รายงานสหกิจศึกษาฉบับสมบูรณ์

แดชบอร์ดการซิงค์ของศูนย์ข้อมูล

Datacenter Synchronization Dashboard

นางสาวนัทธ์หทัย นวนหงษ์

สาขาวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา แดชบอร์ดการซิงค์ของศูนย์ข้อมูล

ชื่อ-สกุล นักศึกษา นางสาวนัทธ์หทัย นวนหงษ์

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์ **สาขาวิชา** วิศวกรรมสารสนเทศ

ชื่อ-สกุล อาจารย์นิเทศ รศ.ดร. อรรถสิทธิ์ หล่าสกุล

ชื่อ-สกุล ผู้นิเทศงาน ชัยสิทธิ์ สุรพฤกษ์

สถานประกอบการ บริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป

บทคัดย่อ

ในปัจจุบันธนาคารกสิกรไทย มีแอปพลิเคชันมากมายที่เปิดให้บริการแก่ผู้ใช้งาน ซึ่งทำงานเกี่ยวข้องกับฐานข้อมูล และมีการซิงค์ข้อมูลเพื่อให้ศูนย์กลางข้อมูลในแต่ละที่ มีข้อมูลเหมือนกัน ซึ่งหากเกิดข้อผิดพลาดในการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลขึ้น จะทำให้เกิดผลกระทบต่อผู้ใช้งานเป็นอย่างมาก และจะต้องหาต้นเหตุของข้อผิดพลาดหรือปัญหาที่แท้จริงให้ได้อย่างรวดเร็วที่สุด และนำไปแก้ไขในขั้นตอนต่อไป ด้วยเหตุผลในการลดทรัพยากรทางเวลาในการค้นหาข้อผิดพลาด จึงมีการจัดทำแผนภาพหรือแดชบอร์ดแสดงข้อมูลสถานะและพฤติกรรมของการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลแบบเรียลไทม์จากแอปพลิเคชันที่มีการซิงค์ข้อมูลของศูนย์กลางข้อมูลขึ้น เพื่อตอบสนองการใช้ทรัพยากรทางเวลาอย่างคุ้มค่า

คำสำคัญ : แดชบอร์ดการซิงค์ข้อมูลของฐานข้อมูล

Cooperative Title: Datacenter Synchronization Dashboard

Student intern name: Ms. Nathathai Nuanhong

Faculty: Engineering **Department:** Computer Engineering

Program: Information Engineering

Advisor name: Assoc.Prof.Dr. Attasit Lasakul

Mentor name: Mr. Chaiyasit Surapreuk

Company: Kasikorn Business-Technology Group

ABSTRACT

At present, Kasikorn Bank has many applications that are open to users. Which works in relation to the database And the data is synced so that each data center has the same data. If a sync error occurs between the data centers, it will have a huge impact on users. And must find the source of the root cause as quickly as possible and take to edit the next step. For this reason to reduce the time resources to find errors. Therefore creating dashboard showing real-time status and behavior data syncing between the data center from the data center sync application. In order to meet the cost-effective use of time resources.

Keywords : Database sync dashboard

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยคำแนะนำ และคำปรึกษาจากหลาย ๆ ฝ่ายด้วยกัน โดยเฉพาะการให้คำปรึกษาเรื่องหัวข้องาน ความเอาใจใส่แนะนำแนวคิดต่าง ๆ ที่เป็นประโยชน์ต่อการทำงานนี้ให้ผ่านลุล่วงไปได้ด้วยดี คือ คุณชัยสิทธิ์ สุรพฤกษ์ ตำแหน่ง Advance Engineer ผู้เป็นคนนิเทศงานและรับผิดชอบตรวจสอบการทำงานทั้งหมด อีกทั้งฝ่ายดูแลรายละเอียดของงาน คอยให้ความรู้ในเชิงเทคนิค การจัดหาข้อมูลต่าง ๆ คือ คุณชยพล แซ่พัง ตำแหน่ง System Engineer และพี่ ๆ ทุกท่านที่ให้ความรู้ ความเข้าใจ คำปรึกษาและแนวทางการแก้ปัญหาต่าง ๆ ในการทำงานครั้งนี้ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณอาจารย์ที่ปรึกษา คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้คำแนะนำในเรื่องการทำสหกิจศึกษาในครั้งนี้

สุดท้ายนี้ขอขอบคุณรุ่นพี่และเพื่อน ๆ ทุกคนที่คอยให้คำปรึกษาในเรื่องการทำงาน แนวทางการแก้ไขปัญหาต่าง ๆ ตั้งแต่เริ่มทำสหกิจศึกษาจนสำเร็จรายงานฉบับนี้ไปด้วยดี

นัทธ์หทัย นวนหงษ์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

บริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป เป็นบริษัทด้านเทคโนโลยีที่ช่วยในการขับเคลื่อนธนาคารกสิกรไทยสู่การเป็นธนาคารแห่งอนาคตแบบดิจิทัลที่สมบูรณ์ มุ่งคิดค้นนวัตกรรมร่วมกับพันธมิตรทางเทคโนโลยีและจับมือกับฟินเทค (FinTech) และ เทคสตาร์ทอัป (Tech Startup) สร้างนวัตกรรมทางการเงินรองรับระบบดิจิทัลแบงกิ้ง เพิ่มความสามารถในการแข่งขันท่ามกลางการเปลี่ยนแปลงของเทคโนโลยีทางการเงินในตลาดโลก พัฒนาระบบไอทีรูปแบบใหม่ เพื่อให้ลูกค้าได้รับบริการทางการเงินในโลกดิจิทัลที่สมบูรณ์แบบที่สุด

ระบบการทำงานภายในบริษัท เป็นส่วนสำคัญในการพัฒนาเทคโนโลยีต่าง ๆ ไม่ว่าจะเป็นการคิดค้นนวัตกรรมใหม่ ๆ การพัฒนาแอปพลิเคชัน การพัฒนาระบบการจัดการเรื่องฐานข้อมูลเพื่อรองรับข้อมูลได้อย่างรวดเร็วและมีประสิทธิภาพ และด้านอื่น ๆ ที่มีส่วนสำคัญต่อระบบธนาคารกสิกรไทย เพื่ออำนวยความสะดวกต่อความต้องการของผู้ใช้งานที่มากขึ้น

ในปัจจุบันเทคโนโลยีเริ่มมีการพัฒนามากขึ้น ทำให้เกิดการพัฒนาแอปพลิเคชันขึ้นมารองรับการใช้งานของลูกค้า ช่วยเพิ่มความสะดวกและรวดเร็วในการใช้งานทางการเงิน ทุกแอปพลิเคชันจะทำงานเกี่ยวข้องกับฐานข้อมูลและศูนย์กลางข้อมูล สิ่งที่สำคำคัญในการจัดการระบบแอปพลิเคชันคือการรองรับการใช้งานของลูกค้าที่มีจำนวนมากได้อย่างมีประสิทธิภาพ รวดเร็ว ไม่เกิดข้อผิดพลาดให้มากที่สุด และใช้เวลาแก้ไขข้อผิดพลาดให้น้อยที่สุดหากมีข้อผิดพลาดนั้นเกิดขึ้น เพื่อความพึงพอใจของลูกค้าที่เข้ามาใช้งาน

การเข้าร่วมโครงการสหกิจศึกษากับทางบริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป ในตำแหน่ง System Engineer ภายใต้บริษัท K-Pro แผนก Infrastructure ทีม Middleware โดยเป้าหมายหลักเปรียบเสมือนตัวกลางในการจัดการ ดูแล พัฒนา และ ปรับปรุงระบบขององค์กร รวมถึงการหาต้นเหตุของข้อผิดพลาดที่เกิดขึ้น หากเกิดปัญหา ซึ่งหากมีข้อผิดพลาดหรือปัญหาที่ทำให้ส่งผลกระทบต่อลูกค้า จะต้องหาต้นเหตุของปัญหาเพื่อนำไปแก้ไขให้อย่างเร็วที่สุด ซึ่งหลายครั้งที่ลูกค้าได้รับผลกระทบนั้นเกิดจากความผิดของการซิงค์ข้อมูลระหว่างศูนย์กลาง

ข้อมูล ทำให้ฐานข้อมูลแต่ละที่มีข้อมูลไม่มีเหมือนกัน ซึ่งความผิดพลาดที่เกิดขึ้นนั้นขึ้นอยู่กับหลากหลายปัจจัย ซึ่งจะสามารถตรวจสอบได้จากข้อมูลจราจรทางคอมพิวเตอร์หรือล็อกไฟล์ (log file) ของระบบหรือแอปพลิเคชันนั้น ๆ ซึ่งพบว่าการตรวจสอบล็อกไฟล์ในแต่ละระบบจนกว่าจะพบต้นเหตุของข้อผิดพลาดที่แท้จริงนั้น ยังไม่ตอบสนองการใช้ทรัพยากรทางด้านเวลา ส่งผลให้แก้ไขปัญหาที่เกิดขึ้นได้อย่างล่าช้า

จากปัญหาที่กล่าวมาข้างต้นนำมาซึ่งแนวคิดการพัฒนาประสิทธิภาพของขั้นตอนในการค้นหาปัญหา หรือข้อผิดพลาดที่แท้จริงที่เกี่ยวกับของการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล ให้อยู่ในรูปแบบของแผนภาพแสดงหรือแดชบอร์ด ที่สามารถตรวจสอบสถานะของระบบโดยรวม หรือ สถานะเครื่องของแต่ละระบบ พฤติกรรมของเครื่องนั้น ๆ รวมถึงมีการแจ้งเตือนเมื่อเกิดปัญหาหรือข้อผิดพลาดนั้นขึ้นในแบบเรียลไทม์ (real-time) ทั้งนี้เพื่อลดระยะเวลาในการหาข้อผิดพลาดหรือปัญหาที่เกิดขึ้นไปจนถึงระยะเวลาที่ส่งผลกระทบต่อลูกค้า และเพื่อวิเคราะห์พฤติกรรมของระบบในเบื้องต้นด้วย

1.2 วัตถุประสงค์ของการทำวิจัย

1.2.1 เพื่อลดเวลาในการเกิดผลกระทบต่อลูกค้า

1.2.2 เพื่อลดระยะเวลาในการหาต้นเหตุที่แท้จริงของปัญหาที่เกิดขึ้น

1.2.3 เพื่อเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูลในเรื่องประสิทธิภาพของระบบ

1.3 ขอบเขตของงานวิจัย

สามารถแบ่งขอบเขตของงานวิจัยได้ดังนี้

1.3.1 การศึกษาเกี่ยวกับปัญหาที่ลูกค้าได้รับผลกระทบจากความผิดพลาดของการซิงค์ข้อมูลของศูนย์กลางข้อมูลและวิธีในการหาต้นเหตุของข้อผิดพลาด

1.3.1.1 ศึกษาโครงสร้างในการซิงค์ข้อมูลของศูนย์กลางข้อมูลในแต่ละแอปพลิเคชัน

1.3.1.2 ศึกษาตัวอย่างแอปพลิเคชันที่ใช้งานศูนย์กลางฐานข้อมูล

1.3.1.3 ศึกษาโครงสร้างของแอปพลิเคชันตัวอย่างที่ใช้งานศูนย์กลางฐานข้อมูล

1.3.1.4 ศึกษาการทำงานเบื้องต้นของตัวอย่างแอปพลิเคชัน

1.3.1.5 ศึกษาเกี่ยวกับวิธีการหาข้อผิดพลาดที่เกิดขึ้นในแบบเก่า

1.3.2 การศึกษาเกี่ยวกับแนวคิดการทำ centralize log

1.3.2.1 ศึกษาเกี่ยวกับวิธีดึงล็อกไฟล์ (log file) ของแต่ละแอปพลิเคชัน และเครื่องมือที่ใช้

1.3.2.2 ศึกษาเกี่ยวกับวิธีใช้เครื่องมือที่ใช้ในการทำ centralize log

1.3.2.3 ศึกษาคำสั่งที่ใช้งานของเครื่องมือนั้น ๆ

1.3.3 การศึกษาเกี่ยวกับเครื่องมือที่ใช้สนับสนุนงานเพื่อเพิ่มประสิทธิภาพของงาน

1.3.3.1 ศึกษาข้อจำกัดหรือข้อบกพร่องของเครื่องมือที่ใช้

1.3.3.2 ศึกษาประโยชน์ของเครื่องมือที่ใช้เพิ่มประสิทธิภาพของงาน

1.3.3.3 ศึกษาวิธีใช้และคำสั่งของเครื่องมือที่ใช้สนับสนุนงาน

1.4 วิธีการดำเนินการวิจัย

การวางแผนการทำงานในแต่ละขั้นตอนเป็นไปตามขอบเขตของการวิจัย ซึ่งแบ่งออกเป็นแต่ละส่วนตามลักษณะการทำงาน ดังนี้

1.4.1 กระบวนการเข้าใจปัญหาที่เกิดขึ้นของการซิงค์ข้อมูลของศูนย์กลางฐานข้อมูลและปัญหาของวิธีการค้นหาข้อผิดพลาดที่เกิดขึ้น

ขอบเขตการทำงาน จะต้องเข้าใจปัญหาที่เกิดขึ้นโดยสังเขปว่าคืออะไร สามารถแก้ไขหรือเพิ่มประสิทธิภาพได้อย่างไร ต้องเข้าใจขั้นตอนการค้นหาต้นเหตุของปัญหาหรือข้อผิดพลาดที่เกิดขึ้น โดยมีกระบวนการดังนี้

1.4.1.1 ศึกษาและทำความเข้าใจถึงปัญหาของลูกค้าที่ได้รับผลกระทบจากข้อผิดพลาดหรือปัญหาจากการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล

1.4.1.2 ศึกษาวิธีการค้นหาต้นเหตุของข้อผิดพลาดหรือปัญหาที่เกิดขึ้นรวมถึงข้อบกพร่องของวิธีการนี้

1.4.1.3 ศึกษาวิธีการตรวจจับข้อผิดพลาดหรือปัญหาในแบบใหม่ คือ แนวคิดแบบเซนทรัลไลซ์ล็อก (centralize log)

1.4.2 การศึกษาและทำความเข้าใจเกี่ยวกับวิธีการสร้างแผนภาพแสดงสถานะของระบบ ตามแนวคิด เซนทรัลไลซ์ล็อก รวมถึงศึกษาเครื่องมือที่นำมาใช้เพื่อเพิ่มประสิทธิภาพของงาน

ขอบเขตการทำงาน ต้องทำความเข้าใจเกี่ยวกับแนวคิดและเครื่องมือในการทำเซนทรัลไลซ์ล็อก และวิธีการใช้เครื่องมือในการสร้างแผนภาพแสดงและการแจ้งเตือน โดยมีกระบวนการดังนี้

1.4.2.1 ศึกษาแนวคิดของการทำเซนทรัลไลซ์ล็อก

1.4.2.2 ศึกษาเครื่องมือที่ใช้ทำเซนทรัลไลซ์ล็อก คือ ELK stack

1.4.2.3 ศึกษาเครื่องมือที่ใช้สร้างแผนภาพแสดงและการแจ้งเตือน คือ กราฟานา (grafana) และ ไลน์โนติฟาย (Line notify)

1.4.2.4 ศึกษาเครื่องมือที่ใช้เพิ่มประสิทธิภาพงาน คือ ด็อกเกอร์ (docker)

1.4.3 การทำความเข้าใจโครงสร้างของแอปพลิเคชันที่นำมาสร้างแผนภาพแสดง

ขอบเขตการทำงาน ต้องเข้าใจเกี่ยวกับแอปพลิเคชันที่มีการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลของทางองค์กรนั้นว่า มีโครงสร้างและวิธีการทำงานที่ไม่เหมือนกัน บางแอปพลิเคชันทำงานบน

แพลตฟอร์ม (platform) ที่แตกต่างกัน มีรายละเอียดและรูปแบบการส่งข้อมูลมายังล็อกไฟล์ (log file) ที่ต่างกัน ซึ่งมีกระบวนการศึกษาดังนี้

1.4.3.1 ศึกษาแอปพลิเคชันที่มีการซิงค์ข้อมูลกันระหว่างศูนย์กลางข้อมูล ที่นำมาสร้างแผนภาพแสดง (dashboard)

1.4.3.2 ศึกษาวิธีการนำข้อมูลที่เกี่ยวข้องกับการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลมาเขียนลงล็อกไฟล์ และเครื่องมือที่จับล็อกไฟล์เพื่อนำไปใช้งานต่อ คือ ไฟล์บีท (filebeat)

1.4.3.3 ศึกษาวิธีการจับล็อกไฟล์ จากเทคโนโลยีที่ต่างกัน คือ อินฟลักซ์ดีบี (influxDB) จากแอปพลิเคชัน Online Retail Payment Gateway (ORPG) ร่วมกับแอปพลิเคชันเคพลัส (KPLUS) และ TANDEM MaInframe จากแอปพลิเคชันเอทีเอม (ATM)

1.4.3.4 ศึกษาโครงสร้างของการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลในแต่ละแอปพลิเคชัน เพื่อให้เข้าใจรายละเอียดและข้อมูลต่าง ๆ ในล็อกไฟล์ ของแอปพลิเคชันนั้น ๆ

1.4.4 การสร้างแผนภาพเพื่อแสดงสถานะหรือค่าตัวแปรต่าง ๆ ที่ต้องการวิเคราะห์

ขอบเขตของการทำงาน จัดเตรียมข้อมูลที่ต้องการ และทำการเชื่อมต่อกับเครื่องมือต่าง ๆ เพื่อเปลี่ยนแปลงข้อมูลในรูปของตัวอักษรให้เป็นรูปภาพ โดยมีกระบวนการดังนี้

1.4.4.1 จัดเตรียมข้อมูลที่ต้องการนำไปสร้างแผนภาพแสดงลงในล็อกไฟล์ (logfile)

1.4.4.2 ติดตั้งไฟล์บีท (filebeat) ไว้บนเครื่องที่มีการเขียนล็อกไฟล์ที่เกี่ยวข้อง

1.4.4.3 เขียนการตั้งค่าไฟล์บีท และส่งข้อมูลออกไปยังล็อกแสตช (logstash)

1.4.4.4 ติดตั้งและเขียนการตั้งค่าล็อกแสตช เพื่อคัดกรองข้อมูลและตกแต่ง แล้วจึงส่งไปเก็บไว้ในฐานข้อมูลอีลาสติคเสิร์ช (elasticsearch)

1.4.4.5 ตรวจสอบรายละเอียดของข้อมูลที่จะนำไปสร้างแผนภาพจากคิบานา (kibana)

1.4.4.6 ตั้งค่ากราฟานา (grafana) เพื่อทำแผนภาพแสดงและการแจ้งเตือนผ่านไลน์โนติฟาย (line notify)

1.4.4.7 ตรวจสอบความถูกต้องของแผนภาพแสดงนี้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่ได้รับจากการเข้าร่วมทำโปรเจกต์ที่ได้รับมอบหมายตลอดระยะเวลาภายใต้โครงการสหกิจศึกษากับทางบริษัท กสิกร บิซิเบส-เทคโนโลยี กรุ๊ป สามารถแบ่งเป็น 3 ส่วนดังต่อไปนี้

1.5.1 ประโยชน์ต่อระบบการทำงาน

1.5.1.1 ช่วยลดความซับซ้อนและระยะเวลาของขั้นตอนในการหาต้นเหตุของข้อผิดพลาดหรือปัญหา

1.5.1.2 ช่วยเพิ่มประสิทธิภาพของการวิเคราะห์ข้อมูลได้

1.5.1.3 ช่วยให้นักพัฒนาหรือผู้แก้ไขปัญหา ทำงานได้อย่างรวดเร็วและมีประสิทธิภาพมากยิ่งขึ้น

1.5.2 ประโยชน์ต่อลูกค้า

1.5.2.1 ลูกค้าได้รับผลกระทบที่น้อยลง

1.5.2.2 ลดปัญหาที่ข้อมูลไม่ถูกต้องตามการใช้งาน

1.5.3 ประโยชน์ต่อผู้ดำเนินงาน

1.5.3.1 มีความรู้ ความเข้าใจในเชิงเทคนิค การเขียนโปรแกรม และการใช้ภาษาคอมพิวเตอร์มากขึ้น

