรฐาน เช่น MQTT, CoAP และ HTTP ซึ่งช่วยให้การเชื่อมต่ออุปกรณ์เป็นไปอย่างราบรื่นและปลอดภัย
- จัดการอุปกรณ์ในรูปแบบกลุ่มหรือรายอุปกรณ์ พร้อมสนับสนุนการตั้งค่าระยะไกล (Remote Configuration) เพื่อเพิ่มความสะดวกในการบริหารจัดการ
- สนับสนุนการอัปเดตเฟิร์มแวร์ระยะไกล ช่วยให้มั่นใจได้ว่าอุปกรณ์ทั้งหมดทำงานด้วยซอฟต์แวร์ที่ทันสมัยที่สุด
- มีแดชบอร์ดที่ใช้งานง่ายสำหรับการจัดการอุปกรณ์และนโยบายความปลอดภัย เช่น การควบคุมการเข้าถึงและการตรวจสอบกิจกรรม

1.2 การประมวลผลข้อมูล (Data Processing):

- Rule Engine ที่ยืดหยุ่นสำหรับการวิเคราะห์ข้อมูลแบบเรียลไทม์ รองรับการกำหนดกฎเกณฑ์ที่ซับซ้อนสำหรับการตอบสนองต่อข้อมูล
- รองรับการประมวลผลข้อมูลล่วงหน้าก่อนส่งไปยังระบบ Cloud เพื่อลดภาระงานของระบบส่วนกลาง
- สนับสนุนการสร้างแบบจำลองข้อมูลและการนำ AI/ML มาใช้เพื่อวิเคราะห์ข้อมูลเชิงลึก เช่น การพยากรณ์และการจัดการความเสี่ยง
- รองรับการแจ้งเตือนผ่านหลายช่องทาง เช่น อีเมล SMS และแอปพลิเคชัน

1.3 การแสดงผลข้อมูล (Data Visualization):

- มาพร้อมแดชบอร์ดที่สามารถปรับแต่งได้อย่างอิสระ ผ่าน HTML, CSS หรือการตั้งค่าการออกแบบเฉพาะทาง เพื่อให้ตรงกับความต้องการของผู้ใช้งาน
- รองรับการ Export ข้อมูลในรูปแบบต่าง ๆ เช่น PDF, CSV และ JSON เพื่อความสะดวกในการใช้งานและการวิเคราะห์
- ระบบสามารถตั้งค่าการแจ้งเตือนได้หลากหลายช่องทาง เช่น อีเมล, SMS, และการแจ้งเตือนผ่านแอปพลิเคชัน เพื่อให้ผู้ใช้สามารถรับข้อมูลสำคัญได้ทันท่วงที

การใช้งาน ThingsBoard IoT Platform ช่วยเพิ่มศักยภาพให้กับระบบ Edge Cloud Compute ทั้งในด้านการประมวลผล การจัดการข้อมูล และการบูรณาการกับระบบภายนอก สิ่งเหล่านี้สนับสนุนการพัฒนาอุตสาหกรรมดิจิทัลในประเทศไทย พร้อมรองรับการเปลี่ยนแปลงและความต้องการในอนาคตได้อย่างมีประสิทธิภาพ

3.2.3 ปัญหาและแนวทางแก้ไข

- **ปัญหา**

คณะทำงานวิจัยได้พบปัญหาระหว่างดำเนินการโครงการดังต่อไปนี้

1. ผู้ให้บริการ Colocation ในแต่ละภูมิภาคยังมีความพร้อมด้านเทคนิคไม่เพียงพอ ซึ่งทำให้เกิดความล่าช้าในการติดตั้งอุปกรณ์ตามแผนงานเดิม ส่งผลให้การพัฒนาระบบ Edge Cloud Computing ไม่ต่อเนื่อง และอาจกระทบกับการใช้งานในพื้นที่ต่าง ๆ

2. ขาดความพร้อมของหน่วยงานทั้งภาครัฐและภาคเอกชนที่มีความต้องการใช้งาน Full Edge Computing และการลงทุนที่สูงทำให้ไม่คุ้มค่าต่อการลงทุน

- **แนวทางแก้ไข**

จากปัญหาที่คณะทำงานวิจัยได้แจ้งไปรายละเอียดข้างต้น จึงมีแนวทางในการแก้ไขดังต่อไปนี้

การแก้ไขปัญหาของข้อที่ 1 คณะทำงานวิจัยได้ดำเนินการติดตั้งอุปกรณ์ Network Switching และ Server ของภาคเหนือ ภาคใต้ และภาคตะวันออกไว้ที่ Colocation จังหวัดนนทบุรี เพื่อใช้พื้นที่เหล่านี้เป็นศูนย์กลางสำหรับการประมวลผลข้อมูลของระบบ Edge Cloud Computing ที่ช่วยสนับสนุนการใช้งานในภูมิภาคต่าง ๆ อย่างมีประสิทธิภาพ และ

ติดตั้งอุปกรณ์ Network Switch และ Server ของภาคตะวันออกเฉียงเหนือไว้ที่จังหวัดขอนแก่น เพื่อให้สอดคล้องกับ Edge Computin ที่ดำเนินการพัฒนาและวิจัยซึ่งทำงานได้ตรงตามเป้าหมายของโครงการ

การแก้ไขปัญหาของข้อที่ 2 ได้ทำการออกแบบโครงสร้างระบบ Edge Cloud Computing เพื่อลดการลงทุนและตรงกับการใช้งานของภาคอุตสาหกรรม โดยแบ่งเป็น 2 รูปแบบ ดังนี้:

- **Full Edge Computing (รูปแบบดั้งเดิม)**

ใช้เป็นศูนย์กลางสำหรับการประมวลผลข้อมูล เก็บข้อมูลขนาดใหญ่ รองรับการเชื่อมต่อที่มีการใช้งานสูง และรองรับจัดการทั้งส่วนของ Compute, Storage และ Network แบบครบวงจร

- **IoT Edge**

เน้นการเก็บข้อมูลขนาดเล็กและประมวลผลสำหรับอุปกรณ์ IoT โดยจะเชื่อมต่อเพื่อเก็บข้อมูลต่างๆ จากอุปกรณ์ Sensor เช่น วัดอุณหภูมิความชื่น, Power Meter, PLC หรืออุปกรณ์ที่เชื่อมต่อโดยตรงกับ IoT Edge โดยลักษณะนี้จะช่วยให้ลดต้นทุนเนื่องจากอุปกรณ์ที่มีขนาดเล็ก ราคาไม่แพง ใช้งานพลังงานน้อย และประยุกต์ใช้กับภาคอุตสาหกรรมได้ดี การออกแบบดังกล่าวจะช่วยรองรับการใช้งานได้หลากหลาย ทั้งในระดับองค์กรและการใช้งานอุปกรณ์ IoT ในพื้นที่ต่าง ๆ ได้อย่างมีประสิทธิภาพและตอบโจทย์ทุกความต้องการ.

3.3 การวิจัยและพัฒนา IoT Edge

3.3.1 การกำหนดความต้องการของระบบ IoT Edge

- **การออกแบบขนาดกะทัดรัด:**

  มีขนาดเล็กน้ำหนักเบา และสามารถเชื่อมต่อกับเซ็นเซอร์ได้โดยตรง

- **รองรับ Thingsboard Edge / Thingsboard IoT Gateway:**

  - การประมวลผลข้อมูลเบื้องต้น: ระบบ Edge Node ช่วยลดภาระการส่งข้อมูลไปยังคลาวด์หลัก โดยประมวลผลข้อมูลและกรองข้อมูลสำคัญในพื้นที่ สามารถตรวจจับความผิดปกติและสรุปผลเบื้องต้นได้ทันที พร้อมทั้งการจัดเก็บข้อมูลสำคัญในสถานการณ์ที่อินเทอร์เน็ตไม่เสถียร
  - Local Dashboard: แสดงข้อมูลแบบเรียลไทม์ในพื้นที่ พร้อมทั้งการแจ้งเตือนเหตุการณ์สำคัญ ช่วยให้ผู้ใช้งานตรวจสอบและจัดการข้อมูลได้ทันที

- **การเชื่อมต่อหลากหลาย:**

  รองรับโปรโตคอลมาตรฐาน เช่น MQTT, CoAP, HTTP รวมถึงการเชื่อมต่อผ่าน RS485 และ Modbus เพื่อรองรับอุปกรณ์ IoT หลากหลายชนิดในภาคอุตสาหกรรม

3.3.2 การออกแบบโครงสร้างสถาปัตยกรรมภายในของ IoT Edge

- **Hardware**
  - **Raspberry Pi**

Raspberry Pi เป็นคอมพิวเตอร์ขนาดเล็กที่ออกแบบมาให้มีต้นทุนต่ำและประสิทธิภาพที่หลากหลาย เพื่อตอบสนองการศึกษา การทดลอง และการพัฒนานวัตกรรมเทคโนโลยี Raspberry Pi Foundation ได้ออกแบบอุปกรณ์นี้ให้สามารถรองรับการใช้งานได้อย่างยืดหยุ่น โดยอุปกรณ์ Raspberry Pi ที่รองรับการประมวลผลข้อมูลและการเชื่อมต่ออุปกรณ์ IoT รวมถึงเซ็นเซอร์หลากหลายชนิดผ่านโปรโตคอลมาตรฐาน เช่น I2C, SPI และ UART นอกจากนี้ยังสามารถติดตั้งระบบปฏิบัติการได้หลากหลาย เช่น Raspberry Pi OS และ Ubuntu อีกทั้งยังมีอุปกรณ์เสริม เช่น เคส (Case) ที่ช่วยปกป้องตัวเครื่องจากฝุ่นและความเสียหาย พร้อมรองรับการติดตั้งพัดลมระบายความร้อนหรือฮีตซิงก์ (Heatsink) เพื่อเพิ่มประสิทธิภาพการทำงานในสภาพแวดล้อมที่มีอุณหภูมิความร้อนสูง จึงทำให้ Raspberry Pi เป็นเครื่องมือสำคัญในโครงการ IoT และ Edge Computing ที่ต้องการการประมวลผลข้อมูลในพื้นที่ใกล้กับแหล่งข้อมูล อุปกรณ์นี้ยังมีขนาดเล็กและประหยัดพลังงาน

  - **Industrial IoT Expansion Module**

รูปที่ 8 Case Industrial IoT Expansion Module

Industrial IoT Expansion Module ถูกออกแบบมาเพื่อเพิ่มความสามารถในการเชื่อมต่อกับอุปกรณ์ IoT และเซ็นเซอร์หลากหลายชนิด พร้อมรองรับการใช้งานในสภาพแวดล้อมอุตสาหกรรม โดยมีสเปคหลักดังนี้

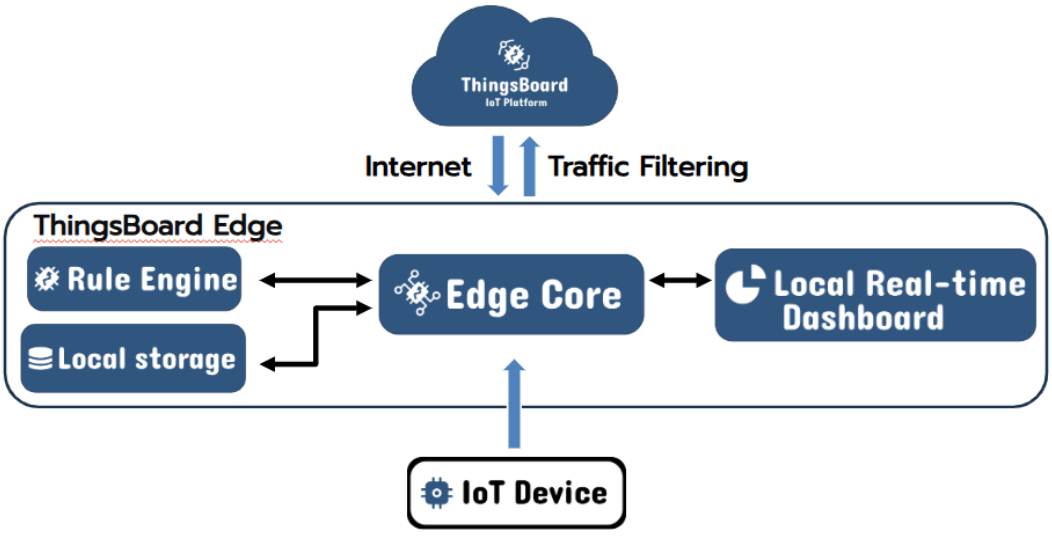
- **รองรับพอร์ตการเชื่อมต่อหลากหลาย:**
  - RS485 สำหรับการสื่อสารในอุตสาหกรรม
  - CAN Bus สำหรับระบบยานยนต์และการควบคุมอัตโนมัติ
  - GPIO แบบขยายเพื่อรองรับเซ็นเซอร์และอุปกรณ์หลากหลาย
- **การป้องกันสภาพแวดล้อม:**
  - รองรับการทำงานในอุณหภูมิระหว่าง -20°C ถึง 85°C
  - การออกแบบที่ทนต่อการสั่นสะเทือนและแรงกระแทก

- **รองรับการเชื่อมต่อไร้สาย:**
    - Wi-Fi และ Bluetooth 5.0 สำหรับการเชื่อมต่ออุปกรณ์ IoT สมัยใหม่

IoT Edge ที่ใช้งาน Raspberry Pi CM4 กับ Case Industrial IoT Expansion Module จพทำหน้าที่เป็น Gateway และ Edge ในตัวเดียวสำหรับการเชื่อมต่อและรวบรวมข้อมูลจากเซ็นเซอร์ภายในพื้นที่ โดยได้รับการออกแบบให้รองรับการติดตั้ง ThingsBoard Edge หรือ IoT Gateway ซึ่งช่วยเพิ่มศักยภาพในการประมวลผลข้อมูลในระดับ Edge Node ได้อย่างเต็มประสิทธิภาพ ไม่ว่าจะเป็นการกรองข้อมูล การจัดเก็บข้อมูลเบื้องต้น หรือการส่งข้อมูลไปยัง Edge Cloud Compute ระบบนี้รองรับการประมวลผลและตอบสนองแบบเรียลไทม์เพื่อลดความล่าช้า (Latency) ในกับระบบ IoT ให้ได้มากที่สุด

- Software
    - Thingsboard Edge

ThingsBoard Edge เป็นแพลตฟอร์ม IoT ที่ได้รับการออกแบบมาเพื่อรองรับการประมวลผลข้อมูลในระดับ Edge Node ซึ่งเป็นส่วนสำคัญของโครงสร้างระบบ IoT ในปัจจุบัน แพลตฟอร์มนี้สามารถทำงานเป็นส่วนขยายของระบบ ThingsBoard Cloud โดยช่วยให้สามารถจัดการอุปกรณ์ IoT และประมวลผลข้อมูลใกล้กับแหล่งกำเนิดข้อมูลได้อย่างมีประสิทธิภาพ ลดความจำเป็นในการส่งข้อมูลทั้งหมดไปยังคลาวด์ ซึ่งช่วยลดเวลาในการตอบสนอง (Latency) และเพิ่มความรวดเร็วในการตัดสินใจ นอกจากนี้ยังช่วยลดต้นทุนในการใช้งานแบนด์วิดท์และเพิ่มความปลอดภัยของข้อมูลในระดับ Edge Node โดยที่ข้อมูลที่สำคัญจะได้รับการกรองและประมวลผลเบื้องต้นก่อนถูกส่งต่อไปยังคลาวด์



รูปที่ 9 Thingsboard Edge Architecture

ด้วยการออกแบบระบบโครงสร้างที่คำนึงถึงความสามารถในการเชื่อมต่อของ ThingsBoard Edge รองรับการใช้งานร่วมกับอุปกรณ์ IoT หลากหลายประเภท ตั้งแต่เซ็นเซอร์ที่ใช้โปรโตคอลมาตรฐานไปจนถึงระบบอัตโนมัติที่มีความซับซ้อน การประมวลผลแบบกระจายทำให้แพลตฟอร์มนี้เหมาะสำหรับสภาพแวดล้อมที่ต้องการความเสถียรและปรับตัว เช่น โรงงานอัจฉริยะ ระบบพลังงานอัจฉริยะ และระบบขนส่งมวลชน แพลตฟอร์มนี้ยังสามารถซิงค์ข้อมูลกับ ThingsBoard Cloud Platform เพื่อการวิเคราะห์และจัดเก็บข้อมูลในระยะยาว พร้อมด้วยระบบแจ้งเตือนแบบเรียลไทม์ที่ช่วยเพิ่มความสะดวกรวดเร็วและความแม่นยำในการตอบสนองต่อเหตุการณ์ต่าง ๆ

คุณสมบัติหลักของ Thingsboard

1. **การประมวลผลข้อมูลในระดับ Edge:**
   ThingsBoard Edge ช่วยให้การประมวลผลข้อมูลใกล้แหล่งกำเนิดข้อมูลทำได้รวดเร็ว
   ลดเวลาในการตอบสนองต่อเหตุการณ์ต่าง ๆ เช่น
   การแจ้งเตือนความผิดปกติหรือการปรับตั้งค่าการทำงานของอุปกรณ์
2. **การบริหารจัดการอุปกรณ์ IoT:**
   ผู้ใช้งานสามารถกำหนดค่าอุปกรณ์ IoT และอัปเดตเฟิร์มแวร์ในระดับท้องถิ่นได้อย่างง่ายดาย
   โดยไม่ต้องพึ่งพาการเชื่อมต่อกับระบบคลาวด์ตลอดเวลา
   ระบบยังรองรับการจัดการสิทธิ์เพื่อเพิ่มความปลอดภัยในการใช้งาน
3. **การแสดงผลข้อมูล:**
   แพลตฟอร์มมีแดชบอร์ดในตัวที่สามารถปรับแต่งได้ตามความต้องการ
   ผู้ใช้งานสามารถติดตามข้อมูลแบบเรียลไทม์ ตรวจสอบสถานะของอุปกรณ์
   และวิเคราะห์ผลการประมวลผลได้ทันที
4. **การเชื่อมต่อและการทำงานร่วมกัน:**
   ThingsBoard Edge รองรับโปรโตคอลมาตรฐาน เช่น MQTT, CoAP, และ HTTP
   ทำให้การเชื่อมต่อกับอุปกรณ์ IoT หลากหลายชนิดเป็นไปอย่างยืดหยุ่น และสามารถใช้งานร่วมกับระบบ IoT เดิมที่มีอยู่ได้
5. **การซิงค์ข้อมูลกับ ThingsBoard Cloud:**

ข้อมูลที่สำคัญสามารถถูกส่งต่อไปยัง ThingsBoard Cloud เพื่อการวิเคราะห์ในระดับสูงหรือการจัดเก็บระยะยาว แพลตฟอร์มยังมีระบบสำรองข้อมูลเพื่อป้องกันการสูญเสียข้อมูลในกรณีที่การเชื่อมต่อเครือข่ายล้มเหลว

การใช้งานร่วมกับ Raspberry Pi กับ ThingsBoard Edge สามารถติดตั้งบน Raspberry Pi เพื่อสร้างระบบ IoT Edge ที่เหมาะสำหรับการประมวลผลข้อมูลภายในพื้นที่ โดย Raspberry Pi ทำหน้าที่เป็นฮาร์ดแวร์หลักที่รองรับการเชื่อมต่อและประมวลผลข้อมูลจากอุปกรณ์ IoT ผ่านพอร์ต GPIO, RS485 และโปรโตคอลมาตรฐาน

การใช้งานนี้เหมาะสำหรับโครงการที่ต้องการความยืดหยุ่น เช่น ระบบ IoT ในอุตสาหกรรม ระบบตรวจวัดในฟาร์มอัจฉริยะ หรือการจัดการพลังงานในอาคาร โดย ThingsBoard Edge จะช่วยเพิ่มประสิทธิภาพ และลดต้นทุนในการจัดการข้อมูลแบบเรียลไทม์

3.4 การออกแบบการทดสอบ

3.4.1 วัตถุประสงค์ของการทดสอบ

- ตรวจสอบความถูกต้องของระบบ: เพื่อประเมินว่าระบบ Edge Cloud Computing และ Mobile Edge Platform (MEP) สามารถทำงานได้ตามข้อกำหนดทางเทคนิค รวมถึงการตรวจสอบฟังก์ชันการทำงานของ DNS, การจัดการเครือข่าย, และการบริหารจัดการเซิร์ฟเวอร์ เพื่อให้มั่นใจว่าทุกฟังก์ชันทำงานได้อย่างถูกต้องและตรงตามมาตรฐานที่กำหนด
- วัดประสิทธิภาพของระบบ: เพื่อวิเคราะห์ศักยภาพของระบบในหลากหลายสถานการณ์ เช่น ความล่าช้า (Latency), ความสามารถในการจัดการแบนด์วิดท์ที่แตกต่างกันในแต่ละแอปพลิเคชัน, และการรองรับการใช้งานภายใต้สภาวะโหลดที่แตกต่างกัน นอกจากนี้ยังรวมถึงการวัด Throughput และการตอบสนองของระบบในเงื่อนไขที่ซับซ้อน
- ประเมินความเสถียร: เพื่อทดสอบว่า MEP และแอปพลิเคชันบุคคลที่สามสามารถทำงานได้อย่างมีเสถียรภาพในสถานการณ์ต่าง ๆ เช่น การทำงานภายใต้ภาระงานสูง การทำงานต่อเนื่องในระยะยาว และการรองรับความล้มเหลวที่อาจเกิดขึ้นในระบบ เพื่อวิเคราะห์ความพร้อมของระบบสำหรับการใช้งานจริง
- ประเมินความปลอดภัยของระบบ: เพื่อให้มั่นใจว่าระบบมีมาตรการรักษาความปลอดภัยที่เพียงพอ เช่น การป้องกันการเข้าถึงที่ไม่ได้รับอนุญาต การตรวจสอบสิทธิ์ผู้ใช้งาน และการเข้ารหัสข้อมูลระหว่างการสื่อสาร นอกจากนี้ยังประเมินการจัดการช่องโหว่และการป้องกันการโจมตีทางไซเบอร์
- การตรวจสอบความสามารถในการจัดการทรัพยากร: เพื่อประเมินความสามารถของ MEAO ในการจัดการทรัพยากร เช่น การเพิ่มหรือลดขนาดทรัพยากร (Scale In/Out), การสำรองข้อมูลที่ไม่กระทบต่อการทำงาน และการกู้คืนข้อมูลในกรณีฉุกเฉิน นอกจากนี้ยังรวมถึงการจัดการทรัพยากรสำหรับแอปพลิเคชันบุคคลที่สามเพื่อเพิ่มความยืดหยุ่นและลดความซับซ้อนในการปฏิบัติงาน

ตารางที่ 12 คุณสมบัติของ MEP (MEP Capability)

กรณีทดสอบ (Test Cases)	ผลลัพธ์ที่คาดหวัง (Expected Result)
1.1 การแก้ไขชื่อโดเมน DNS (DNS Domain Name Resolution Function)	อุปกรณ์ UE แต่ละตัวได้รับ IP ที่แตกต่างกัน
1.2 การตอบกลับแบบ Polling ของ DNS (DNS Polling Response Function)	ระบบตอบกลับด้วย IP ปลายทางหลายตัว
1.3 การกระจายโหลดของ DNS (DNS Load Balancing Function)	อุปกรณ์ UE เดียวกันใน session ต่างกันได้รับ IP ต่างกัน
1.4 การทำงานแบบ Recursive ของ DNS (DNS Recursive Function)	ส่งคำขอไปยัง DNS ภายนอกเมื่อไม่มีข้อมูลในระบบท้องถิ่น
1.5 การจัดการแบนด์วิดท์ระดับแอปพลิเคชัน (Application-Level Bandwidth Management)	แอปพลิเคชันแต่ละตัวได้รับแบนด์วิดท์ที่แตกต่างกัน
1.6 การจัดการแบนด์วิดท์ระดับ Session (Session-Level Bandwidth Management)	session แต่ละตัวได้รับแบนด์วิดท์ที่แตกต่างกัน
1.7 การทำงานของ NAT (NAT Function)	ระบบเปลี่ยน IP ต้นทางได้อย่างถูกต้อง

ตารางที่ 13 ตารางแสดงการจัดการ 3rd Party Application Management

กรณีทดสอบ (Test Cases)	ผลลัพธ์ที่คาดหวัง (Expected Result)
2.1 การปรับใช้แอปพลิเคชันจากบุคคลที่สาม (Deploy 3rd Party Application)	MEAO สามารถจัดการและปรับใช้แอปพลิเคชันจากบุคคลที่สาม พร้อมแสดงผลที่ใช้งานง่าย
2.2 การลดขนาดทรัพยากร (Scale In Resources)	MEAO สามารถลดขนาดทรัพยากรสำหรับแอปพลิเคชันบุคคลที่สามได้
2.3 การขยายขนาดทรัพยากร (Scale Out Resources)	MEAO สามารถขยายขนาดทรัพยากรสำหรับแอปพลิเคชันบุคคลที่สามได้
2.4 การสร้าง Snapshot/Clone ของ Volume (Volume Snapshot/Clone)	MEAO สามารถจัดการโคลน VM ในระบบ MEC ได้
2.5 การจัดการทรัพยากรของบุคคลที่สามหลายตัว (Multi 3rd Party Resource Management)	แยกและจัดการทรัพยากรให้กับแอปพลิเคชันบุคคลที่สามได้อย่างถูกต้องตามที่กำหนด
2.6 การจัดการผู้ใช้งาน (User Management)	ระบบรองรับการแยกผู้ใช้งานระดับ Admin และ Super Admin สำหรับการจัดการแอปพลิเคชัน

2.7 การจัดการเครือข่ายสำหรับบุคคลที่สามหลายตัว (Multi 3rd Party Network Management)	รองรับการจัดการ VIP หรือ NAT ไปยัง IP สาธารณะสำหรับแอปพลิเคชันแต่ละตัว
2.8 การระงับหรือบล็อกบริการแอปพลิเคชัน (Suspend/Block 3rd Party Applications)	ระบบสามารถระงับหรือคืนค่าการทำงานของแอปพลิเคชันบุคคลที่สามได้ชั่วคราว
2.9 การจัดการคลัสเตอร์ (Cluster Management)	รองรับการจัดการโหลดบาลานซ์แบบ N+1 หรือ 2N
2.10 การสำรองข้อมูลแอปพลิเคชันบุคคลที่สาม (Backup 3rd Party Application)	ระบบสามารถสำรองข้อมูลได้โดยไม่รบกวนการบริการ
2.11 การกู้คืนแอปพลิเคชันบุคคลที่สาม (Restore 3rd Party Application)	ระบบสามารถกู้คืนแอปพลิเคชันได้โดยไม่รบกวนการบริการ
2.12 การจัดการการแจ้งเตือนทรัพยากร (Alarm Resource Management)	ระบบสามารถแยกการแจ้งเตือนสำหรับแอปพลิเคชันบุคคลที่สามแต่ละตัวได้ โดยระบุประเภทของทรัพยากรที่เกี่ยวข้อง เช่น การแจ้งเตือนเกี่ยวกับการใช้งาน CPU, หน่วยความจำ หรือพื้นที่จัดเก็บข้อมูล ทั้งนี้เพื่อให้การบริหารจัดการแอปพลิเคชันบุคคลที่สามมีความแม่นยำและสามารถระบุปัญหาได้อย่างรวดเร็ว นอกจากนี้ยังสนับสนุนการตั้งค่าการแจ้งเตือนเฉพาะสำหรับแต่ละแอปพลิเคชันตามความต้องการของผู้ดูแลระบบ

การทดสอบระบบ Edge Cloud Computing และ Mobile Edge Platform (MEP) แสดงให้เห็นว่าระบบมีศักยภาพในการรองรับการใช้งานจริงในอุตสาหกรรม โดยสามารถตอบสนองความต้องการด้านประสิทธิภาพ ความเสถียร ความปลอดภัย และการจัดการทรัพยากรได้อย่างมีประสิทธิภาพ ระบบรองรับการทำงานของฟังก์ชันหลัก เช่น DNS, NAT, และการจัดการเครือข่ายได้ตรงตามมาตรฐาน นอกจากนี้ ยังสามารถดำเนินการสำรองและกู้คืนข้อมูล รวมถึงปรับขนาดทรัพยากรได้อย่างเหมาะสม สะท้อนถึงความพร้อมสำหรับการพัฒนาและปรับปรุงต่อไปในอนาคต เพื่อเพิ่มขีดความสามารถในการแข่งขันในยุคดิจิทัล.

บทที่ 4
ผลการวิจัย และการวิจารณ์ผล

4.1 ผลการวิจัย และการวิจารณ์ผล

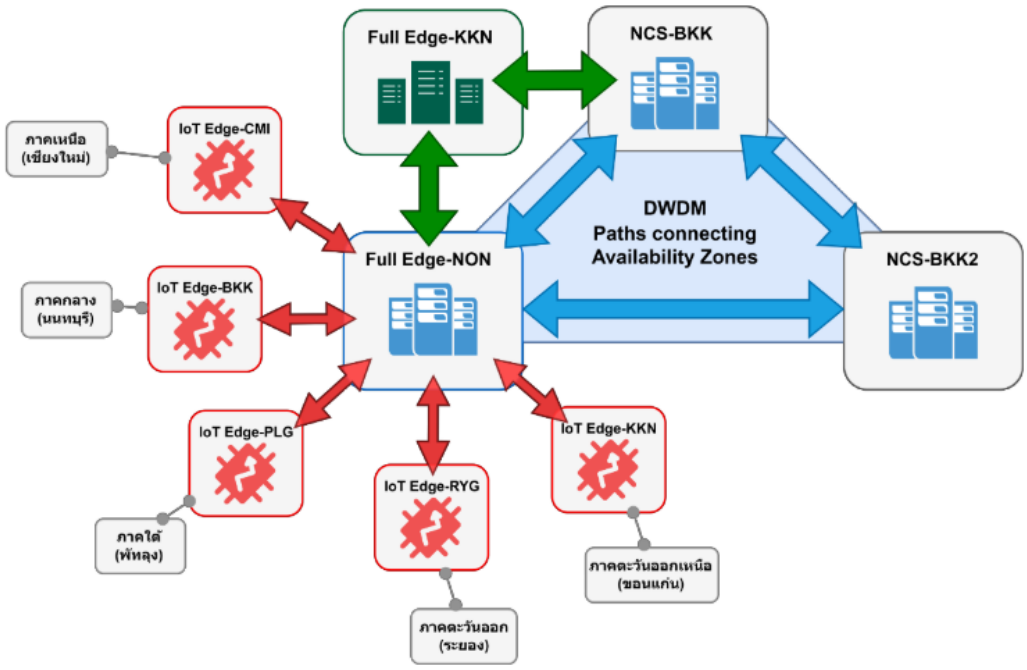
โครงการนี้มีการดำเนินงานและพัฒนาภายใต้กรอบเป้าหมายและวัตถุประสงค์ที่กำหนดไว้อย่างชัดเจนและเป็นระบบ โดยในกระบวนการดำเนินการนั้นได้มีการวางแผนเชิงกลยุทธ์ที่ครอบคลุมทุกแง่มุมของการวิจัยและพัฒนา ตั้งแต่การกำหนดเป้าหมายเชิงกลยุทธ์ที่ชัดเจน การกำหนดขั้นตอนปฏิบัติการที่ครอบคลุม และการใช้ทรัพยากรอย่างมีประสิทธิภาพ ทั้งนี้โครงการได้พิจารณาถึงความซับซ้อนในแต่ละขั้นตอนและจัดสรรบุคลากรผู้เชี่ยวชาญในสาขาต่าง ๆ เพื่อร่วมกันพัฒนาผลลัพธ์ที่มีคุณภาพสูงและตอบสนองต่อความต้องการที่หลากหลายของภาคส่วนต่าง ๆ ที่เกี่ยวข้อง การดำเนินงานในลักษณะนี้ช่วยให้โครงการสามารถบรรลุผลลัพธ์ที่สอดคล้องกับเป้าหมายและสร้างผลกระทบเชิงบวกที่มีความยั่งยืนต่อภาคส่วนต่าง ๆ ที่เกี่ยวข้องในเชิงเศรษฐกิจและเทคโนโลยี ผลการวิจัยที่ได้รับจึงเป็นผลลัพธ์ของความพยายามและความร่วมมือจากทีมงานผู้เชี่ยวชาญที่มุ่งมั่นในการบรรลุวัตถุประสงค์ที่สำคัญของโครงการ พร้อมทั้งมีการติดตามประเมินผลอย่างต่อเนื่องเพื่อปรับปรุงและพัฒนาให้สอดคล้องกับเป้าหมายที่กำหนดไว้ ซึ่งมีรายละเอียดดังต่อไปนี้:

4.1.1 ผลการออกแบบและพัฒนาเทคโนโลยี Edge Cloud Computing

การออกแบบและพัฒนา Edge Cloud Computing ซึ่งเป็นเทคโนโลยีในการกระจายการให้บริการ Infrastructure as as Service (IaaS) ในพื้นที่หรือสถานที่ตามความจำเป็นที่ต้องการใช้งานได้ เช่น พื้นที่ภายในโรงงานอุตสาหกรรม, พื้นที่ของ Data Center หรือ colocation เป็นต้น ซึ่งทำให้ผู้ใช้งานสามารถเชื่อมต่อข้อมูลและประมวลผลข้อมูลภายในพื้นที่นั้นๆ ได้อย่างมีประสิทธิภาพ ทำให้ลดระยะเวลาในการส่งข้อมูล การประมวลผลข้อมูล มีความปลอดภัยของข้อมูลเพราะข้อมูลถูกเก็บและประมวลผลภายในพนพื้นที่ และเพื่อให้สอดคล้องกับขนาดของภาคธุรกิจตั้งแต่ขนาดเล็ก กลาง และขนาดใหญ่ จึงมีการออกแบบและพัฒนา Edge Cloud Computing เป็น 2 ลักษณะคือ

1. Full Edge Computing เหมาะสมกับภาคธุรกิจขนาดกลางและขนาดใหญ่ เพราะมีการทำงานที่ครอบคลุมในการให้บริการ IaaS รองรับการประมวลผลข้อมูลที่มีขนาดใหญ่ และต้องมีงบลงทุนที่สูง
2. IoT Edge เหมาะสมกับภาคธุรกิจขนาดเล็กและขนาดกลาง เพราะมีการทำงานที่ไม่ซับซ้อน ประมวลผลข้อมูลขนาดเล็ก และมีงบลงทุนที่ไม่สูง

โครงสร้างระบบ Edge Cloud Computing และ IoT Edge



รูปที่ 10 รูปแบบการติดตั้งของ Edge Cloud Computing

ในโครงสร้างระบบ Edge Cloud Computing จะมีการออกแบบให้เห็นการเชื่อมต่อและการทำงานของ Full Edge Computing และ IoT Edge โดยมีรายละเอียดของแต่ละประเภทดังนี้

Full Edge Computing

- การทำงานในระดับ Infrastructure as a service (IaaS) โดยมีการทำงานดังต่อไปนี้
  - Compute (Virtualization)
    - Dynamic Resource Management (CPU, Memory)
    - Live Migration
  - Networking
    - Virtual Private Cloud (VPC)
    - External IP (Public IP)
    - Firewall L2 and L3 (Security Group)
    - VPC direct connect (Private IP)

- Storage
  - Block Storage (Volume)
  - Backup
  - Restore
- Management UI with NIPA Cloud Portal

- ติดตั้ง Full Edge Computing ที่จังหวัดนนทบุรี (NON) อยู่ในภูมิภาคภาคกลางและปริมลฑล และจังหวัดขอนแก่น (KKN) อยู่ในภูมิภาคตะวันออกเฉียงเหนือ
- ประโยชน์หลักของ Full Edge Computing มีดังนี้
  - ลดเวลาแฝงหรือเวลาหน่วง (Reduced Latency)
    เนื่องจากข้อมูลจะถูกเก็บและประมวลผลใกล้กับของผู้ใช้งานข้อมูล
  - เพิ่มความปลอดภัยของข้อมูล (Better Data Security)
    เนื่องจากข้อมูลจะถูกจัดเก็บไว้ในสถานที่เฉพาะ มีความปลอดภัย
    และมีการเข้าถึงได้เฉพาะผู้มีสิทธิเท่านั้น ลดความเสี่ยงจากการโจมตีทางไซเบอร์
  - การวิเคราะห์ข้อมูลแบบเรียลไทม์ (Real-Time Analytics)
    เนื่องจากข้อมูลจะถูกเก็บและประมวลผลใกล้กับของผู้ใช้งานข้อมูล
    ซึ่งทำให้เหมาะสมกับการที่ต้องการตัดสินใจรวดเร็ว เช่น ระบบขับเคลื่อนแบบไร้คนขับ
    หรือการควบคุมเครื่องจักรในโรงงานอุตสาหกรรม
  - ความคุ้มทุน (Cost-Effectiveness) หากเปรียบเทียบกับการใช้งาน public cloud
    พบว่ามีการใช้งาน bandwidth
    น้อยลงเนื่องจากไม่มีความจำเป็นต้องส่งข้อมูลไปประมวลผลและส่งกลับมา ทำให้ลดค่าใช้จ่าย
  - ความสามารถในการปรับลดขนาด (Enhanced Scalability)
    เนื่องจากเป็นการจัดการทรัพยากรหลายๆ เครื่องมาเป็นทรัยกรแบบรวมศูนย์
    ทำให้สามารถเพิ่มหรือลดขนาดหรือจำนวนของทรัพยากรที่ต้องการได้ เช่น Disk, RAM เป็นต้น
  - เพิ่มความเป็นเจ้าของข้อมูล (Improved Data Privacy)
    เนื่องจากข้อมูลจะถูกจัดเก็บไว้ในสถานที่เฉพาะ มีความปลอดภัย
    และมีการเข้าถึงได้เฉพาะผู้มีสิทธิ ทำให้ข้อมูลของผู้ใช้งานมีความเป็นส่วนตัวมากขึ้น

IoT Edge

- อุปกรณ์คอมพิวเตอร์ขนาดเล็กถูกติดตั้งให้กระจายตัวอยู่ในพื้นที่ภูมิภาค เช่น เชียงใหม่ (CMI), กรุงเทพฯ (BKK), ขอนแก่น (KKN), พัทลุง (PLG) และระยอง (RYG) IoT Edge มีหน้าที่รวบรวมข้อมูลจากอุปกรณ์ sensor ที่อยู่ในพื้นที่ รวมถึงดำเนินการประมวลผลข้อมูลเบื้องต้น
  โดยทำการกรองหรือวิเคราะห์ข้อมูลในขั้นตอนแรก
- การทำงานหลักของ IoT Edge มีดังต่อไปนี้
  - เก็บข้อมูลจากอุปกรณ์ sensor โดยเก็บไว้ในตัวอุปกรณ์เอง (local storage) กรณีเมื่อมี Internet เกิดขัดข้องยังสามารถเก็บข้อมูลได้ตามปกติ ข้อมูลไม่สูญหาย

- แบ่งแยกข้อมูลที่จะทำการส่งจาก IoT Edge ไปยัง IoT platform โดยสามารถเลือกข้อมูลที่ต้องส่งตามที่ต้องการได้
- ทำการแสดงผลแจ้งเตือนในกรณีที่ต้องการได้ เช่น
  แจ้งเตือนเมื่อมีอุณหภูมิเกินกว่าที่กำหนดในพื้นที่ควบคุม

- ประโยชน์หลักของ IoT Edge มีดังต่อไปนี้
  - ลดเวลาแฝงหรือเวลาหน่วง (Reduced Latency)
    เนื่องจากข้อมูลจะถูกเก็บและประมวลผลใกล้กับของผู้ใช้งานข้อมูล
  - เพิ่มความปลอดภัยของข้อมูล (Better Data Security)
    เนื่องจากข้อมูลจะถูกจัดเก็บไว้ในสถานที่เฉพาะ มีความปลอดภัย
    และมีการเข้าถึงได้เฉพาะผู้มีสิทธิเท่านั้น ลดความเสี่ยงจากการโจมตีทางไซเบอร์
  - การวิเคราะห์ข้อมูลแบบเรียลไทม์ (Real-Time Analytics)
    เนื่องจากข้อมูลจะถูกเก็บและประมวลผลใกล้กับของผู้ใช้งานข้อมูล
    ซึ่งทำให้เหมาะสมกับการที่ต้องการตัดสินใจรวดเร็ว เช่น ระบบขับเคลื่อนแบบไร้คนขับ
    หรือการควบคุมเครื่องจักรในโรงงานอุตสาหกรรม
  - ความคุ้มทุน (Cost-Effectiveness) หากเปรียบเทียบกับการใช้งาน public cloud
    พบว่ามีการใช้งาน bandwidth
    น้อยลงเนื่องจากไม่มีความจำเป็นต้องส่งข้อมูลไปประมวลผลและส่งกลับมา ทำให้ลดค่าใช้จ่าย

DWDM Paths

- การเชื่อมต่อเครือข่ายระหว่าง Full Edge Computing ใช้เทคโนโลยี DWDM (Dense Wavelength Division Multiplexing) ทำหน้าที่รองรับการส่งข้อมูลระหว่าง Full Edge และ Availability Zones เช่น NCS-BKK และ NCS-BKK2
- การทำงานหลักของ DWDM มีดังต่อไปนี้
  - การมัลติเพล็กซ์ความยาวคลื่น (Wavelength Multiplexing) โดย DWDM
    ส่งข้อมูลหลายช่องสัญญาณในรูปของคลื่นแสงที่มีความยาวคลื่นต่างกัน
    โดยแต่ละความยาวคลื่นสามารถส่งข้อมูลได้อิสระ
  - การแปลงสัญญาณไฟฟ้าเป็นแสง โดยทำการแปลงสัญญาณข้อมูลจากแหล่งต่าง ๆ
    ถูกแปลงให้เป็นแสงด้วยเลเซอร์ที่สร้างคลื่นแสงที่ความยาวคลื่นเฉพาะ
  - การรวมและส่งข้อมูล โดยตัวรวมสัญญาณ (Multiplexer) จะรวบรวมคลื่นแสงจากแหล่งต่าง ๆ
    ลงบนไฟเบอร์ออฟติกเส้นเดียว
  - การแยกสัญญาณที่ปลายทาง โดยเมื่อส่งข้อมูลถึงปลายทาง ตัวแยกสัญญาณ (Demultiplexer)
    จะแยกคลื่นแสงตามความยาวคลื่น และส่งข้อมูลไปยังผู้รับที่เกี่ยวข้อง
  - การขยายสัญญาณ โดยจะมีการใช้ตัวขยายสัญญาณ (Optical Amplifiers) เช่น EDFA
    (Erbium-Doped Fiber Amplifiers)

- ประโยชน์หลักของ DWDM มีดังต่อไปนี้
    - เพิ่มแบนด์วิดท์ โดยรองรับการส่งข้อมูลความจุสูงมากบนไฟเบอร์ออปติกเส้นเดียว
    - ลดต้นทุน โดยใช้ไฟเบอร์ออปติกที่มีอยู่ให้คุ้มค่ามากขึ้น ลดความจำเป็นในการติดตั้งไฟเบอร์ใหม่
    - รองรับการขยายตัวในอนาคต สามารถเพิ่มจำนวนช่องสัญญาณ (ความยาวคลื่น) ได้ง่าย โดยไม่ต้องเปลี่ยนโครงสร้างเครือข่ายเดิม
    - รองรับโปรโตคอลหลากหลาย สามารถใช้งานได้กับหลายโปรโตคอล เช่น Ethernet, SONET/SDH, ATM และอื่น ๆ ในเครือข่ายเดียวกัน
    - การส่งข้อมูลในระยะไกล โดย DWDM ช่วยให้สามารถส่งข้อมูลในระยะทางไกลโดยไม่สูญเสียคุณภาพของสัญญาณ
    - การแยกอิสระของช่องสัญญาณ โดยแต่ละความยาวคลื่นทำงานได้อิสระจากกัน ทำให้จัดการและดูแลได้ง่าย
    - ความยืดหยุ่นในการปรับแต่ง โดยสามารถกำหนดช่องสัญญาณเพื่อใช้เฉพาะงานหรือบริการได้ เช่น แยกช่องสำหรับข้อมูลเสียง วิดีโอ และข้อมูลทั่วไป
    - ประสิทธิภาพสูงในการใช้พลังงาน โดยใช้พลังงานต่ำกว่าเมื่อเทียบกับการใช้หลายไฟเบอร์แยกกัน

โครงสร้างดังกล่าวช่วยให้ระบบ Edge Cloud Computing มีประสิทธิภาพสูงสุด รองรับการขยายตัวในอนาคต และเพิ่มความมั่นคงปลอดภัยให้กับข้อมูลที่ประมวลผลและจัดเก็บไว้ในแต่ละโหนด

จุดเด่นของโครงสร้าง

- การกระจายศูนย์กลางข้อมูลช่วยลดภาระบนเครือข่ายหลัก (Core Network)
- เพิ่มความเร็วในการตอบสนองของระบบเนื่องจากข้อมูลส่วนใหญ่ถูกประมวลผลที่ IoT Edge ก่อน
- ความมั่นคงปลอดภัยของข้อมูลเพิ่มขึ้นด้วยการประมวลผลในพื้นที่ใกล้เคียง
- รองรับการขยายตัวของระบบในอนาคตด้วยโครงสร้างแบบโมดูลาร์ (Modular)

เทคโนโลยีที่ใช้

- **OpenStack**

OpenStack เป็นแพลตฟอร์ม Open Source ที่มีบทบาทสำคัญในการจัดการโครงสร้างพื้นฐานของระบบคลาวด์ โดยช่วยบริหารจัดการทรัพยากรด้านการประมวลผล หน่วยความจำ และพื้นที่จัดเก็บข้อมูลในลักษณะการกระจายตัว (Distributed) ทำให้สามารถรองรับการขยายตัวของระบบได้อย่างมีประสิทธิภาพ และสนับสนุนการประมวลผลแบบเรียลไทม์อย่างเต็มรูปแบบ นอกจากนี้ OpenStack ยังช่วยลดความซับซ้อนในการจัดการโครงสร้างพื้นฐานและรองรับการทำงานร่วมกับแพลตฟอร์มและเครื่องมืออื่น ๆ เช่น Kubernetes และ OpenSDN เพื่อเพิ่มความยืดหยุ่นของระบบโดยรวม

- **OpenSDN**

OpenSDN หรือ Open Software-Defined Networking เป็นเทคโนโลยีสำหรับการจัดการอุปกรณ์เครือข่ายด้วยซอฟต์แวร์ เพื่อเพิ่มให้ง่ายต่อการจัดการเครือข่ายที่มีขนาดใหญ่และอุปกรณ์เครือข่ายที่อยู่หลายพื้นที่ให้เกิดความง่ายและยืดหยุ่น ซึ่งเหมาะกับการนำมาเพื่อพัฒนา Edge Cloud Computing เทคโนโลยีนี้ช่วยให้ผู้ดูแลระบบสามารถกำหนดนโยบายเครือข่ายได้อย่างแม่นยำถูกต้อง และควบคุมการรการจัดการทราฟฟิกหรือการเข้าถึงได้อย่างมีประสิทธิภาพ โดยรองรับโปรโตคอลมาตรฐาน เช่น BGP, MPLS, XMPP เป็นต้น ซึ่งช่วยเพิ่มศักยภาพของเครือข่ายให้สามารถตอบสนองต่อความต้องการที่เปลี่ยนแปลงได้อย่างรวดเร็ว OpenSDN ยังช่วยลดความยุ่งยากในการจัดการโครงสร้างเครือข่ายแบบดั้งเดิม และเพิ่มความมั่นคงปลอดภัยของระบบโดยรวม

- **SD-WAN (Software-Defined Wide Area Network)**

SD-WAN
เป็นเทคโนโลยีที่มุ่งเน้นการปรับปรุงประสิทธิภาพและเพิ่มความยืดหยุ่นของการเชื่อมต่อเครือข่ายระหว่างโหนดในระบบด้วยความสามารถในการกำหนดเส้นทางในการส่งข้อมูลได้แบบอัตโนมัติตามสภาวะเครือข่ายในขณะนั้น ทำให้ SD-WAN เหมาะสมอย่างยิ่งสำหรับการใช้งานในระบบที่ต้องการการเชื่อมต่อที่มั่นคงและมีความเร็วสูง เช่น ระบบ Edge Cloud Computing ซึ่งมีการกระจายตัวในหลากหลายภูมิภาค

ด้วยการปรับแต่งเครือข่ายได้แบบเรียลไทม์ SD-WAN ช่วยลดความซับซ้อนในการบริหารเครือข่ายและลดต้นทุนโครงสร้างพื้นฐาน อีกทั้งยังช่วยเพิ่มความเสถียรของข้อมูลที่ส่งผ่านเครือข่ายขนาดใหญ่ การใช้ SD-WAN ในระบบ Edge Cloud Computing ไม่เพียงช่วยให้การสื่อสารระหว่างโหนดเป็นไปอย่างราบรื่น แต่ยังช่วยปรับปรุงความเร็วและลดความล่าช้า ทำให้สามารถตอบสนองความต้องการของแอปพลิเคชันที่มีความสำคัญสูงได้อย่างมีประสิทธิภาพ

นอกจากนี้ SD-WAN ยังมีความสามารถในการจัดการทราฟฟิกในระดับสูง สามารถกำหนดเส้นทางการไหลของข้อมูลให้เหมาะสมกับความต้องการใช้งานในแต่ละช่วงเวลาได้อย่างมีประสิทธิภาพ โดยเฉพาะการใช้งานที่เกี่ยวข้องกับการส่งข้อมูลที่มีความสำคัญและต้องการความแม่นยำสูง เช่น การสื่อสารระหว่างศูนย์ข้อมูลหลักและโหนด Edge ซึ่งเป็นหัวใจสำคัญของระบบ Edge Cloud Computing

- **DWDM (Dense Wavelength Division Multiplexing)**

DWDM เป็นเทคโนโลยีที่ออกแบบมาเพื่อการส่งข้อมูลผ่านแสงด้วยความเร็วสูงและรองรับการแบ่งช่องสัญญาณได้หลายร้อยช่องในเส้นทางเดียว ทำให้สามารถจัดการกับปริมาณข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ โดยเฉพาะในระบบ Edge Cloud Computing ซึ่งมีการเชื่อมต่อระหว่างโหนดหลักและโหนดรองในระยะทางไกล DWDM

ช่วยลดปัญหาความหน่วงเวลา เพิ่มเสถียรภาพในการส่งข้อมูล และทำให้การเชื่อมต่อระหว่างโหนดต่าง ๆ เป็นไปอย่างราบรื่น เทคโนโลยีนี้ยังรองรับการทำงานในเครือข่ายที่มีปริมาณทราฟฟิกสูง
โดยช่วยให้ข้อมูลสามารถส่งต่อได้อย่างต่อเนื่องและมีประสิทธิภาพสูงสุด แม้ในสภาวะที่มีข้อจำกัดด้านโครงสร้างเครือข่าย

4.1.2 ผลการติดตั้งฮาร์ดแวร์และซอฟต์แวร์

รายงานผลการติดตั้งฮาร์ดแวร์

รูปที่ 11 การตรวจเช็คเครื่องก่อนนำเข้าติดตั้งที่ไซต์นนทบุรี (1)

รูปที่ 12 การตรวจเช็คเครื่องก่อนนำเข้าติดตั้งที่ไซต์นนทบุรี (2)

รูปที่ 13 สวิตช์ และเครื่องแม่ข่ายที่ดำเนินการติดตั้งไซต์นนทบุรี

รูปที่ 14 สวิตช์ และเครื่องแม่ข่ายที่ดำเนินการติดตั้งไซต์ขอนแก่น (1)

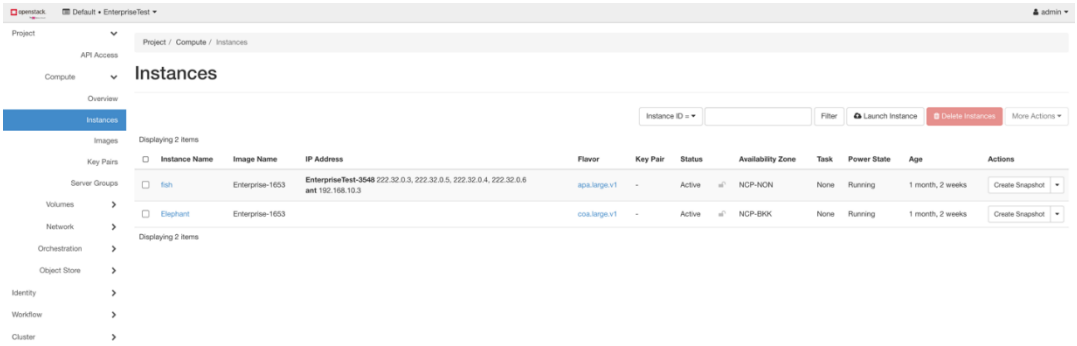
รูปที่ 15 สวิตช์ และเครื่องแม่ข่ายที่ดำเนินการติดตั้งไซต์ขอนแก่น (2)

รูปที่ 16 อุปกรณ์ IoT Edge ของภาคตะวันออก (ระยอง)

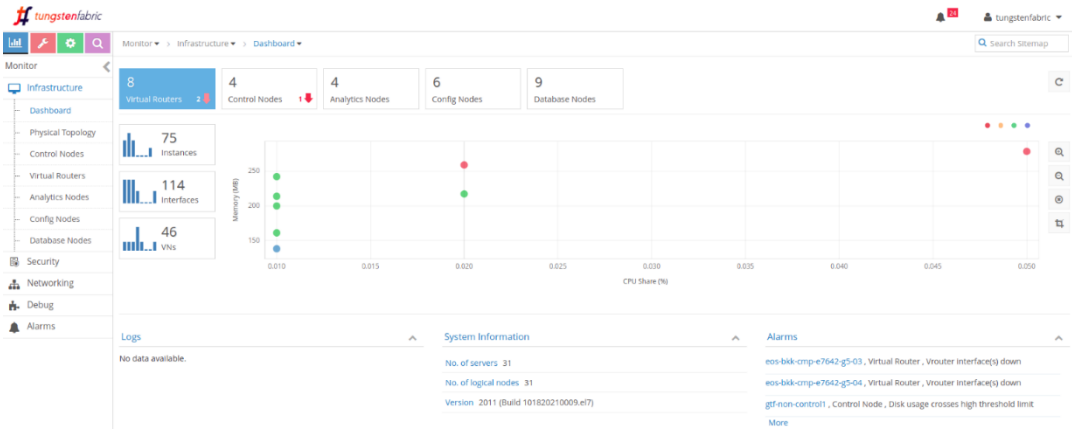
การรายงานในส่วนนี้มุ่งเน้นการสรุปผลเชิงลึกเกี่ยวกับการติดตั้งฮาร์ดแวร์ที่มีความสำคัญต่อโครงการ โดยครอบคลุมกระบวนการในทุกขั้นตอน ตั้งแต่การวิเคราะห์ความต้องการเชิงลึกเบื้องต้นที่เกี่ยวข้องกับลักษณะการใช้งานและบริบทเฉพาะของโครงการ การวางแผนและจัดหาอุปกรณ์ที่เหมาะสมที่สุด ซึ่งคำนึงถึงความคุ้มค่าและความทันสมัยของเทคโนโลยี ไปจนถึงการดำเนินการติดตั้งที่มีความซับซ้อน โดยมีการออกแบบกระบวนการอย่างเป็นระบบเพื่อรองรับความหลากหลายในสภาพแวดล้อมการทำงาน การติดตั้งฮาร์ดแวร์ในครั้งนี้ไม่ได้เป็นเพียงการตอบสนองต่อความต้องการในปัจจุบัน แต่ยังถือเป็นต้นแบบสำคัญสำหรับการพัฒนาระบบ Edge Cloud Computing และ IoT Infrastructure ในอนาคตอย่างยั่งยืน กระบวนการดังกล่าวยังได้รับการสนับสนุนด้วยมาตรฐานการดำเนินงานที่เข้มงวด เพื่อรับรองว่าฮาร์ดแวร์สามารถตอบสนองความต้องการในระดับสูง ทั้งในด้านความเสถียร การประหยัดพลังงาน และความสามารถในการรองรับการใช้งานที่ซับซ้อนในระยะยาวนอกจากนี้ ยังมีการดำเนินการประเมินผลการติดตั้งอย่างละเอียดถี่ถ้วน ซึ่งครอบคลุมการวิเคราะห์ปัจจัยที่ส่งผลต่อประสิทธิภาพของระบบ เพื่อระบุโอกาสในการปรับปรุงและพัฒนาต่อยอดในระยะยาว ความสำเร็จของกระบวนการนี้จะสะท้อนให้เห็นถึงความร่วมมือที่แข็งแกร่งระหว่างทีมงานและผู้มีส่วนเกี่ยวข้องทั้งหมด ซึ่งช่วยสนับสนุนให้ระบบที่ติดตั้งสามารถตอบสนองความต้องการที่เปลี่ยนแปลงและท้าทายได้อย่างต่อเนื่อง พร้อมทั้งสนับสนุนความยั่งยืนในระยะยาวของได้โครงการในทุกมิติ.

