รายงานสหกิจศึกษาฉบับสมบูรณ์

การใช้ AI/ML เพื่อจัดการทรัพยากรของ Kubernetes Cluster บน AWS

Using AI/ML to automate Kubernetes Cluster on AWS

ศศินา หาบุศย์

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อโครงงานสหกิจศึกษา การใช้ AI/ML เพื่อจัดการทรัพยากรของ Kubernetes Cluster บน AWS

ชื่อ-สกุล นักศึกษา นางสาวศศินา หาบุศย์

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์

ชื่อ-สกุล อาจารย์นิเทศ ผศ.บัณฑิต พัสยา

ชื่อ-สกุล ผู้นิเทศงาน นายปรัชญ์เกษม เวสารัชช์

ชื่อสถานประกอบการ บริษัท กสิกร บิซิเนส - เทคโนโลยี กรุ๊ป

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อสร้างเครื่องมือสำหรับการทำการทำนายโหลดที่จะเข้ามา และทำการ Autoscaling สำหรับแอปพลิเคชันที่รันอยู่บน Kubernetes เพื่อไม่ให้เกิด Traffic ที่เข้ามามากจนเกินไป จนทำให้แอปพลิเคชันไม่สามารถทำงานต่อได้ และเป็นให้มีค่าใช้จ่ายที่เหมาะสมสำหรับจำนวนโหลดที่เข้ามา เนื่องจากไม่ต้องทำการสำรองเครื่องโหนดเผื่อไว้โดยไม่จำเป็น

โดยเครื่องมือที่เลือกใช้คือการใช้โมเดล ARIMA ซึ่งเป็นโมเดลสำหรับทำนายข้อมูลที่เป็น Time series มาใช้ในการทำนาย และใช้เครื่องมือของ AWS ที่ชื่อว่า AWS lambda ในการสั่งการให้เพิ่มเครื่องโหนด เมื่อมีการทำนายว่าจะมีการใช้ CPU Utilization ของเครื่องใน Kubernetes เกินกำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Co-operative Title: Using AI/ML to automate Kubernetes Cluster on AWS

Student Intern Name: Miss Sasina Habut

Faculty: Engineering **Department:** Computer Engineering

Advisor Name: Asst. Prof. Bundit Passaya

Mentor Name: Mr. Pratkasem Vesarach

Company: KASIKORN Business - Technology Group

ABSTRACT

The purposes of this study were to create a tool for forecasting and autoscaling Kubernetes cluster on AWS to make sure that the application would run perfectly when there are more loads coming, and to optimize the cost of number of running worker nodes.

The tool that was used to forecast the incoming loads is ARIMA models, a model that widely used for forecasting time series data. The tool for scaling number of running worker nodes is AWS Lambda, a service of AWS that allow the user to write function to manage resources on AWS.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ ด้วยความช่วยเหลือของบุคคลดังต่อไปนี้

ขอขอบคุณ อาจารย์บัณฑิต พัสยา อาจารย์นิเทศงานสหกิจศึกษา ที่ช่วยให้คำชี้แนะในการทำโครงงาน ให้คำแนะนำและช่วยแก้จุดบกพร่องของโครงงาน และให้คำปรึกษาด้านงานสหกิจ

ขอขอบคุณ อาจารย์จิระศักดิ์ สิทธิกร ที่ช่วยให้คำแนะนำในการเตรียมตัวสำหรับการทำงานสหกิจศึกษา และการทำเอกสารโครงงาน

ขอขอบคุณ นายปรัชญ์เกษม เวสารัชช์ ผู้นิเทศงาน และสมาชิกในทีม Technology Research ที่ช่วยให้ความรู้ทั้งด้านการทำ Cloud computing การใช้งาน Kubernetes การทำงานในธุรกิจหรือการนำเสนองาน รวมถึงความรู้ในด้านอื่นๆ

สุดท้ายนี้ ขอขอบคุณ บิดา มารดา เพื่อนๆ และบุคคลท่านอื่นที่อาจไม่ได้เอ่ยนามในที่นี้ ที่ให้การสนับสนุนและเป็นกำลังใจให้ จนทำให้ปริญญานิพนธ์นี้สำเร็จได้ด้วยดี

ศศินา หาบุศย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อ I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VII

สารบัญภาพ VIII

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญ 1

1.2 วัตถุประสงค์ของการวิจัย 3

1.3 ขอบเขตของการวิจัย 3

1.4 วิธีดำเนินการวิจัย 3

1.5 ประโยชน์ที่คาดว่าจะได้รับ 4

บทที่ 2 แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 5

2.1 ทฤษฎีที่เกี่ยวข้อง 5

2.1.1 Cloud computing 5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

2.1.2 การประมวลผลบนระบบคลาวด์ด้วย AWS 8

2.1.4 Kebernetes 14

2.1.5 Infrastructure as Code 22

2.1.6 Pandas 23

2.1.7 Crontab 25

2.1.8 ARIMA (AutoRegressive Integrated Moving Average) 27

บทที่ 3 วิธีดำเนินการวิจัย 33

3.1 สถาปัตยกรรมของระบบ 33

3.2 การทำงานของระบบ 34

3.2.1 การจัดเตรียมระบบผ่าน AWS CloudFormation 35

3.2.2 Kubernetes Configuration 38

3.2.3 การเก็บเมตริกของคลัสเตอร์จาก Amazon CloudWatch 40

3.2.4 การสร้าง Load Generator ด้วย Siege ผ่าน Spot Instance 43

3.2.4 สร้าง Scaling policy บน Autoscaling group ของ Node 43

3.2.5 สร้างฟังก์ชั่นสั่งให้ไป Execute policy 46

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

3.2.6 การทำนาย Traffic 46

3.2.7 สร้าง CloudWatch Event Rule 50

3.2.8 Deploy Horizontal Pod Autoscaler 52

3.2.9 Deploy Cluster Autoscaling 52

บทที่ 4 ผลการวิจัย 54

4.1 ผลการทำนาย Traffic 54

4.2 ผลการเพิ่มเครื่อง worker node 55

บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ 57

5.1 สรุปผลการวิจัย 57

5.2 ปัญหาอุปสรรคและข้อเสนอแนะ 57

เอกสารอ้างอิง 59

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่ **หน้า**

1.1 เปรียบเทียบการใช้งาน cloud และ on-premise 1

3.1 ค่าพารามิเตอร์ในการสร้างระบบ 37

3.2 ค่าพารามิเตอร์ที่ใส่เมื่อต้องการเก็บเมตริก 40

3.3 ค่า Properties ของ Scaling policy (scale out) 44

3.4 ค่า Properties ของ Alarm 44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

ภาพที่ **หน้า**

2.1 โลโก้ของ Amazon Web Services 8

2.2 วิธีการทำงานของ Amazon EKS 9

2.3 วิธีการทำงานของ AWS CloudFormation 10

2.4 วิธีการทำงานของ CloudWatch 11

2.5 วิธีการทำงานของ AWS Lambda 12

2.6 Master node และ Worker node ของ Kubernetes 16

2.7 Pods ใน Kubernetes cluster 18

2.8 Service ใน Kubernetes Cluster 19

2.9 Kubernetes Deployment 21

2.10 ตัวอย่าง DataFrame ของ Pandas 24

2.11 รูปแบบคำสั่ง cron 27

2.12 ตรวจสอบความเป็น Stationary ของข้อมูล 28

2.13 ข้อมูลการใช้พลังงานของ US ที่ผ่านการ Transforming และ Differencing 30

3.1 สถาปัตยกรรมของระบบ 33

3.2 ขั้นตอนการทำงานของระบบโดยรวม 35

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

3.3 การทำงานของ AWS CloudFormation 35

3.4 การสร้าง Stack ใน AWS CloudFormation 36

3.5 ลำดับการสร้างเทมเพลตใน CloudFormation 36

3.6 ตรวจสอบเซอร์วิส Kubernetes 39

3.7 การตรวจสอบ Node 39

3.8 การเก็บค่า Metric ผ่าน CloudWatch Log Insight 40

3.9 วิธีการเก็บค่า Metric ลงไฟล์ csv 42

3.10 การสร้าง Load Generator 43

3.11 Datetime index 47

3.12 ตัวอย่างข้อมูลหลังจาก Resample 47

3.13 กราฟการใช้งาน CPU เฉลี่ยทุกๆ 20 นาที 48

3.14 ผลการรัน Model diagnostic 49

3.16 ClusterAutoScaler policy 53

4.1 ตรวจสอบการทำนาย Traffic 54

4.2 ผลการเพิ่มเครื่องโหนดจากการกำหนดเวลาเอง 55

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

4.3 ผลการเพิ่มเครื่องเมื่อ CPU Utilization ถึงค่าที่กำหนด 55

4.4 ขั้นตอนการลดเครื่องโหนดเมื่อมีการใช้งาน CPU ต่ำกว่าที่กำหนดไว 56

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

การติดตั้งเซิร์ฟเวอร์ของแอปพลิเคชั่นต่างๆ จะต้องมีการเลือกว่าจะติดตั้ง on-premise (การติดตั้งลงบนเครื่องเซิร์ฟเวอร์และคอมพิวเตอร์จริงๆ ของเจ้าของผลิตภัณฑ์) หรือ on-cloud (การติดตั้งบนเครื่องคอมพิวเตอร์ของผู้ให้บริการคลาวด์ และเข้าถึงได้ผ่านอินเทอร์เน็ต) โดยข้อแตกต่างหลักๆ ที่ทำให้ตัดสินใจได้ว่าจะใช้ on-premise หรือ on-cloud ก็คือความยากง่ายในการใช้งาน ความปลอดภัยของข้อมูล และราคา

เราเปรียบเทียบข้อดีและข้อเสียของการใช้ cloud และ on-premise ได้คร่าวๆ ตามตารางที่ 1.1

ตารางที่ 1.1 เปรียบเทียบการใช้งาน cloud และ on-premise

	Cloud	On-premise
การใช้งาน	สร้างทรัพยากรบนคอมพิวเตอร์ของผู้ให้บริการ โดยสามารถเข้าใช้งานได้ตราบเท่าที่ยังจ่ายค่าบริการ	ทรัพยากรทุกอย่างสร้างใน Infrastructure ในเครื่องคอมพิวเตอร์ของเจ้าของผลิตภัณฑ์
ค่าใช้จ่าย	จ่ายตามจำนวนทรัพยากรที่สร้างเป็นรายเดือนหรือตามที่ผู้ให้บริการกำหนด	มีค่าใช้จ่ายตั้งแต่ค่าเครื่องเซิร์ฟเวอร์ ค่าไฟที่ใช้ ค่าดูแลรักษาต่างๆ
สิทธิ์การเข้าถึง	การเข้าถึงข้อมูลด้วยการใช้ encryption key หากเกิด downtime ที่ผู้ให้บริการอาจจะทำให้ไม่สามารถเข้าถึงข้อมูลได้	เจ้าของผลิตภัณฑ์เป็นผู้ควบคุมสิทธิ์การเข้าถึงทรัพยากรทั้งหมด
ความปลอดภัย	ความปลอดภัยของข้อมูลในคลาวด์นั้นขึ้นอยู่กับผู้ให้บริการคลาวด์ ถึงแม้ว่าจะมีการดูแลความปลอดภัยของข้อมูลได้ดี แต่	บริษัทที่มีข้อมูลที่เป็นข้อมูลสำคัญเช่น รัฐบาลหรือธนาคารต่างๆ อาจต้องการรักษาความปลอดภัยและเก็บความเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	บางบริษัทอาจจะยังมีความกังวลที่จะเลือกใช้งาน	ส่วนตัวของข้อมูลสูง จึงเลือกใช้ on-premise
การปรับเปลี่ยนขนาด	สามารถทำการเพิ่ม ลดขนาดโครงสร้างพื้นฐานได้ง่ายและค่าใช้จ่ายไม่สูง เมื่อธุรกิจมีการเติบโตขึ้น หรือซบเซาลง	การเพิ่มหรือลดขนาดโครงสร้างพื้นฐานทำได้ยาก และมีค่าใช้จ่ายสูง

จากตาราง ทำให้เห็นว่า on-premise นั้นเหมาะกับโครงสร้างธุรกิจหรือแอปพลิเคชั่นที่ต้องการความปลอดภัยสูงและมีเงินทุนในการที่จะปรับเปลี่ยบโครงสร้างและดูแลรักษาอุปกรณ์ สำหรับธุรกิจใหม่ๆ ที่เพิ่งเริ่มต้นนั้น การเริ่มใช้คลาวด์ ก็เป็นสิ่งที่ดี เพราะตัวคลาวด์ เองก็มีความปลอดภัย สามารถสร้างทรัพยากรต่างๆ ได้อย่างรวดเร็ว และไม่ต้องเสี่ยงแบกรับภาระค่าใช้จ่ายที่สูงสำหรับ Infrastructure ทำให้สามารถนำเงินไปลงทุนด้านอื่นได้เพิ่มขึ้น

ในบริษัท KBTG เองก็มีแอปพลิเคชั่นที่มีการวางโครงสร้างบนคลาวด์เช่นกัน โดยในการสร้างแต่ละครั้งนั้น การที่จะสร้างแบบ manual นั้นก็มีความเสี่ยงที่จะทำให้เกิดความผิดพลาดได้สูง จึงใช้การเขียน template file ให้เป็นการ automate สร้างทรัพยากรที่ต้องการใช้ขึ้นมา และหากมีการแก้ไขบางส่วนหรือลบก็สามารถทำผ่านเทมเพลตได้เลย

การจัดการเซิร์ฟเวอร์ให้สามารถทำงานได้อย่างราบรื่นนั้น ต้องคอยดูไม่ให้มีการรับ load ที่หนักเกินไปจนเซิร์ฟเวอร์รองรับไม่ไหว เพราะอาจทำให้ระบบล่มได้ ถ้าเป็นแบบ on-premise การที่จะเพิ่มเครื่องเพื่อให้รองรับการใช้งานนั้นอาจจะทำได้ลำบากและเสียค่าใช้จ่ายมาก แต่ถ้าบนคลาวด์นั้นสามารถทำได้เลยโดยไม่เสียค่าใช้จ่ายมากและหากการใช้งานกลับมาเป็นปกติก็สามารถลดเครื่องได้เลยทำให้ไม่ต้องเสียเงินเพิ่มในส่วนที่ไม่จำเป็น

การที่จะดูว่าเวลาใดควรเพิ่มหรือลดเครื่องนั้น จำเป็นต้องมีการมอนิเตอร์การใช้งานของแอปพลิเคชั่น บนเครื่องเซิร์ฟเวอร์นั้น และเนื่องจากเราไม่สามารถมานั่งมอนิเตอร์ได้ตลอดเวลา จึงต้องให้มีการแจ้งเตือนเมื่อมีการใช้งานเกินกว่าที่กำหนด และสามารถเพิ่มเครื่องได้โดยอัตโนมัติ เมื่อมีการใช้งานในระยะหนึ่งก็สามารถทำนายได้ว่าเวลาไดที่จะมีการใช้งานเยอะและสามารถเพิ่มเครื่องเพื่อเตรียมรอรับการใช้งานได้ล่วงหน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 สามารถสร้างทรัพยากรได้โดยอัตโนมัติผ่านคลาวด์

1.2.2 สามารถทำการเพิ่มหรือลดขนาดโครงสร้างพื้นฐานได้โดยอัตโนมัติเมื่อมีการใช้งานถึงระดับที่กำหนด