1.5.3.2 มีความเข้าใจในการคิด วิเคราะห์ และวางแผนการจัดการงาน

1.5.3.3 เรียนรู้วิธีการตรวจสอบงาน

1.5.3.4 มีความรับผิดชอบกับงานที่ได้รับมอบหมายมากขึ้น

1.5.3.5 เรียนรู้ทักษะการสื่อสาร และทักษะการทำงานร่วมกับผู้อื่นได้

1.5.3.6 เรียนรู้การปรับตัวให้เข้ากับสภาพแวดล้อมการทำงานได้

1.5.3.7 เรียนรู้การแก้ปัญหาที่เกิดขึ้นในการทำงานด้วยตัวเองได้

1.5.3.8 เรียนรู้ข้อผิดพลาดของตนเองและนำไปแก้ไขข้อผิดพลาดนั้นได้

บทที่ 2

แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

เนื่องจากงานวิจัยนี้เป็นการสร้างแผนภาพแสดงสถานะและข้อมูลที่เกี่ยวกับระบบการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล โดยจะต้องมีความรู้พื้นฐานเกี่ยวกับล็อกไฟล์ ความเข้าใจเกี่ยวกับแนวคิดเกี่ยวกับการจัดการล็อก (centralized log) ความรู้เกี่ยวกับซอฟต์แวร์ต่าง ๆ ที่เกี่ยวข้อง และความรู้ในเชิงเทคนิคอื่น ๆ มีรายละเอียดดังนี้

2.1 แนวคิดเกี่ยวกับข้อมูลการจัดการล็อก (centralize log management)

2.1.1 ความรู้พื้นฐานเกี่ยวกับล็อกไฟล์ (Log File)

ล็อกไฟล์เป็นไฟล์ที่ใช้สำหรับบันทึกการกระทำต่าง ๆ บนระบบคอมพิวเตอร์ ซึ่งมีการเก็บข้อมูลจำนวนมากโดยมีการเก็บข้อมูลแบบบันทึกต่อ ๆ กัน ล็อกไฟล์ส่วนใหญ่จะเก็บในรูปของ Text files ที่เป็นการบันทึกข้อมูลทีละบรรทัด โดยมีรูปแบบของข้อมูลที่แตกต่างกันตามชนิดของระบบปฏิบัติการหรือ โปรแกรมประยุกต์ โดยมีการเก็บรายละเอียดหลัก ๆ ดังนี้

1) วันและเวลาที่เกิดเหตุการณ์นั้นขึ้น

2) บันทึกเหตุการณ์ (event) ที่เกิดขึ้นพร้อมทั้งชื่อของเหตุการณ์ ซึ่งถ้าระบบแบ่งการจดบันทึกเป็นหลายหมวดหมู่ (Category) ก็จะระบุด้วยว่าเหตุการณ์ดังกล่าวอยู่ในหมวดหมู่ไหน

3) รายละเอียดของเหตุการณ์ (description) ซึ่งจะบอกว่าเหตุการณ์ดังกล่าวมีรายละเอียดอะไร เกิดความผิดปกติที่ตรงตำแหน่งไหน อาจเป็นที่ตัวอุปกรณ์หรือไฟล์ข้อมูลในระบบ

4) ชนิดของเหตุการณ์ที่ผิดปกติ (type) หรือระดับความรุนแรงของเหตุการณ์ที่ผิดปกติดังกล่าว เช่น ถ้าเหตุการณ์นั้นไม่มีความรุนแรงกับระบบมากนัก แต่เข้าข่ายผิดสังเกต ก็แจ้งเตือนในระดับ

ของ Warning หรือถ้าเหตุการณ์นั้นร้ายแรงมากจนถึงขั้นทำให้ระบบเกิดความเสียหายจนไม่สามารถปฏิบัติการได้ ก็จะแจ้งเตือนว่าเป็น Error

ล็อกไฟล์สามารถแบ่งประเภทได้เป็น 4 ประเภท ดังนี้

1) System logs เป็นเหตุการณ์ของระบบที่ถูกดำเนินการโดยระบบปฏิบัติการ เช่น การบู๊ต (boot) เครื่องหรือปัญหาที่พบระหว่างการทำงาน มีเซอร์วิสใดทำงานได้หรือไม่ได้บ้าง ไดรเวอร์ของอุปกรณ์ต่าง ๆ เป็นปกติหรือไม่ ฯลฯ โดยตลอดระยะเวลาที่ระบบปฏิบัติการนั้นยังคงทำงานอยู่ถ้ามีเหตุการณ์ใด ๆ เกิดขึ้นและส่งผลกระทบต่อระบบ ก็จะมีการเก็บข้อมูลเหตุการณ์นั้นไว้ใน System log เสมอ โดยการทำงานของอุปกรณ์ ต่าง ๆ โดยส่วนใหญ่มักจะเก็บ Timestamp, event, status, error code, service name, user หรือ system account โดย Syslog message ประกอบด้วย วันเวลาที่บันทึกข้อมูลล็อก ระดับความสำคัญ ชื่อ Process และ Process ID ชื่อเครื่อง computer หรือ IP Address และ ข้อความที่ส่งมายัง syslog

2) Application logs ทุกแอปพลิเคชันจะมีการสร้างล็อกไฟล์ของตัวเอง โดยจะเก็บรายละเอียดการทำงานของแอปพลิเคชันต่าง ๆ ที่รันอยู่บนระบบปฏิบัติการ เช่น Web Server, Mail Server, DNS Server ฯลฯ ล็อกไฟล์ประเภทนี้จะบันทึกการทำงานของแอปพลิเคชันนั้น ไม่ว่าจะเป็นการทำงานที่สำเร็จก็ตาม ซึ่งผู้ดูแลระบบจะสามารถกลับมาตรวจสอบ และแก้ไขความผิดปกติได้อย่างตรงจุด

3) Personal Firewall and Network Firewall Log โปรแกรมประเภทไฟร์วอลล์ (Firewall) เป็นโปรแกรมที่ใช้ปกป้องผู้ใช้งานอินเทอร์เน็ตจากการล่วงล้ำเข้าสู่ระบบจากปัจจัยภายนอก และป้องกันไม่ให้ข้อมูลในระบบถูกส่งออกไปโดยรู้เท่าไม่ถึงการณ์ โดยข้อมูลทั้งหมดที่จะผ่านเข้าหรือออกจากคอมพิวเตอร์ที่ติดตั้งโปรแกรม Personal Firewall นี้ไว้ จะต้องผ่านไฟร์วอลล์ ซึ่งจะคอยตรวจสอบโดยใช้เงื่อนไขด้านความปลอดภัยเป็นกฎเกณฑ์ โดยจะเก็บการกระทำนั้นไว้ในรูปแบบล็อกไฟล์

4) FTP Server Log ซึ่ง FTP คือ เครื่องบริการการรับ-ส่งข้อมูล ซึ่งเปิดให้ผู้ใช้ที่เป็นสมาชิกเข้าใช้แต่บางเครื่องอาจเป็นเครื่องให้ผู้ใช้ทั่วไปเข้าใช้ โดยใช้รหัสสมาชิก anonymous ซึ่งเป็นที่รู้

กันทั่วโลกว่า 29 เป็นรหัสผู้ใช้สำหรับผู้ที่ไม่ประสงค์ออกนาม โดยเครื่องที่เปิดบริการ FTP จะเปิด TCP port 21 ไว้เชื่อมต่อ โดยมี 2 mode คือ

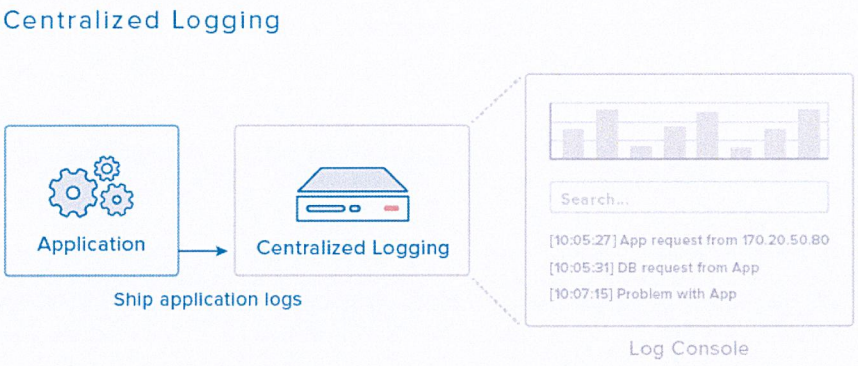
- FTP standard mode คือ การเชื่อมต่อที่ server เชื่อมต่อกับ client ผ่าน port 20 เป็น Out going port ส่วน port ฝั่ง client จะแล้วแต่ตกลงกัน แต่ถ้า client มี firewall ที่ไม่เปิดบริการก็จะติดต่อไม่ได้

- FTP passive mode คือ การเชื่อมต่อที่ client เป็นผู้เชื่อมต่อไปยัง server เพื่อใช้หมายเลขport ที่แล้วแต่จะตกลงในการส่งข้อมูล

โดยการกระทำต่าง ๆ นั้นตัวแม่ข่ายของ FTP จะทำการเก็บ log file ไว้ ตัวอย่างเช่น FTP Log File ในระบบปฏิบัติการ Unix

2.1.2 ความสำคัญของการจัดการล็อก

ล็อก (Log) คือ ข้อมูลจราจรของคอมพิวเตอร์ ซึ่งเป็นสิ่งจำเป็นในการระบุถึงประวัติของผู้ใช้งานคอมพิวเตอร์ โดยล็อกจะช่วยให้ผู้ดูแลระบบทราบถึงภัยที่จะเกิดขึ้น เช่น ใครที่พยายามสุ่มรหัสเข้าไฟล์สำคัญ ใครที่เข้ามาใช้งานในเวลาที่ไม่ให้ใช้ เป็นต้น แต่เนื่องจากรูปแบบของล็อกเป็นไฟล์ตัวอักษรที่อ่านได้ยาก พร้อมทั้งมีปริมาณเยอะและมีที่มาจากหลายแหล่ง เช่น Firewall log, AD log, Web Server log เป็นต้น จึงทำให้ผู้ดูแลระบบยากต่อการตรวจสอบถึงเหตุการณ์ดังกล่าว บริษัทจึงต้องมีตัวช่วยในการบริหารและจัดเก็บ Log ที่สามารถดูได้จากศูนย์กลาง (Centralize Log Management) และวิเคราะห์ให้เห็นถึงภัยด้วยการแจ้งเตือน (Alerting) เพื่อความง่ายในการป้องกันและดูแลความปลอดภัยทางคอมพิวเตอร์



ภาพที่ 2.1 แนวคิดการจัดการล็อกส่วนกลาง

2.2 ซอฟต์แวร์ที่เกี่ยวข้อง

2.2.1 อีเอลเคสแต็ค (ELK Stack) และไฟล์บีท (Filebeat)

ELK Stack ประกอบด้วย Elasticsearch, Logstash และ Kibana เป็นชุดซอฟต์แวร์ที่ถูกออกแบบมาให้สามารถทำงานร่วมกันดังภาพที่ 2.2 เป็นระบบจัดเก็บข้อมูล ระบบจัดเรียงข้อมูล และระบบแสดงผล ตามลำดับ เพื่อเพิ่มประสิทธิภาพการจัดเก็บและการวิเคราะห์ข้อมูล ส่วน Filebeat เป็นซอฟต์แวร์ที่ทำหน้าที่เป็น Log Agent ในการส่งข้อมูลล็อก ทำให้สามารถนำมาทำเป็นระบบจัดเก็บข้อมูลจราจรคอมพิวเตอร์ได้เป็นอย่างดี และถูกออกแบบมาเพื่อใช้งานร่วมกับ Elasticsearch และ Logstash โดยมีรายละเอียด ดังนี้

- Elasticsearch คือ ระบบฐานข้อมูลแบบ NoSQL แบบหนึ่ง ที่สามารถทำการค้นหาข้อมูลได้แบบ distributed search โดยระบบจะทำการ index ข้อมูลที่เก็บไว้ ทำให้สามารถทำการค้นหาข้อมูลได้อย่างรวดเร็ว และแทบเป็น real-time ถูกพัฒนาต่อยอดมาจาก Apache lucene

- Logstash คือ ระบบ log parser ชนิดหนึ่งสำหรับจัดการ log และ event ต่าง ๆ ที่เกิดขึ้นมา จากหลาย ๆ แหล่งพร้อมกัน เพื่อทำการสะสม กรอง และ จัดเก็บข้อมูล ในเวลาเดียวกัน เพื่อการใช้งานต่อไป ส่วนขั้นตอนการทำงานของ Logstash ประกอบไปด้วย 3 ขั้นตอน คือ

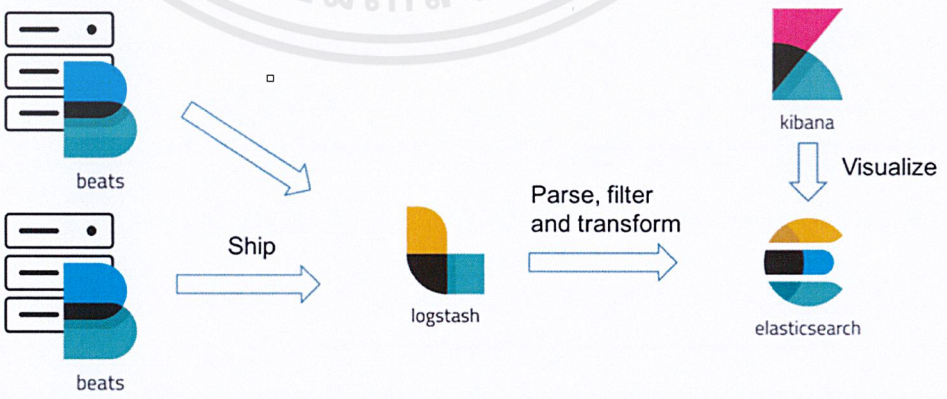
1) Inputs เพื่อรับข้อมูลเข้ามา Input plugins ที่ถูกใช้บ่อย ๆ ได้แก่ file, syslog, redis และ beats

2) Filters เป็นตัวกลางในการประมวลผลของ logstash ซึ่งสามารถนำตัวกรองหลาย ๆ ตัวมารวมกันใน Logstash pipeline เพื่อกรองข้อมูลให้อยู่ในรูปแบบที่ต้องการได้

3) Outputs เป็นเฟส (phase) สุดท้ายของ Logstash pipeline เพื่อนำข้อมูลออกไปยังปลายทางต่าง ๆ Output plugins ที่ถูกใช้กันทั่วไปได้แก่ elasticsearch, file, graphite และ statsd

- Kibana คือ user interface (UI) ของชุดซอฟต์แวร์ ELK ที่ช่วยให้ผู้ใช้สามารถจัดการกับข้อมูลได้ง่ายมากยิ่งขึ้น ถูกพัฒนาด้วย JavaScript โดยเรียกข้อมูลจาก Elasticsearch ผ่าน REST APIs อนุญาตให้ผู้ใช้งานสามารถสร้างส่วนการแสดงผลข้อมูล หรือ Dashboard ได้ตามความต้องการ และสามารถแสดงผลข้อมูลแบบ real-time ในรูปแบบต่าง ๆ เช่น ตาราง ฮิสโทแกรม แผนที่ และ กราฟต่าง ๆ เช่น กราฟเส้น กราฟแท่ง และ กราฟวงกลม ได้

- Filebeat คือ Log Agent ชนิดหนึ่ง ซึ่งจะคอยจัดการส่งต่อ Log ต่าง ๆ ที่เก็บได้จากเครื่อง Client ไปเก็บไว้ยังเครื่อง Master เพียงเครื่องเดียว ถูกออกแบบมาให้ทำงานร่วมกับ Elasticsearch และ Logstash โดยเฉพาะ เมื่อใดก็ตามที่ข้อมูลใน Elasticsearch และ Logstash มีขนาดเยอะมาก ก็สามารถส่งข้อมูลบอกให้ Filebeat ชะลอการอ่านและส่งข้อมูลล็อกได้



ภาพที่ 2.2 แผนผังการทำงานร่วมกันของ ELK Stack

2.2.2 กราฟานา (Grafana)

กราฟานา คือ เครื่องมือสร้างแดชบอร์ด (dashboard) ที่เป็นโอเพนซอร์ส (open source) หรือเปิดให้ใช้งานฟรี โดยสามารถทำงานร่วมกับแหล่งข้อมูล (Datasource) ได้อย่างหลากหลาย เช่น Graphite, InfluxDB, OpenTSDB หรือ Elasticsearch เป็นต้น สามารถดึงข้อมูลออกมาได้ในระดับเรียลไทม์ (realtime) ครอบคลุมการสร้างแดชบอร์ดในหลาย ๆ รูปแบบ เช่น ตาราง ฮิสโทแกรม หรือ กราฟ เป็นต้น และสามารถต่อยอดได้ทั้งในเรื่องการหาความผิดปกติของระบบ เช่น การแจ้งเตือน (Alerting) ไปยังระบบต่าง ๆ เช่น slack, PagerDuty, Victor หรือ VictorOps เมื่อเกิดหรือก่อนเกิดปัญหา รวมไปถึงการวางแผนการจัดการทรัพยากรของระบบ (Capacity Planning)

2.2.3 วีไอ (Vi)

Vi เป็นซอฟต์แวร์แก้ไขข้อความ ที่ถูกพัฒนาโดยบิล จอย เมื่อ ค.ศ. 1976 เพื่อใช้ในระบบปฏิบัติการ BSD ยุคแรก ๆ ชื่อ vi นั้นนำมาจากจำนวนอักษรที่สั้นที่สุดของคำสั่ง visual ที่ไม่ซ้ำกับคำสั่งอื่น ของโปรแกรม ex ซึ่งใช้ในการเปลี่ยนโหมดของ ex จาก line editor เป็น visual อ่านออกเสียงว่า "วี-ไอ" ไม่ใช่ "ไว" และไม่ได้หมายถึงเลข 6 ตามอักษรโรมัน

vi กลายเป็นโปรแกรมแก้ไขข้อความที่นิยมที่สุด และเป็นมาตรฐานของระบบปฏิบัติการยูนิกซ์แทบทุกตัว และสามารถแบ่งโหมดการทำงานได้เป็น 3 โหมดหลัก ๆ ดังนี้

1) Normal Mode หรือเรียกว่า Command Mode เป็น Mode ที่เอาไว้ใช้รับคำสั่ง หรือ Key ลัดบางตัว หรืออาจมองได้ว่าเป็น Standby Mode เป็นตัวกลางเพื่อสามารถไปยัง Mode อื่น ๆ ได้

2) Insert Mode ใช้เมื่อต้องการ เพิ่มเติม แก้ไข หรือ ลบข้อมูลในไฟล์ เข้าโดยการกด i

3) Visual Mode ถ้าเทียบกับ Text Editor ที่เป็น GUI คือ การเอาเมาส์ลากคลุมดำเลย เข้าได้โดยการกด v วิธีการดูว่ากำลังทำงานอยู่ Mode ไหน และสามารถดูได้จากด้านล่างของหน้าจอ หรือ

ใช้ Theme ต่าง ๆ ก็จะมีแสดงให้ดูเหมือนกัน และเมื่อต้องการออกจากโหมดนั้น ๆ สามารถใช้ปุ่ม esc เพื่อกลับไปยัง Normal Mode ได้

2.2.4 เชลล์ สคริปต์ (Shell script)

shell script (เชลล์ สคริปต์) คือ โปรแกรมหนึ่งบนระบบ unix/linux ซึ่ง shell script ทำหน้าที่เป็นส่วนติดต่อผู้ใช้ (interface) ระหว่างผู้ใช้กับระบบปฏิบัติการ unix/linux (เคอร์เนล) ซึ่ง Shell ไม่ได้เป็นส่วนหนึ่งของเคอร์เนล แต่ใช้เคอร์เนลในการประมวลผล ผู้ใช้สามารถสั่งงานระบบปฏิบัติการโดยผ่านทาง secure shell เท่านั้น โปรแกรม Secure Shell ยังมีคุณสมบัติของ Shell Programming Language ทำให้ผู้ใช้สามารถนำคำสั่งต่าง ๆ ของ Shell มาเขียนโปรแกรมเก็บเป็นไฟล์ไว้ได้ เรียกว่า shell script (เชลล์ สคริปต์) และสามารถแบ่งออกเป็น 4 ประเภท ดังนี้

1) Bourne shell (/bin/sh) เป็น Shell ในยุคแรกๆ ที่มีการใช้กันอย่างแพร่หลาย มีการกำหนดโครงสร้างภาษาคล้าย ๆ กับภาษาอัลกอ (AlgoLanguage) สามารถเขียนเป็น shellscript ได้ และยังเป็นเชลล์มาตรฐานที่มีในระบบปฏิบัติการ Unix ทุกตัว และยังสามารถย้าย shell script ไปยัง Unix ระบบอื่นได้โดยไม่ต้องแก้ไขสคริปท์ Bourne shell มี default prompt เป็นเครื่องหมาย $

2) C shell (/bin/csh) เป็น Shell ที่พัฒนาขึ้นมาหลังจาก Bourne shell มีรูปแบบคำสั่งและไวยากรณ์เหมือนกับภาษาซี (C Language) มีฟังก์ชั่นการทำงานหลากหลาย สะดวก อีกทั้งยังสามารถควบคุมการไหลของข้อมูลได้ดีกว่า Bourneshell และยังมีความสามารถในการเรียกใช้คำสั่งที่ใช้ไปแล้ว C shell มี default prompt เป็นเครื่องหมาย %