รายงานผลการติดตั้งซอฟต์แวร์

ดำเนินการติดตั้งซอฟต์แวร์ OpenStack และ OpenSDN ที่ไซต์นนทบุรี

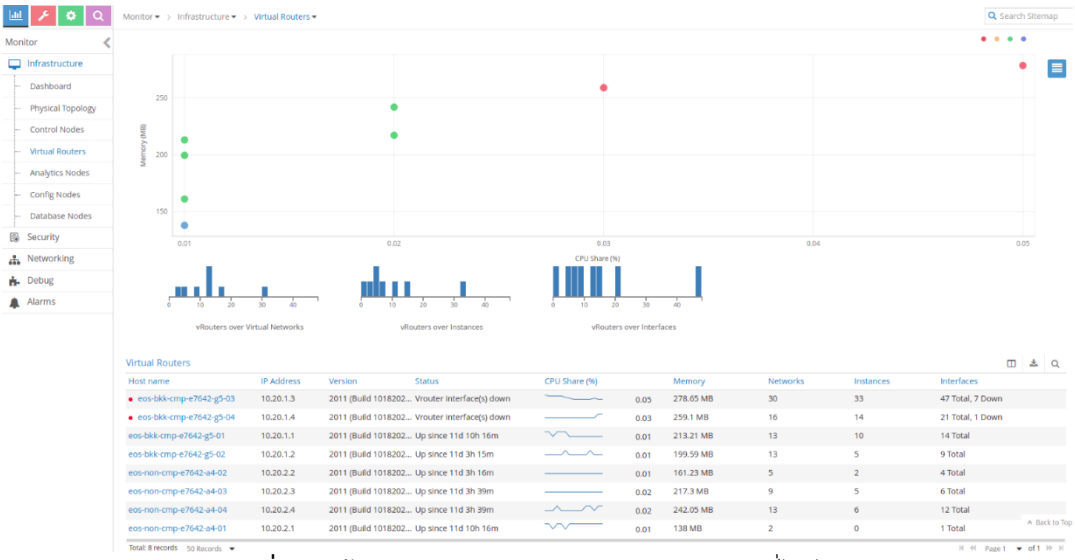


รูปที่ 17 หน้า Dasdhbarod ของ ซอฟต์แวร์ OpenStack



รูปที่ 18 หน้า Dasdhbarod ของ OpenSDN

รูปที่ 19 หน้า Web UI แสดงผลของ Control Nodes ที่ไซต์นนทบุรี (1)

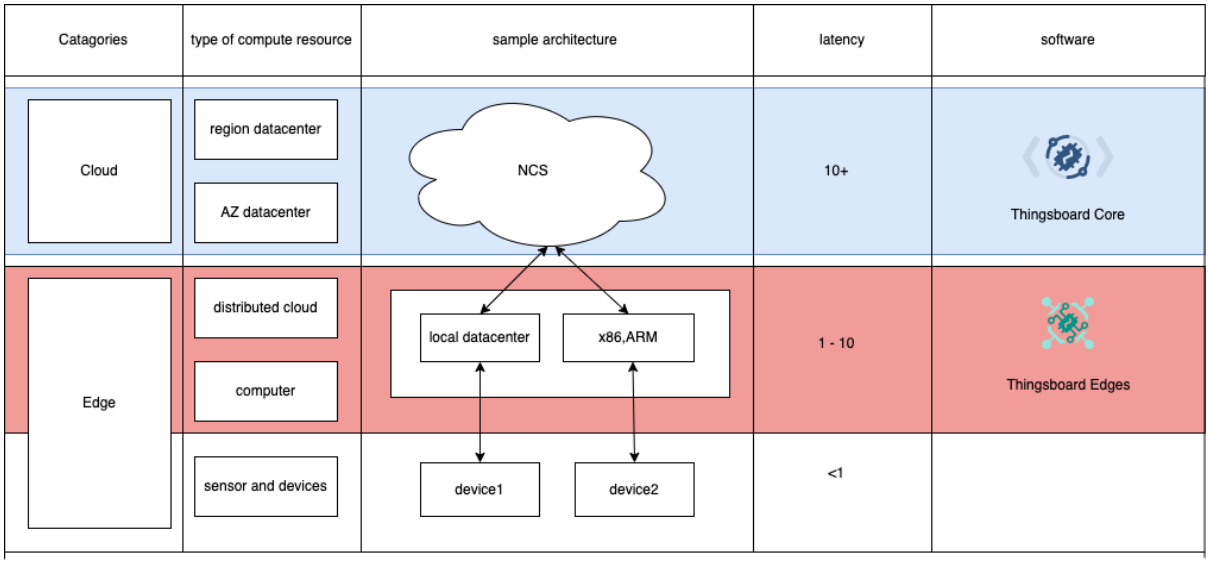


รูปที่ 20 หน้า Web UI แสดงผลของ Control Nodes ที่ไซต์นนทบุรี (2)

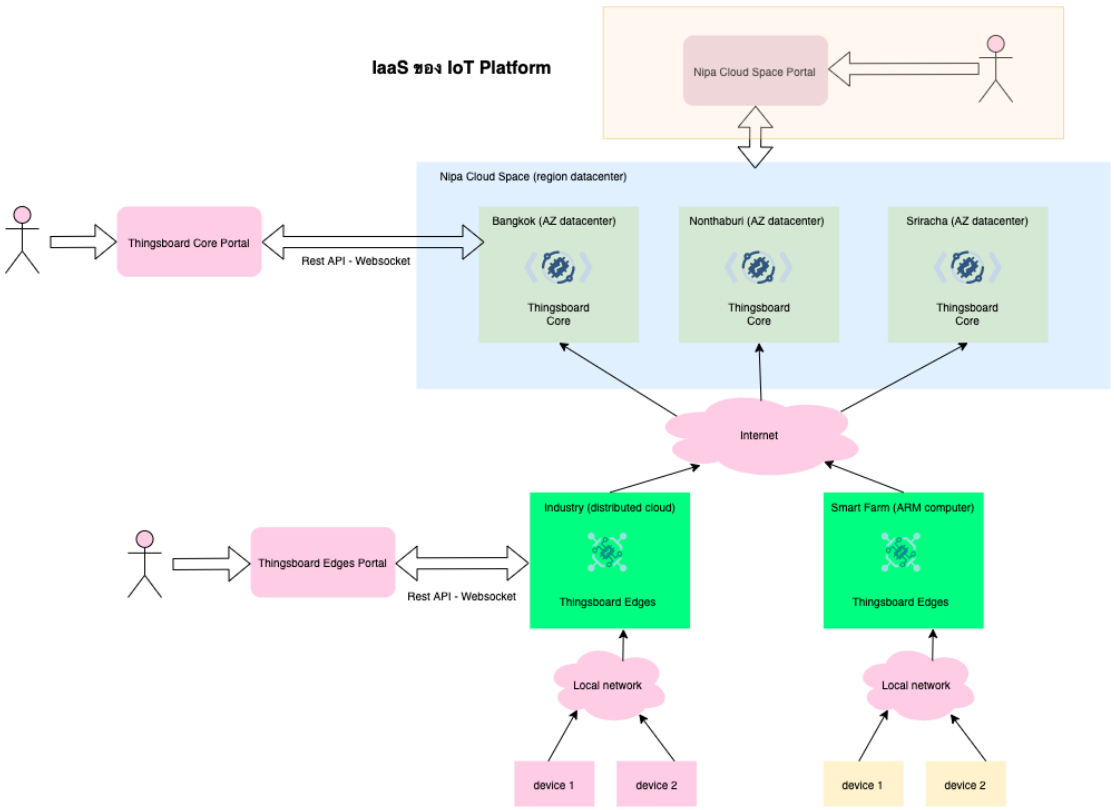
การรายงานในส่วนนี้มุ่งเน้นการสรุปผลเกี่ยวกับการติดตั้งอุปกรณ์ฮาร์ดแวร์ที่มีความสำคัญต่อโครงการ ครอบคลุมการวิเคราะห์ความต้องการเบื้องต้น การจัดหาอุปกรณ์ที่เหมาะสม และการติดตั้งที่ออกแบบมาอย่างเป็นระบบ เพื่อตอบสนองความหลากหลายในบริบทการทำงาน ฮาร์ดแวร์ดังกล่าวถือเป็นต้นแบบสำคัญสำหรับการพัฒนาระบบ Edge Cloud Computing และ IoT Infrastructure ในอนาคต โดยมีการกำหนดมาตรฐานการดำเนินงานที่เข้มงวด เพื่อรับรองความเสถียรและความยั่งยืนในระยะยาว นอกจากนี้ ยังมีการประเมินผลอย่างละเอียดเพื่อปรับปรุงประสิทธิภาพของระบบให้สอดคล้องกับความต้องการที่เปลี่ยนแปลงได้อย่างมีประสิทธิภาพ.

ดำเนินการติดตั้งซอฟต์แวร์ IoT Platform

การติดตั้งซอฟต์แวร์ IoT Platform ในโครงการนี้ถือเป็นขั้นตอนสำคัญที่แสดงให้เห็นถึงความก้าวหน้าในเชิงปฏิบัติอย่างชัดเจน โดยการดำเนินการนี้ไม่เพียงแต่สะท้อนถึงศักยภาพในการพัฒนาเทคโนโลยีของทีมงานและองค์กร แต่ยังเป็นเครื่องยืนยันถึงความพร้อมในการให้บริการระบบ IoT ที่มีคุณภาพสูง สามารถรองรับการประมวลผลข้อมูลแบบเรียลไทม์ได้อย่างมีประสิทธิภาพ อีกทั้งยังตอบสนองต่อความต้องการที่หลากหลายของภาคอุตสาหกรรมและหน่วยงานต่าง ๆ ได้อย่างครบถ้วนและครอบคลุม ความสำเร็จของขั้นตอนนี้ยังแสดงให้เห็นถึงการเตรียมความพร้อมในการประยุกต์ใช้งานระบบ IoT ในสถานการณ์จริง ทั้งในด้านการเพิ่มประสิทธิภาพการทำงานและการสนับสนุนความต้องการของผู้ใช้ในระดับองค์กรได้อย่างมีประสิทธิภาพสูงสุด



รูปที่ 21 Type ของ Resource ของ Thingisboard IoT Platform



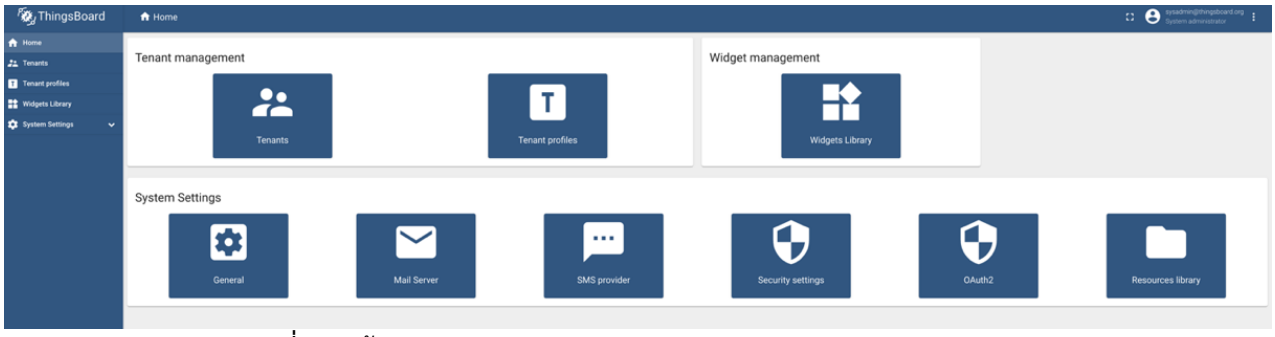
รูปที่ 22 Infrastructure As A Service ของ Thingsboard IoT Platform

การสร้าง instance บนหน้า Nipa Public Cloud Portal ถือเป็นขั้นตอนสำคัญที่ช่วยเสริมศักยภาพของระบบ IoT Platform โดยขั้นตอนนี้ได้รับการออกแบบให้เป็น Infrastructure As A Service (IaaS) ซึ่งช่วยให้ระบบสามารถจัดการและประมวลผลข้อมูล IoT ได้อย่างมีประสิทธิภาพสูงสุด สำหรับ IoT Platform นั้นถูกปรับให้ทำงานอยู่บน instances ที่ดำเนินการอยู่ภายใต้ระบบ NIPA PUBLIC CLOUD โดยเฉพาะ ซึ่งมีบทบาทสำคัญในการทำหน้าที่เป็น IaaS รองรับการทำงานของระบบ IoT ทั้งในด้านโครงสร้างพื้นฐาน การจัดเก็บข้อมูล และการเชื่อมต่อกับอุปกรณ์ IoT ในหลากหลายสถานการณ์

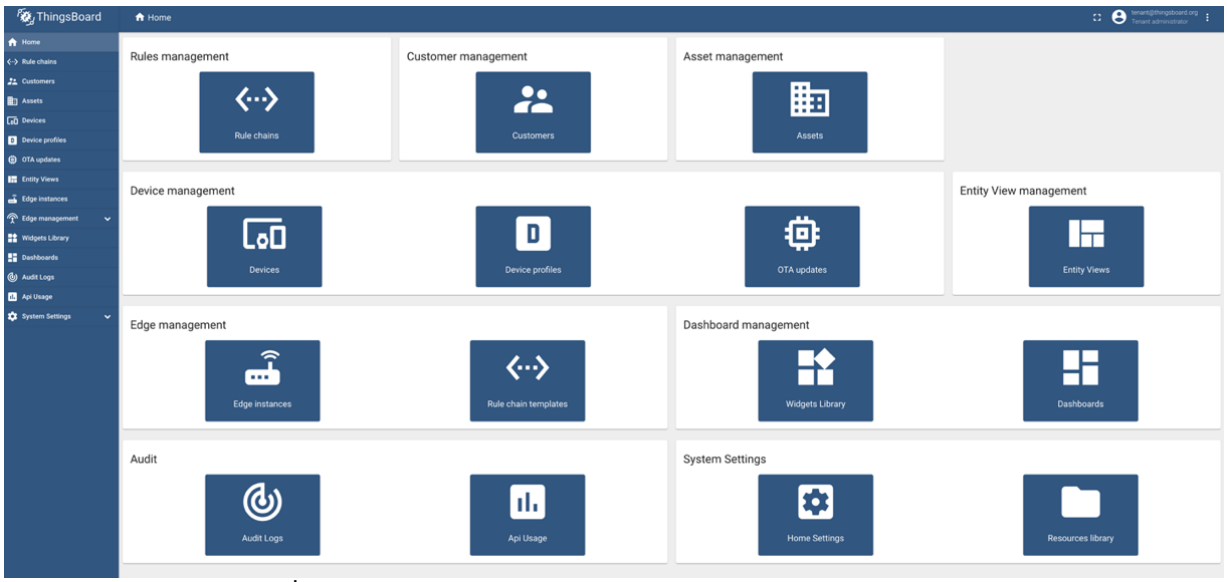
การดำเนินการดังกล่าวไม่เพียงแค่สร้างโครงสร้างพื้นฐานที่มั่นคงสำหรับ IoT Platform แต่ยังช่วยเพิ่มขีดความสามารถในการปรับเปลี่ยนระบบให้เหมาะสมกับความต้องการที่เปลี่ยนแปลงไปขององค์กรและผู้ใช้งาน
ด้วยความสามารถในการปรับขยาย (Scalability) ที่รองรับจำนวนอุปกรณ์ IoT ได้เพิ่มขึ้นอย่างต่อเนื่อง รวมถึงการใช้ทรัพยากรที่เหมาะสมเพื่อให้การทำงานมีประสิทธิภาพสูงสุดและลดความซับซ้อนในการจัดการโครงสร้างพื้นฐาน

นอกจากนี้ การใช้ NIPA PUBLIC CLOUD ยังช่วยให้ระบบ IoT Platform สามารถผสานรวมกับเทคโนโลยีอื่นๆ เช่น การประมวลผลข้อมูลเชิงลึก (Data Analytics) และปัญญาประดิษฐ์ (AI) ได้อย่างลงตัว ช่วยเพิ่มศักยภาพในการวิเคราะห์ข้อมูลแบบเรียลไทม์และการคาดการณ์ล่วงหน้า เพื่อตอบสนองต่อความต้องการทางธุรกิจในระดับสูงสุด ทั้งยังลดเวลาและค่าใช้จ่ายในการพัฒนาโซลูชัน IoT ให้มีประสิทธิภาพในระยะยาว

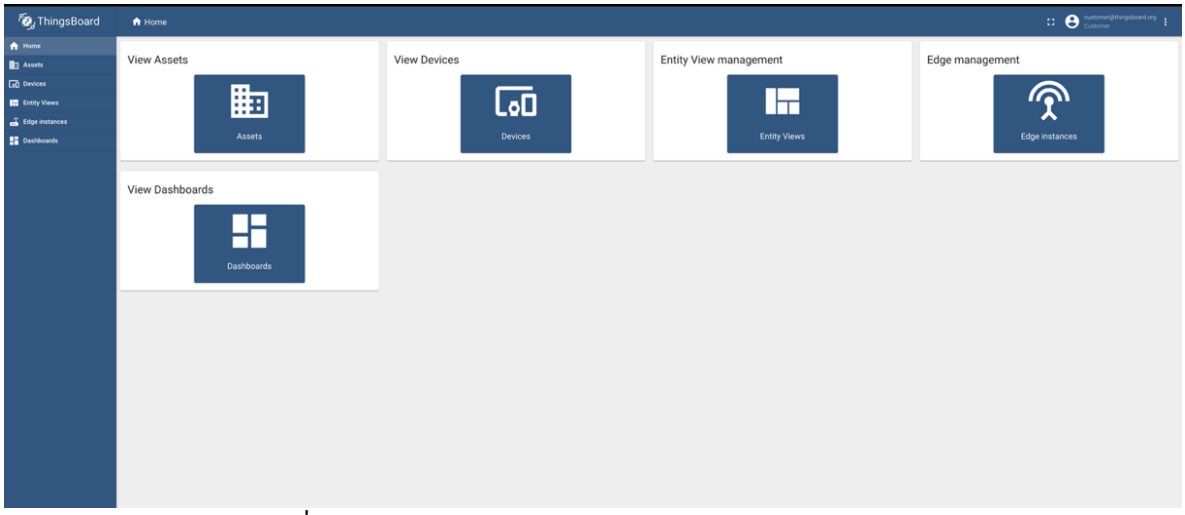
การสร้าง instance ดังกล่าวยังเปิดโอกาสให้มีการพัฒนานวัตกรรมในระดับองค์กร โดยสร้างระบบที่สามารถปรับเปลี่ยนและตอบสนองต่อสถานการณ์ใหม่ๆ ได้อย่างรวดเร็ว อันเป็นประโยชน์ต่อการขับเคลื่อนอุตสาหกรรมในยุคดิจิทัล ทั้งนี้ระบบที่ถูกติดตั้งจะสามารถทำงานร่วมกับโครงสร้างพื้นฐานเดิมได้อย่างราบรื่นและช่วยสนับสนุนการทำงานของทุกหน่วยงานในองค์กรอย่างมีประสิทธิภาพสูงสุด



รูปที่ 23 หน้าแรก System admin ของ Thingsboard IoT Platform (3.4.3)



รูปที่ 24 หน้าแรก Tenant admin ของ Thingsboard IoT Platform (3.4.3)

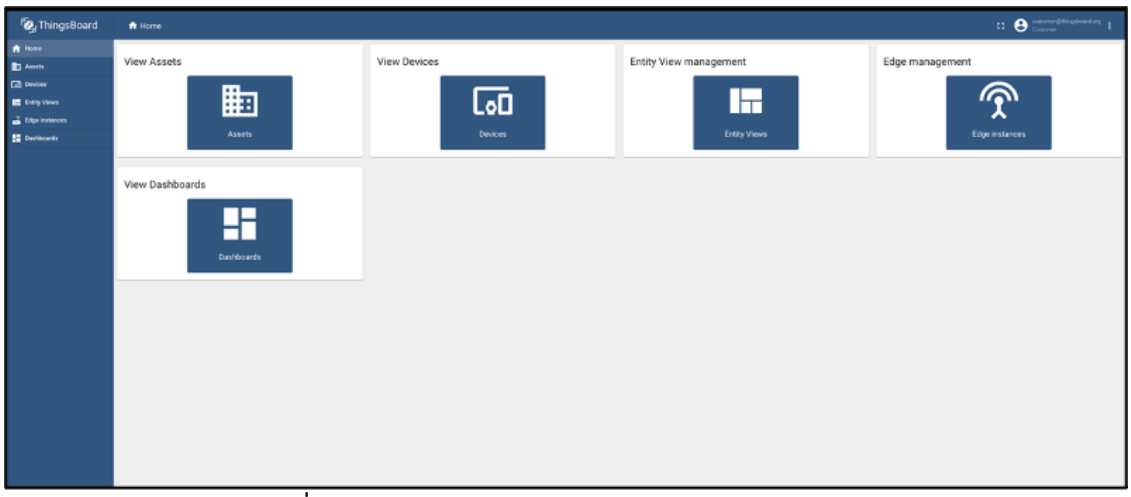


รูปที่ 25 หน้าแรก Customer ของ Thingsboard IoT Platform (3.4.3)

System Admin เหมาะสำหรับการจัดการในภาพรวมของแพลตฟอร์มทั้งหมด โดยมีหน้าที่สำคัญในการกำกับดูแลและจัดการทรัพยากรต่าง ๆ อย่างครอบคลุม เพื่อให้ระบบมีประสิทธิภาพสูงสุดและทำงานได้อย่างสอดคล้องกันทั้งในด้านเทคนิคและการใช้งาน นอกจากนี้ ยังสามารถกำหนดค่าต่าง ๆ ในระดับโครงสร้างพื้นฐาน เช่น การจัดการ Tenant, ระบบความปลอดภัย, และการเชื่อมต่อกับบริการอื่น ๆ ซึ่งช่วยเพิ่มความเสถียรและความปลอดภัยของระบบในภาพรวม

ส่วนของ Tenant Admin เน้นไปที่การบริหารจัดการอุปกรณ์และข้อมูลที่เกี่ยวข้องกับผู้ใช้งานในแต่ละองค์กร โดยให้ความสำคัญกับการตั้งค่าอุปกรณ์ IoT การจัดการลูกค้า และการออกแบบ Dashboard ที่ตอบโจทย์การใช้งานในระดับปฏิบัติการ ทั้งนี้ยังช่วยให้การดำเนินงานต่าง ๆ เป็นไปอย่างราบรื่นและปรับเปลี่ยนได้ง่ายตามความต้องการขององค์กร ซึ่งเป็นส่วนสำคัญในการเพิ่มประสิทธิภาพและตอบสนองต่อความต้องการเฉพาะของผู้ใช้งานได้อย่างตรงจุด

ในส่วนของ Customer ระบบช่วยให้สามารถบริหารจัดการข้อมูลลูกค้าไม่ว่าจะเป็นการเพิ่มข้อมูลลูกค้าภายใต้ใหม่ การจัดกลุ่มลูกค้าตามประเภทหรือความต้องการ และการเชื่อมโยงข้อมูลลูกค้ากับอุปกรณ์หรือบริการที่ใช้งานในระบบ IoT สิ่งนี้ช่วยให้ Tenant Admin สามารถวางแผนและให้บริการได้ตรงตามความต้องการของลูกค้าแต่ละรายได้อย่างมีประสิทธิภาพ พร้อมทั้งสนับสนุนการสร้างความสัมพันธ์ที่ดีกับลูกค้าผ่านการจัดการข้อมูลที่โปร่งใสและเชื่อถือได้



รูปที่ 26 หน้าแรก Edge ของ Thingsboard IoT Platform (3.4.3)

สำหรับ Edge (ThingsBoard Edge) สามารถถูกติดตั้งอยู่บนแพลตฟอร์มที่หลากหลายและมีความยืดหยุ่นสูง เพื่อรองรับการใช้งานในสภาพแวดล้อมที่แตกต่างกัน โดยมีตัวเลือกที่หลากหลายและเหมาะสมกับการใช้งานในทุกระดับ ดังนี้:

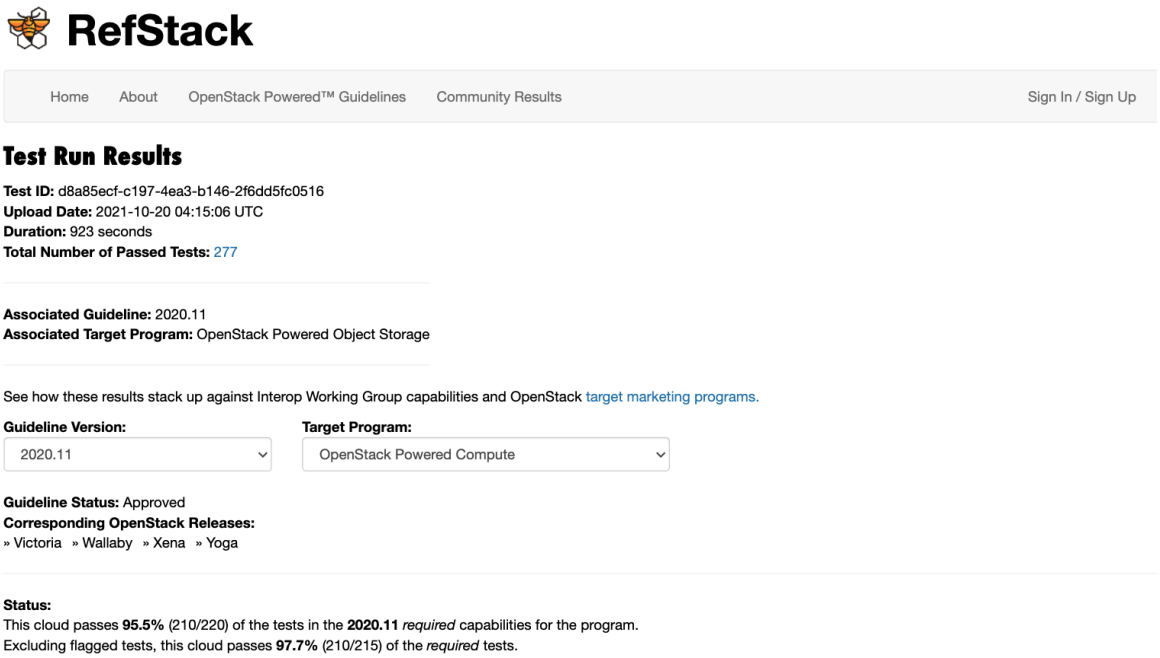
- **VM บน Cloud**: ช่วยให้สามารถปรับขยายทรัพยากรได้อย่างรวดเร็วและมีความปลอดภัยสูง
  รองรับการดำเนินงานขององค์กรที่ต้องการระบบคลาวด์ที่มีความเสถียรสูง
  สามารถจัดการทรัพยากรได้อย่างมีประสิทธิภาพ
  และเพิ่มศักยภาพในการรองรับการขยายตัวในอนาคตได้โดยง่าย
- **Private Server**: รองรับการติดตั้งในสถานที่ที่มีการควบคุมทรัพยากรภายในองค์กรอย่างเข้มงวด
  เพื่อเพิ่มความมั่นคงและปกป้องข้อมูลสำคัญขององค์กร
  เหมาะสำหรับหน่วยงานที่ต้องการควบคุมระบบและทรัพยากรในระดับสูง
  พร้อมทั้งสามารถปรับแต่งระบบให้ตรงกับความต้องการเฉพาะทางได้
- **Raspberry Pi**: เป็นตัวเลือกที่เหมาะสำหรับการทดสอบ การพัฒนา
  และการใช้งานในโครงการขนาดเล็กหรือระบบ IoT ที่ต้องการต้นทุนต่ำ
  ด้วยขนาดเล็กและความยืดหยุ่นในการปรับใช้ Raspberry Pi
  ยังสามารถรองรับการทดลองเชิงนวัตกรรมและการใช้งานในสภาพแวดล้อมที่มีข้อจำกัดด้านทรัพยากรอย่างมีประสิทธิภาพ

4.1.3 ผลการทดสอบประสิทธิภาพฮาร์ดแวร์และซอฟต์แวร์

รายงานผลการทดสอบซอฟต์แวร์ OpenStack สำหรับ Edge Compute

ทีมงานดำเนินการตรวจสอบประสิทธิภาพและความสอดคล้องของระบบ OpenStack ผ่านการใช้เครื่องมือ RefStack ซึ่งเป็นเครื่องมือมาตรฐานในการทดสอบการทำงานร่วมกันของซอฟต์แวร์ OpenStack โดยเน้นการประมวลผลแบบร่วมกัน (Interoperability Testing)
และการจัดเก็บข้อมูลสำรองในบริบทของการใช้งานที่ครอบคลุมในระดับอุตสาหกรรม
กระบวนการนี้ได้รับการออกแบบมาเพื่อให้มั่นใจว่าระบบสามารถตอบสนองต่อความต้องการของผู้ใช้งานได้อย่างเต็มที่ในสภาพแวดล้อมที่หลากหลาย นอกจากนี้ ยังได้เผยแพร่ผลการทดสอบสู่ชุมชนผู้ใช้งาน OpenStack อย่างโปร่งใส เพื่อเสริมสร้างความเชื่อมั่นในมาตรฐานและคุณภาพของระบบ

ผลการประเมินแสดงให้เห็นว่าระบบสามารถรองรับมาตรฐาน "OpenStack Powered Compute" ได้ถึง 97.7% (210/215) ของเกณฑ์ที่กำหนดในข้อบังคับการทดสอบความสอดคล้อง โดยเน้นถึงความสามารถในการทำงานร่วมกันและความยืดหยุ่นของระบบในบริบทที่มีความซับซ้อนชั้นสูง ความสำเร็จนี้สะท้อนถึงความน่าเชื่อถือและประสิทธิภาพที่ตอบโจทย์มาตรฐานระดับสากล นอกจากนี้ รายงานผลการประเมินยังมีการแสดงผลข้อมูลเชิงลึกเกี่ยวกับประสิทธิภาพของระบบในแต่ละด้าน เช่น การรองรับภาระงานที่หลากหลาย การลดข้อผิดพลาด และการปรับปรุงเวลาการตอบสนอง



รูปที่ 27 การทดสอบประสิทธิภาพซอฟต์แวร์ OpenStack ด้วย RefStack

Capability Overview

Test Filters:

All | Passed | Not Passed | Flagged

Required (Total: 58 capabilities, 220 tests) (Passed: 210 tests)

1. compute-availability-zones-list [1]
   - ✔ tempest.api.compute.servers.test_availability_zone.AZV2TestJSON.test_get_availability_zone_list_with_non_admin_user
2. compute-flavors-list [2]
   - ✔ tempest.api.compute.flavors.test_flavors.FlavorsV2TestJSON.test_list_flavors
   - ✔ tempest.api.compute.flavors.test_flavors.FlavorsV2TestJSON.test_list_flavors_with_detail
3. compute-instance-actions-get [1]
   - ✔ tempest.api.compute.servers.test_instance_actions.InstanceActionsTestJSON.test_get_instance_action
4. compute-instance-actions-list [1]
   - ✔ tempest.api.compute.servers.test_instance_actions.InstanceActionsTestJSON.test_list_instance_actions
5. compute-keypairs-create [1]
   - ✔ tempest.api.compute.servers.test_servers.ServersTestJSON.test_create_specify_keypair
6. compute-keypairs-create-type [1]
   - ✔ tempest.api.compute.keypairs.test_keypairs_v22.KeyPairsV22TestJSON.test_keypairsv22_create_list_show_with_type
7. compute-list-api-versions [1]
   - ✔ tempest.api.compute.test_versions.TestVersions.test_list_api_versions
8. compute-quotas-get [2]
   - ✔ tempest.api.compute.test_quotas.QuotasTestJSON.test_get_default_quotas
   - ✔ tempest.api.compute.test_quotas.QuotasTestJSON.test_get_quotas
9. compute-servers-create [8]
   - ✔ tempest.api.compute.servers.test_servers.ServersTestJSON.test_create_server_with_admin_password
   - ✔ tempest.api.compute.servers.test_servers.ServersTestJSON.test_create_with_existing_server_name
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_numeric_server_name
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_server_metadata_exceeds_length_limit
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_server_name_length_exceeds_256
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_with_invalid_flavor
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_with_invalid_image
   - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_create_with_invalid_network_uuid

รูปที่ 28 ผลการทดสอบประสิทธิภาพด้วย RefStack (1)

10. compute-servers-create-multiple [1]
    - ✔ tempest.api.compute.servers.test_multiple_create.MultipleCreateTestJSON.test_multiple_create
11. compute-servers-delete [3]
    - ✔ tempest.api.compute.servers.test_delete_server.DeleteServersTestJSON.test_delete_active_server
    - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_delete_server_pass_id_exceeding_length_limit
    - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_delete_server_pass_negative_id
12. compute-servers-get [1]
    - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_get_non_existent_server
13. compute-servers-host [1]
    - ✔ tempest.api.compute.servers.test_create_server.ServersTestJSON.test_host_name_is_same_as_server_name — [Aliases]
14. compute-servers-invalid [1]
    - ✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_invalid_ip_v6_address
15. compute-servers-list [24]
    - ✔ tempest.api.compute.servers.test_create_server.ServersTestJSON.test_list_servers — [Aliases]
    - ✔ tempest.api.compute.servers.test_create_server.ServersTestJSON.test_list_servers_with_detail — [Aliases]
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_detailed_filter_by_flavor
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_detailed_filter_by_image
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_detailed_filter_by_server_name
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_detailed_filter_by_server_status
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_detailed_limit_results
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filter_by_active_status — [Aliases]
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filter_by_flavor
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filter_by_image
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filter_by_limit
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filter_by_server_name
    - ✔ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filtered_by_name_wildcard
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_changes_since_future_date
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_changes_since_invalid_date
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_limits_greater_than_actual_count
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_limits_pass_negative_value
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_limits_pass_string
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_non_existing_flavor
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_non_existing_image
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_by_non_existing_server_name
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_detail_server_is_deleted
    - ✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_status_non_existing

รูปที่ 29 ผลการทดสอบประสิทธิภาพด้วย RefStack (2)

✔ tempest.api.compute.servers.test_list_servers_negative.ListServersNegativeTestJSON.test_list_servers_with_a_deleted_server
16. compute-servers-lock [1]
✔ tempest.api.compute.servers.test_server_actions.ServerActionsTestJSON.test_lock_unlock_server
17. compute-servers-metadata-delete [1]
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_delete_server_metadata_item
18. compute-servers-metadata-get [1]
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_get_server_metadata_item
19. compute-servers-metadata-list [1]
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_list_server_metadata
20. compute-servers-metadata-set [2]
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_set_server_metadata
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_set_server_metadata_item

รูปที่ 30 ผลการทดสอบประสิทธิภาพด้วย RefStack (3)

21. compute-servers-metadata-update [1]
✔ tempest.api.compute.servers.test_server_metadata.ServerMetadataTestJSON.test_update_server_metadata
22. compute-servers-name [1]
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_server_name_blank
23. compute-servers-reboot [2]
✔ tempest.api.compute.servers.test_server_actions.ServerActionsTestJSON.test_reboot_server_hard
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_reboot_non_existent_server
24. compute-servers-rebuild [3]
✔ tempest.api.compute.servers.test_server_actions.ServerActionsTestJSON.test_rebuild_server
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_rebuild_deleted_server — [Aliases]
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_rebuild_non_existent_server
25. compute-servers-stop [2]
✔ tempest.api.compute.servers.test_server_actions.ServerActionsTestJSON.test_stop_start_server
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_stop_non_existent_server
26. compute-servers-update [5]
✔ tempest.api.compute.servers.test_servers.ServersTestJSON.test_update_access_server_address
✔ tempest.api.compute.servers.test_servers.ServersTestJSON.test_update_server_name
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_update_name_of_non_existent_server
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_update_server_name_length_exceeds_256
✔ tempest.api.compute.servers.test_servers_negative.ServersNegativeTestJSON.test_update_server_set_empty_name
27. compute-servers-verify [2]
✔ tempest.api.compute.servers.test_create_server.ServersTestJSON.test_verify_created_server_vcpus — [Aliases]
✔ tempest.api.compute.servers.test_create_server.ServersTestJSON.test_verify_server_details — [Aliases]
28. compute-volume-attach [2]
✔ tempest.api.compute.volumes.test_attach_volume.AttachVolumeTestJSON.test_attach_detach_volume
✔ tempest.api.compute.volumes.test_attach_volume.AttachVolumeTestJSON.test_list_get_volume_attachments
29. identity-v3-api-discovery [3]
✔ tempest.api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_media_types — [Aliases]
✔ tempest.api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_version_resources — [Aliases]
✔ tempest.api.identity.v3.test_api_discovery.TestApiDiscovery.test_api_version_statuses — [Aliases]
30. identity-v3-catalog [1]
✔ tempest.api.identity.v3.test_catalog.IdentityCatalogTest.test_catalog_standardization
31. identity-v3-tokens-create [1]
✔ tempest.api.identity.v3.test_tokens.TokensV3Test.test_create_token
32. identity-v3-tokens-delete [1]
✔ tempest.api.identity.v3.test_tokens.TokensV3Test.test_token_auth_creation_existence_deletion

รูปที่ 31 ผลการทดสอบประสิทธิภาพด้วย RefStack (4)

33. identity-v3-tokens-validate [1]
    ✔ tempest.api.identity.v3.test_tokens.TokensV3Test.test_validate_token
34. images-v2-delete [4]
    ✔ tempest.api.image.v2.test_images.BasicOperationsImagesTest.test_delete_image
    ✔ tempest.api.image.v2.test_images_negative.ImagesNegativeTest.test_delete_image_null_id
    ✔ tempest.api.image.v2.test_images_negative.ImagesNegativeTest.test_delete_non_existing_image
    ✔ tempest.api.image.v2.test_images_tags_negative.ImagesTagsNegativeTest.test_delete_non_existing_tag
35. images-v2-get [5]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_get_image_schema — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_get_images_schema — [Aliases]
    ✔ tempest.api.image.v2.test_images_negative.ImagesNegativeTest.test_get_delete_deleted_image
    ✔ tempest.api.image.v2.test_images_negative.ImagesNegativeTest.test_get_image_null_id
    ✔ tempest.api.image.v2.test_images_negative.ImagesNegativeTest.test_get_non_existent_image
36. images-v2-index [1]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_no_params — [Aliases]
37. images-v2-list [7]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_container_format — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_disk_format — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_limit — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_min_max_size — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_size — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_status — [Aliases]
    ✔ tempest.api.image.v2.test_images.ListUserImagesTest.test_list_images_param_visibility — [Aliases]
38. images-v2-update [3]
    ✔ tempest.api.image.v2.test_images.BasicOperationsImagesTest.test_update_image
    ✔ tempest.api.image.v2.test_images_tags.ImagesTagsTest.test_update_delete_tags_for_image
    ✔ tempest.api.image.v2.test_images_tags_negative.ImagesTagsNegativeTest.test_update_tags_for_non_existing_image

รูปที่ 32 ผลการทดสอบประสิทธิภาพด้วย RefStack (5)

39. networks-t2-CRUD [22]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_all_attributes — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_with_allocation_pools — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_with_dhcp_enabled — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_with_gw — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_with_gw_and_allocation_pools — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_with_host_routes_and_dns_nameservers — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_delete_subnet_without_gateway — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_create_update_delete_network_subnet — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_delete_network_with_subnet — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_list_networks — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_list_subnets — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_list_subnets_fields — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_show_network — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_show_subnet — [Aliases]
    ✔ tempest.api.network.test_networks.NetworksTest.test_show_subnet_fields — [Aliases]
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_create_bulk_port
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_create_port_in_allowed_allocation_pools
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_create_update_delete_port
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_list_ports
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_list_ports_fields
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_show_port
    ✔ tempest.api.network.test_ports.PortsTestJSON.test_show_port_fields

รูปที่ 33 ผลการทดสอบประสิทธิภาพด้วย RefStack (6)

40. networks-l3-CRUD [11]
- ✔ tempest.api.network.test_networks.NetworksTest.test_external_network_visibility — [Aliases]
- ✔ tempest.api.network.test_ports.PortsTestJSON.test_port_list_filter_by_router_id
- ✔ tempest.api.network.test_routers.RoutersTest.test_add_multiple_router_interfaces
- ✔ tempest.api.network.test_routers.RoutersTest.test_add_remove_router_interface_with_port_id
- ✔ tempest.api.network.test_routers.RoutersTest.test_add_remove_router_interface_with_subnet_id
- ✔ tempest.api.network.test_routers.RoutersTest.test_create_show_list_update_delete_router
- ✔ tempest.api.network.test_routers.RoutersTest.test_update_router_admin_state
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_add_router_interfaces_on_overlapping_subnets_returns_400
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_delete_non_existent_router_returns_404
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_show_non_existent_router_returns_404
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_update_non_existent_router_returns_404

41. networks-l3-router [3]
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_router_add_gateway_invalid_network_returns_404
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_router_add_gateway_net_not_external_returns_400
- ✔ tempest.api.network.test_routers_negative.RoutersNegativeTest.test_router_remove_interface_in_use_returns_409

42. networks-list-api-versions [1]
- ✔ tempest.api.network.test_versions.NetworksApiDiscovery.test_api_version_resources

43. networks-security-groups-CRUD [19]
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_list_update_show_delete_security_group
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_security_group_rule_with_additional_args
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_security_group_rule_with_icmp_type_code
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_security_group_rule_with_protocol_integer_value
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_security_group_rule_with_remote_group_id
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_security_group_rule_with_remote_ip_prefix
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_create_show_delete_security_group_rule
- ✔ tempest.api.network.test_security_groups.SecGroupTest.test_list_security_groups
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_additional_default_security_group_fails
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_duplicate_security_group_rule_fails
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_bad_ethertype
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_bad_protocol
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_bad_remote_ip_prefix
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_invalid_ports
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_non_existent_remote_groupid
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_create_security_group_rule_with_non_existent_security_group
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_delete_non_existent_security_group

รูปที่ 34 ผลการทดสอบประสิทธิภาพด้วย RefStack (7)

✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_show_non_existent_security_group
✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_show_non_existent_security_group_rule

44. volumes-list-api-versions [1]
✔ tempest.api.volume.test_versions.VersionsTest.test_list_versions

45. volumes-v3-availability-zones [1]
✔ tempest.api.volume.test_availability_zone.AvailabilityZoneTestJSON.test_get_availability_zone_list — [Aliases]

46. volumes-v3-clone [1]
✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete_as_clone — [Aliases]

47. volumes-v3-copy-image-to-volume [2]
✔ tempest.api.volume.test_volumes_actions.VolumesActionsTest.test_volume_bootable — [Aliases]
✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete_from_image — [Aliases]

48. volumes-v3-create-delete [7]
✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_invalid_size — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_nonexistent_source_volid — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_nonexistent_volume_type — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_size_negative — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_size_zero — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_without_passing_size — [Aliases]

49. volumes-v3-extensions [1]
✔ tempest.api.volume.test_extensions.ExtensionsTestJSON.test_list_extensions — [Aliases]

50. volumes-v3-get [3]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_get_invalid_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_get_volume_without_passing_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_volume_get_nonexistent_volume_id — [Aliases]

51. volumes-v3-list [19]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_by_name — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_by_name — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_pagination — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_with_multiple_params — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_pagination — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_param_display_name_and_status — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_detail_param_display_name_and_status — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_detail_param_metadata — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_details — [Aliases]

รูปที่ 35 ผลการทดสอบประสิทธิภาพด้วย RefStack (8)

✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_param_metadata — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volumes_list_by_availability_zone — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volumes_list_by_status — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volumes_list_details_by_availability_zone — [Aliases]
✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volumes_list_details_by_status — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_list_volumes_detail_with_invalid_status — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_list_volumes_detail_with_nonexistent_name — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_list_volumes_with_invalid_status — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_list_volumes_with_nonexistent_name — [Aliases]

52. volumes-v3-metadata [2]
✔ tempest.api.volume.test_volume_metadata.VolumesMetadataTest.test_crud_volume_metadata — [Aliases]
✔ tempest.api.volume.test_volume_metadata.VolumesMetadataTest.test_update_show_volume_metadata_item — [Aliases]

53. volumes-v3-readonly [1]
✔ tempest.api.volume.test_volumes_actions.VolumesActionsTest.test_volume_readonly_update — [Aliases]

54. volumes-v3-snapshot-create-delete [12]
✔ tempest.api.volume.test_snapshot_metadata.SnapshotMetadataTestJSON.test_crud_snapshot_metadata — [Aliases]
✔ tempest.api.volume.test_snapshot_metadata.SnapshotMetadataTestJSON.test_update_show_snapshot_metadata_item — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_nonexistent_snapshot_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_delete_invalid_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_delete_volume_without_passing_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_volume_delete_nonexistent_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots.VolumesSnapshotTestJSON.test_snapshot_create_get_list_update_delete — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots.VolumesSnapshotTestJSON.test_volume_from_snapshot — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots_list.VolumesSnapshotListTestJSON.test_snapshots_list_details_with_params — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots_list.VolumesSnapshotListTestJSON.test_snapshots_list_with_params — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots_negative.VolumesSnapshotNegativeTestJSON.test_create_snapshot_with_nonexistent_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_snapshots_negative.VolumesSnapshotNegativeTestJSON.test_create_snapshot_without_passing_volume_id — [Aliases]

55. volumes-v3-update [3]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_update_volume_with_empty_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_update_volume_with_invalid_volume_id — [Aliases]
✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_update_volume_with_nonexistent_volume_id — [Aliases]

56. volumes-v3-upload [1]
✔ tempest.api.volume.test_volumes_actions.VolumesActionsTest.test_volume_upload — [Aliases]

รูปที่ 36 ผลการทดสอบประสิทธิภาพด้วย RefStack (9)

- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_show_non_existent_security_group
- ✔ tempest.api.network.test_security_groups_negative.NegativeSecGroupTest.test_show_non_existent_security_group_rule

44. volumes-list-api-versions [1]
    - ✔ tempest.api.volume.test_versions.VersionsTest.test_list_versions
45. volumes-v3-availability-zones [1]
    - ✔ tempest.api.volume.test_availability_zone.AvailabilityZoneTestJSON.test_get_availability_zone_list — [Aliases]
46. volumes-v3-clone [1]
    - ✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete_as_clone — [Aliases]
47. volumes-v3-copy-image-to-volume [2]
    - ✔ tempest.api.volume.test_volumes_actions.VolumesActionsTest.test_volume_bootable — [Aliases]
    - ✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete_from_image — [Aliases]
48. volumes-v3-create-delete [7]
    - ✔ tempest.api.volume.test_volumes_get.VolumesGetTest.test_volume_create_get_update_delete — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_invalid_size — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_nonexistent_source_volid — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_nonexistent_volume_type — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_size_negative — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_with_size_zero — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_create_volume_without_passing_size — [Aliases]
49. volumes-v3-extensions [1]
    - ✔ tempest.api.volume.test_extensions.ExtensionsTestJSON.test_list_extensions — [Aliases]
50. volumes-v3-get [3]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_get_invalid_volume_id — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_get_volume_without_passing_volume_id — [Aliases]
    - ✔ tempest.api.volume.test_volumes_negative.VolumesNegativeTest.test_volume_get_nonexistent_volume_id — [Aliases]
51. volumes-v3-list [19]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_by_name — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_by_name — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_pagination — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_details_with_multiple_params — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_pagination — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_param_display_name_and_status — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_detail_param_display_name_and_status — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_detail_param_metadata — [Aliases]
    - ✔ tempest.api.volume.test_volumes_list.VolumesListTestJSON.test_volume_list_with_details — [Aliases]

รูปที่ 37 ผลการทดสอบประสิทธิภาพด้วย RefStack (10)

Test Filters:

All | Passed | Not Passed | Flagged

Required (Total: 58 capabilities, 220 tests) (Not passed: 10 tests)

1. compute-servers-list [2]
    - ✖ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filtered_by_ip
    - ✖ tempest.api.compute.servers.test_list_server_filters.ListServerFiltersTestJSON.test_list_servers_filtered_by_ip_regex
2. identity-v3-list-projects [1]
    - ✖ tempest.api.identity.v3.test_projects.IdentityV3ProjectsTest.test_list_projects_returns_only_authorized_projects
3. networks-l2-CRUD [5]
    - ✖ tempest.api.network.test_networks.NetworksTest.test_list_networks_fields — [Aliases]
    - ✖ tempest.api.network.test_networks.NetworksTest.test_show_network_fields — [Aliases]
    - ✖ tempest.api.network.test_networks.NetworksTest.test_update_subnet_gw_dns_host_routes_dhcp — [Aliases]
    - ✖ tempest.api.network.test_ports.PortsTestJSON.test_update_port_with_security_group_and_extra_attributes
    - ✖ tempest.api.network.test_ports.PortsTestJSON.test_update_port_with_two_security_groups_and_extra_attributes
4. networks-l3-CRUD [1]
    - ✖ tempest.api.network.test_routers.RoutersTest.test_update_delete_extra_route — [Aliases]
5. networks-subnet-pools-CRUD [1]
    - ✖ tempest.api.network.test_subnetpools_extensions.SubnetPoolsTestJSON.test_create_list_show_update_delete_subnetpools

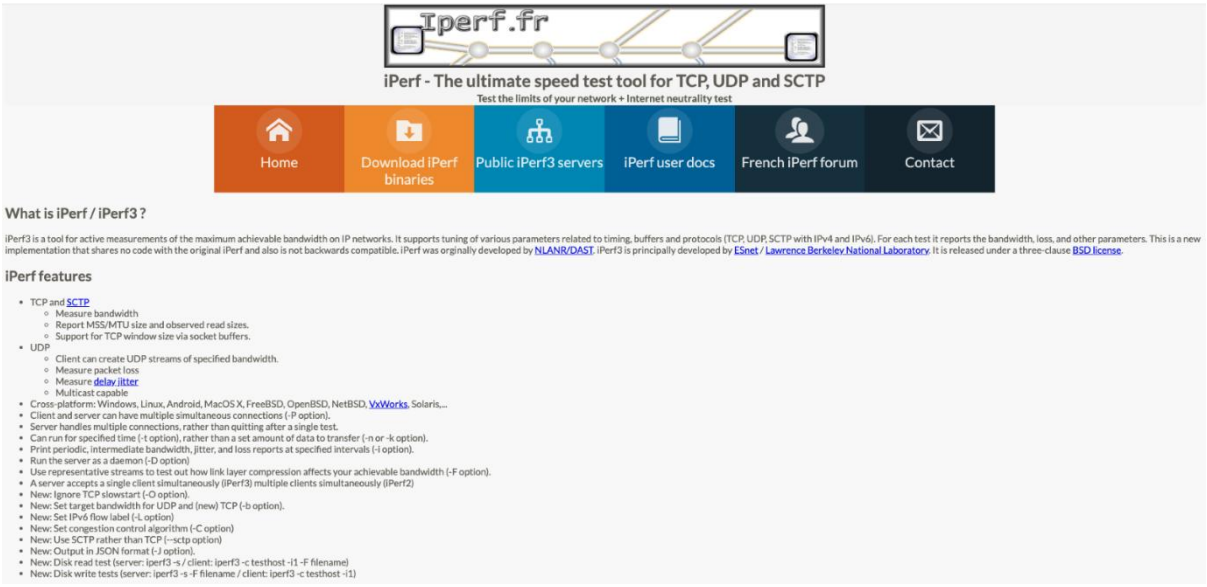
รูปที่ 38 ผลการทดสอบประสิทธิภาพด้วย RefStack (11)

รายงานผลการทดสอบประสิทธิภาพของเครือข่ายการเชื่อมต่อของ Edge Computing

การทดสอบประสิทธิภาพการทำงานของระบบ Full Edge Cloud Computing มีเป้าหมายเพื่อประเมินความสามารถในการจัดการโหลดงานที่หลากหลายและความเสถียรของระบบในสถานการณ์ที่แตกต่างกัน การทดสอบครอบคลุมถึงการวิเคราะห์การประมวลผลข้อมูลในปริมาณมาก ความรวดเร็วในการตอบสนองของระบบเมื่อมีการใช้งานพร้อมกันหลายจุด และการตรวจสอบเสถียรภาพของเครือข่ายในช่วงเวลาที่มีการถ่ายโอนข้อมูลขนาดใหญ่

ผลการทดสอบชี้ให้เห็นว่าระบบ Full Edge Cloud Computing มีความสามารถในการรองรับการประมวลผลในระดับสูง โดยสามารถตอบสนองต่อคำขอจากผู้ใช้งานได้อย่างรวดเร็วและแม่นยำ แม้ในช่วงที่มีปริมาณงานหนาแน่น ระบบยังคงรักษาประสิทธิภาพไว้ได้ในระดับที่น่าพอใจ อีกทั้งการวิเคราะห์แสดงให้เห็นว่าระบบมีความเสถียรอย่างต่อเนื่องในเครือข่ายที่ซับซ้อน นอกจากนี้ การวางโครงสร้างระบบที่ใช้เทคโนโลยี OpenStack และ OpenSDN ยังช่วยเพิ่มประสิทธิภาพของการจัดการทรัพยากร และลดความหน่วงเวลาในการถ่ายโอนข้อมูลอย่างมีประสิทธิผล

สรุปได้ว่าผลการทดสอบแสดงถึงความพร้อมของระบบ Open Cloud Computing ในการรองรับการใช้งานที่หลากหลาย พร้อมทั้งสนับสนุนการดำเนินงานของโครงการในระยะยาวได้อย่างยั่งยืน.