1.2.3 สามารถทำนายได้ว่าในเวลาใดจะมีการใช้งานถึงระดับที่กำหนดและทำการเพิ่มหรือลดขนาดโครงสร้างพื้นฐานล่วงหน้า

1.3 ขอบเขตของการวิจัย

โปรแกรมแสดงผลการทำนายการใช้งานระบบ และทำการเพิ่มลดเครื่องโหนดของ kubernetes ได้ล่วงหน้า

1.4 วิธีดำเนินการวิจัย

1.4.1 ศึกษาการใช้งานคลาวด์ของ AWS

1.4.2 ศึกษาการทำงานของ kubernetes

1.4.3 ออกแบบโครงสร้างพื้นฐานของระบบ

1.4.4 เขียนเทมเพลตเพื่อสร้างทรัพยากรของระบบ

1.4.5 ติดตั้งตัวอย่างแอปพลิเคชั่นลงบน kubernetes และเขียนโปรแกรมให้มีการมอนิเตอร์และทำนายระดับการใช้งาน

1.4.6 ทดสอบการใช้งานของโปรแกรมจนถึงระดับที่น่าพอใจ

1.4.7 จัดทำรูปเล่มรายงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถนำคลาวด์เซอร์วิสของ AWS มาใช้งานได้ รู้ว่าเซอร์วิสไหนใช้ทำอะไรและสามารถสร้างโครงสร้างพื้นฐานของระบบบนคลาวด์ได้

1.5.2 สามารถใช้งาน kubernetes กับ AWS ในการติดตั้งแอปพลิเคชั่น การเพิ่มลดจำนวนเครื่องโหนดและพอดในคลัสเตอร์

1.5.3 สามารถทำนายได้ว่าการใช้งานแอปพลิเคชั่นจะเป็นอย่างไรและบอกได้ว่าจะเพิ่มเครื่องโหนดตอนไหน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 Cloud computing

2.1.1.1 นิยามของ Cloud Computing

เนื่องจากข้อมูลข่าวสารที่เพิ่มมากขึ้นเป็นทวีคูณในปัจจุบัน ความซับซ้อนที่เกิดขึ้นจากความไม่เป็นระเบียบของข้อมูล ค่าใช้จ่ายที่เพิ่มมากขึ้นจากการลงทุน ความหลากหลายของฮาร์ดแวร์และซอฟต์แวร์ การดูแลรักษา ค่าไฟฟ้า ค่า Bandwidth บุคลากรและอื่นๆ จึงทำให้เกิด Cloud Computing ขึ้นมาเพื่อทดแทนและแก้ไขความซับซ้อนของข้อมูลเหล่านี้

Cloud Computing ถูกนิยามและให้คำจัดกัดความว่า การประมวลผลใดๆ ผ่านเครือข่าย Internet ที่พร้อมให้บริการกับผู้ใช้งาน เมื่อไรก็ตามที่มีความต้องการใช้ โดยไม่ยึดติดกับสถานที่ อุปกรณ์และไม่มีข้อจำกัดเรื่องปริมาณและเวลา

2.1.1.2 คุณลักษณะของ Cloud Computing

On-demand self-serviceCloud Computingต้องสามารถจัดการได้ด้วยตัวเอง ตามความต้องการของผู้ใช้บริการ

Broad Network AccessCloud Computingต้องสามารถเข้าถึงได้จากที่ใดก็ได้ โดยไม่มีข้อจำกัดของสถานที่และอุปกรณ์ที่ต้องการเชื่อมต่อ

Resource PoolingCloud Computingต้องสามารถนำทรัพยากรมารวมกันและใช้งานร่วมกันได้

Rapid ElasticityCloud Computingต้องมีความยืดหยุ่นสูง สามารถเพิ่มลดทรัพยากรได้ทันที โดยไม่มีข้อจำกัดเรื่องปริมาณและระยะเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Measured ServiceCloud Computingต้องสามารถวัดปริมาณการใช้งานได้ และสามารถคิดค่าใช้งานได้ตามการใช้งานจริง

2.1.1.3 ประเภทของ Cloud Technology แบ่งตามรูปแบบการใช้งาน

Private Cloud เป็นการใช้งานภายในองค์กร สำหรับองค์กรใดองค์กรหนึ่ง ผู้ใช้บริการเป็นผู้บริหารจัดการระบบ สามารถปรับเปลี่ยนระบบต่างๆได้ด้วยตนเอง ผู้ให้บริการจะมีหน้าที่ติดตั้งและดูแลรักษาให้เท่านั้น ข้อดีของบริการนี้คือมีความปลอดภัยสูงสุด

Community Cloud บริการ Cloud แบบกลุ่ม ประกอบไปด้วย Private Cloud ตั้งแต่ 2 ขึ้นไป ซึ่งมีการกำหนดข้อตกลงและแชร์ข้อมูลร่วมกัน

Public Cloud เป็นบริการเข้าถึงข้อมูลผ่านเครือข่ายสาธารณะ โดยปกติจะใช้งานผ่านผู้ให้บริการซึ่งให้บริการผ่านเครือข่ายสาธารณะ จุดเด่นของบริการนี้คือ ประสิทธิภาพสูงและต้นทุนต่ำ

Hybrid Cloud เป็นการผสมผสานกันระหว่าง Private Cloud และ Public Cloud โดยการนำข้อดีของแต่ละบริการมาใช้งานร่วมกัน

2.1.1.4 แบ่งตามรูปแบบการให้บริการ

SaaS (Software as a service) เป็นรูปแบบการให้บริการใช้ซอฟต์แวร์หรือแอพพลิเคชันผู้ใช้สามารถใช้บริการได้โดยไม่จำเป็นต้องติดตั้งซอฟต์แวร์ ผ่านผู้ให้บริการ เช่น Google Docutment, Office365, Gmail เป็นต้น

PaaS (Platform as a service) เป็นรูปแบบการให้บริการแพลตฟอร์ม สำหรับนักพัฒนาในการพัฒนาโปรแกรม เช่น บริการ Google App Engine, Azure DB และ Amazon RDS เป็นต้น

IaaS (Infrastructure as a Service) เป็นรูปแบบการให้บริการโครงสร้างพื้นฐาน การประมวณผล Storage และ Network เป็นต้น ตัวอย่างผู้ให้บริการเช่น Azure, AWS และ Netway Cloud เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.1.5 ประโยชน์ของ Cloud Computing

ประหยัดการลงทุนทรัพยากรคอมพิวเตอร์ เพราะใช้รูปแบบบริการแบบเช่าผ่านผู้ให้บริการ สามารถใช้บริการได้เท่าที่ต้องการ โดยไม่จำเป็นต้องติดตั้ง พัฒนาและขึ้นระบบเองทั้งหมด

สามารถสร้างระบบใหม่ขึ้นมาทันที ผู้ให้บริการจะจัดเตรียมทรัพยากรเพื่อรองรับผู้ใช้บริการ การสร้างและลบจึงสามารถทำได้ทันที โดยไม่มีข้อจำกัดเรื่องเวลา

เพิ่มขนาดทรัพยากรได้ง่ายดายและรวดเร็ว ตามปริมาณการใช้งานหรือลดการใช้งานของผู้ที่ต้องการใช้

เข้าถึงบริการได้จากทุกที่ ด้วยการเชื่อมต่อข้อมูลผ่านเครือข่าย Internet ผ่านผู้ให้บริการจึงไม่มีข้อจำกัดเรื่องสถานที่และอุปกรณ์

ลดปัญหาการดูแลระบบ เนื่องจากบริการเป็นรูปแบบของการเช่าใช้ จึงเป็นหน้าที่ของผู้ให้บริการที่จะต้องดูแลระบบแทน ซึ่งจะช่วยลดทั้งความยุ่งยากของการดูแลและลดจำนวนบุคลากรที่ต้องจ้างมาเพื่อมาดูแลระบบ

2.1.1.6 ตัวอย่างการประยุกต์ใช้ Cloud ในองค์กร

Compute Engine – ตัวอย่างที่เห็นได้ชัดเจนคือการสร้าง Cloud Server สำหรับทำ Web Server หรือ Mail Server เป็นต้น

Test and Developmnet – องค์กรที่มีการพัฒนาแอปพลิเคชันขึ้นมาใช้งานด้วยตนเองอาจพบปัญหาระบบที่มีอยู่ที่ไม่เพียงพอ หรือต้องเสียเวลาในการขึ้นระบบซึ่งค่อนข้างยาก ซับซ้อนและใช้เวลานาน เมื่อใช้ Cloud นักพัฒนาสามารถทดสอบระบบบน Cloud ได้อย่างรวดเร็ว ทั้งยังสามารถย้าย Application จากระบบหนึ่งไปอีกระบบหนึ่งได้อย่างง่ายดาย โดยไม่ต้องจำเป็นต้องขึ้นระบบขึ้นใหม่

Big Data Analytics - เนื่องจากข้อมูลข่าวสารเพิ่มขึ้นอย่างรวดเร็วในทุกๆ ปี องค์กรขนาดใหญ่จึงเลือกที่จะลงทุนหรือพัฒนา Big Data เพื่อรองรับการเติบโตและการประมวลผลในอนาคต ซึ่งในปัจจุบันก็มีผู้ให้บริการ Big Data เป็นจำนวนมาก ตัวอย่างที่เรารู้จักเช่น Google เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Disaster Recovery – ในสถานการณ์ที่คาดไม่ถึง อาจเกิดจากความไม่สงบ สงคราม หรือภัยพิบัติทางธรรมชาติ การมองหาไซต์สำรองหรือ DR Site จึงมีความสำคัญและจำเป็นอย่างมาก ทำให้เรามั่นใจได้ว่าไม่ว่าจะเกิดเหตุการณ์อะไรขึ้น ระบบของเราจะยังสามารถทำงานต่อไปได้

Storage and Backup - การเก็บข้อมูลบน Data Center ของตนเองอาจยังปลอดภัย หรือมีข้อจำกัดเรื่องของพื้นที่การจัดเก็บ การใช้บริการ Cloud ช่วยให้เรามั่นใจได้ว่ามีข้อมูลสำรองกรณีที่ต้องการใช้ โดยไม่มีข้อจำกัดเรื่องของการปรับขนาดในอนาคต

2.1.2 การประมวลผลบนระบบคลาวด์ด้วย AWS

ภาพที่ 2.1 โลโก้ของ Amazon Web Services

Amazon Web Services (AWS) เป็นแพลตฟอร์มระบบคลาวด์ที่ครอบคลุมและใช้งานกันมากที่สุด โดยมีบริการอันโดดเด่นเต็มรูปแบบกว่า 165 บริการจากศูนย์ข้อมูลไปทั่วโลก ลูกค้ากว่าหลายล้านคน ทั้งสตาร์ทอัพที่เติบโตเร็วที่สุด องค์กรที่ใหญ่ที่สุด และหน่วยงานราชการชั้นนำ ต่างวางใจให้ AWS เพิ่มประสิทธิภาพให้โครงสร้างพื้นฐาน เพิ่มความคล่องตัว และลดค่าใช้จ่าย

2.1.2.1 Amazon Elastic Kubernetes Service

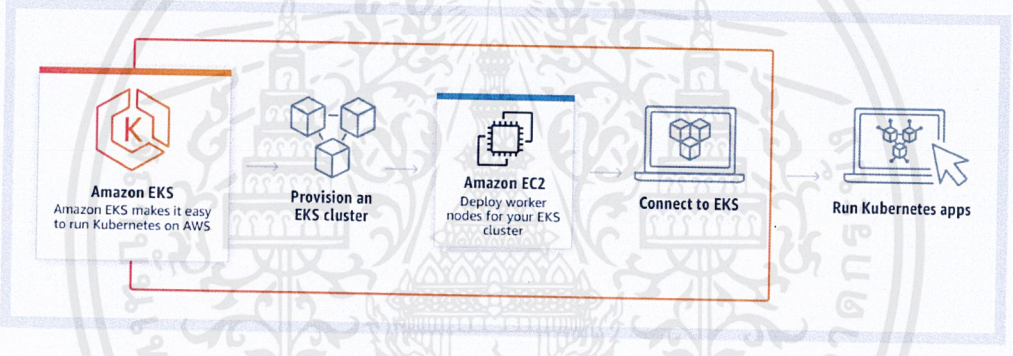
Amazon Elastic Kubernetes Service (EKS) เป็นเซอร์วิสสำหรับจัดการ kubernetes บน AWS โดยที่ผู้ใช้งานไม่มีความจำเป็นต้องมาจัดการดูแลส่วน control plane ของ kubernetes ด้วยตัวเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Amazon EKS รัน kubernetes ในหลายๆ Availability Zones เพื่อให้ได้การใช้งานอย่างมีประสิทธิภาพ ช่วยตรวจสอบและสร้างเครื่อง control plane ใหม่มาทดแทนหากเครื่องเก่าสถานะ unhealthy และรองรับการอัพเกรดเวอร์ชั่นโดยอัตโนมัติ

EKS ยังสามารถที่จะใช้งานร่วมกับเซอร์วิสอื่นๆ ของ AWS เพื่อช่วยในการปรับเปลี่ยนขนาดและความปลอดภัยของแอปพลิเคชั่น

EKS นั้นใช้ kubernetes เวอร์ชั่นล่าสุด ทำให้สามารถใช้งานปลั๊กอินและเครื่องมือต่างๆ จาก kubernetes community ได้ แอปพลิเคชั่นที่รันอยู่บน EKS สามารถเข้ากันได้กับ kubernetes ปกติ ดังนั้นการย้ายแอปพลิเคชั่นบน kubernetes มาไว้ใน EKS จึงไม่จำเป็นต้องแก้ไขโค้ดใดๆ



ภาพที่ 2.2 วิธีการทำงานของ Amazon EKS

2.1.1.2 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) เป็นบริการเว็บที่ทำให้ความสามารถในการประมวลผลอย่างปลอดภัยและปรับขนาดได้ในระบบคลาวด์ ซึ่งออกแบบมาเพื่อช่วยให้ Developer ประมวลผลแบบ Web-Scale ได้ง่ายขึ้น

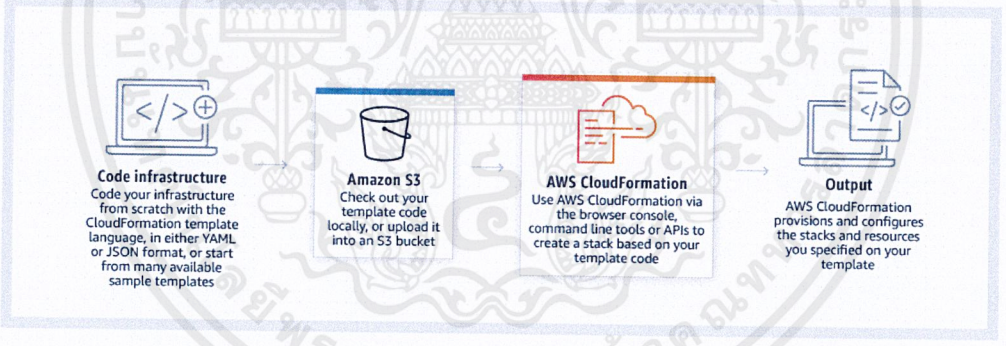
อินเทอร์เฟซบริการเว็บของ Amazon EC2 ที่ใช้งานง่ายช่วยให้ได้รับประสิทธิภาพและกำหนดค่าความสามารถได้โดยมีข้อผิดพลาดน้อยที่สุด โดยจะควบคุมทรัพยากรการประมวลผลได้อย่างสมบูรณ์ รวมทั้งเรียกใช้ด้วยการเชื่อมต่อสภาพแวดล้อมการประมวลผลที่ผ่านการรับรองของ Amazon