3) Kornshell คอร์นเชลล์ (/bin/ksh) เป็น shell ที่พัฒนามาจากต้นแบบของ Bourneshell และ CShell สามารถทำงานใน function ของ Bourne shell ได้ทุกอย่าง การเขียน shell script ทำได้ง่าย และรัดกุมขึ้น สามารถนำคำสั่งที่ใช้ไปแล้ว กลับมา execute ใหม่ได้ ถือได้ว่า Korn shell เป็นการรวมเอาข้อดีของ Bourneshell และ CShell เข้ามาไว้ด้วยกัน แต่ไม่ได้มีในUNIX ทุกตัว Kornshell จะมี default prompt เป็นเครื่องหมาย $

4) Bourne again shell หรือ bash (แบช) /bin/bash หรือ /usr/local/bin/bash เป็นการเอา Bourne shell กลับมาพัฒนาใหม่ ทำให้สามารถทำงานแบบ line editing ได้ และยังได้เพิ่มประสิทธิภาพในการทำงานอีกหลายอย่าง bashshell นี้ไม่ใช่มาตรฐานของ Unix Shell แต่เป็น default shell ของ linux ในปัจจุบัน Bash จะมี default prompt เป็นเครื่องหมาย # หรือ $

2.3 ทฤษฎีเกี่ยวกับศูนย์กลางข้อมูล (Data Center)

Data Center หรือศูนย์กลางข้อมูล คือ สิ่งที่เปรียบเสมือนห้องที่ถูกออกแบบมาเพื่อใช้สำหรับเป็นที่อาศัยและพักพิงของเครื่องเซิร์ฟเวอร์ ไม่ว่าจะเป็น Web Hosting ขนาดเล็ก ไปจนถึง Super Computer ของธุรกิจขนาดใหญ่ ที่มีการใช้ข้อมูลในปริมาณมาก ๆ ดังนั้น Data Center จะต้องถูกออกแบบโดยมุ่งเน้นที่ความสามารถเพื่อตอบสนองให้เครื่องเซิร์ฟเวอร์นั้นทำงานได้อย่างเสถียรที่สุดเป็นอันดับแรก ซึ่งต้องมีการพร้อมใช้งานและเข้าถึงข้อมูลจากทั่วทุกมุมโลกได้อย่างรวดเร็วและตลอดเวลา เครื่องเซิร์ฟเวอร์จะทำงานโดยไม่มีการหยุดพัก จึงจำเป็นต้องมีห้องที่เก็บเครื่องเซิร์ฟเวอร์ที่มีความเสถียรภาพสูง ห้องจะต้องมีความเย็นคงที่ มีไฟสำรองให้พร้อมใช้งานอยู่เสมอ และมีเจ้าหน้าที่คอยประจำการเพื่อป้องกันไม่ให้ข้อมูลสูญหายและมีตรวจสอบระบบอยู่ตลอดเวลา

มาตรฐานของ Data center กำหนดคุณภาพออกเป็น 4 ระดับ (Tier) โดยมีรายละเอียดดังนี้

1) Tier 1 ระบบต้องทำงานได้อย่างน้อย 99.671% หรือภายในหนึ่งปีระบบสามารถล่ม หรือ ไม่สามารถใช้งานได้ไม่เกิน 28.8 ชั่วโมง แต่ไม่มีระบบทำงานสำรอง

2) Tier 2 ระบบต้องทำงานได้อย่างน้อย 99.741% หรือภายในหนึ่งปีระบบสามารถล่ม หรือ ไม่สามารถใช้งานได้ไม่เกิน 22 ชั่วโมง โดยต้องมีระบบทำงานสำรองด้วย

3) Tier 3 ระบบต้องทำงานได้อย่างน้อย 99.982% หรือภายในหนึ่งปีระบบสามารถล่ม หรือ ไม่สามารถใช้งานได้ไม่เกิน 1.6 ชั่วโมง โดยต้องมีระบบทำงานสำรอง มีไฟฟ้าสำรอง และช่องทางเชื่อมต่ออินเทอร์เน็ตสำรอง

4) Tier 4 ระบบต้องทำงานได้อย่างน้อย 99.995% หรือภายในหนึ่งปีระบบสามารถล่ม หรือ ไม่สามารถใช้งานได้ไม่เกิน 24 นาที โดยต้องมีระบบทำงานสำรอง มีไฟฟ้าสำรอง ช่องทางเชื่อมต่ออินเทอร์เน็ตสำรอง และทุกอย่างที่เกี่ยวกับระบบใน IDC ก็ต้องมีตัวสำรอง เช่น ระบบแอร์ ระบบดับเพลิง ฯลฯ

2.4 ทฤษฎีเกี่ยวกับฐานข้อมูล

ได้ใช้ทฤษฎีที่เกี่ยวกับระบบฐานข้อมูล ดังนี้

2.4.1 ระบบไคลเอนต์/เซิร์ฟเวอร์ (Client/Sever)

ระบบฐานข้อมูลประกอบด้วยส่วนประกอบสองส่วน คือ ระบบการจัดการฐานข้อมูล (Database Management System (DBMS)) ซึ่งเป็นโปรแกรมที่ใช่เพื่อจัดระเบียบและบำรุงรักษารายการของข้อมูล อีกส่วนหนึ่งคือแอปพลิเคชันฐานข้อมูล (Database Application) เป็นโปรแกรมที่ช่วยนำข้อมูลออกมา เพื่อนำเสนอและแก้ไขข้อมูลที่ถูกเก็บอยู่ใน DBMS ในปัจจุบันเทคโนโลยีในกาปฏิวัติ DBMS คือ เทคโนโลยีผู้รับ-ผู้ให้บริการ (C/S) ระบบฐานข้อมูลแบบผู้รับ-ผู้ให้บริการ ได้เพิ่มประสิทธิภาพในการจัดการข้อมูลด้วยการแยกส่วนของ DBMS ออกจากส่วนของแอปพลิเคชัน ฐานข้อมูลแอปพลิเคชันทำงานอยู่บนเครื่องเวิร์กสเตชัน (Work station) ของผู้ใช้ในหนึ่งหรือหลายเครื่องและติดต่อถึงกันโดยผ่านระบบเน็ตเวิร์กซึ่งมีระบบ DBMS หนึ่งหรือหลายระบบทำงานอยู่บนเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ระบบฐานข้อมูลแบบผู้รับ-ผู้ให้บริการ เป็นวิธีที่ใช้เครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ระบบฐานข้อมูลผู้รับ-ผู้ให้บริการเป็นวิธีที่ใช้เครื่องคอมพิวเตอร์ที่ประสิทธิภาพดีที่สุดในปัจจุบันและยังสามารถรองรับความซับซ้อนได้เป็นอย่างมาก เช่น มีเครื่องเซิร์ฟเวอร์ 1 เครื่อง มีเครื่องไคลเอนต์ 5 เครื่อง และเครื่องพิมพ์ 1 เครื่อง การทำงานแบบไคลเอนต์/เซิร์ฟเวอร์ (Client/Server) มีประโยชน์ต่อองค์กรที่มีผู้ใช้จำนวนมากและต้องการเข้าถึงข้อมูลอย่างต่อเนื่อง ซึ่งระบบเครือข่ายไคลเอนต์/เซิร์ฟเวอร์ เป็นระบบที่มีประสิทธิภาพสูงในการเข้าถึงและจัดการฐานข้อมูลสำหรับโปรแกรมประยุกต์ต่าง ๆ

สถาปัตยกรรมไคลเอนต์-เซิร์ฟเวอร์ แบบ 2 ระดับ (Two-Tier Client-Server Architecture) เป็นรูปแบบการทำงานของระบบงานที่มีการแบ่งเทียร์ออกเป็นสองส่วน โดยทั้งสองส่วนทำงานประสานกัน เพื่อให้ระบบงานสามารถทำงานสำเร็จตามวัตถุประสงค์ โดยการประมวลผลที่เกี่ยวข้องกับการติดต่อกับผู้ใช้งาน ทั้งเซิร์ฟเวอร์และไคลเอนต์ทำงานสื่อสารกัน ด้วยระบบเครือข่ายคอมพิวเตอร์โดยอาศัยรูปแบบโปรโตคอล (Protocol) การสื่อสารทั้งที่ไคลเอนต์ และเซิร์ฟเวอร์ได้ตกลงกันไว้ เป็นไปได้เหมือนกันที่เครื่องคอมพิวเตอร์เครื่องเดียวจะติดตั้งเทียร์ทั้งในส่วนเซิร์ฟเวอร์และไคลเอนต์ไว้ในเครื่องเดียวกัน ซึ่งส่วนใหญ่ใช้ในการทดสอบหรือทดลองการทำงานก่อนการใช้งานจริง

สถาปัตยกรรมไคลเอนต์-เซิร์ฟเวอร์ แบบ 3 ระดับ (Three-Tier Client-Server Architecture) เพื่อแก้ปัญหาของทูเทียร์จึงเพิ่มจากสองเทียร์เป็นสามเทียร์ โดยในแบบทูเทียร์ เดิมไคลเอนต์จะติดต่อโดยตรงกับฐานข้อมูล หากมีการเปลี่ยนแปลงใด ๆ เกิดขึ้นในฐานข้อมูล การแสดงผลทางด้านไคลเอนต์จำเป็นต้องเปลี่ยนแปลงตามไปด้วย ในการแก้ปัญหานี้จะเพิ่มเทียร์ใหม่เข้ามาขั้นระหว่างไคลเอนต์และเซิร์ฟเวอร์โดยไคลเอนต์จะติดต่อกับเซิร์ฟเวอร์ผ่านทางออบเจ็คที่อยู่บนมิดเดิลเทียร์ (Middle Tier) จากนั้นมิดเดิลเทียร์จะติดต่อกับเซิร์ฟเวอร์ โดยไคลเอนต์จะเห็นเฉพาะออบเจ็คในมิดเดิลเทียร์เท่านั้น การเปลี่ยนแปลงใด ๆ จะทำผ่านมิดเดิลเทียร์เท่านั้น โดยเทียร์ในส่วนนี้มีชื่อเรียกทั่วไปว่าแอปพลิเคชันเซิร์ฟเวอร์ (Application Server)

2.4.2 ระบบฐานข้อมูล (Database System)

ฐานข้อมูล (Database) มีบทบาทสำคัญมากต่อหน่วยงานด้านต่าง ๆ โดยเฉพาะอย่างยิ่งกับระบบงานที่ใช้คอมพิวเตอร์ ความหมายของฐานข้อมูล (Database) คือ การจัดเก็บข้อมูลอย่างมีระบบ ซึ่งผู้ใช้สามารถเรียกใช้ข้อมูลในลักษณะต่าง ๆ ได้ ข้อมูลเป็นส่วนประกอบที่สำคัญอันนำมาซึ่งระบบสารสนเทศ คือ การแปลงข้อมูลให้อยู่ในรูปสารสนเทศที่ใช้เป็นกลยุทธ์ที่สำคัญที่จะทำให้องค์กรบรรลุเป้าหมายที่กำหนดไว้ ดังนั้นข้อมูลก็เปรียบเสมือนวัตถุดิบ สารสนเทศก็เปรียบเสมือนกระบวนการผลิต ที่สามารถดึงข้อมูลออกมาใช้ประโยชน์จากแหล่งจัดเก็บที่เรียกว่า ฐานข้อมูล(Databases) ในโครงสร้างข้อมูลในฐานข้อมูล (Data Structure) ซึ่งโครงสร้างข้อมูลจะแสดงถึงข้อมูลต่าง ๆ ที่มีความสัมพันธ์ต่อกันภายในข้อมูลนั้น การจัดการฐานข้อมูล นับว่ามีความสำคัญต่อความสำเร็จในการใช้ข้อมูลเป็นอย่างยิ่ง ใน

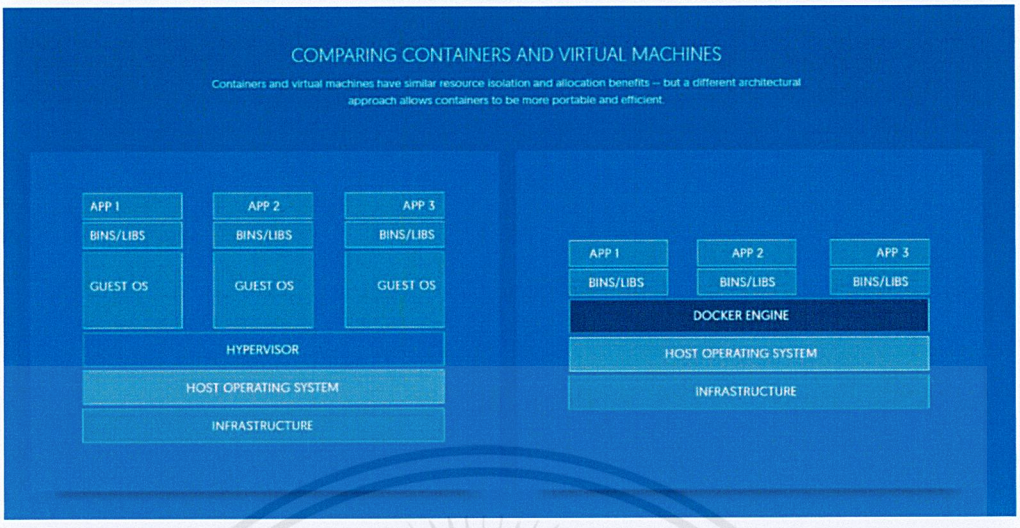
ด้านการจัดการ ผู้จัดการฐานข้อมูลจะเป็นผู้ดูแลพจนานุกรมข้อมูล และให้คำแนะนำหรือชี้แนะการพัฒนาโปรแกรมประยุกต์ใหม่ ๆ

2.5 แนวคิดเกี่ยวกับระบบด็อกเกอร์ (Docker)

ด็อกเกอร์เป็นเครื่องมือทางซอฟต์แวร์ที่สร้างขึ้นมาเพื่อช่วยให้นักพัฒนาสามารถจัดการกับแอปพลิเคชันได้ง่ายขึ้น มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการ run service ที่ต้องการ ทำงานคล้ายคลึงกับ Virtual Machine ซึ่งมีรายละเอียดดังต่อไปนี้

2.5.1 ความแตกต่างระหว่าง Virtual Machine กับ container (docker)

Docker คือ เครื่องมือทางซอฟต์แวร์ที่มีการทำงานในลักษณะการจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการ run service ที่ต้องการ มีการทำงานคล้ายคลึงกับ Virtual Machine เช่น VMWare, VirtualBox, XEN, KVM แต่ข้อแตกต่างที่ชัดเจนคือ Virtual Machine ที่รู้จักกันก่อนหน้านี้ เป็นการจำลองทั้ง OS เพื่อใช้งานและหากต้องการใช้งาน service ใด ๆ จะต้องทำการติดตั้งเพิ่มเติมบน OS นั้นๆ แต่สำหรับ docker แล้วจะใช้ container ในการจำลองสภาพแวดล้อมขึ้นมาเพื่อใช้งานสำหรับ 1 service ที่ต้องการใช้งานเท่านั้น โดยไม่ต้องมีส่วนของ OS เข้าไปเกี่ยวข้องเหมือน Virtual Machines อื่น ๆ ดังภาพที่ 2.3



ภาพที่ 2.3 ความแต่งต่างระหว่าง Virtual Machine กับ Container

2.5.2 การทำงานและสถาปัตยกรรมของด็อกเกอร์ (docker engine)

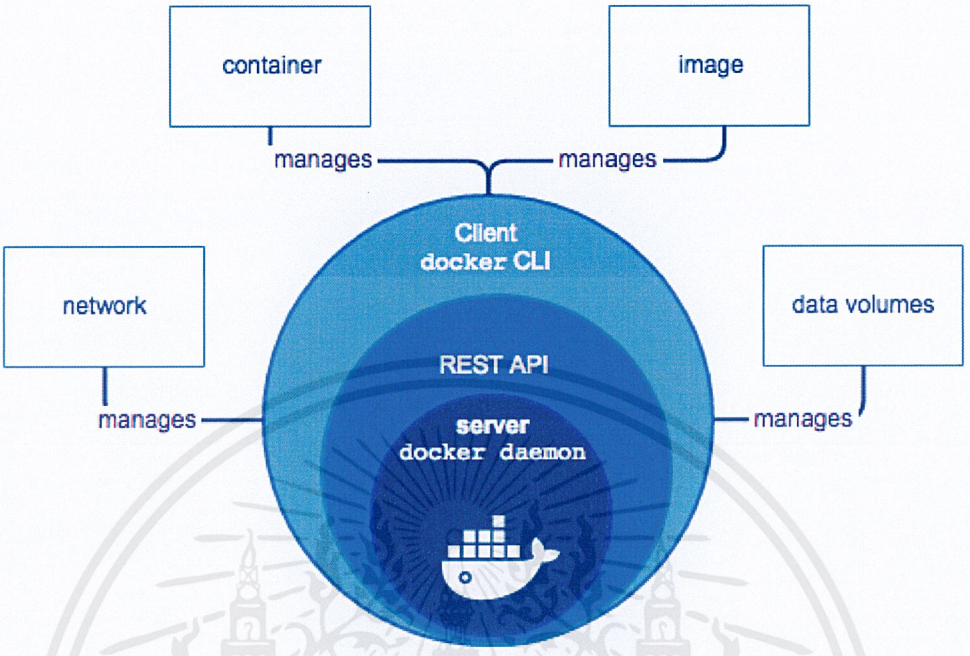
ส่วนที่สำคัญที่สุดของระบบด็อกเกอร์คือ docker engine โดยมีองค์ประกอบและรายละเอียด ดังนี้

2.5.2.1 Docker engine หรือที่เรียกกันว่า docker เป็นส่วนสำคัญของระบบ docker ที่มีการทำงานแบบแอปพลิเคชันไคลเอนต์-เซิร์ฟเวอร์ (client-server application) มีองค์ประกอบ 3 ส่วนที่มีการทำงานเกี่ยวเนื่องกัน ดังภาพที่ 2.4 โดยมีรายละเอียดดังนี้

1) Docker Daemon ทำหน้าจัดการ Docker images, containers, network และ storage volumes โดยจะรับฟังคำขอ (request) ผ่าน Docker API และทำการประมวลผลคำขอนั้น

2) Docker Engine REST API คือ API (Application programming Interface) ที่แอปพลิเคชันใช้ เพื่อโต้ตอบกับ Docker daemon และใช้ไคลเอนต์ HTTP ในการเข้าถึง

3) Docker CLI (Command Line Interface) คือ ไคลเอนต์ (Client) ที่ใช้สำหรับการโต้ตอบกับ Docker daemon ซึ่งเป็นวิธีการลดความซับซ้อนในการจัดการกับอินสแตนซ์คอนเทนเนอร์ (container instances)



ภาพที่ 2.4 การทำงานของด็อกเกอร์

ภาพที่ 2.5 สถาปัตยกรรมด็อกเกอร์ (Docker architecture) มีการทำงานในรูปแบบไคลเอนต์-เซิร์ฟเวอร์ (client-server model) ประกอบไปด้วย Docker Client, Docker Host, Docker Objects และ Docker Registries ซึ่งมีรายละเอียดดังนี้

2.5.2.2 Docker Client เป็นวิธีหลักในการโต้ตอบระหว่างผู้ใช้กับด็อกเกอร์ ซึ่งเมื่อใช้คำสั่ง (command) เช่น docker run ไคลเอ็นต์จะส่งคำสั่งเหล่านี้ไปยัง dockerd (docker daemon) เพื่อดำเนินการคำสั่งเหล่านี้ โดยใช้ Docker API เป็นสื่อกลาง และสามารถสื่อสารกับ daemon ได้มากกว่าหนึ่งตัว

2.5.2.3 Docker Host ทำหน้าที่จัดเตรียมสภาพแวดล้อม (environment) เพื่อ execute และ run แอปพลิเคชัน ประกอบไปด้วย Docker daemon, Images, Containers, Networks และ Storage ซึ่งตามที่ได้กล่าวไว้ก่อนหน้านี้ daemon จะรับผิดชอบการกระทำทั้งหมดที่เกี่ยวข้องกับคอน

เทนเนอร์และรับคำสั่งผ่าน CLI (command line interface) หรือ REST API นอกจากนี้ยังสามารถสื่อสารกับ daemon อื่น ๆ เพื่อจัดการเซอร์วิสของมันได้