รูปที่ 39 เครื่องมือ iPerf สำหรับทดสอบประสิทธิภาพเครือข่าย

```
[  5] 86396.00-86397.00 sec   544 MBytes  4.56 Gbits/sec  127   2.12 MBytes
[  5] 86397.00-86398.00 sec   552 MBytes  4.63 Gbits/sec    0   2.12 MBytes
[  5] 86398.00-86399.00 sec   526 MBytes  4.41 Gbits/sec    0   2.40 MBytes
[  5] 86399.00-86400.00 sec   544 MBytes  4.56 Gbits/sec    0   2.40 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-86400.00 sec  45.2 TBytes  4.60 Gbits/sec  3011060             sender
[  5]   0.00-86400.00 sec  45.2 TBytes  4.60 Gbits/sec                  receiver

iperf Done.
```

รูปที่ 40 ผลการทดสอบประสิทธิภาพด้วย iPerf (1)

```
72 bytes from 192.168.20.102: icmp_seq=20860 ttl=64 time=1.29 ms
72 bytes from 192.168.20.102: icmp_seq=20861 ttl=64 time=1.04 ms
72 bytes from 192.168.20.102: icmp_seq=20862 ttl=64 time=0.926 ms
72 bytes from 192.168.20.102: icmp_seq=20863 ttl=64 time=0.879 ms
72 bytes from 192.168.20.102: icmp_seq=20864 ttl=64 time=1.14 ms

--- 192.168.20.102 ping statistics ---
86400 packets transmitted, 86394 received, 0.00694444% packet loss, time 87378957ms
rtt min/avg/max/mdev = 0.689/1.314/26.150/0.879 ms
```

รูปที่ 41 ผลการทดสอบประสิทธิภาพด้วย iPerf (2)

```
[  5] 86397.00-86398.00 sec   824 MBytes  6.91 Gbits/sec  197   1.49 MBytes
[  5] 86398.00-86399.00 sec   825 MBytes  6.92 Gbits/sec   33   1.39 MBytes
[  5] 86399.00-86400.00 sec   826 MBytes  6.93 Gbits/sec  119   1.26 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-86400.00 sec  81.1 TBytes  8.26 Gbits/sec  19704677             sender
[  5]   0.00-86400.00 sec  81.1 TBytes  8.26 Gbits/sec                  receiver

iperf Done.
```

รูปที่ 42 ผลการทดสอบประสิทธิภาพด้วย iPerf (3)

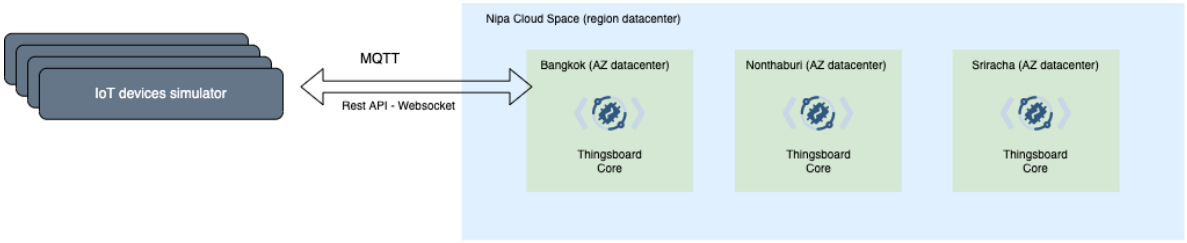
```
72 bytes from 192.168.20.100: icmp_seq=20858 ttl=64 time=0.397 ms
72 bytes from 192.168.20.100: icmp_seq=20859 ttl=64 time=0.297 ms
72 bytes from 192.168.20.100: icmp_seq=20860 ttl=64 time=0.227 ms
72 bytes from 192.168.20.100: icmp_seq=20861 ttl=64 time=0.309 ms
72 bytes from 192.168.20.100: icmp_seq=20862 ttl=64 time=0.297 ms
72 bytes from 192.168.20.100: icmp_seq=20863 ttl=64 time=0.303 ms
72 bytes from 192.168.20.100: icmp_seq=20864 ttl=64 time=0.229 ms

--- 192.168.20.100 ping statistics ---
86400 packets transmitted, 86400 received, 0% packet loss, time 88265576ms
rtt min/avg/max/mdev = 0.072/0.310/14.197/0.347 ms
```

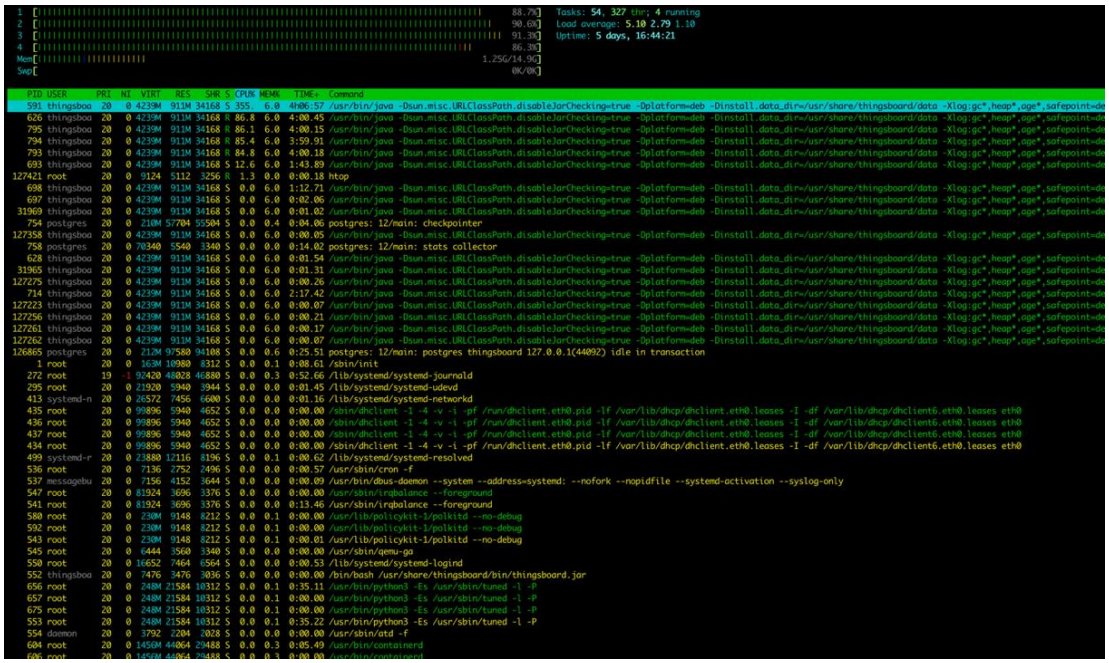
รูปที่ 43 ผลการทดสอบประสิทธิภาพด้วย iPerf (4)

รายงานผลการทดสอบประสิทธิภาพการทำงานของระบบ IoT Platform

สำหรับการทดสอบ IoT Platform ได้มีการใช้เครื่องมือจำลองอุปกรณ์ IoT (IoT devices simulator) เพื่อสร้างข้อมูลจำลองและส่งผ่านโปรโตคอล MQTT ไปยัง Core Platform (ThingsBoard Core) ซึ่งเป็นแพลตฟอร์มหลักสำหรับการจัดการและประมวลผลข้อมูลอุปกรณ์ IoT การทดสอบนี้มีเป้าหมายเพื่อประเมินประสิทธิภาพของระบบในด้านการรับส่งข้อมูล ความเร็วในการประมวลผล และการรองรับปริมาณข้อมูลที่เพิ่มขึ้นอย่างต่อเนื่องในระบบ นอกจากนี้ การเชื่อมต่อผ่านโปรโตคอล MQTT ยังช่วยให้ระบบสามารถรองรับการใช้งานที่หลากหลายและเหมาะสมกับการใช้งานจริงในสถานการณ์ที่แตกต่างกัน ดังแสดงในรูป



รูปที่ 44 โครงสร้างการทำงานของ IoT Platform สำหรับการทดสอบประสิทธิภาพ



รูปที่ 45 การใช้ทรัพยากรในระบบ ThingsBoard

รายละเอียดการทดสอบระบบ Core Platform

สเปกของ Core Platform:

- CPU: 4 vCore
- Memory: 16 GB
- จำนวนอุปกรณ์ที่จำลองในการทดสอบ: 500 อุปกรณ์
- ปริมาณข้อมูลที่ส่งต่อวินาที: 1000 Data points per second (msg/sec)

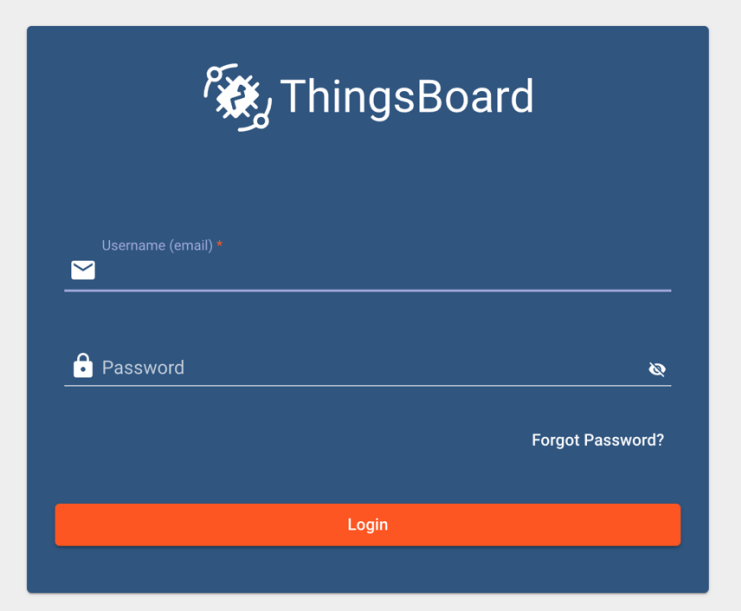
ผลการทดสอบ

จากการจำลองระบบโดยใช้อุปกรณ์ 500 ตัวที่ส่งข้อมูลจำนวน 1000 ข้อมูลต่อวินาที พบว่า CPU ของ Core Platform ทำงานที่ระดับใกล้เคียง 100% อย่างต่อเนื่อง ซึ่งแสดงถึงการใช้งานทรัพยากรที่เต็มประสิทธิภาพของระบบ ในระหว่างการทดสอบ ระบบสามารถจัดการข้อมูลได้อย่างมีเสถียรภาพ โดยไม่มีการหยุดชะงักหรือเกิดความล่าช้าในกระบวนการประมวลผล

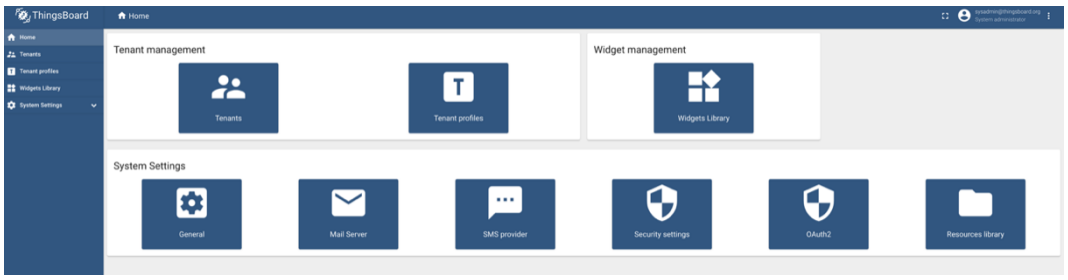
วิเคราะห์ผลการทดสอบ

การทดสอบนี้ช่วยยืนยันว่าระบบ Core Platform สามารถรองรับการประมวลผลข้อมูลในระดับสูงได้ แม้จะใช้งานทรัพยากร CPU อย่างหนัก การทำงานใกล้เคียง 100% ของ CPU บ่งบอกถึงความจำเป็นในการเพิ่มทรัพยากรในกรณีที่มีการขยายระบบหรือเพิ่มจำนวนอุปกรณ์ในอนาคต ทั้งนี้ ระบบยังมีศักยภาพที่จะพัฒนาเพิ่มเติมเพื่อรองรับปริมาณงานที่มากขึ้น โดยไม่กระทบต่อความเสถียรและประสิทธิภาพของแพลตฟอร์ม

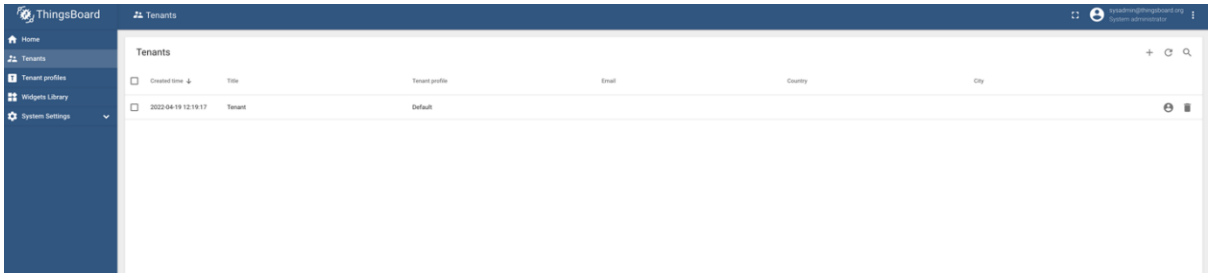
รายละเอียดการทดสอบระบบ Multi-Tenancy



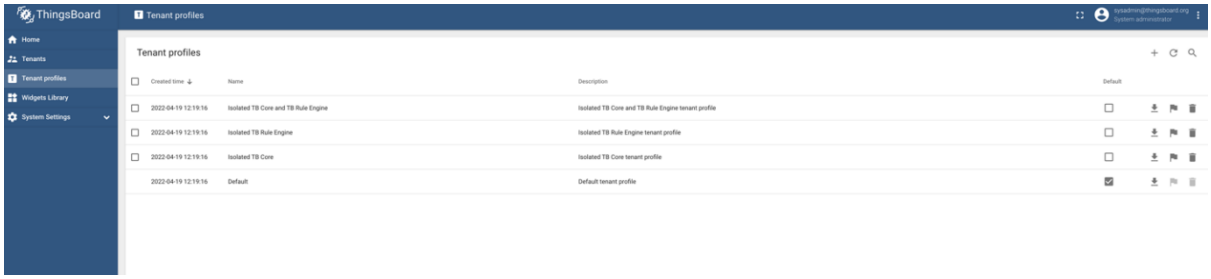
รูปที่ 46 หน้า Login ของ Thingsboard



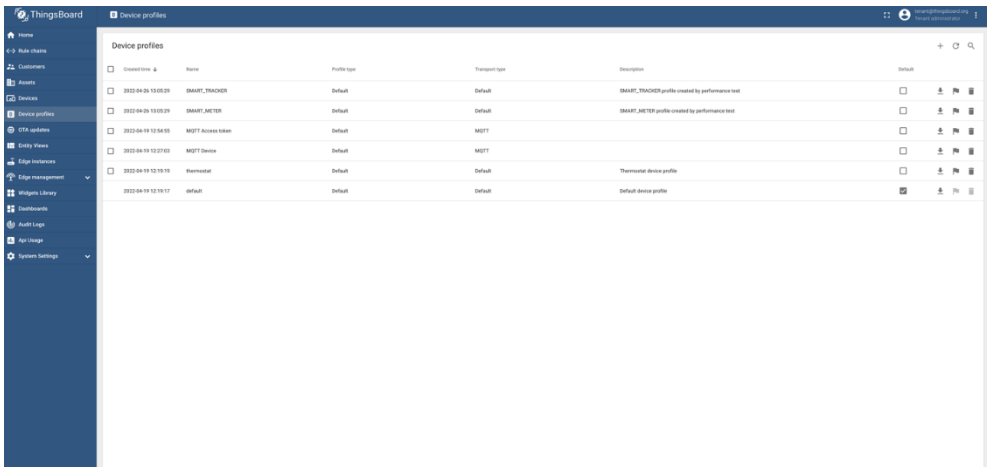
รูปที่ 47 หน้าแรก System Administrator ของระบบ IoT Platform



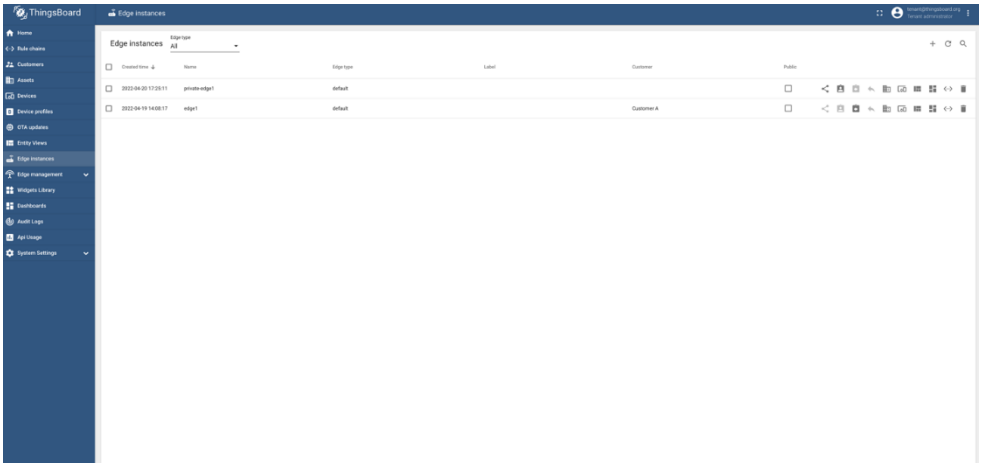
รูปที่ 48 หน้าแสดงผล Tenant ทั้งหมด ของระบบ IoT Platform



รูปที่ 49 หน้าแสดงผลและตั้งค่า Tenant Profile ของระบบ IoT Platform



รูปที่ 50 หน้าแสดงผลและตั้งค่า Device Profile ใน Tenant ของระบบ IoT Platform



รูปที่ 51 หน้าการจัดการ Edges ของ IoT Platform

ภายในระบบ ThingsBoard IoT Platform รองรับการทำงานแบบ Multi-Tenancy ได้อย่างมีประสิทธิภาพสูงสุด โดยมีความสามารถในการแยกข้อมูลและการประมวลผลของแต่ละองค์กรหรือผู้ใช้งาน (Tenants) อย่างชัดเจนและเป็นอิสระจากกันอย่างแท้จริง เพื่อลดความเสี่ยงจากการเข้าถึงข้อมูลที่ไม่ได้รับอนุญาตและเพื่อความปลอดภัยที่เหนือชั้น นอกจากนี้ระบบยังสามารถจัดสรรทรัพยากร เช่น การประมวลผล หน่วยความจำ และพื้นที่จัดเก็บข้อมูลให้ตรงกับความต้องการเฉพาะของแต่ละ Tenant ได้อย่างแม่นยำและมีประสิทธิภาพ อีกทั้งยังรองรับการทำงานของหลาย Tenants พร้อมกันในเวลาเดียวกัน โดยไม่ส่งผลกระทบต่อประสิทธิภาพการดำเนินงานหลักของระบบเลยแม้แต่น้อย ระบบยังมีความสามารถในการขยายตัวเพื่อรองรับจำนวน Tenants ที่เพิ่มขึ้นได้อย่างรวดเร็วและยืดหยุ่น พร้อมกันนี้ยังได้นำเทคโนโลยีการรักษาความปลอดภัยขั้นสูงมาใช้ เพื่อป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตในทุกกรณี ทำให้ผู้ใช้งานมั่นใจได้ว่าข้อมูลสำคัญของพวกเขาจะได้รับการปกป้องในทุกสถานการณ์

4.1.4 รายงานสรุปการให้บริการ IoT Platform ตลอดระยะเวลา 2 ปี

ตลอดระยะเวลา 2 ปีที่ผ่านมา มีผู้ลงทะเบียนใช้งาน IoT Platform รวม 70 Tenant โดยในจำนวนนี้มี 15 รายที่เชื่อมต่ออุปกรณ์รวม 345 อุปกรณ์ ซึ่งกลุ่มผู้ใช้งานหลัก (เช่น AiyaraHarn และบริษัท พรีวาเลนซ์ อุตสาหกรรม จำกัด) ถือครองอุปกรณ์กว่าครึ่งหนึ่งของยอดรวม สะท้อนการใช้งานที่ยังคงกระจุกตัวในบางองค์กร ขณะเดียวกันยังมี Tenant อีกจำนวนมากที่ยังไม่เริ่มใช้งาน ทำให้การบริหารจัดการและสนับสนุนต้องมุ่งเน้นกลยุทธ์ที่แตกต่างกันระหว่างผู้ใช้งานที่มีอุปกรณ์จำนวนมากกับผู้ที่อยู่ระหว่างการเตรียมความพร้อมในการเชื่อมต่อระบบ IoT เพิ่มเติม

ตารางที่ 14 ตารางสรุปข้อมูล Tenant และจำนวนอุปกรณ์ที่เชื่อมต่อ (ตลอดระยะเวลา 2 ปี)

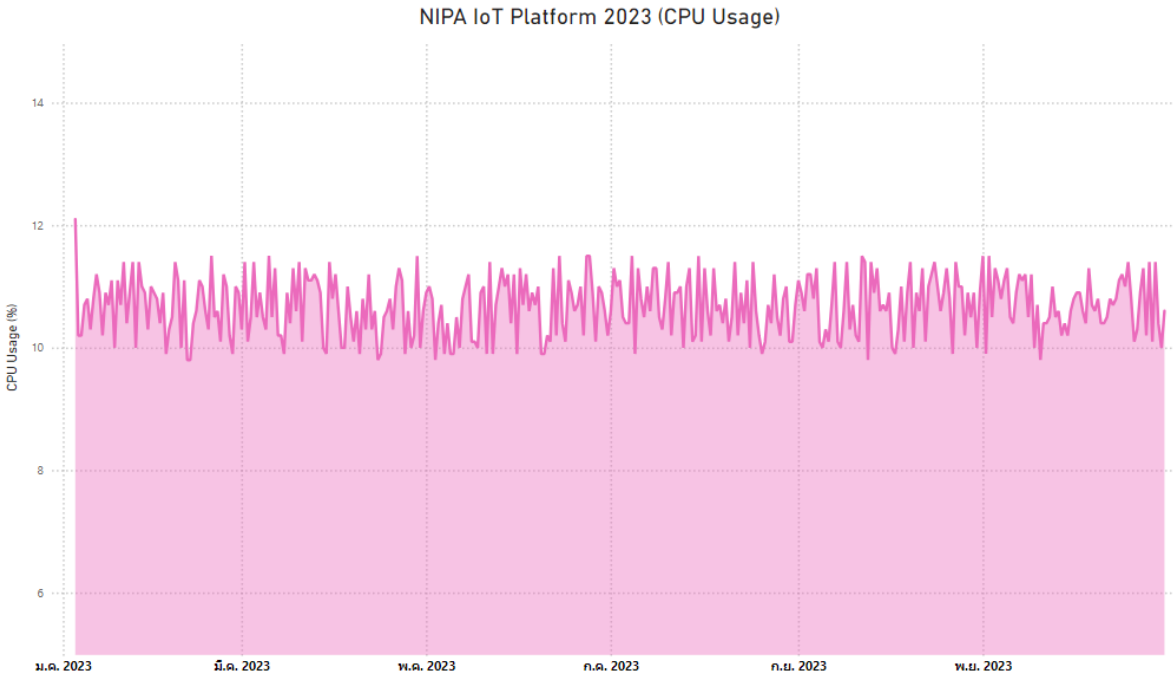
No.	Organization	Register Date	Devices	Status/Remark
1	AiyaraHarn	12/01/2566	172	✅
2	บริษัท พรีวาเลนซ์ อุตสาหกรรม จำกัด	12/01/2566	82	✅
3	บริษัท บลูโอ๊ค จำกัด	8/02/2566	34	-
4	LEMOCO	27/07/2566	14	✅
5	Sabuy wash 4	19/07/2566	12	-
6	Sabuy wash 3	19/07/2566	9	-
7	มช.quantum computer	8/02/2566	5	-
8	ศูนย์ถ่ายทอดวิศวกรรมและเทคโนโลยี มหาวิทยาลัยราชภัฏลำปาง	12/01/2566	4	-
9	fablab	21/07/2566	4	-
10	Epower	10/02/2566	3	-
11	Methaya Technology Co., Ltd.	27/04/2566	2	✅
12	คณะบริหารธุรกิจ มหาวิทยาลัยเชียงใหม่	12/01/2566	1	-
13	สภาโลหิตแห่งชาติ	12/01/2566	1	-
14	IES TECH COMPANY LIMITED	24/05/2566	1	-
15	บริษัท ไทย-ตรัง รับเบอร์ จำกัด	10/11/2566	1	-
16	บริษัท เอสซีจี ซิเมนต์-ผลิตภัณฑ์ก่อสร้าง จำกัด	12/01/2566	0	-
17	บริษัท ซูซันน์ สมาร์ท โซลูชั่น จำกัด	12/01/2566	0	-
18	มหาลัยเกษตรศาสตร์ กำแพงแสน	12/01/2566	0	-
19	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง	12/01/2566	0	-
20	มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี	12/01/2566	0	-
21	วิทยาลัยเทคนิคขอนแก่น	12/01/2566	0	-
22	โทรคมนาคม แห่งชาติ จำกัด (มหาชน)	12/01/2566	0	-

No.	Organization	Register Date	Devices	Status/Remark
23	บริษัท พี.เค. แอนด์ เค.เอส. คอนซัลแทนท์ จำกัด	12/01/2566	0	-
24	บริษัท เอสพีเจ เอทดูเคท อินโนเวชั่น	12/01/2566	0	-
25	วิทยาลัยเทคนิคพัทลุง	12/01/2566	0	-
26	บริษัท ซูโด้ฟาร์ม ภูเก็ต จำกัด	12/01/2566	0	-
27	คิงทูลส์ออโต จำกัด	12/01/2566	0	-
28	หจก. แฮปปี้ ทรานส์ โลจิสติกส์	12/01/2566	0	-
29	บริษัท อินสไปร์เทค จำกัด	12/01/2566	0	-
30	บริษัท ซีนเนอร์ยี่ เทคโนโลยี จำกัด	12/01/2566	0	-
31	มรภ.วไลยอลงกรณ์ ในพระบรมราชูปถัมภ์	12/01/2566	0	-
32	พระจอมเกล้าลาดกระบังชุมพร	12/01/2566	0	-
33	บริษัท ฮันนี่คอร์ปอเรชั่น จำกัด	12/01/2566	0	-
34	ห้างหุ้นส่วนจำกัด ขอนแก่น สมาร์ท ไอโอที	12/01/2566	0	-
35	คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี	12/01/2566	0	-
36	มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา พิษณุโลก	12/01/2566	0	-
37	บ.มงคลสมัย จำกัด	12/01/2566	0	-
38	Computech	12/01/2566	0	-
39	AI and Robotics Ventures Co.,Ltd	23/03/2566	0	-
40	eloc8	31/05/2566	0	-
41	Sabuy wash	19/07/2566	0	-
42	Sabuy wash 2	19/07/2566	0	-
43	เอส.เอส.อิเล็คทริคแอนด์ออโตเมชั่น	10/08/2566	0	-
44	OVAL Thailand Limited	21/08/2566	0	-
45	บริษัท ฟรียาบาดี (ประเทศไทย) จำกัด	4/09/2566	0	-
46	V R ENGITECH COMPANY LIMITED	5/09/2566	0	-
47	Telemerce Co.,Ltd	18/09/2566	0	-
48	IT Green	29/09/2566	0	-
49	Tinamics Co.,Ltd.	17/10/2566	0	-
50	บริษัท แสงชัยมิเตอร์ จำกัด	25/10/2566	0	-
51	บริษัท ทีพีไอ คอนกรีต	6/11/2566	0	-

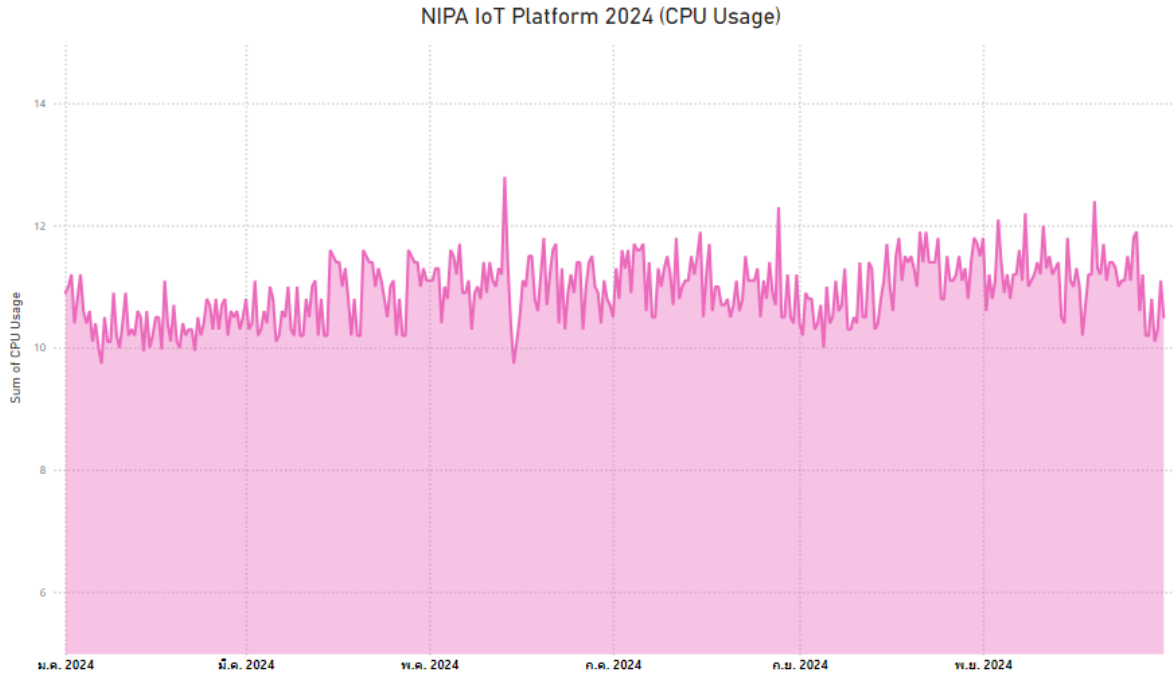
No.	Organization	Register Date	Devices	Status/Remark
52	บริษัท มิตรเทคนิคัลคอนซัลแท้นท์ จำกัด	6/11/2566	0	-
53	บริษัท สยามซินดิเคทเทคโนโลยี จำกัด	6/11/2566	0	-
54	บริษัท ชาร์จทเวนตี้โฟร์ จำกัด	6/11/2566	0	-
55	บริษัท อัลฟ่า เวิลด์ แอสเสท จำกัด	6/11/2566	0	-
56	บริษัท ยงเฮ้าส์ จำกัด	6/11/2566	0	-
57	บริษัท พร้อมขนส่ง จำกัด	6/11/2566	0	-
58	บริษัท เอสซีไอ อีเล็คตริค จำกัด	6/11/2566	0	-
59	ฟูจิฟิล์ม (ประเทศไทย) จำกัด	6/11/2566	0	-
60	บริษัท ซี-คอน ซิสเต็ม เทคโนโลยี จำกัด	6/11/2566	0	-
61	SUSUNN	6/11/2566	0	-
62	บริษัท ไอ.พี. วัน จำกัด	6/11/2566	0	-
63	บริษัท ยูเอซี โกลบอล จำกัด	6/11/2566	0	-
64	บริษัท ยงคอนกรีต จำกัด	6/11/2566	0	-
65	บริษัท แวลู ซอร์สซิ่ง จำกัด	6/11/2566	0	-
66	บริษัท สุรพลฟู้ดส์ จำกัด	6/11/2566	0	-
67	นันทพงศ์ บุญญวรรณ	10/11/2566	0	-
68	บริษัท เทคสแควร์ จำกัด	15/11/2566	0	-
69	warapon.dev	16/11/2566	0	-
70	kulrachart.d	17/11/2566	0	-

4.1.5 รายงานสถิติการทำงานของระบบ IoT Platform ตลอดระยะเวลา 2 ปี

CPU (32 Core)

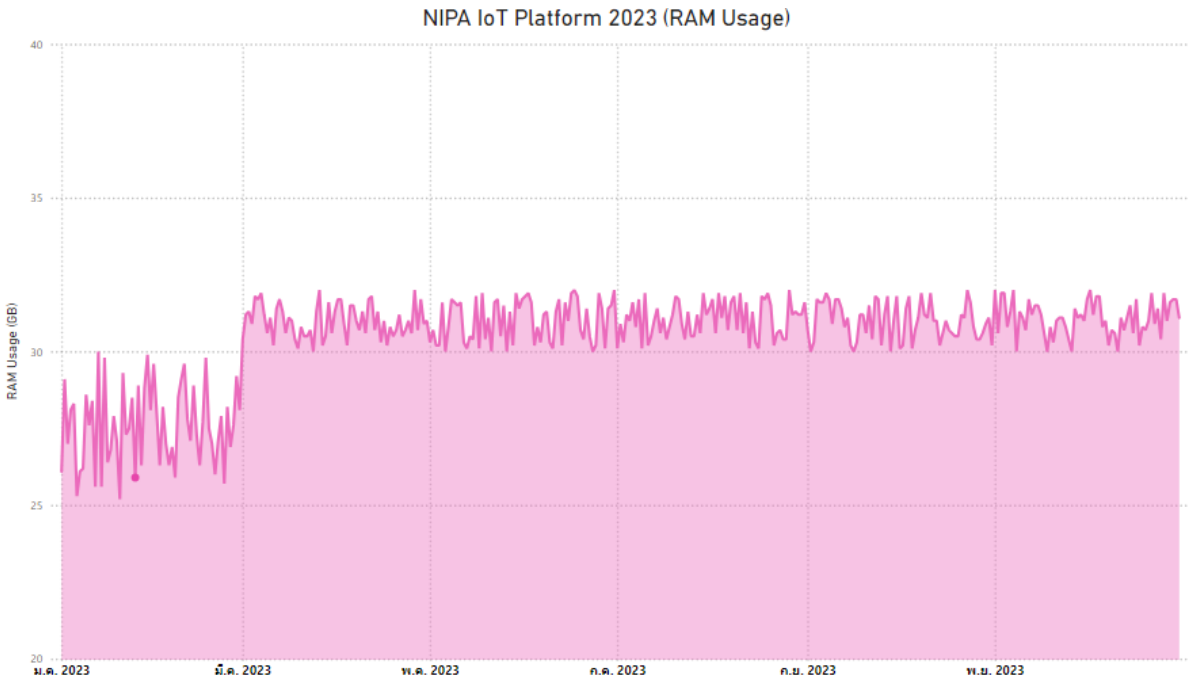


รูปที่ 52 อัตราการใช้งาน CPU ประจำปี ค.ศ. 2023 (พ.ศ. 2566)

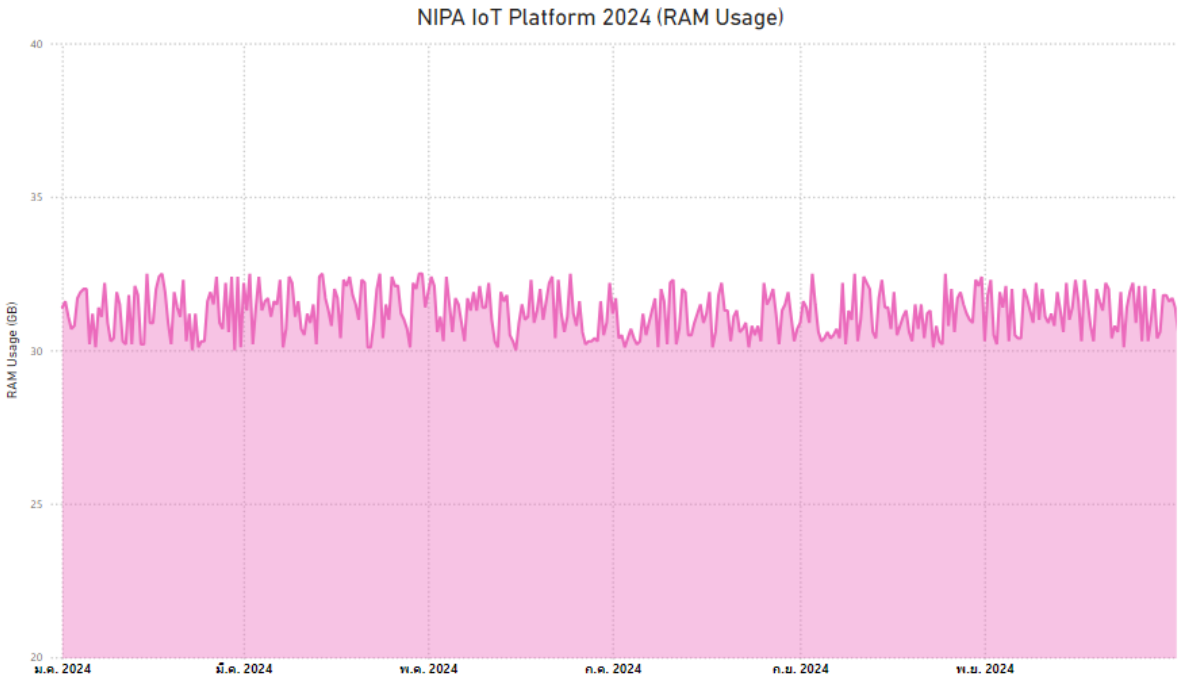


รูปที่ 53 อัตราการใช้งาน CPU ประจำปี ค.ศ. 2024 (พ.ศ. 2567)

Memory (64 GB)

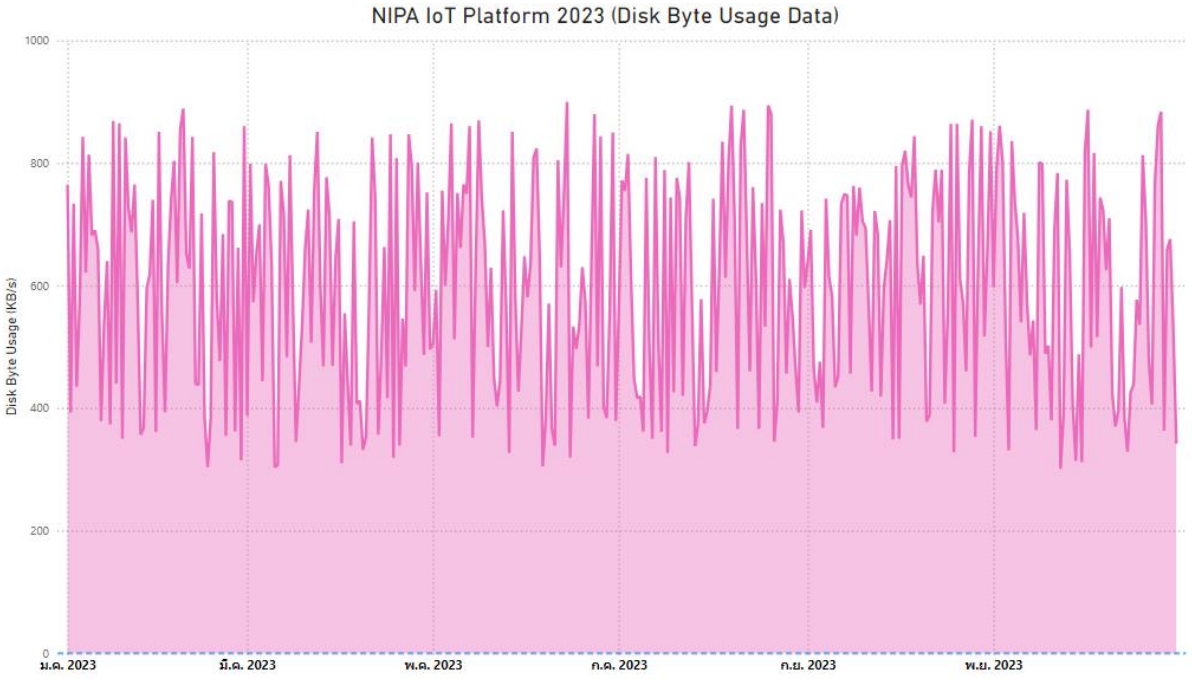


รูปที่ 54 อัตราการใช้งาน Memory ประจำปี ค.ศ. 2023 (พ.ศ. 2566)

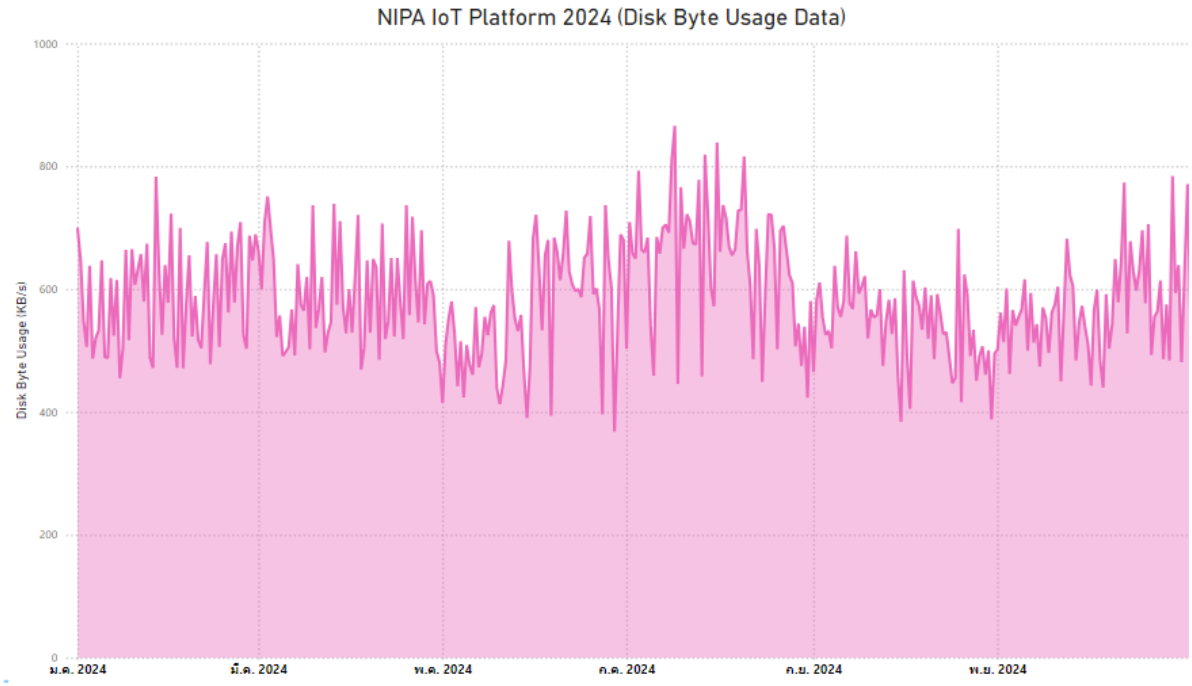


รูปที่ 55 อัตราการใช้งาน Memory ประจำปี ค.ศ. 2024 (พ.ศ. 2567)

Storage (SSD 450 GB)

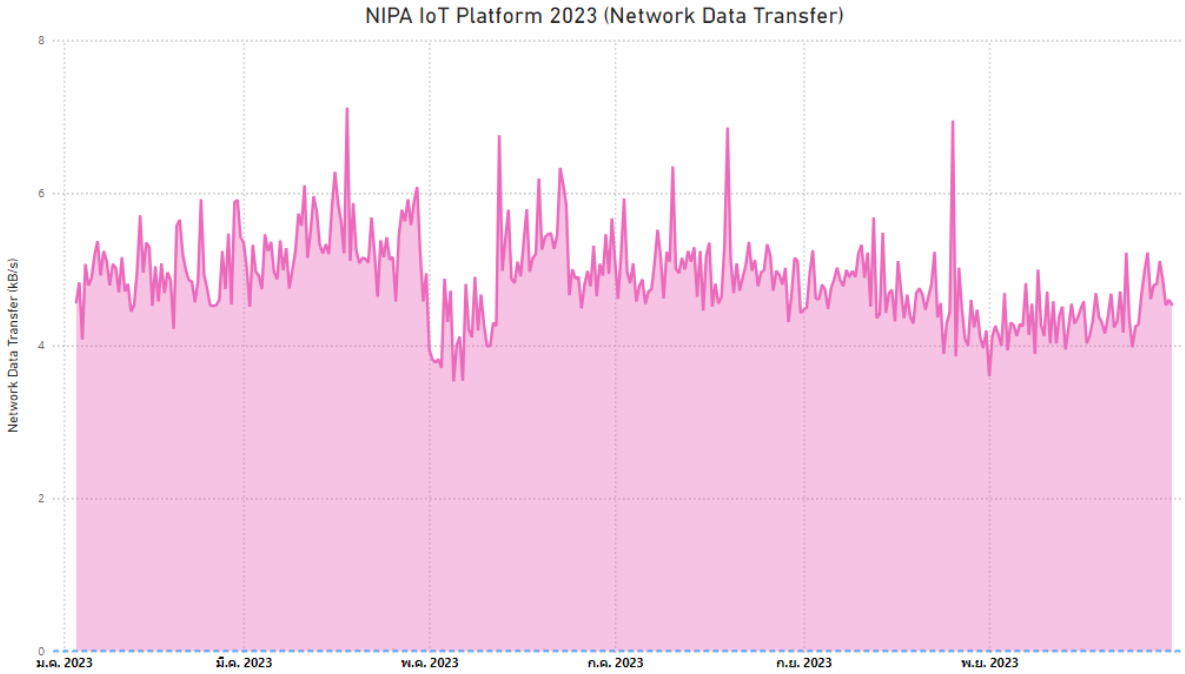


รูปที่ 56 อัตราการใช้งาน Storage ประจำปี ค.ศ. 2023 (พ.ศ. 2566)

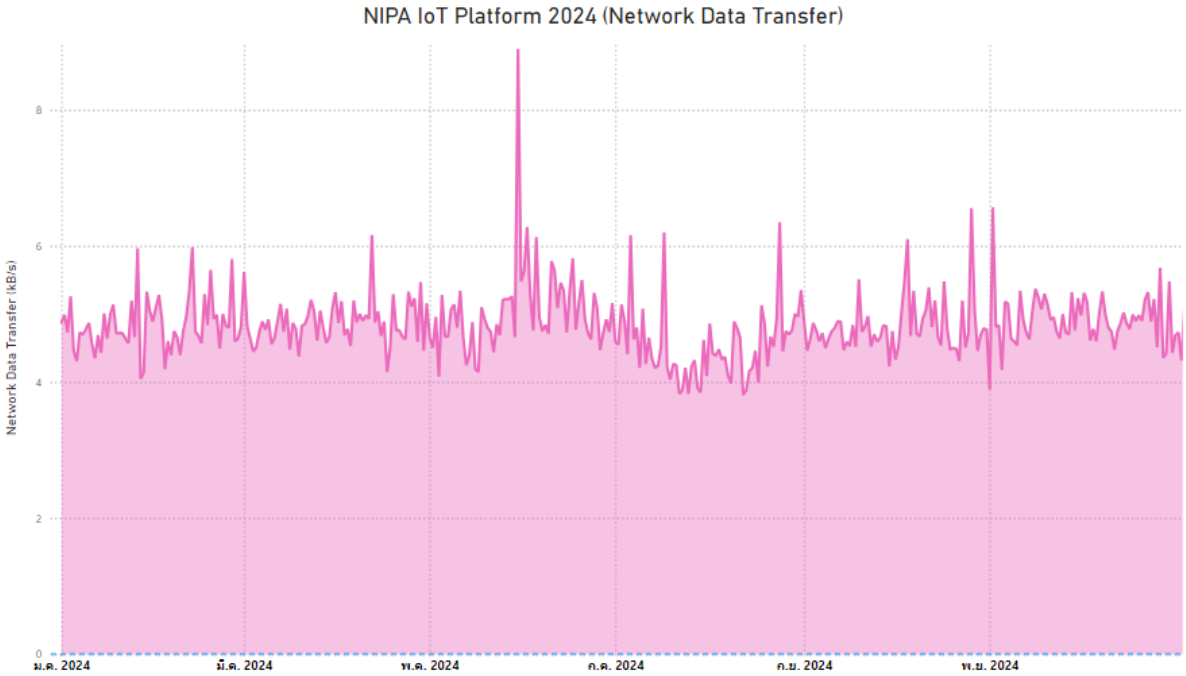


รูปที่ 57 อัตราการใช้งาน Storage ประจำปี ค.ศ. 2024 (พ.ศ. 2567)

Network (10G)



รูปที่ 58 อัตราการใช้งาน Network ประจำปี ค.ศ. 2023 (พ.ศ. 2566)

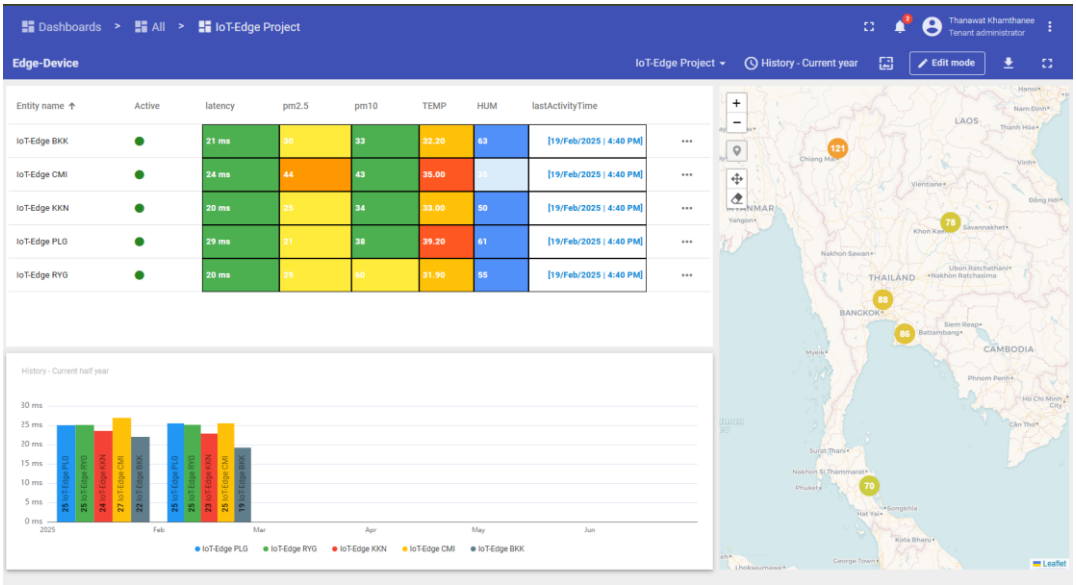


รูปที่ 59 อัตราการใช้งาน Network ประจำปี ค.ศ. 2024 (พ.ศ. 2567)

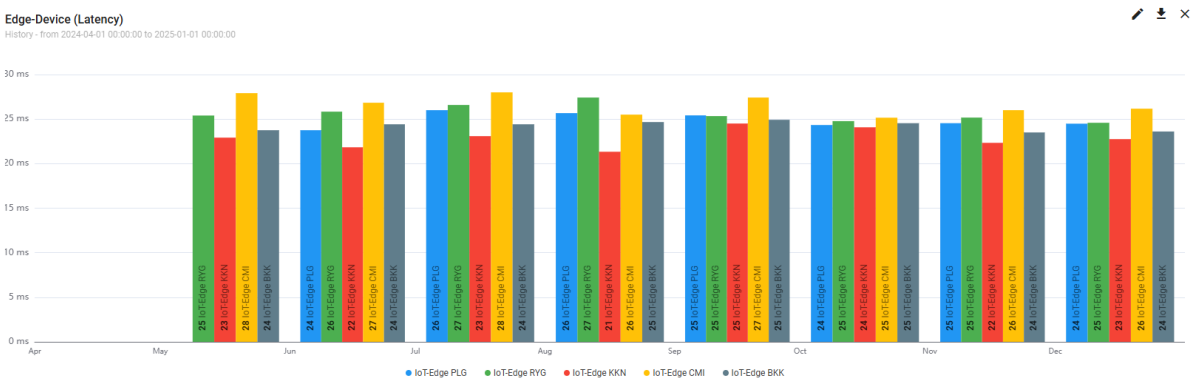
Latency

การวิเคราะห์ค่าหน่วงเวลา (Latency) เป็นองค์ประกอบสำคัญในการประเมินประสิทธิภาพของระบบเครือข่ายและอุปกรณ์ IoT การวัดค่า Latency ระหว่าง Platform IoT กับอุปกรณ์ Full-Edge และ IoT-Edge ทำให้สามารถวิเคราะห์ประสิทธิภาพการรับส่งข้อมูล รวมถึงระบุจุดที่เป็นข้อจำกัดของระบบเครือข่ายได้อย่างแม่นยำ ความสามารถของเครือข่ายขึ้นอยู่กับปริมาณการใช้งานในช่วงเวลานั้น ซึ่งอาจส่งผลให้ค่า Latency เปลี่ยนแปลงได้ ปัจจัยอื่น ๆ เช่น ลักษณะของสภาพแวดล้อมเครือข่ายของแต่ละผู้ให้บริการหรือข้อจำกัดทางฮาร์ดแวร์ของอุปกรณ์ซึ่งมีผลโดยตรงต่อเสถียรภาพของระบบและคุณภาพของการสื่อสารข้อมูล