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Amazon EC2 ลดเวลาที่ใช้ในการเริ่มต้นอินสแตนซ์เซิร์ฟเวอร์ใหม่ให้เหลือภายในไม่กี่นาที ทำให้สามารถปรับขนาดความสามารถให้มากขึ้นหรือน้อยลงให้สอดคล้องกับการประมวลผลที่เปลี่ยนแปลงได้ มอบเครื่องมือที่ช่วยให้ Developer สามารถสร้างแอปพลิเคชันที่สามารถกู้คืนได้หลังจากเกิดข้อผิดพลาด และ Amazon EC2 นั้นจ่ายค่าบริการตามความสามารถที่ใช้จริงเท่านั้น

2.1.1.3 AWS CloudFormation

AWS CloudFormation ใช้ภาษาทั่วไปที่ทำให้สามารถอธิบายและจัดหาทรัพยากรโครงสร้างพื้นฐานทั้งหมดในสภาพแวดล้อมของคลาวด์ได้ CloudFormation ช่วยให้ใช้ไฟล์ข้อความที่เรียบง่ายในการสร้างแบบจำลองและจัดเตรียมทรัพยากรทั้งหมดที่จำเป็นสำหรับแอปพลิเคชันของคุณในทุกภูมิภาคและทุกบัญชีได้โดยอัตโนมัติและปลอดภัย ไฟล์นี้คือแหล่งข้อมูลเดียวสำหรับสภาพแวดล้อม Cloud ของคุณ



ภาพที่ 2.3 วิธีการทำงานของ AWS CloudFormation

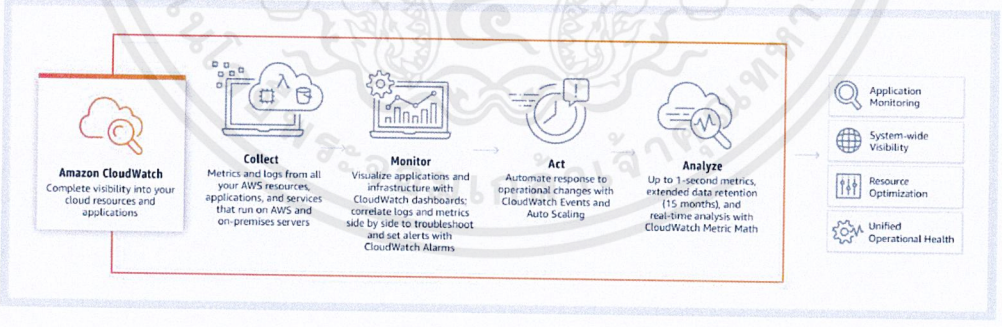
2.1.1.4 Amazon CloudWatch

Amazon CloudWatch เป็นบริการเฝ้าระวังและดูแลที่สร้างขึ้นสำหรับวิศวกร DevOps, Developer, วิศวกรด้านความน่าเชื่อถือของไซต์ (SRE) และผู้จัดการแผนกไอที CloudWatch มีบริการข้อมูลและข้อมูลเชิงลึกที่ปฏิบัติตามได้เพื่อเฝ้าสังเกตแอปพลิเคชัน ตอบสนองต่อการเปลี่ยนแปลงของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประสิทธิภาพตลอดทั้งระบบ ปรับการใช้ทรัพยากรให้เหมาะสม และดูภาพรวมการทำงานของแอปพลิเคชัน CloudWatch จะรวบรวมข้อมูลที่ได้จากการเฝ้าสังเกตและการดำเนินการต่างๆ ไว้เป็นบันทึก ตัววัด และเหตุการณ์ ช่วยให้คุณมองเห็นภาพรวมของทรัพยากร แอปพลิเคชัน และบริการต่างๆ ของ AWS ที่ทำงานบน AWS และเซิร์ฟเวอร์ในองค์กรได้ คุณสามารถใช้ CloudWatch เพื่อตรวจจับการทำงานที่ผิดปกติในสภาพแวดล้อม ตั้งการแจ้งเตือน แสดงข้อมูลบันทึกและตัววัดด้วยแผนภาพควบคู่ด้านข้าง ดำเนินการจัดการโดยอัตโนมัติ แก้ไขปัญหา และสำรวจข้อมูลเชิงลึกให้แอปพลิเคชันของคุณทำงานอย่างราบรื่น

CloudWatch จะรวบรวมข้อมูลที่ได้จากการเฝ้าสังเกตและการดำเนินการต่างๆ ไว้เป็นบันทึก ตัววัด และเหตุการณ์ และแสดงภาพโดยใช้แดชบอร์ดอัตโนมัติ เพื่อให้คุณเห็นภาพรวมของทรัพยากร แอปพลิเคชัน และบริการต่างๆ ของ AWS ที่ทำงานบน AWS และภายในอาคารได้ เชื่อมโยงความสัมพันธ์ตัววัดและบันทึกของคุณเพื่อให้เข้าใจสภาพและประสิทธิภาพของทรัพยากรมากขึ้น คุณยังสามารถสร้างการแจ้งเตือนตามเกณฑ์ค่าตัววัดที่ระบุ หรือการแจ้งเตือนที่สามารถเฝ้าระวังการทำงานของตัววัดที่ผิดปกติตามอัลกอริทึมแมชชีนเลิร์นนิ่งได้ หากต้องการจัดการอย่างรวดเร็ว สามารถตั้งค่าการจัดการอัตโนมัติเพื่อแจ้งเตือนคุณหากการแจ้งเตือนทริกเกอร์และเริ่มใช้ Auto Scaling ได้ เช่น เพื่อลดเวลาเฉลี่ยการแก้ปัญหา คุณสามารถเจาะลึกและวิเคราะห์ตัววัด บันทึก รายการติดตาม เพื่อให้เข้าใจวิธีการปรับปรุงประสิทธิภาพแอปพลิเคชันได้ดียิ่งขึ้น



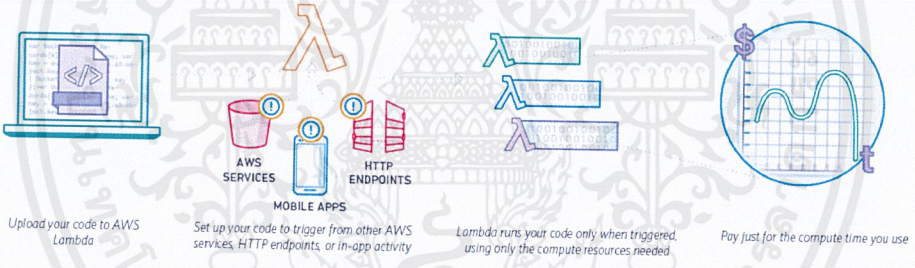
ภาพที่ 2.4 วิธีการทำงานของ CloudWatch

2.1.2.5 AWS Lambda

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AWS Lambda เป็นบริการประมวลผลแบบไร้เซิร์ฟเวอร์ที่รันโค้ดเพื่อตอบสนองต่อเหตุการณ์ต่างๆ และจัดการทรัพยากรการประมวลผลพื้นฐานโดยอัตโนมัติ โดยสามารถใช้ AWS Lambda เพื่อขยายบริการอื่นๆ ของ AWS โดยใช้ตรรกะที่กำหนดเอง หรือสร้างบริการแบ็คเอนด์ของตัวเองที่ทำงานด้วยขนาด ประสิทธิภาพ และความปลอดภัยระดับ AWS และ AWS Lambda สามารถรันโค้ดได้อัตโนมัติเพื่อตอบสนองต่อหลายเหตุการณ์ เช่น คำขอ HTTP ผ่าน Amazon API Gateway, การแก้ไขอ็อบเจ็กต์ในบัคเก็ต Amazon S3, การอัปเดตตารางใน Amazon DynamoDB และการเปลี่ยนสถานะใน AWS Step Functions

Lambda จะรันโค้ดในโครงสร้างพื้นฐานด้านการประมวลผลที่มีประสิทธิภาพสูงและดำเนินการจัดการทรัพยากรในการประมวลผลทั้งหมด รวมถึงการบำรุงรักษาเซิร์ฟเวอร์และระบบปฏิบัติการ การจัดเตรียมความจุและการปรับขนาดอัตโนมัติ การปรับใช้โค้ดและแพทช์ความปลอดภัย ตลอดจนการตรวจสอบและบันทึกโค้ด สิ่งที่ต้องทำคือป้อนโค้ดเท่านั้น



ภาพที่ 2.5 วิธีการทำงานของ AWS Lambda

2.1.3 Container

คำว่า "คอนเทนเนอร์" (container) เป็นเทคนิคการจัดการแพ็กเกจซอฟต์แวร์ประเภทหนึ่งที่มีความสัมพันธ์กับเทคนิค virtualization ที่อยู่ในโลกองค์กรมานาน ดังนั้นการอธิบายว่าคอนเทนเนอร์คืออะไร จึงมักถูกเปรียบเทียบว่าแตกต่างกับ virtualization อย่างไร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เทคนิค virtualization คือการสร้างคอมพิวเตอร์เสมือน (virtual machine หรือ VM) ที่มีทั้งซีพียู แรม สตอเรจ ระบบปฏิบัติการ ฯลฯ ขึ้นมารันบนคอมพิวเตอร์จริงๆ อีกทีหนึ่ง โดยตัวระบบปฏิบัติการของคอมพิวเตอร์เสมือน (Guest OS) จะไม่รู้ว่าตัวเองรันอยู่บน VM แต่เข้าใจว่ารันอยู่บนฮาร์ดแวร์คอมพิวเตอร์จริงๆ

วิธีการนี้ทำให้เกิดการแยกส่วน (isolation) ระหว่าง VM แต่ละตัวอย่างสมบูรณ์ สามารถรันระบบปฏิบัติการที่แตกต่างกันระหว่าง Guest OS กับ Host OS ได้ แต่ข้อเสียคือใช้ทรัพยากรซ้ำซ้อน ทำงานช้า เปลืองพื้นที่เก็บ OS และซอฟต์แวร์ต่างๆ ที่มักจะใช้เหมือนกันใน VM ทุกตัว

คอนเทนเนอร์จึงถูกสร้างขึ้นมาเพื่อแก้ปัญหาข้างต้น โดยมีฮาร์ดแวร์และ OS เพียงชุดเดียวกัน ลดความซ้ำซ้อนของการใช้ทรัพยากรลง ส่วนตัวแอพพลิเคชันและซอฟต์แวร์ซึ่งเป็นจุดที่แตกต่างกันไปก็จะมี "container" (เทียบได้กับ VM) มาครอบเพื่อแบ่งส่วนทรัพยากรไว้ไม่ให้ยุ่งกัน

จุดเด่นของคอนเทนเนอร์จึงเป็นเรื่องการใช้ทรัพยากรที่น้อยกว่า virtualization มาก อิมเมจของคอนเทนเนอร์อาจมีขนาดเพียงกี่ไม่กี่สิบ MB ในขณะที่อิมเมจของ VM ต้องใช้พื้นที่ระดับหลาย GB นอกจากนี้ ระยะเวลาที่ใช้บูต, ซีพียูและปริมาณแรมที่ต้องใช้ ก็ลดลงตามไปด้วย ส่งผลให้เซิร์ฟเวอร์หนึ่งเครื่องสามารถยัดคอนเทนเนอร์จำนวนมากกว่าการรัน VM ที่ให้ผลแบบเดียวกันถึง 2-3 เท่าตัว

บางครั้ง คอนเทนเนอร์ถูกเรียกชื่อในทางเทคนิคว่า Operating-system-level virtualization หรือการสร้าง VM ที่ระดับ OS โดยเราไม่ต้องสร้างเครื่องคอมพิวเตอร์เสมือนขึ้นมาทั้งตัว

ข้อเสียของคอนเทนเนอร์ก็ย่อมเป็นความยืดหยุ่นที่น้อยกว่า virtualization แบบดั้งเดิม โดยเฉพาะไม่สามารถใช้ OS ที่แตกต่างกันระหว่าง Guest และ Host ได้ (เพราะจุดเด่นของคอนเทนเนอร์คือการแชร์ OS ก็อปปี้เดียวกัน)

แนวคิดของคอนเทนเนอร์ไม่ใช่เรื่องใหม่ ในโลกของยูนิกซ์เกิดแนวคิดนี้ขึ้นมาตั้งแต่ปี 2000 จากแนวคิด jails ของ FreeBSD จากนั้นในปี 2004 ระบบปฏิบัติการ Solaris ของบริษัท Sun Microsystems ก็มีฟีเจอร์แบบเดียวกันโดยใช้ชื่อว่า Zones (หรือ Solaris Containers) ฝั่งลินุกซ์เองก็นำไอเดียนี้มาสืบสานต่อในโครงการอย่าง OpenVZ หรือ LXC (Linux Containers)

แต่คอนเทนเนอร์กลายมาเป็นเรื่องแพร่หลายในวงกว้างจาก Docker ที่เริ่มต้นในปี 2013 ซึ่งช่วงแรกยังอิงอยู่บนโครงการยุคก่อนหน้าอย่าง LXC หรือ libvirt แต่ภายหลัง Docker ก็พัฒนาส่วนต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขึ้นมาเอง (libcontainer) จนสมบูรณ์พร้อมใช้งาน ทำให้แนวคิดคอนเทนเนอร์ "จุดติด" และได้รับการยอมรับในวงการอย่างรวดเร็ว มีตัวอย่างการใช้งานจากบริษัทใหญ่ๆ อย่างกูเกิลที่พัฒนาฟีเจอร์ของ Google Compute Engine ให้รองรับอย่างรวดเร็ว

ถึงแม้ Docker จะถูกสร้างขึ้นมาเพื่อลินุกซ์ แต่ความนิยมของมันทำให้ระบบปฏิบัติการฝั่งวินโดวส์ ทำให้ไมโครซอฟท์เข้ามาร่วมวงตั้งแต่ปี 2014 และสำเร็จลุล่วงใน Windows Server 2016

ความสำเร็จของ Docker ทำให้เกิดคู่แข่งขึ้นบ้าง เช่น Rocket หรือ rkt ของบริษัท CoreOS (ปัจจุบันถูก Red Hat ซื้อไปแล้ว) ทำให้โลกคอนเทนเนอร์แยกออกเป็นสองส่วน แต่ภายหลังก็หาทางออกได้ ด้วยการออกมาตรฐานกลางภายใต้การดูแลของ Open Container Initiative (OCI) (ภายหลังยังพัฒนาต่อมาเป็นโครงการ containerd และ runc ซึ่งจะไม่กล่าวถึงในที่นี้)

2.1.4 Kebernetes

2.1.4.1 ประวัติของ kubernetes

ชื่อ Kubernetes มาจากภาษากรีก **κυβερνήτης** ซึ่งแปลเป็นภาษาอังกฤษได้ว่า helmsman หรือ "กัปตันเรือ" จึงเป็นที่มาของโลโก้รูปพวงมาลัยเรือ (คำว่า Kubernetes มีรากศัพท์มาจากคำเดียวกับคำว่า Governor ด้วย) แต่เนื่องจากชื่อโครงการค่อนข้างยาว เรียกยาก ในวงการเลยมักเรียกกันย่อๆ ว่า K8s หรือบ้างก็เรียก Kube