2.5.2.4 Docker Objects ซึ่งวัตถุของด็อกเกอร์ (Docker objects) ที่มีการใช้งานร่วมกันมี 4 วัตถุดังนี้

1) อิมเมจ (Images) เป็นไบนารีเทมเพลต (binary template) แบบ read-only เพื่อสร้างคอนเทนเนอร์ และปรับแต่ง หรือ เพิ่มองค์ประกอบเพิ่มเติม โดยประกอบไปด้วย เมตาดาต้า (metadata) ที่อธิบายถึงความสามารถและความต้องการของคอนเทนเนอร์ และสามารถแชร์อิมเมจนั้น ๆ กับทีมภายในองค์กรโดยใช้คอนเทนเนอร์รีจิสตรี (registry) ส่วนตัว หรือรีจิสตรีสาธารณะ (public registry) เช่น Docker Hub ได้

2) คอนเทนเนอร์ (container) เป็นอินสแตนซ์ (instance) ที่ใช้รัน (run) อิมเมจ สามารถ create, start, stop, move หรือ delete คอนเทนเนอร์ ได้โดยใช้ command line interface (CLI) ผ่าน Docker API สามารถเชื่อมต่อคอนเทนเนอร์เข้ากับเครือข่าย (network) หนึ่ง หรือ หลายเครือข่าย โดยคอนเทนเนอร์จะถูกกำหนดโดยอิมเมจของมันรวมถึงการกำหนดค่าใด ๆ

3) เครือข่าย (Networking) เป็นทางผ่านที่ใช้แยกทุกการสื่อสารของคอนเทนเนอร์ โดยแบ่งออกเป็นประเภทหลักได้ ดังนี้

- None Network หรือเครือข่ายแบบปิด หมายถึงไม่ต้องการเชื่อมต่อกับอินเทอร์เน็ต และไม่สามารถเข้าถึงเครือข่ายจาก host ได้ จึงจัดได้ว่าเป็นเครือข่ายที่มีความปลอดภัยสูงสุด

- Host Network จัดได้ว่าเป็นเครือข่ายที่มีความปลอดภัยน้อยที่สุด เนื่องจากคอนเทนเนอร์จะรันอยู่บน host network ซึ่งสามารถเข้าถึงได้เต็มที่ (full access) จาก host interface จึงเรียกคอนเทนเนอร์เหล่านี้ว่า คอนเทนเนอร์เปิด (open containers)

- Bridge Network เป็นเครือข่ายเริ่มต้น (default network) สามารถเข้าถึงได้จาก host อีกทั้งยังสามารถใช้งานอินเทอร์เน็ตจากภายนอกคอนเทนเนอร์ได้

- Overlay Network เป็นเครือข่ายที่ใช้เมื่อต้องการสื่อสารกันระหว่างคอนเทนเนอร์ที่มีการทำงานบน host ต่างกัน (swarm service)

- Macvlan Network เครือข่ายนี้จะกำหนด mac address ให้กับคอนเทนเนอร์ให้ดูเหมือนอุปกรณ์ทางกายภาพ (physical devices) และสื่อสารระหว่างกันผ่าน mac address

4) Storage ด็อกเกอร์มีทางเลือกในการเก็บข้อมูล 4 ทางให้แก่ผู้ใช้งานได้เลือกใช้ดังนี้

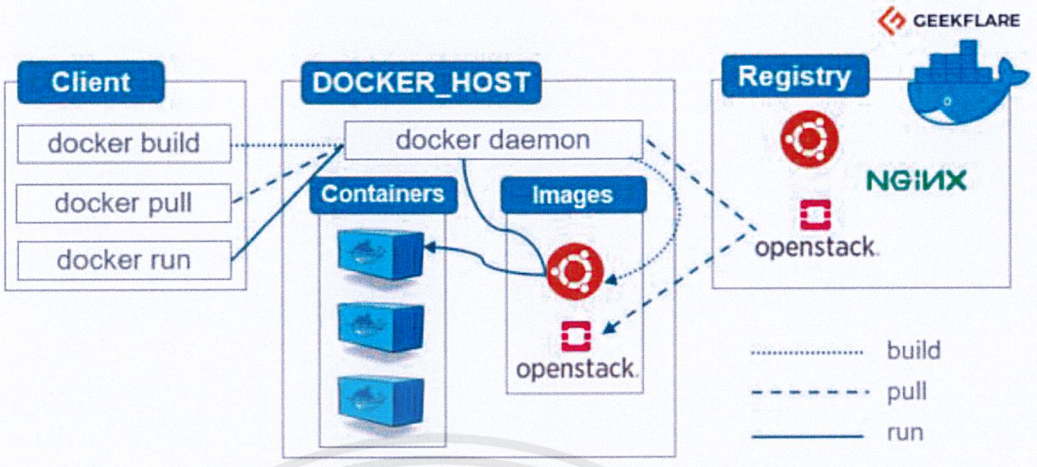
- Data volumes ให้ความสามารถในการสร้างหน่วยเก็บข้อมูลถาวร (persistent storage) ด้วยการ rename และ list volumes รวมถึงคอนเทนเนอร์ที่มีส่วนเกี่ยวข้องกับ volume นั้น

- Data volumes container เป็นอีกทางเลือกให้คอนเทนเนอร์ที่ถูกแยกกันด้วย host และต้องการเชื่อมต่อไดรฟ์นั้นกับคอนเทนเนอร์อื่น ในกรณีที่ไดรฟ์ข้อมูลคอนเทนเนอร์เป็นอิสระจากแอปพลิเคชัน และจะสามารถใช้ร่วมกันได้มากกว่าหนึ่งคอนเทนเนอร์

- Directory mounts อีกหนึ่งทางเลือกคือการกำหนด local directory ของ host ลงในคอนเทนเนอร์ ในกรณีก่อนหน้า ข้อมูลจะอยู่ในโฟลเดอร์ Docker volumes ในขณะที่กรณีนี้ มีการติดตั้ง directory บนเครื่องโฮส (host machine) และสามารถใช้เป็นแหล่งที่มา (source) สำหรับข้อมูลได้

- Storage plugins ให้ความสามารถในการเชื่อมต่อกับแพลตฟอร์มหน่วยเก็บข้อมูลภายนอก (external storage) ปลั๊กอินเหล่านี้จะทำการแม็พ (map) ที่เก็บข้อมูลจากโฮสต์ไปยังแหล่งที่มา (source) ภายนอก

2.5.2.5 Docker Registries เป็นเซิร์ฟเวอร์ไว้เก็บรวบรวม Docker image (ลักษณะเดียวกับการเก็บ Source Code ไว้บน Github) โดย Docker registry ณ ปัจจุบันก็มีให้เลือกใช้งานอย่างหลากหลาย โดยจะมี Docker Hub เป็น Docker registry หลักในการเรียกใช้ (pull) Docker Image และนอกจากนี้ยังมีผู้ให้บริการ docker registry เจ้าอื่น ๆ ด้วย เช่น Gitlab, Quay.io, Google Cloud เป็นต้น



ภาพที่ 2.5 โครงสร้างของด็อกเกอร์ (Docker Architecture)

2.5.3 จุดเด่นและข้อดีของด็อกเกอร์

ด็อกเกอร์มีจุดเด่นในหลาย ๆ เรื่อง ไม่ว่าจะเป็นเรื่องความหลากหลายของแพลตฟอร์มที่สามารถใช้งานได้ หรือ ประสิทธิภาพในการทำงาน ซึ่งมีรายละเอียดดังนี้

- เครื่องด็อกเกอร์ (Docker engine) สามารถใช้งานได้บนหลากหลายแพลตฟอร์ม ทั้ง Linux, Mac และ Windows

- ด็อกเกอร์มีขนาดที่เล็ก สามารถใช้งานและติดตั้งได้อย่างรวดเร็ว อีกทั้งยังสะดวกในการ start/stop หรือแม้แต่การเคลื่อนย้ายไปใช้งานบนเครื่องเซิร์ฟเวอร์อื่น ที่มี docker engine ทำงานอยู่ ซึ่งทำได้อย่างไม่ซับซ้อน

- ด็อกเกอร์มีความต้องการในการใช้ CPU, RAM และพื้นที่ที่น้อยกว่า Virtual Machine อีกทั้งยังสามารถทำงานได้อย่างมีประสิทธิภาพมากกว่า Virtual Machine ในทรัพยากรที่มีเท่ากัน

- เนื่องจากผู้ใช้งานสามารถสร้าง docker image ได้เองจาก dockerfile ดังนั้นการใช้งาน docker จึงสามารถลดปัญหาเรื่องสภาพแวดล้อม (environment) ที่ต่างกัน อย่างเช่น ในบางแอปพลิเคชันที่ทำงานได้บน development server แต่ไม่สามารถทำงานได้บน production server

- ด็อกเกอร์มี docker registry ที่ผู้ใช้งานสามารถเลือก pull image ต่าง ๆ ที่มีการสร้างไว้แล้วมาใช้งานได้ และ สามารถ push image ที่เราสร้างไว้เองขึ้นไปได้ โดยมี Docker Hub เป็น registry หลักในการเรียกใช้ image

2.6 ทฤษฎีอื่น ๆ ที่เกี่ยวข้อง

2.6.1 ระบบปฏิบัติการลินุกซ์ (Linux Operation System)

ลีนุกซ์ เป็นระบบปฏิบัติการเช่นเดียวกับ ดอส ไมโครซอฟต์วินโดวส์ หรือ ยูนิกซ์ โดยลีนุกซ์นั้นจัดว่าเป็นระบบปฏิบัติการยูนิกซ์ประเภทหนึ่ง การที่ลีนุกซ์เป็นที่กล่าวขานกันมากในขณะนี้ เนื่องจากความสามารถของตัวระบบปฏิบัติการและโปรแกรมประยุกต์ที่ทำงานบนระบบลีนุกซ์ โดยเฉพาะโปรแกรมในตระกูลของ GNU (GNU's Not UNIX) และสิ่งที่สำคัญที่สุดก็คือ ระบบลีนุกซ์เป็นระบบปฏิบัติการประเภทฟรีแวร์ (Free Ware) นั่นคือไม่เสียค่าใช้จ่ายในการซื้อโปรแกรม

ระบบลีนุกซ์ตั้งแต่เวอร์ชั่น 4 นั้น สามารถทำงานได้บนซีพียูทั้ง 3 ตระกูล คือ บนซีพียูของอิลเทล (PC Intel) ดิจิตอลอัลฟาคอมพิวเตอร์ (Digital Alpha Computer) และซันสปาร์ค (SUN SPARC) เนื่องจากใช้เทคโนโลยีที่เรียกว่า RPM (Red Hat Package Management) ถึงแม้ว่าในขณะนี้ลีนุกซ์ยังไม่สามารถแทนที่ไมโครซอฟต์ วินโดวส์ บนพีซี หรือ แมคโอเอส (Mac OS) ได้ทั้งหมดก็ตาม แต่ก็มีผู้ใช้จำนวนไม่น้อยที่หันมาใช้และช่วยพัฒนาโปรแกรมประยุกต์บนลีนุกซ์กัน และเรื่องของการดูแลระบบลีนุกซ์นั้น ภายในระบบลีนุกซ์เองมีเครื่องมือช่วยสำหรับดำเนินการให้สะดวกยิ่งขึ้น

ปัจจุบันได้มีการนำระบบปฏิบัติการลีนุกซ์ไปประยุกต์เป็นระบบปฏิบัติการสำหรับงานด้านต่าง ๆ เช่นงานด้านการคำนวณทางวิทยาศาสตร์ใช้เป็นสถานีงาน สถานีบริการ อินเทอร์เน็ต อินทราเน็ต หรือ ใช้ในการเรียนการสอนและการทำวิจัยทางคอมพิวเตอร์ ใช้พัฒนาโปรแกรม เนื่องจากมีเครื่องมือมากมาย เช่น โปรแกรมภาษาซี (C) ซีพลัสพลัส (C++) ปาสคาล (Pascal) ฟอร์แทรน (Fortran) ลิสป์ (Lisp) โปรล็อก (Prolog) เอดา (ADA) มีภาษาสคริปต์ เช่น เชลล์ (Shell) บาสซ์เชลล์ (Bash Shell) ซีเชลล์ (C Shell) คอร์นเชลล์ (Korn Shell) เพิร์ล (Perl) พายตัน (python) TCL/TK

ก่อนที่จะทำการติดตั้งก็ต้องเตรียมความพร้อมทางด้านอุปกรณ์ฮาร์ดแวร์และซอฟต์แวร์ให้เป็นที่เรียบร้อยก่อน ระบบลีนุกซ์ต้องการฮาร์ดแวร์ที่มีคุณสมบัติขั้นต่ำสุดดังต่อไปนี้

1) หน่วยประมวลผลกลางของ Intel 80386 ขึ้นไป

2) หน่วยประมวลผลทางคณิตศาสตร์ มีหรือไม่มีก็ได้ เพราะระบบปฏิบัติการ Red Hat Linux ได้มีการจำลอง หน่วยประมวลผลทางคณิตศาสตร์ไว้ในระดับของเคอร์เนล (Kernel) แล้ว

3) หน่วยความจำอย่างน้อย 8 เมกะไบต์ แต่แนะนำให้มีอย่างน้อย 16 เมกะไบต์จะทำให้ระบบมีประสิทธิภาพที่ดีกว่า

4) ฮาร์ดดิสก์อย่างน้อย 101 เมกะไบต์ สำหรับการติดตั้งแบบพื้นฐาน 266 เมกะไบต์ สำหรับการติดตั้งแบบทั่วไป และ 716 เมกะไบต์ สำหรับการติดตั้งแบบทั้งหมด ตัวเลขที่ระบุทั้งหมดเฉพาะส่วนระบบปฏิบัติการ ถ้าต้องการใช้เป็น File Server หรือ Database Server จะต้องสำรองพื้นที่ไว้สำหรับใช้งานด้วย

บทที่ 3

วิธีการดำเนินงาน

งานชิ้นนี้เป็นการสร้างแผนภาพแสดงเพื่อแสดงสถานะและข้อมูลที่เกี่ยวข้องกับการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลของแต่ละแอปพลิเคชัน โดยขั้นตอนการทำงานต่าง ๆ จะต้องเริ่มทำความเข้าใจตั้งแต่ ปัญหาหรือข้อบกพร่องที่เกิดขึ้น วิธีการค้นหาต้นเหตุของข้อผิดพลาดหรือปัญหาดังกล่าว วิธีการแบบใหม่ที่จะเข้ามาเพื่อเพิ่มประสิทธิภาพให้กับวิธีการแบบเดิม และหลังจากที่ได้ทำความเข้าใจวิธีการทำงานและปัญหาแล้ว นำมาซึ่งกระบวนการการดำเนินงานทั้งหมดเพื่อสร้างเป็นแผนภาพแสดงสถานะและพฤติกรรมของระบบ โดยมีขั้นตอนการดำเนินงานซึ่งแบ่งเป็น 4 ส่วน ดังต่อไปนี้

3.1 กระบวนการเข้าใจปัญหาที่เกิดขึ้นของระบบการทำงานและออกแบบระบบ

การวิเคราะห์ปัญหาและการทำงานของระบบ จะต้องมีความเข้าในการทำงานของการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล และปัญหาหรือข้อบกพร่องที่อาจเกิดขึ้นและส่งผลกระทบต่อลูกค้า รวมถึงปัญหาในการค้นหาต้นเหตุของข้อผิดพลาดหรือปัญหาดังกล่าวในวิธีแบบเดิม และวิธีการที่จะเพิ่มประสิทธิภาพของวิธีการแบบเดิม โดยมีกระบวนการดังนี้

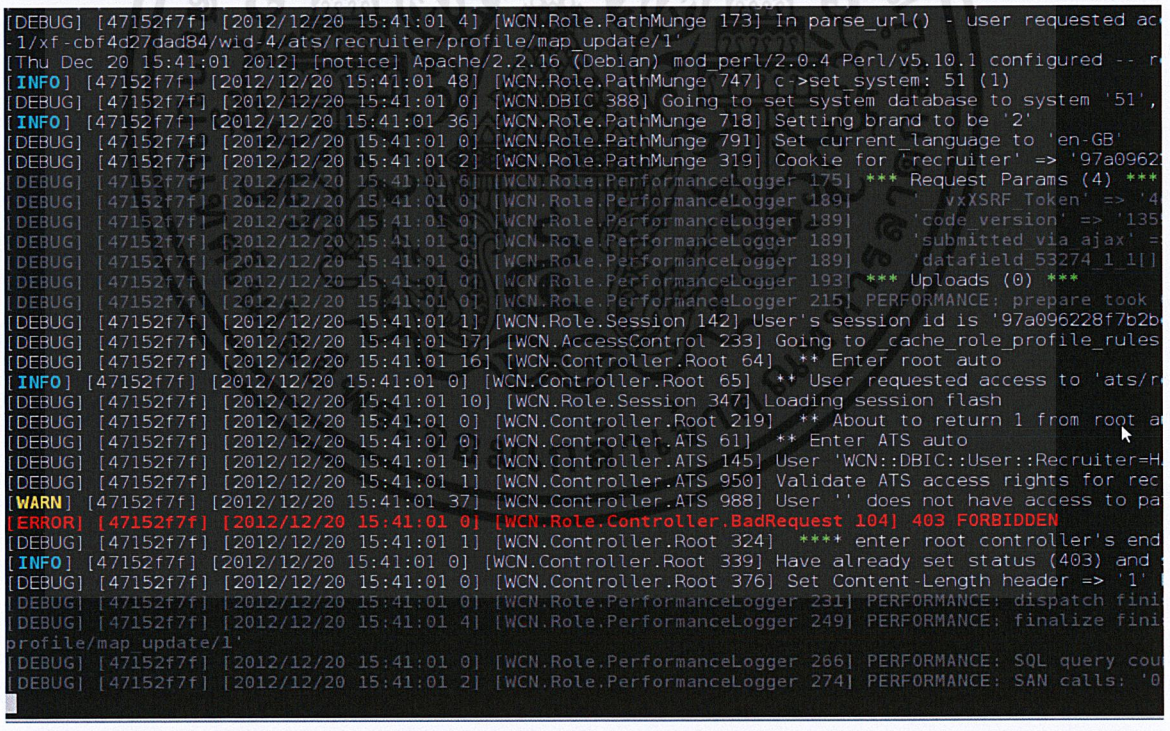
3.1.1 การศึกษาปัญหาและผลกระทบที่ลูกค้าได้รับ

การทำความเข้าใจถึงปัญหาที่เกิดขึ้นกับลูกค้าที่ได้รับผลกระทบจากปัญหาหรือข้อผิดพลาดจากการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล ซึ่งสามารถเกิดขึ้นได้จากหลากหลายปัจจัย โดยมีปัจจัยหลัก ๆ ได้แก่ ความผิดปกติทางการจราจรทางอินเทอร์เน็ต (Network traffic) หรือ สมรรถนะของเครื่องที่เกี่ยวข้อง เช่น ความจุของเครื่อง และ ความผิดปกติทาง Infrastructure เช่น เครื่องดับหรือดาวน์ (down) เป็นต้น ซึ่งปัจจัยที่กล่าวมาข้างต้นนั้น เมื่อมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นก็จะทำให้การซิงค์

ข้อมูลมีปัญหา ฐานข้อมูลในแต่ละศูนย์กลางข้อมูลมีข้อมูลไม่เหมือนกัน ทำให้ลูกค้าได้รับผลกระทบจากการปัญหาหรือข้อผิดพลาดในครั้งนี้

3.1.2 การศึกษาปัญหาของการค้นหาต้นเหตุที่แท้จริงของปัญหาแบบเดิม

การศึกษาและทำความเข้าใจถึงปัญหาของการค้นหาต้นเหตุที่แท้จริงของปัญหาหรือข้อผิดพลาดในแบบเดิม เพื่อให้สามารถแก้ไขปัญหาหรือเพิ่มประสิทธิภาพได้อย่างตรงจุด โดยวิธีการแบบเดิมนั้น ใช้ระยะเวลาค่อนข้างมากจากการใช้วิธีการค้นหาข้อมูลจากล็อกไฟล์ เนื่องจากล็อกไฟล์มีรูปแบบเป็นตัวอักษรดังภาพที่ 3.1 ซึ่งในแต่ละแอปพลิเคชันไม่ได้มีแค่เครื่อง server เครื่องเดียวที่ใช้ในการทำงาน อีกทั้งเครื่อง server หนึ่งเครื่องมีล็อกไฟล์อีกหลายประเภท จึงเป็นที่มาของจุดบกพร่องของการใช้ทรัพยากรทางเวลาในวิธีการแบบเดิม



ภาพที่ 3.1 ตัวอย่างรูปแบบของล็อกไฟล์

3.1.3 การศึกษาแนวคิดการจัดการล็อกร่วมกับอีเอลเคสแต็ค

การศึกษาและทำความเข้าใจแนวคิดการจัดการล็อก เพื่อเพิ่มประสิทธิภาพในการค้นหาต้นเหตุที่แท้จริงของปัญหาหรือข้อผิดพลาดที่เกิดจากการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล โดยการรวบรวมล็อกไฟล์จากทุก ๆ แหล่งที่มา มาไว้ที่เครื่องที่เป็นศูนย์กลางในการจัดเก็บข้อมูลล็อก โดยใช้ไฟล์บีทเป็นเครื่องมือจับล็อกร่วมกับชุดเครื่องมืออีแอลเคสแต็ค