สำหรับระบบ IoT-Edge ทำหน้าที่เป็นโครงสร้างพื้นฐานสำคัญสำหรับการเชื่อมต่อและประมวลผลข้อมูลจากเซ็นเซอร์ เช่น เซ็นเซอร์วัดค่าฝุ่น PM และเซ็นเซอร์ตรวจวัดอุณหภูมิและความชื้นในอากาศ โดยอาศัยการสื่อสารผ่านเครือข่าย 4G LTE ซึ่งประสิทธิภาพในการรับส่งข้อมูลอาจได้รับผลกระทบจากปริมาณการใช้งานเครือข่ายของผู้ให้บริการในขณะนั้น ดังนั้น การเฝ้าติดตาม Latency และการวิเคราะห์ข้อมูลที่ได้รับจะช่วยให้สามารถปรับปรุงโครงสร้างเครือข่าย ลดความล่าช้า และเพิ่มเสถียรภาพของระบบ IoT Platforms เพื่อให้รองรับการประมวลผลแบบเรียลไทม์ได้อย่างมีประสิทธิภาพ



รูปที่ 60 Real-Time IoT-Edge Dashboard



รูปที่ 61 Latency เฉลี่ยรายเดือนของ IoT-Edge ประจำปี ค.ศ. 2024 (พ.ศ. 2567)

ตารางที่ 15 สรุปค่า Latency ระหว่าง Edge Computing (Full-Edge & IoT-Edge) กับ ThingsBoard Platform IoT

Source	Destination	Min Latency (ms)	Max Latency (ms)	Avg Latency (ms)
Full-Edge BKK	IoT platform (https://iot.nipa.cloud)	0.227	0.397	0.347
Full-Edge KKN	IoT platform (https://iot.nipa.cloud)	0.879	1.29	0.879
IoT-Edge BKK	IoT platform (https://iot.nipa.cloud)	20	28	24
IoT-Edge KKN	IoT platform (https://iot.nipa.cloud)	19	26	23
IoT-Edge CMI	IoT platform (https://iot.nipa.cloud)	22	31	26
IoT-Edge RYG	IoT platform (https://iot.nipa.cloud)	21	29	25
IoT-Edge PLG	IoT platform (https://iot.nipa.cloud)	22	29	25

4.1.6 ผลลัพธ์การประเมินและความคิดเห็นเกี่ยวกับ IoT Platform ของผู้ใช้บริการ

จากการรวบรวมข้อมูลความคิดเห็นของผู้ใช้บริการเกี่ยวกับ IoT Platform ThingsBoard ซึ่งได้จากแบบสำรวจหลังการทดลองใช้งานในกิจกรรม อบรม Workshop “โครงการการเรียนรู้และเสริมศักยภาพอุตสาหกรรมไทยด้วยระบบ IoT" พบว่าผู้เข้าร่วม ซึ่งเป็นกลุ่มผู้เข้าร่วมจากหลากหลายประเภท ได้แก่ Startup (40%), ผู้ประกอบการ SME (20%), Systems Integrator (20%), องค์กรขนาดใหญ่ (13%), และบุคคลทั่วไป (7%) โดยมีความสนใจในอุปกรณ์และเทคโนโลยีที่สามารถประยุกต์ใช้ในภาคอุตสาหกรรม ให้ข้อมูลเชิงลึกเกี่ยวกับประเด็นสำคัญ ได้แก่ อุปสรรคในการใช้งาน แนวทางการประยุกต์ใช้ในภาคอุตสาหกรรม และข้อเสนอแนะในการพัฒนาและส่งเสริมการใช้งาน IoT ในภาคธุรกิจไทย โดยสรุปได้ดังนี้

อุปสรรคในการใช้งาน IoT

- ต้นทุนและงบประมาณ (15%)
  ผู้ใช้ระบุว่าต้นทุนในการเริ่มต้นติดตั้งระบบเป็นอุปสรรคสำคัญต่อการตัดสินใจลงทุน
- การขาดความรู้และความเข้าใจ (8%)
  ผู้ใช้สะท้อนถึงปัญหาการขาดความรู้เกี่ยวกับการนำ IoT มาประยุกต์ใช้ในกระบวนการทางธุรกิจ
- ข้อจำกัดด้านฮาร์ดแวร์ (21%)
  ผู้ใช้พบว่าอุปกรณ์ที่มีอยู่ไม่สามารถเชื่อมต่อกับแพลตฟอร์มได้อย่างมีประสิทธิภาพ
- ข้อจำกัดด้านซอฟต์แวร์ (24%)
  ผู้ใช้รายงานว่าซอฟต์แวร์ไม่รองรับหรือไม่สอดคล้องกับระบบที่มีอยู่ ส่งผลให้เกิดความล่าช้าและข้อผิดพลาดในการประมวลผลข้อมูล
- ปัญหาด้านการบริหารการเปลี่ยนแปลง (15%)
  ผู้ใช้ระบุว่าองค์กรยังขาดแนวทางการบริหารจัดการการเปลี่ยนแปลง (Change Management) ทำให้เกิดความล่าช้าในการนำระบบไปใช้จริง
- ความคิดเห็นเพิ่มเติม (6%)
  ผู้ใช้เสนอแนวทางแก้ไขเพิ่มเติมเกี่ยวกับอุปสรรคเฉพาะที่ส่งผลกระทบต่อการใช้งานระบบ IoT

การประยุกต์ใช้ความรู้จากการอบรม

- การใช้งานกับเครื่องจักรในภาคอุตสาหกรรม: การติดตาม/ควบคุมกระบวนการผลิต (43%)
  ผู้เข้าร่วมอบรมระบุว่าการใช้ IoT กับเครื่องจักรสามารถช่วยเพิ่มประสิทธิภาพในการควบคุมกระบวนการผลิต ลดต้นทุนการดำเนินงาน และเพิ่มความสามารถในการวิเคราะห์ข้อมูลแบบเรียลไทม์
- การใช้งานกับอุปกรณ์ Power Meter: การแสดงผลข้อมูลการใช้พลังงาน (24%)
  การนำ IoT มาใช้ในการวัดและแสดงผลข้อมูลการใช้พลังงาน ช่วยให้องค์กรสามารถติดตามการใช้พลังงานและวิเคราะห์แนวโน้มการใช้ไฟฟ้าเพื่อเพิ่มประสิทธิภาพในการจัดการต้นทุนพลังงาน
- การประยุกต์ใช้ในอุตสาหกรรมการบำรุงรักษาและซ่อมแซม: การติดตามและวิเคราะห์ข้อมูลจากเครื่องจักร (24%)
  IoT ช่วยให้สามารถตรวจสอบสภาพการทำงานของเครื่องจักร วิเคราะห์แนวโน้มการเสียหาย และคาดการณ์การซ่อมบำรุงล่วงหน้า (Predictive Maintenance) เพื่อลดเวลาหยุดทำงานของอุปกรณ์
- การประยุกต์ใช้งานด้านอื่น ๆ (10%)
  ผู้เข้าร่วมอบรมยังพบว่า IoT สามารถนำไปประยุกต์ใช้ในหลากหลายด้าน เช่น การบริหารจัดการระบบขนส่ง การเกษตรอัจฉริยะ และการควบคุมระบบแสงสว่างในอาคาร เพื่อลดต้นทุนและเพิ่มประสิทธิภาพในการดำเนินงาน

ข้อเสนอแนะเพื่อส่งเสริมการใช้งาน IoT/IIoT ในอุตสาหกรรมไทย

- สนับสนุนการอบรมบุคลากรและผู้ที่สนใจ (38%)
  การเพิ่มหลักสูตรอบรมและกิจกรรมเชิงปฏิบัติการเพื่อให้บุคลากรสามารถเรียนรู้และนำ IoT ไปประยุกต์ใช้ในอุตสาหกรรมได้อย่างมีประสิทธิภาพ
- กำหนดมาตรฐานสำหรับการใช้งาน (15%)
  การพัฒนามาตรฐานกลางสำหรับอุตสาหกรรมไทยเพื่อให้การใช้งาน IoT เป็นไปในทิศทางเดียวกันและสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ
- จัดแสดงตัวอย่างโครงการที่ประสบความสำเร็จ (19%)
  การเผยแพร่กรณีศึกษาขององค์กรที่ประสบความสำเร็จในการใช้ IoT เพื่อสร้างความเชื่อมั่นและเป็นแนวทางให้แก่ธุรกิจอื่น ๆ
- สนับสนุนและจัดหาอุปกรณ์คุณภาพสูงในราคาที่เหมาะสม (23%)
  การสนับสนุนให้มีการผลิตและจัดหาฮาร์ดแวร์ IoT ที่มีคุณภาพดีในราคาที่สามารถเข้าถึงได้ เพื่อลดต้นทุนการลงทุนของธุรกิจที่ต้องการนำเทคโนโลยีไปใช้
- ข้อเสนอแนะอื่น ๆ (4%)
  การพัฒนาโครงสร้างพื้นฐานและนโยบายสนับสนุนเพิ่มเติม เช่น การลดภาษีอุปกรณ์ IoT หรือให้สิทธิประโยชน์แก่บริษัทที่ลงทุนด้าน IoT

สรุปผลการประเมิน

ภาคธุรกิจและอุตสาหกรรมหลายแห่งยังอยู่ในช่วงเริ่มต้นของการนำเทคโนโลยี IoT มาใช้งาน โดยเผชิญกับอุปสรรคหลัก ได้แก่ ต้นทุนที่สูง การขาดแคลนบุคลากรที่มีทักษะเฉพาะ และความเข้าใจที่ไม่เพียงพอเกี่ยวกับเทคโนโลยี ดังนั้น การให้บริการแพลตฟอร์ม IoT ควรมุ่งเน้นการพัฒนาความรู้เชิงเทคนิค การเสริมศักยภาพบุคลากรผ่านการอบรม และการสร้างความตระหนักถึงคุณค่าของข้อมูลที่เก็บรวบรวม เพื่อผลักดันให้เกิดการใช้งานที่มีประสิทธิภาพและสร้างประโยชน์สูงสุดให้กับอุตสาหกรรมไทย

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

5.1.1 เชิงการวิจัย

ผลการดำเนินงานและข้อมูลที่ได้จากโครงการนี้ได้สร้างฐานรากสำคัญสำหรับการวิจัยในอนาคตในด้าน Edge Cloud Computing และ IoT Platform โดยเน้นการพัฒนาความสามารถในการประมวลผลข้อมูลแบบกระจายที่ลดข้อจำกัดของศูนย์ข้อมูลกลาง ระบบที่พัฒนาขึ้นมีศักยภาพในการรองรับการเชื่อมต่อกับอุปกรณ์จำนวนมากในเครือข่าย โดยยังคงรักษาเสถียรภาพได้ดีแม้ในสถานการณ์ที่มีการใช้งานหนาแน่น ระบบดังกล่าวถูกออกแบบให้สามารถรองรับการทำงานแบบขยายตัวได้โดยไม่ส่งผลกระทบต่อประสิทธิภาพการทำงานในระดับสูงสุด
ทั้งนี้ยังมีการประยุกต์เทคโนโลยีการประมวลผลที่หลากหลายเพื่อเพิ่มความยืดหยุ่นและรองรับการทำงานในหลากหลายบริบท เช่น การวิเคราะห์ข้อมูลในพื้นที่ที่มีความสำคัญเร่งด่วน การใช้งานในพื้นที่ที่มีข้อจำกัดด้านโครงสร้างพื้นฐาน และการพัฒนาความสามารถให้รองรับความต้องการที่เพิ่มขึ้นของผู้ใช้งาน

การพัฒนาในลักษณะนี้ยังช่วยเพิ่มความมั่นใจแก่ผู้ใช้งานในด้านความปลอดภัยของข้อมูล โดยเฉพาะในบริบทที่ต้องการการจัดการข้อมูลในลักษณะสำคัญ เช่น ข้อมูลการแพทย์ หรือข้อมูลอุตสาหกรรมที่มีมูลค่าสูง ความสามารถในการรองรับและประมวลผลข้อมูลแบบเรียลไทม์ของระบบยังเป็นหนึ่งในจุดเด่นสำคัญที่ช่วยลดเวลาในการตัดสินใจและเพิ่มประสิทธิภาพในการดำเนินงานอย่างมีนัยสำคัญ โดยระบบยังคงเสถียรแม้ในสถานการณ์ที่เครือข่ายถูกใช้งานในปริมาณมาก นอกจากนี้โครงการยังสร้างกลไกที่สามารถรองรับการขยายตัวของเทคโนโลยีในอนาคต ช่วยเพิ่มโอกาสในการนำไปปรับใช้ในบริบทใหม่ ๆ ได้อย่างต่อเนื่องและยั่งยืน

- **ปัญหาที่พบ**

1. **ข้อจำกัดด้านฮาร์ดแวร์และซอฟต์แวร์**:
   อุปกรณ์ทั้ง hardware และ software
   ที่ผู้ประกอบการได้ใช้งานในปัจจุบันนั้นไม่สามารถรองรับการเชื่อมต่อ IoT/IIoT ได้โดยตรง เช่น
   ขาดการสนับสนุนโปรโตคอลสมัยใหม่ การเชื่อมต่อเครือข่าย หรือการประมวลผลที่เหมาะสม
   รวมถึงระบบซอฟต์แวร์ที่ไม่สามารถบูรณาการกับแพลตฟอร์ม IoT ได้ ส่งผลให้ต้องใช้อุปกรณ์เสริม เช่น IoT gateway
   หรือปรับปรุงระบบให้ทันสมัยเพื่อรองรับการเก็บและวิเคราะห์ข้อมูลแบบเรียลไทม์อย่างมีประสิทธิภาพ
2. **ความกังวลด้านความปลอดภัยของข้อมูล**:
   ผู้ประกอบการยังคงมีความกังวลเกี่ยวกับความปลอดภัยของข้อมูล
   เนื่องจากข้อมูลส่วนใหญ่จำเป็นต้องเก็บไว้ภายในองค์กรเพื่อป้องกันการรั่วไหลหรือการโจมตีจากภายนอก ซึ่งเป็นความท้าทายที่ต้องการมาตรการด้านความปลอดภัยที่มีประสิทธิภาพสูง

3. **การขาดความรู้ในเทคโนโลยี**:
   ผู้ประกอบการขาดแคลนความรู้และความเข้าใจในการประยุกต์ใช้งาน Edge Computing และ IoT/IIoT ในภาคธุรกิจของตนเอง ทำให้การนำเทคโนโลยีเหล่านี้ไปใช้งานจริงมีข้อจำกัด
   และผู้ประกอบการบางส่วนยังไม่ตระหนักถึงศักยภาพของเทคโนโลยีดังกล่าวในการเพิ่มประสิทธิภาพการทำงาน
4. **ค่าใช้จ่ายในการลงทุน**: การลงทุนในระบบ Edge Computing และ IoT ยังคงถูกมองว่าไม่คุ้มค่า โดยเฉพาะในกลุ่มธุรกิจขนาดเล็กถึงขนาดกลางที่มีข้อจำกัดด้านงบประมาณ
   เนื่องจากค่าใช้จ่ายเริ่มต้นและค่าบำรุงรักษาระบบยังคงสูงเมื่อเปรียบเทียบกับผลตอบแทนที่คาดหวังในระยะสั้น

- **แนวทางแก้ไข**

1. **การติดตั้งอุปกรณ์ฮาร์ดแวร์ในภูมิภาคที่เหมาะสม**: ติดตั้งอุปกรณ์ Hardware Full Edge-อยู่ใน 2 ภูมิภาคหลักคือ 1. นนทบุรี (NON) 2.ขอนแก่น (KKN)
   เพื่อเพิ่มความสามารถในการประมวลผลข้อมูลแบบกระจาย ลดเวลาในการส่งข้อมูล
   และเพิ่มความเสถียรในการทำงานของระบบโดยรวม
2. **ออกแบบโครงสร้าง IoT Edge ที่มีความยืดหยุ่น**: ออกแบบโครงสร้าง IoT Edge สำหรับ 3 ภูมิภาคใหม่ (รวมทั้ง 2 ภูมิภาคเดิม) รวมเป็น 5 ภูมิภาค โดยคำนึงถึงความยืดหยุ่นใช้งาน
   รวมถึงการเชื่อมต่อและจัดการอุปกรณ์ IoT โดยใช้ ThingsBoard IoT Edge
   ซึ่งช่วยเพิ่มความสามารถในการจัดการข้อมูลแบบกระจาย ลดความหน่วงของเครือข่าย
   และเพิ่มประสิทธิภาพการทำงานในแต่ละภูมิภาค
3. **ส่งข้อมูลจาก IoT Edge ไปยังแพลตฟอร์มกลาง**: ส่งข้อมูล อุณหภูมิ, ความชื้น, ค่า PM จาก IoT Edge ทั้ง 5 ภูมิภาคไปยัง IoT Platform กลาง (นนทบุรี) ด้วยเทคโนโลยีที่ลด Latency
   และเพิ่มความปลอดภัยในการส่งผ่านข้อมูล
   เพื่อให้มั่นใจว่าข้อมูลสำคัญจะถูกประมวลผลได้อย่างรวดเร็วและถูกต้อง
4. **พัฒนาหน้าแสดงผลบนแพลตฟอร์มกลาง**: ออกแบบและพัฒนาหน้าแสดงผลข้อมูลบน IoT Platform กลาง (Centralized Management) โดยเพิ่มฟีเจอร์ในการควบคุมอุปกรณ์แบบเรียลไทม์
   และอินเทอร์เฟซที่ใช้ง่ายเพื่อช่วยให้ผู้ใช้งานสามารถติดตามสถานะและตัดสินใจได้อย่างมีประสิทธิภาพ

- **ผลลัพธ์ตามเป้าของโครงการ**

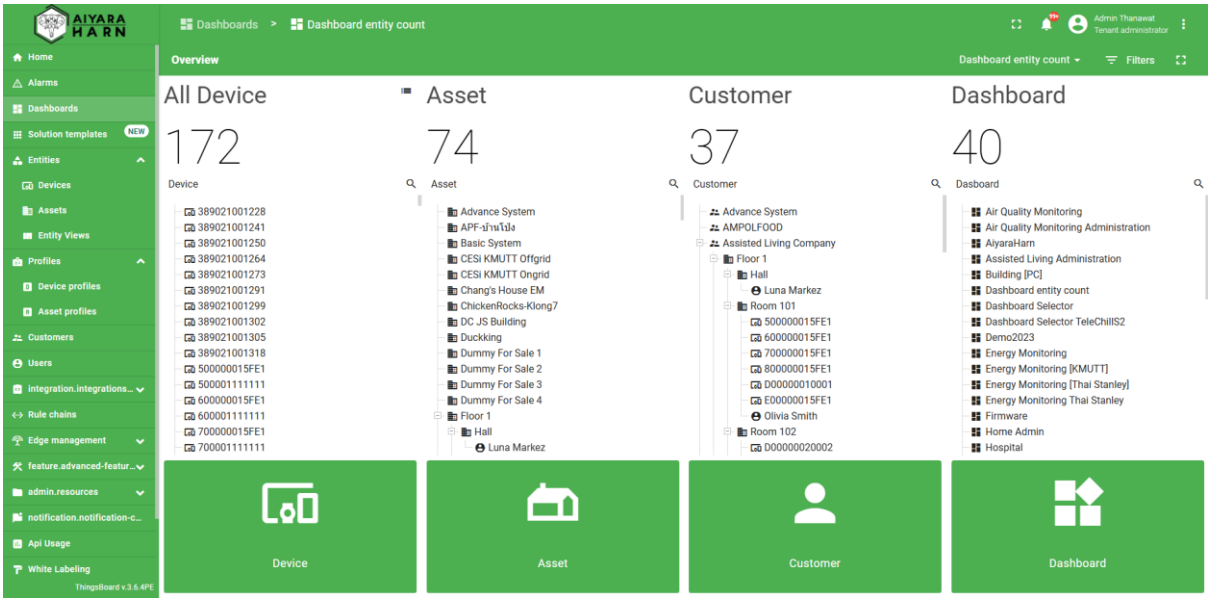
1. **Edge Cloud Computing & IoT Platform ที่ใช้งานง่าย** :
   ที่ใช้งานง่ายและออกแบบมาให้รองรับการเชื่อมต่อกับอุปกรณ์ IoT จำนวนมากได้อย่างราบรื่น
   โดยระบบนี้สามารถประมวลผลข้อมูลได้ใกล้แหล่งกำเนิดช่วยลด Latency
   และเพิ่มประสิทธิภาพการทำงาน
   อีกทั้งยังรองรับการเชื่อมต่อแบบเรียลไทม์เพื่อส่งต่อข้อมูลสำคัญไปยังระบบคลาวด์หรือศูนย์กลางการจัดการ

2. **Dashboard สำหรับผู้ใช้งานระบบ** : Thingsboard IoT Platform มี Dashboard สำหรับผู้ใช้งานระบบ (Tenat admin) หรือ Customer ที่ช่วยให้การบริหารจัดการง่ายและรวดเร็ว โดยผู้ใช้งานสามารถเข้าถึงข้อมูลสำคัญ เช่น ข้อมูลอุปกรณ์ IoT และสถานะการทำงาน พร้อมการแสดงผลในรูปแบบกราฟที่สามารถปรับแต่งได้ตามความต้องการ
3. **Dashboard สำหรับการบริหารจัดการต่าง ๆ ในระบบ** : มี Dashboard สำหรับการบริหารจัดการ (System Admin Management Service) ใน ThingsBoard IoT Platform ออกแบบมาเพื่อให้ผู้ดูแลระบบสามารถจัดการผู้เช่า (Tenants) ได้อย่างมีประสิทธิภาพ สามารถเพิ่ม-ลบ หรือปรับแต่งสิทธิ์ของผู้เช่า รวมถึงตรวจสอบสถานะการใช้งานของผู้เช่าแต่ละรายอย่างละเอียด โดยข้อมูลจะแสดงในรูปแบบกราฟและรายงานที่เข้าใจง่าย นอกจากนี้ยังสามารถตั้งค่าการแจ้งเตือนเมื่อมีความผิดปกติในระบบหรือการใช้งานที่เกินขีดจำกัด เพื่อเพิ่มประสิทธิภาพและความปลอดภัยในการจัดการระบบ
4. **API** : ThingsBoard มีระบบ API ที่ครอบคลุมทั้ง REST API และ MQTT API ซึ่งช่วยให้ผู้ใช้งานสามารถเชื่อมต่อและสื่อสารข้อมูลกับอุปกรณ์ IoT หรือระบบภายนอกได้อย่างยืดหยุ่น REST API ช่วยในการจัดการทรัพยากร เช่น ผู้เช่า อุปกรณ์ และข้อมูลเทเลเมตริก ขณะที่ MQTT API เหมาะสำหรับการส่งและรับข้อมูลแบบเรียลไทม์ ทำให้สามารถผสานรวมกับ Application ต่าง ๆ ได้ง่ายและมีประสิทธิภาพ
5. **Multi-Tenancy** : ThingsBoard IoT Platform รองรับการใช้งานในรูปแบบ Multi-Tenancy อย่างมีประสิทธิภาพ โดยช่วยให้องค์กรสามารถจัดการผู้เช่า (Tenants) หลายรายในระบบเดียวได้อย่างเป็นระเบียบ ผู้ดูแลระบบสามารถเพิ่ม ลบ และกำหนดสิทธิ์การเข้าถึงของแต่ละ Tenant ได้ตามความเหมาะสม รวมถึงการจัดการทรัพยากร เช่น การแยกข้อมูลเทเลเมตริกของแต่ละ Tenant เพื่อความปลอดภัยและความเป็นส่วนตัวของข้อมูล นอกจากนี้ยังรองรับการขยายตัวของระบบได้อย่างยืดหยุ่น ทำให้องค์กรขนาดใหญ่สามารถรองรับการใช้งานพร้อมกันได้ถึง 500 บริษัทหรือมากกว่า

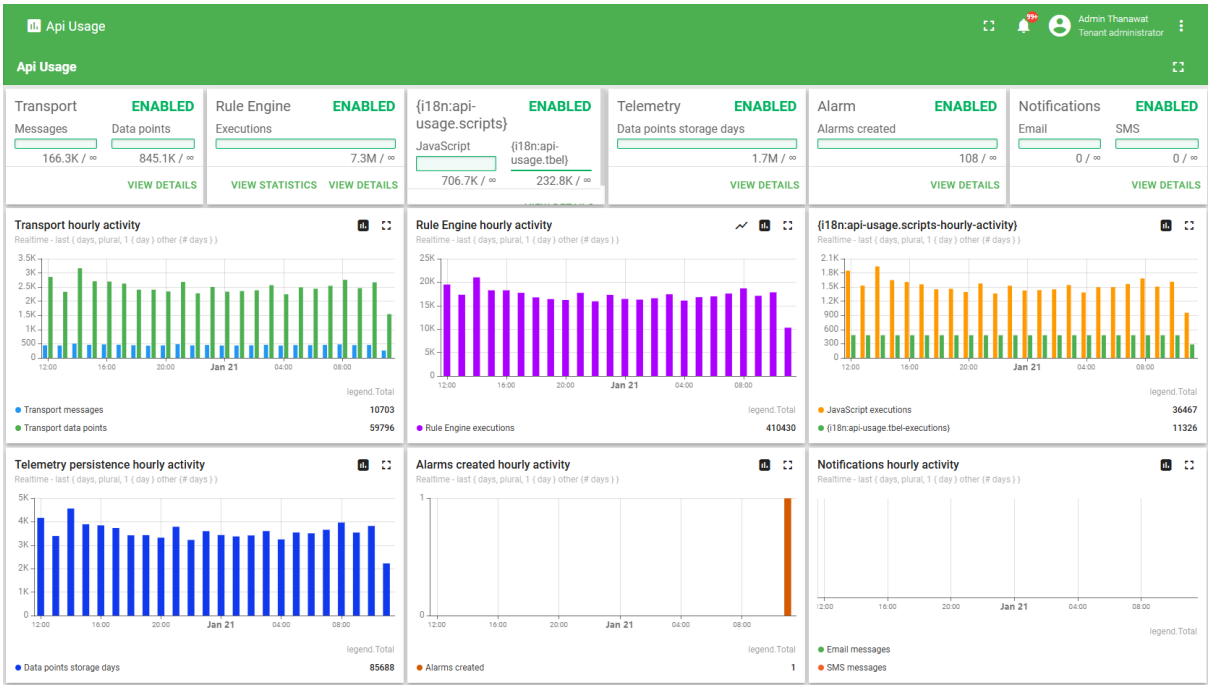
- **สรุปผลกลุ่มตัวอย่างการใช้งาน IoT Platform ตลอดระยะเวลา 2 ปี**

1. **AiyaraHarn**:

AiyaraHarn เป็นหนึ่งในกลุ่มผู้ใช้งานที่มีปริมาณการใช้งานสูงสุดของ IoT Platform และถือเป็นรายใหญ่ที่สุดในระบบ โดยมีจำนวนอุปกรณ์เชื่อมต่อมากที่สุดและยังคงมีการขยายตัวอย่างต่อเนื่อง ข้อมูลดังกล่าวสรุปจากฐานผู้ใช้งานจริงในช่วง 2 ปีที่ผ่านมา ซึ่งสะท้อนให้เห็นถึงศักยภาพและโอกาสในการต่อยอดเทคโนโลยี IoT เพื่อนำมาใช้บริหารจัดการและเพิ่มประสิทธิภาพในกับธุรกิจได้อย่างยั่งยืน



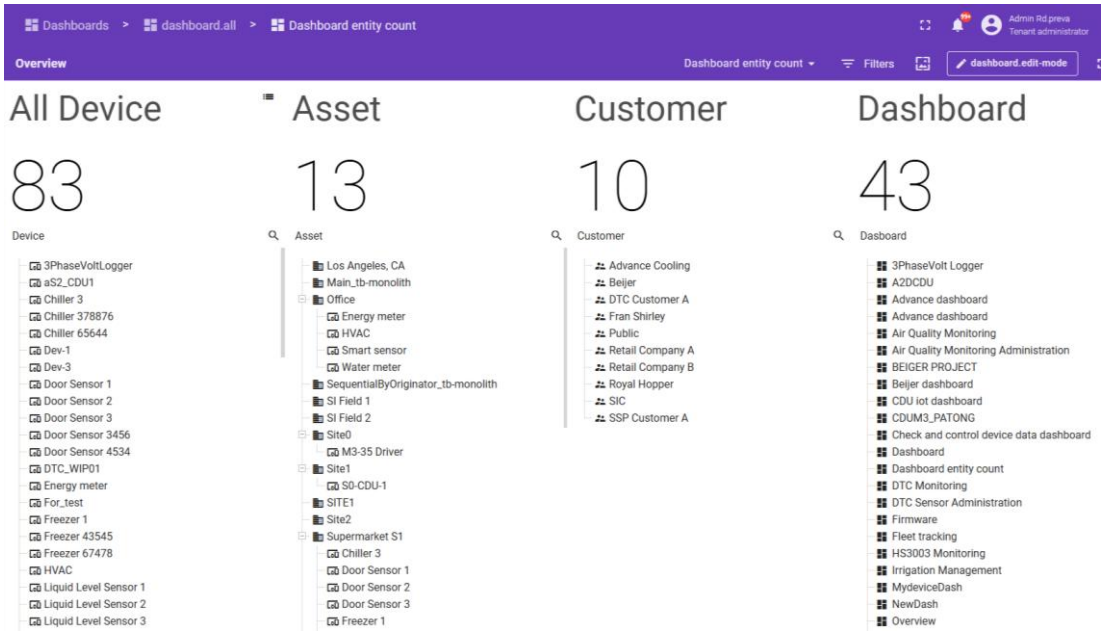
รูปที่ 62 หน้า Dashboard overviews ของ AiyaraHarn



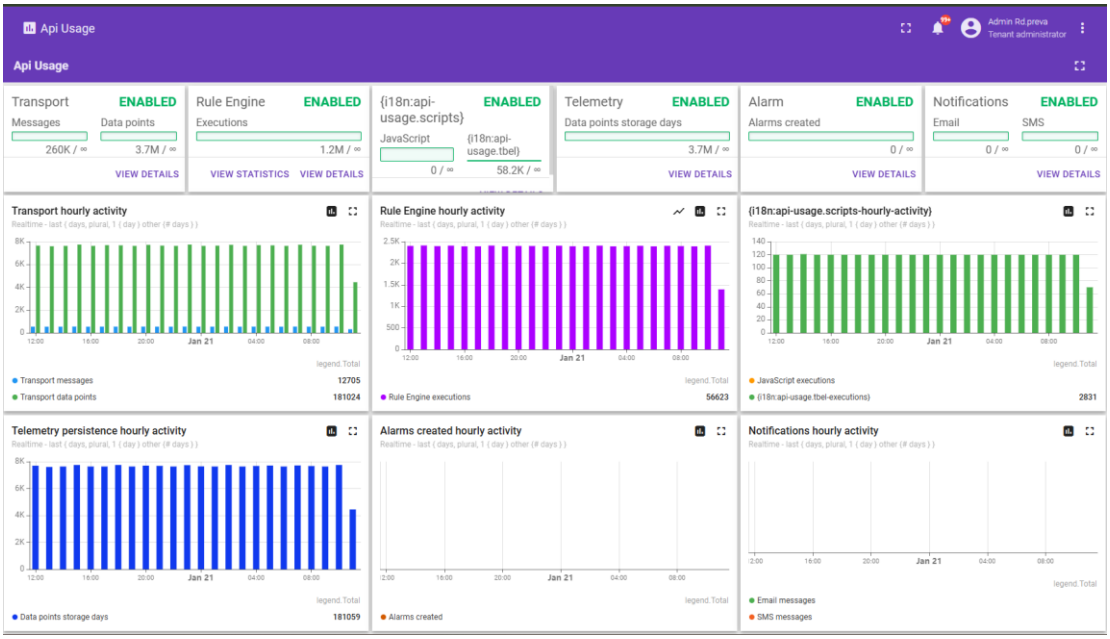
รูปที่ 63 หน้า API Usage ของ AiyaraHarn

2. **บริษัท พรีวาเลนซ์ อุตสาหกรรม จำกัด:**

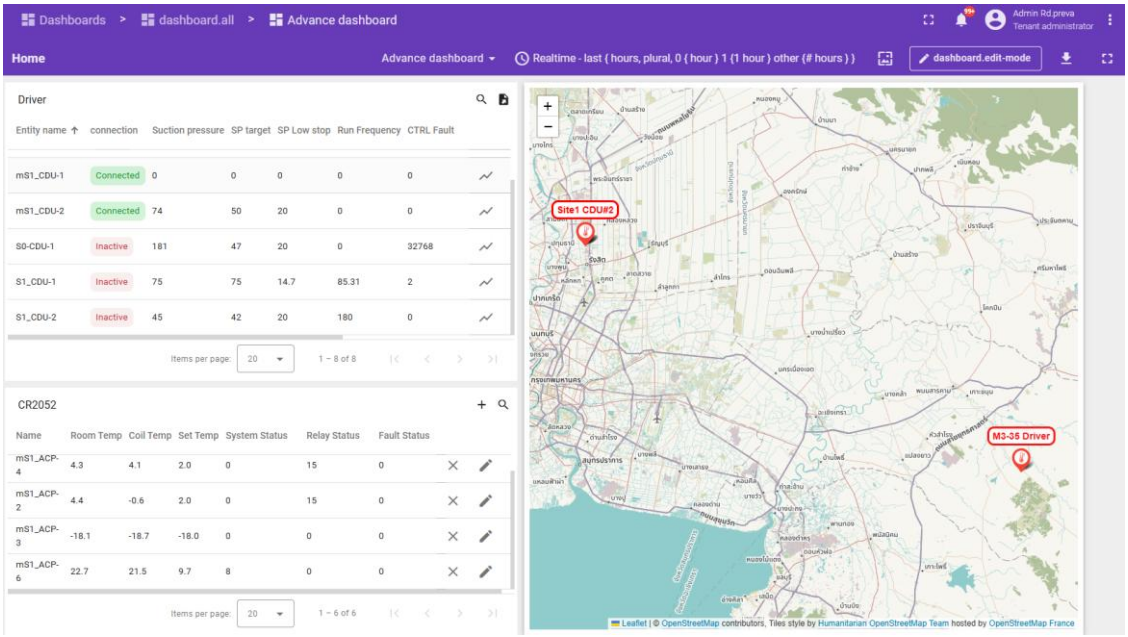
บริษัท พรีวาเลนซ์ อุตสาหกรรม จำกัด ได้นำ IoT Platform มาประยุกต์ใช้กับระบบควบคุมอุณหภูมิ ห้องแช่ ระบบแอร์ และเทอร์โมสตัท (Thermostat) ระยะไกล เพื่อยกระดับกระบวนการจัดการสภาพแวดล้อมภายในองค์กรให้มีประสิทธิภาพมากยิ่งขึ้น โดยการเก็บและวิเคราะห์ข้อมูลแบบเรียลไทม์ส่งผลให้สามารถวางแผนบำรุงรักษาเชิงป้องกันและบริหารพลังงานได้อย่างเหมาะสม อีกทั้งยังต่อยอดแนวคิดสู่ระบบอัตโนมัติในอนาคต เพื่อรองรับการเติบโตและรองรับการขยายขอบเขตการใช้งานด้านอื่น ๆ อย่างเต็มรูปแบบ



รูปที่ 64 หน้า Dashboard overviews ของ พรีวาเลนซ์



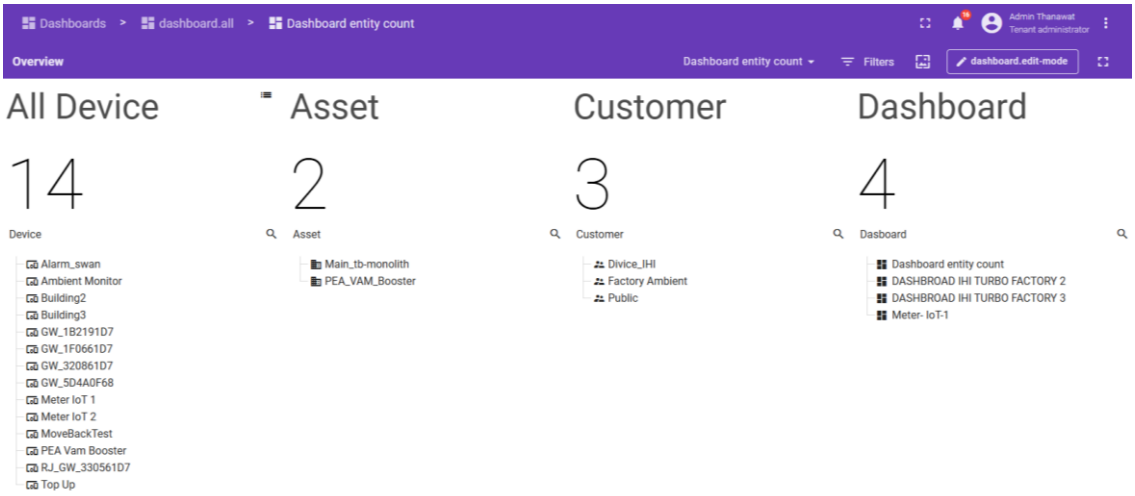
รูปที่ 65 หน้า Dashboard overviews ของ พรีวาเลนซ์



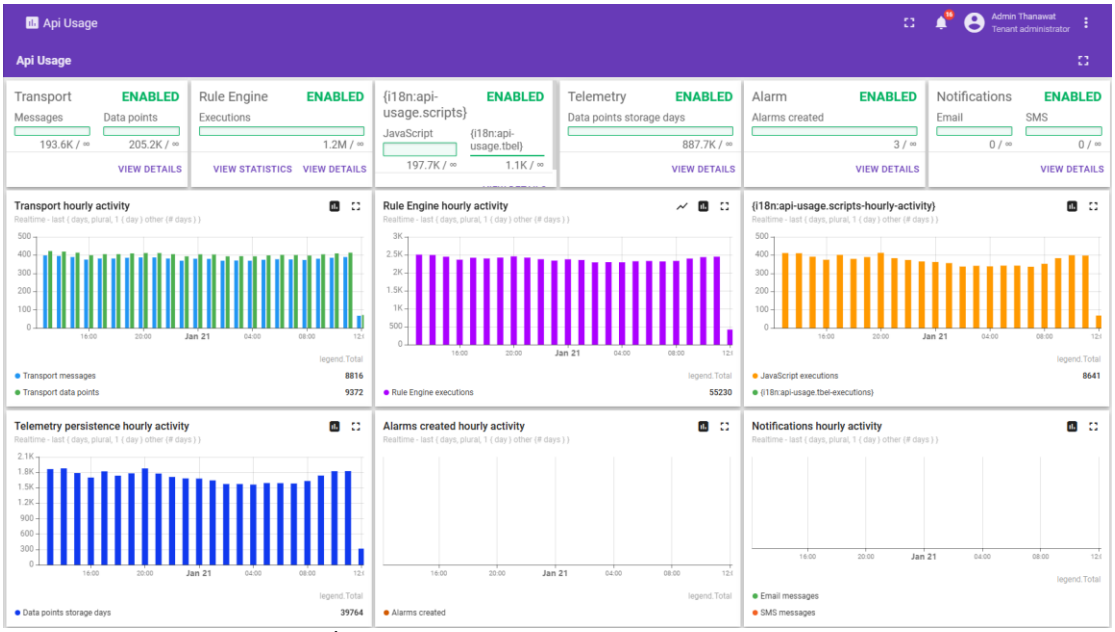
รูปที่ 66 หน้า Dashboard ระบบควบคุมอุปกรณ์เทอร์โมสตัท ของ พรีวาเลนซ์

3. **Lemoco**:

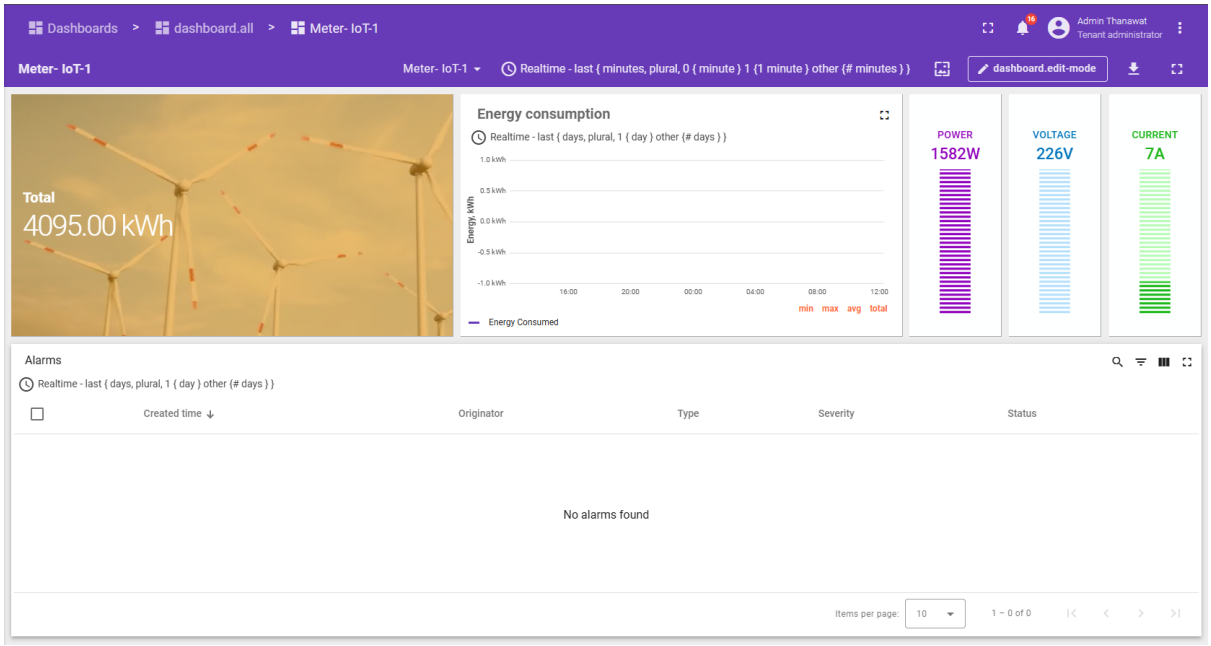
LEMOCO ได้ผสาน IoT Platform เข้ากับ “Smart IoT Meter” เพื่อมอนิเตอร์และควบคุมอุปกรณ์ต่าง ๆ แบบเรียลไทม์ รองรับระบบเติมเงินสำหรับการใช้งานไฟฟ้า ช่วยให้ผู้พักอาศัยในห้องพัก อพาร์ตเมนต์ และคอนโดมิเนียมสามารถคำนวณค่าใช้จ่ายพลังงานได้ทั้งรายวันและรายเดือนอย่างแม่นยำ พร้อมบริหารจัดการต้นทุนและควบคุมการใช้พลังงานได้อย่างมีประสิทธิภาพ



รูปที่ 67 หน้า Dashboard overviews ของ Lemoco



รูปที่ 68 หน้า Dashboard overviews ของ Lemoco



รูปที่ 69 หน้า Dashboard ระบบSmart IoT Meter ของ Lemoco

5.1.2 เชิงพาณิชย์

ThingsBoard คือแพลตฟอร์ม IoT ชั้นนำที่ถูกออกแบบมาเพื่อจัดการข้อมูลจากอุปกรณ์ IoT อย่างมีประสิทธิภาพ โดยมีความสามารถหลากหลาย เช่น Multi-Tenancy การแสดงผลข้อมูลแบบเรียลไทม์ และการปรับแต่งระบบที่ยืดหยุ่น เพื่อรองรับการใช้งานในอุตสาหกรรมและธุรกิจหลากหลายประเภท แพลตฟอร์มนี้เน้นการปรับขนาดและการทำงานที่ตอบโจทย์ความต้องการของผู้ใช้งาน ทั้งในด้านการจัดการข้อมูลที่มีความซับซ้อน และการบริหารจัดการอุปกรณ์ IoT ในปริมาณมากได้อย่างเสถียร อย่างไรก็ตาม การนำแพลตฟอร์มนี้ไปใช้งานในเชิงพาณิชย์ยังคงพบความท้าทายที่สำคัญ เช่น การขาดการตระหนักรู้เกี่ยวกับศักยภาพของเทคโนโลยีนี้ในกลุ่มผู้ใช้งานเป้าหมาย ความซับซ้อนของการใช้งานสำหรับผู้เริ่มต้น และข้อจำกัดในการปรับแต่งระบบที่ต้องอาศัยทักษะเฉพาะทาง ผู้ประกอบการหลายรายยังไม่สามารถใช้ศักยภาพของระบบนี้ได้เต็มที่ เนื่องจากขาดทรัพยากรบุคลากรที่มีความชำนาญเฉพาะด้าน

นอกจากนี้ ความซับซ้อนในส่วนของ Interface และการจัดการ Widget หรือ Rule Engine บนแพลตฟอร์มยังสร้างความยุ่งยากสำหรับผู้ใช้งานเริ่มต้น ทำให้เกิดความล่าช้าในการนำระบบไปใช้ในกระบวนการทำงานจริง แม้ว่าแพลตฟอร์มนี้จะมีการรองรับการปรับแต่งที่ลึกซึ้ง

แต่ยังต้องการเอกสารคู่มือที่ครอบคลุมและการสนับสนุนด้านเทคนิคเพิ่มเติมเพื่อให้เหมาะสมกับผู้ใช้งานใหม่ อีกทั้งยังมีปัญหาด้านการลงทุนเริ่มต้นในระบบ IoT/IIoT ที่ค่อนข้างสูง โดยเฉพาะในธุรกิจขนาดเล็กที่มีข้อจำกัดด้านงบประมาณ

อย่างไรก็ตาม การพัฒนานโยบายสนับสนุนและการให้ความรู้เพิ่มเติมเกี่ยวกับแพลตฟอร์ม ThingsBoard สามารถช่วยแก้ปัญหาเหล่านี้ได้ นอกจากนี้ การปรับปรุงระบบให้มีความง่ายต่อการใช้งานมากยิ่งขึ้น เช่น การออกแบบ Interface ที่ใช้งานง่ายและการพัฒนาโซลูชันแบบสำเร็จรูป ยังเป็นทางออกที่สำคัญเพื่อช่วยขยายการใช้งานแพลตฟอร์มนี้ให้แพร่หลายในกลุ่มธุรกิจและอุตสาหกรรมต่าง ๆ ได้อย่างเต็มศักยภาพ

- **ปัญหาที่พบ**

1. **เป้าหมายผู้ใช้งาน** : มีผู้ใช้งานไม่เป็นไปตามเป้าหมาย 500 ราย
   เนื่องจากการขาดการตระหนักรู้เกี่ยวกับเทคโนโลยี IoT และแพลตฟอร์ม ThingsBoard
   ในกลุ่มเป้าหมายหลัก
2. **ความซับซ้อนของ Interface** : Interface ของ ThingsBoard
   มีความซับซ้อนเกินไปสำหรับผู้ใช้งานทั่วไป
   ทำให้เกิดความยุ่งยากในการเริ่มต้นใช้งานและการจัดการระบบ
3. **ข้อจำกัดด้านการปรับแต่งและทักษะ** : การปรับแต่ง Widget หรือ Rule Engine
   บนแพลตฟอร์มจำเป็นต้องใช้ทักษะเฉพาะทาง ซึ่งบุคลากรหลายคนในองค์กรขาดทักษะดังกล่าว
   นอกจากนี้ยังมีความยุ่งยากในการปรับแต่ง Hardware ให้รองรับการเชื่อมต่อกับ ThingsBoard
   อย่างเหมาะสม
4. **ค่าใช้จ่ายในการลงทุน** : ค่าใช้จ่ายสูงสำหรับอุปกรณ์ IoT/IIoT เป็นอุปสรรคสำคัญ
   โดยเฉพาะสำหรับธุรกิจขนาดเล็กที่ต้องการเริ่มต้นใช้งาน
   นอกจากนี้ยังขาดการสนับสนุนในการลดค่าใช้จ่ายเริ่มต้น
5. **ความต้องการโซลูชันครบวงจรและความปลอดภัย :** ผู้ใช้งานมีความต้องการโซลูชันแบบ Total Solution ที่รวมทั้ง Hardware และ Software ที่ทำงานร่วมกันได้อย่างไร้รอยต่อ
   พร้อมทั้งยังมีความกังวลในด้านความปลอดภัยและความเป็นส่วนตัวของข้อมูล

- **แนวทางแก้ไข**

การส่งเสริมความรู้และการเข้าถึง

1. ออกบูธในงาน FTI 2022 - 2023 เพื่อสร้างความรู้และความตื่นตัวในภาคธุรกิจและอุตสาหกรรม
2. จัดกิจกรรม Workshop เชิงลึกที่เน้นการใช้งาน ThingsBoard และการพัฒนาโซลูชัน IoT/IIoT

การปรับปรุงแพลตฟอร์ม

3. ปรับปรุง Interface ของ ThingsBoard ให้ใช้งานง่ายขึ้น ลดความซับซ้อน
   และเพิ่มฟังก์ชันที่เป็นมิตรต่อผู้ใช้งานทั่วไป
4. เสริมการสนับสนุนด้านเทคนิค เช่น การพัฒนาเอกสารคู่มือการใช้งานและการให้คำปรึกษา

- **ผลลัพธ์ตามเป้าของโครงการ**

1. **ช่วยกระตุ้นและส่งเสริมให้เกิดการใช้งานและพัฒนา IoT และ AI :**
   โครงการนี้ช่วยกระตุ้นและส่งเสริมการใช้งาน IoT และ AI ผ่านกิจกรรมหลากหลาย เช่น
   การจัดอบรมเชิงปฏิบัติการ (Workshops) ที่เน้นการพัฒนาเทคโนโลยี
   และการประยุกต์ใช้ในธุรกิจและอุตสาหกรรม รวมถึงการจัดนิทรรศการในงานFTI 2022 -2023
   เพื่อแสดงศักยภาพของ ThingsBoard นอกจากนี้ ระบบ ThingsBoard ยังรองรับการใช้งานด้าน AI
   อย่างเต็มรูปแบบ เช่น การทำ Machine Learning การวิเคราะห์ข้อมูลเชิงลึก
   และการสร้างระบบอัตโนมัติที่สามารถตอบโจทย์ความต้องการเฉพาะของผู้ใช้งานได้อย่างมีประสิทธิภาพ
2. **การขยายตัวของ IoT สำหรับภาคธุรกิจ และภาคอุตสาหกรรม :**
   ระบบ IoT ช่วยเพิ่มขีดความสามารถในการจัดการ ควบคุมและดูแลอุปกรณ์ IoT
   จำนวนมากในระบบเดียวได้อย่างมีประสิทธิภาพ แบบเรียลไทม์ นอกจากนี้ ระบบ IoT
   ยังเหมาะสมกับการขยายตัวของอุปกรณ์ IoT ในปัจจุบัน
   เนื่องจากสามารถรองรับการเพิ่มจำนวนอุปกรณ์ที่หลากหลายประเภทได้อย่างไร้ข้อจำกัด
   ทั้งยังมีการจัดการข้อมูลแบบกระจายที่ช่วยเพิ่มความเสถียรของระบบและลดความแออัดในเครือข่าย
3. **ระบบ Edge Cloud Computing & IoT Platfortm สามารถรองรับการใช้งานพร้อมกันได้สูงสุด 500 ราย :** ระบบนี้รองรับการใช้งานพร้อมกันได้มากกว่า 500 ราย
   และสามารถขยายความสามารถเพิ่มเติมได้อย่างง่ายดายเพื่อรองรับความต้องการที่เพิ่มขึ้นในอนาคต
   ด้วยโครงสร้าง Edge Computing ที่ช่วยลดเวลาในการประมวลผลและเพิ่มความเร็วในการตอบสนอง
   ระบบยังมีความยืดหยุ่นสูง
   สามารถเพิ่มจำนวนอุปกรณ์ที่เชื่อมต่อและบริหารจัดการได้อย่างมีประสิทธิภาพ

5.1.3 สรุปผลลัพธ์ดัชนีตัวชี้วัดทั้งหมด

ตารางที่ 16 ตารางแสดงข้อมูลตัวชี้วัดผลผลิต

หัวข้อ	รายละเอียด	ผลลัพธ์
Edge Cloud Computing และ IoT Platform ที่ใช้งานง่าย	Edge Cloud Computing และ IoT Platform ที่สามารถติดตั้งและใช้งานได้ง่าย เหมาะสมกับการนำไปใช้งานของภาครัฐ กลุ่มอุตสาหกรรมและประชาชนทั่วไป	ตามเป้าหมาย
Dashboard สำหรับผู้ใช้งานระบบ	มี Dashbarod สำหรับผู้ใช้งานระบบ (User portal) ที่ช่วยบริหารจัดการ พร้อมคู่มือใช้งาน	ตามเป้าหมาย
Dashboard สำหรับการบริหารจัดการต่าง ๆ ในะระบบ	มี Dashbarod สำหรับการบริหารจัดการต่าง ๆ (Admin management service) เพื่อประมวลผล วิเคราะห์แนวโน้ม และคาดการณ์การใช้งานที่จะเกิดขึ้น	ตามเป้าหมาย
API	มีระบบ API ที่สามรถเชื่อมต่อใช้งานกับ Application ต่าง ๆ ของ Users ได้	ตามเป้าหมาย
Multi-Tenancy	IoT Platform ที่รองรับการใช้งานพร้อมกันสูงสุด 500 บริษัท	ตามเป้าหมาย

ตารางที่ 17 ตารางแสดงข้อมูลตัวชี้วัดผลลัพธ์

หัวข้อ	รายละเอียด	ผลลัพธ์
ช่วยกระตุ้นและส่งเสริมให้เกิดการใช้งานและพัฒนา IoT และ AI	โครงการนี้จะช่วยกระตุ้นและส่งเสริมให้เกิดการใช้งานและพัฒนา IoT และ AI ในอุตสาหกรรมไทยอย่างแพร่หลาย	ตามเป้าหมาย
การขยายตัวของ IoT สำหรับภาคธุรกิจและภาคอุตสาหกรรม	โครงการนี้จะทำให้เกิดการขยายตัวของ IoT สำหรับภาคธุรกิจอุตสาหกรรมและโรงงานการผลิต (Industrial Internet of Thing-IIoT) เกิดโรงงานอัจฉริยะ (Smart Factory) และเทคโนโลยีการผลิตแบบ Automation รวมถึงเกิดการตื่นตัวในการใช้งานเทคโนโลยี Edge Computingเพราะสามารถลดระยะเวลาการเข้าถึงและให้การคำนวณใกล้ผู้ใช้งานมากที่สุด	ตามเป้าหมาย
ระบบ Edge Cloud Computing & IoT Platfortm สามารถรองรับการใช้งานพร้อมกันได้สูงสุด 500 ราย	ระบบ Edge Cloud Computing & IoT Platfortm สามารถรองรับการใช้งานพร้อมกันได้สูงสุด 500 ราย โดยจะต้องเป็นหน่วยงานภาครัฐเป็นอย่างน้อย 2 หน่วยงาน	ตามเป้าหมาย