Kubernetes เปิดตัวต่อสาธารณะครั้งแรกในเดือนมิถุนายน 2014 (เป็นข่าวใน Blognone ครั้งแรกเดือนกรกฎาคม 2014) แต่ประวัติของมันยาวนานกว่านั้น

กูเกิลมีธรรมเนียมการสร้างซอฟต์แวร์เพื่อใช้เองภายในอยู่แล้ว และมีระบบจัดการคอนเทนเนอร์ของตัวเองเรียกว่า Borg (มาจากเผ่าเอเลี่ยนกึ่งหุ่นยนต์ในซีรีส์ Star Trek) ซึ่งทำงานอยู่บน cgroups ฟีเจอร์ของเคอร์เนลลินุกซ์ และเป็นต้นแบบของคอนเทนเนอร์ในยุคปัจจุบัน

Borg วิวัฒนาการมาเรื่อยๆ โดยไม่ได้คำนึงถึงการออกแบบสถาปัตยกรรมมาตั้งแต่ต้น เพราะเน้นสร้างฟีเจอร์ใหม่ตามความต้องการของทีมงานภายในแต่ละทีม สุดท้าย Borg จึงไปต่อไม่ไหว กูเกิลจึง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สร้างซอฟต์แวร์ตัวใหม่ขึ้นมาแทนโดยใช้ชื่อว่า Omega ซึ่งนำแนวคิดหลายอย่างจาก Borg มาใช้งาน แล้วเพิ่มระบบการเก็บสถานะของคลัสเตอร์ไว้ที่ส่วนกลางเพื่อป้องกันข้อมูลขัดแย้งกันเอง

เมื่อเทคโนโลยีคอนเทนเนอร์เริ่มได้รับความนิยม และกูเกิลเองก็มีบริการลักษณะนี้ต่อลูกค้าภายนอกบริษัท (Google Cloud Platform) วิศวกรกลุ่มหนึ่งของกูเกิลจึงตัดสินใจสร้างโครงการแบบนี้ขึ้นมาใหม่อีกครั้ง ซึ่งก็คือ Kubernetes

เนื่องจาก Kubernetes สืบทอดแนวคิดหลายอย่างมาจาก Borg ในช่วงแรก โครงการจึงมีโค้ดเนมว่า Project Seven ซึ่งมาจากตัวละคร Seven of Nine บอร์กสาวจากเรื่อง Star Trek เช่นกัน (และเป็นเหตุผลว่าทำไมไอคอนพวงมาลัยเรือถึงมี 7 แกน)

Kubernetes ได้รับความนิยมอย่างรวดเร็ว และได้รับการสนับสนุนจากบริษัทยักษ์ใหญ่อย่าง Microsoft, Red Hat, IBM จนเมื่อกูเกิลออก Kubernetes 1.0 ในเดือนกรกฎาคม 2015 ก็ตัดสินใจตั้งมูลนิธิ Cloud Native Computing Foundation (CNCF) มาดูแลโครงการต่อแทน

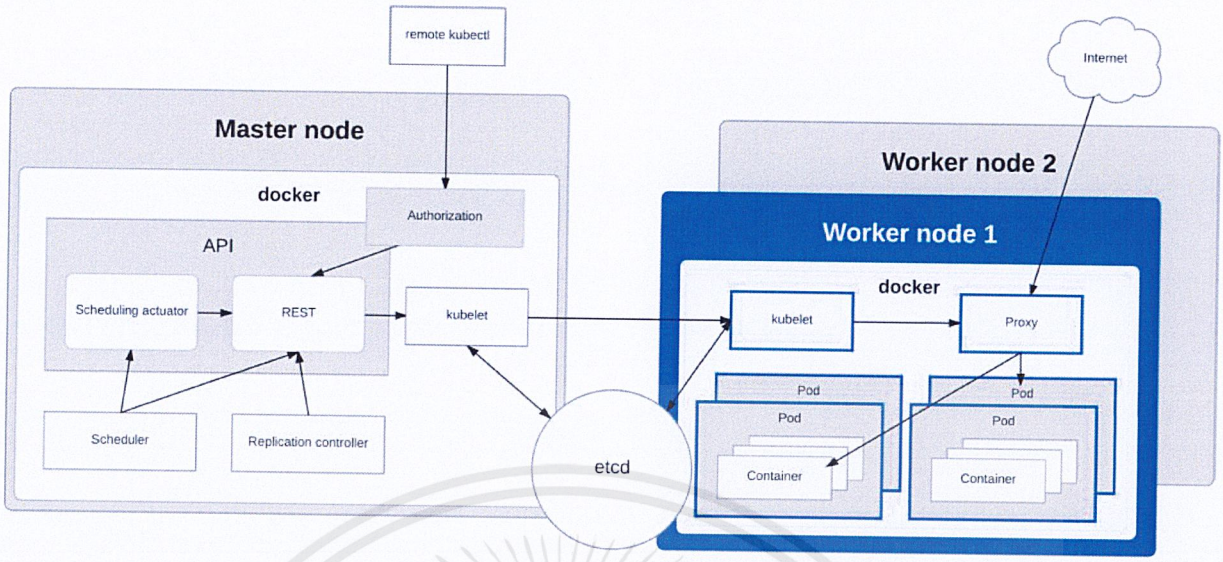
2.1.4.2 หลักการของ Kubernetes

หลักการของ kubernete คือการทำ Orchestration ต้องการให้เป็นในรูปแบบไหนสามารถกำหนดให้มันจัดการตามรูปแบบที่ต้องการได้เลย โดยใน Kubernetes จะมีไฟล์ที่เป็น .yaml หรือ manifest จะคล้ายกับไฟล์ configuration ของ kubernetes ที่ใช้ระบุว่าต้องการให้ออกมาเป็นรูปแบบไหน เพียงแค่ระบุว่าอยากได้แบบไหน ผ่านไฟล์ .yaml แล้วมันจะดำเนินการตามไฟล์นั้น โดนใน Kubernetes จะมี kubectl ช่วยในการจัดการต่างๆ

สถาปัตยกรรมคลัสเตอร์ของ Kubernetes ไม่ต่างอะไรจากคลัสเตอร์ทั่วๆ ไป นั่นคือมีเครื่องที่เป็น Master และ Slave (ในโลกของ Kubernetes ใช้คำว่า Node)

โดยใน Kubernetes จะมี Node ที่ใช้อยู่ 2 ประเภท คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.6 Master node และ Worker node ของ Kubernetes

1. Master node : เป็นส่วน front-end สำหรับ Client ในการเข้าถึงเพื่อสั่งงาน หรือ ตั้งค่าต่างๆ เป็นส่วนที่คอยควบคุม จัดการ component ที่อยู่ภายใน รวมถึงการทำ Scalability และ มีหน้า Dashboard ที่สามารถ monitor สถานะ หรือดูว่ามีการทำงานใดบ้างอยู่ภายใน component ข้างในประกอบด้วย

Kube-apiserver : คือ REST webservice เป็นส่วนที่รับคำสั่ง เมื่อมีการส่ง manifest (ไฟล์ yaml) เข้ามาโดยจะผ่านทางคำสั่ง kubectl apply โดย manifest จะอยู่ในรูปของไฟล์ JSON ซึ่งจะส่งคำสั่งให้ API Server จากนั้น API Server จะเป็นคนจัดการว่าใน manifest มีการสั่งงานให้ทำอะไรบ้าง ใครควรเป็นคนทำคำสั่งนี่ต่อไป ก็จะส่งงานไปให้คนนั้น

Kube-controller-manager (Replication Controller) : เป็นตัวควบคุม การทำงานระหว่าง node โดยการเชื่อมต่อระหว่าง node จะทำโดยการ เรียกผ่าน Service ตัว kube-controller จะเป็นคนจัดการในส่วนนี้ รวมการ scale เพิ่ม / ลด ของ pod ภายใน เครื่องที่เป็น minions การ scale นอกจากจะรับคำสั่งจาก client ที่สั่งเข้ามาผ่านทาง Api-server แล้ว ยังจะทำการ scale ในกรณีของ Actual state ไม่เท่ากับ Desired state

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Actual state คือสถานะปัจจุบันของ pod ว่า pod มีจำนวนเท่าใด ส่วน Desired state เป็นสถานะตอนแรกที่มาจาก manifest หรือ เรียกอีกอย่างว่าเป็นการ Control Plan คือเป็น ค่าตั้งแต่ต้น ว่าแต่ล่ะ pod ต้องมีจำนวนเท่าใด

เมื่อเริ่มแรกจำนวน pod จะถูกสร้างตามจำนวนที่ระบุไว้ใน manifest แล้วถ้าใช้ไประยะหนึ่ง pod บางตัวหยุดทำงานทำให้ Actual state ไม่เท่ากับ Desired state ส่วนของ Kube-controller-manager ก็จะคอย monitor และทำการ เพิ่ม pod ให้เท่าตามเดิม

kube-scheduler : จะทำหน้าที่ auto-scales ของ pod ตามที่ตั้ง policy ไว้ ว่าให้เพิ่ม pod ตอนไหน หรือ ให้เพิ่มตอน resource ของเครื่องเหลือเท่าไร ตัวอย่างเช่น ถ้ามีการใช้ CPU มากกว่า 80 % ให้ทำการเพิ่ม pod เป็นต้น เป็นตัวที่คอย monitor resource ว่าถึงเมื่อไรควรจะขยาย resource เพื่อที่ pod ใน minions จะได้เพียงพอต่อจำนวนการใช้งาน

Key/Value stores (etcd) : เป็น Database สำหรับเก็บข้อมูล configuration ของ manifest ว่าแต่ล่ะครั้งที่มีการส่งไฟล์ .yaml เข้ามามีการตั้งค่าแบบไหน ถือว่าเป็นส่วนที่สำคัญส่วนหนึ่ง เพราะใช้เก็บ configuration ถ้าส่วนนี้พังไป Kubernetes จะไม่สามารถใช้งานได้ ในการใช้งานจริงควรต้องทำการ Backup ในส่วนนี้

2. Worker node / Minion : เป็นเครื่องที่ใช้ run งานจริง มีหน้าที่ทำงานอย่างเดียว สามารถใช้ทั้ง VM หรือ เครื่อง Server จริงๆ มาทำก็ได้ แต่ล่ะ Worker node จะมีหน้าที่ run pod และจะถูกจัดการโดย Master Node ประกอบด้วย 3 ส่วน

Kubelet : เป็น agent ในการรับคำสั่งจาก Master ถ้าไม่มี kubelet เครื่อง minion จะไม่สามารถรับคำสั่งจาก Master ได้ และจะไม่ทำงานตามคำสั่ง โดย kubelet จะคอย monitor งานที่ต้องทำจาก API Server มีการยิงคำสั่งเข้ามาที่ kubelet จากนั้น kubelet จะเป็นตัวจัดการภายใน minion ให้ทำตามคำสั่งที่ได้รับมา ไม่ว่าจะเป็น สั่งให้ Docker Engine สร้าง image ใหม่ขึ้นมา หรือ config Network เป็นต้น นอกจากจะเป็นตัวที่รับคำสั่งจาก master ตัวมันยังทำหน้าที่รายงานผลและ log กลับไปหา Master ผ่าน port :10255 อีกด้วย

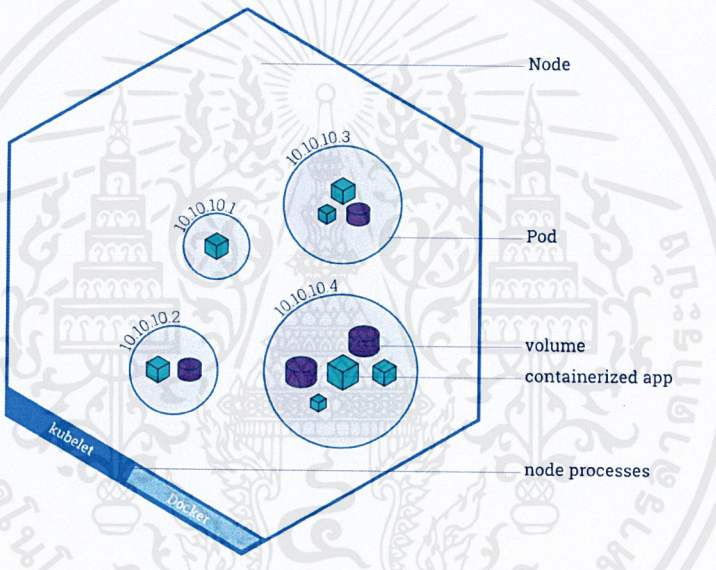
Container Engine : เป็นส่วนของตัว Container Engine ซึ่ง Kubernetes ไม่ได้ Support ของ Docker อย่างเดียว แต่ส่วนใหญ่ที่ใช้งานจะเป็น Docker Engine เป็นตัวที่ทำหน้าที่ ดึง image มา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จาก cloud ทำการ start image ให้ run หรือทำงานได้ ซึ่ง image จะอยู่ภายใต้ pod โดย Kubelet จะเป็นคนสร้าง pod แล้วสั่งให้ Docker Engine ทำการ start image แล้ว run container ขึ้นมาใช้งาน

Kube-proxy : ทำหน้าที่ในส่วนของ network ถือว่าเป็น kubernetes network จะเป็นตัวที่ Assign ip ให้ pod ใช้งาน แล้วยังเป็นตัวทำหน้าที่ Load Balance จากการเรียกของ Service กระจายไปยัง pod แต่ละตัว

3. Pod



ภาพที่ 2.7 Pods ใน Kubernetes cluster

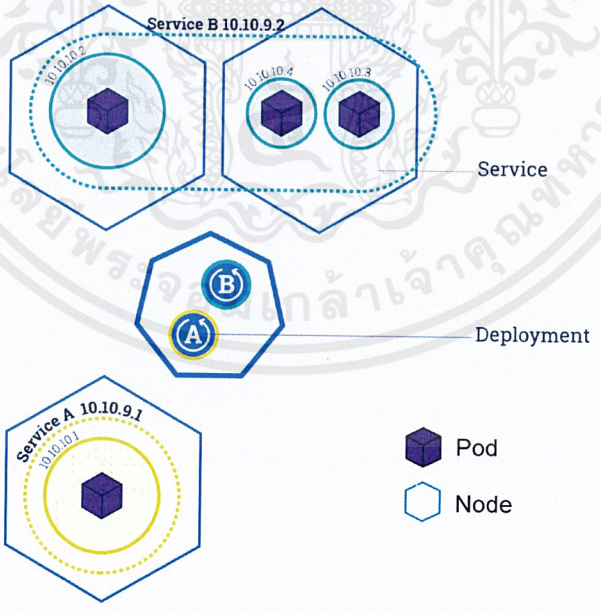
Pod เป็นกลุ่มของ Container การทำงานของ Kubernetes ที่เด่นในเรื่องของการ scale มันคือการ scale pod โดยใน pod สามารถมี Container ตั้งแต่ 1 เครื่องขึ้นไป หรือสามารถมีได้หลาย Container แต่ที่ทาง Kubernetes แนะนำคือ 1 pod ควรจะมี 1 Container เพราะถ้าสร้าง Container หลายตัวไว้ใน pod เดียวกัน ถ้าเกิด pod นั้นหยุดทำงาน Container ทุกตัวใน pod จะไม่สามารถใช้งานได้ ส่วนใหญ่กรณีที่ใช้ Container มากกว่า 1 ตัวใน 1 pod จะเป็น Container ที่ต้องการทำ Load