3.2 กระบวนการศึกษาแอปพลิเคชันขององค์กรที่มีการซิงค์ข้อมูล

การศึกษาแอปพลิเคชันของทางองค์กรที่มีการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล จะช่วยให้สามารถเข้าใจโครงสร้างการทำงานของการซิงค์ข้อมูลของฐานข้อมูล ปัญหาหรือข้อผิดพลาดที่อาจเกิดขึ้นไปจนถึงความเข้าใจรายละเอียดของข้อมูลล็อกไฟล์ที่จะนำมาใช้สร้างแผนภาพแสดงสถานะต่าง ๆ ซึ่งแอปพลิเคชันที่นำมาศึกษา มีรายละเอียดดังต่อไปนี้

3.2.1 แอปพลิเคชันเอทีเอ็มแทนเด็ม (ATM-TANDEM)

เป็นระบบที่เกี่ยวข้องกับตู้ ATM ของธนาคารกสิกรไทย ที่ช่วยให้ลูกค้าสามารถทำธุรกรรม ถอน โอน เติม จ่าย ผ่านตู้ ATM ต่าง ๆ ทั่วประเทศ แอปพลิเคชัน ATM ทำงานอยู่บนเครื่อง Mainframe (TANDEM) ที่มีการทำ DR-Site Solution เพื่อช่วยให้ธุรกิจสามารถดำเนินไปได้อย่างต่อเนื่อง หาก Primary-Site เกิดปัญหา ผลกระทบที่เกิดหากระบบมีปัญหา คือ ลูกค้าไม่สามารถทำธุรกรรมทางการเงินผ่านตู้ ATM ของธนาคารกสิกรไทย เช่น การถอนเงิน การโอนเงิน ได้ หรืออาจทำรายได้ล่าช้า ซึ่งส่งผลต่อการทำธุรกรรมทางการเงินหลักของธนาคารกสิกรไทย

3.2.2 แอปพลิเคชัน Online Retail Payment Gateway (ORPG)

แอปพลิเคชัน ORPG (Online Retail Payment Gateway) เป็นหนึ่งในระบบของธนาคารกสิกรไทยบนแอปพลิเคชัน KPLUS ที่เชื่อมต่อไปยังผู้ประกอบการค้า (Merchant) เจ้าอื่น ๆ ของธนาคารกสิกรไทย เช่น AIS, TRUE, DTAC และ PEA (การไฟฟ้าแห้งประเทศไทย) เป็นต้น เพื่อดำเนินธุรกรรมทางธนาคาร เช่น เติมเงินโทรศัพท์ จ่ายบิลค่าไฟฟ้า และ จ่ายบิลค่าโทรศัพท์

แอปพลิเคชัน ORPG ใช้ฐานข้อมูลของ MariaDB ในการเก็บข้อมูลเกี่ยวกับรายการธุรกรรมทางการเงิน (Transaction) ในรูปแบบ Garela Cluster เพื่อเพิ่มความ High Availability ให้สามารถในการเก็บรายการธุรกรรมทางการเงินได้ตลอดเวลา พร้อมกับประสิทธิภาพที่สูง

ผลกระทบที่เกิดหากระบบมีปัญหา คือ ลูกค้าไม่สามารถทำธุรกรรมจำพวกการจ่ายบิลและการเติมเงิน บนแอปพลิเคชัน KPLUS ได้ (รายการในหมวด "จ่ายบิล" กับ "เติมเงิน") หรืออาจทำได้ล่าช้า ส่งผลต่อการทำธุรกรรมทางการเงินของธนาคารกสิกรไทยและส่งผลกระทบต่อผู้ประกอบการค้าเจ้าอื่น ๆ ที่ธนาคารกสิกรไทยเป็นหุ้นส่วน (Partner) ด้วย

3.2.3 แอปพลิเคชันเคพลัส (KPLUS)

แอปพลิเคชัน KPLUS คือ แอปพลิเคชันธนาคารบนมือถือ (Mobile Banking Application) ของธนาคารกสิกรไทย ใช้สำหรับทำธุรกรรมทางธนาคารผ่านโทรศัพท์มือถือ ทั้งรายการโอนเงิน เติมเงิน และจ่ายบิลต่าง ๆ ช่วยให้ลูกค้าสามารถทำธุรกรรมต่าง ๆ ได้ทุกที่ทุกเวลา ซึ่งมีการทำธุรกรรมหลายล้านรายการภายในหนึ่งวัน

แอปพลิเคชัน KPLUS ใช้ฐานข้อมูล Microsoft SQL (MSSQL) ในการเก็บข้อมูลของรายการการทำธุรกรรม (Transaction) ทั้งหมด ที่ผ่านการใช้งานบนแอปพลิเคชัน KPLUS ซึ่งจะต้องเก็บไว้ตามนโยบาย (Policy) คือ 10 ปี และต้องสามารถสืบค้นออกมาได้

ผลกระทบที่เกิดหากระบบมีปัญหา คือ ลูกค้าจะไม่สามารถทำธุรกรรมทางธนาคารผ่านแอปพลิเคชัน KPLUS ได้เลย หรืออาจทำรายได้ล่าช้า หรือไม่สำเร็จ ซึ่งส่งผลต่อการทำธุรกรรมทางการเงินหลักของธนาคารกสิกรไทย

3.3 กระบวนการออกแบบแผนงานและขั้นตอนการทำงาน

การออกแบบแผนงานในกระบวนการนี้ มีขึ้นเพื่อตรวจสอบความเข้าใจของโครงสร้างการทำชิ้นงานนี้ และเพื่อจัดลำดับการทำงานได้อย่างถูกต้อง รวมถึงการเริ่มกระบวนการทำงาน โดยมีรายละเอียดดังนี้

3.3.1 ออกแบบแผนภาพโครงสร้างขั้นตอนการทำงาน

หลังจากกระบวนการศึกษาเพื่อทำความเข้าใจเกี่ยวกับงานชิ้นนี้แล้ว ต่อไปเป็นขั้นตอนการออกแบบกระบวนการทำงานโดยภาพรวม ดังภาพที่ 3.2 เพื่อให้ได้ผลลัพท์เป็นแผนภาพแสดงสถานะต่าง ๆ ของแอปพลิเคชันที่มีการใช้งานการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล โดยมีขั้นตอนดังนี้

1) จัดเตรียมข้อมูลล็อกไฟล์

2) เริ่มจากการติดตั้ง beats ประเภท filebeat ไว้ในแต่ละเครื่องที่ต้องการนำข้อมูลล็อกมาใช้งาน และเขียนการตั้งค่าให้กับ filebeat

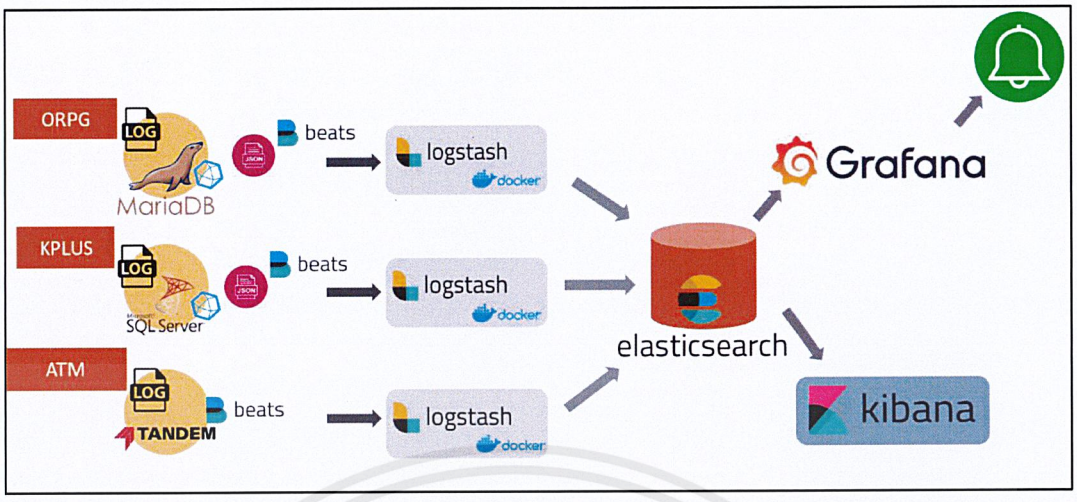
3) ติดตั้งและเขียนการตั้งค่า Logstash ให้ตรงตามความต้องการ

4) ติดตั้งและเขียนการตั้งค่า Elasticsearch และ Kibana

5) เริ่มการทำงาน filebeat และ logstash และส่งข้อมูลที่ถูกคัดกรองแล้วไปยัง Elasticsearch

6) ติดตั้ง Grafana และตั้งค่าต่าง ๆ เพื่อนำข้อมูลจาก Elasticsearch มาแสดงเป็นแผนภาพต่าง ๆ

7) สร้างการแจ้งเตือนผ่าน Line notify จาก การตั้งค่าของ Grafana



ภาพที่ 3.2 แผนภาพโครงสร้างขั้นตอนการทำงาน

3.3.2 วิธีการจัดเตรียมข้อมูลล็อกไฟล์

เนื่องจากแอปพลิเคชันที่นำมาศึกษามีรูปแบบการทำงานที่ต่างกัน จึงมีวิธีในการจัดเตรียมล็อกไฟล์แตกต่างกันไป โดยมีรายละเอียดดังนี้

3.3.2.1 วิธีการจัดเตรียมล็อกไฟล์ของแอปพลิเคชัน ORPG

แอปพลิเคชัน ORPG ใช้เทคโนโลยี MariaDB เป็นฐานข้อมูล ข้อมูลที่เกี่ยวกับการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลนั้นถูกเก็บไว้ในฐานข้อมูล InfluxDB โดยจะต้องทำการ query ข้อมูลจาก InfluxDB เพื่อนำมาสร้างล็อกไฟล์ที่เครื่อง server log โดยมีขั้นตอนดังนี้

1) เนื่องจากเครื่อง server log มี Operation System (OS) เป็น Linux ดังนั้นขั้นตอนต่อไปนี้จะเป็นการเขียน Shell script ดังภาพที่ 3.3 เพื่อใช้ในการรัน (run) คำสั่ง curl เพื่อ query ข้อมูลจาก InfluxDB มาเขียนลงไฟล์ โดยใช้คำสั่ง $vi ตามด้วยชื่อไฟล์ shell script (curl_Influxdb_maria.sh) และบันทึกไว้ที่ path /app/script/

```
CURRENT_DATE=$(date +'%Y-%m-%d')

curl 'http://10.209.47.83:8086/query' -G -u user:passw0rd --data-urlencode
"db=maria_orpg" --data-urlencode "q=SELECT * FROM check_slave WHERE
host='sprdlrorpgdb04' order by time DESC limit 1" >>
/app/script/timesync/result_orpgmariadb-$CURRENT_DATE.json

curl 'http://10.209.47.83:8086/query' -G -u user:passw0rd --data-urlencode
"db=maria_orpg" --data-urlencode "q=SELECT * FROM check_slave WHERE
host='pprdlrorpgdb04' order by time DESC limit 1" >>
/app/script/timesync/result_orpgmariadb-$CURRENT_DATE.json
```

ภาพที่ 3.3 การเขียน shell script เพื่อนำข้อมูลของแอปพลิเคชัน ORPG ออกมา

2) เขียน crontab เพื่อรัน job หรือ query ข้อมูลจาก InfluxDB นี้ ทุก ๆ 1 นาที (* * * * * /app/script/curl_Influxdb_maria.sh >/dev/null 2>&1) โดยใช้คำสั่ง $crontab -e เมื่อทำการแก้ไข crontab เสร็จสิ้น บันทึกเพื่อใช้งาน crontab จะได้ผลลัพท์เป็นข้อมูลที่ได้ทำการ query มาจาก InfluxDB เขียนไฟล์ต่อกัน และแยกเก็บข้อมูลในแต่ละวัน หรือการทำ Archiving จะได้ข้อมูล ดังภาพที่ 3.4


```
{"results":[{"statement_id":0,"series":[{"name":"check_slave","columns":["tim
e","host","master","slave_delay","slave_io_running","slave_io_state","slave_r
unning","slave_sql_running","slave_sql_state"],"values":[["2019-11-
20T16:55:06.592328817Z","sprdlrorpgdb04","pprdlrorpgdb01",0,"Yes","Waiting for
master to send event",0,"Yes","Slave has read all relay log; waiting for the
slave I/O thread to update it"]]}]}]}
{"results":[{"statement_id":0,"series":[{"name":"check_slave","columns":["tim
e","host","master","slave_delay","slave_io_running","slave_io_state","slave_r
unning","slave_sql_running","slave_sql_state"],"values":[["2019-11-
20T16:55:08.010677829Z","pprdlrorpgdb04","172.31.9.200",0,"Yes","Waiting for
master to send event",0,"Yes","Slave has read all relay log; waiting for the
slave I/O thread to update it"]]}]}]}
```


ภาพที่ 3.4 ข้อมูลที่ได้จากการรันสคริปต์ curl_Influxdb_maria.sh

3.3.2.2 วิธีการจัดเตรียมล็อกไฟล์ของแอปพลิเคชัน KPLUS

แอปพลิเคชัน KPLUS ใช้เทคโนโลยี MSSQL เป็นฐานข้อมูล ข้อมูลที่เกี่ยวกับการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลนั้นถูกเก็บไว้ในฐานข้อมูล InfluxDB ซึ่งขั้นตอนของการจัดเตรียมล็อกไฟล์ของแอปพลิเคชันนี้จะมีขั้นตอนที่คล้ายกันกับแอปพลิเคชัน ORPG ที่ได้อธิบายไว้ข้างต้น โดยจะต้องทำการ query ข้อมูลจาก InfluxDB เพื่อนำมาสร้างล็อกไฟล์ที่เครื่อง server log โดยมีขั้นตอนดังนี้

1) เขียน Shell script ดังภาพที่ 3.5 เพื่อใช้ในการรัน (run) คำสั่ง curl เพื่อ query ข้อมูลจาก InfluxDB มาเขียนลงไฟล์ โดยใช้คำสั่ง $vi ตามด้วยชื่อไฟล์ shell script (curl_influxdb_mssql.sh) และบันทึกไว้ที่ path /app/script/

```
curl 'http://10.209.47.83:8086/query' -G -u user:passw0rd --data-urlencode
"db=mssql_km_emobile" --data-urlencode "q=SELECT * FROM ag_estimateddataloss
order by time DESC limit 1" >> /app/script/timesync/result_mssql-
$CURRENT_DATE.json
```

ภาพที่ 3.5 การเขียน shell script เพื่อนำข้อมูลของแอปพลิเคชัน KPLUS ออกมา

2) เขียน crontab เพื่อรัน job หรือ query ข้อมูลจาก InfluxDB นี้ ทุก ๆ 1 นาที (* * * * * /app/script/curl_influxdb_mssql.sh >/dev/null 2>&1) โดยใช้คำสั่ง $crontab -e เมื่อทำการแก้ไข crontab เสร็จสิ้น บันทึกเพื่อใช้งาน crontab จะได้ผลลัพท์เป็นข้อมูลที่ได้ทำการ query มาจาก InfluxDB เขียนไฟล์ต่อกัน และแยกเก็บข้อมูลในแต่ละวัน หรือการทำ Archiving จะได้ข้อมูลดังภาพที่ 3.6


```
{"results":[{"statement_id":0,"series":[{"name":"ag_estimateddataloss","colum
ns":["time","PPRDWKMBEDB01_EMOBILE_consolidated","PPRDWKMBEDB01_EMOBILE_emobi
le","PPRDWKMBEDB01_EMOBILE_emobile_common","PPRDWKMDB07_EMOBILE_consolidated"
,"PPRDWKMDB07_EMOBILE_emobile","PPRDWKMDB07_EMOBILE_emobile_common","PPRDWKMD
B08_EMOBILE_consolidated","PPRDWKMDB08_EMOBILE_emobile","PPRDWKMDB08_EMOBILE_
emobile_common","PPRDWKMDB09_EMOBILE_consolidated","PPRDWKMDB09_EMOBILE_emobi
le","PPRDWKMDB09_EMOBILE_emobile_common","SDRWKMDB07_EMOBILE_consolidated","S
DRWKMDB07_EMOBILE_emobile","SDRWKMDB07_EMOBILE_emobile_common","servername"],
"values":[["2019-11-20T16:59:25.189995725Z",null,null,null,null,null,null,
null,null,null,null,null,null,null,null,0,"PPRDWKMDB07_EMOBILE"]]}]}]}
```


ภาพที่ 3.6 ข้อมูลที่ได้จากการรันสคริปต์ curl_Influxdb_mssql.sh

3.3.2.3 วิธีการจัดเตรียมล็อกไฟล์ของแอปพลิเคชัน ATM

แอปพลิเคชันสุดท้ายที่นำมาศึกษาคือ แอปพลิเคชันเอทีเอ็ม ซึ่งการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลนั้น ใช้ TANDEM Mainframe ดังนั้นขั้นตอนต่อไปนี้จะแนวิธีการดึงข้อมูลที่เกี่ยวข้องกับการซิงค์ข้อมูลจากเครื่องเมนเฟรมมาไว้ที่เครื่อง server log โดยมีวิธีการดังนี้

1) เขียน Shell script ดังภาพที่ 3.7 เพื่อดึงล็อกไฟล์จากเครื่อง TANDEM Mainframe มาเขียนลงไฟล์ที่เครื่อง server log และบันทึกไว้ที่ path /app/script/ และให้ชื่อว่า ftp_tandem.sh

```
#!/bin/bash
HOST=172.31.163.185
USER=userftp
PASSWORD=password
PATH='$ATMD10.GGMON'
DATE=$(/bin/date +"%Y-%m-%d")

/bin/ftp -n $HOST <<EOF
user $USER $PASSWORD
cd $PATH
lcd /app/app_logs/log_tandem
get GGOUT GGOUT.txt
bye
EOF

/bin/cat /app/app_logs/log_tandem/GGOUT.txt >> /app/app_logs/log_tandem/GGOUT-$DATE.txt
```

ภาพที่ 3.7 การเขียน shell script เพื่อนำข้อมูลออกจากเครื่อง TADEM Mainframe

2) เขียน crontab เพื่อรัน job นี้ เพื่อเข้าไปหยิบล็อกไฟล์จากเครื่อง TANDEM ทุก ๆ 1 นาที (* * * * * /app/script/ftp_tamdem.sh >/dev/null 2>&1) โดยใช้คำสั่ง $crontab -e เมื่อทำการแก้ไข crontab เสร็จสิ้น บันทึกเพื่อใช้งาน crontab จะได้ผลลัพท์เป็นข้อมูลล็อกไฟล์ของ TANDEM Mainframe และแยกเก็บข้อมูลในแต่ละวัน หรือการทำ Archiving log จะได้ข้อมูลดังภาพที่ 3.8

```
10/11/2019 00:00:00
\KNB2 Oracle GoldenGate Status
Program   Process   Status      Group       Lag           Time Since Chkpt
Manager   $GGMGR    RUNNING
Replicat  $GGR18    RUNNING     B2400       00:00:00      00:00:00
Replicat  $GGR19    RUNNING     B2401       00:00:01      00:00:14
Replicat  $GGR20    RUNNING     B2402       00:00:00      00:00:00
Replicat  $GGR21    RUNNING     B2403       00:00:00      00:00:08
Replicat  $GGR24    RUNNING     B2404       00:00:00      00:00:00
Replicat  $GGR23    RUNNING     B24TSS      00:00:00      00:00:50
```

ภาพที่ 3.8 ข้อมูลที่ได้จากการรันสคริปต์ ftp_tandem.sh

3.3.3 ขั้นตอนติดตั้งและการตั้งค่าไฟล์บีท

หลังจากที่จัดเตรียมล็อกไฟล์เสร็จ ขั้นตอนต่อไปเป็นการติดตั้งและตั้งค่า filebeat เพื่อประมวลผลข้อมูลและส่งข้อมูลให้ Logstash โดยมีรายละเอียดในแต่ละขั้นตอนดังนี้

3.3.3.1 ขั้นตอนการติดตั้ง filebeat

ในขั้นตอนต่อไปนี้จะเป็นวิธีการติดตั้ง filebeat บนเครื่อง server log บนระบบปฏิบัติการลินุกซ์ (Linux OS) โดยมีขั้นตอนดังนี้

1) ดาวน์โหลด package file ของ filebeat บนเครื่อง server log ซึ่ง ณ ที่นี้เป็นระบบปฏิบัติการลินุกซ์ (Linux OS) โดยการใช้คำสั่ง $curl ดังภาพที่ 3.9 เพื่อดาวน์โหลด package file ของ filebeat (กรณีที่เครื่อง server สามารถเชื่อมต่อกับอินเทอร์เน็ตภายนอกได้)

```
$ curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.5.0-linux-x86_6
4.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 23.1M  100 23.1M    0     0  3391k      0  0:00:07  0:00:07 --:--:-- 3490k
```