จากการวิเคราะห์โครงการ ThingsBoard พบว่าแพลตฟอร์มนี้มีศักยภาพสูงในการจัดการข้อมูล IoT และ Edge Computing ในธุรกิจและอุตสาหกรรม ด้วยความสามารถรองรับการปรับแต่งและการจัดการอุปกรณ์ในเครือข่ายเดียว ช่วยเพิ่มประสิทธิภาพและลดต้นทุนในกระบวนการทำงานอย่างมีนัยสำคัญ

แม้โครงการจะพบความท้าทาย เช่น ความซับซ้อนของระบบและต้นทุนที่สูงสำหรับธุรกิจขนาดเล็ก แต่แพลตฟอร์มได้รับการปรับปรุงต่อเนื่องเพื่อแก้ปัญหา การสร้างความเข้าใจผ่านกิจกรรมประชาสัมพันธ์ เช่น การออกบูธ และการจัดอบรมเชิงปฏิบัติการ ช่วยเพิ่มจำนวนผู้ใช้งาน

โครงการยังพัฒนาระบบ Edge Cloud Computing ให้รองรับผู้ใช้งานในอนาคต ด้วยโครงสร้างยืดหยุ่น และประยุกต์ใช้งานได้ในอุตสาหกรรม เช่น การจัดการพลังงานภายในและการวิเคราะห์ข้อมูลเชิงลึก ทั้งนี้ คู่มือใช้งานแพลตฟอร์มได้ถูกรวบรวมไว้ในภาคผนวก ก เพื่อความสะดวกในการอ้างอิง

5.2 ข้อเสนอแนะ

การพัฒนาเทคโนโลยี Cloud & Edge Cloud Computing สำหรับอุตสาหกรรมไทยสู่ยุคดิจิทัล ได้ดำเนินงานตามวัตถุประสงค์ที่กำหนดไว้ในโครงการ โดยมีการติดตั้งระบบ Edge Cloud Computing และ IoT Platform ที่สามารถรองรับการใช้งานจากทั้งหน่วยงานภาครัฐและเอกชน อย่างไรก็ตาม เพื่อการพัฒนาที่ต่อเนื่องและสามารถขยายผลในวงกว้างได้ในอนาคต มีข้อเสนอแนะดังต่อไปนี้

1. การสนับสนุนบุคลากร

- **สนับสนุนการอบรมให้กับบุคลากรหรือผู้ที่สนใจ**

  การจัดอบรมถือเป็นกลไกสำคัญในการพัฒนาบุคลากร โดยการอบรมจะครอบคลุมหัวข้อที่เกี่ยวข้องกับการใช้งานเทคโนโลยี Edge cloud computing และ IoT/IIoT ตั้งแต่พื้นฐานจนถึงขั้นสูง ซึ่งรวมถึงการเรียนรู้เกี่ยวกับการประยุกต์ใช้งาน Edge cloud computing ร่วมกับ IoT/IIoT โดยการอบรมยังช่วยสร้างความมั่นใจให้กับผู้เข้าร่วมในด้านการนำใช้งานจริงและการนำไปประยุกต์ใช้ในภาคอุตสาหกรรมตามความเหมาะสม การจัดอบรมในรูปแบบเชิงปฏิบัติการควรจัดให้ครอบคลุมทุกภาคส่วนอุตสาหกรรม และเพิ่มโอกาสให้บุคลากรได้รับการพัฒนาอย่างต่อเนื่องเพื่อสนับสนุนการเปลี่ยนแปลงในยุคดิจิทัลอย่างมีประสิทธิภาพ

2. การกำหนดมาตรฐาน

- **กำหนดให้มีมาตรฐานสำหรับการใช้งาน**

  การมีมาตรฐานการใช้งานถือเป็นปัจจัยสำคัญที่ช่วยให้เกิดความชัดเจนในกระบวนการต่าง ๆ ภายในอุตสาหกรรม รวมถึงการกำหนดกฎเกณฑ์ที่เหมาะสมสำหรับการติดตั้ง การใช้งาน และการบำรุงรักษาเทคโนโลยี Edge cloud computing และ IoT/IIoT ทั้งนี้ การมีมาตรฐานที่ชัดเจนยังช่วยเสริมสร้างความมั่นใจให้กับผู้ใช้งาน ลดข้อผิดพลาดที่อาจเกิดขึ้นจากการใช้งานที่ไม่ถูกต้อง และช่วยให้ผู้ผลิตหรือผู้ให้บริการสามารถพัฒนาระบบให้สอดคล้องกับข้อกำหนดที่เป็นมาตรฐานเดียวกันในอุตสาหกรรมต่าง ๆ การสร้างมาตรฐานควรได้รับการสนับสนุนจากหน่วยงานที่เกี่ยวข้องเพื่อให้ครอบคลุมในทุกมิติของการใช้งาน รวมถึงการปรับปรุงมาตรฐานอย่างต่อเนื่องเพื่อตอบสนองความเปลี่ยนแปลงในอนาคต

3. การพัฒนาต้นแบบและการสนับสนุน

- **การทำต้นแบบที่ใช้งานได้จริง**

การพัฒนาต้นแบบที่สามารถใช้งานได้จริงเป็นกุญแจสำคัญในการส่งเสริมการนำเทคโนโลยี Edge Cloud Computing และ IoT/IIoT ไปใช้งานในภาคส่วนต่าง ๆ เช่น ภาคอุตสาหกรรมการผลิต: โรงงานอัจฉริยะที่ใช้ระบบ IoT เพื่อติดตามและควบคุมการทำงานของเครื่องจักร, การเกษตรอัจฉริยะ: ฟาร์มที่ใช้เซ็นเซอร์เพื่อตรวจวัดความชื้นในดินและควบคุมระบบรดน้ำอัตโนมัติ การขนส่งและโลจิสติกส์: ระบบติดตามยานพาหนะแบบเรียลไทม์, รวมถึงภาคพลังงาน: ระบบจัดการพลังงานในอาคารเพื่อเพิ่มประสิทธิภาพการใช้พลังงาน, ภาคการท่องเที่ยว: การสร้างประสบการณ์การท่องเที่ยวแบบสมาร์ทด้วยระบบแนะนำสถานที่ผ่านแอปพลิเคชัน, และภาคครัวเรือน: บ้านอัจฉริยะที่ควบคุมเครื่องใช้ไฟฟ้าผ่านสมาร์ทโฟน การพัฒนาต้นแบบถือเป็นรากฐานสำคัญที่ช่วยผลักดันประเทศเข้าสู่ยุคดิจิทัลได้อย่างเต็มรูปแบบ

ต้นแบบเหล่านี้ควรได้รับการออกแบบโครงสร้างให้ตอบโจทย์กับความต้องการเฉพาะของแต่ละภาคส่วน ทั้งในด้านการใช้งานและประสิทธิภาพโดยรวม พร้อมทั้งต้องแสดงให้เห็นถึงประโยชน์ได้อย่างชัดเจน นอกจากนี้ การออกแบบต้นแบบควรมีความยืดหยุ่นสูงเพื่อปรับเปลี่ยนฟังก์ชันให้เหมาะสมกับบริบทการใช้งานจริง ต่าง ๆ ที่หลากหลาย การทดสอบต้นแบบในสภาพแวดล้อมจริงยังเป็นขั้นตอนสำคัญในการประเมินความเหมาะสมและความน่าเชื่อถือของระบบ การนำต้นแบบที่ผ่านการปรับปรุงเข้าสู่โครงการนำร่องจะช่วยลดความเสี่ยงในกระบวนการประยุกต์ใช้งานจริง และเพิ่มโอกาสในการสร้างความสำเร็จของเทคโนโลยี IoT/IIoT ในภาคอุตสาหกรรมและส่วนอื่น ๆ อย่างมีประสิทธิภาพ

- **สนับสนุนแหล่งเงินทุน**

การสนับสนุนทางการเงินถือเป็นปัจจัยสำคัญในการกระตุ้นให้ผู้ประกอบการสามารถลงทุนในเทคโนโลยี Edge Cloud Computing และ IoT/IIoT ได้อย่างมั่นใจ โดยควรมีการจัดตั้งโครงการเงินทุนสนับสนุนหรือเงินกู้ดอกเบี้ยต่ำสำหรับผู้ประกอบการที่ต้องการเริ่มต้นหรือขยายการใช้งานเทคโนโลยีในอุตสาหกรรม นอกจากนี้ ควรมีการจัดทำการวิเคราะห์ผลตอบแทนการลงทุน (ROI) เพื่อให้ผู้ประกอบการเข้าใจถึงความคุ้มค่าทางเศรษฐกิจและประโยชน์ที่เทคโนโลยีสามารถสร้างได้ในระยะยาว การร่วมมือกับธนาคารหรือสถาบันการเงินเพื่อพัฒนาผลิตภัณฑ์ทางการเงินที่ตอบโจทย์จะช่วยให้ผู้ประกอบการสามารถลดความเสี่ยงด้านการลงทุนในระยะเริ่มต้นได้อย่างมีประสิทธิภาพ

- **จัดหาฮาร์ดแวร์ที่มีคุณภาพในราคาคุ้มค่า**

การสนับสนุนเรื่องอุปกรณ์ที่มีคุณภาพและราคาที่เหมาะสมถือเป็นปัจจัยสำคัญที่ช่วยลดอุปสรรคในการนำ IoT/IIoT ไปใช้ในอุตสาหกรรมไทยการจัดหาฮาร์ดแวร์ที่ได้รับการรับรองมาตรฐานระบบสากลพร้อมทั้งมีการสนับสนุนด้านการติดตั้งและดูแลบำรุงรักษาจะมีส่วนช่วยเพิ่มความเชื่อมั่นให้กับผู้ใช้งาน อีกทั้งการส่งเสริมการผลิตฮาร์ดแวร์ภายในประเทศเพื่อลดการพึ่งพาการนำเข้าจากต่างประเทศ

5.3 ต้นแบบในการใช้งาน IoT Platform ที่เหมาะสม

ระบบที่พัฒนาขึ้นได้รับการออกแบบมาเพื่อตอบสนองความต้องการเฉพาะตัวอย่างเช่น ในฟาร์มกัญชา โดยใช้เทคโนโลยี Edge Computing และ IoT Platform เพื่อเพิ่มประสิทธิภาพและลดต้นทุนในกระบวนการเพาะปลูก ตัวอย่างของการนำไปใช้งานในระดับธุรกิจฟาร์มกัญชา ได้แก่:

1. **ระบบตรวจวัดสภาพอากาศและดินแบบเรียลไทม์**: การติดตั้งเซ็นเซอร์ในโรงปลูกเพื่อวัดอุณหภูมิ ความชื้น และระดับแสงสว่างที่เหมาะสมสำหรับการเจริญเติบโตของกัญชา ข้อมูลทั้งหมดจะถูกส่งผ่าน IoT Edge ไปยัง Cloud Platform เพื่อการวิเคราะห์เชิงลึกและปรับปรุงสภาพแวดล้อมการปลูกได้อย่างเหมาะสมกับพันธุ์พืช
2. **การจัดการน้ำและปุ๋ยอัจฉริยะ**: ระบบสามารถควบคุมการให้น้ำและสารอาหารที่เหมาะสมกับกัญชาในแต่ละช่วงการเจริญเติบโต โดยวิเคราะห์จากข้อมูลที่เก็บรวบรวมได้แบบเรียลไทม์ ซึ่งช่วยลดการใช้ทรัพยากรเกินความจำเป็นและเพิ่มคุณภาพของผลผลิต
3. **การตรวจสอบสุขภาพของต้นกัญชา**: ใช้กล้องและเซ็นเซอร์ AI ในการตรวจหาปัญหาเกี่ยวกับศัตรูพืช โรค หรือความผิดปกติในต้นกัญชาได้ตั้งแต่ระยะแรก พร้อมทั้งส่งข้อมูลแจ้งเตือนให้ผู้จัดการฟาร์มดำเนินการแก้ไขทันที นอกจากนี้ยังสามารถใช้โดรนในการตรวจสอบภาพรวมของฟาร์มขนาดใหญ่ได้
4. **การวางแผนการปลูกและเก็บเกี่ยว**: ข้อมูลที่สะสมจากการเพาะปลูกหลายรอบจะถูกนำมาวิเคราะห์เพื่อวางแผนการปลูกในอนาคต เช่น การกำหนดช่วงเวลาที่เหมาะสมที่สุดในการเพาะปลูกและการเก็บเกี่ยว รวมถึงการแนะนำสายพันธุ์กัญชาที่เหมาะสมกับสภาพแวดล้อมของฟาร์ม
5. **การติดตามคุณภาพผลผลิต**: ระบบสามารถติดตามและรายงานข้อมูลคุณภาพของผลผลิต เช่น ความเข้มข้นของสาร THC และ CBD เพื่อให้แน่ใจว่าได้มาตรฐานสำหรับการจำหน่ายทั้งในและต่างประเทศ

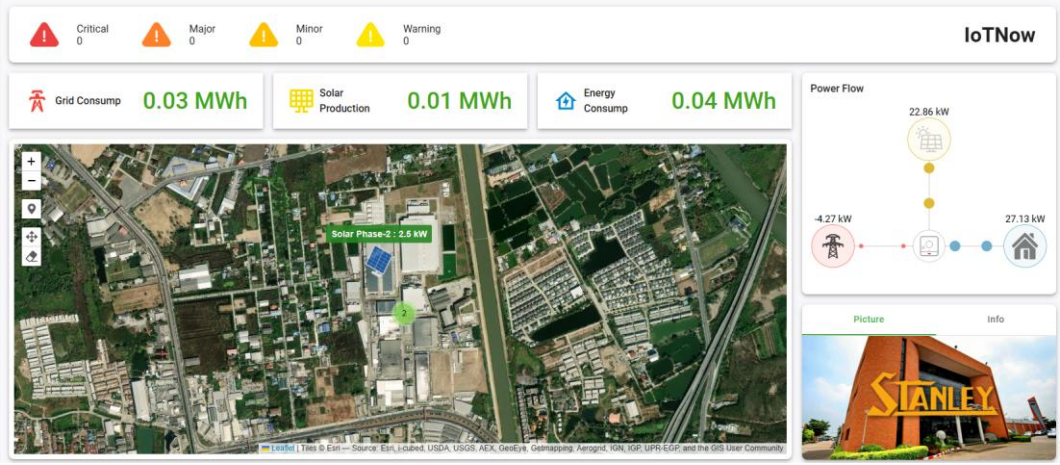
การนำระบบนี้มาใช้ในฟาร์มกัญชาไม่เพียงช่วยเพิ่มผลผลิต แต่ยังช่วยลดต้นทุนและทรัพยากรที่ต้องใช้ในกระบวนการเพาะปลูก ทำให้ฟาร์มสามารถบริหารจัดการได้อย่างยั่งยืน พร้อมทั้งเพิ่มความเชื่อมั่นในคุณภาพของผลผลิตที่สอดคล้องกับมาตรฐานอุตสาหกรรม นอกจากนี้ การเก็บข้อมูลคุณภาพ เช่น ความเข้มข้นของสาร THC และ CBD ช่วยให้ผู้ประกอบการสามารถพัฒนาผลิตภัณฑ์ที่มีมูลค่าเพิ่ม เช่น น้ำมันสกัดกัญชา ผลิตภัณฑ์เพื่อสุขภาพ หรือวัตถุดิบสำหรับอุตสาหกรรมยา ซึ่งช่วยขยายโอกาสทางการตลาดและเพิ่มรายได้ให้กับฟาร์มได้ในระยะยาว

Aiyara Harn [14]

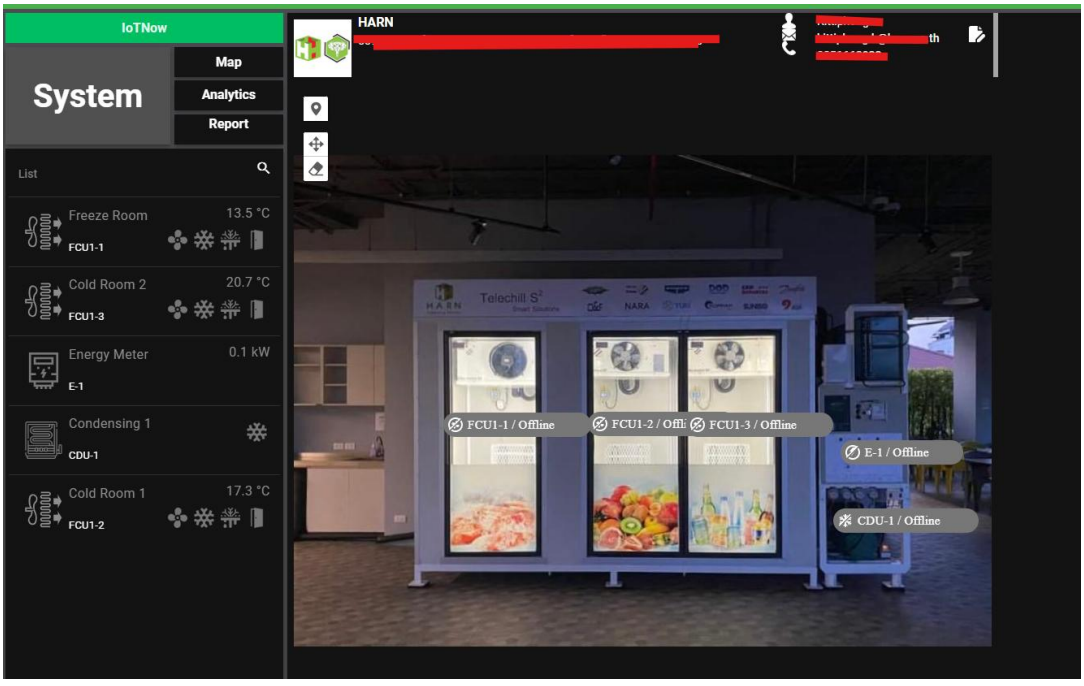
บริษัท ไอยราหาญ จำกัด ซึ่งเป็นบริษัทในเครือของ บริษัท หาญ เอ็นจิเนริ่ง โซลูชั่น จำกัด (มหาชน) (Harn Engineering Solutions Pcl.) ถูกก่อตั้งขึ้นโดยมีวัตถุประสงค์หลักในการพัฒนาผลิตภัณฑ์และโซลูชันที่เกี่ยวข้องกับเทคโนโลยี IoT และ BMS โดยแพลตฟอร์ม IoTNow! (พัฒนาบนพื้นฐานของ Thingsboard) ได้รับการนำเสนอเพื่อใช้ในงานด้านการจัดการอาคาร ระบบห้องทำความเย็น อุปกรณ์ตรวจสอบเครื่องจักรและระบบใช้การกำคำนวณแนวทางการบำรุงรักษาเชิงคาดการณ์ ระบบการจัดการพลังงาน และระบบบริหารจัดการอาคาร (BMS) ทั้งนี้ บริษัทฯ ยังได้นำแพลตฟอร์มดังกล่าวมาทดลองใช้งานในหลากหลายภาคส่วน เพื่อปรับปรุงและพัฒนาความสามารถของระบบให้สอดคล้องกับความต้องการของตลาดอย่างแท้จริง

นอกจากนี้ บริษัท ไอยราหาญ จำกัด ยังตั้งเป้าหมายในการเป็นผู้นำด้านนวัตกรรมแพลตฟอร์มเทคโนโลยี IoT โดยให้ความสำคัญกับการสร้างสภาพแวดล้อมที่ยั่งยืน ความปลอดภัย และการเชื่อมต่อผ่านเทคโนโลยี IoT ที่ล้ำสมัย

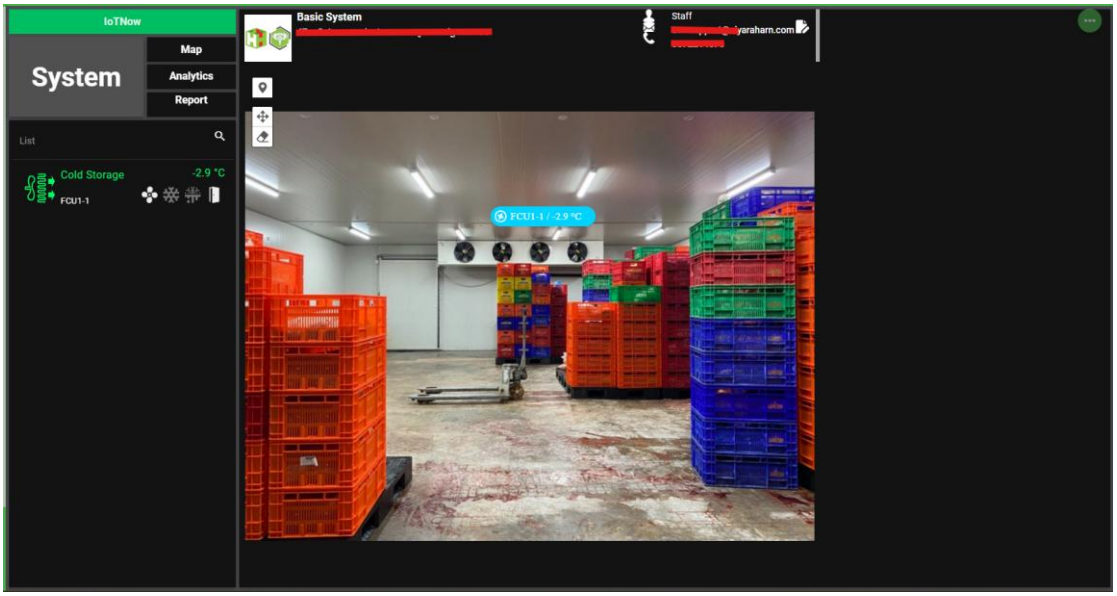
บริษัท ไอยราหาญ จำกัด ได้แสดงศักยภาพอย่างโดดเด่นในการนำเทคโนโลยี Edge Cloud Computing ไปใช้ในกระบวนการธุรกิจได้อย่างมีประสิทธิภาพ โดยสามารถยกระดับการจัดการข้อมูล ลดต้นทุนการดำเนินงาน และเพิ่มความรวดเร็วและแม่นยำในการบริหารจัดการระบบ IoT อย่างมีประสิทธิผล การใช้งานเทคโนโลยีนี้ยังช่วยสนับสนุนการพัฒนาโซลูชันที่ตอบโจทย์ความต้องการเฉพาะของลูกค้า เช่น ระบบ IoT สำหรับควบคุมพลังงานอัจฉริยะ การบริหารจัดการอาคารที่ปรับเปลี่ยนได้ตามลักษณะโครงการ และระบบจัดการ/ควบคุมห้องแช่เย็นที่ทันสมัย เพื่อลดการสูญเสียพลังงานและเพิ่มอายุการใช้งานของอุปกรณ์ในระบบได้อย่างมีประสิทธิภาพสูงสุด ทั้งนี้ ความสำเร็จของบริษัทในการขยายฐานลูกค้าและสร้างความเชื่อมั่นในตลาดสะท้อนถึงการประยุกต์ใช้เทคโนโลยีทันสมัย และการมีบุคลากรที่มุ่งมั่นในการพัฒนานวัตกรรมได้อย่างต่อเนื่อง เพื่อตอบสนองต่อความต้องการในภาคอุตสาหกรรม 4.0 ของประเทศได้อย่างมีประสิทธิภาพ



รูปที่ 70 ต้นแบบระบบจัดการพลังงานของ Aiyara Harn



รูปที่ 71 ต้นแบบระบบจัดการห้องแช่ของ Aiyara Harn (1)



รูปที่ 72 ต้นแบบระบบจัดการห้องแช่ของ Aiyara Harn (2)

บรรณานุกรม

[1] https://docs.hpc.cam.ac.uk/cloud/userguide/03-nova.html

[2] https://docs.hpc.cam.ac.uk/cloud/background/01-general_concepts.html

[3] https://docs.hpc.cam.ac.uk/cloud/userguide/06-cinder_swift.html

[4] https://docs.openstack.org/glance/latest/

[5] https://openstack.in.th/core-components/keystone-identity-service/

[6] https://openstack.in.th/core-components/horizon-dashboard/

[7] https://opensdn.io/

[8] https://www.etsi.org/?option=com_easyblog&lang=&Itemid=573

[9] https://ieeexplore.ieee.org/document/9445331

[10] https://ieeexplore.ieee.org/document/9114929

[11] https://thingsboard.io/products/trendz/

[12] https://thingsboard.io/products/thingsboard-edge/

[13] https://thingsboard.io/docs/iot-gateway/what-is-iot-gateway/

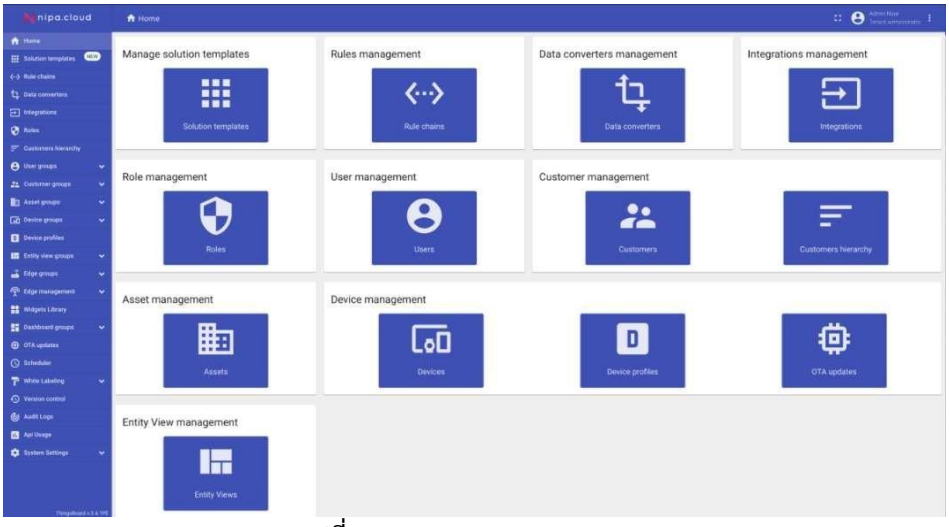
[14] https://www.harn.co.th/

ภาคผนวก ก

[คู่มือการใช้งาน Thingsboard]

เรียนรู้ใช้งาน IoT Platform (Thingsboard)

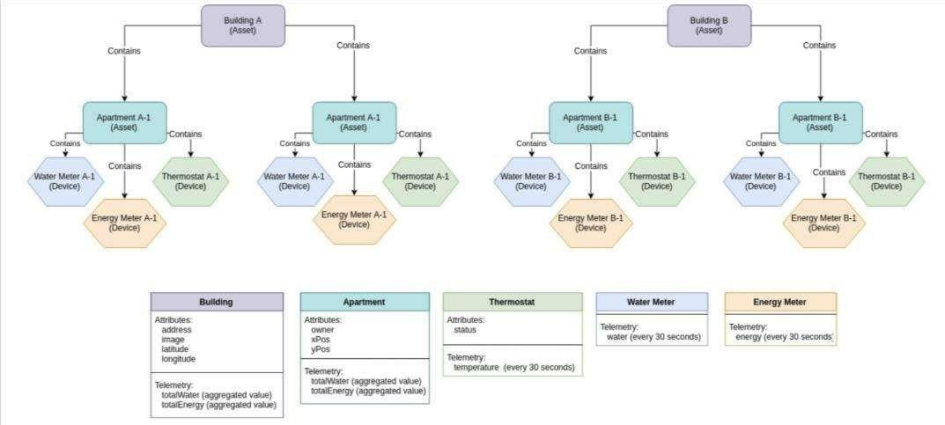
หลังจาก Login เข้ามาใช้งาน Thingsboard IoT Platform เป็นเรียบร้อยแล้วโดยที่ทางด้านขวามือบนจะพบว่าผู้ใช้งานจะได้รับสิทธิใช้งานเป็น Tenant Administrator ซึ่งจะสามารถจัดการ Entity ต่างๆ ได้อย่างอิสระ โดยแยกจากผู้ใช้งานที่เป็นสิทธิ Tenant Administrator ด้วยกัน และผู้ใช้จะพบ Menu Feature ต่างๆที่ทางซ้ายมือ



รูปที่ 73 Overview IoT Platform

IoT Platform ไม่ได้เป็นแค่ Dashboard ในการแสดงผลของข้อมูลจากอุปกรณ์ IoT ที่รับมาเท่านั้นโดยสามารถที่จะสร้างความสัมพันธ์ของอุปกรณ์ต่างๆ ตามลำดับ เพื่อง่ายต่อการจัดการและให้ตรงกับความต้องการในการใช้งาน

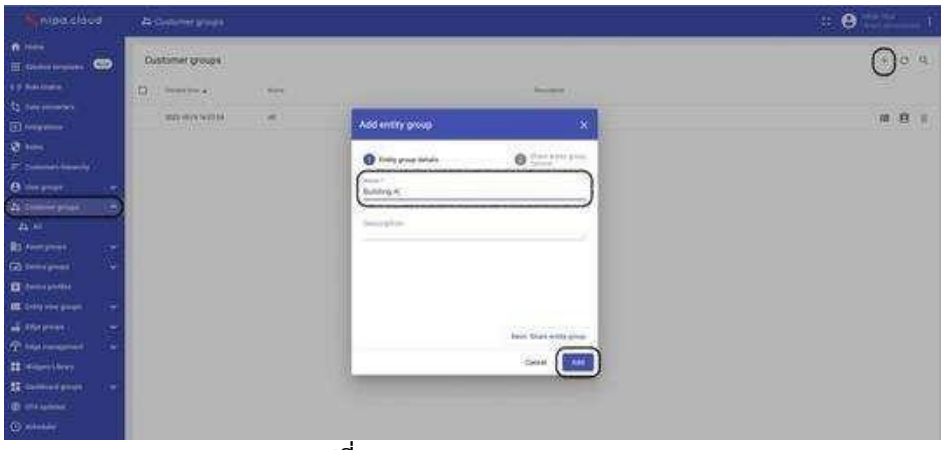
Section 1.01 ลำดับความสัมพันธ์ของ Entity ในระบบที่อยู่ภายใต้ Tenant Administrator เป็นลำดับขั้น



รูปที่ 74 ความสัมพันธ์ของ Entity บน IoT Platform

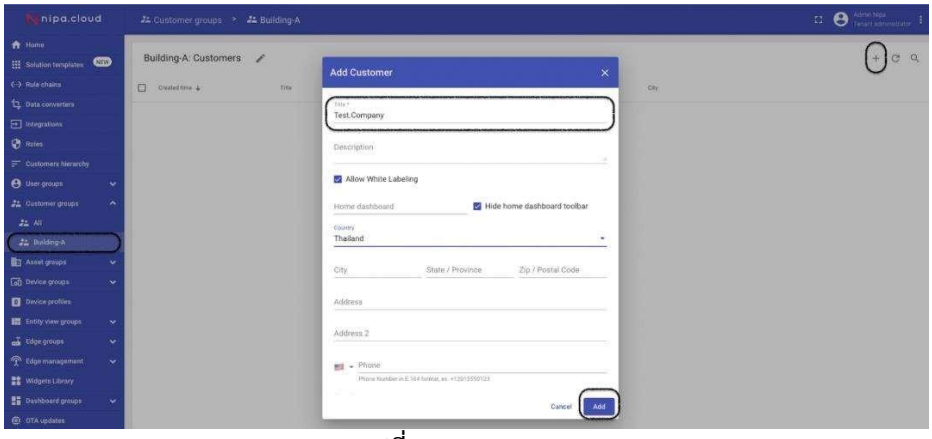
โดยความสัมพันธ์ระหว่าง Entity กับ Entity บน IoT Platform ภายใต้เจ้าของเดียวกันนั้น จะมีความสัมพันธ์ต่างๆ เช่น Contains, Manages โดยความสัมพันธ์หรือ Relation นี้จะมีทิศทางที่ชัดเจน

Section 1.02 การจัดการ Customer



รูปที่ 75 Create Customer Groups

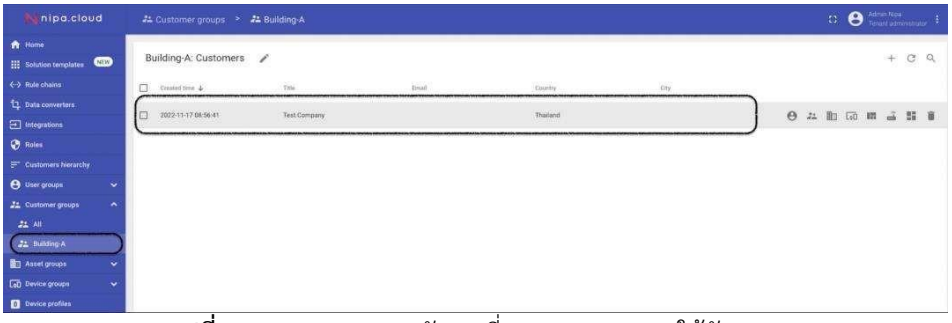
บน IoT Platform นั้น ผู้ใช้สามารถจัดการและกำหนดสิทธิการเข้าถึง Entity ต่างๆ ให้กับ Customer ได้ ซึ่ง Customer ไม่ได้หมายถึง Customer เดียวเท่านั้น ภายใน Customer สามารถที่จะเพิ่ม Customer เข้าไปได้ในภายหลังโดยการจะเพิ่ม Customer ผู้ใช้สามารถใช้งานผ่าน Customer Groups ที่ Menu ทางด้านซ้าย แสดงรายการ Customers แล้วผู้ใช้สามารถทำการเพิ่ม Customers โดยผู้ใช้ต้อง create group ให้กับ Customers นั้น โดย click ที่เครื่องหมาย “+”



รูปที่ 76 Add Customer

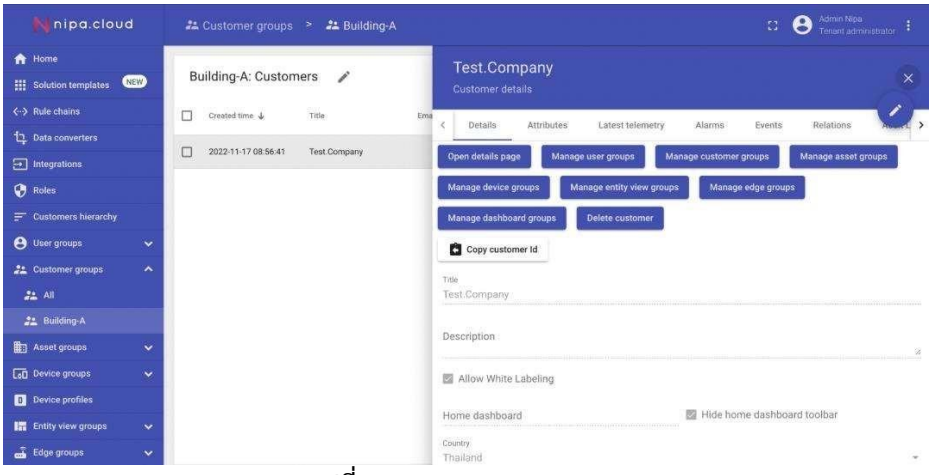
หลังจากที่ Click ที่เครื่องหมาย “+” แล้วจากนั้นผู้ใช้สามารถที่จะใส่ชื่อให้กับ Customer Groups แล้วจากนั้น Click ที่ Add ตามลำดับจากนั้นจะเป็นการเสร็จสิ้นในการ Create Customer Groups หลังจากนั้นผู้ใช้สามารถที่จะ Create Customer

ในรูปจะเป็นการ Add Customer ให้กับ Tenant นั้นโดยจะเห็นได้ว่าเมื่อ Create Customer Groups แล้วที่ Menu ซ้ายมือที่เป็น Menu หลักจะมี Group ของ Customer ที่ผู้ใช้พึ่ง Create หลังจากนั้นให้ผู้ใช้ Click “+”แล้วจากนั้นผู้ใช้สามารถที่จะใส่ผู้ใช้รายละเอียดเพื่อที่จะ Add Customer นั้นได้หลังจากนั้น Click ที่ “Add”



รูปที่ 77 Show Detail หลังจากที่ Add Customer ให้กับ Tenant

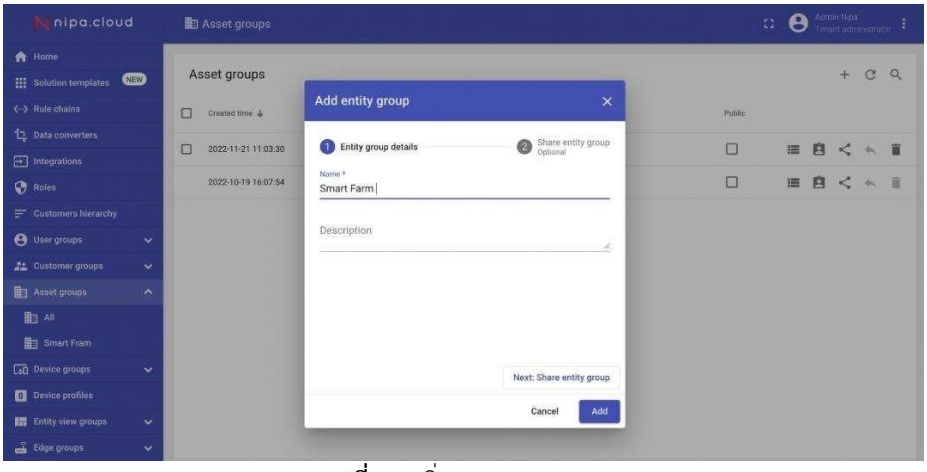
Click ที่แท็ป Customer ที่ผู้ใช้พึ่ง Create ไปจะมีแท็ป Menu show ขึ้นมา โดยที่ Menu ที่ show สามารถที่จะจัดการได้ตาม Menu ให้กับ User หรือ Customer ต่อไปได้โดยที่จะไม่มีความสัมพันธ์กับ Customer Groups อื่นๆ



รูปที่ 78 Menu Customer Tenant

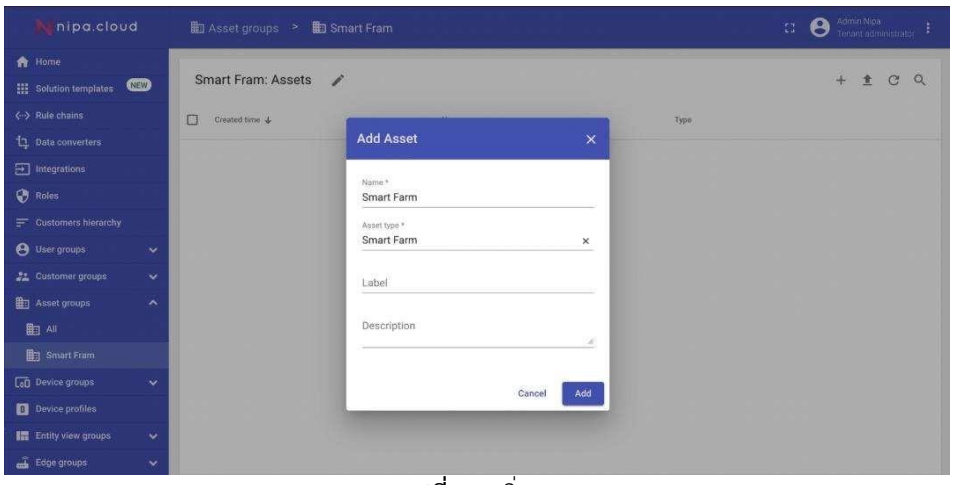
Section 1.03 Asset และ การจัดการ Asset

บน ThingsBoard จะมอง entity เป็นลำดับชั้นของ asset เช่น ระบบ farm ผู้ใช้สามารถมี farm ได้หลายแหล่งโดยที่ในแต่ละ farm สามารถมีหลายแปลงปลูกได้ ในแต่ละแปลงปลูก สามารถมีอุปกรณ์เซ็นเซอร์ต่างๆ ได้ ด้วยแนวคิดนี้ asset จะสามารถประกอบด้วย asset ย่อยหรือ device ด้วยก็ได้ ตามความสัมพันธ์ (relation) แบบ contain



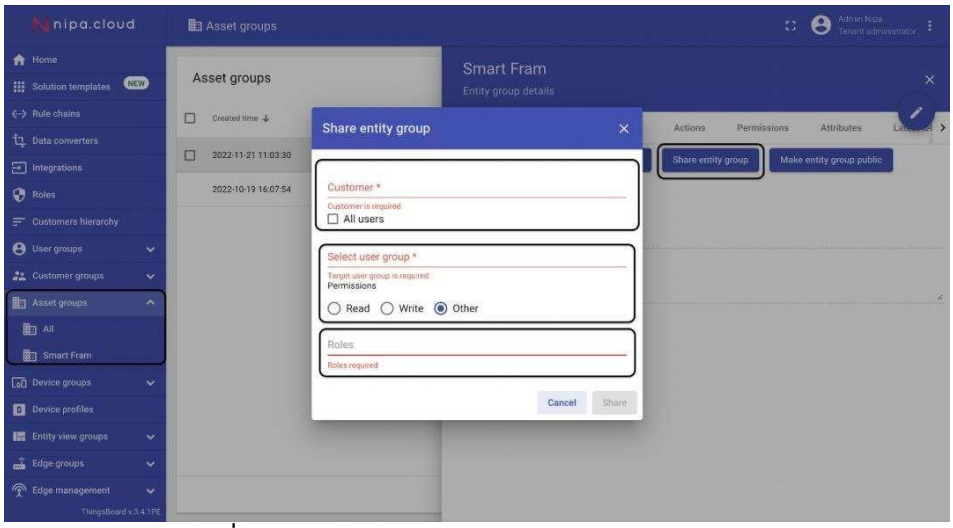
รูปที่ 79 เพิ่ม Asset Group

ในขั้นตอนแรกะมาทดลองสร้าง asset หลัก 1 asset โดยจะตั้งชื่อว่า Smart Farm ให้เป็น asset ที่เป็นตัวแทนของระบบ Smart Farm เพื่อใช้ในการทดลองต่อไปโดยเข้าไปที่ Menu Asset Group จากนั้น click ที่ “+” จากมี Menu ให้เพิ่ม Asset Group โดยใส่ชื่อเพื่อที่จะสร้าง Asset ใน group



รูปที่ 80 เพิ่ม Asset

Menu Asset Group จากนั้น click ที่ “+” จากมี Menu ให้เพิ่ม Asset โดยใส่ชื่อของ Asset และ Asset type โดยที่ Asset type มีส่วนสำคัญที่จำเป็นต่อการนำไปใช้งานด้านอื่นๆ และสามารถที่จะไปแสดงหน้า dashboard ได้เมื่อเพิ่ม asset เรียบร้อยแล้วกลับไปยัง หน้า Assets จะพบรายการ asset จะพบรายการ asset ที่เพิ่มเข้าไปใหม่โดยค่าที่ใช้อ้างอิงถึง asset คือ asset id

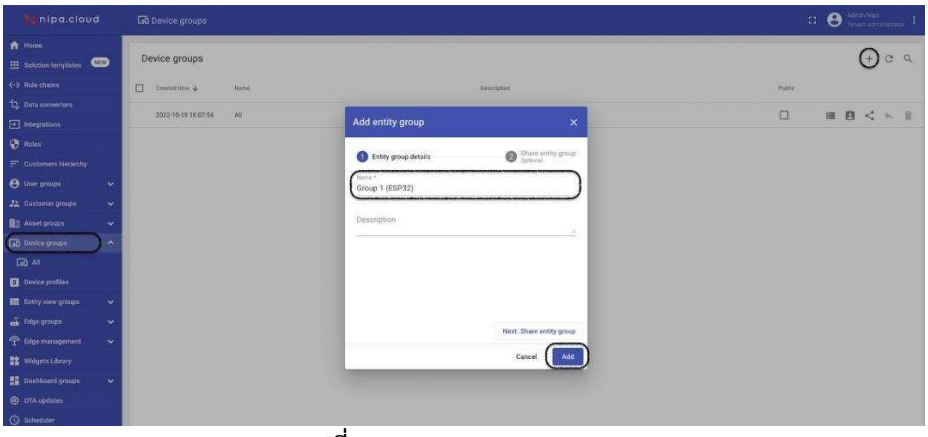


รูปที่ 81 กำหนด Asset ให้กับ User group, Customer group

โดย asset ที่เพิ่มเข้าไปยัง Thingsboard นั้นจำเป็นต้องกำหนดความเป็นเจ้าของให้กับ User หรือ Customer group กับ asset และ role โดยสามารถทำได้ โดย Click ที่ share entity group

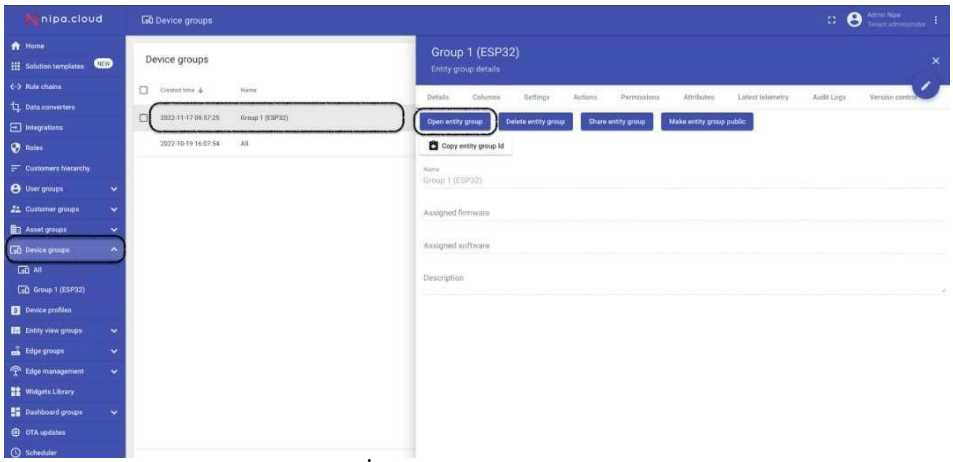
Section 1.04 การเพิ่มอุปกรณ์ใหม่บน Thingsbo

โดยเข้าไปที่ Menu “Device Group” ทางแท็ปซ้ายมือ เริ่มจากการที่ต้องสร้าง Group ให้กับ Device นั้นสามารถ click ที่ “+” จากนั้นใส่รายละเอียดของ Device Group หลังจากเสร็จแล้ว click ที่ “Add”

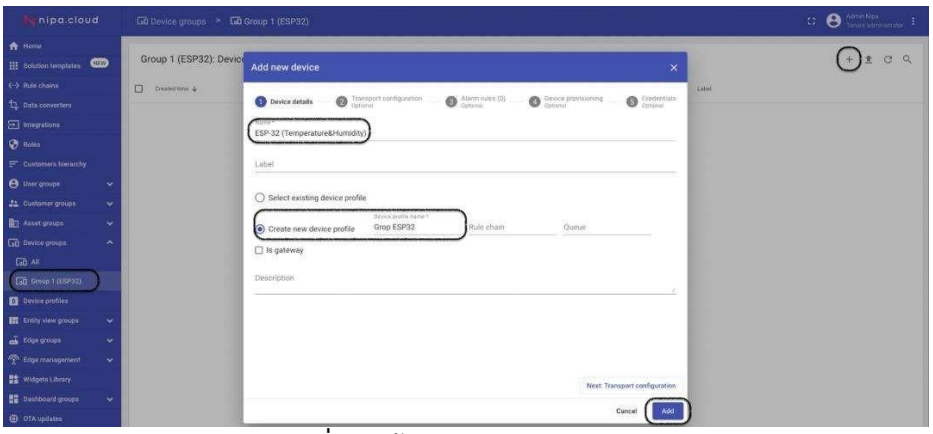


รูปที่ 82 Create Device Group

หลังจากที่สร้าง Device Group ขึ้นมาเรียบร้อยแล้วจากนั้นให้ click บน Device Group ที่สร้างขึ้นมาจะมี Menu ขึ้นมาที่แท็ปซ้ายมือจากนั้นสามารถที่เข้าไป Group ที่สร้างได้โดย click ที่ “Open Entity Group” เพื่อที่จะสร้าง Device บน IoT Platform

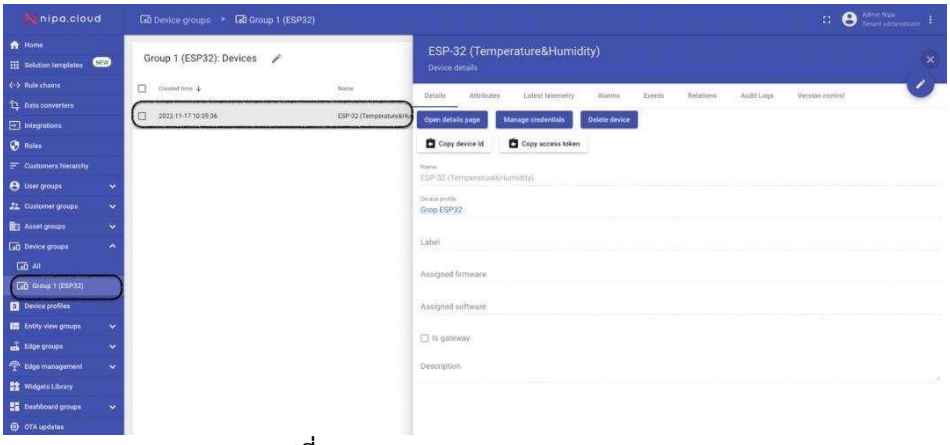


รูปที่ 83 Open Entity Group



รูปที่ 84 สร้าง Detail IoT Platform

หลังจากที่ Open Entity Group เข้ามาแล้ว จะเป็นการสร้าง Device IoT Platform เพื่อเชื่อมต่อกับ Device หรือ sersor ต่างๆของผู้ใช้งานเอง โดยที่เริ่มจาก click ที่ “+” ที่มุมขวาบนจากนั้นที่จะใส่รายละเอียดต่างๆในส่วนของ Device และแนะนำให้สร้าง Device Profile เพื่อที่จะในการทำ Rule Chaine นั้นจะอ้างอิงกับ Device Profile เพื่อที่จะไม่ทำให้ Device Profile ที่ไม่เกี่ยวข้องได้รับผลกระทบกับ Rule Chaine และ Device Profile นั้น สามารถดูได้ที่ เอกสาร Rule Chaine หลังจากที่ Create Detail Device เรียบร้อยแล้ว click ที่ “Add”



รูปที่ 85 Menu Detail Device IoT Platform

หลังจากที่ Create Detail Device บน IoT Platform จากนั้น click 1 ครั้ง จะมี Menu ที่สามารถจะจัดการ Device นั้นได้ตาม

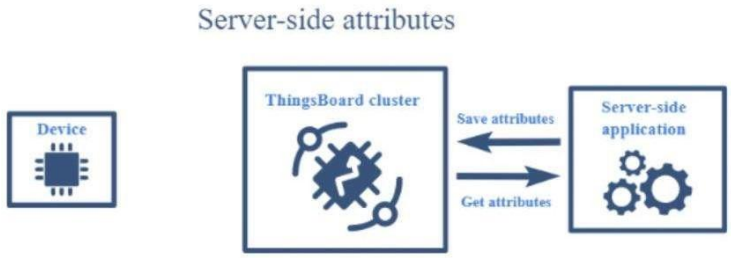
Section 1.05 ค่าคุณลักษณะของ Entity บน ThingsBoard

บน ThingsBoard นั้น entity เป็นได้ทุกอย่างไม่ว่าจะเป็น user, asset หรือ device ล้วนแล้วแต่มี attribute โดย attribute แบ่งออกได้ 3 ประเภทคือ

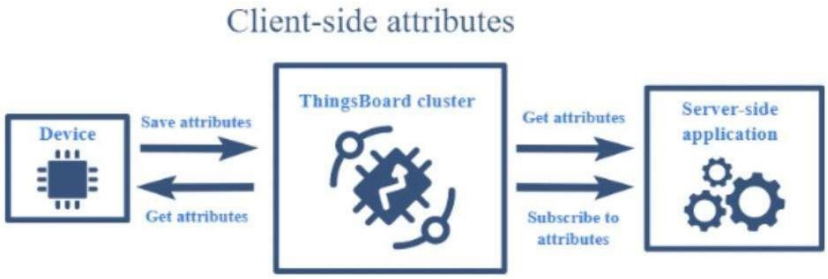
Server attribute เป็น attribute ที่ใช้งานบน servers เท่านั้น device ไม่สามารถเข้าถึงค่า attribute

Client attribute เป็น attribute ของ device เช่น ความชื้นในอากาศ อุณหภูมิ บน ThingsBoard

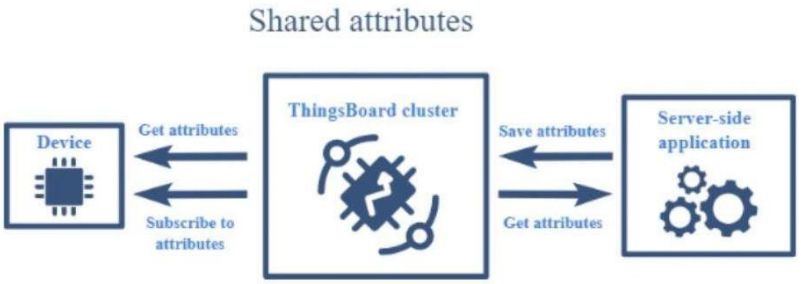
Shared attribute เป็น ค่า attribute ที่สร้างหรือกำหนดได้บน ThingsBoard ซึ่ง device สามารถดูค่าหล่านั้นได้ แต่ไม่สารมารถแก้ไขได้ เช่น ค่า firmware version เพื่อจะรู้ว่า device ต้องการ update แบบ OTA