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Balance ภายใน คือมี Container ที่เป็น Load Balance และ Container ที่ใช้ run งาน นอกจากนี้ใน Container ยังมี resource ต่างๆ ที่จำเป็นสำหรับ container เช่น storage, network ip และมันยังเก็บ config ต่างๆ เกี่ยวกับ container แต่ละอัน เช่น ต้องรันยังไง, ต้องคุยผ่าน port ไหน, Container ทุกตัวที่อยู่ใน pod จะใช้ resource ร่วมกันเสมอ

4. Services : เนื่องจาก pod ที่ใช้ run Container ที่ติดตั้งแอปพลิเคชันมีโอกาส เพิ่มหรือลดของ pod ตลอดเวลาด้วยคำสั่งของ Client หรือการ Auto scale จากเวลาที่ Actual state ไม่เท่ากับ Desired state ก็ตาม ทำให้ IP ของ pod ที่ใช้ run Application มีโอกาสเปลี่ยนแปลงได้ตลอดเวลา ทำให้ไม่สามารถเรียกใช้งาน หรือเชื่อมต่อ Application ผ่าน ip-address เพราะฉะนั้นสิ่งที่ทำให้การทำงานในคลัสเตอร์ของ Kubernetes สามารถค้นหาและสื่อสารกับโลกภายนอกได้ คือ Service

Service ทำหน้าที่กระจาย Load balance ไปยังแต่ละ pod สามารถทำหน้าที่เป็น API Gateway ในตัวทำให้เวลามีการเพิ่มหรือลดของ pod ตัวที่ทำการเรียกมายัง pod ดังกล่าวจะไม่รู้สึกว่าหลังบ้านของ pod มีการเปลี่ยนแปลงเพราะมี service เป็นตัวกลาง ทั้งเป็นตัวกลางจาก Client มายัง pod หรือเป็นตัวกลางระหว่าง pod เองก็ตาม



ภาพที่ 2.8 Service ใน Kubernetes Cluster

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

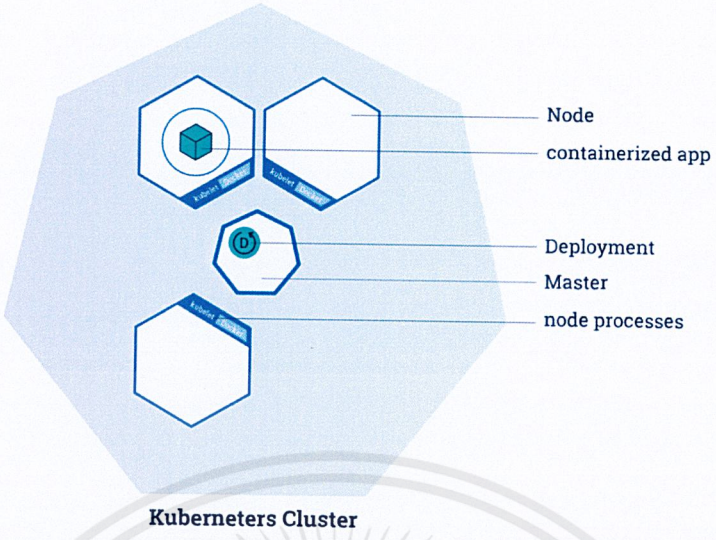
นอกจากการที่ Service ทำให้ผู้ใช้ที่เรียก เข้ามาไม่รู้สึกได้รับผลกระทบเมื่อมีการเพิ่มหรือลด ของ pod แล้ว service ยังช่วยในเรื่องของการทำ Blue Green – Deployment คือการ Deploy New version โดยไม่เกิด Down-time ซึ่ง Default ของ Kubernetes เป็น Blue Green -Deployment อยู่แล้วโดยหลักการของการทำ Blue Green – Deployment ถ้ามีการเปลี่ยนเวอร์ชั่น Kubernetes จะยังคงรักษา work-load ไว้เท่าเดิม ไม่ลดลง เคยมี 2 เครื่องก็จะมี 2 เครื่องเท่าเดิม คือถ้ามีการเปลี่ยนเวอร์ชั่นจาก 1 ไปเป็น 2 ตอนแรกมี pod ที่เป็นเวอร์ชั่น 1 จำนวน 2 pod งานที่มันจะทำคือ สร้าง pod แรกที่เป็น version 2 (New-version) ก่อน และจะสร้าง pod อีกเครื่องให้เป็น version 2 เพื่อไม่ให้ work-load ลดลง แล้วจะปลด version เก่าทิ้ง

การเขียน code ต้องเรียกด้วย DNS ทั้งหมดห้ามเรียกด้วย ip ไม่งั้นจะไม่สามารถ scale หรือ ทำการ update version ได้เลย

5. Labels : ในการตัดสินใจว่า Service ไหนจะเชื่อมต่อไปยัง pod ใด Kubernest ใช้ labels ในการ route traffic ลง pod แต่ล่ะตัว การตั้งค่า labels จะถูกตั้งค่าไว้ที่ manifest (config file) ตัวอย่างเช่น Service ถูกตั้งค่าไว้ว่ามี label คือ Prod และ BE เพราะฉนั้น service ตัวนี้จะทำการกระจาย Load ไปยัง Pod ที่มี label ตรงกับ service แม้ว่า pod ที่กระจายไปจะมี label ของ version ก็ตาม ถ้ามีครบตรงตาม Service มันก็จะทำการกระจาย Load ไปยังทุก pod สามารถใช้ในเรื่องของการอัพเดทเวอร์ชั่นของแอพพลิเคชั่นได้ แต่ถ้าอยากทำการระบุเวอร์ชั่น สามารถทำได้โดยการเพิ่ม label ของเวอร์ชั่นที่ต้องการใน Service มันก็จะทำการกระจาย Load ไปยังเวอร์ชั่นที่ระบุ

6. Deployment : ใน manifest file จะมี type กำกับที่จะบ่งบอกว่าเราต้องการสร้างอะไร type หนึ่งในนั้นคือ deployment

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.9 Kubernetes Deployment

เบื้องต้นเราได้บอกว่า ReplicationController เป็นตัวสร้าง Pod แล้ว แต่เจ้าตัว Deployment คือผู้สร้าง ReplicationController อีกทีนึง แต่จริงๆ มันใช้ ReplicaSet ที่เป็น API ในการสร้าง pod อาจกล่าวได้ว่า ReplicaSet เป็น next-generation ของ ReplicationController

ครั้งแรกเมื่อเราสร้าง Deployment มันจะไปสร้าง ReplicaSet เพื่อให้ ReplicaSet เป็นคนสร้าง pod ตามที่ระบุไว้ใน manifest

พอมีการเปลี่ยนแปลง หรือการเพิ่มเวอร์ชั่น Deployment จะสั่งสร้าง ReplicaSet เพื่อให้ ReplicaSet ไปสั่งสร้าง pod เวอร์ชั่นที่มี การเปลี่ยนแปลงแก้ไข หลังจาก pod ที่สร้างใช้งานได้มันจะทำการลดขนาด ReplicaSet เดิมลง และเพิ่มขนาด ReplicaSet ใหม่อีก ทำซ้ำไปเรื่อยๆ จนกระทั่ง ReplicaSet ใหม่ขนาดเท่าเดิม และอันเดิมขนาดเหลือ 0

Deployment จะแทนที่ Pod เก่าด้วย Pod ใหม่ทีละ Pod ทำให้เวลา deploy แล้วระบบเราจะไม่มี downtime เกิดขึ้นเลย

นอกจากนี้ Deployment ยังจะเก็บ ReplicaSet ของเก่าไว้ด้วย นั่นแปลว่าถ้า deploy แล้วพังก็สามารถสั่ง kubectl rollout undo เพื่อถอยกลับได้อีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. Namespace : เป็นตัวแบ่ง Project ของ pod เพราะ เมื่อเรามีการใช้ cluster กัน หลายๆ project อาจจะเกิดความสับสน เราสามารถแยกของไม่ให้ปนกันได้ด้วยการใช้ Namespace

เวลาเราจะพิมพ์คำสั่งต่างๆ ถ้าระบุ kubectl --namespace app2 ... ก็จะเห็นเฉพาะของที่อยู่ใน namespace นั้นๆ โดยปกติแล้ว Kubernetes จะมี namespace สำคัญๆ 2 อัน ก็คือ default ที่เราใช้ได้เวลาไม่ระบุชื่อ namespace และ kube-system ที่เก็บ service ของ Kubernetes

แน่นอนว่า Kubernetes (หรือ container orchestration) ไม่ใช่โซลูชันที่เหมาะสำหรับทุกคน แถมการใช้งานยังต้องแลกมาด้วยความซับซ้อนของระบบที่เพิ่มจากเดิมมาก

รูปแบบงานที่เหมาะสมกับการทำ container orchestration จึงควรมีปัจจัยเหล่านี้

แอพพลิเคชันที่ใช้งาน ต้องถูกปรับให้อยู่ในสภาพแวดล้อมแบบ container มาก่อนแล้ว ถึงค่อยมาทำ orchestration ในภายหลัง หากเป็นแอพพลิเคชันยุคเก่า (legacy) ที่ไม่ได้ปรับให้เป็น container มาก่อน ก็ต้องไปทำให้เสร็จก่อน

งานมีความต้องการ scalable ขยายขีดความสามารถให้รองรับจำนวนเวิร์คโหลดที่แปรเปลี่ยนไป หากไม่จำเป็นขนาดนั้น ก็ใช้วิธีรันงานใน container เพียงอย่างเดียว (docker-compose) ไปก่อน แล้วค่อยขยับไปใช้ Kubernetes ในภายหลังได้ ซึ่งตรงนี้มีเครื่องมืออย่าง Kompose ช่วยแปลงอยู่แล้ว

ตัวบริการมีขนาดใหญ่เกินกำลังของเครื่องเดียว (หรือ VM ตัวเดียว) จึงถูกบังคับให้ต้องต่อเครื่องเป็นคลัสเตอร์เพื่อรองรับผู้ใช้งาน กรณีแบบนี้จึงเหมาะกับการทำ orchrestration

2.1.5 Infrastructure as Code

โดยปกติแล้วถ้าเป็นเมื่อก่อน เวลาเราจะ Deploy ระบบอะไรก็ตาม ผู้ดูแลระบบก็อาจจะมีการไปติดตั้งแบบ Manual แล้ว Integrate ระบบเหล่านั้นเข้ากันเป็น Platform ก่อนจะส่งรายละเอียดต่างๆ ที่จำเป็นให้ทางผู้พัฒนาใช้ ซึ่งแน่นอนว่าขั้นตอนเหล่านี้กินเวลาค่อนข้างมาก และอาจเกิดความผิดพลาดได้เยอะในระหว่างการตั้งค่าต่างๆ ซึ่งรวมถึงการตั้งค่าให้ระบบใช้งานได้, การปรับแต่งประสิทธิภาพ, การเสริมความทนทาน และการเพิ่มความปลอดภัย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถัดจากนั้นมาเราก็เริ่มมีแนวคิดการทำ Template เพื่อให้ Deploy ระบบต่างๆ ได้โดยอาศัยเทคโนโลยี Virtualization เข้ามาช่วย ด้วยการสร้าง Virtual Machine ที่จำเป็นสำหรับระบบต่างๆ ขึ้นมาเป็นแม่แบบ ซึ่งถึงแม้จะช่วยลดเวลาในการติดตั้ง และปัญหาเรื่องการกำหนดค่าต่างๆ ผิดพลาดลงไปได้บ้าง แต่ก็ยังไม่ถือว่าตอบโจทย์เพียงพออยู่ดี เพราะถ้าหากต้องมีการอัพเกรด Component ใดๆ ของระบบ ก็ต้องมาเตรียม Template กันใหม่ ซึ่งเป็นขั้นตอนที่เสียเวลามาก

และเราก็มีวิธีใหม่ๆ กันอีกเต็มไปหมด แต่เราจะขอข้ามช็อตไปเลยนะครับเพราะโดยรวมๆ แล้ววิธีการส่วนใหญ่นั้นนอกจากจะช้า และไม่รองรับต่อการเปลี่ยนแปลงแล้ว ยังมีความผิดพลาดเกิดขึ้นในระหว่างการเตรียมระบบค่อนข้างมาก และนี่ก็เป็นจุดเริ่มต้นของแนวคิด Infrastructure as Code ครับ ซึ่งบางทีคนก็เรียกกันว่า Programmable Infrastructure เหมือนกัน

แนวคิดของการทำ Infrastructure as Code นั้น ก็คือการเปลี่ยนขั้นตอนการ Deploy ระบบทั้งหมดให้เป็นแบบ Automation โดยมี Code เป็นตัวกำหนดทุกสิ่งอย่างในการ Deploy นั่นเอง ทำให้งานการติดตั้ง Infrastructure และตั้งค่าทั้งหมดถูกแปลงให้อยู่ในรูปของ Script เพื่อทำคำสั่งต่างๆ ตามที่เรากำหนดเอาไว้แทน แต่แทนที่จะเป็นภาษา Script เหมือนการเขียนโปรแกรมกัน ตัวซอฟต์แวร์ที่ใช้ทำ IAC นั้นก็อาจให้เราใช้ภาษาแบบ High-level แทน

ตัวอย่างเช่นการติดตั้ง Linux Package ใดๆ ด้วยระบบ IAC นั้น ผู้ดูแลระบบก็สามารถทำการติดตั้ง, กำหนดค่าเบื้องต้น, สร้าง User, กำหนดสิทธิ์ และ Start Service เหล่านั้นขึ้นมาให้พร้อมใช้งานได้เลยจากการกำหนดค่าต่างๆ ใน IAC นั่นเอง

2.1.6 Pandas

Pandas เป็น Library ใน Python ที่ทำให้เราเล่นกับข้อมูลได้ง่ายขึ้น เหมาะมากสำหรับทำ Data Cleaning / Wrangling

วิธีการใช้งาน Pandas คือ โหลดไฟล์ข้อมูล เช่น CSV เข้าไป แล้วเราจะได้ข้อมูลในรูปแบบตาราง (DataFrame) ที่แบ่งข้อมูลตามแถวและคอลัมน์ หรือเหมือน Excel ที่เราใช้กัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	20140502T000000	313k	3	1.50	1340	7912	1.5	0	0	3	1340	0	1955	NaN
1	20140502T000000	2.384m	5	2.50	3650	9050	2.0	0	4	5	3370	280	1921	NaN
2	20140502T000000	342k	3	2.00	1930	11947	1.0	0	0	4	1930	0	1966	NaN
3	20140502T000000	420k	3	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	NaN
4	20140502T000000	550k	4	2.50	1940	10500	1.0	0	0	4	1140	800	1976	NaN

ภาพที่ 2.10 ตัวอย่าง DataFrame ของ Pandas

วิธีการโหลดไฟล์ CSV (Import)

จุดเริ่มต้นของการทำ Data Exploration & Analysis ใน Pandas ก็คือการโหลดไฟล์ข้อมูลแบบ CSV มาใช้งาน เราสามารถใช้คำสั่ง .head หรือ .tail เพื่อดูข้อมูลแถวบนสุด หรือแถวล่างสุดได้

```
# Read DF

csvdf = pd.read_csv('data.csv')

# Sometimes reading CSV for Excel need encoding

csvdf = pd.read_csv('data.csv',encoding = "ISO-8859-1")

# Print head and tail

csvdf.head()

csvdf.tail()
```