ภาพที่ 3.9 การดาวน์โหลดแพ็กเกจไฟล์ของไฟล์บีท

2) ติดตั้ง filebeat โดยการแตก (extract) ไฟล์ที่ได้ทำการดาวน์โหลดมา โดยใช้คำสั่ง $tar xzvf ตามด้วยชื่อไฟล์ที่ดาวน์โหลดมา เป็นอันติดตั้งสำเร็จ

3.3.3.2 ขั้นตอนการตั้งค่า filebeat

หลังจากทำการติดตั้งเสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปจะเป็นการตั้งค่า ณ ที่นี้จะใช้ filebeat 2 ตัวด้วยกัน เนื่องจากรูปแบบข้อมูลที่ได้จากการ query จากฐานข้อมูล InfluxDB และข้อมูลที่ได้จาก TANDEM Mainframe นั้น มีรูปแบบที่ไม่เหมือนกัน ดังนั้นจึงมีการตั้งค่า filebeat ที่แตกต่างกัน โดยแบ่งออกเป็น 2 รูปแบบซึ่งมีรายละเอียดดังนี้

1) การตั้งค่า filebeat สำหรับข้อมูลที่ได้มาจากฐานข้อมูล InfluxDB โดยการกำหนดประเภทของอินพุท (input) ส่วนของที่อยู่ (path) ของไฟล์ และกำหนดเอาท์พุท (output) ว่าจะให้ filebeat ส่งข้อมูลไปยังที่ใด ดังภาพที่ 3.10

```
#========================== Filebeat inputs =============================
filebeat.inputs:
- type: log
  paths:
    - /app/app_logs/*.json
  tail_files: true
#============================== Outputs ===================================
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

ภาพที่ 3.10 การตั้งค่า filebeat สำหรับข้อมูลจาก influxDB

2) การตั้งค่า filebeat สำหรับข้อมูลที่ได้มาจาก TANDEM Mainframe โดยการกำหนดประเภทของอินพุท (input) ส่วนของที่อยู่ (path) ของไฟล์ เนื่องจากลักษณะของล็อกไฟล์ที่ได้มา จึงต้องตั้งค่าให้อ่านบรรทัดที่ขึ้นต้นด้วยตัวเลข ไปจนถึงก่อนบรรทัดที่เข้าเงื่อนไขนี้ ให้อ่านเป็น 1 event และกำหนดเอาท์พุท (output) ว่าจะให้ filebeat ส่งข้อมูลไปยังที่ใด ดังภาพที่ 3.11

```
#========================== Filebeat inputs =============================
filebeat.inputs:
- type: log
  paths:
    - /app/app_logs/*.txt
  tail_files: true
  multiline.pattern: '^\d'
  multiline.negate: true
  multiline.match: after
#============================== Outputs ===================================
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

ภาพที่ 3.11 การตั้งค่า filebeat สำหรับข้อมูลจาก TANDEM Mainframe

3.3.4 ขั้นตอนการติดตั้งและการตั้งค่าล็อกแสตช

ขั้นตอนต่อไปนี้จะเป็นวิธีการติดตั้ง logstash ด้วย docker ซึ่งจะช่วยเพิ่มประสิทธิภาพและอำนวยความสะดวกให้แก่เครื่องมือ และวิธีการตั้งค่า Logstash เพื่อคัดกรองและตัดแต่งข้อมูลที่รับมาจาก filebeat เพื่อให้ได้ข้อมูลที่สามารถทำไปสร้างแผนภาพแสดงได้ และส่งไปเก็บไว้ที่ Elasticsearch โดยมีขั้นตอนดังนี้

3.3.4.1 ขั้นตอนการติดตั้ง Logstash ด้วย Docker compose

ขั้นตอนการติดตั้ง Logstash ด้วย Docker compose นั้น จะสามารถทำงานได้ต่อเมื่อมีการทำงานของ docker อยู่บนเครื่องของผู้ใช้ หากไม่มีจะต้องทำการติดตั้งและเริ่มต้นการใช้งานก่อน โดยในการติดตั้ง logstash ด้วย docker compose จะเริ่มต้นด้วยการเขียนไฟล์ docker-compose.yml ดังภาพที่ 3.12 และไฟล์ .env เพื่อเก็บค่าตัวแปลที่ใช้ในไฟล์ docker-compose.yml ดังภาพที่ 3.13 และบันทึกไว้ที่ path /app/logstash-docker/ และเนื่องจากปัญหาเรื่องสิ่งแวดล้อมในการทำงานของแอปพลิเคชัน จึงต้องมีการใช้งาน logstash 2 ตัว คือ logstash ที่ใช้กับข้อมูลที่มาจาก influxDB และ logstash ที่ใช้กับข้อมูลที่มาจาก TANDEM Mainframe

```
version: '3'
services:
  logstash:
    image: mdw-docker.artifactory.kasikornbank.com:8443/elk/logstash:${LOGSTASH_VERSION}
    container_name: logstash-${PROJECT_NAME}-${LOGSTASH_NAME}
    network_mode: bridge
    environment:
      - "LS_JAVA_OPTS=-Xms${MIN_HEAP} -Xmx${MAX_HEAP}"
      - "BEATS_PORT=${FILEBEAT_PORT}"
      - "PROJECT_NAME=${PROJECT_NAME}"
      - "LOGSTASH_NAME=${LOGSTASH_NAME}"
    ports:
    - "${FILEBEAT_PORT}:${FILEBEAT_PORT}"
    command: ["logstash", "-f", "/usr/share/logstash/pipeline/test.conf", "-w", "8" ]
    volumes:
    - ${PWD}/pipeline:/usr/share/logstash/pipeline/
```

ภาพที่ 3.12 การเขียนไฟล์ docker-compose เพื่อติดตั้ง Logstash

```
## Logstash Configure ##
LOGSTASH_NAME=tandem_pt
PROJECT_NAME=datasync
LOGSTASH_VERSION=6.8.0
MIN_HEAP=512m
MAX_HEAP=512m

## BEAT Configure ##
FILEBEAT_PORT=5044
```

ภาพที่ 3.13 การเขียนไฟล์ .env

3.3.4.2 ขั้นตอนการเขียนการตั้งค่า Logstash

ขั้นตอนต่อไปจะเป็นการเขียนการตั้งค่าของ logstash เพื่อคัดกรองและตัดแต่งข้อมูลที่ได้รับจาก filebeat เพื่อนำไปสร้างเป็นแผนภาพแสดง โดยเขียนไฟล์ไว้ที่เดียวกับไฟล์ docker compose ซึ่งจะแบ่งส่วนของการเขียนออกเป็น 3 ส่วน ประกอบด้วย input, filter และ output โดยมีรายละเอียดดังนี้

1) การตั้งค่า input ของ Logstash เป็นส่วนที่รับข้อมูลจาก filebeat ซึ่งจะกำหนดให้ filebeat และ logstash ติดต่อสื่อสารกันผ่าน port 5044 ดังภาพที่ 3.14 ซึ่ง Logstash ที่ใช้ทั้ง 2 ตัวนั้นมีการตั้งค่าที่เหมือนกัน

```
input {
beats {
    port => 5044
  }
}
```

ภาพที่ 3.14 การตั้งค่า Logstash ในส่วน input

2) การตั้งค่า filter ของ logstash สำหรับข้อมูลของ TANDEM Mainframe ซึ่งเป็นส่วนที่ทำการคัดกรองและตกแต่งข้อมูลตามที่ผู้ใช้งานต้องการ ข้อมูลที่ต้องการเก็บไปใช้งานนั้นมีข้อมูลเวลา (timestamp) ข้อมูลสถานะของแต่ละเครื่องซึ่งมีประเภทข้อมูล (data type) เป็นข้อความ (string) และข้อมูลเวลาล่าช้า (lag time) ในการซิงค์ข้อมูล โดยใช้ grok ในการคัดกรองข้อมูลดังภาพที่ 3.15

```
filter {

  grok {
      match => { "message" => "(?<timestamp>%{DATE}\s%{TIME}).*?\$GGMGR\s*%{WORD:
status_mng}.*?\$GGR\d+\s*%{WORD:status_2400}\s*\w*\s*(%{TIME:lag_2400}|unknown|ti
me.unavail)\s*(%{TIME:t1}|unknown|time.unavail).*?\$GGR\d+\s*%{WORD:status_2401}\
s*\w*\s*(%{TIME:lag_2401}|unknown|time.unavail)\s*(%{TIME:t2}|unknown|time.unavai
l).*?\$GGR\d+\s*%{WORD:status_2402}\s*\w*\s*(%{TIME:lag_2402}|unknown|time.unavai
l)\s*(%{TIME:t3}|unknown|time.unavail).*?\$GGR\d+\s*%{WORD:status_2403}\s*\w*\s*(
%{TIME:lag_2403}|unknown|time.unavail)\s*(%{TIME:t4}|unknown|time.unavail).*?\$GG
R\d+\s*%{WORD:status_2404}\s*\w*\s*(%{TIME:lag_2404}|unknown|time.unavail)\s*(%{T
IME:t5}|unknown|time.unavail).*?\$GGR\d+\s*%{WORD:status_24Tss}\s*\w*\s*(%{TIME:l
ag_24Tss}|unknown|time.unavail)\s*(%{TIME:t6}|unknown|time.unavail)" }
  }
```

ภาพที่ 3.15 การกรองข้อมูลของ TANDEM โดยใช้ grok

หลังจากได้ข้อมูลที่ต้องการแล้วขั้นตอนต่อไปจะเป็นการนำ lag time ของแต่ละเครื่องมาคำนวณเป็นจำนวนเต็มเพื่อนำไปสร้างกราฟ ด้วยใช้ csv และ ruby ดังภาพที่ 3.16

```
  csv {
    columns => [ "h_2400", "m_2400", "s_2400"]
    source => "lag_2400"
    separator => ":"
  }
  csv { …
  }
  csv { …
  }
  csv { …
  }
  csv { …
  }
  csv { …
  }
  ruby {
    code => "event.set('lag_time_2400', ((event.get('h_2400').to_i*60*60) + (even
t.get('m_2400').to_i*60) + (event.get('s_2400').to_i)))"
  }
  ruby { …
  }
  ruby { …
  }
  ruby { …
  }
  ruby { …
  }
  ruby { …
  }
```

ภาพที่ 3.16 การแปลงค่าเวลาล่าช้าในแต่ละเครื่องให้เป็นจำนวนเต็ม

หลังจากนั้นนำสถานะในแต่ละเครื่องไปเข้าเงื่อนไข if และแทนสถานะที่เป็นข้อความด้วยตัวเลขเพื่อจะนำไปคำนวณสถานะโดยรวมของแอปพลิเคชันนี้ ดังภาพที่ 3.17

```
if "" in [status_mng] {
  if  [status_mng] == "ABENDED" {
    mutate {
      replace => { "status_mng" => 10 }
    }
  }
  else if  [status_mng] == "RUNNING" {
    mutate {
      replace => { "status_mng" => 0 }
    }
  }
  else {
    mutate {
      replace => { "status_mng" => 1 }
    }
  }
}
if "" in [status_2400] {···
}
if "" in [status_2401] {···
}
if "" in [status_2402] {···
}
if "" in [status_2403] {···
}
if "" in [status_2404] {···
}
if "" in [status_24Tss] {···
}
```

ภาพที่ 3.17 เงื่อนไข if เพื่อแทนค่าสถานะด้วยตัวเลขของ TANDEM Mainframe

ภาพที่ 3.18 เป็นขั้นตอนการนำค่าสถานะจากทุกเครื่องมาคำนวณหาค่าเฉลี่ยเพื่อนำไปแสดงเป็นค่าสถานะโดยรวมของแอปพลิเคชัน โดยใช้ ruby code หลังจากนั้นก็ทำการลบข้อมูลอื่นที่ไม่ได้ใช้ทิ้ง ด้วย mutate remove_field และทำการเปลี่ยน data type ของสถานะทุกเครื่องให้เป็นจำนวนเต็มเพื่อนำไปแสดงบนตารางแสดงสถานะ และใช้ค่าเวลาที่ได้จากข้อมูลโดยตรงในการทำ dashboard เพื่อให้ได้ข้อมูลที่เป็นความจริงมากที่สุด

```
  ruby {
    code => "event.set('check_status', ((event.get('status_mng').to_i) + (event.g
et('status_2400').to_i) + (event.get('status_2401').to_i) + (event.get('status_24
02').to_i) + (event.get('status_2403').to_i) + (event.get('status_2404').to_i) +
(event.get('status_24Tss').to_i)))"
  }

  mutate {
    remove_field => ["h_2400","h_2401","h_2402","h_2403","h_2404","h_24Tss","m_24
00","m_2401","m_2402","m_2403","m_2404","m_24Tss","s_2400","s_2401","s_2402","s_2
403","s_2404","s_24Tss"]
    convert => {
        "status_mng" => "integer"
        "status_2400" => "integer"
        "status_2401" => "integer"
        "status_2402" => "integer"
        "status_2403" => "integer"
        "status_2404" => "integer"
        "status_24Tss" => "integer"
    }
  }
  date {
    match => ["timestamp" , "dd/MM/yyyy HH:mm:ss"]
    timezone => "Asia/Bangkok"
    target => "timestamp"
  }
}
```

ภาพที่ 3.18 การหาเฉลี่ยสถานะโดยรวมและการเปลี่ยนประเภทข้อมูล

ส่วนต่อไปเป็นการเขียนการตั้งค่า filter สำหรับข้อมูลที่ทำการ query จากฐานข้อมูล InfluxDB โดยมีด้วยกัน 2 แอปพลิเคชัน นั่นคือ แอปพลิเคชัน ORPG (Online Retail Payment Gateway) และแอปพลิเคชัน KPLUS ซึ่งสามารถใช้ logstash ตัวเดียวกันได้ เนื่องจากข้อมูลที่ต้องการใช้งานถูกเก็บไว้ในสิ่งแวดล้อม (environment) เดียวกัน เนื่องจากข้อมูลที่ได้จากการ query จาก InfluxDB มีรูปแบบเป็น json จึงสามารถใช้ json plungin ใน filter เพื่ออ่านค่าจากข้อมูลในรูปแบบนี้ได้เลย หลังจากนั้นทำการตกแต่งชื่อ field ข้อมูล และเปลี่ยน data type ของข้อมูลให้เป็นจำนวนเต็มเพื่อนำไปสร้างกราฟบนแผนภาพแสดง ดังภาพที่ 3.19

```
filter {
json {
    source => "message"
    target => "data"
  }

  if "maria" in [log][file][path] {
    mutate {
      add_field => {
          "db_type" => "mariaorpg"
          "%{[data][results][0][series][0][columns][0]}" => "%{[data][results][0]
[series][0][values][0][0]}"
          "%{[data][results][0][series][0][columns][1]}" => "%{[data][results][0]
[series][0][values][0][1]}"
          "%{[data][results][0][series][0][columns][2]}" => "%{[data][results][0]
[series][0][values][0][2]}"
          "%{[data][results][0][series][0][columns][3]}" => "%{[data][results][0]
[series][0][values][0][3]}"
          "%{[data][results][0][series][0][columns][4]}" => "%{[data][results][0]
[series][0][values][0][4]}"
          "%{[data][results][0][series][0][columns][5]}" => "%{[data][results][0]
[series][0][values][0][5]}"
          "%{[data][results][0][series][0][columns][6]}" => "%{[data][results][0]
[series][0][values][0][6]}"
          "%{[data][results][0][series][0][columns][7]}" => "%{[data][results][0]
[series][0][values][0][7]}"
          "%{[data][results][0][series][0][columns][8]}" => "%{[data][results][0]
[series][0][values][0][8]}"
      }
    }

    mutate {
      convert => { "slave_delay" => "integer" }
    }
  }
```

ภาพที่ 3.19 การตั้งค่า filter ของ แอปพลิเคชัน ORPG (MariaDB)

ต่อไปเป็นการเขียนการตั้งค่า filter สำหรับข้อมูลจากแอปพลิเคชัน KPLUS ที่ใช้งานรวมกับฐานข้อมูล MSSQL โดยมีรูปแบบข้อมูลเป็น json ซึ่งเป็นแบบเดียวกับแอปพลิเคชัน ORPG จากนั้นก็ทำการตัดแต่งข้อมูล ลบ field ข้อมูลที่ไม่ได้ใช้งานทิ้งเพื่อลดการใช้พื้นที่กับการจัดเก็บ ดังภาพที่ 3.20

```
if "mssql" in [log][file][path] {
   mutate {
      add_field => {
        "db_type" => "mssql"
        "%{[data][results][0][series][0][columns][0]}" => "%{[data][results][0][series][0][values][0][0]}"
        "%{[data][results][0][series][0][columns][2]}" => "%{[data][results][0][series][0][values][0][2]}"
        "%{[data][results][0][series][0][columns][5]}" => "%{[data][results][0][series][0][values][0][5]}"
        "%{[data][results][0][series][0][columns][8]}" => "%{[data][results][0][series][0][values][0][8]}"
        "%{[data][results][0][series][0][columns][11]}" => "%{[data][results][0][series][0][values][0][11]}"
        "%{[data][results][0][series][0][columns][14]}" => "%{[data][results][0][series][0][values][0][14]}"

    }
  }
mutate {
  remove_field => [ "message","data" ]
}
grok {
      match => { "time" => "\wT%{NUMBER:time_check}:" }
  }
```

ภาพที่ 3.20 การตัดแต่งข้อมูลจากฐานข้อมูล MariaDB

ภาพที่ 3.21 เป็นการเขียนเงื่อนไข if เพื่อตรวจสอบข้อมูลเวลาล้าช้าที่แสดงถึงพฤติกรรมการทำงานของการซิงค์ข้อมูลของฐานข้อมูล MSSQL แล้วเปลี่ยน data type ให้เป็นตัวเลขเพื่อนำไปสร้างกราฟ และเพิ่ม field ข้อมูลขึ้นมาใหม่ ให้กับแต่ละสถานะเพื่อนำไปคำนวณหาค่าเฉลี่ยของสถานะโดยรวมของการทำงาน

```
if "" in [PPRDWKMBEDB01_EMOBILE_emobile] {
  if "%" in [PPRDWKMBEDB01_EMOBILE_emobile] {
        mutate {
          replace => { "PPRDWKMBEDB01_EMOBILE_emobile" => 0 }
          convert => { "PPRDWKMBEDB01_EMOBILE_emobile" => "float" }
          add_field => { "status_check01" => 0 }
        }
    }
  else {
    mutate {
      convert => { "PPRDWKMBEDB01_EMOBILE_emobile" => "float" }
    }
    if [PPRDWKMBEDB01_EMOBILE_emobile] >= 60 {
      mutate {
        add_field => { "status_check01" => 1 }
      }
    }
    else {
      mutate {
        add_field => { "status_check01" => 0 }
      }
    }
  }
}
if "" in [PPRDWKMDB07_EMOBILE_emobile] {···
if "" in [PPRDWKMDB08_EMOBILE_emobile] {···
if "" in [PPRDWKMDB09_EMOBILE_emobile] {···
if "" in [SDRWKMDB07_EMOBILE_emobile] {···
```

ภาพที่ 3.21 เงื่อนไข if เพื่อแทนค่าสถานะด้วยตัวเลขของ MSSQL

ภาพที่ 3.22 เป็นการนำค่าสถานะของแต่ละเครื่องเซิร์ฟเวอร์มาคำนวณเป็นค่าเฉลี่ยของค่าสถานะโดยรวมในการซิงค์ข้อมูลของฐานข้อมูล MSSQL จากแอปพลิเคชัน KPLUS และลบข้อมูลที่ไม่ได้ใช้งานทิ้งเพื่อลดการใช้พื้นที่ในการจัดเก็บ

```
  ruby {
        code => "event.set('status_check', ((event.get('status_check01').to_i) + (eve
nt.get('status_check07').to_i) + (event.get('status_check08').to_i) + (event.get('sta
tus_check09').to_i) + (event.get('status_checks7').to_i)))"
        }
  mutate {
      remove_field => [ "status_check01","status_check07","status_check08","status_ch
eck09","status_checks7" ]
  }
}
  mutate {
    remove_field => [ "message","data" ]
  }
}
```