รูปที่ 86 Server Attribute

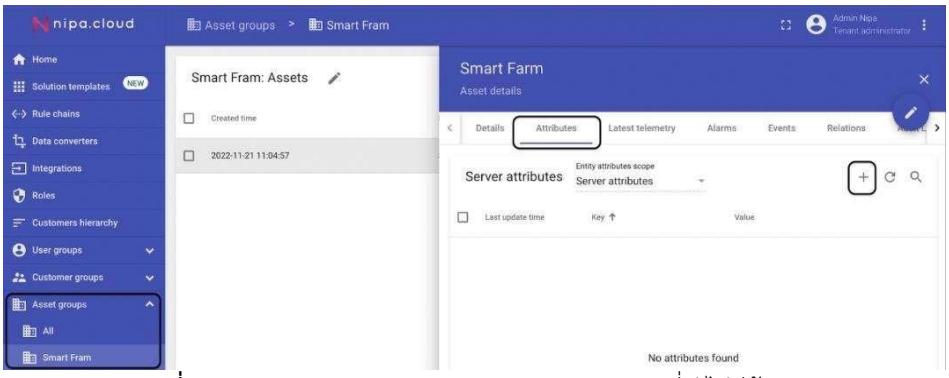


รูปที่ 87 Client Attribute



รูปที่ 88 Shared attribute

การเพิ่มค่า Attribute



รูปที่ 89 Server Attribute ของ Asset Smart Farm ที่ยังไม่มีข้อมูล

การจัดการ attribute ของ entity ในทุกๆ entity จะอยู่ในส่วน attributes โดยจะสามารถเลือก entity attribute scope ได้ถ้าหากสามารถเลือก entity เช่น user, asset จะมีเพียงแค่ค่าของ server attribute ให้เลือกเท่านั้น โดยหากต้องการเพิ่ม attribute สามารถทำได้โดย click ที่ “+” เพื่อที่จะทำการเพิ่ม attribute

โดยที่การ add attribute ที่ปรากฏขึ้นนั้นจะให้กำหนดค่า attribute ได้โดยค่า attribute จะมีลักษณะของข้อมูลเป็นในรูปแบบ Key-value คือค่า key จะถูกใช้อ้างอิงเพื่อที่จะอ่านค่า valued กลับมาโดยที่ค่า value สามารถเลือกประเภทของข้อมูลได้ 5 ประเภทคือ

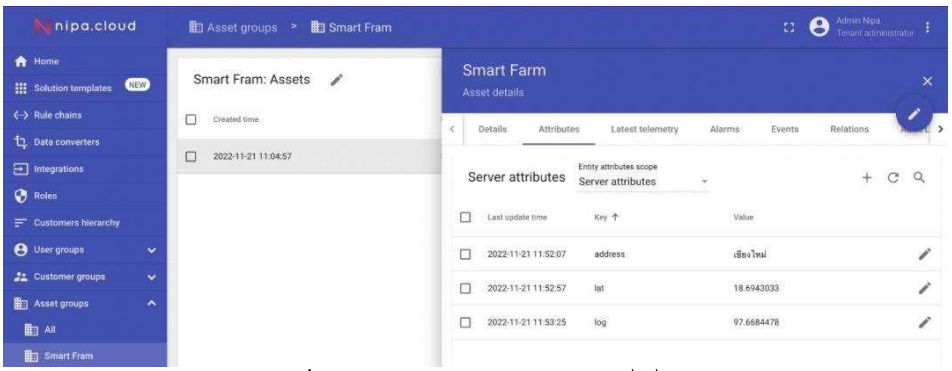
String ข้อมูลตัวอักษร

Integer ข้อมูลตัวเลขมีทศนิยม

Boolean ข้อมูลตรรกะจริง/เท็จ

JSON ข้อมูลในรูปแบบ JSON

เช่น ต้องการที่จะใส่พิกัดของ asset ของผู้ใช้คือ “Smart Farm” ได้โดยที่เพิ่มค่า lat,log ที่ Servers Attribute



รูปที่ 90 ข้อมูล Servers Attribute ที่เพิ่มลงไป

Section 1.06 การเชื่อมและส่งข้อมูลไปที่ IoT Platform

การส่งข้อมูลจาก Device ไปยัง IoT Platform หรือ ThingsBoard สามารถทำได้หลายวิธีโดยรองรับ Protocol ดังนี้

MQTT, CoAP, HTTP, LwM2, SNMP

ส่วนใหญ่การใช้งานที่ใช้กันมากที่สุดจะเป็น HTTP และ MQTT

โดยประเภทข้อมูลของ Device ที่ส่งไปยัง ThingsBoard สามารถที่จะส่งข้อมูลได้ 2 ประเภท คือ

Attribute และ Telemetry ในส่วนของ Telemetry จะส่งข้อมูลแบบเป็น time series ไปที่ ThingsBoard หรือ IoT Platform

HTTP

HTTP สามารถส่งข้อมูลที่เป็น Attribute และ Telemetry ไปยัง ThingsBoard หรือ IoT Platform ได้ โดยใช้ Method Post ที่จะสามารถส่งข้อมูลไปที่ IoT Platform และใช้ access token ที่ได้หลังจากที่ Cerate อุปกรณ์แล้วใช้ในการยืนยัน โดย access token ใช้ในการที่จะอ้างอิง Device ต่างๆ ของ ThingsBoard และสามารถที่จะดูได้จาก Device Profile บน Menu แท็ปซ้ายมือบน IoT Platform โดยที่การส่งข้อมูลไปจาก Device ไปที่ IoT Platform โดยใช้ HTTP ในการส่งถ้าไม่มีอะไรผิดพลาด จะขึ้น Status 200 นอกเหนือจากนั้น คือ

400 : Bad Request : ส่งข้อมูลขอ Request ไม่ถูกต้อง

401 : Bad Unauthorized : $ACCESS_TOKEN ไม่ถูกต้อง

402 : Not Found : ไม่มี URL นี้

รูปแบบของข้อมูลที่ส่ง

```
{
 "stringKey":"value1",
 "booleanKey":true,
 "doubleKey":42.0,
 "longKey":73,
 "jsonKey": {
    "someNumber": 42,
    "someArray": [1,2,3],
    "someNestedObject": {"key": "value"}
 }
}
```

รูปที่ 91 รูปแบบ Key และ Value

โดยปกติแล้ว ThingsBoard หรือ IoT Platform Support รับข้อมูลที่ถูกส่งมาจาก Device ในรูปแบบของ JSON โดย Key ต้องเป็นรูปแบบของ String เสมอ และในส่วนของ Value สามารถที่จะเป็นได้ทั้ง String Boolean, double หรือ JSON

การส่งข้อมูลไปที่ Telemetry ผ่าน HTTP

ในการ update ข้อมูล หรือ ค่าต่าง ส่งไปที่ Telemetry ขึ้นไปบน Telemetry หรือ IoT Platform สามารถใช้ Method Post ได้ตาม URL

```
http(s)://host:port/api/v1/$ACCESS_TOKEN/telemetry
```

รูปที่ 92 URL ส่งข้อมูลไปที่ Telemetry

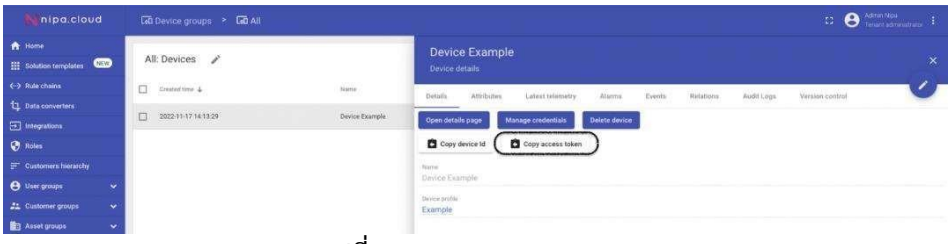
การส่งข้อมูลไปยัง ThingsBoard หรือ IoT Platform โดยที่ไม่ระบุ Timestamp (โดยใช้ค่า Timestamp จาก Servers) $THINGSBOARD_HOST_NAME และ $ACCESS_TOKEN โดยที่ค่าตัวแปรต่างๆ สามารถใช้ตามได้ดังต่อไปนี้

$THINGSBOARD_HOST_NAME : iot.nipa.cloud

$ACCESS_TOKEN : เป็น access token ที่ได้รับมาจากหลังจากที่สร้าง device บน ThingsBoard

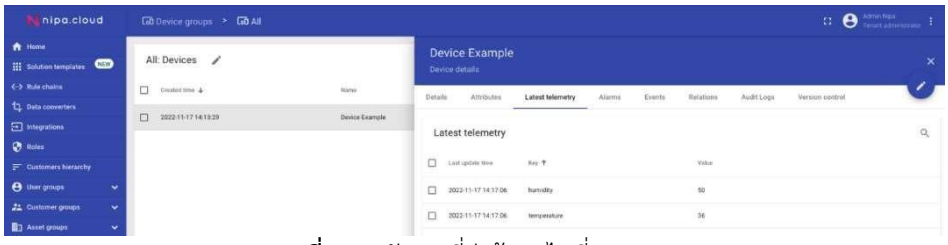
```
curl -v -X POST --data "{"temperature":42,"humidity":73}"
http://$THINGSBOARD_HOST_NAME/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:application/json"
```

รูปที่ 93 ส่งข้อมูลไปที่ Telemetry โดย Method Post



รูปที่ 94 copy access token

ตัวอย่างการใช้งาน



รูปที่ 95 หลังจากที่ส่งข้อมูลไปที่ Telemetry

จากตัวอย่างการส่งข้อมูลไปที่ Telemetry ผ่าน HTTP หลังจากที่ Create Device บน ThingsBoard จะสามารถ copy access token แล้วมาใส่แทนตัวแปร $ACCESS_TOKEN ตามตัวอย่าง และ run command ตาม URL

การส่งข้อมูลไปที่ Attributes ผ่าน HTTP

ในการ upload ข้อมูล หรือค่าต่าง ส่งไปที่ Attributes ขึ้นไปยังบน ThingsBoard หรือ IoT Platform สามารถใช้ Method Post ได้ตาม URL

```
http(s)://host:port/api/v1/$ACCESS_TOKEN/attributes
```

รูปที่ 96 URL ส่งข้อมูลไปที่ Attributes

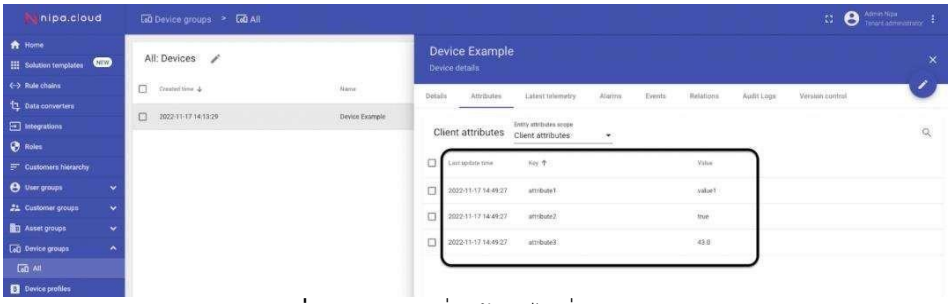
และการส่งค่าไปที่ Attributes สามารถทำได้โดย URL โดยที่จะเห็นว่ามีการ Publish เป็นแบบ Client-Side โดย Update Attribute ค่าไปที่ IoT Platform โดยส่งค่าเป็นดังนี้

attribute1 : value1, attribute2 : true, attribute3 ตามลำดับ

```
curl -v -X POST --data "{"attribute1": "value1", "attribute2":true, "attribute3": 43.0}"
http://$THINGSBOARD_HOST_NAME/api/v1/$ACCESS_TOKEN/attributes --header
"Content-Type:application/json"
```

รูปที่ 97 ส่งข้อมูลไปที่ Attribute โดย Method Post

ตัวอย่างการใช้งาน 2



รูปที่ 98 หลังจากที่ส่งข้อมูลไปที่ Attribute

จากตัวอย่างการใช้งาน upload Attribute สามารถที่จะส่งข้อมูลไปที่ Attribute ผ่าน HTTP โดยที่เป็น Scope ของ Client Attribute ก็จะเห็นได้ว่า ข้อมูลได้ถูกส่งมาที่ Attribute

MQTT

MQTT คือ Protocol ในการรับและส่งข้อมูล ในรูปแบบ Publish และ Subscribe เป็น Protocaol ที่ใช้กันอย่างแพร่หลาย โดย Server สามารถที่จะ Communicate หรือรับและส่งข้อมูลกับ Device หรือ Client แม้ Servers หรือ Client เองจะอยู่คนละ Local Network และสามารถที่จะกำหนด Premission ในการเข้าถึงข้อมูลโดยสามารถที่จะกำหนดชื่อผู้ใช้ และรหัสผ่าน ในการเชื่อมต่อ และรับ หรือส่งข้อมูลระหว่าง ThingsBoard (Servers) และ Device (Client) โดยที่ ThingsBoard Support การรับส่งข้อมูลแบบ Qos (Quality of Services)

การเชื่อมต่อ Device กับ ThingsBoard

โดยที่ ThingsBoard หรือ IoT Platform จะใช้งาน Access Token ในการระบุหรือยืนยันตัวตนของ Device ดังนั้นในการเชื่อมต่อ ThingsBoard กับ Device ต้องใช้ Access Token ของ Device บน ThingsBoard เพื่อที่จะเชื่อมต่อ สามารถดูได้จากตอนที่สร้าง Device แล้ว

รูปแบบของการส่งข้อมูลมาที่ ThingsBoard

```
{
 "stringKey":"value1",
 "booleanKey":true,
 "doubleKey":42.0,
 "longKey":73,
 "jsonKey": {
    "someNumber": 42,
    "someArray": [1,2,3],
    "someNestedObject": {"key": "value"}
 }
}
```

รูปที่ 99 รูปแบบ Key และ Value

โดยปกติแล้ว ThingsBoard หรือ IoT Platform Support รับข้อมูลที่ถูกส่งมาจาก Device ในรูปแบบของ JSON โดย Key ต้องเป็นรูปแบบของ String เสมอ และในส่วนของ Value สามารถที่จะเป็นได้ทั้ง String, Boolean, double หรือ JSON เหมือน

การส่งข้อมูลไปที่ Telemetry ผ่าน MQTT

โดยในการ Pulish ค่าไปยัง Telemetry สามารถที่จะใช้ Topic ได้ตามแต่ละรูปแบบ platform

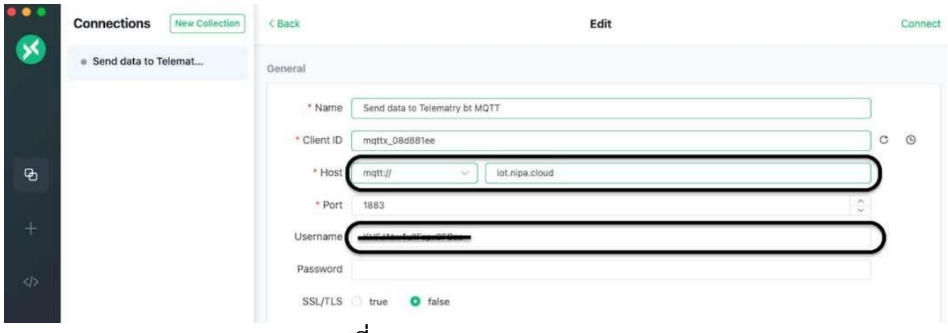
```
v1/devices/me/telemetry
```

รูปที่ 100 Topic publish to telemetry

โดย Telemetry บน ThingsBoard Support การส่งรูปแบบ JSON

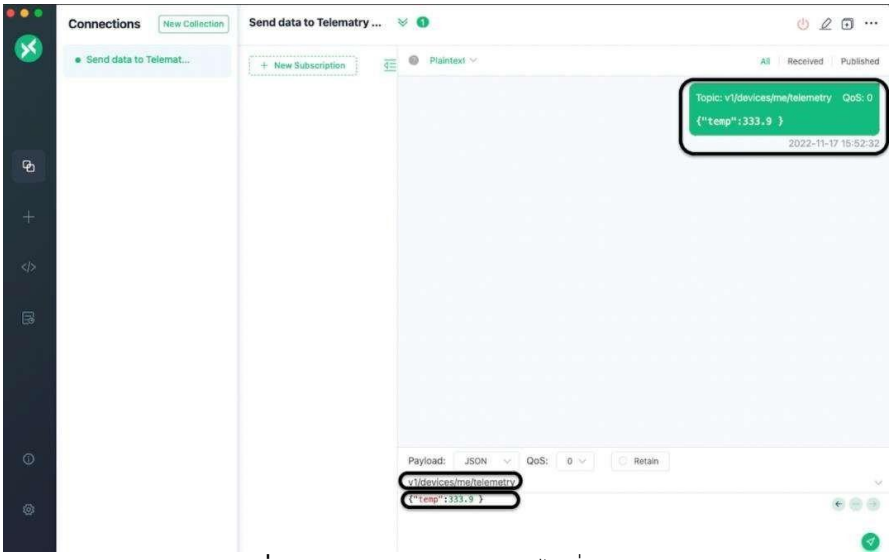
```
{"key1":"value1", "key2":"value2"}
```

รูปที่ 101 รูปแบบ JSON

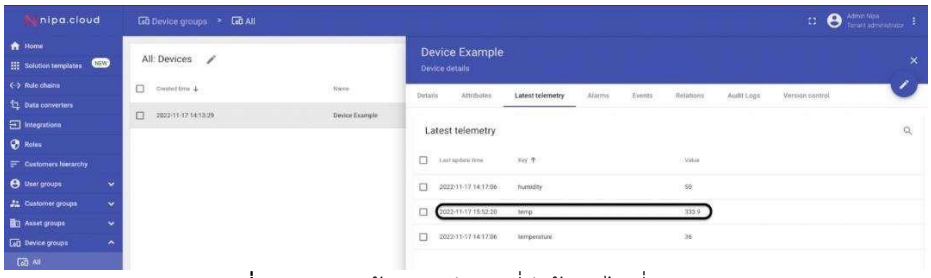


รูปที่ 102 Connect ThingsBoard

ตัวอย่างการใช้งานโดยส่งข้อมูลไปที่ Telemetry ผ่าน MQTT



รูปที่ 103 ส่ง {“temp”:333.9} ไปที่ Telemetry



รูปที่ 104 แสดงข้อมูลหลังจากที่ส่งข้อมูลไปที่ Telemetry

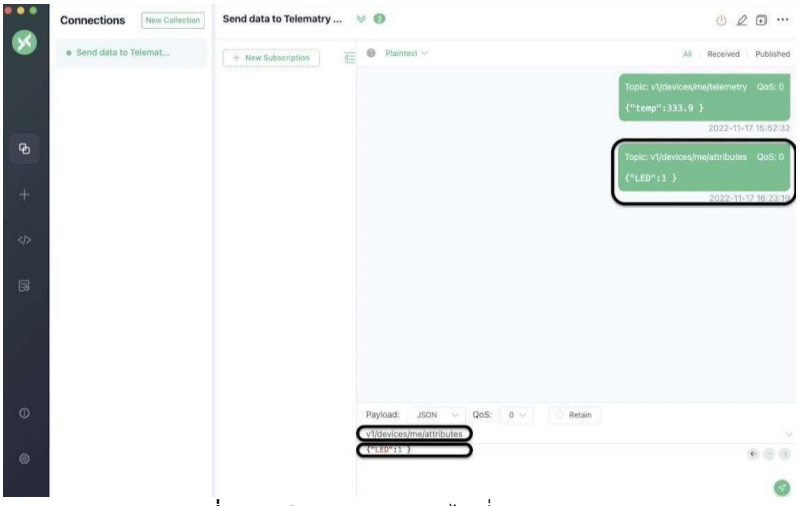
จากตัวอย่างการใช้งานโดยส่งข้อมูลไปที่ Telemetry ผ่าน MQTT โดยที่ใช้ Program MQTT ที่เป็น Program ที่ไว้ใช้เชื่อมต่อ MQTT Broker ผ่าน Protocol MQTT เป็นการ Config การเชื่อมต่อไปยัง IoT Platform (MQTT Broker) หรือ ThingsBoard โดยที่ Host : mqtt://iot.nipa.cloud, Post : 1883 (mqtt), Username : Access Token ที่ได้จาก deviceและหลังจาก connect ไปยัง MQTT Broker เป็นการส่งค่าขึ้นไปยัง MQTT Broker โดยที่ใส่ Topic&Payload ในรูปแบบของ JSON ได้โดย Topic : v1/devices/me/telemetry,Payload : {“temp”:333.9}

การส่งข้อมูลไปที่ Attribute ผ่าน MQTT

โดยในการ Pulish ค่าไปยัง Attribute สามารถที่จะใช้ Topic ได้

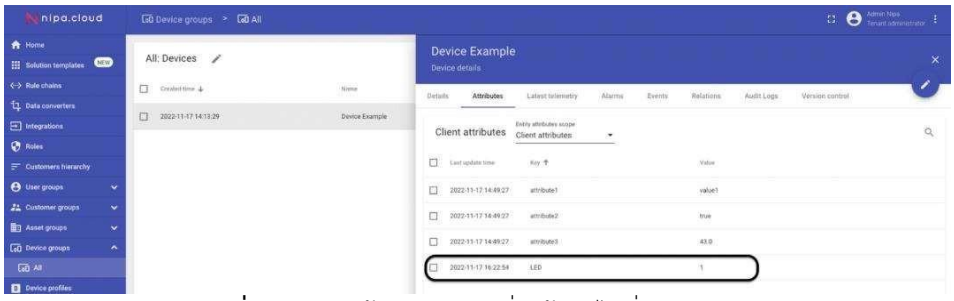
```
v1/devices/me/attributes
```

รูปที่ 105 Topic Pulish to Attribute



รูปที่ 106 ส่ง {“LED” : 1} ไปที่ Telemetry

ตัวอย่างการใช้งานโดยส่งค่าไปที่ Attribute โดยผ่าน MQTT

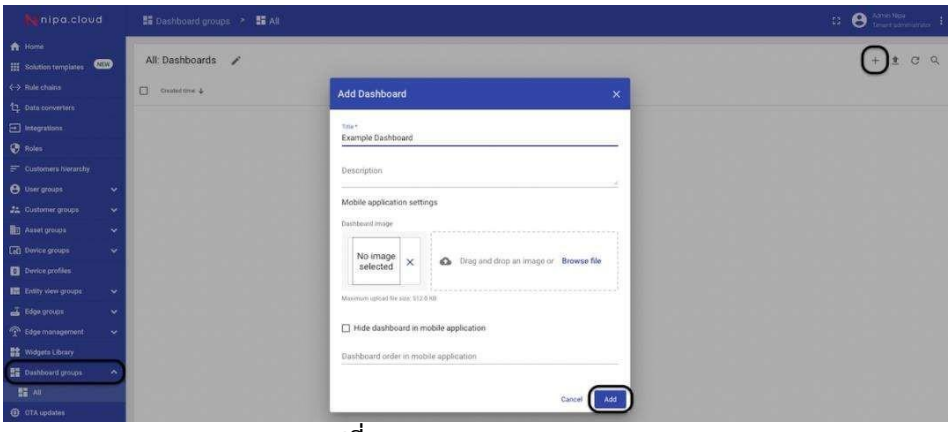


รูปที่ 107 แสดงข้อมูลหลังจากที่ส่งข้อมูลไปที่ Attribute

จากตัวอย่างการใช้งานโดยส่งข้อมูลไปที่ Attribute ผ่าน MQTT โดยที่ใช้ Program MQTTX ที่เป็น Proogram ที่ไว้ใช้เชื่อมต่อ MQTT Broker ผ่าน Protocol MQTT เป็นการ Config การเชื่อมต่อไปยัง IoT Platform (MQTT Broker) หรือ ThingsBoard โดยที่ Host : mqtt://iot.nipa.cloud, Post : 1883 (mqtt), Username : Access Token ที่ได้จาก device

หลังจาก connect ไปยัง MQTT Broker เป็นการส่งค่าขึ้นไปยัง MQTT Broker โดยที่ใส่ Topic และใส่ Paylosd ในรูปแบบ JSON ได้โดย Topic : v1/devices/me/attribute,Payload : {“LED”:1}

การสร้าง Dashboard



รูปที่ 108 Create Dashboard

ในการสร้าง Dashboard สามารถที่จะสร้างได้บน ThingsBoard ที่ Menu Dashboard Group แท็ปซ้ายมือ และสามารถที่จะปรับแต่งได้เพิ่มเติม โดยสามารถ Click ที่ “+” ขวามือบนและใส่รายละเอียดเมื่อเรียบร้อยแล้ว click ที่ “Add”

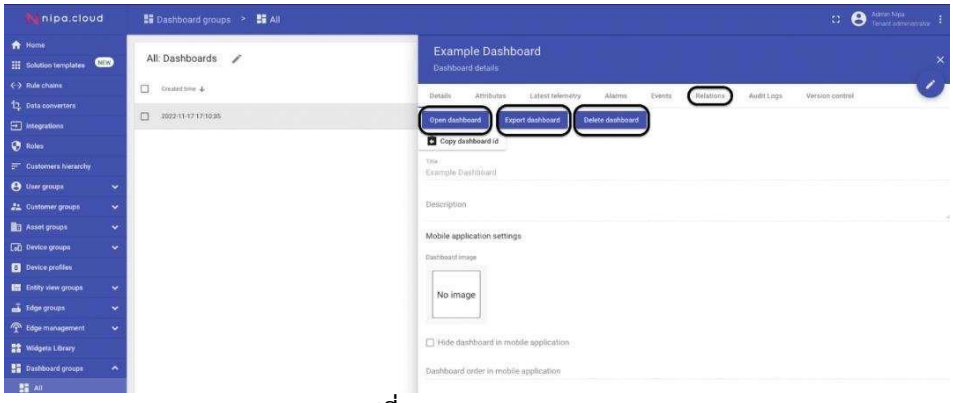
หลังจากที่สร้าง Dashboard เสร็จแล้ว สามารถ click หนึ่งครั้งที่ Dashboard ที่ผู้ใช้สร้างจะมี Menu จากทางด้านซ้ายเข้ามาให้สามารถที่จะจัดการในส่วนของ Dashboard ได้โดยมี

Open dashboard: เปิดหน้า Dashboard

Export dashboard: การ Export Dashboard โดย Dashboard ที่ Export จะอยู่ในรูปของ JSON

Delete dashboard: ลบ Dashboard

Relation: การทำความสัมพันธ์ในการอนุญาตสิทธิ์ในการเข้าถึง Dashboard



รูปที่ 109 Menu Dashboard

หลังจากที่เข้า Dashboard แล้วสิ่งต่อไปที่ทำคือการสร้าง Widget เพื่อที่จะใช้ในการแสดงผลในรูปแบบต่าง ๆ ที่ต้องการ บน Dashboard ที่สร้างขึ้นได้โดย click ที่รูปดินสอ เพื่อที่เข้าสู่โหมด Edit และหลังจากนั้นผู้ใช้ต้องทำการ เพิ่ม Entity เข้ามาที่ Entity aliases ที่มุมขวาบน ถ้าผู้ใช้ต้องการให้ Dashboard แสดงข้อมูลในส่วนของที่ได้รับมาจาก Device ที่เชื่อมต่อเข้ามาโดยเริ่มจาก click add alias ต่อไปจะมี Menu ให้เลือกโดยที่

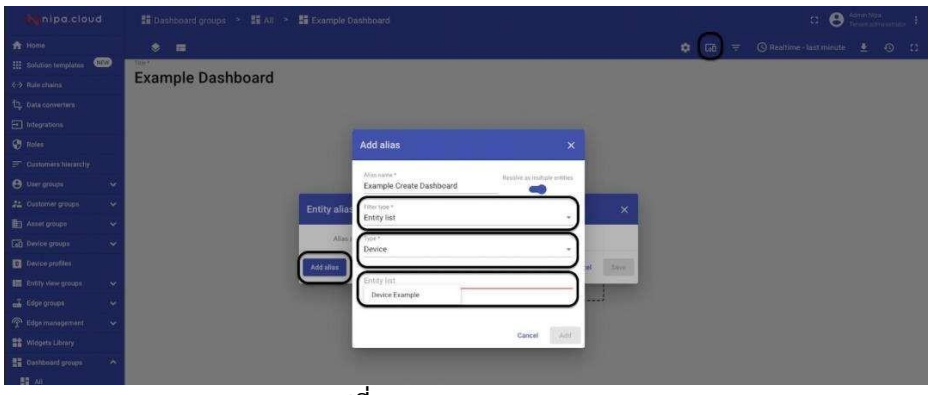
Alias name : เป็นการตั้งชื่อให้กัน Alias ที่ผู้ใช้สร้าง

Filter type : โดยเลือกเป็น Entity list

Type : Device

Entity list : เป็น device ที่ต้องการจะให้ Dashboard แสดงข้อมูล

หลังจากนั้น click ที่ Add และ Save ตามลำดับ



รูปที่ 110 Create Entity aliases

ต่อไปก็พร้อมที่จะสร้าง widget สามารถที่จะเริ่มสร้างได้โดย click จะแสดง widget ต่างๆ โดยที่การใช้งานส่วนใหญ่จะเป็น

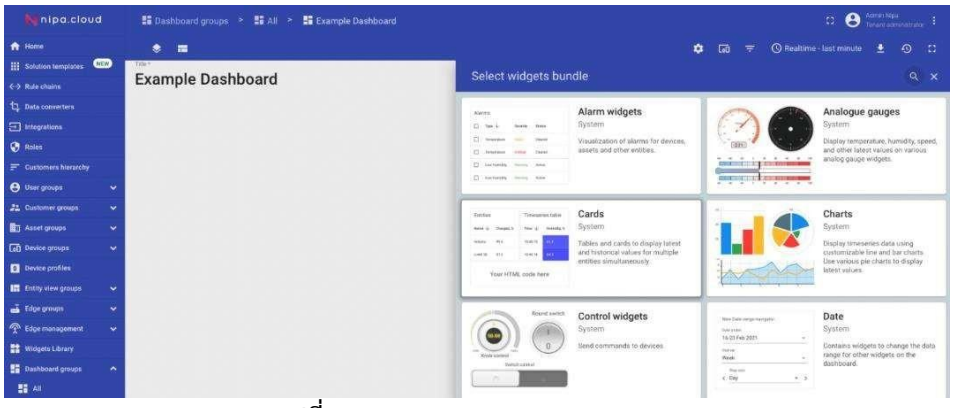
Gauges widget : ที่มีทั้งรูปแบบที่เป็น Digital และ Analogus สามารถที่จะแสดงค่า Telemetry และ Attribute ของ Entity ต่างๆ

Charts widget : เป็นการพล็อตกราฟของค่า Telemetry และ Attribute ของ Entity ต่างๆ

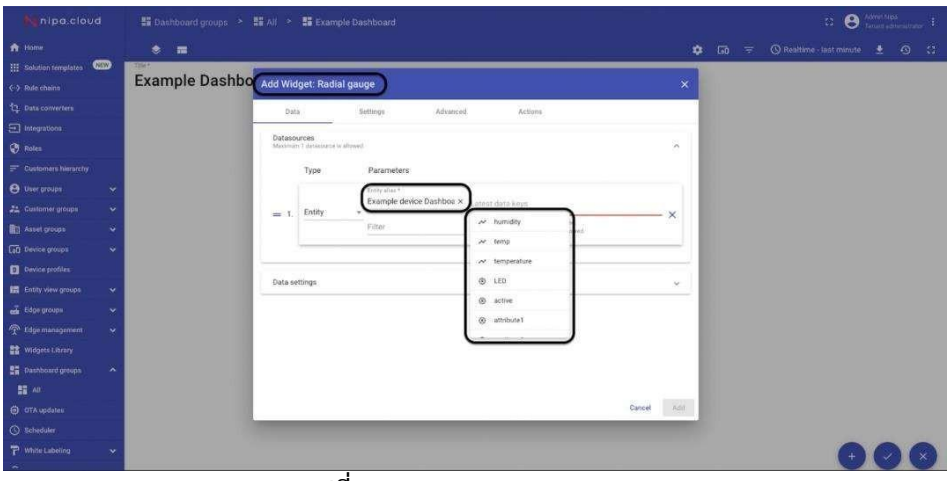
Card widget : เป็นการใช้แสดงรายการของ Entity ต่างๆ โดยเลือกจาก Datasource เพื่อนที่จะกรองข้อมูลที่จะมาแสดงผลได้

Map widget : ใช้แสดงผลผ่านแผนที่ และรูปภาพ แบบ image map

Alarm widget : ใช้ในการแสดง Alarm ผ่าน Dashboard

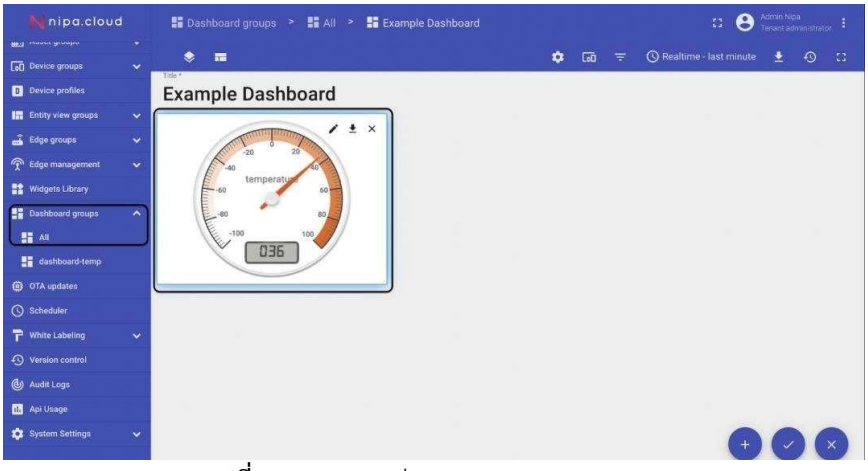


รูปที่ 111 รายการ widget บน ThingsBoard



รูปที่ 112 Add Telemetry to widget

Create widget ที่ใช้โดย widget ที่เลือกใช้เป็น Radial gauge, Data source type ใช้ Entity และ Parameters Entity alias ใช้เป็น Entity ที่สร้างก่อนหน้านี้ จากนั้น Latest datakey โดยที่สามารถเลือกได้ทั้ง Telemetry และ Attribute ที่มีข้อมูลเข้ามา

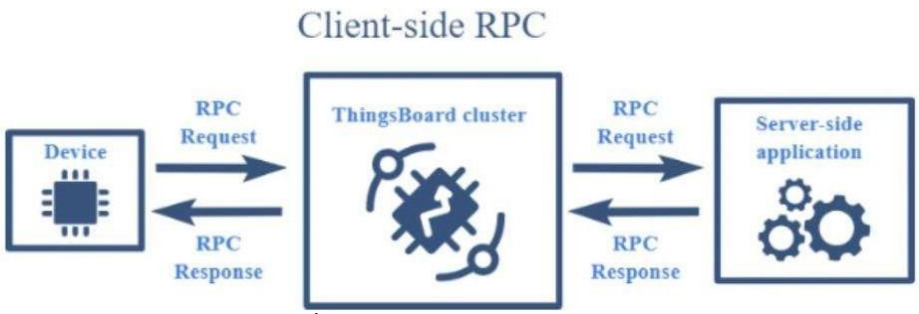


รูปที่ 113 แสดงผลค่า Temp บน Dashboard

จากรูปจะเป็นการเลือกค่าที่จะแสดงบน Dashboard ผ่าน widget โดยที่เลือกค่าแสดงที่เป็น Temp

Section 1.07 Remote procedure call กับการใช้งานบน ThingsBoard

Remote procedure call (RPC) คือ โปรโตคอลในการทำงานระหว่างอุปกรณ์สองอุปกรณ์ในเครือข่ายโดยอุปกรณ์นั้น สามารถเรียกใช้ Procedure ของอุปกรณ์ปลายทางได้ ThingsBoard สามารถใช้งาน RPC ได้สองรูปแบบ ดังนี้



รูปที่ 114 การทำงานของ Client-side RPC

Client-side RPC เป็นรูปแบบที่อุปกรณ์ปลายทางสามารถเรียกใช้งาน RPC บน ThingsBoard Server ได้และได้ผลลัพธ์ของ Procedure ที่เรียกกลับไป

ยกตัวอย่างเช่น

- ระบบรดน้ำโดยที่สอบถามจากพยากรณ์อากาศจากบริการออนไลน์ผ่านระบบ
- อุปกรณ์ปลายทางขอค่าเวลาปัจจุบันจาก ThingsBoard Server

โดยอธิบายการทำงานเบื้องต้นได้ดังนี้ เมื่ออุปกรณ์ส่งค่า RPC มายัง Server ประมวลผล โดย Rule Engine โดย Rule Engine จะสามารถทำการคำนวณการทำงานได้จากค่าต่างๆ ที่ถูกเก็บอยู่ในระบบแล้วส่งกลับไป หรือสามารถส่งต่อคำสั่งที่รับเข้าไปยัง Servers ภายนอกได้

การเรียกใช้งาน RPC แบบ Client-side RPC นั้นข้อมูลที่ส่งไปจะประกอบไปด้วยข้อมูล 2 ชุดที่จำเป็น คือ Method : ชื่อของวิธีการแยกแยะการเรียก RPC ตัวอย่างเช่น “getCurrentTime” หรือ “getWeatherForecast” ค่าของ parameter ที่เป็น stringParams : Parameter เพิ่มเติมที่ส่งไปให้ servers เพื่อประมวลผลโดยจะอยู่ในรูปแบบของ JSON ถ้าหากเป็นค่าเว้นว่างให้ส่งเป็น “{}”

โดยที่วิธีการส่ง Client-side RPC ไปยัง Servers สามารถที่จะทำผ่าน Protocol ได้ดังนี้ MQTT, HTTP, CoAP

```
1  {
2     "method": "getCurrentTime",
3     "params": {}
4  }
```

รูปที่ 115 ตัวอย่าง RPC Request

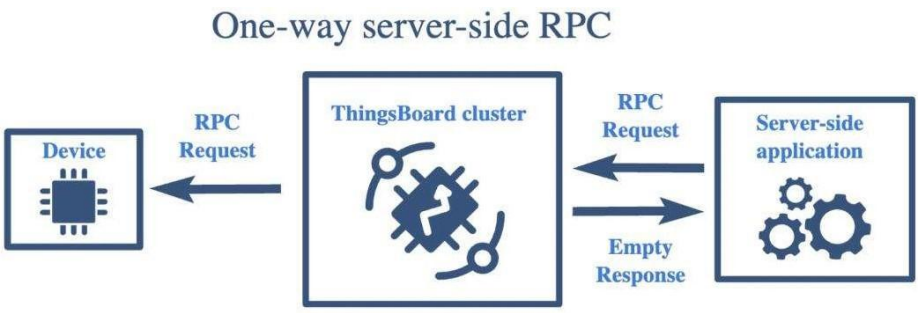
```
1631881236974
```

รูปที่ 116 ผลลัพธ์ RPC Request

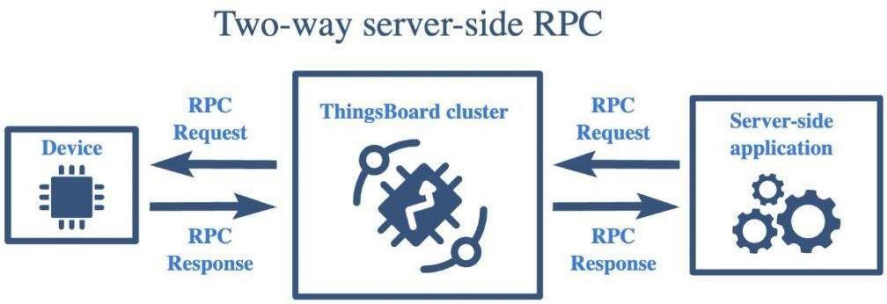
Server- side RPC

ลักษณะการทำงานของ Server- side RPC สามารถที่จะส่ง Request จาก ThingsBoard ไปยังอุปกรณ์และสามารถรับค่าตอบกลับมาที่ Platform ได้โดยที่ usercase ใช้งานทั่วไปของ Server- side ฝั่ง Server จะเป็นการควบคุมระยะไกล เช่น Reboot, on/off หรือแก้ไข config อุปกรณ์

One-way RPC เป็นการส่ง Request โดยไม่สนใจว่าอุปกรณ์จะตอบกลับหรือไม่



รูปที่ 117 RPC Server-side One-way



รูปที่ 118 RPC Server-side Two-way

โดยที่การส่งข้อมูลแบบ Servers-side RPC นั้นจะประกอบได้ด้วยข้อมูลต่างๆ ดังนี้

Method : ชื่อของ Method เพื่อแยกแยะในการเรียก RPC เช่น “getCurrentTime” หรือ “getWeatherForecast” ค่าของ parameter ที่เป็น string (จำเป็นต้องระบุ Method นี้)

Params : parameters เพิ่มเติมที่ส่งไปให้ servers เพื่อประมวลผลโดยจะอยู่ในรูปแบบของ JSON ถ้าหากเป็นค่าเว้นว่างให้ส่งเป็น “{}” (จำเป็นต้องระบุ Method นี้)

Timeout : เป็นการกำหนดค่าของ server ที่ตอบกลับ โดยการประมวลผลเป็นแบบ มิลลิวินาที โดย default จะเริ่มต้นที่ 10,000 (10 Sec) และค่าต่ำสุดที่ 5,000 (5Sec)

Expiration Time : โดยทำงานเหมือน timeout แต่ระบุเป็นค่า Epoch time หรือ Unix timestamp บน timezone ที่ UTC(GMT+0) แทน เช่น บอกว่าจะหมดเวลาที่ 01/01/2024 Retries : เป็นการกำหนด จำนวนครั้งที่จะ RPC ไปที่ Servers ถ้าเชื่อมต่อไม่ได้

```
{
   "method": "setGPIO",
   "params": {
     "pin": 4,
     "value": 1
   },
  "timeout": 30000
}
```

รูปที่ 119 เป็นตัวอย่าง RPC Request

```
{
   "pin": 4,
   "value": 1,
   "changed": true
}
```

รูปที่ 120 เป็นตัวอย่างการตอบกลับในรูปแบบ JSON

โดยที่การส่ง Servers-side RPC โดยปกติจะเป็นการส่งจาก Rest API หรือ Dashboard Widget ซึ่งในความจริง Dashboard Widget เองก็สามารถที่จะ Rest API ได้เองเช่นเดียวกัน ในการทำงานเมื่อ Platform ได้รับ RPC จะทำการตรวจสอบความถูกต้องของข้อมูล และตรวจสอบสิทธิในการส่งการทำงาน หลังจากนั้นคำสั่ง Servers-side RPC จะถูก Convers ไปอยู่ในรูปแบบของ String บน Rule Engine และ Rule Engine จะจัดการสั่งคำสั่งและข้อมูลไปยังอุปกรณ์ปลายทาง

Section 1.08 การใช้งานแบบ Rest API

โดยการส่ง Request RPC ในรูปแบบ HTTP POST สามารถส่ง Request ไปได้ตาม URL

```
http(s)://host:port/api/plugins/rpc/{callType}/{deviceId}
```

รูปที่ 121 ส่ง RPC Request ไปยัง Servers

Http(s)://host:port : URL ของ servers เช่น https://iot.nipa.cloud

CallType : โดยสามารถเลือกได้ว่าจะส่งแบบ One-way หรือ Two-way

Device ID : เป็น ID ของ Device

โดยที่ Request Body ควรที่จะอยู่ไรรูปแบบ JSON โดยใช้ Method setGPIO

```
curl -v -X POST -d @set-gpio-request.json http://localhost:8080/api/plugins/rpc/twoway/$DEVICE_ID \
--header "Content-Type:application/json" \
--header "X-Authorization: $JWT_TOKEN"
```

รูปที่ 122 ส่งข้อมูล Servers-side RPC ด้วย HTTP Request

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept:
application/json' -d '{"username":"tenant@thingsboard.org", "password":"tenant"}'
'http://THINGSBOARD_URL/api/auth/login'
```

รูปที่ 123 คำสั่งขอ $JWT_TOKEN

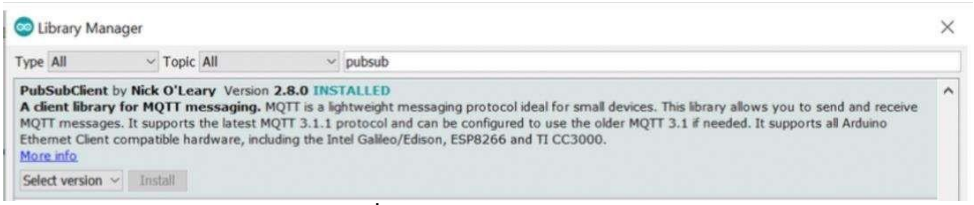
โดยที่ $JWT_TOKEN นั้นสามารถรับได้จากคำสั่งโดยที่ User และ Password ใช้ที่ Login ThingsBoard ใน role ของ Tanant Account

Function Topic	Subscribe	Tx	Rx
Telemetry		① v1/devices/me/telemetry	
Request attributes	① v1/devices/me/attributes/response/+	② v1/devices/me/attributes/request/$request_id	③ v1/devices/me/attributes/response/$request_id
Publish attributes		① v1/devices/me/attributes	
Subscribe attributes update	① v1/devices/me/attributes		② v1/devices/me/attributes
Server-Side RPC	① v1/devices/me/rpc/request/+	③ v1/devices/me/rpc/response/$request_id	② v1/devices/me/rpc/request/$request_id
Client-Side RPC	① v1/devices/me/rpc/response/+	② v1/devices/me/rpc/request/$request_id	③ v1/devices/me/rpc/response/$request_id
Claiming device		① v1/devices/me/claim	

รูปที่ 124 Device MQTT Topic

Section 1.09 ทดลองสร้างอุปกรณ์เพื่อรับและส่งค่าไปยัง ThingsBoard จากบอร์ด ESP32 โดย ArduinoIDE

รูปที่ 125 ThingsBoard library (1)



รูปที่ 126 ThingsBoard library (2)

รูปที่ 127 ArduinoJson PubSabClient by Nick O'leary

หลังจากนั้นสามารถ copy code เชื่อมต่อ ESP32 กับ ThingsBoard จากโค้ดด้านล่าง

```
#include <WiFi.h>              // WiFi control for ESP32
#include <ThingsBoard.h> // ThingsBoard SDK


// WiFi access point
#define WIFI_AP_NAME          “SSID”
#define WIFI_PASSWORD        “PASSWORD”
// ThingsBoard data
#define THINGBOARD_SERVER “iot.nipa.cloud”
```

```
#define TOKEN “TOKEN”

// Helper macro to calculate array size
#define COUNT_OF(x) ((sizeof(x)/sizeof(0[x])) / ((size_t)(!(sizeog(x) % sizeof(0[x])))))

// Initialize ThingsBoard client
WIFIClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the WiFi radio’s status
Int status = WL_IDLE_STATUS;
// LED builtin on board
Int led_pin = LED_BUILTIN;
Int button_pin = 0;
unsigned long previousMillis = 0;
// Interval time for sensor read&send
Const long interval = 5000; // 5 secs
// Set to true if application is subscribed for the RPC messages.
Bool subscribed = false;
// ledState used to set the LED
Int led_state = LOW;
Int button_state = HIGH,prev_botton_state = LOW;
Int RPC_temp = 0;

RPC_Response processGetLedState(const RPC_Data&data)
{
Serial.prinln(“Received the get value method”);

return RPC_Response(NULL, Led_state);
}

RPC_Responce processSetLedState(const RPC_Data&data)
{
Serial.prinln(“Received the set GPIO RPC method”);
```

```
Bool state = data;//data[“value”];
digitalWrite(led_pin, state);
led_state=state;

return RPC_Response(“LED”, state);
}
RPC_Responce processGetTempValue(const RPC_Data&data)
{
Serial.prinln(“Received the set GPIO RPC method”);
return RPC_Response(“NULL”, RPC_temp);
}

RPC_Responce processSetTempValue(const RPC_Data&data)
{
Serial.prinln(“Received the set temp method”);
RPC_temp=(int)data;
return RPC_Response(“temp”, RPC_temp);
}

// RPC handlers
RPC_Callback callbacks[] = {
{ “setLedValue”,  processSetLedState },
{ “getLedValue”,  processGetLedState },
{ “setTempValue”,  processSetTempValue },
{ “setTempValue”,  processGetTempValue },
};

// Setup an application
Void setup() {
// Initialize serial for debugging
Serial.begin(115200);
pinMode(led_pin, OUTPUT);
pinMode(button_pin, INPUT_PULLUP);
WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
InitWiFi();
```

```
}

// Main application loop
void loop() {
Unsigned long currentMillis = millis();
// Reconnect to WiFi, if needed
If (WiFi.status() !=WL_CONNECTED) {
Reconnect();
Return;
}
// Reconnect to ThingsBoard, if needed
If (!tb.connected()) {
  Subscribed = false;
  // connect to the ThingsBoard
       Serail.print("Connecting to: ");
       Serail.print(THINGSBOARD_SERVER);
       Serail.print("with token");
       Serail.print(TOKEN);
If (!tb.connected(THINGSBOARD_SERVER, TOKEN)) {
       Serail.println("Failed to connect");
       Return;
   }
{
 // Subscribe for RPC, if needed
If (Subscribed) {
       Serail.println("Subscribed for RPC...");
        Return;
}
Serail.println("Subscribed done");
Subscribed = true;
}

// Check if it is a time to send DHT22 temperature and humidity
If (currentMillis – previousMillis >= interval) {
    previosMillis = currentMillis;
```

```
    Serial.println("Sending data...")
    tb.sendTelemetryFloat("temp"RPC_temp);
    tb.sendTelemetryFloat("hum",random(30,60));
}
 button_state = digitalRead(button_pin);

// check if the pushbutton is pressed. If it is, the buttonState is HIGH:
If (botton_state == LOW && prev_button_state==HIGH) {
    prev_button_state;
    led_state=!led_state;
    digitalWrite(led_pin, led_state);
    tb.sendAttributeInt("LED", led_state);
    delay(100);
 }  else if(button_state == HIGH) {
    prev_button_state=HIGH;
 }
 // Process messages
    Tb.loop();
}
Void InitWiFi()
{
Serial.println("Connecting to AP...");
// attempt to connect to WiFi network

WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
While (WiFi.status() != WL_CONNECTED) {
     Delay(500);
     Serial.print(".");
}
Serail.println("Connected to AP");
}

Void reconnect() {
 // Loop until we're reconnected
     Status = WiFi.status();
```

```
    If (status ! = WL_CONNECTED) {
      WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
      While (WiFi.status() != WL_CONNECTED) {
        Delay(500);
        Serial.print(“.”);
    }
        Serail.println(“Connected to AP”);
    }
}
```

รูปที่ 128 Code Arduino board ESP32 connect to thingboard

โค้ดคำสั้งด้านล่างเป็นการเชื่อมต่อ thingboard โดยใช้ Arduino IDE โดยที่ Config

```
#define WIFI_AP_NAME  “SSID” // ใส่ชื่อ wifi ที่เชื่อมต่อ
#define WIFI_PASSWORD “PASSWORDSSID” // ใส่ password wifi ที่เชื่อมต่อ
#define THINKSBOARD_SERVER “server” // ใส่ servers iot platform ที่ต้องการเชื่อมต่อ
#define TOKEN “TOKEN” // ใส่ token หลังจากสร้าง device
```

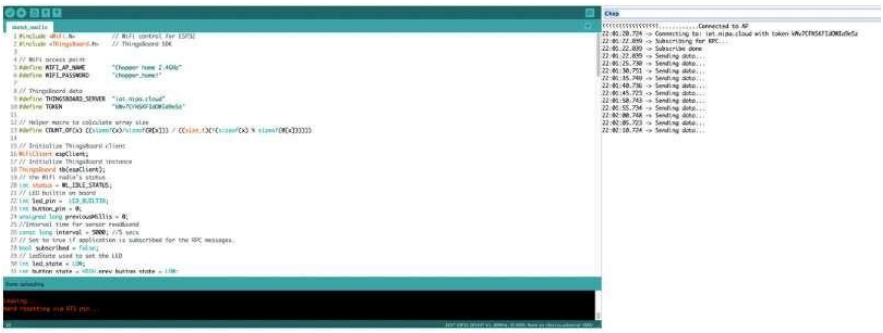
```
// ประกาศใช้งาน WiFiClient และ ThingsBoard
WiFiClient espClient;
ThingsBoard tb(espClient);
```

```
// function เพื่อทำงานตาม RPC request ที่รับมา
RPC_request processGetLedState(const RPC_Data&data)
{
Serail.println(“Received the get value method”);
Return RPC_Response(ULL, led_state);
}
RPC_Response (“Received the set GPIO RPC method”);
Bool state = data;//data[“value”];
digitalWrite(led_pin, state);
led_state=state;
Return RPC_Response(“LED”, state);
}
```

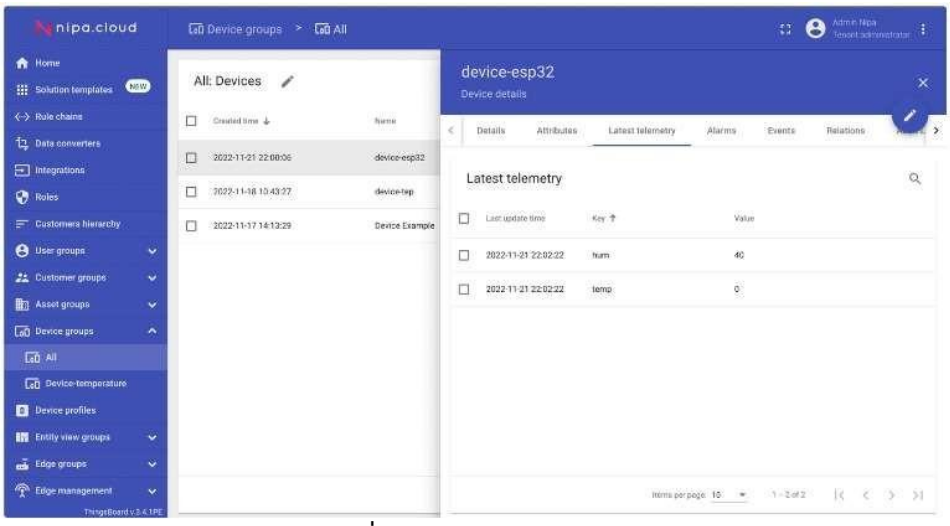
```
// RPC handle ประกาศ function กับ RPC request ที่จะใช้งาน
```