วิธีแปลงประเภทข้อมูล (Data Type) ใน Data Frame

บางครั้งประเภทข้อมูลของคอลัมน์เป็น String แต่เราต้องการ Integer หรือเราต้องการ Date เราสามารถแปลงข้อมูลได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

df['hour'] = pd.to_numeric(df['hour']) # แปลงเป็น Numeric

การกำหนดดัชนีใหม่

ปกติแล้วเดตาเฟรมประกอบไปด้วยข้อมูลหลายแถวหลายคอลัมน์ โดยจะมีคอลัมน์ที่อยู่ทางซ้ายสุดคือคอลัมน์ดัชนี

ดัชนีนั้นแท้จริงแล้วก็เป็นข้อมูลที่เรียงกันเป็นแถว ไม่ต่างอะไรจากคอลัมน์อื่น แต่จะต่างกันตรงที่ว่ามันมีหน้าที่พิเศษเพราะถูกใช้เพื่อเป็นตัวอ้างอิงหลักของตาราง

เราสามารถเปลี่ยนคอลัมน์หนึ่งในเดตาเฟรมให้เป็นคอลัมน์ดัชนีได้โดยใช้เมธอด set_index แล้วใส่ชื่อคอลัมน์ที่ต้องการใช้ลงไป

df = df.set_index('hour')

2.1.7 Crontab

Crontab คือคำสั่งที่จะทำงานตามเวลาที่กำหนด ซึ่งจะช่วยทำให้เราสะดวกขึ้นมากในการที่จะต้องสั่งงานบางอย่างซ้ำๆ กันหลายๆ ครั้ง อาจจะเป็นทุกๆ ชั่วโมง ทุกๆ วัน หรือ ทุกๆ เดือน เช่นการสั่งให้ Server ทำ Backup ทุกๆ สัปดาห์

คำสั่งและ option ของ crontab มีดังนี้

`crontab filename` การนำเอาคำสั่ง crontab เข้ามาจาก ไฟล์อื่น

`crontab -e` แก้ไข crontab ปัจจุบัน (ส่วนมากเราจะเพิ่มคำสั่งลงไปในนี้เลย)

`crontab -l` ดูคำสั่ง crontab ทั้งหมดที่มีอยู่

`crontab -r` ลบคำสั่ง crontab ที่มีทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

`crontab -u user` เป็นคำสั่งของผู้ดูแลระบบเท่านั้น เพื่อใช้ดู แก้ไข ลบ crontab ของ user แต่ละคน

เมื่อเรียกคำสั่งตามข้างบนแล้ว crontab จะเข้าสู่ระบบการ กำหนด หรือแก้ไข ซึ่งการกำหนด หรือแก้ไขนี้จะเหมือนกับการใช้งาน vi

รูปแบบของคำสั่ง crontab มีทั้งหมด 6 fields ดังนี้

1 = minute มีค่า 0 – 59 เวลาเป็นนาที จะสั่งให้คำสั่งที่กำหนดทำงานทันที่เมื่อถึงนาทีที่กำหนด

2 = hour มีค่า 0 – 23 เวลาเป็นชั่วโมง จะสั่งให้คำสั่งที่กำหนดทำงานทันที่เมื่อถึงชั่วโมงที่กำหนด

3 = day มีค่า 1 – 31 เวลาเป็นวัน จะสั่งให้คำสั่งที่กำหนดทำงานทันที่เมื่อถึงวันที่กำหนด

4 = month มีค่า 1 – 12 เวลาเป็นเดือน จะสั่งให้คำสั่งที่กำหนดทำงานทันที่เมื่อถึงเดือนที่กำหนด

5 = weekday มีค่า 0 – 6 วันขะงแต่ละสัปดาห์ มีค่าดังนี้ (อาทิตย์ = 0, จันทร์ = 1, อังคาร = 2, พุธ = 3, พฤหัส = 4, ศุกร์ = 5 ,เสาร์ = 6)

6 = command คำสั่ง เราสามารถกำหนดคำสั่งได้มากมาย รวมทั้ง script ต่าง ๆ ตามที่เราต้องการ