ภาพที่ 3.22 การคำนวณของสถานะโดยรวมและการตัดแต่งข้อมูลจาก MSSQL

3) ส่วนสุดท้ายของการเขียนการตั้งค่าของ logstash คือ ส่วน output สามารถกำหนดได้ว่าจะส่งข้อมูลที่ถูกคัดกรองและตัดแต่งแล้วไปยังที่ใด โดย logstash ที่ใช้ทั้งสองตัวนั้น ส่งข้อมูลไปเก็บไว้ยัง Elasticsearch เหมือนกัน ดังภาพที่ 3.23 แต่ใช้ Elasticserach คนละตัวเนื่องจากปัญหาในเรื่อง environment

```
output {
    elasticsearch {
     hosts => ["http://172.30.150.66:9200"]
     index => "tandem-monitor-datasync-%{+YYYY.MM.dd}"
     user => "user"
     password => "passw0rd"
    }
}
```

ภาพที่ 3.23 การเขียนการตั้งค่า output ของ logstash

3.3.5 ขั้นตอนการติดตั้งและการตั้งค่าอีลาสติคเสิร์ช

เนื่องจากทางองค์กรมีการใช้งานเครื่องมือ Elasticsearch อยู่ก่อนหน้าแล้ว จึงมีการขอเพื่อใช้งานร่วมกันเพื่อลดพื้นที่การใช้งาน แต่ผู้ใช้สามารถติดตั้งใหม่ได้ด้วยการดาวน์โหลดแพ็กเกจไฟล์ตามระบบปฏิบัติใช้งานได้จาก https://www.elastic.co/guide /en/elasticsearch/reference/current/install-elasticsearch.html จากภาพที่ 3.24 เป็นการใช้คำสั่ง $curl ในการดาวน์โหลดแพ็กเกจไฟล์สำหรับระบบปฏิบัติการลินุกซ์ (Linux) หลังจากนั้นทำการแตก (extract) ไฟล์ที่ดาวน์โหลดมา และรัน (run) เซอร์วิสนี้โดยใช้ command $./bin/elasticsearch & โดยมีการกำหนดไว้เป็นค่าเริ่มต้นให้ใช้งานที่ port 9200 และสามารถแก้ไขการตั้งค่าได้ที่ /config/elasticsearch.yml

```
$ curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.5.1-linux-x
86_64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  276M  100  276M    0     0  2463k      0  0:01:55  0:01:55 --:--:-- 3300k
```

ภาพที่ 3.24 การดาวน์โหลดแพ็กเกจไฟล์ Elasticsearch

3.3.6 ขั้นตอนการติดตั้งและการตั้งค่าคิบานา

อย่างที่ทราบกันดีว่า Kibana เป็นเครื่องมือที่ใช้สำหรับแสดงผลข้อมูลจาก Elasticsearch ดังนั้นก่อนที่จะเริ่มใช้งาน Kibana จะต้องมีการใช้งานของ Elasticsearch ก่อน หลังจากที่ run เซอร์วิสทุกอย่าง (filebeat, logstash และ Elasticsearch) ผู้ใช้งานสามารถตรวจสอบข้อมูลและสร้าง dashboard ได้โดยใช้ Kibana โดยการติดตั้งเครื่องมือ Kibana สามารถดาวน์โหลดแพ็กเกจไฟล์ได้จาก https://www.elastic.co/ guide/en/kibana/current/install.html

จากภาพที่ 3.25 เป็นการใช้คำสั่ง $curl ในการดาวน์โหลดแพ็กเกจไฟล์สำหรับระบบปฏิบัติการลินุกซ์ (Linux) หลังจากนั้นทำการแตก (extract) ไฟล์ที่ดาวน์โหลดมา และรัน (run) เซอร์วิสนี้โดยใช้ command $./bin/kibana & โดยมีการกำหนดไว้เป็นค่าเริ่มต้นให้ใช้งานที่ port 5601 และกำหนดให้แสดงผลให้ Elasticsearch เครื่อง http://localhost:9200 ซึ่งสามารถแก้ไขการตั้งค่าได้ที่ /config/kibana.yml

```
$ curl -O https://artifacts.elastic.co/downloads/kibana/kibana-7.5.1-lin
ux-x86_64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0   227M    0  691k    0     0   691k      0  0:05:36  0:00:01  0:05:35
100   227M  100  227M    0     0  3280k      0  0:01:11  0:01:11 --:--:-- 2402k
```

ภาพที่ 3.25 การดาวน์โหลดแพ็กเกจไฟล์ Kibana

หลังจากติดตั้งและรัน (run) Kibana ขึ้นมาใช้งาน ทำการสร้าง index pattern ซึ่งเป็นการสร้างชุด index ข้อมูลจากฐานข้อมูล Elasticsearch และสามารถดูรายละเอียดของข้อมูลใน index pattern นั้น ๆ ได้จากแท็บ (tab) เมนู discover ดังภาพที่ 3.26 และเลือกใช้งานเมนูอื่น ๆ ได้จากแท็บด้านซ้าย

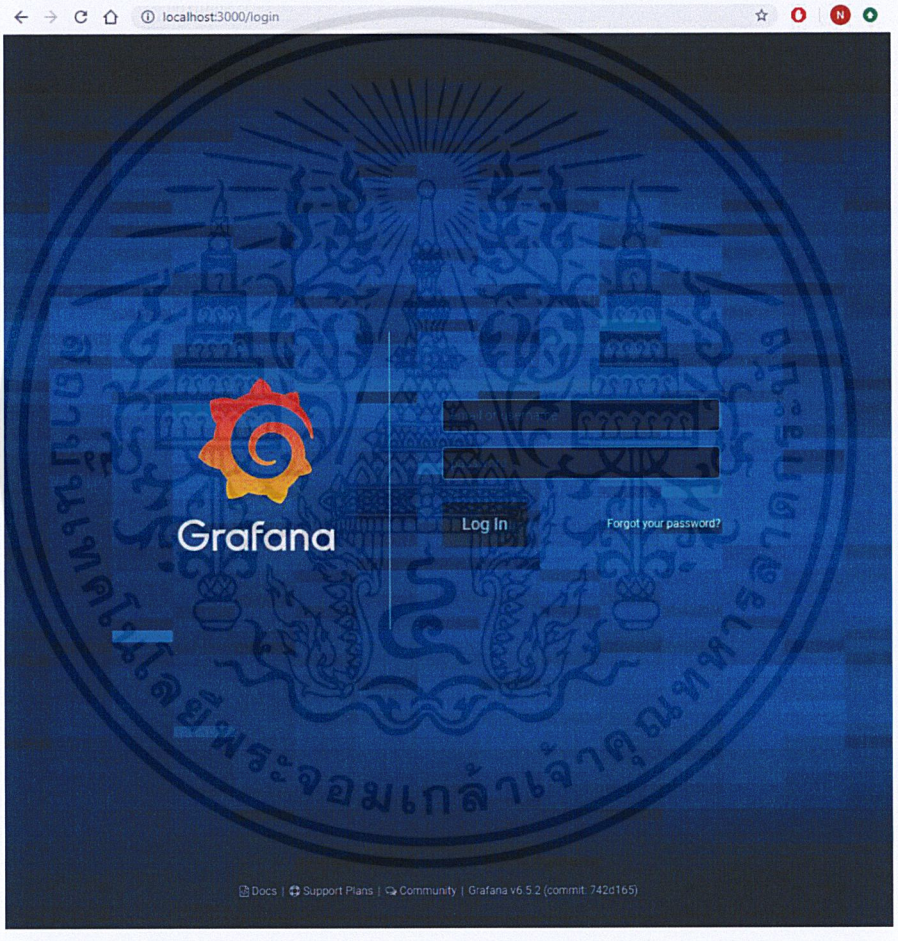
ภาพที่ 3.26 การแสดงผลรายละเอียดข้อมูลจาก Elasticsearch ผ่าน Kibana

3.4 กระบวนการสร้างแผนภาพแสดง การแจ้งเตือน และการทดสอบ

หลังจากได้จัดเตรียมและคัดกรองข้อมูลที่ต้องการเรียบร้อยแล้ว ขั้นตอนต่อไปนี้จะเป็นการสร้างแผนภาพเพื่อแสดงสถานะของแต่ละแอปพลิเคชันและข้อมูลที่นำมาแสดงเพื่อวิเคราะห์พฤติกรรมของแอปพลิเคชันนั้นที่เกี่ยวข้องกับการซิงค์ข้อมูลของฐานข้อมูล โดยมีรายละเอียดดังนี้

3.4.1 ขั้นตอนการติดตั้งและการตั้งค่ากราฟานา

ในขั้นตอนนี้ผู้ใช้งานสามารถดาวน์โหลดเครื่องมือ Grafana ตามระบบปฏิบัติการที่เลือกใช้ได้จากhttps://grafana.com/docs/grafana/latest/installation หลังจากดาวน์โหลดเสร็จเรียบร้อยแล้ว ให้ทำการ แตก (extract) ไฟล์ที่ได้ทำการดาวน์โหลดมาและทำการ start \bin\grafana-server โดยใช้ port 3000 ซึ่งถูกกำหนดไว้เป็นค่าเริ่มต้น (ตั้งค่าหรือแก้ไขค่าเพิ่มเติมได้ที่ \conf\defaults.ini) จะได้หน้าต่าง Login เข้าสู่การใช้งานตามภาพที่ 3.27



ภาพที่ 3.27 หน้าต่างการเข้าสู่ระบบของ grafana

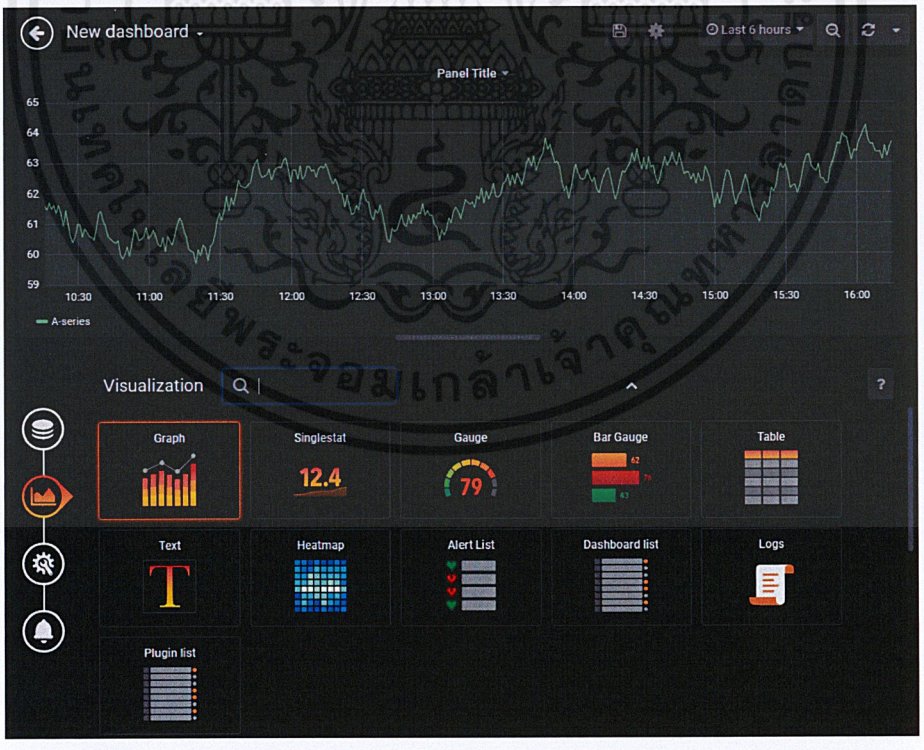
3.4.2 ขั้นตอนการสร้างภาพแสดงจากกราฟานา

เริ่มจากการ add datasources จากฐานข้อมูล Elasticsearch เพื่อนำข้อมูลที่ถูกคัดกรองจาก Logstash มาสร้างเป็นแผนภาพแสดง (dashboard) ซึ่งสามารถเลือกใช้เมนูต่าง ๆ ได้จากแท็บด้านซ้าย เมื่อ add datasources เสร็จจะได้ผลลัพธ์ดังภาพที่ 3.28

ภาพที่ 3.28 การตั้งค่าการเพิ่มแหล่งที่มาของข้อมูล

หลังจากที่ได้เพิ่มแหล่งที่มาของข้อมูลเรียบร้อยแล้วขั้นตอนต่อไปเป็นการนำข้อมูลนั้นมาสร้างแผนภาพแสดง (dashboard) ตามที่ต้องการ สามารถเลือกแหล่งที่มาของข้อมูลและข้อมูลต่าง ๆ ภายในฐานข้อมูลเพื่อนำแสดงได้ดังภาพที่ 3.29 รวมถึงเลือกรูปแบบในการนำเสนอและการตั้งค่าเพิ่มเติมได้ดังภาพที่ 3.30

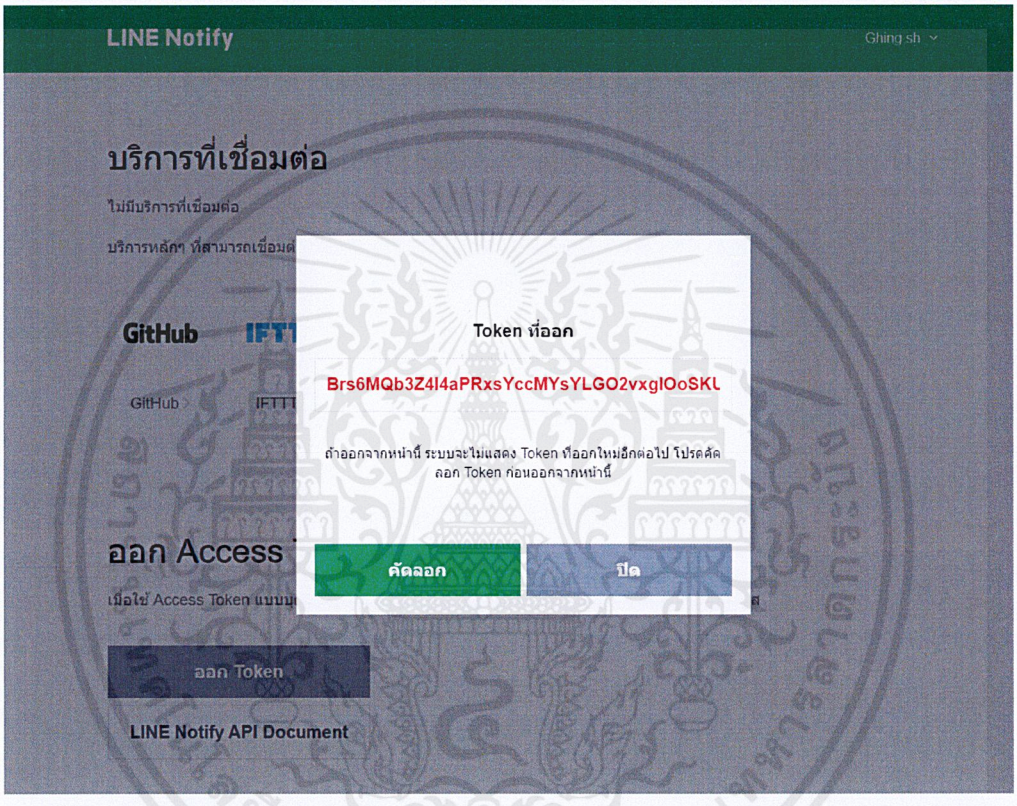
ภาพที่ 3.29 วิธีการนำข้อมูลมาแสดงบน dashboard



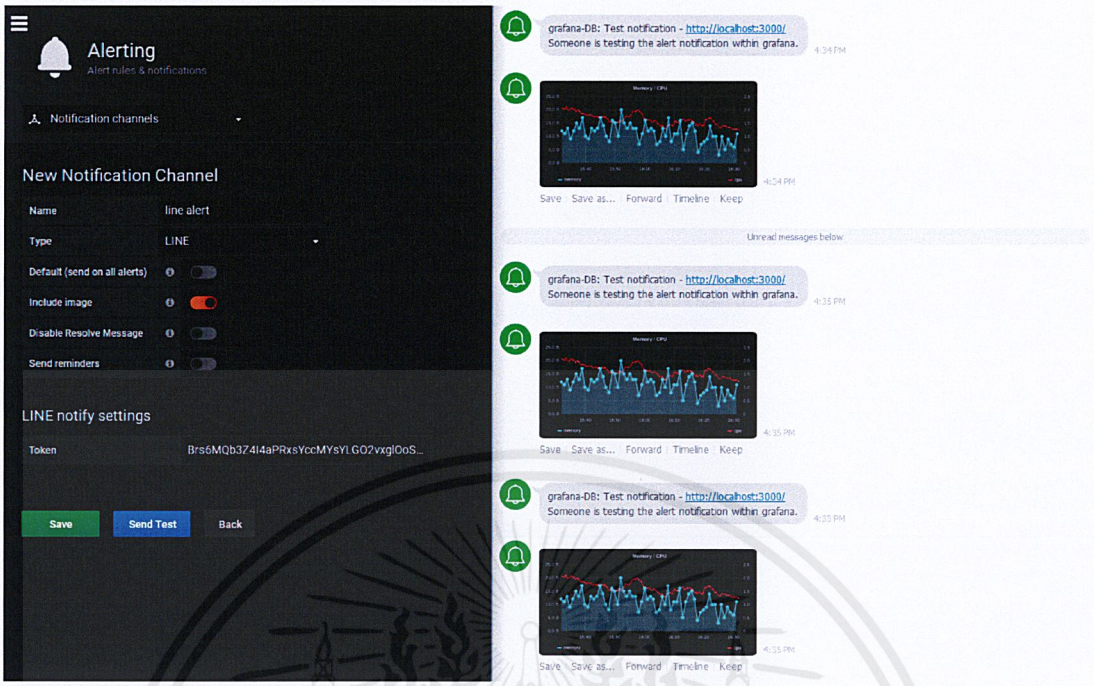
ภาพที่ 3.30 วิธีการเลือกรูปแบบการนำเสนอในการสร้าง dashboard

3.4.3 ขั้นตอนการตั้งค่าการแจ้งเตือนกราฟานาผ่านแอปพลิเคชันไลน์โนติฟาย

การตั้งค่าการแจ้งเตือนด้วย Grafana สามารถขอออก Token ได้จาก https://notify-bot.line.me/th/ ดังภาพที่ 3.31 และนำ Token ที่ได้ไปใส่ในการตั้งค่าบน Grafana ดังภาพที่ 3.32



ภาพที่ 3.31 การออก Token ของ Line Notify



ภาพที่ 3.32 การตั้งค่าและทดสอบการแจ้งเตือนจาก grafana ไปยัง Line

บทที่ 4

ผลการดำเนินงาน

ตลอดระยะเวลาที่ได้ศึกษากระบวนการสร้างแผนภาพแสดงข้อมูลสถานะและข้อมูลต่าง ๆ ของแอปพลิเคชันที่เกี่ยวกับการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลเพื่อเพิ่มประสิทธิภาพในการค้นหาต้นเหตุของข้อผิดพลาดหรือปัญหาที่เกิดขึ้นจนสิ้นสุดโครงการสหกิจศึกษานั้น ทำให้ได้ผลลัพธ์ของแผนภาพแสดงที่สามารถอำนวยความสะดวกให้กับผู้พัฒนาและผู้ดูแลระบบได้ในแบบเรียลไทม์

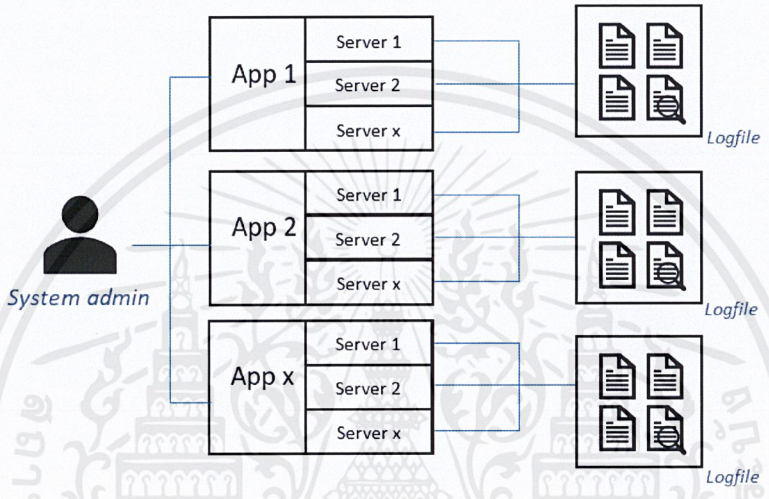
โดยผลการดำเนินงานจะแบ่งเป็นลักษณะของภาพรวมของวิธีการในการเข้าถึงข้อมูลและแผนภาพแสดงสถานะและข้อมูลต่าง ๆ ที่เกี่ยวข้องกับการซิงค์ข้อมูลในแต่ละแอปพลิเคชัน ดังต่อไปนี้