```
RPC_Callback Callbacks[] = {
{ “setLedValue”, processSetLedState },
{ “setLedValue”, processGetLedState },
};
Viod loop() {
// เช็ค connection ไปยัง ThingBoard ถ้ายังเชื่อมต่อไม่ได้จะทำการเชื่อมต่อซ้ำ
If (!tb.connected()) {
Subscribed = false;
If (!tb.connect(THINGBOARD_SERVER, TOKEN)) {
Serail.println(“Failed to connect”);
Return;
   }
}
// check การ subscribe RCP request ไป ThingBoard ถ้ายังจะทำการ subscribe
If (!subscribe) {
Serail.println(“Subscribing for RPC…”);
If (!tb.RPC_Subscribe(callbacks, COUNT_OF(callbacks))) {
Serail.println(“Failed to subscribe for RPC”);
Return;
}
Serail.println(“Subscribe done”);
Subscribed = true; }
// ส่งข้อมูลไปยัง ThingBoard โดยส่งได้ทั้ง Attribute และ Telemetry
tb.sendTelemetryFloat(“hum”,random(30,60));
tb.sendAttributeInt(“LED”, led_state);
// สั่งให้ maintain ThingBoard connection
Tb.loop(); }
```

รูปที่ 129 รายละเอียดของโค้ดการเชื่อมต่อ ESP32 กับ ThingsBoard



รูปที่ 130 ผลลัพธ์ ESP32 Connect to ThingBoard เรียบร้อย



รูปที่ 131 ผลลัพธ์บน ThingBoard

Section 1.10 Rule Engine

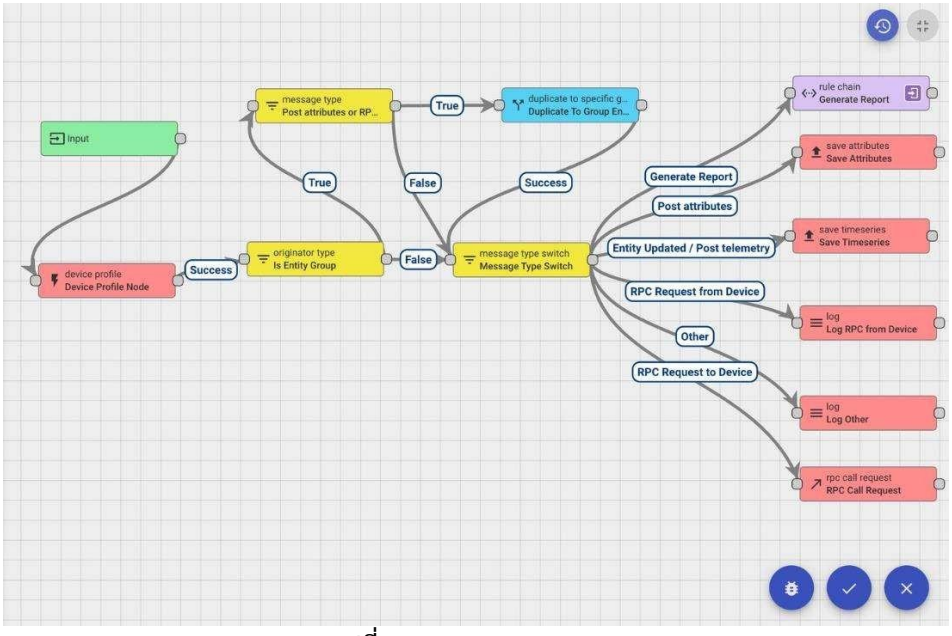
โดยที่ ThingBoard Rule Engine เป็นระบบที่ช่วยให้ผู้ใช้สามารถจัดการกับการประมวลข้อมูลที่มีความซับซ้อนได้โดยสามารถที่จะ filter หรือแก้ไขเนื้อหาของข้อมูลที่สั่งมาจากอุปกรณ์ IoT และ Asset ที่เกี่ยวข้อง และรายงานยังสามารถสร้าง trigger ได้หลากหลาย เช่น การสร้างระบบแจ้งเตือน เมื่อมีค่าจากอุปกรณ์ที่ส่งมาเข้ามาผิดปกติ หรืออุปกรณ์ offline ไปจากระบบ รวมถึงสามารถส่งต่อข้อมูลไปยังระบบภายนอกได้

ส่วนประกอบของ Rule Engine

Rule Engine Message : เป็นข้อมูลแบบโครงสร้างที่ใช้แทน Message ต่างๆในระบบ เช่น ข้อมูล Telemetry ที่เข้ามาข้อมูล Attribute หรือ RPC call จากอุปกรณ์ข้อมูลสถานะของอุปกรณ์ เช่น เชื่อมต่อ หรือ ยกเลิกการเชื่อมต่อ

Rule Engine Message จะประกอบด้วย Message ID โดยเป็นเลขสำหรับระบุข้อความจะไม่ซ้ำกัน และมีค่ามากขึ้นไปเรื่อยๆOriginator of the message บอกว่าข้อมูลชุดนี้ ถูกสั่งมาจาก Entity อะไรใน servers Type of the message ประเภทของ message เช่น Post telemetry หรือ Inactivity EventPayload of the message : ข้อมูล JSON ที่เก็บข้อมูลที่ใช้งานจริงๆ Metadata กลุ่มของข้อมูล JSON ที่เก็บข้อมูลที่เกี่ยวข้องอื่นๆ ของข้อความ Rule Node ถือเป็นส่วนประกอบพื้นฐานของ Rule Engine ซึ่งจะทำหน้าที่ ประมวลผลข้อมูลที่รับเข้ามาในแต่ละครั้ง และให้ผลลัพธ์ออกมา ซึ่งการประมวลผล เงื่อนไขการทำงานต่างๆ จะทำบน Rule node เช่น การ filter หรือ การเพิ่มข้อมูล และการแปลงข้อมูล การส่งต่อข้อมูลออกไปยังระบบภายนอก

Rule Node Relation Rule Node จะมีความสัมพันธ์ กับ Rule Node อื่นๆ ได้โดยทุกๆ ค่าความสัมพันธ์จะต้องมีประเภทของความสัมพันธ์ โดยข้อมูลขาออกไปยัง rule node อื่นๆ จะต้องมีการเลือกเงื่อนไขของความสัมพันธ์นี้ทุกครั้ง โดยปกติข้อมูลขาออกจะเป็น “seccess” หรือ “failure” เพื่อให้สามารถเลือกได้ว่าการประมวลผลใน Rule Node ถ้า success จะต้องไปที่ Rule node ไหน หรือถ้า failure จะไปทำใน Rule Node ไหนต่อไป ในบาง Rule node จะมี relation พิเศษให้เลือกใช้ได้อีก โดยสามารถดูได้ตอนที่ลากเชื่อมต่อ Rule node เข้าหากัน



รูปที่ 132 Root Rule Chain

Rule Chain คือการนำส่วนประกอบต่างๆมารวมกันเป็นกลุ่มของการทำงาน สามารถเข้าไปจัดการได้ยัง Menu Rule Chain ซึ่งจะพบกับ Rule Chain ต่างๆ ที่มีในระบบ สามารถสร้าง Rule Chain ใหม่ เข้าไปในระบบได้โดย Rule Chain หลัก คือ Root Rule Chain ซึ่งใน Root Rule Chain จะประกอบด้วย Rule node การทำงานต่างๆ

ข้อมูลถูกรับเข้ามายัง Input node ซึ่งสามารถมาจาก Rule node อื่นๆได้

ข้อมูลถูกส่งต่อไปยัง Device profile node เพื่อตรวจสอบค่าเพื่อเปรียบเทียบเงื่อนไข Alarm rule ที่สามารถตั้งได้ใน Device profile ของอุปกรณ์

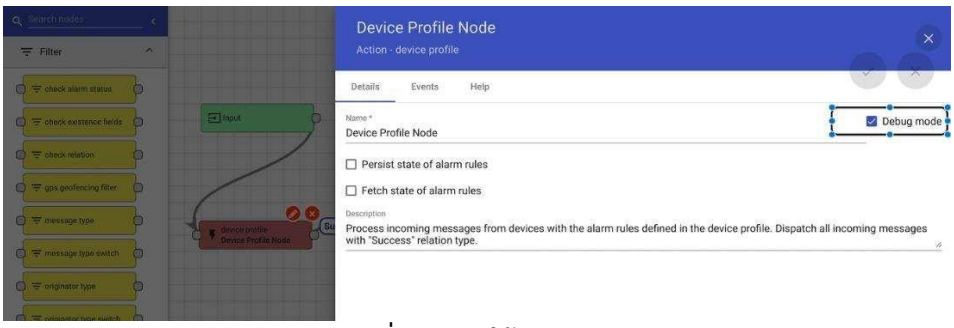
หากข้อมูลถูกต้องไม่มีปัญหาหรือ Success จะถูกส่งต่อไปยัง Entity Group เพื่อที่จะ Process ต่อไป ส่งไปยัง Message type switch เพื่อทำการส่งข้อมูลไปยัง node ถัดไปให้ตรงกับ Message type ที่ส่งเข้ามาไม่ว่าจะเป็น attribute, telemetry, RPC request จากอุปกรณ์, RPC request ไปยังอุปกรณ์อื่นๆ

Node Save client attribute ทำการบันทึกข้อมูล client attribute ลงฐานข้อมูล

Node Save timeseries ทำการบันทึกข้อมูล telemetry ลงฐานข้อมูล

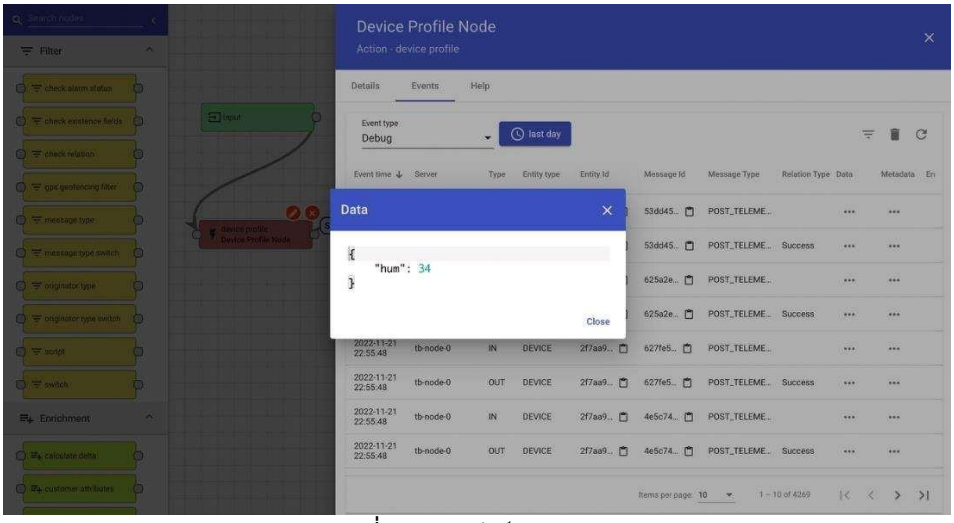
Node Log ทำการบันทึก RPC request log ลงฐานข้อมูล

Node RPC call Request ทำการส่ง RPC request ไปยังอุปกรณ์ปลายทางที่ระบุ



รูปที่ 133 การใช้ debug mod

การ debug ข้อมูลใน Rule Node ในการ Rule Node ทุกตัว จะสามารถ debug ข้อมูลเข้าออกเพื่อตรวจสอบความถูกต้องของข้อมูล ได้โดยการเข้าไปยัง node แล้วเลือกที่ debug mod

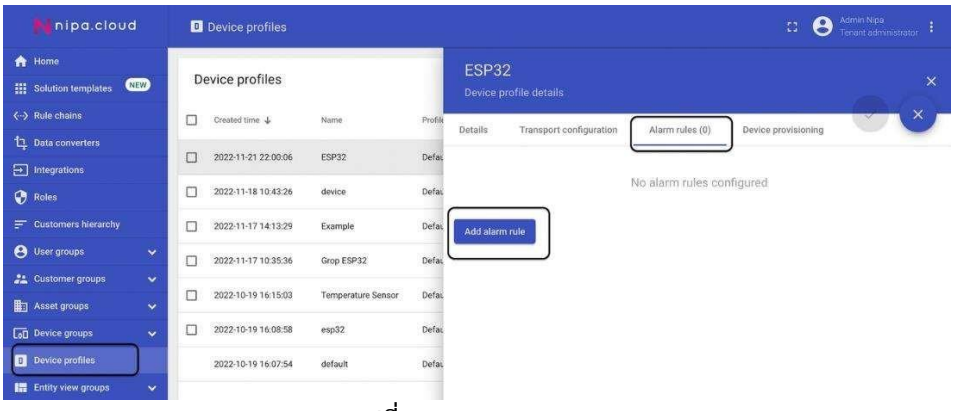


รูปที่ 134 ผลลัพธ์จากการ debug

และจากนั้นสามารถเข้าไปยัง tab event เพื่อดูข้อมูล event ต่างๆ ได้โดยหากเลือกเป็น debug ก็จะสามารถดูข้อมูลเข้าและออกได้โดยดูได้ทั้ง data, metadata และ error ต่างๆ

เขียน rule chains เพื่อส่งการแจ้งเตือนไปยัง Application LINE ตามเงื่อนไขที่กำหนด

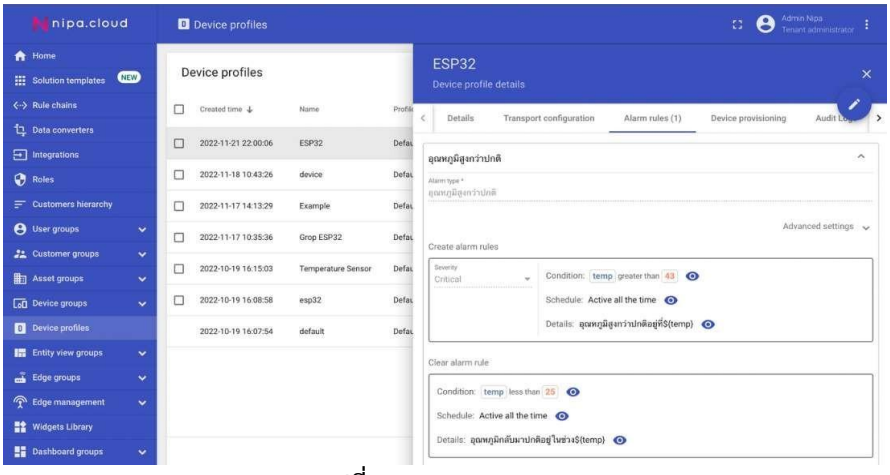
ทดลองการทำ Alarm ของอุปกรณ์ต่างๆ เพื่อแสดงผล Alarm บน Dashboard และนอกจากนั้นจะมีการส่งข้อมูลแจ้งเตือนผ่าน Line Nitify การตั้งค่า Alarm ของอุปกรณ์



รูปที่ 135 config alarm rule

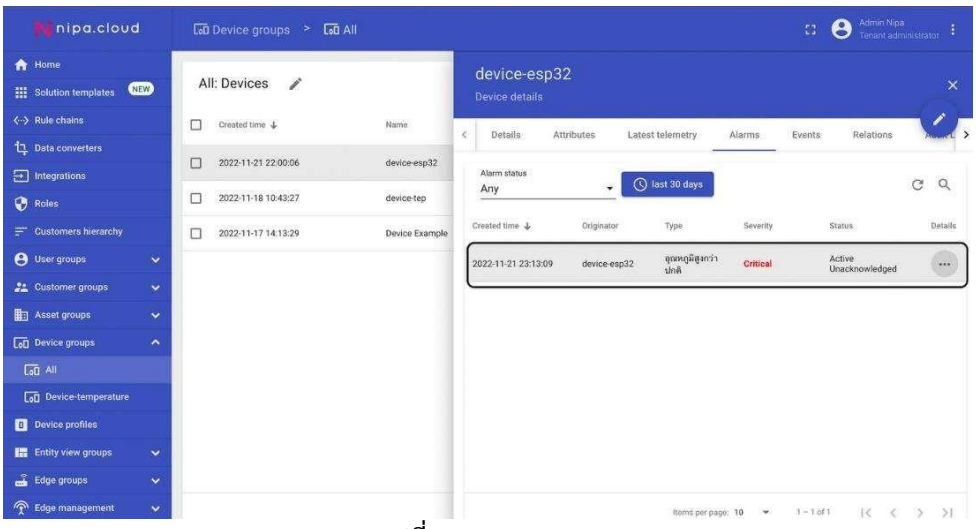
สามารถตั้งค่าการแจ้งเตือนของอุกรณ์ต่างๆ ได้ โดยเข้าไปตั้งค่าใน device profile ที่อุปกรณ์ เลือกใช้อยู่โดยเข้าไปยัง device profile detail จะมี tab “alarm rule” ให้สามารถเข้าไปตั้งค่าได้

โดยที่ในส่วนของ alarm rule นั้นจะสามารถตั้งค่าเงื่อนไขการสร้าง alarm ขึ้นและเงื่อนไขการ



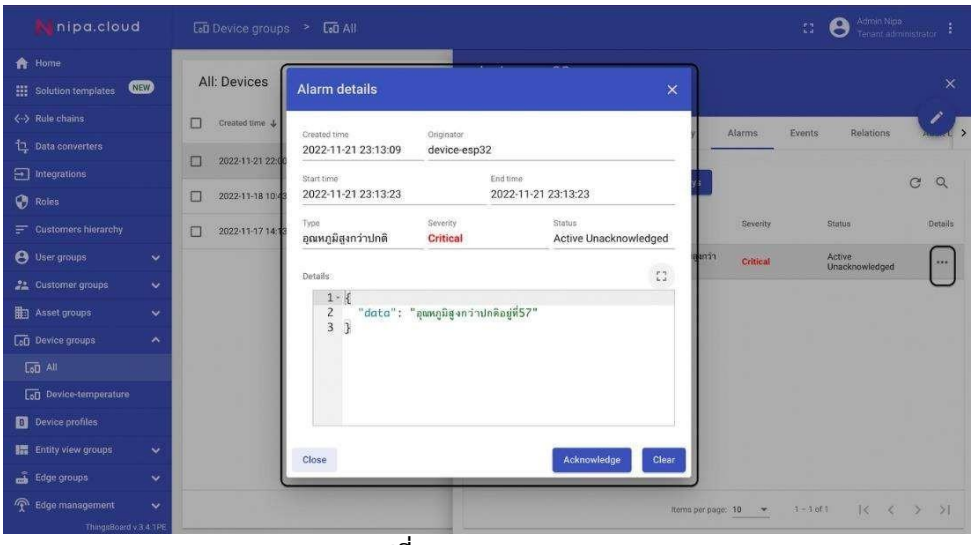
รูปที่ 136 create alarm

ลบ alarm ที่เกิดขึ้นได้คือกำหนดเงื่อนไขให้หากค่า temp สูงกว่า 43 องศาเซลเซียส จะให้มีการแจ้ง alarm ข้อความที่จะทำการลบการแจ้งเตือน เมื่อเงื่อนไขพบว่าค่า temp ต่ำกว่า 25 องศาเซลเซียส



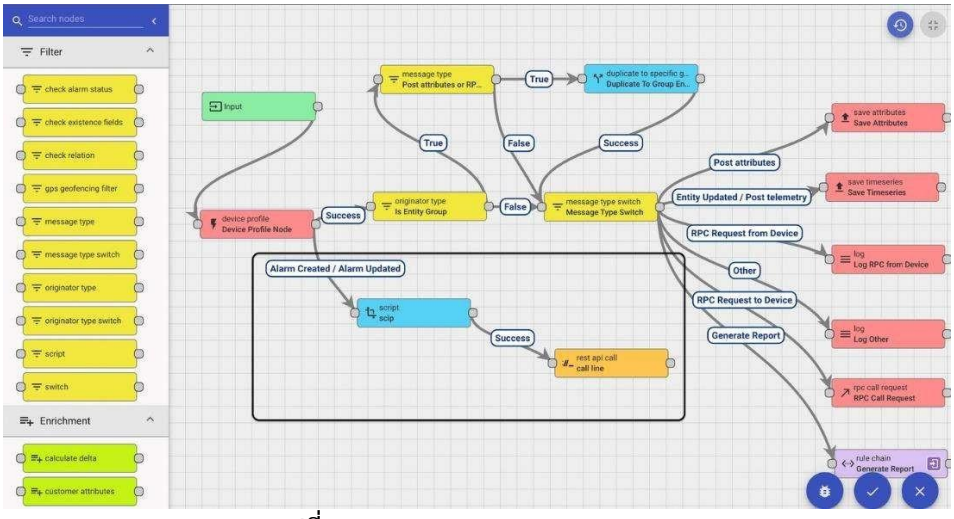
รูปที่ 137 alarm tab

หลังจากที่กำหนดค่า mqtt publish ให้ค่าเกินได้แล้ว จากนั้น สามารถเข้าไปดูว่าเกิด alarm ขึ้น หรือไม่ที่ device ภายใน tab alarm และสามารถดู detail ภายในได้



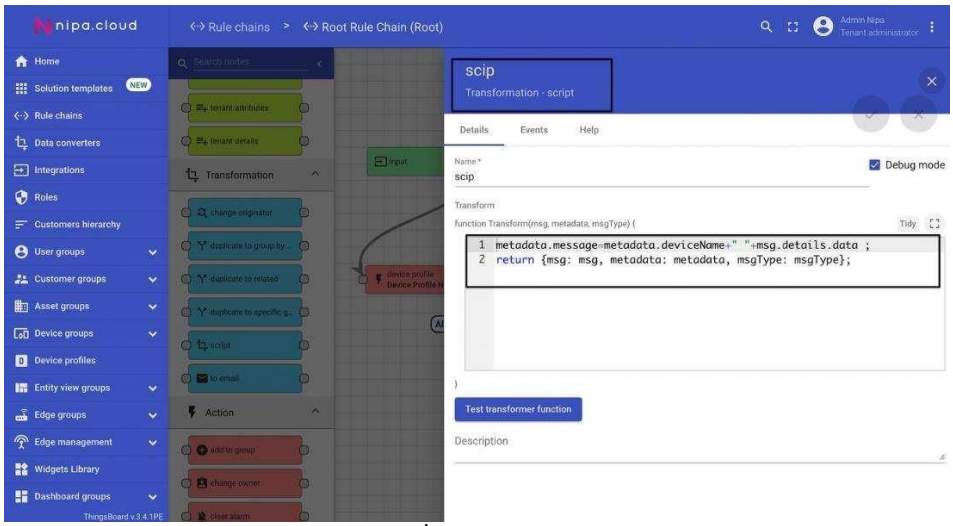
รูปที่ 138 alarm detail

หลังจากนั้นไปที่ cerate rule node เพื่อที่จะส่ง alarm ไปยัง LINE Notification โดยที่จะมี 2 ส่วน



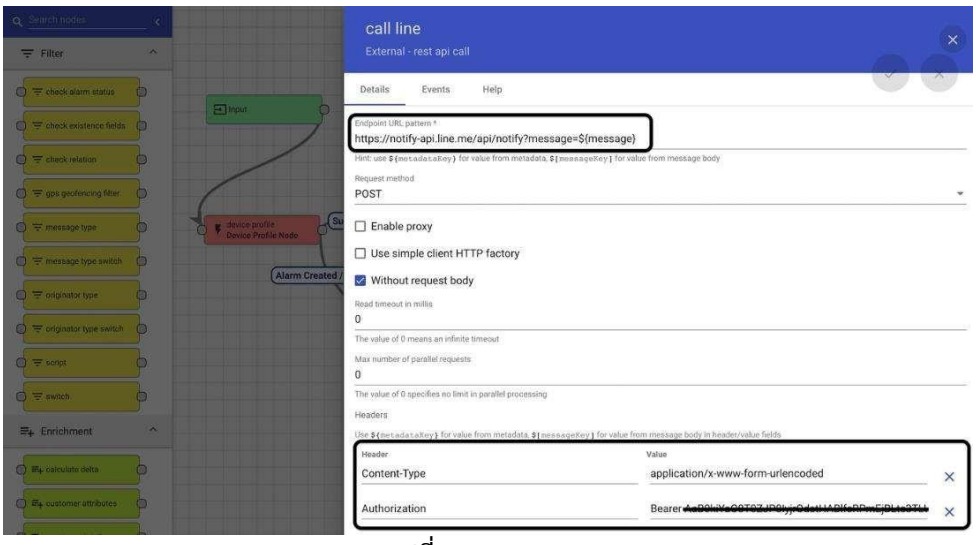
รูปที่ 139 cerate rule send alarm to LINE

Script node โดยผู้ใช้เพิ่ม Transformation script node มาแก้ไขเพื่อเพิ่มชุดคำสั่งในตารางด้านล่างเข้าไป โดยประกาศให้ metadata.message มีค่าเป็น metadata.deviceName ซึ่งเป็นชื่อของอุปกรณ์ที่เกิด alarm และ msg.details.data หรือคำอธิบายของ alarm ที่ผู้ใช้ได้ใส่ไว้ในตอนสร้าง alarm จากนั้น กดเลือก debug mode เพื่อให้ง่ายในการตรวจสอบข้อมูลภายหลัง

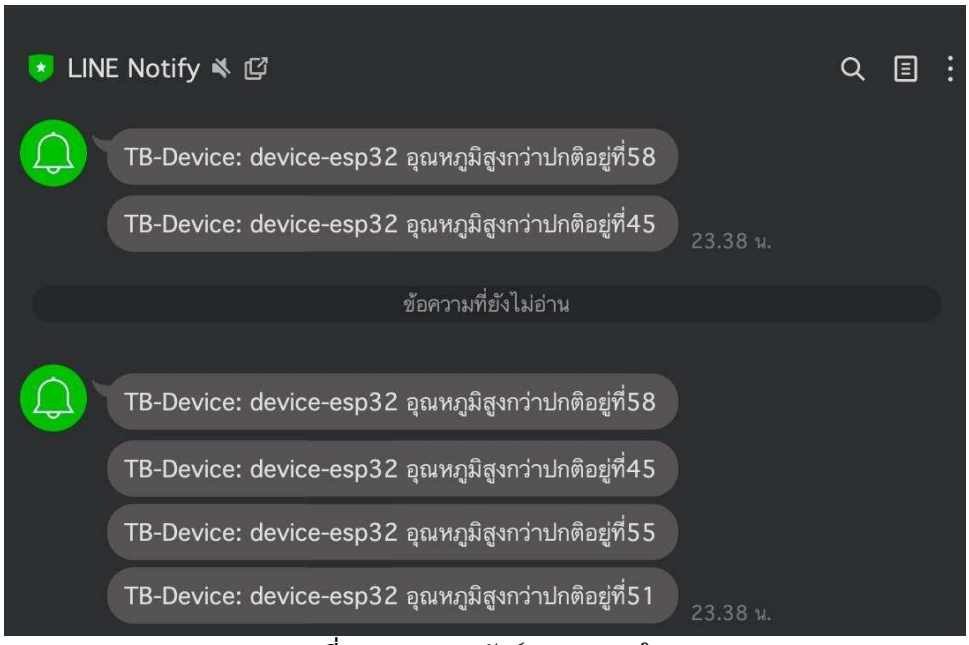


รูปที่ 140 Script node

Rest API Call โดยที่ใส่ข้อมูล URL ของ LINE Notify คือ https://notify-bot.line.me/doc/en/ ลงไปโดยเติมข้อมูล ?message=${message} ลงไปต่อท้ายในส่วน ${message} นี้จะเป็นข้อมูลใน metadata ที่จะใช้มาเป็นข้อความที่ส่งไปแจ้งเตือนผ่าน LINE Notify



รูปที่ 141 Rest API Call



รูปที่ 142 แสดงผลลัพธ์การ alarm ใน LINE

White-labeling

ThingBoard Web interface อนุญาตให้กำหนดสามารถที่จะกำหนด Logo หรือสีโดยไม่ต้องเขียน code หรือ restart service ใหม่ สามารถที่จะกำหนดชุด color, Icon และ facvicon ที่ในระดับของ system admin ได้

Tenant และ Customer Admin UI จะ sync config โดย default

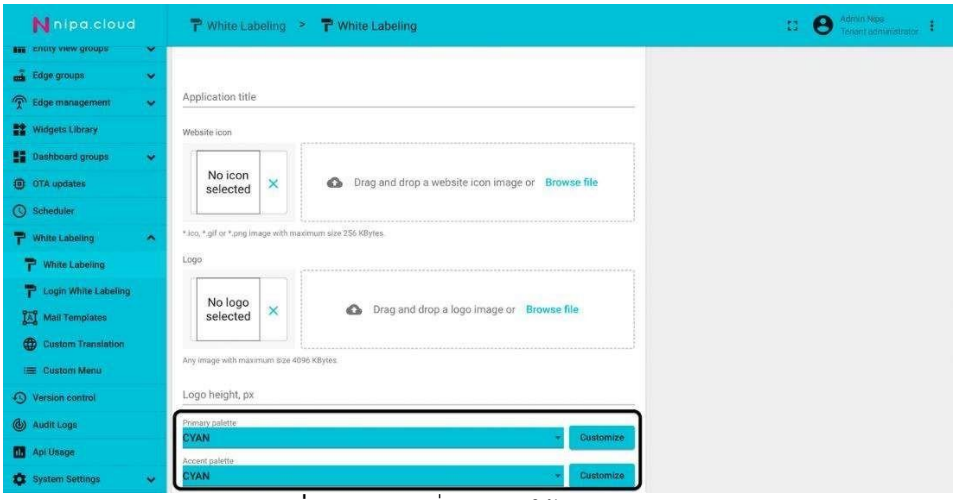
Tenant และ Customer Admin จะสามารถที่จะ config White-labeling เองได้

System และ Tanant Admin สามารถที่จะตั้งค่า กำหนด Email server และ กำหนด Email templates เพื่อที่จะ interact กับ user

Allow System Admin ในการ enable/disable white-labeling สำหรับแต่ละ tanant

Allow System Admin ในการ enable/disable white-labeling สำหรับแต่ละ customer

Allow System Admin ในการกำหนด translation สำหรับ component และ end-user



รูปที่ 143 การเปลี่ยนสี การใช้งาน IoT Platform

Advanced Role-Based Access Control (RBAC) for IoT devices and applications

ThingBoard Community Edition (CE) จะ support ความปลอดภัยด้วย main role 3 หลัก คือ System Administrator, Tanant Administrator และ Customer user

Systemadmin : สามารถที่จะ Manage tenants ได้

Tanants : สามารถที่จะ Manage Device, Dashboard, Customer ที่เป็น entities ของ tenants

Customer user : สามารถที่จะดู Dashboard และ Device ที่ถูก assigned ให้กับ Customer

Section 3.01 Glossary

Tanant : จะเป็นการแยก business-entity : โดยจะมองว่าเป็นบุคคล หรือ องค์กร ที่เป็น Owner ของ device หรือ assets และ Tanant เองสามารถมีได้หลาย Tanant admin, สามารถมีได้หลายล้าน customer

Entity : สามารถเป็นได้ทั้ง Device, Asset, User, dashboard, entity view และทุก entity สามารถที่ manage ได้โดย ThingBoard

Entity Group (EG) : Entity Group เป็นกลุ่มสำหรับ entities ที่เป็น type เดียวกัน ยกตัวอย่าง เช่น Device Group, Assets Group โดยที่ 1 Entity สามารถที่จะเป็นของหลาย Entity Group

Customer โดย customer สามารถที่จะแยก Business – Entity : โดยจะมองว่าเป็น บุคคล หรือ องค์กร หรือที่ใช้ Tanant Device และ Asset โดยที่ customer สามารถที่จะอยู่ใน Tanant Oranization ได้ และตัว customer เองสามารถที่จะหลาย user ใน customer และมีได้หลายล้าน device, asset ใน customer

User : สามารถที่จะ login ได้จากทาง Web ThingBoard และสามารถที่จะ REST API call, เข้าถึง Device และ Asset ถ้า Asset นั้นเปิด Allow ให้เข้าถึงได้ และ user ก็เป็น Entity ของ ThingBoard

User Group (UG) : โดย User Group ก็คือ EG (Entity Group) และมีความสามารถหรือ Feature เหมือน EG โดยต่างกันแค่ชื่อเพื่อง่ายต่อการเรียก ThingBoard Professional Edition (PE) จะมีความ flexibility ใน terms สำหรับ user, customer และ role ในการ management โดยที่ ThingBoard PE ถูก designed ให้ครอบคลุมในการใช้งานสำหรับ businesses และ Enterprises โดยที่มี user groups หลาย groups และมีสิทธิที่ต่างกัน แต่สามารถที่จะ interact กับ device และ Assets ที่เหมือนกันได้

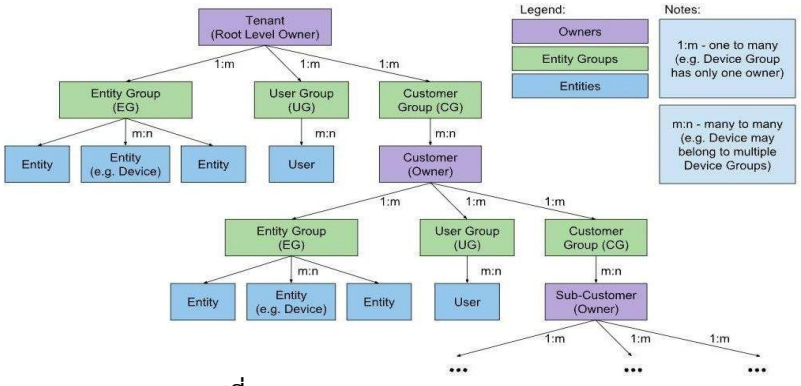
ThingBoard PE V.2.3.0 จะมี feature security ที่ support advance RBAC ดังนี้
ในระบบ ThingsBoard สามารถสร้างลำดับของลูกค้าได้หลายระดับ เช่น ลูกค้าย่อย (Sub-Customer) ผู้ใช้งานอิสระ อุปกรณ์เพื่อช่วยจัดการข้อมูลและการใช้งานได้อย่างเป็นระเบียบและมีประสิทธิภาพ
โครงสร้างนี้เหมาะสำหรับองค์กรที่มีระบบการทำงานซับซ้อนหรือมีการแบ่งหน้าที่การใช้งานที่ชัดเจน

- **Owner** : โดยแต่ละ EG (Entity Group) เป็นของ 1 Owner และ Owner สามารถที่จะเป็นได้ทั้ง Tanant และ customer แต่ละ customer สามารถมีได้ 1 Owner เท่านั้น ถ้า customer owner คือ Tanant ตัว customer owner จะมีสิทธิหรือเทียบเท่า customer ทันที และ customer owner เป็น customer อื่นๆ ตัว customer owner ก็คือ Sub-customer และนั้นสามารถเป็นได้หลายระดับ customer ใน ThingBoard

Section 3.02 Customer hierarchy

ThingBoard support การ recursive ของลำดับของ customer ได้ไม่จำกัดจำนวน สำหรับ Sub-customer โดยที่ Owner ระดับของ Root นั้นคือ Tanant โดยแต่ละ Owner อาจจะมีได้หลาย Entity Group (EGs), User Group (UGs), และ Customer Groups (CGs)

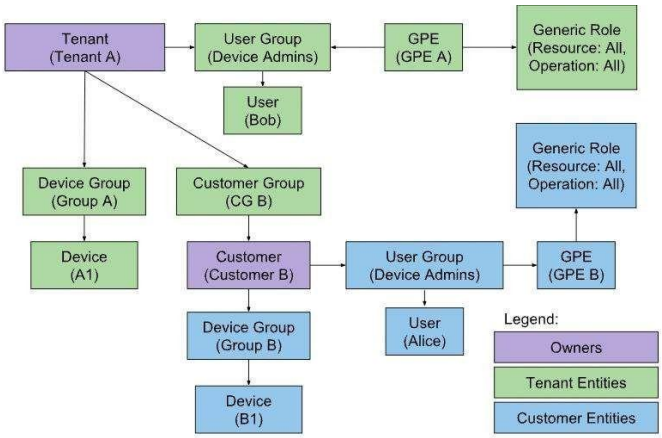
Note : แต่ละ Entity จะสามารถมีได้แค่ 1 Owner ยังไงก็ตาม Entity สามารถที่จะเป็นส่วนหนึ่งของหลาย EGs และเป็นส่วนหนึ่งของ Owner เดียวกันเนื่องจาก CGs สามารถที่จะมีได้หลาย customer โดยแต่ละ customer สามารถที่จะเป็น Owner ของ EGs, UGs, CGs



รูปที่ 144 relations between those entity

- Generic roles

โดยที่แต่ละ role ของ user group ที่เกี่ยวข้องมากกว่าหนึ่ง group แต่ละ user group จะมีได้แค่หนึ่ง owner เท่านั้น โดยที่ role ทั่วไปจะยอมให้ user group เดียวกันมีสิทธิทุก entities ที่เป็นของ owner เดียวกันและทุก Sub-customer Review ได้จากตัวอย่างข้างล่าง

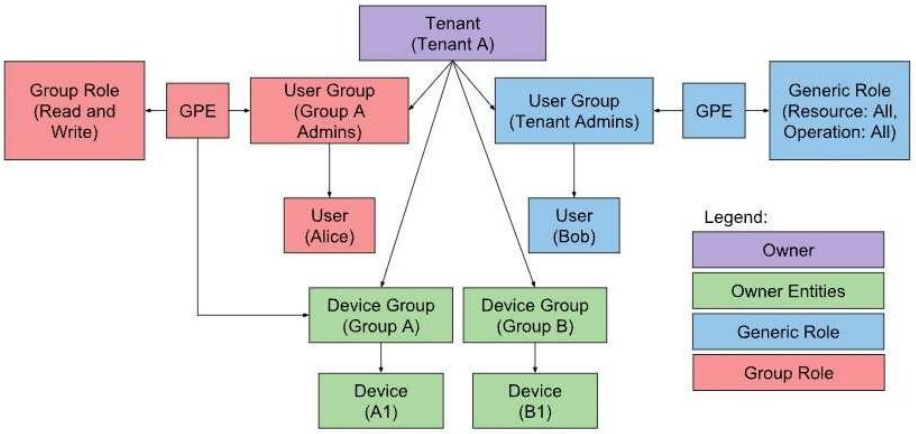


รูปที่ 145 Generic roles

User Bob สามารถที่จะ perform อะไรก็ได้กับทุก entity ที่เป็นของ Tanant A หรือ Customer B ตราบใดที่ Customer และ Sub-customer เป็นของ Tanant เดียวกัน ยังไงก็ดี User Alice จะสามารถ Prefrom อะไรก็ได้กับทุก entity ที่เป็นของ Customer B และทุก all sub-customer ทั้งหมด โดยที่ alice และ Bob สามารถที่จะเข้าถึง Device B1 แต่จะมีแค่ Bob เท่านั้นที่สามารถเข้าถึง Device A1

- Group roles

โดย Group roles สามารถที่จะ allow ให้ map ตั้ง permission สำหรับ specific user group ให้เฉพาะ entity group โดยใช้ special “connection” object โดย call group permission entity ในการสร้างการเชื่อมต่อ ระหว่าง User Group, Entity Group และ Group Role



รูปที่ 146 Group roles

โดยที่ User Bob อยู่ใน group “Tenant Admin” และสามารถที่จะทำอะไรก็ได้กับทุก tanant entities โดยทั่วไปแล้ว Bob จะสามารถ Control ได้ทั้งหมดที่อยู่ Device Group A และ Device Group B ในส่วน User Alice ที่อยู่ใน “Group A Admin” สามารถที่จะเข้าถึงการ reading/writing ทุก device ที่อยู่ใน Device group A ยังไงก็ตาม Alice ไม่สามารถที่จะเห็นหรือใช้งาน device จาก Device Group B Note: เนื่องจาก Entity Group สามารถมีได้แค่ 1 Owner ผู้ใช้สามารถ assign group role ให้กับทุก user group เพื่อที่จะเป็น owner เดียวกัน

Section 3.03 Example Use case

- **Smart Buildings: Separate user groups per facility**

ยกตัวอย่างว่าผู้ใช้กำลังจัดการในส่วนของอาคาร โดย Customer หลักต้องการที่จะให้สามารถที่จะตรวจสอบระบบ HVAC, ปริมาณการใช้ไฟฟ้า และ smart device อื่นๆในอาคาร และ Building Manager ต้องการที่จะ design ให้ share dashboard บางส่วนให้กับ End-user และ office worker และนอกจากนั้น Engineer ที่รับผิดชอบในการ maintenance สนใจที่จะ interest สถานะ device เช่น รับการแจ้งเตือนเมื่อระดับแบตเตอรี่ลดลงต่ำกว่าเกณฑ์ที่กำหนด

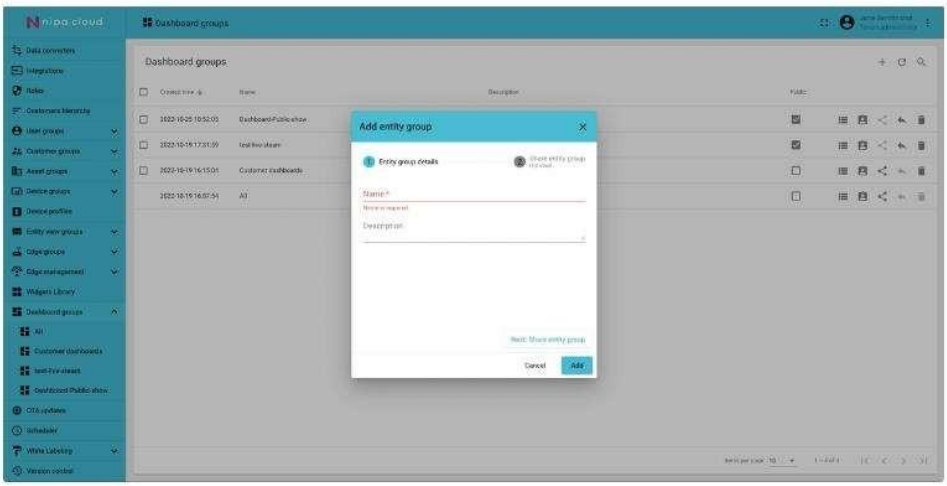
สรุป Requirement

- **Supervisors** :
  ต้องการที่จะอ่านค่าทุก device ที่อยู่ในอาคาร และสามารถที่จะสร้าง dashboard แบบกำหนดเองได้ แต่ไม่สามารถเข้าถึง dashboard ที่ถูกสร้างโดย usergroup ที่แตกต่างกัน
- **Facility Manager** :
  อนุญาตให้ Provisioning device ใหม่ สำหรับแต่ละ facility ได้ และกำหนดเกณฑ์ในการจัดการ user และกำหนด dashboard
- **End User** :
  อนุญาตให้อ่านได้สถานะของ facility ได้เท่านั้น ที่ user เป็น owner

Supervisors : โดยที่ผู้ใช้จะต้อง Create User Group แยกชื่อ “Supervisors” และ Dashboard Group “Supervisors Dashboard” โดยที่เป้าหมายคือ อนุญาตให้ Supervisors สามารถที่จะจัดการ Dashboard ใน group “Supervisors Dashboard” แต่ละ Entities อื่นในระบบ จะสามารถอ่านได้เท่านั้น

Step - 1 Create Dashboard

เริ่มจากที่ผู้ใช้เข้าไปที่ Dashboard Group และกดปุ่ม “+” จากนั้นตั้งชื่อ “Supervisors Dashboard” และกดปุ่ม “Add”

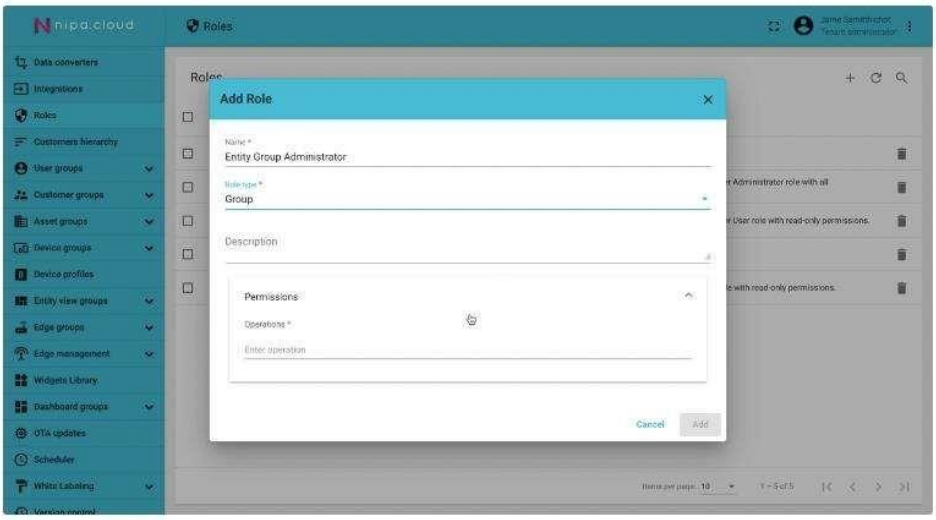


รูปที่ 147 Create Dashboard

Step - 2 Create 2 Role

โดย Role ที่ 1 “All Entities Read-only” สามารถที่จะอนุญาตเข้าถึงทุก entitiesเริ่มจาก select click “+” จากนั้นตั้งชื่อ “All Entities Read-only” เลือก Role type “Generic” และ Permission resource เลือกเป็น “All” และสำหรับ Operation “Read”, “Read Attibuted” และ “Read Telemetry” จากนั้น click “Add”โดย Role ที่ 2 “Entity Group Administrator” สามารถที่จะอนุญาตทุกอย่างที่มาจาก Group

เริ่มจาก select click “+” ตอไปตั้งชื่อ “Entity Group Administrator” เลือก role type “Group” และ permission สำหรับ operation “All” จากนั้น click “Add”

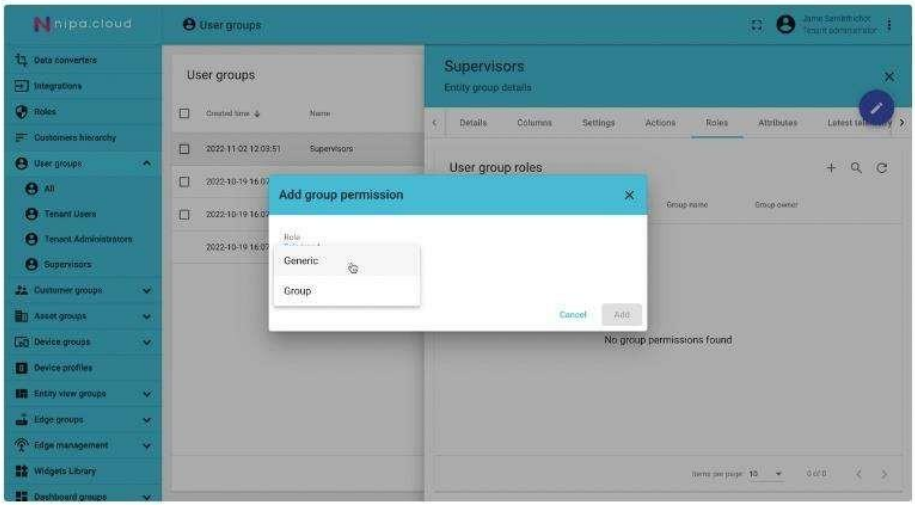


รูปที่ 148 Create 2 role

Step - 3 assign role to user group

ใน section group click “+” (add entity group)

จากนั้นตั้งชื่อ “Supervisors” และ click ปุ่ม “add” ก็จะเห็น “Supervisors Group” เพิ่มเข้ามา Click บน “Supervisors Group” 1 ครั้ง จะมี menu แสดงขึ้นมาหลังจากนั้น click รูป “Pen” และเลือก “Role” และ Click “+” เลือก Role type “Generic” และเลือก “All Entities Read-only” ในตัวเลือก click “add”click “+” อีกที ต่อไปเลือก role type “Group” และเลือก Role “Entity Group Administrator” เลือก type “Dashboard” และหลังจากนั้น เลือก “Supervisors Dashboard” ใน Entity Group

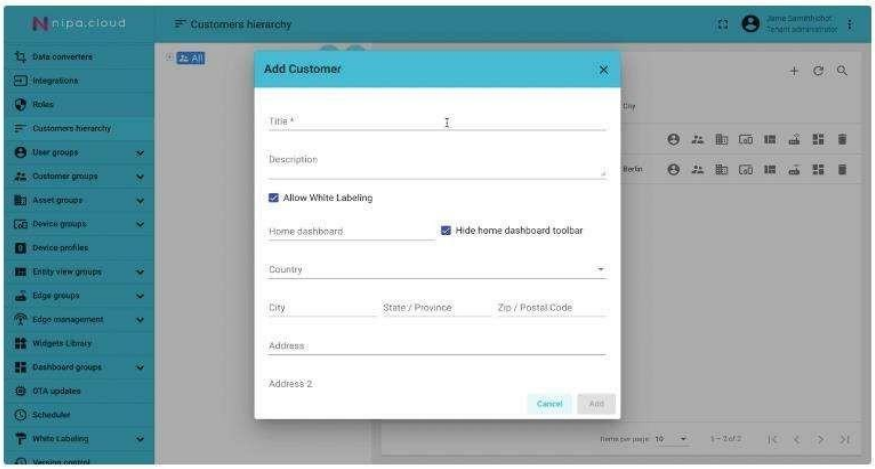


รูปที่ 149 assign role to user group

Facility Manager : โดยที่จะ create customer entity แยกสำหรับแต่ละอาคาร หรือ Group สำหรับอาคาร โดยจะเพิ่ม Facility Manage user account โดยที่ Default จะเป็น Group “Customer Admin” และสามารถที่จะ design dashboard, provision device, end user ได้

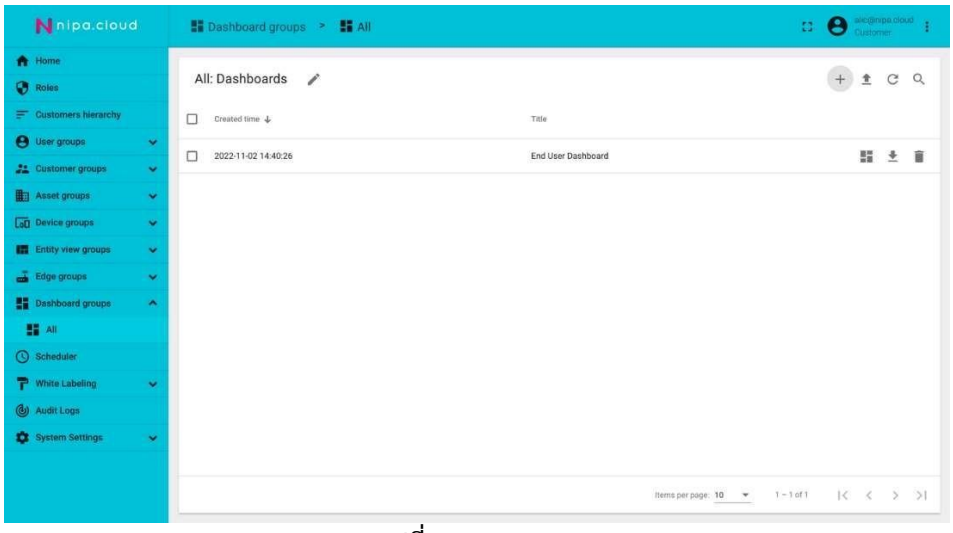
Step - 4 add Customer Hierarchy

ใน Customer Hierarchy click “+” จากนั้นในส่วน Title ตั้งชื่อ “Building A” แล้ว click “Add” หลังจาก click down-down menu icon “All” ไปที่ path Building àUser GroupsàCustomer Admin ต่อไป click “+” ที่ Path : “Customer Admin User” ต่อไปกำหนด email address สำหรับ instance โดยใช้ alic@nipa.cloud จากนั้น click “Add” จะแสดงหน้าต่าง Activation link จะมีข้อความขึ้นมา click “OK”



รูปที่ 150 Customer Hierarchy

End Users : จากนั้น login “Alice”, Admin Building A และ create ได้หลาย Dashboards เปิดที่หน้า Dashboard pang หลังจากนั้น click “+” เพื่อเลือก “Create new dashboard” ตั้งชื่อ ในส่วนของ dashboard “End User Dashboard”และ click “Add” เพื่อที่จะสร้าง Dashboards

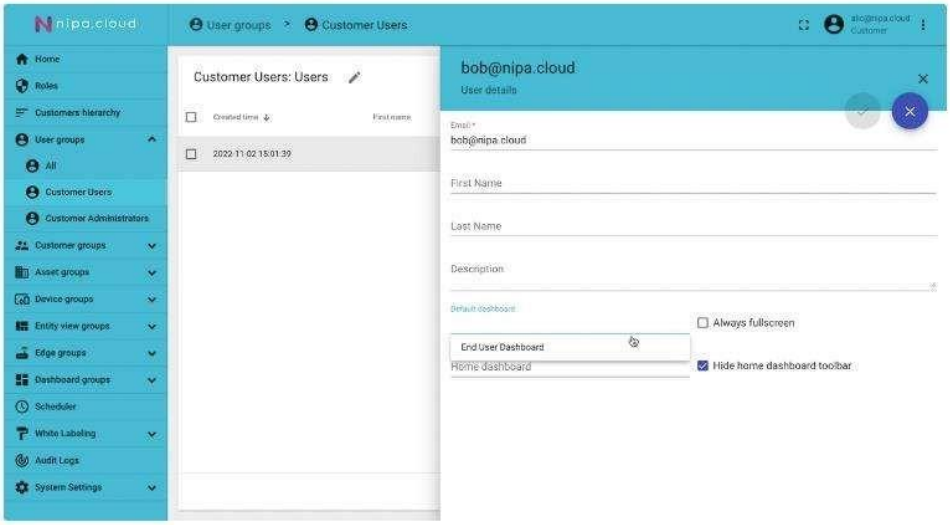


รูปที่ 151 Enduser

ตอนนี้จะเริ่ม Create read-only ให้ user สมมุติว่าผู้ใช้ต้องการที่จะกำหนด “End user dashboard” ให้กับ user ที่ login เข้าระบบโดย user สามารถ read-only โดยไม่สามารถเข้าถึง admin menu

โดยเลือก Customer user ใน user group หลังจากนั้น click “+” และโดยที่ Email address ที่ใช้ test เป็น bob@nipa.cloud click “add” จะมีหน้า window user activation link ขึ้นมา click “OK”

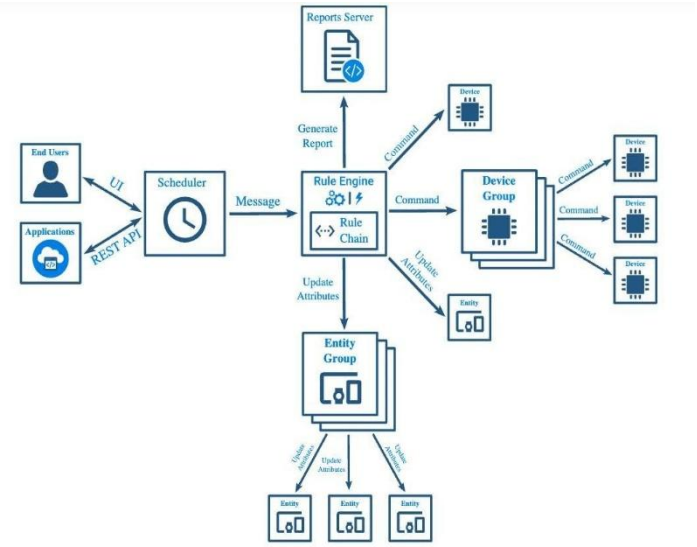
click user ที่ create แล้ว check กล่อง “Always full screen” และเลือก “End User Dashboard” ใน menu เลือก Dashboard



รูปที่ 152 Create Read-only ให้ User

Section 3.04 Scheduler

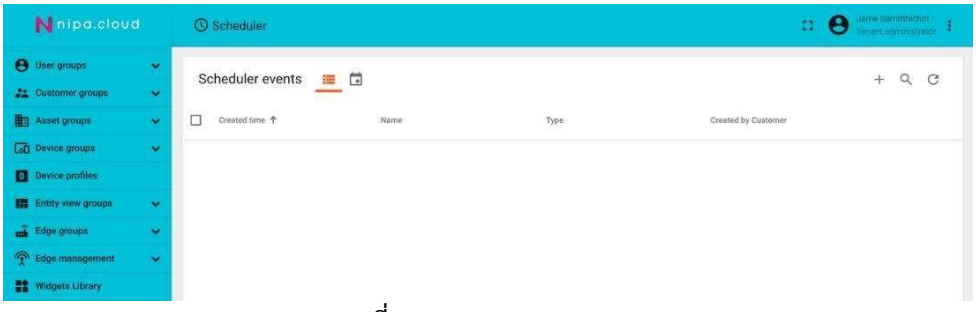
ThingBoard ช่วยให้คุณจัดกำหนดการกิจกรรม type ต่างๆ ด้วยการ schedule configuration ที่ Flexible ThingBoard Schedule เป็นตัวกำหนดตารางเวลาที่ schedule configuration เมื่อมีเหตุการณ์ตัว scheduler event จะเริ่มต้นการทำงาน และ Rule Engine Message จะถูก generate จาก event ที่ config ไว้คล้ายกับ Rule Engine Message และ Message ที่ถูก Generate จะถูก forward ไปที่ Rule Engine และ Process ต่อโดยเริ่มจาก Root Rule Chain



รูปที่ 153 Schedule

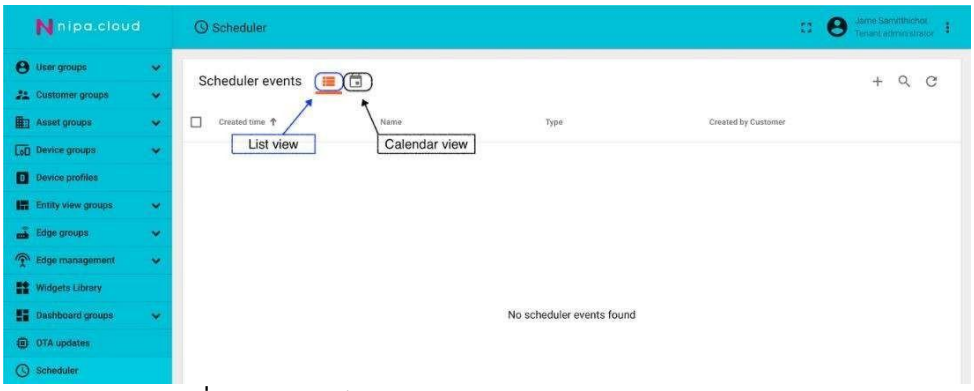
Scheduler Administration

โดยที่ Tanant Admin และ Customer User สามารถที่จะ config Scheduler event ใน ThingBoard



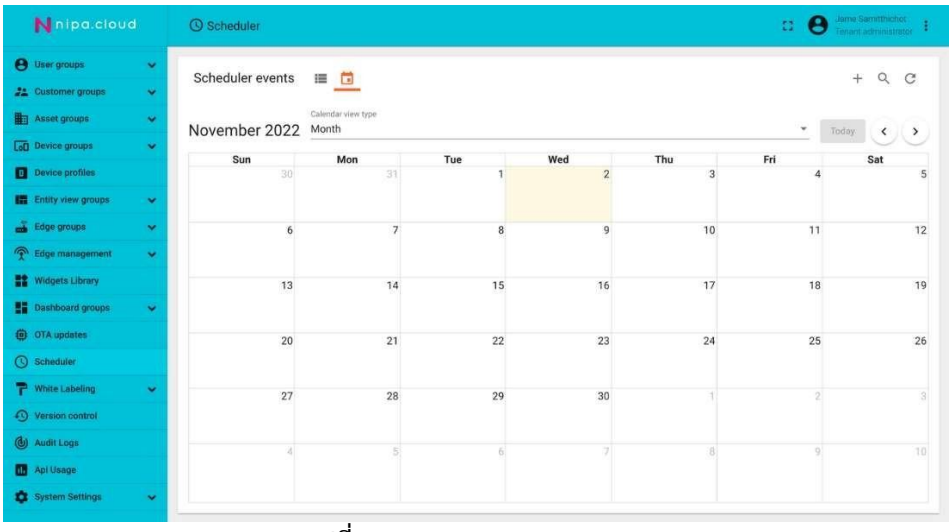
รูปที่ 154 config scheduler

โดยที่หน้า Scheduler Events อนุญาตที่จะทำ update, add หรือ delete scheduler event และ โดยที่หน้า Scheduler Events สามารถที่จะนำเสนอได้ใน 2 mode list view หรือ Calendar view สามารถเลือกสลับดูได้ที่รูปด้านล่าง



รูปที่ 155 นำเสนอได้ใน 2 mode list view หรือ Calendar view

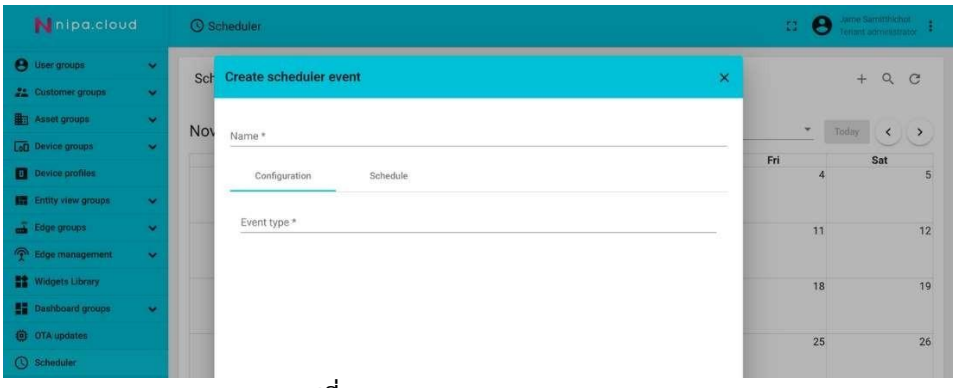
โดยที่ใน Mode Calendar View Scheduler Events ของตัวกำหนดตารางแสดงเป็น Lables กำกับตาม Schedule



รูปที่ 156 Mode Calendar View

โดย default calendar ที่แสดงเป็น type month โดยที่ calendar view type Dropdown ช่วยให้สลับไปใช้ view type อื่นๆได้ สามารถเลือก view type ได้

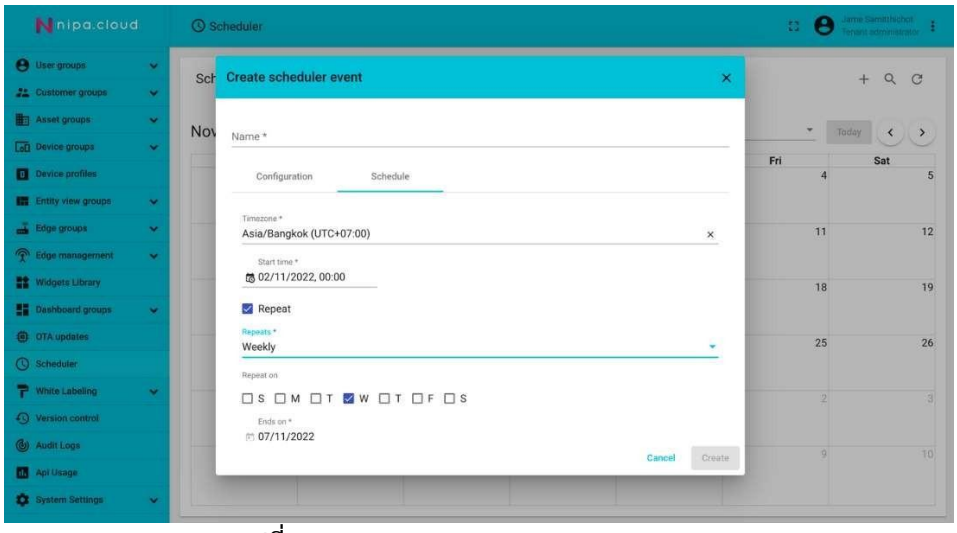
Month / Week / Day / List Year / List Month / List Week / List Day / Agenda Week / Agenda Day โดยสามารถที่จะ create scheduler event ใหม่ได้โดย click “+” ที่มุมขวาบนการ Edit Scheduler Event Dialog จะประกอบด้วย 2 รูปแบบ Configuration และ Schedule



รูปที่ 157 Edit Scheduler Event Dialog

Configuration การกำหนดค่าอนุญาตให้ Set Event Type และ Parameters การ Configuration Event สามารถเลือก Event type ได้

Schedule การกำหนดการอนุญาตให้ set Event Schedule Configuration ได้



รูปที่ 158 set Event Schedule Configuration

Schedule Form มี Parameters ตามนี้

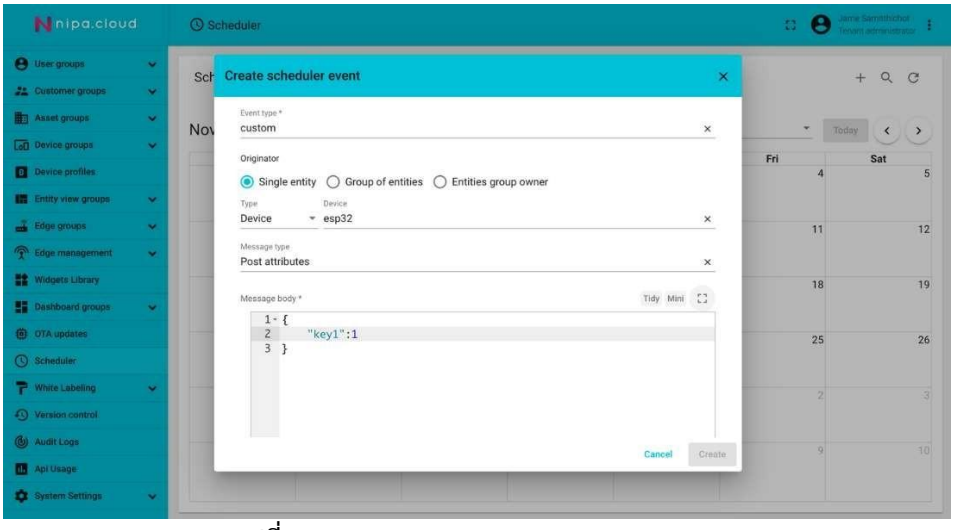
- **Timezone** – Timezone ที่กำหนดใน Schedule event จะถูก processed
- **Star Date/Time** – date/time ที่กำหนดว่าเมื่อไหร่ Schedule event เริ่มทำงาน
- **Repeat** - กำหนด Schedule event ที่ทำช่วง 1 ช่วงเวลา หรือ ทำซ้ำ
- **Repeats** – Repeat rule สามารถที่จะทำเป็น Daily หรือ Weekly
- **Repeat on** – สามารถใช้ได้กับ Weekly Repeat rule โดย Specifies weekday ให้เมื่อไหร่ที่ Schedule event เริ่มทำงาน
- **Ends on** - วันที่สิ้นสุด Schedule event

Schedule Event Types

ในการ Configuration Event Type สามารถ Select event ที่มีอยู่ หรือ specified type ได้เอง

Custom type

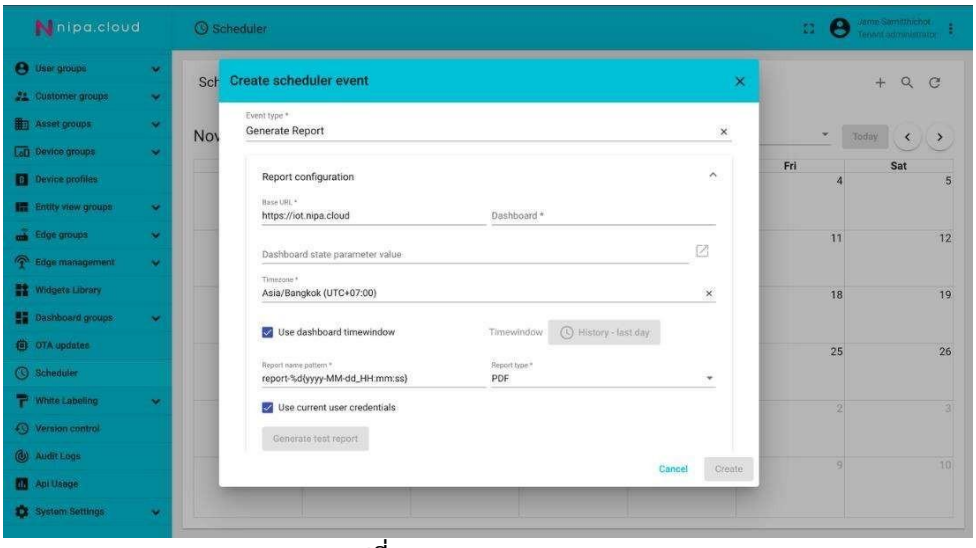
โดยที่ Custom type ใช้ default scheduler event configuration ตาม message structure



รูปที่ 159 default scheduler event configuration

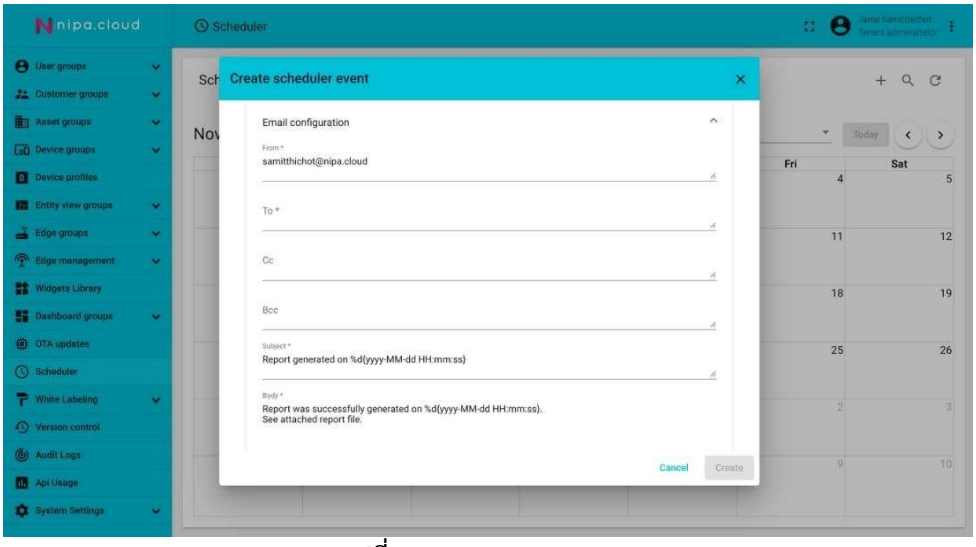
- **Originator** - โดยที่ message originator สามารถที่จะเป็น Entity ได้ เช่น Device, Asset etc. หรือ Group of Entities ถ้าไม่ระบุ scheduler event entity มันจะถูกเป็นตัวเริ่ม
- **Message Type** - message type ขึ้นอยู่กับ Rule engine message type สามารถที่จะเป็น existing message type หรือ customer ถ้าไม่มีการ specific scheduler event type จะถือว่าเป็น message type
- **Message body** - โดยที่ Message Body จะใช้ในรูปแบบของ JSON
- **Metadata** - โดยรูปแบบของ Message Metadata จะเป็นในรูปแบบ key/value

Generate Report



รูปที่ 160 Generate Report

อนุญาตให้ schedule สร้าง report ได้โดย feature ที่ support ก็คือ reporting feature



รูปที่ 161 Generate Report

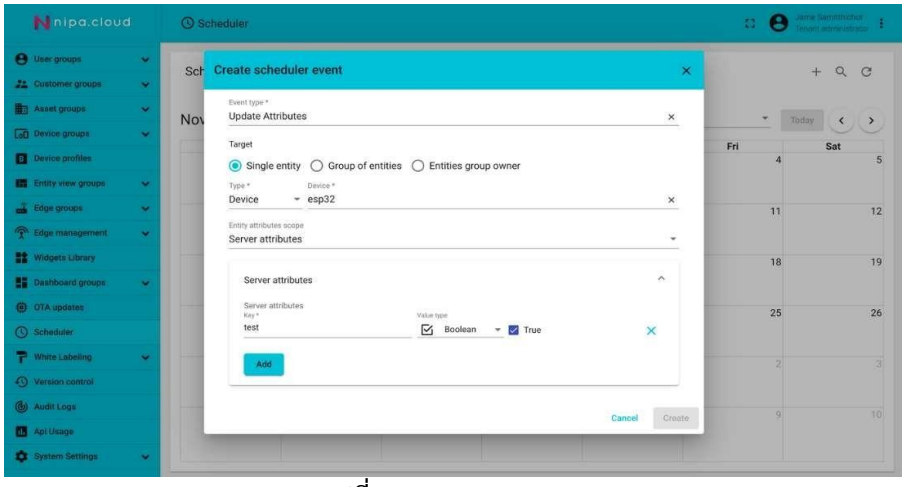
- **Report Configuration**
  - **Base URL** - โดย base URL ของ ThingBoard UI ควรที่จะเข้าถึงได้จาก Report servers
  - **Dashboard** - โดยที่ Dashboard นั้นจะใช้สำหรับ Genaration Report
  - **Dashboard state parameter value** - ใช้ในการ specify target ของสถานะ Dashboard สำหรับ generation report สามารถที่จะ set ตั้งเป็น automation ได้
  - **Timezone** – เป็น Timezone ใน target dashboard ที่ใช้ present ในการ Report

- **Use dashboard timewindow** - ถ้าตั้ง Configured ไว้ที่ timewindow ใน dashboard ในส่วนของ target ที่ report จะถูก generate ในระหว่างนั้น
- **Timewindow** - ในการ specific dashboard timewindow ที่จะใช้ในระหว่างการ generate report
- **Report name pattern** – โดยจะเป็น Patten ของชื่อ file ที่สำหรับ generate report
- **Report type** – ในส่วนนี้จะเป็นการกำหนด type ของ file report โดยมี PDF | PNG | JPEG
- **Use current user credentials** - ถ้ามีการกำหนด credentials สำหรับ user ที่ create มันจะทำการไป configuration ที่กำหนด และ file ที่ถูก process แล้วจะถูก download โดย Auto ถ้า generate report เสร็จ

- **Sent email** – ถ้า set email message ไว้ตอนที่ส่ง email จะ attachment report ไปด้วย
- **Email configuration**
  - **From** – from address
  - **To** – comma และ address list ของ recipients
  - **Cc** – comma และ address list
  - **Bcc** – comma และ address list
  - **Subject** – mail subject สามารถที่จะเพิ่ม date-time ได้จาก pattern นี้ %d{date-timepattern}
  - **Body** – mail body สามารถที่จะเพิ่ม date-time ได้จาก pattern นี้ %d{date-timepattern}

Update Attributes

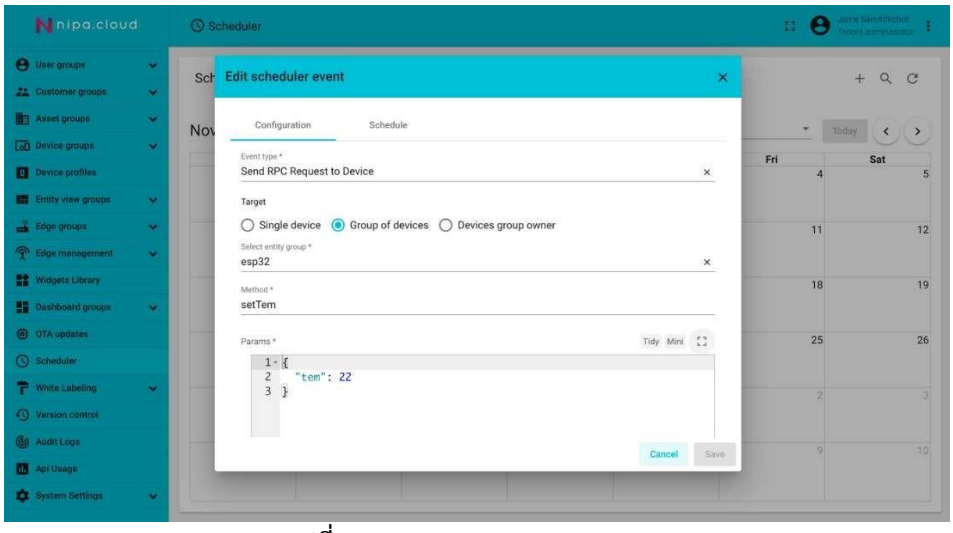
อนุญาตให้ schrdule update ของในส่วน attribute สำหรับ entity หรือ entity group



รูปที่ 162 Update Attributes

- **Target** – คือ target entity ของ attribute ที่ควรจะ update และสามารถที่จะเป็นแบบหนึ่ง entity (ex. Device, Asset etc.) หรือ entity group
- **Entity attribute scope** - คือ Scope ที่ update attribute สามารถเลือก Specified ใน target ที่ device entity ได้ โดยที่สามารถเลือกได้ทั้ง Server attribute และ Share attribute
- **Server / Share attribute** - ในตาราง Key/Value เป็น representing attribute ที่ไป update values

Send RPC Request to Device
อนุญาตที่จะให้ schrdule command (RPC call) ไปที่ device หรือ group สำหรับ device

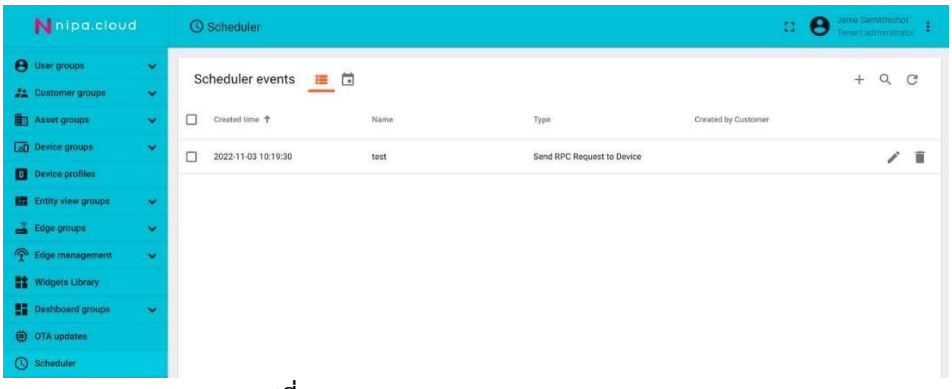


รูปที่ 163 Send RPC Request to Device

- **Target** – Target device ที่จะถูกส่งสามารถส่งได้ทั้งที่เป็น Single device หรือ group
- **Method** – วิธีที่จะใช้ Call RPC
- **Params** - เป็น Params Call RPC ที่เป็นรูปแบบ JSON

Schrduler Widget

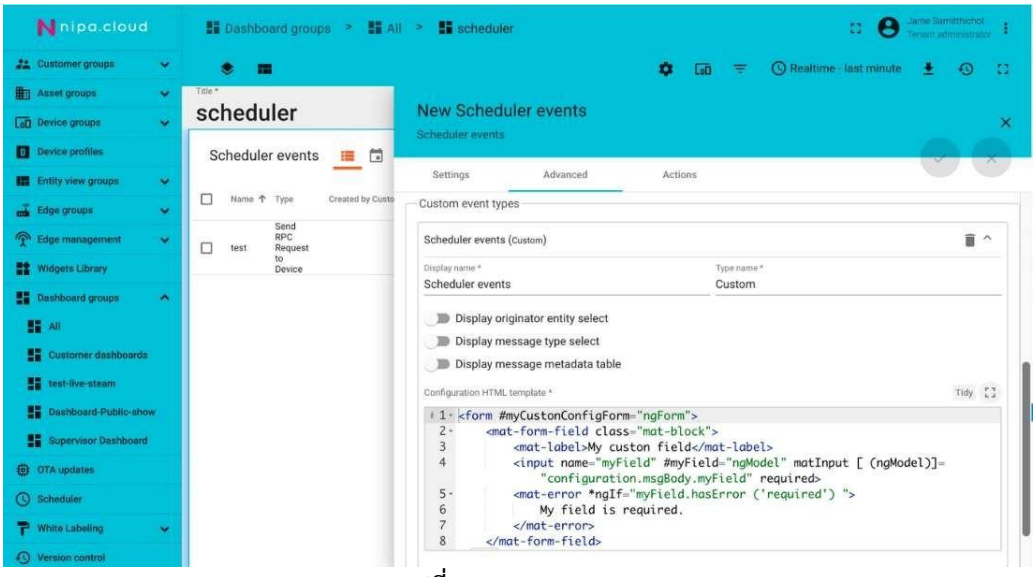
ThingBoard จะเป็น provide ability ในการ manage schrduler event ด้วย Schrduler event หรือ Report Schrduler Widgets ซึ่งเป็นส่วนหนึ่งของ Schrduling Widgets Bundle



รูปที่ 164 Cheduling Widgets Bundle

ในส่วนของ Schrduler event widget มีเหมือนกับ Schrduler event page และนอกจากนั้นสามารถที่จะ customized กับค่าที่กำหนดไว้ สำหรับ custom scheduler event type และ ตัวอย่างข้างล่างเป็น Config สำเร็จ สำหรับ Custom event type ใน advance ของ widget config