โดยสามารถจำเป็นรูปแบบง่ายได้ดังนี้

```
# ┌──────────── minute (0 - 59)
# │ ┌────────── hour (0 - 23)
# │ │ ┌──────── day of the month (1 - 31)
# │ │ │ ┌────── month (1 - 12)
# │ │ │ │ ┌──── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                7 is also Sunday on some systems)
# │ │ │ │ │
# * * * * * command to execute
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาพที่ 2.11 รูปแบบคำสั่ง cron

เริ่มเพิ่ม Cron Jobs โดยใช้คำสั่ง crontab -e

จากนั้นเพิมพ์คำสั่งดังต่อไปนี้ได้เลย 1 2 3 4 5 /path/to/script.sh

โดยปกติแล้วเมื่อ Cron Jobs ทำงานระบบจะส่ง E-Mail เพื่อแจ้งให้ Server Admin หรือ User ที่ตั้ง Cron Jobs ทราบว่า Cron Jobs ทำงานเรียบร้อยหรือติดปัญหาอย่างไร

หากต้องการให้ Cron Jobs ส่ง E-Mail ไปยังบัญชีที่เราตั้งค่าไว้เราสามารถเพิ่มคำสั่งได้ดังนี้

MAILTO="admin@domain.com"

0 3 * * * /root/backup.sh

หากเราไม่ต้องการให้ระบบส่ง E-Mail เราสามารถตั้งค่า Cron Jobs ได้โดยเพิ่ม /dev/null 2>&1 ตามหลัง command ที่จะสั่งให้ทำงาน ยกตัวอย่างเช่น 0 1 5 10 * /path/to/script.sh >/dev/null 2>&1

ทำการ save แล้วสั่ง /etc/init.d/crond restart

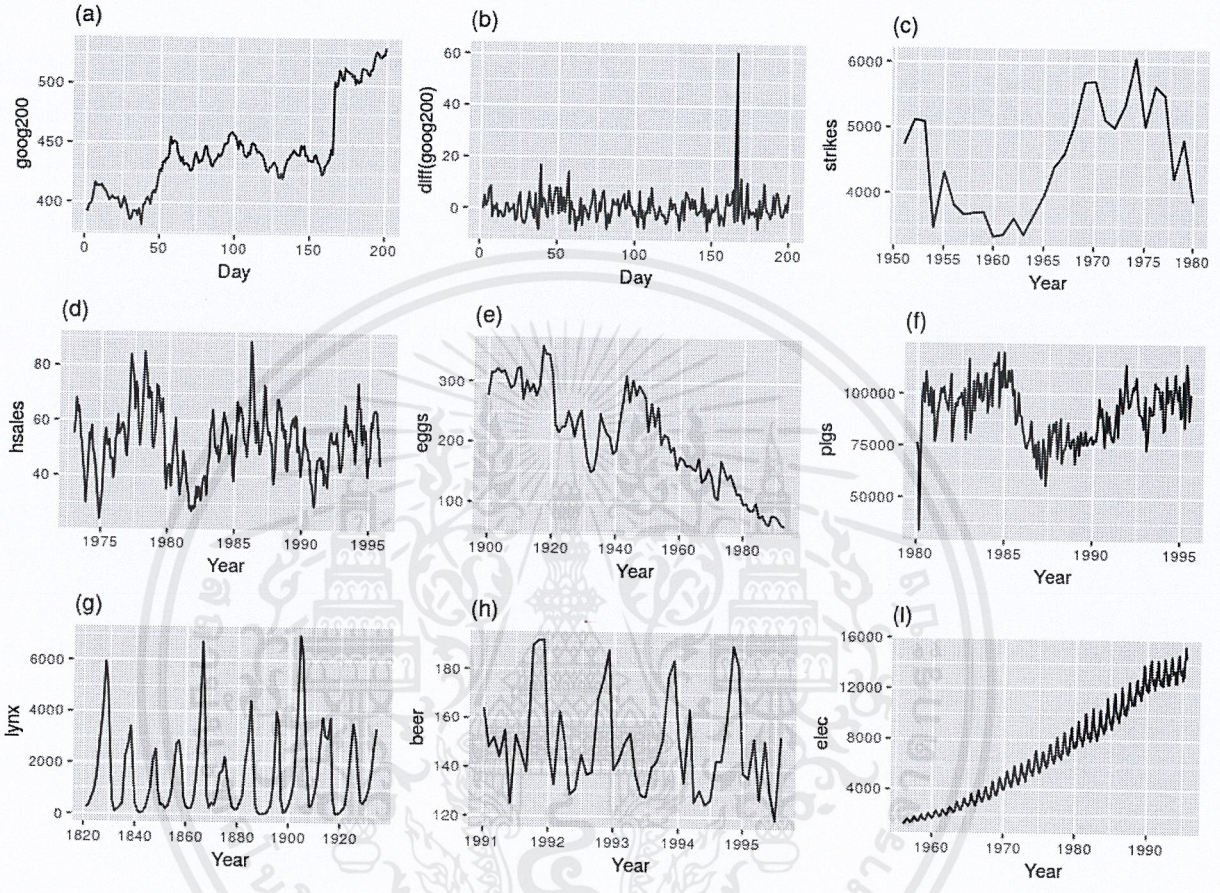
2.1.8 ARIMA (AutoRegressive Integrated Moving Average)

ARIMA (AutoRegressive Integrated Moving Average) ประกอบไปด้วย Combination ของ Time Series 3 เทคนิค คือ AR (Autoregressive), I (Integrated), MA (Moving average)

ก่อนจะเข้าใจโมเดล ARIMA เราต้องเข้าใจสิ่งที่เรียกว่า Non-Stationary Data VS Stationary Data ก่อน โดยข้อมูลที่ Stationary คือ “ถ้าเราสุ่มข้อมูลออกมา 2 ครั้งจากบน Time Series จะต้องได้กราฟหน้าตาเหมือนกัน” เพราะฉะนั้นข้อมูลที่มี Trend หรือ Seasonality ถ้าเราสุ่มข้อมูลออกมา 2 ครั้งจะได้กราฟไม่เหมือนกันแน่นอน ถือเป็น Non-Stationary Data ซึ่ง ARIMA จะทำงานได้ไม่ดีกับข้อมูลประเภทนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การตรวจสอบความเป็น Stationary



ภาพที่ 2.12 ตรวจสอบความเป็น Stationary ของข้อมูล

รูป (d), (h) และ (I) มีความเป็น Seasonality อย่างชัดเจน รูป (a), (c), (e), (f) และ (i) มีการเปลี่ยนแปลงแนวโน้มของข้อมูล จึงเหลือเพียงรูป (b) และ (g) ที่เป็น Stationary

ในช่วงแรกรูป (g) อาจดูเหมือนเป็น Non-stationary แต่รอบของกราฟมีความสม่ำเสมอ เนื่องจากเมื่อจำนวนประชากรของแมวป่าสูงขึ้นเรื่อยๆ จนอาหารของมันมีจำนวนไม่เพียงพอ แมวป่าก็จะเลิกขยายพันธุ์ ทำให้จำนวนประชากรลดลง และเมื่อมีอาหารเยอะขึ้นมันก็จะเริ่มขยายพันธุ์อีกครั้ง เป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เช่นนี้ไปเรื่อยๆ เราจึงไม่สามารถทำนายรอบการเพิ่มลดของจำนวนประชากรได้ในระยะยาว ดังนั้นกราฟนี้จึงเป็น Stationary

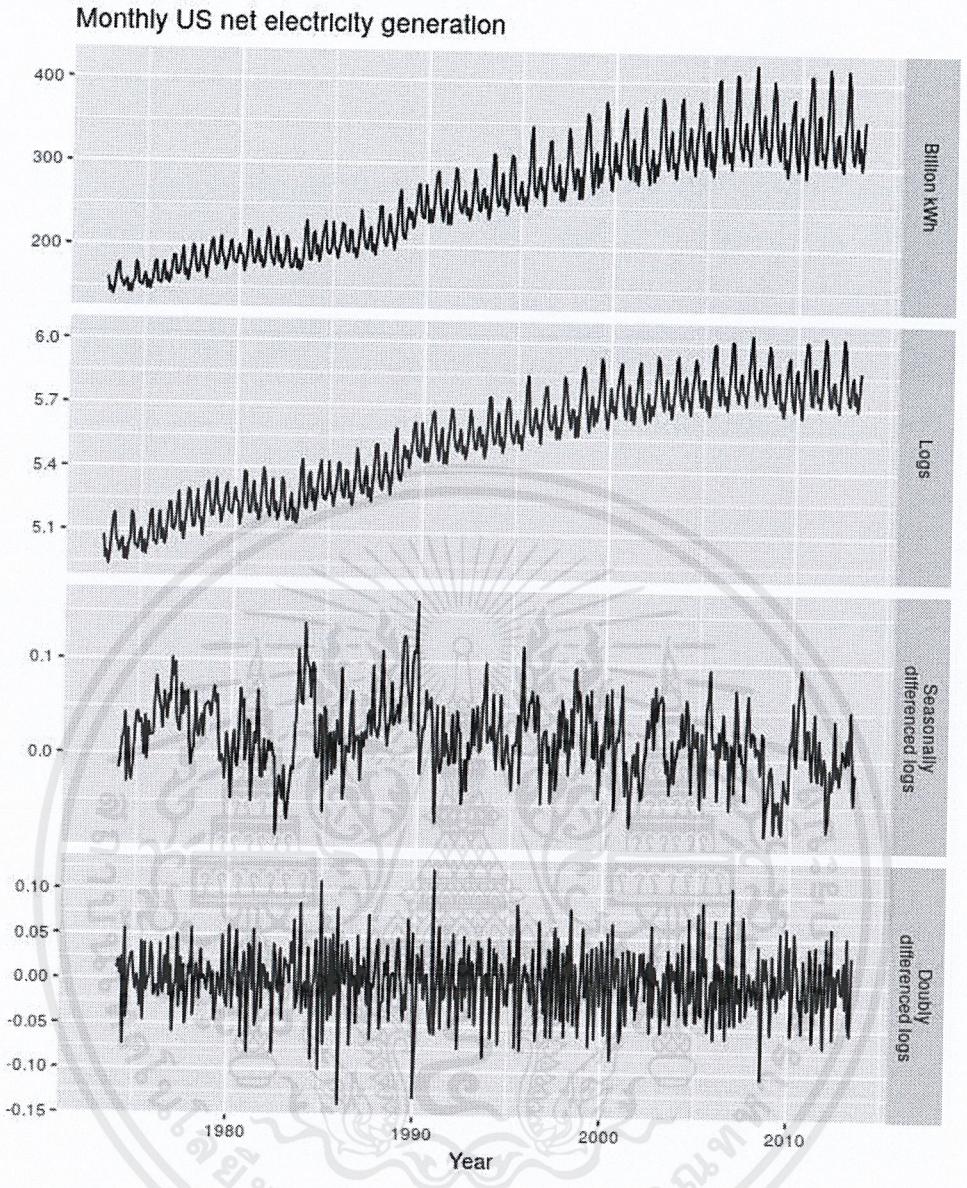
องค์ประกอบของ ARIMA(p,d,q) ประกอบด้วยพารามิเตอร์ 3 ตัว จาก 3 วิธีการที่นำมารวมเป็น โมเดล ARIMA โดยที่ p เป็นพารามิเตอร์ของ AR, d เป็นพารามิเตอร์ของ I, q เป็นพารามิเตอร์ของ MA โดยจะเริ่มอธิบายจาก I ก่อน

Integrated (I)

พารามิเตอร์ d ของวิธีการ Integrated ตัวนี้นำมาหาความเป็น Stationary ของ Time Series ซึ่งต้องคงสถานะนี้เมื่อเวลาผ่านไป เป็นสิ่งที่ สำคัญมากในโมเดล ARIMA หรือก็คือ ข้อมูลอย่างค่าเฉลี่ย (Mean), ความแปรปรวน (Variance) ต้องมีค่าคงที่เมื่อเวลาผ่านไป กล่าวคือ ถ้าข้อมูลมีแนวโน้ม (Trend) หรือ มี seasonal มันก็จะไม่ใช่ Stationary แล้ว เพราะข้อมูลจำพวก Mean และ Variance จะไม่คงที่ เราจึงต้องทำให้ข้อมูลเป็น Stationary ก่อนจึงจะสร้างโมเดลจากข้อมูลนั้นๆ ได้ การทำให้ข้อมูลเป็น Stationary โดยปกติแล้วสามารถทำได้โดยการ Differencing ข้อมูล จากนั้นก็นำมาเข้าสมการทางคณิตศาสตร์ Augmented Dickey–Fuller test (ADF) เพื่อเช็คว่าข้อมูลเป็น Stationary หรือยัง

ตัวอย่างการ Differencing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.13 ข้อมูลการใช้พลังงานของ US ที่ผ่านการ Transforming และ Differencing

รูปที่ 1 : เป็นข้อมูลการใช้พลังงานของอเมริกา จะเห็นได้ชัดเจนว่าข้อมูลดิบยังมี Trend ชัดเจนมากก็คือ เห็นได้ชัดว่า ไม่เป็น Stationary

รูปที่ 2 : ข้อมูล Log Scale ซึ่งสร้างมาจาก การนำข้อมูลดิบมา take Logarithm เข้าไป จะเห็นได้ว่า ก็ยังมี T r e n d ชัดเจนอยู่ ก็คือ ยังไม่เป็น S t a t i o n a r y

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3 : รูปนี้มีการเริ่มทำ Differencing รอบแรก จะเห็นว่าข้อมูลเริ่มเป็น Stationary มาขึ้นแล้ว (Trend เริ่มหายไป) แล้วแต่ยังมี Seasonality อยู่

รูปที่ 4 : นำข้อมูล Seasonality จากรูปที่ 3 มา Differencing อีกรอบ และ ในขั้นตอนนี้เองข้อมูลของเราก็ได้กลาย เป็น Stationary โดยสมบูรณ์

Auto Regressive (AR)

Auto Regressive ตามความหมายก็คือ การใช้ค่าของตัวมันเองในการทำนายวันถัดไป โดยที่พารามิเตอร์ p ของ AR ก็คือ ลำดับของคาบเวลาในอดีตที่เหมาะสม เพื่ออธิบายค่าในเวลาปัจจุบัน โดยการคำนวนจะมีการถ่วงน้ำหนักด้วยค่าของ time series ก่อนหน้า p อีกทีหนึ่ง สรุปคือ สมการนี้ตั้งอยู่บนสมมุติฐานที่ว่า ค่าของวันที่เราจะทำนายมีความสัมพันธ์กับค่าในอดีตของตัวมันเอง เช่น ราคาหุ้นวันนี้จะขึ้นกับราคาหุ้นของ x วันก่อน เป็นต้น

สามารถเขียนสมการโมเดล AR ให้อยู่ในลำดับของ p ได้ดังนี้

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t$$

โดยที่ c คือ ค่าคงที่

ϕ คือ ค่าสัมประสิทธิ

t คือ เวลา

ε_t คือ ค่า error ของโมเดล

Moving Average (MA)

คอนเซปของ MA นั้นจะค่อนข้างคล้ายกับคอนเซปของ AR แต่จะตั้งบนสมมุติฐานของ Error แทน โดยสามารถเขียนสมการโมเดลของ MA ให้อยู่ในลำดับของ q ได้ดังนี้

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

โดยที่ c คือ ค่าคงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

θ คือ ค่าสัมประสิทธิ

t คือ เวลา

ε_t คือ ค่า error ของโมเดล

เมื่อนำสมการมารวมกัน เราจะเขียนโมเดลที่สมบูรณ์ได้ดังนี้

$$y'_t = c + \varphi_1 y'_{t-1} + \cdots + \varphi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

โดยที่ y' คือ โมเดลที่มีการ Differencing แล้ว

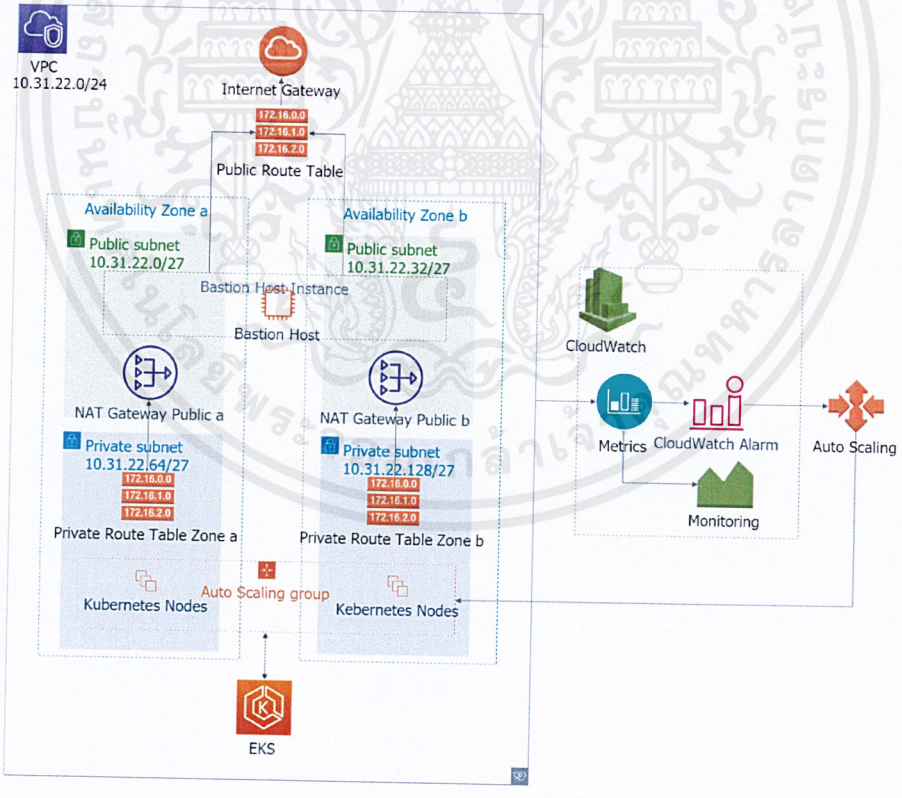
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

วิธีดำเนินการวิจัย

รายงานสหกิจฉบับนี้เป็นการพัฒนาอัลกอริทึมที่ใช้สำหรับการทำ Kubernetes Autoscaling สำหรับโหนดและพอด ในกรณีที่มีการใช้งานคลัสเตอร์มากกว่าปกติ รวมถึงการทำ Node forecasting ที่ใช้ทำนายการใช้งานคลัสเตอร์ล่วงหน้าและสเกลโหนดล่วงหน้าถ้าหากจะมีการใช้งานมากกว่าที่กำหนดไว้ ซึ่งในบทนี้จะกล่าวถึงการออกแบบโครงสร้างและขั้นตอนการดำเนินงานของระบบ โดยมีรายละเอียดดังนี้

3.1 สถาปัตยกรรมของระบบ



ภาพที่ 3.1 สถาปัตยกรรมของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

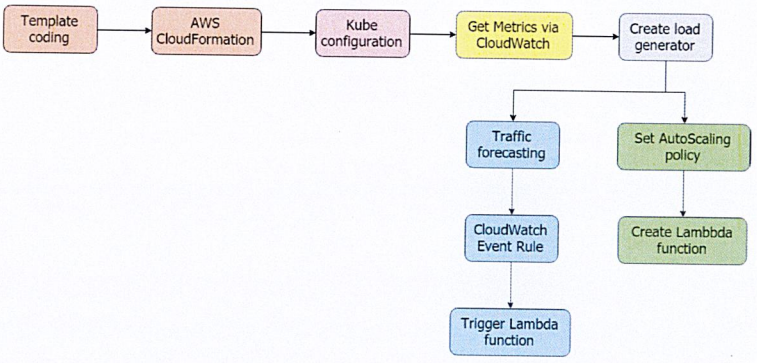
ระบบ Kubernetes ที่ใช้งานนั้นตั้งอยู่ใน Amazon EKS ซึ่งเป็น Kubernetes ที่อเมซอนดูแลในส่วนของ Control plane ส่วนที่เราต้องทำคือการจัดการ Worker node ที่เป็น EC2 Instance โดยเครื่องโหนดนั้นถูกวางไว้อยู่บน 2 Availability zone โดยในแต่ละโซนจะมี Public subnet และ Private subnet ใน Private subnet เราจะวางโหนดและตั้ง Route table โดยกำหนด Default route ไปยัง NAT gateway ที่อยู่ใน Public subnet และกำหนด Default route ใน Public subnet ให้ออกไปยัง Internet gateway และวาง Bastion host ซึ่งเป็นเครื่อง Instance ที่เอาไว้เข้าไปจัดการโหนด

ในส่วนของการ Monitoring ผู้จัดทำได้ใช้เซอร์วิสที่ชื่อว่า Amazon CloudWatch ในการมอนิเตอร์เมตริกที่มาจาก Kubernetes และจะนำเมตริกที่ได้มาใช้ในการทำ Auto scaling ในระดับโหนดและพอด

3.2 การทำงานของระบบ

เริ่มต้นเราจะสร้างระบบผ่าน AWS CloudFormation ซึ่งเป็นเซอร์วิสหนึ่งของอเมซอนที่ช่วยในการสร้าง Infrastructure จากนั้นก็ Config ให้ Kubernetes กับเครื่องโหนดที่สร้างมารู้จักกัน ทดลองนำแอปพลิเคชั่นมาลงในคลัสเตอร์ และสร้าง Load generator ยิงไปที่แอปพลิเคชั่นเพื่อดูการใช้งาน ต่อมาจะทำการรับเมตริกจาก CloudWatch เพื่อมาคำนวณหา Traffic ล่วงหน้า สร้างฟังก์ชั่นสำหรับสเกลเครื่องโหนดผ่าน AWS Lambda และใช้ CloudWatch สร้าง Event เพื่อมา Trigger ฟังก์ชั่นที่สร้างไว้ และทำ Auto scaling สำหรับพอดและโหนดในกรณีที่มีการใช้งานสูงในเวลาสั้นๆ โดยจะอธิบายแต่ละขั้นตอนโดยละเอียดต่อไป

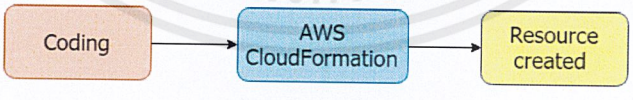
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.2 ขั้นตอนการทำงานของระบบโดยรวม

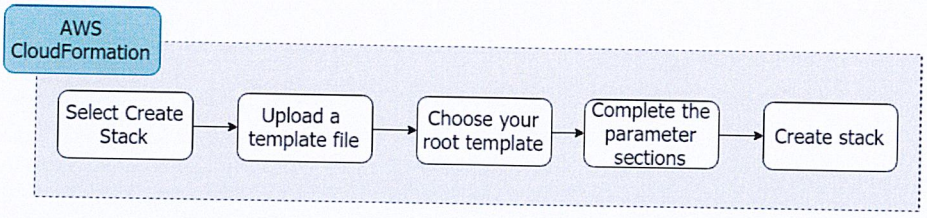
3.2.1 การจัดเตรียมระบบผ่าน AWS CloudFormation

ในการจัดเตรียมระบบนั้นสามารถทำได้ 3 วิธีคือ 1. การสร้างจาก AWS Console เป็นวิธีที่แนะนำสำหรับผู้เริ่มต้นใช้งานสำหรับการสร้างระบบเล็กๆ เพราะสามารถเข้าใจได้ง่าย แต่ถ้าเป็นการสร้างระบบที่มีองค์ประกอบมากนั้น การใช้วิธีนี้จะทำให้เสียเวลาและอาจมีการผิดพลาดเกิดขึ้นได้ในการสร้างแต่ละครั้ง 2. การสร้างผ่าน AWS Command Line Interface (CLI) เป็นการสร้างโมเดลผ่านการใช้คำสั่ง Command line วิธีนี้จะทำให้ระยะเวลาที่ใช้ในการสร้างระบบลดลง แต่ก็อาจจะยังมีข้อผิดพลาดเกิดขึ้นได้อยู่ในการใช้คำสั่งแต่ละครั้ง 3. การสร้างผ่าน AWS CloudFormation ซึ่งเป็น service ที่ช่วยให้สามารถสร้างโมเดลสถาปัตยกรรมและทรัพยากรของระบบโดยผ่านการเขียนให้อยู่ในรูปแบบ template ของ JSON หรือ YAML เป็นวิธีที่นิยมใช้มากที่สุดในการสร้าง Network infrastructure เพราะเทมเพลตที่เขียนไว้แล้วนั้นสามารถนำมาใช้ใหม่ได้และหากต้องการปรับเปลี่ยนอะไรก็แก้ไขผ่านโค้ดได้เลย ทำให้ลดระยะเวลาการสร้างและลดความผิดพลาดที่อาจเกิดขึ้นได้จากการสร้างด้วยมืออีกด้วย



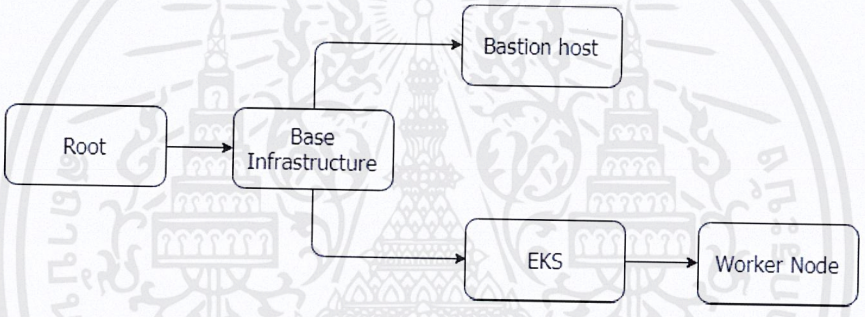
ภาพที่ 3.3 การทำงานของ AWS CloudFormation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.4 การสร้าง Stack ใน AWS CloudFormation

โดยทางผู้จัดทำเลือกใช้วิธีการสร้างระบบผ่าน AWS CloudFormation และเขียนโค้ดในรูปแบบของ YAML และได้เขียนเทมเพลตที่ใช้สำหรับสร้างระบบไว้ทั้งหมด 5 เทมเพลต ดังภาพที่ 3.5 และค่าพารามิเตอร์ดังตารางที่ 3.1



ภาพที่ 3.5 ลำดับการสร้างเทมเพลตใน CloudFormation

Root เป็นเทมเพลตที่เชื่อมต่อไปยังเทมเพลตอื่นๆ เมื่อเราสร้าง Root ตัว Root จะไปเรียกเทมเพลตอื่นขึ้นมาสร้างใน CloudFormation ให้เองโดยอัตโนมัติ โดยที่เราสามารถตั้งค่าได้ว่าจะให้เทมเพลตไหนสร้างก่อนหรือหลัง ในที่นี้จะให้สร้าง Base Infrastructure ขึ้นมาก่อนเพราะเป็นการสร้างทรัพยากรที่จำเป็นต้องใช้ในระบบ ต่อมาจะให้สร้าง Bastion host ที่เอาไว้ Access เข้า Kubernetes และ EKS ที่เป็นเซอร์วิสที่เอาไว้ Control Kubernetes ขึ้นพร้อมกัน และสร้าง Worker node ต่อจาก EKS จึงเสร็จเรียบร้อย

Base Infrastructure เป็นเทมเพลตที่ใช้สร้างทรัพยากรที่ต้องใช้ เช่น VPC, Public subnets, Private subnets, NAT gateway, IAM Role, Security Group, Route table เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Bastion host เป็นการสร้าง EC2 Instance ที่เอาไว้เข้าไปจัดการ Kubernetes คลัสเตอร์

EKS เป็นเทมเพลตที่ใช้สร้างเซอร์วิส Amazon EKS ซึ่งเป็น Kubernetes ที่ทาง Amazon จัดการในส่วนของ Control plane ให้

Worker node เป็นการสร้าง EC2 Instance มาเชื่อมต่อกับคลัสเตอร์ EKS ที่เราสร้างไว้ก่อนหน้านี้ โดยให้ทำหน้าที่เป็นโหนดของคลัสเตอร์

ตารางที่ 3.1 ค่าพารามิเตอร์ในการสร้างระบบ

ชื่อพารามิเตอร์	ค่าพารามิเตอร์	หน้าที่
Stack name	K8S-Autoscaling	ชื่อ Stack ของ CloudFormation
Project Name	K8S-Autoscaling	ชื่อโปรเจกต์
Vpc Block	10.31.22.0/24	ขนาดเน็ตเวิร์กของ VPC
Public Subnet 01 Block	10.31.22.0/27	Public subnet ของ Avalability Zone a
Public Subnet 02 Block	10.31.22.32/27	Public subnet ของ Avalability Zone b
Private Subnet 01 Block	10.31.22.64/26	Private subnet ของ Avalability Zone a
Private Subnet 02อBlock	10.31.22.128/26	Private subnet ของ Avalability Zone b
Bastion Host SSH Key Parameter	***	Key ที่ใช้สำหรับ Authentication เข้า Bastion host (ต้องสร้างไว้ก่อนจะสร้าง stack)
AMI Parameter	ami-01f7527546b557442	Amazon Machine Images (AMI) เลือก OS ที่จะใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Instance Type Parameter	t2.micro	สเป็กของเครื่อง Bastion host
EKS Cluster Name	K8S-Autoscaling	ชื่อคลัสเตอร์ EKS
Node Group Name	K8S-Autoscaling-Node	ชื่อ Node group
Node Key Name	***	Key ที่ใช้สำหรับ Authentication เข้าไปจัดการ Node
Node Image Id	ami-0361e14efd56a71c7	Amazon EKS-optimized AMI สำหรับโหนด
Node Instance Type	t2.small	สเป็กของเครื่องโหนด
Node Auto Scaling Group Min Size	2	จำนวนโหนดต่ำสุดที่สามารถมีได้
Node Auto Scaling Group Max Size	4	จำนวนโหนดสูงสุดที่สามารถมีได้
Node Auto Scaling Group Desired Capacity	2	จำนวนโหนดที่ต้องการในปัจจุบัน
Node Volume Size	60	จำนวน Volume ของโหนด

3.2.2 Kubernetes Configuration

เมื่อเราทำการสร้างคลัสเตอร์และโหนดเสร็จเรียบร้อยแล้ว ต่อมาจะต้องทำการติดตั้งเซอร์วิส Kebernetes เข้าไปที่คลัสเตอร์ผ่านการสร้างไฟล์ kubeconfig และเชื่อมโหนดเข้ากับคลัสเตอร์

สร้างไฟล์ kubeconfig ผ่าน Cli โดยใช้

aws eks --region **ap-southeast-1** update-kubeconfig --name **K8S-Autoscaling**

สามารถตรวจสอบว่าสร้างเซอร์วิสสำเร็จหรือไม่ด้วยคำสั่งด้านล่าง จะได้ผลลัพท์ตามภาพที่ 3.6

kubectl get service หรือ kubectl get svc

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
[ec2-user@ip-10-31-22-14 ~]$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   172.20.0.1   <none>        443/TCP   20d
[ec2-user@ip-10-31-22-14 ~]$
```