4.1 ระบบการทำงานเพื่อเพิ่มประสิทธิภาพให้กับวิธีในการค้นหาข้อผิดพลาดที่แท้จริง

ผลการดำเนินงานจากการศึกษาปัญหาของวิธีการค้นหาต้นเหตุของข้อผิดพลาดหรือปัญหาของการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูลในแต่ละแอปพลิเคชัน จึงมีการสร้างแผนภาพแสดงนี้เกิดขึ้น เพื่อลดการใช้ทรัพยากรทางเวลาและผลกระทบที่เกิดกับลูกค้า โดยจะแสดงผลลัพธ์ของวิธีการแบบเดิมเปรียบเทียบกับวิธีการแบบใหม่ดังต่อไปนี้

4.1.1 ผลลัพธ์ของวิธีการแบบเดิม

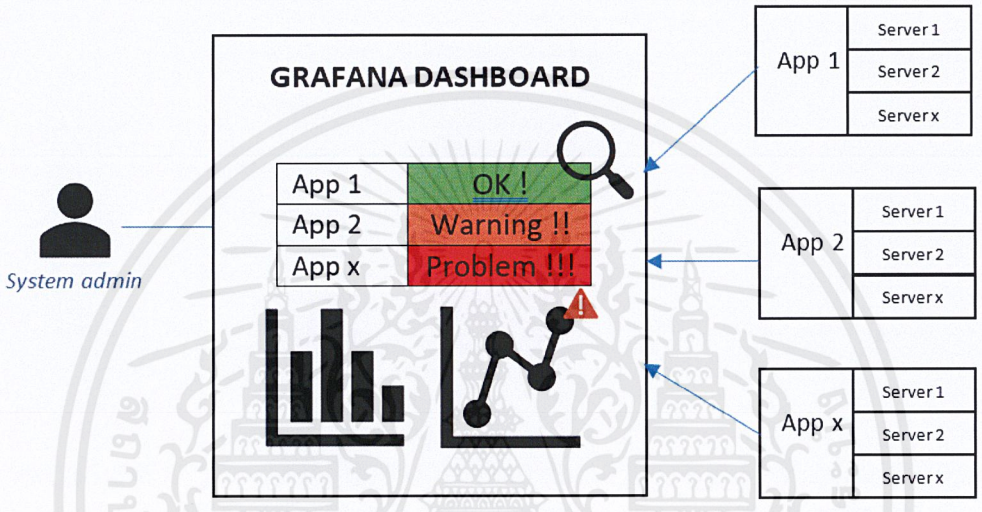
ภาพที่ 4.1 เป็นแผนภาพแสดงวิธีในการเข้าไปอ่านข้อมูลจากล็อกไฟล์ ซึ่งใช้วิธีในการอ่านล็อกไฟล์ที่เป็นรูปแบบตัวอักษรจากข้อมูลของแต่ละเครื่อง server ของแต่ละแอปพลิเคชัน จึงใช้ระยะเวลามากในการค้นหาข้อมูลที่ต้องการและเวลาในการเข้าถึงในแต่ละเครื่อง



ภาพที่ 4.1 ขั้นตอนในการวิเคราะห์ข้อมูลแบบเดิม

4.1.2 ผลลัพธ์ของวิธีการแบบใหม่

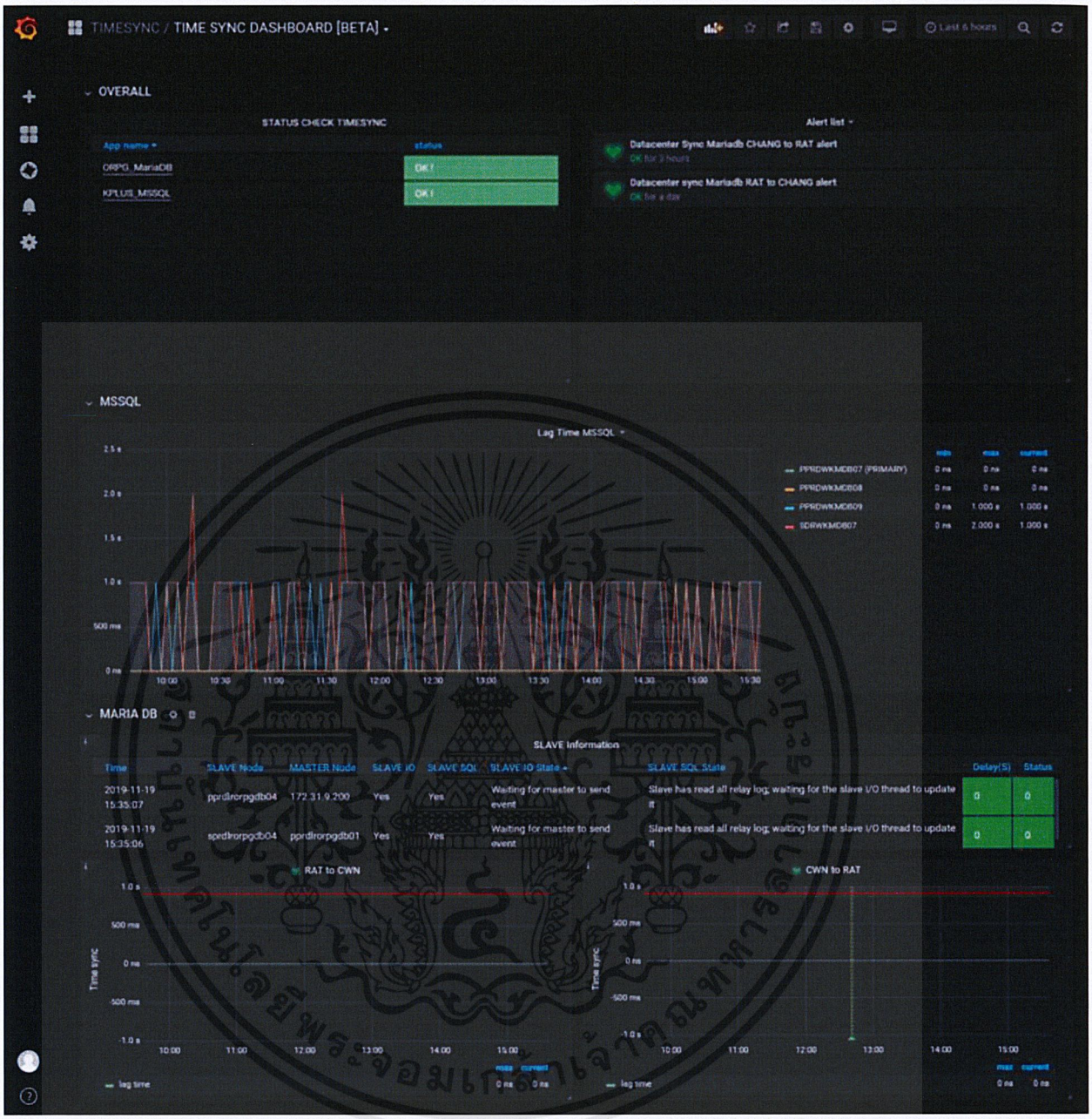
ภาพที่ 4.2 เป็นแผนภาพแสดงวิธีการวิเคราะห์ข้อมูลในรูปแบบใหม่ เพื่อเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูล ลดระยะเวลาในการเข้าถึงเครื่อง server เพื่อวิเคราะห์ข้อมูล โดยการนำข้อมูลจากล็อกไฟล์มาคัดกรองและเปลี่ยนรูปแบบเป็นแผนภาพแสดง (dashboard) ด้วยเครื่องมือทางซอฟต์แวร์



ภาพที่ 4.2 ขั้นตอนในการวิเคราะห์ข้อมูลรูปแบบใหม่

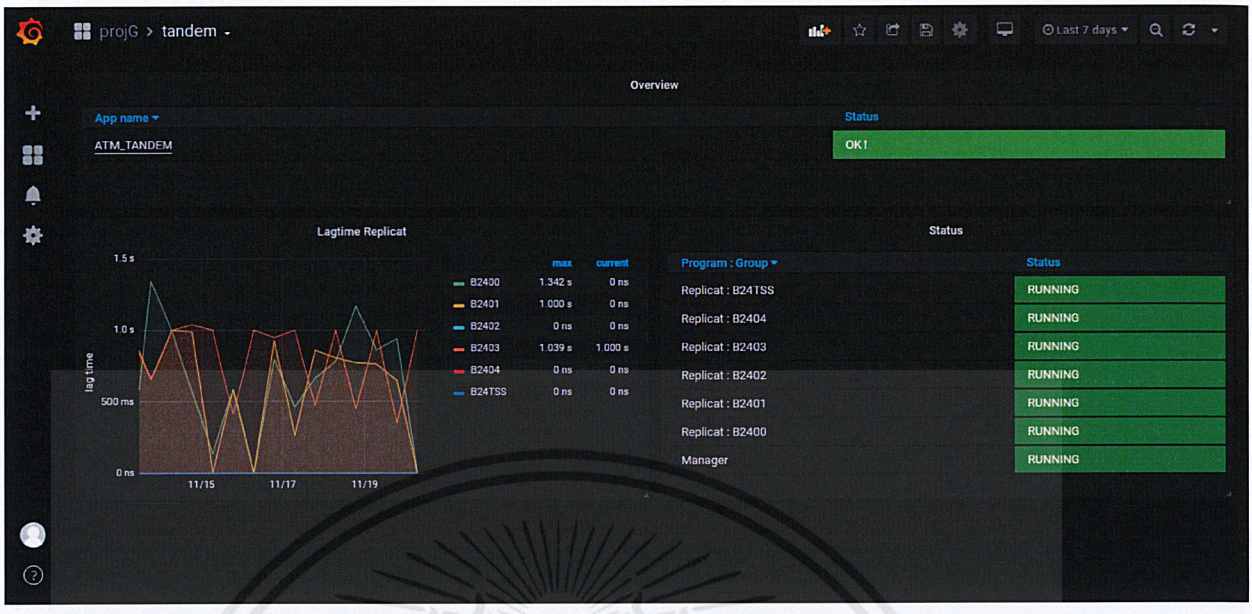
4.2 ข้อมูลแสดงบนแผนภาพจากกราฟานาและการแจ้งเตือน

ภาพที่ 4.3 แสดงข้อมูลที่ได้จากล็อกไฟล์จากแอปพลิเคชัน ORPG (MariaDB) และ KPLUS (MSSQL) แบบเรียลไทม์ โดยส่วนแรกแสดงสถานะโดยรวมของฐานข้อมูลที่แต่ละแอปพลิเคชันใช้งาน และรายการของสถานะการแจ้งเตือน ส่วนต่อไปเป็นกราฟแสดงค่า Lag time ในการซิงค์ข้อมูลแต่ละเครื่อง server ของฐานข้อมูล MSSQL จากแอปพลิเคชัน KPLUS และในส่วนสุดท้ายแสดงตารางข้อมูลรายละเอียดของการซิงค์ข้อมูลและกราฟแสดงค่า Lag time ของการซิงค์ข้อมูลจากฐานข้อมูล MariaDB โดยแอปพลิเคชัน ORPG



ภาพที่ 4.3 ข้อมูลบน dashboard จากแอปพลิเคชัน ORPG และ KPLUS

ภาพที่ 4.4 แสดงสถานะโดยรวมของแอปพลิเคชัน ATM (Tandem) โดยประกอบไปด้วยตารางที่แสดงสถานะการทำงานของการซิงค์ข้อมูลโดยรวมของระบบ กราฟแสดงค่า Lagtime ของแต่ละเครื่อง และ ตารางแสดงสถานะการทำงานของแต่ละเครื่อง server



ภาพที่ 4.4 ข้อมูลบน dashboard จากแอปพลิเคชัน ATM (Tandem)

ภาพที่ 4.5 เป็นภาพที่แสดงการทำงานของการแจ้งเตือนของการตั้งค่าผ่านเครื่องมือ Grafana ไปยังแอปพลิเคชัน Line notify โดยเมื่อเกินค่าที่เราได้ตั้งค่าไว้ก็จะทำการแจ้งเตือนไปยังแอปพลิเคชัน Line ของผู้ใช้งานที่ได้ตั้งค่าไว้ดังภาพ

ภาพที่ 4.5 การทำงานของการแจ้งเตือนจาก Grafana ไปยัง Line notify

บทที่ 5

สรุปผลการดำเนินงานและข้อเสนอแนะ

ตลอดระยะเวลาการดำเนินงานวิเคราะห์รูปแบบการทำงานไปจนถึงการปรับปรุงกระบวนการต่าง ๆ จนกระทั่งสิ้นสุดโครงการสหกิจศึกษา สามารถสรุปผลการดำเนินงาน ปัญหาและอุปสรรค และข้อเสนอแนะแนวทางในอนาคตได้ ดังต่อไปนี้

5.1 สรุปผลการดำเนินงาน

จากการดำเนินงานการสร้างแผนภาพแสดงสถานะและข้อมูลต่าง ๆ ที่เกี่ยวข้องจากล็อกไฟล์ของแอปพลิเคชันที่มีการซิงค์ข้อมูลระหว่างศูนย์กลางข้อมูล ส่งผลให้ผู้ดูแลระบบทำงานได้อย่างรวดเร็วและมีประสิทธิภาพที่ดีขึ้น ไม่ว่าจะเป็นเรื่องของการหาต้นเหตุข้อผิดพลาดหรือปัญหาที่เกิดขึ้น การวิเคราะห์พฤติกรรมของแอปพลิเคชันนั้น ๆ ในระหว่างทำการซิงค์ข้อมูลในแบบเรียลไทม์ หรือ การทราบถึงเหตุการณ์ที่ผิดปกติเกี่ยวกับการทำงานของการซิงค์ข้อมูลในแต่ละแอปพลิเคชันได้อย่างรวดเร็วผ่านการแจ้งเตือน เนื่องจากการเปลี่ยนขั้นตอนจากการอ่านข้อมูลจากล็อกไฟล์ที่เป็นรูปแบบตัวอักษรเป็นการแสดงข้อมูลผ่านแผนภาพ ไม่ว่าจะเป็นในรูปแบบของตาราง หรือ กราฟ อีกทั้งยังส่งผลให้นักพัฒนาทำงานได้สะดวกมากยิ่งขึ้น และลดช่วงเวลาที่ลูกค้าได้รับผลกระทบอีกด้วย

5.2 ปัญหาและอุปสรรค

5.2.1 เนื่องจากเป็นชิ้นงานใหม่จึงต้องใช้ระยะเวลาในการร้องขอข้อมูลและรายละเอียดที่เกี่ยวข้องต่าง ๆ จากผู้ดูแลในแต่ละแผนก จึงทำให้เกิดช่องว่างการทำงานในระหว่างรอข้อมูล

5.2.2 เนื่องด้วยความแตกต่างของสิ่งแวดล้อม (environment) ที่แอปพลิเคชันทำงานนั้น ทำให้ต้องแยกการสร้างแผนภาพแสดงตามสิ่งแวดล้อมของแอปพลิเคชัน

5.3 ข้อเสนอแนะและแนวทางในอนาคต

5.3.1 สามารถนำแนวคิดนี้ไปปรับใช้ได้กับข้อมูลทุกรูปแบบที่ต้องการ ไม่ว่าจะเป็นข้อมูลด้านความปลอดภัย ไปจนถึงข้อมูลที่เกี่ยวกับธุรกิจ เพื่อเพิ่มประสิทธิภาพในการวิเคราะห์ข้อมูล

เอกสารอ้างอิง

[1] บริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป. 2562. ที่มาและความสำคัญของบริษัท กสิกร บิซิเนส-เทคโนโลยี กรุ๊ป. [Online].

แหล่งที่มา http://www.kbtg.tech/th/Pages/default.aspx#contact-us (25 พฤศจิกายน 2562)

[2] Log file คืออะไร สำคัญต่อการส่งข้อมูลอย่างไร เรื่องนี้เราอยากเล่า.[ม.ป.ป]. [Online].

แหล่งที่มา https://www.quickserv.co.th/knowledge-base/solutions/Log (25 พฤศจิกายน 2562)

[3] softnix. 2560. แนวทางการใช้ Centralize Log Management ร่วมกับ SIEM. [Online].

แหล่งที่มา https://medium.com/softnix/แนวทางการใช้-centralize-log-management-ร่วมกับ-siem-b948693bb2b (25 พฤศจิกายน 2562)

[4] สาธิต บุญเทือง. 2559. การจัดเก็บข้อมูลจราจรคอมพิวเตอร์แบบเห็นชื่อผู้ใช้งาน. [Online].

แหล่งที่มา http://www.msit.mut.ac.th/thesis/Thesis_2559/(MISS)%20การจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์แบบเห็นชื่อผู้ใช้งาน.pdf (26 พฤศจิกายน 2562)

[5] Elastic. 2562. HowLogstash Works. [Online].

แหล่งที่มา https://www.elastic.co/guide/en/logstash/current/pipeline.html (26 พฤศจิกายน 2562)

[6] Supawan Ngamlap. 2560. ทำความรู้จักกับ Grafana Dashboard. [Online].

แหล่งที่มา https://developers.ascendcorp.com/ทำความรู้จักกับ-grafana-dashboard-1a5efe6d170a (27 พฤศจิกายน 2562)

เอกสารอ้างอิง (ต่อ)

[7] What is Grafana? Why Use It? Everything You Should Know About It.[ม.ป.ป]. [Online].

แหล่งที่มา https://www.8bitmen.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/ (27 พฤศจิกายน 2562)

[8] Th.wikipedia. 2556. วีไอ. [Online].

แหล่งที่มา https://th.wikipedia.org/wiki/%E0%B8%A7%E0%B8%B5%E0%B9%84%E0%B8%AD (28 พฤศจิกายน 2562)

[9] arnondora. 2559. Vi หรือ Vim คืออะไร มาทำความรู้จักกัน. [Online].

แหล่งที่มา https://arnondora.in.th/basic-vim (28 พฤศจิกายน 2562)

[10] Aoo Studio. 2562. shell script (เชลล์ สคริปต์) คืออะไร. [Online].

แหล่งที่มา https://aoostudio.com/single-blog.php?id=37&shell%20script%20(เชลล์%C2%A0สคริปต์)%20คืออะไร (28 พฤศจิกายน 2562)

[11] G-ABLE. 2562. Data Center Transformation คืออะไร. [Online].

แหล่งที่มา https://www.g-able.com/thinking/data-center-transformation/ (29 พฤศจิกายน 2562)

[12] beartai. 2562. มาตรฐานของ Data Center. [Online].

แหล่งที่มา https://www.beartai.com/article/tech-article/94524 (29 พฤศจิกายน 2562)

เอกสารอ้างอิง (ต่อ)

[14] aquasec. 2561. Docker Architecture. [Online].

แหล่งที่มา https://www.aquasec.com/wiki/display/containers/Docker+Architecture (1 ธันวาคม 2562)

[15] Rachata Tongpagdee. 2561. Docker คืออะไร ใช้งานอย่างไร. [Online].

แหล่งที่มา https://medium.com/@rachatatongpagdee/docker-คืออะไร-ใช้งานอย่างไร-7e77145967b6 (1 ธันวาคม 2562)

[16] Docker Inc. 2562. Docker overview. [Online].

แหล่งที่มา https://docs.docker.com/engine/docker-overview/ (2 ธันวาคม 2562)

[17] Admin hostpacific. 2560. ทำความรู้จัก Docker และการใช้งานบน CentOS 7. [Online].

แหล่งที่มา https://www.hostpacific.com/using-docker-on-centos7/ (2 ธันวาคม 2562)

[18] AVI. 2562. Docker Architecture and its Components for Beginner. [Online].

แหล่งที่มา https://geekflare.com/docker-architecture/ (3 ธันวาคม 2562)

[19] Somprasong Damyos. 2560. Docker: Docker Networking. [Online].

แหล่งที่มา https://medium.com/@somprasongd/docker-networking-59b6637de3df (3 ธันวาคม 2562)

[20] Linux คืออะไร ?.[ม.ป.ป]. [Online].

แหล่งที่มา https://web.ku.ac.th/schoolnet/snet1/software/linux/ (4 ธันวาคม 2562)

ประวัติผู้เขียน

หัวข้อโครงงาน	แดชบอร์ดการซิงค์ของศูนย์ข้อมูล
ชื่อ – สกุล	นางสาวนัทธ์หทัย นวนหงษ์
รหัสนักศึกษา	59010718
คณะ	วิศวกรรมศาสตร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
สาขาวิชา	วิศวกรรมสารสนเทศ
วัน เดือน ปีเกิด	27 ตุลาคม 2540
สถานที่อยู่ปัจจุบัน	36/3 ถนนควนขนุน ตำบลทับเที่ยง อำเภอเมือง จังหวัดตรัง รหัสไปรษณีย์ 92000
ประวัติการศึกษา	พ.ศ. 2559 – ปัจจุบัน สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (ระดับปริญญาตรี) พ.ศ. 2553 - พ.ศ. 2558 โรงเรียนสภาราชินี จังหวัดตรัง (ระดับมัธยมศึกษา) พ.ศ. 2546 - พ.ศ. 2552 โรงเรียนพรศิริกุล (ระดับประถมศึกษา)