```
<form #myCustonConfigForm=“ngForm”>
  <mat-form-field class=“mat-block”>
    <mat-label>My custom field</mat-label>
    <input name=“myField” #myField=“ngMdel” matInput [ (ngModel)]>=
<“Configuration.msgBody.myField” required>
    <mat-error *nglf=“myField.hasError (‘required’)”>
    My field is required.
    </mat-error>
  </mat-form-field>
</form>
```



รูปที่ 165 HTLM template

ประวัติบุคลากรในโครงการ

ชื่อ – นามสกุล : ดร.อภิศักดิ์ จุลยา

ตำแหน่งงานปัจจุบัน : Chief Executive Officer & Founder, Nipa.Cloud

ตำแหน่งในโครงการ : Project Owner

สถานที่ทำงาน : บริษัท นิภา เทคโนโลยี จำกัด เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

1983 - 1987	:	Cleveland State University, Doctor of Engineer
1981 - 1982	:	Youngstown State University, Master Degree in Engineering
1976 - 1980	:	Chulalongkorn University, Bachelor of Science in Engineering

ประสบการณ์การทำงาน :

1996 – Present	:	CEO and Founder’s, Nipa Technology Co., Ltd.
2002 – 2006	:	Chairman, Board of Directors, Asia & Pacific Internet Association (APIA)
2002 – 2003	:	Executive Committee Office of Science and Technology (NSTDA)
1999 – 2000	:	Director, Thailand Science Park, NSTDA
1988 – 1995	:	Senior Research Scientist, NASA John H. Glenn Research Center, Cleveland, Ohio USA

ความเชี่ยวชาญเฉพาะทาง : Cloud Technology researcher and Digital transformation specialist.

ผลงานที่ผ่านมา :

22 research publications published in refereed journals and articles globally

ชื่อ – นามสกุล : ฐิติวัชร์ นันทนิตติ
ตำแหน่งงานปัจจุบัน : General Manager, Cloud
ตำแหน่งในโครงการ : Project Manager

โทรสาร : -

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2003 - 2010	:	Bachelor of Business Administration, Program in Information Systems Rajamangala University of Technology Krungthep

ประสบการณ์การทำงาน :

2015 – Present	:	*General Manager* - NIPA Technology Co., Ltd.
2013 - 2015	:	*IT Teacher* - Phetkasem Management Technological College
2004 - 2007	:	*Lecturer* – D Computer Co., Ltd.

ผลงานที่ผ่านมา :

2020 - Present	:	ความรู้ความชำนาญในกลุ่มผลิตภัณฑ์ Intel® Data Center Solution เครื่องแม่ข่าย (Intel® Server System Data Center Block) หน่วยประมวล (Intel® Processor) หน่วยความจำ (Intel® Optane Memory) หน่วยจัดเก็บข้อมูล (Intel® SSD) หน่วยรับส่งข้อมูล (Intel® Network Card) และซอฟแวร์บริหารจัดการในกลุ่มผลิตภัณฑ์ของอินเทล

ชื่อ – นามสกุล : ณัชชา พูนศรีธนากูล
ตำแหน่งงานปัจจุบัน : Test Lead QA
ตำแหน่งในโครงการ : Software Developer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2004 - 2007	:	Bachelor of Information Technology, Suranaree University of Technology

ประสบการณ์การทำงาน :

Nov 2020 - Present	:	Test Lead QA - NIPA Technology Co., Ltd.
Jul 2020 – Oct 2020	:	Test Manager - Buzzebees
Oct 2019 – Jun 2020	:	Senior QA – Azure Computer Thailand Co.,Ltd.
Nov 2018 -Sep 2019	:	Lead Software Tester – TP Talent Company
Aug 2013 – Apr 2017	:	Software Tester – SoftSQ Company Limited
Sep 2010 – Aug 2013	:	QA Engineer – NetSolution Source Force
Dec 2009 – Jul 2010	:	Test Website Engineer – White Sparkle Jewerly Thai
Apr – Nov 2009	:	Software Tester – Aware Corporation Limited
Sep 2008 – Feb 2009	:	Software Tester & Support – Puumsoft company
Feb– Sep 2008	:	Software Tester – SDL International Co.,Ltd.

ชื่อ – นามสกุล : พิพิธพนธ์ นวลงาม
ตำแหน่งงานปัจจุบัน : Software Developer & System Analysis
ตำแหน่งในโครงการ : Software Developer

โทรสาร : -

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2014 - 2018 : Bachelor of Science, Program in Information Technology
King Mongkut's University of Technology Thonburi

ประสบการณ์การทำงาน :

2019 - Present : *Software Developer & System Analyst* – NIPA Technology Co., Ltd.
2018 - 2019 : *Software Developer* - NIPA Technology Co., Ltd.

ความเชี่ยวชาญเฉพาะทาง :

Development skill

- NodeJS Programming
- Basic Go Programming
- Java Programming
- JSP Web Programming
- Basic Programming with Cobol
- Web Automate Testing with Robot Framework and Selenium2Library
- Worked with Agile environment

ชื่อ – นามสกุล : ชยพล ลิมานนท์
ตำแหน่งงานปัจจุบัน : Software Architect
ตำแหน่งในโครงการ : Software Developer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2015 - 2019 : Bachelor of Science, Program in Computer Science
King Mongkut's Institute of Technology Ladkrabang

ประสบการณ์การทำงาน :

2019 - Present : *Software Architect* - NIPA Technology Co., Ltd.
2018 - 2019 : *Software Developer* - NIPA Technology Co., Ltd.

ความเชี่ยวชาญเฉพาะทาง :

- Agile Software Development
- Software Architecture Design & Patterns
- Software Design Patterns
- Clean Code Architecture
- Cloud Computing
- DevOps
- ASP.NET Core
- TypeScript, JavaScript
- Golang

ชื่อ – นามสกุล : ชาญศิลป์ ชิ้นประเสริฐ

ตำแหน่งงานปัจจุบัน : Chief Technology Officer (CTO)

ตำแหน่งในโครงการ : Solution Architecture Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2016 - Present	:	Master of Science, Program in Information Technology Management Mahidol University
2009 - 2014	:	Bachelor of Science, Program in Computer Science King Mongkut's Institute of Technology Ladkrabang

ประสบการณ์การทำงาน :

Jan 2019 - Present	:	*Chief Technology Officer (CTO)* - NIPA Technology Co., Ltd.
2016 – Dec 2018	:	*Cloud Architect Specialist* – NIPA Technology Co., Ltd.
2016 – Dec 2018	:	*System Application Programmer* – Billme (Thailand) Co., Ltd.
Apr – Mar 2016	:	*Co-Founder & Development Team Leader* – ODEO Solution Co., Ltd.
Dec – Mart 2014	:	*System Engineer* – True Digital Content and Media Co., Ltd.
Apr – Nov 2012	:	*Enhanced System Administrator skill and Ruby Programming* – TARAD Dot Com Co., Ltd.

ใบอนุญาต / ทะเบียน :

2018 - 2020	:	*Certified Kubernetes Administrator* – Cloud Native Computing Foundation Certificate ID Number CKA-1800-0932-0100
2017 -2020	:	*Certified OpenStack Administrator* – OpenStack Foundation Certificate ID Number COA-1700-0364-0100

System Administrator skills

- Specialist of OpenStack, cloud computing technology
- Specialist of Ceph, distributed storage system
- Specialist of Zabbix, enterprise network management system
- Linux Specialist (Debian, Ubuntu and Red Hat)
- Web application and System Security
- VMware vSphere (ESX, ESXI and vCenter)

- MySQL Multi-Master with Galera
- Automate tasks & deployment with Ansible
- Check Point, Juniper, SonicWall, Palo Alto and Open Source Firewall
- F5 Global Traffic Manager and F5Load Balance
- IT Technical consult

Development skills

- Specialist of PHP framework for Laravel, CodeIgnitor and Yii
- Frontend Programming (Vuejs, jQuery, Javascript, CSS)
- Web Automate testing with Ruby and Watir
- Basic programming with Ruby, JavaSE, Python, Scala and Shell Script
- Worked with Agile environment
- Experience with Git

ผลงานที่ผ่านมา :

2020 – Present : ความรู้ความชำนาญในกลุ่มผลิตภัณฑ์ Intel® Data Center Solution เครื่องแม่ข่าย (Intel® Server System Data Center Block) หน่วยประมวล (Intel® Processor) หน่วยความจำ (Intel® Optane Memory) หน่วยจัดเก็บข้อมูล (Intel® SSD) หน่วยรับส่งข้อมูล (Intel® Network Card) และซอฟแวร์บริหารจัดการในกลุ่มผลิตภัณฑ์ของอินเทล

ชื่อ – นามสกุล : คมกฤช เวียงวิเศษ
ตำแหน่งงานปัจจุบัน : Principal Engineer
ตำแหน่งในโครงการ : OpenStack Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2009 - 2013	:	Bachelor of Science, Program in Computer Science King Mongkut's Institute of Technology Ladkrabang

ประสบการณ์การทำงาน :

2016 - Present	:	*Principal Engineer, Cloud Innovator* - NIPA Technology Co., Ltd.
Nov 2013 - Present	:	Research new feature of OpenStack
Nov 2013 – May 2014	:	*Teacher Assistant Administrator Course* – Mahidol University and IT Bakery
Jun – Nov 2013	:	*System Engineer "Clusterkit"* – Clusterkit Training Center and True IDC

ใบอนุญาต / ทะเบียน :

Jan 20, 2017	:	Mirantis Certified Administrator for OpenStack (MCA200) Mirantis License 200-089-673
Aug 2015 – Aug 2018	:	Red Hat Certified System Administrator Res Hat License 140-175-498
Sep 2014 – Oct 2017	:	Red Hat Certified System Administrator in Red Hat OpenStack Red Hat License 140-175-498

นักวิจัยด้าน : OpenStack
ความเชี่ยวชาญเฉพาะทาง : OpenStack

ชื่อ – นามสกุล : ประทิน บุญรอด
ตำแหน่งงานปัจจุบัน : System Engineer
ตำแหน่งในโครงการ : OpenStack Engineer

โทรสาร : -

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2015 - 2018 : Bachelor of Engineering, Program in Computer Engineering
Kasetsart University

ประสบการณ์การทำงาน :

2019 - Present : *System Engineer* - NIPA Technology Co., Ltd.
2018 : *Internship (Network Operation & Maintenance)* – Triple T Broadband, PLC.

ความเชี่ยวชาญเฉพาะทาง :

Development skills

- Manage code with GIT version control.
- Develop application with C#, java, PHP python language with backend MySQL, Mongo.
- Load test webserver with JMeter.

System Engineer skills

- Install, manage and tunning Galera MariaDB cluster.
- Install and manage Ceph storage cluster.
- Install, manage, and troubleshooting problem in OpenStack cluster.
- Install and manage Logstash-Elasticsearch-Kibana cluster.
- Install and manage Kubernetes cluster.
- Install and manage High available Load balancer using Nginx
- Written Script with BASH, Ansible, Python.
- Basic knowledge and troubleshooting for network

Senior Project

- System of Infrastructure as a service for Department of Computer Engineering (CPE Cloud)

ชื่อ – นามสกุล : ปัณณวิชญ์ บุรินทร์ทรัพย์ฤดี
ตำแหน่งงานปัจจุบัน : Senior System Engineer
ตำแหน่งในโครงการ : Network Engineer

โทรสาร : -

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2009 – 2012	:	Bachelor of Information and Technology Engineering Mahanakorn University of Technology
2007 - 2009	:	Diploma in Industrial Electronics Thatum Industrial and Community Education College

ประสบการณ์การทำงาน :

2015 - Present : Network & System Engineer - NIPA Technology Co., Ltd.

ความเชี่ยวชาญเฉพาะทาง :

System Administrator skills

- Designed and configuration core network of Data Center
- Traffic Engineering via BGP
- Expertise in Data Center infrastructure and Network Operation services, providing to the customers in colocation.
- Designed and implemented Datacenter products such as Cisco, Arista, Brocade, HP
- Designed and modified the electrical blueprint of DC

ชื่อ – นามสกุล : นายจีรพัฒน์ ศุภอภินันต์

ตำแหน่งงานปัจจุบัน : Lead System Engineer

ตำแหน่งในโครงการ : Network Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2006 - 2008	:	Bachelor of Engineering, Program in Computer Engineering King Mongkut's Institute of Technology Ladkrabang

ประสบการณ์การทำงาน :

2020 – Present	:	*Senior Network Engineer* - NIPA Technology Co., Ltd.
2017 – 2020	:	*Senior Network Engineer* – Digital Government Development Agency (DGA)
2016 - 2017	:	*Senior Network Engineer* – MFEC (Cisco Gold Partner)
2013 - 2016	:	*Network Engineer* – Electronic Government Agency
2012 - 2013	:	*Network Engineer* - Netbright Internet Solution Co., Ltd.
2011 – 2012	:	*Network Engineer* – United Information Highway
2009 - 2010	:	*Network Engineer* – Unified Communication Co., Ltd.

นักวิจัยด้าน :

- Software Define Network: OpenSDN, ONOS, OpenDaylight.
- Cloud Computing: Openstack, Google cloud platform.

ความเชี่ยวชาญเฉพาะทาง :

- IPv4 Routing: RIP, OSPF, EIGRP, IS-IS, BGPv4 (RR and Confed), MP-BGP
- Pv6 Routing: RIPng, OSPFv3, IS-ISv6, MP-BGP, Dual Stack IPv4/IPv6, IPv6 Tunneling
- Multicast: PIM-DM, PIM-SM, PIM-Bidir, SSM, MSDP, Anycast RP
- Ethernet Switching: STP, MSTP, 802.1Q, 802.1x, VTP, LACP, SPAN, HSRP/VRRP, VSS, QoS, Open vSwitch, Linux Bridge
- MPLS: LDP, RSVP, L3VPN (VRF), VPLS, H-VPLS, MPLS-TE, Fast Reroute, QoS, Multicast VPN, 6PE, 6VPE

- Carrier/Metro Ethernet: QinQ, EVC, Bridge Domain, REP, G.8032 (Ethernet Ring Protection), 802.3ad, 802.3ag, MEF services – VLL (EINE), VPLS (ELAN), IP-VRF, VRF-Lite
- Data Center: L2/L3, Port Channel, VDC, VPC & VPC+, FEX, MLAG, FabricPath, VXLAN, OTV, LISP, BGP-EVPN, OVSDB, Palo Alto NGFW, Load Balancer, VMware, NFV/AFV, Nexus 9K/7K/5K/2K, UCS Rack/Blade Server (B & C Series), SAN Switch
- Software Defined Networking (SDN): VMware NSX, Cisco ACI (APIC + Nexus 9000), SDN Controller, 3rd Party Integration (Palo Alto Firewall, F5 Load Balancer, VMware vSwitch), Arista CVX, OpenStack Neutron Network, OpenFlow, OpenDaylight, Mininet
- Network Function Virtualization: vRouter (namespace), vSwitch (openvswitch, linux bridge), FWaaS (iptables), LBaaS
- Storage Area Network (SAN): FC, FCIP, FCoE, Multihop FCoE, iSCSI, SAN OS
- DWDM: Sponder, Transponder, Xponder, Mux/Demux, ROADM
- Security: Firewall, IDS/IPS, GRE, IPSEC, DMVPN, GET VPN, SSL, AAA (RADIUS, TACACS+), Linux Firewall (iptables)
- Network Operating System: IOS, IOS-XE, IOS-XR, NX-OS, JUNOS, Arista EOS, Cumulus, Pica8, OVS
- Operating System: CentOS, Ubuntu, Window Server
- Cloud: OpenStack community, OpenStack commercial (Huawei)
- Hypervisor: KVM, QEMU, VMware
- Programming: Python and C
- Automate Tool: Ansible, Pupet
- Sniffer Tool: Wireshark, TCPDUMP

ผลงานที่ผ่านมา :

Advanced Info Services Plc (AIS)

- Implement and deploy VXLAN with MP-BGP.
- Implement and deploy OTV

Sukhothai Thammathirat Open University

- Implement and Replacement network infrastructure and data center (Nexus 7K, Cisco 6504, Cisco 3750 and Cisco 2960)
- Implement Cisco UCS B-Serie Solution
- Implement VMWare solution
- Implement SAN Storage solution (IBM V7000)

PTT Rayong Site

- Implement Nexus 5K and 2K (vPC, FEX, FCoE, SAN) Solution

Muangthai Life Assurace

- Implement Cisco UCS Server B-Serie Solution
- Implement VMWare solution (vCenter)

Intellectual property

- Implement and replace the network infrastructure (Cisco 6508, Cisco 3750 and Firewall ASA5540).
- Implement Cisco UCS B-Serie Solution

The Prince Royal's College

- Implement and replace the network infrastructure (Cisco 3850, Cisco 2960 and Firewall (Fortigate).

ชื่อ – นามสกุล : ณัฐพร พูลสุขเสริม
ตำแหน่งงานปัจจุบัน : Operation Manager
ตำแหน่งในโครงการ : Operation Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2014 – 2017	:	Bachelor of Science, Program in Information and Communication Technology) Sripatum University
2012 – 2014	:	Computer/Information Technology Administration and Management SIBA College High Vocational Certificate of Computer Applications (M.C.A.)
2009 – 2012	:	Computer/Information Technology Administration and Management SIBA College Vocational Certificate of Computer Applications (M.C.A.)

ประสบการณ์การทำงาน :

2018 – Present	:	Operation Manager - NIPA Technology Co., Ltd.
2017 – 2018	:	Lead IT Infrastructure – Bluefinix Digital Co., Ltd.
2015 – 2017	:	Microsoft Student Partner – Business Lead, Microsoft
2014 - 2015	:	Live Broadcasting on Internet – MaxLive TV
Mar – Jun 2013	:	Internship – PROEN Internet
2011 - 2013	:	System Administrator – SIBA College

ความเชี่ยวชาญเฉพาะทาง :

System Administrator skills

- Technology Infrastructure with working and operations public cloud Thailand
- Specialist of OpenStack, cloud computing technology
- Specialist of Ceph, distributed storage system
- Specialist of Zabbix, enterprise network management system
- Specialist of Proxmox, Cluster, OpenShift, Kubernetes, Docker, Nginx, HAProxy, MariaDB Galera, Consul, Prometheus, Grafana, Ceph

Development skills

- Specialist of DevOps automate CI/CD use tools Team City, Gitlab, alter developer activity to slack and Strong knowledge Cloud computing and DevOps.

ผลงานที่ผ่านมา : -

ชื่อ – นามสกุล : เปมิกา โรจนรัตน์
ตำแหน่งงานปัจจุบัน : Senior System Engineer
ตำแหน่งในโครงการ : Operation Engineer

: -

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2014 - 2017	:	Master of Science, Program in Information Technology King Mongkut's Institute of Technology Ladkrabang
2007 - 2011	:	Bachelor of Science, Program in Information Technology King Mongkut's Institute of Technology Ladkrabang

ประสบการณ์การทำงาน :

2020 - Present	:	*Senior System Engineer* - NIPA Technology Co., LTD.
2017 - 2020	:	*System Engineer (Outsource)* – Advance Info Service, PLC.
2012 -2017	:	*System Administrator* – Yip In Tsoi & Co., Ltd.

นักวิจัยด้าน : System Engineer

ความเชี่ยวชาญเฉพาะทาง :

- Operating System: Linux & Unix, Windows Server
- Cloud Openstack
- Install and Setup Software for Server Service support
- MongoDB replication
- Mysql Master-slave
- PostgreSQL Master-slave and set Failover
- Cloud Support and Consulting Customer
- Shell Script
- Setting and Renew SSL Certification for Customer Support
- Setting Zabbix monitoring

- Ansible Install Software
- Migrate Cloud for Customer Support
- Setting Firewall Connection
- Database: Oracle, PostgreSQL
- Software: Java, NodeJS, PM2, Edraw Max
- Install and Setup NginX

ผลงานที่ผ่านมา : Robotic-Eye System for Jelly Defect Detection Project (Bachelor Degree)

ชื่อ – นามสกุล : บุณยทัต อินสว่าง
ตำแหน่งงานปัจจุบัน : Senior System Engineer
ตำแหน่งในโครงการ : Operation Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2010 - 2014 : Bachelor of Engineering, Program in Computer Engineering
Kasetsart University

ประสบการณ์การทำงาน :

2020 – Present : *Senior System Engineer* – NIPA Technology Co., Ltd.
2014 - 2020 : *Compute Engineer* - Triple T Broadband, PCL.

นักวิจัยด้าน :

- Cloud Infrastructure (Openstack)
- Virtual Machine

ความเชี่ยวชาญเฉพาะทาง :

- Linux Server
- San Storage
- Shell script
- Backup Solution (Veeam)
- VMware Infrastructure

ผลงานที่ผ่านมา :

- Provision Oracle Database Production for Triple T Broadband PCL
- Production Infrastructure on Triple T Broadband PCL (VMware)
- San Storage (Hitachi VSP G200, IBM v7000, HP EVA)
- Oracle administrator 10g 11g and 18c

ชื่อ – นามสกุล : อภิวัฒน์ เปลี่ยนจิตรดี
ตำแหน่งงานปัจจุบัน : Senior System Engineer
ตำแหน่งในโครงการ : Operation Engineer

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2001	:	Master of Engineering, Program in Computer Engineering, specialize in Networks and Distributed Systems, Kasetsart University
2003	:	Bachelor of Engineering, Program in Computer Engineering Kasetsart University

ประสบการณ์การทำงาน :

2020 – Present	:	*Senior System Engineer* – NIPA Technology Co., Ltd.
2017 – 2020	:	*Senior Network & System Engineer* - Netdesign Host Co., Ltd.
2011 – 2015	:	*Core Network Operation* **-** Total Access Communication Public Company Limited (DTAC)
2006 - 2011	:	*Mobile Core Network Operation* – True Move Co., Ltd.
2004 - 2006	:	*Network & System Administrator* - M Link Asia Corporation Public Co., Ltd.
2001 - 2003	:	*Researcher* - Faculty of Computer, Engineering of Kasetsart University

ใบอนุญาต / ทะเบียน :

- Cisco Certified Network Associate (CCNA)
- Cisco Certified Network Professional (CCNP)
- Completion of the Gateway GPRS Service Node (GGSN)
- One-NDS – Introduction to X. 500 and One-NDS 7.1
- One-HLR Operation.

ความเชี่ยวชาญเฉพาะทาง :

- In depth expertise in the implementation, analysis, optimization, troubleshooting and documentation of LAN/WAN network systems.
- Strong Knowledge and experience on OpenStack components Neutron, Nova, Cinder, Horizon, Swift, Glance and Ceilometer.
- Strong hands on technical knowledge with GSM/GPRS/EDGE/3G technology, TCP/IP, Unix, LAN, routers, protocol/ LAN analyzer and CCNP certifications.

ผลงานที่ผ่านมา :

2017 - 2020	:	Install and support cloud customer of Netdesign host such as Asia news, GMO Z.com security, GMO Myanmar etc.
2011 - 2015	:	Support 3G/4G Core network implementation and UAT for DTAC Preventive maintenance the 3G core network equipment Periodic Performance and Capacity report
2006 – 2011	:	Support 2G/3G Core network implementation and UAT for TrueMove Preventive maintenance the 2G/3G core network equipment Periodic Performance and Capacity report
2003	:	Thesis title: “A Policy Monitoring System for Campus Network”
2001	:	Project title: “DNS Analyzer Tool”

ชื่อ – นามสกุล : จำเริญ สีด้วง
ตำแหน่งงานปัจจุบัน : Data Center Manager
ตำแหน่งในโครงการ : Network Operation Center (NOC)

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2009 - 2014 **:** Bachelor of Science, Program in Computer Engineering
Mahanakorn University of Technology

ประสบการณ์การทำงาน :

2014 – Present **:** *Data Center Manager* - NIPA Technology Co., Ltd.

ใบอนุญาต / ทะเบียน : -

นักวิจัยด้าน :

- Taxi Meter

ความเชี่ยวชาญเฉพาะทาง : -

ผลงานที่ผ่านมา :

- ตรวจสอบและวิเคราะห์เกี่ยวกับ Attack traffic พร้อมแก้ไขปัญหา
- ดูแลระบบ Virtual Machine Service
- เดินสายสัญญาณ (LAN, Fiber Optic) ให้กับลูกค้าและทีม Cloud พร้อม Configure Switch ที่เชื่อมต่อกับ Server
- Monitor Traffic Network, PRTG, Cacti, Zabbix

ชื่อ – นามสกุล : ตรีภพ ชุณหะสันต์
ตำแหน่งงานปัจจุบัน : Network Operation Center (NOC)
ตำแหน่งในโครงการ : Network Operation Center (NOC)

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2014 – 2016	:	Master of Information Technology, Program in Information Technology King Mongkut's University of Technology North Bangkok
2008	:	Bachelor of Engineering, Program in Mechanical Engineering King Mongkut's University of Technology North Bangkok

ประสบการณ์การทำงาน :

2017 – Present	:	*Network Operation Center (NOC)* – NIPA Technology Co., Ltd.
2012 – 2013	:	*Mathematic Lecturer* – Siam Business Administration Nonthaburi Technological College (SBAC NON.)
2008 - 2009	:	*Calibration Engineer & Quality Assurance* – Calibration Laboratory Co., Ltd.

ใบอนุญาต / ทะเบียน :

- Cisco certified CCNA routing and switching No. 43485416941FNCL

นักวิจัยด้าน : -

ความเชี่ยวชาญเฉพาะทาง :

- การติดตั้งเครื่อง server เข้า Rack server รวมทั้งการจัดการระบบสายต่าง ๆ
- ตรวจเช็ค วิเคราะห์ปัญหา แก้ปัญหา Hardware คอมพิวเตอร์หรือ server
- วิเคราะห์ suspicious network package หาสาเหตุและป้องกันปัญหาที่จะเกิดกับระบบเครือข่ายขององค์กร
- การตั้งค่าอุปกรณ์เกี่ยวกับเครือข่ายในระดับ access level และ distributed level
- ระบุปัญหา server, traffic, etc. จากโปรแกรม monitor เช่น PRTG, MRTG, Cacti

ชื่อ – นามสกุล : สรสิช ศรีเอี่ยมสะอาด
ตำแหน่งงานปัจจุบัน : Network Operation Center (NOC)
ตำแหน่งในโครงการ : Network Operation Center (NOC)

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2016 – 2018	:	Master of Engineering, Program in Computer Engineering Dhurakij Pundit University
2009 - 2012	:	Bachelor of Science, Program in Information and Communication Technology Nakhon Pathom Rajabhat University

ประสบการณ์การทำงาน :

2018 - Present	:	*Network Operation & Support (NOC)* - NIPA Technology Co., Ltd.
2017 - 2018	:	*Network Admin* – Pharmahof Co., Ltd.

ใบอนุญาต / ทะเบียน : -
นักวิจัยด้าน : -
ความเชี่ยวชาญเฉพาะทาง :
Design, analyze and fix Network problems
ผลงานที่ผ่านมา :
งานประชุมวิชาการเครือข่ายวิศวกรรมไฟฟ้า ครั้งที่ 11 (EENET2019)
งานวิชาการเรื่อง ประสิทธิภาพการทำงานของ Multipath TCP ในระบบโครงข่ายส่วนบุคคล

ชื่อ – นามสกุล : พันเทพ ไทยประทุม
ตำแหน่งงานปัจจุบัน : Network Operation Center (NOC)
ตำแหน่งในโครงการ : Network Operation Center (NOC)

สถานที่ทำงาน : NIPA Technology Co., Ltd. เลขที่ 72 อาคาร กสท โทรคมนาคม ชั้น 4 ห้อง 407-408 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพมหานคร 10500

ประวัติการศึกษา :

2009 - 2013 : Bachelor of Science, Program in Information Technology
Chandrakasem Rajabhat University

ประสบการณ์การทำงาน :

2016 – Present : *Network Operation* – NIPA Technology Co., Ltd.
2013 - 2016 : *Technical Support* – Smart Corporation, PLC.

ใบอนุญาต / ทะเบียน :

- CCNA Routing and Switching (CCNA 200-120)

นักวิจัยด้าน :

- ระบบสั่งซื้อสินค้า Shopping Cart

ความเชี่ยวชาญเฉพาะทาง :

- TCP/IP Network
- The 7 layers of the OSI Model
- Configuration Cisco Router and Switch
- LAN, WAN Technologies and IP Routing EIGRP, OSFP, BGP, MPLS, RIP, etc.

ผลงานที่ผ่านมา :

- ดูแลระบบเครือข่าย ISP CISCO ROUTER ของ Internet Data Center
- Support/Troubleshoot Computer System, Including of Hardware, Software
- System Veeam Backup & Replication, VMware vCenter Server
- ติดตั้ง Open Source OS: Linux, fedora, Ubuntu, Debian, Esxi, Proxmox, pfSense
- ติดตั้ง ดูแล ตรวจสอบระบบเครือข่าย แก้ไขปัญหาเฉพาะหน้าให้กับลูกค้า


กทปส