ภาพที่ 3.6 ตรวจสอบเซอร์วิส Kubernetes

จากนั้นเชื่อมต่อ Cluster กับ Node ด้วย AWS authenticator configuration map โดยสร้างไฟล์ YAML และใส่ค่า rolearn ของเครื่องโหนดที่สร้างไปในไฟล์ ConfigMap

จากนั้น apply ไฟล์ลงบน Cluster และทดสอบด้วยการใช้คำสั่ง kubectl get nodes

```
7-eks-c57ff8
[ec2-user@ip-10-31-22-14 ~]$ kubectl get nodes
NAME                                              STATUS   ROLES    AGE   VERSION
ip-10-31-22-157.ap-southeast-1.compute.internal   Ready    <none>   20d   v1.13.7-eks-c57ff8
ip-10-31-22-93.ap-southeast-1.compute.internal    Ready    <none>   20d   v1.13.7-eks-c57ff8
[ec2-user@ip-10-31-22-14 ~]$
```

ภาพที่ 3.7 การตรวจสอบ Node

และทดลองรันแอปพลิเคชั่นบน Kubernetes โดยแอปพลิเคชั่นที่นำมารันคือ 2048-game

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 การเก็บเมตริกของคลัสเตอร์จาก Amazon CloudWatch

Amazon CloudWatch Log Insight มีหน้าที่เก็บข้อมูลการใช้งานคลัสเตอร์ของเราทั้งหมด เช่น Cluster CPU Utilization, Cluster Memory Utilization, Node CPU Utilization, Node Memory Utilization, จำนวนพอดในแต่ละโหนด, จำนวนโหนดทั้งหมด, จำนวนเซอร์วิสในแต่ละพอด เป็นต้น

เมื่อสามารถวัดเมตริกได้แล้วจึงใช้ AWS SDK for Python เขียนฟังก์ชั่นเพื่อเรียกดูค่าเมตริกที่ต้องการโดยกำหนดชื่อเมตริกและระยะเวลาที่ต้องการเก็บค่า โดยผลลัพท์ที่ออกมาจะอยู่ในรูป Timestamp คือเวลา คู่กับ Value (Float) คือค่าของเมตริก เราต้องทำการแปลงตัวแปรชนิด Timestamp เป็น String ก่อน แล้วจึงบันทึกค่าเมตริกที่เก็บมาได้ลงไฟล์ csv เพื่อนำไปคำนวณต่อไป

ภาพที่ 3.8 การเก็บค่า Metric ผ่าน CloudWatch Log Insight

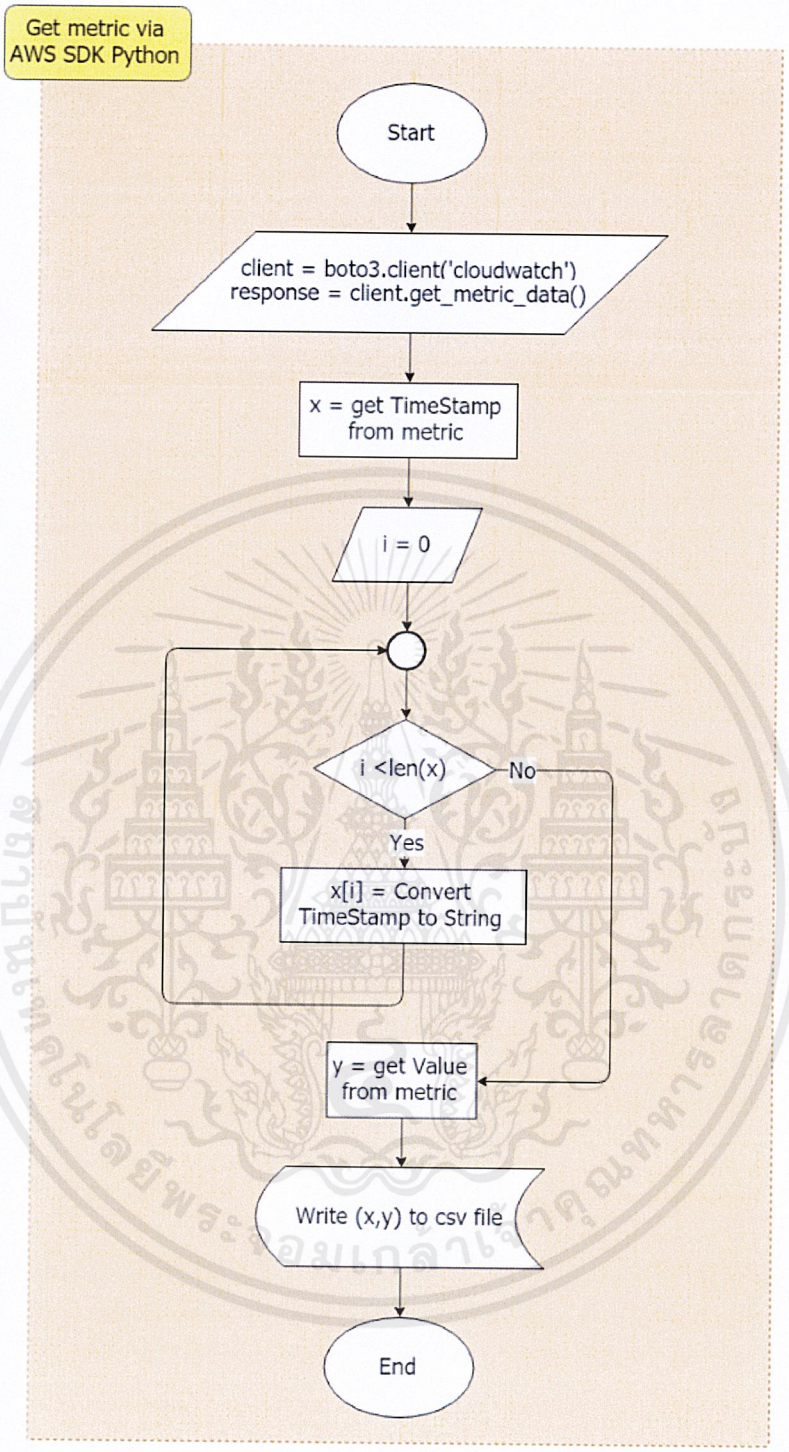
ตารางที่ 3.2 ค่าพารามิเตอร์ที่ใส่เมื่อต้องการเก็บเมตริก

ชื่อพารามิเตอร์	ค่าพารามิเตอร์	หน้าที่
Namespace	ContainerInsights	ระบุว่าต้องการเก็บเมตริกจากเซอร์วิสไหน ซึ่งคลัสเตอร์จะเก็บผ่าน ContainerInsights
MetricName	node_cpu_utilization	ชื่อเมตริกที่ต้องการเก็บค่า
Dimensions: Name	ClusterName	ระบุว่าเก็บค่าจาก EKS Cluster
Dimensions: Value	K8S-Autoscaling	ชื่อคลัสเตอร์ที่ต้องการเก็บค่า
Period	300 (seconds)	ความถี่ของช่วงเวลาที่ต้องการเก็บค่าเป็นวินาที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Stat	Average	ค่าเชิงสถิติของข้อมูล โดยให้เก็บเป็นค่าเฉลี่ยจากช่วงเวลาที่กำหนด
Unit	Percent	หน่วยที่เก็บเป็นเปอร์เซ็นต์
StartTime	datetime.datetime.utcnow() - datetime.timedelta(hours = 48)	เวลาที่เริ่มเก็บค่า (ในที่นี้ให้นับตั้งแต่ปัจจุบันย้อนไป 2 วัน) เป็นเวลา UTC
EndTime	datetime.datetime.utcnow()	เวลาสิ้นสุด กำหนดให้เป็นเวลาปัจจุบัน UTC
ScanBy	TimestampAscending	การเรียงข้อมูลที่ได้มา โดยให้เรียงตามเวลาจากเก่าสุดมาใหม่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



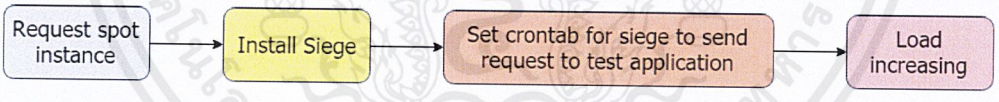
ภาพที่ 3.9 วิธีการเก็บค่า Metric ลงไฟล์ csv

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.4 การสร้าง Load Generator ด้วย Siege ผ่าน Spot Instance

ทดลองสร้าง Load Generator เพื่อระดมยิงรีเควสมายังแอปพลิเคชั่น 2048-game ที่รันอยู่บนโหนด โดยโปรแกรมที่ใช้คือ Siege ซึ่งเป็นโปรแกรมที่ใช้สร้าง Load Test ที่รันอยู่บน Linux โดยเมื่อทดลองยิงจาก Instance 1 เครื่อง จะทำให้ CPU Utilization ของโหนดเพิ่มขึ้นมาประมาณ 5% ดังนั้นหากต้องการให้มีการใช้งานในปริมาณมาก จึงต้องสร้าง Instance ขึ้นมาหลายเครื่อง ซึ่งการใช้ On-Demand หรือ EC2 Instance แบบปกติก็สามารถทำได้แต่ก็อาจจะมีค่าใช้จ่ายสูง ผู้จัดทำจึงเลือกใช้ EC2 Spot Instance เป็น EC2 ที่อาจถูกอะเมซอนเรียกคืนได้หลังจากสร้างมา แล้วค่อยสร้างใหม่ให้ แต่ก็มีราคาถูกกว่าา On-demand ถึง 70% และยังสามารถสร้างได้ทีละหลายเครื่องพร้อมกัน และสามารถกำหนดเวลาที่จะให้เครื่อง Terminate ตัวเองได้ เพราะการสร้าง Load Generator นั้นต้องการทำเป็นครั้งคราวจึงเหมาะที่จะใช้ Spot instance มากกว่า

เมื่อทำการสร้าง Spot Instance แล้ว จากนั้นจึงติดตั้ง Siege ลงบน Instance ที่สร้างมา หลังจากติดตั้งเสร็จจึงใช้คำสั่ง crontab สั่งให้ siege ยิงไปที่ URL ของ 2048 ที่อยู่บนโหนดของเราตามระยะเวลาที่กำหนดไว้ โดยขั้นตอนนี้ทางผู้จัดทำได้เขียนเป็น Bash script เก็บไว้แล้วใส่สคริปต์ไปพร้อมกับตอนที่ Request spot instance เลย ทำให้เมื่อสร้างเครื่องเสร็จ Siege ก็จะทำงานตามเวลาที่กำหนดทันที



ภาพที่ 3.10 การสร้าง Load Generator

3.2.4 สร้าง Scaling policy บน Autoscaling group ของ Node

การที่จะให้เครื่องโหนดเพิ่มขึ้นเองได้นั้น เราจะต้องกำหนด Policy ให้ Autoscaling group ให้ทำการเพิ่มหรือลดจำนวนโหนด เมื่อถึงเงื่อนไขที่กำหนดหรือมีการ Execute policy โดยสามารถสร้าง Scaling policy ผ่านการเขียนเทมเพลตบน CloudFormation โดยใส่ค่า Properties ดังตารางที่ 3.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.3 ค่า Properties ของ Scaling policy (scale out)

ชื่อพารามิเตอร์	ค่าพารามิเตอร์	หน้าที่
AutoScalingGroupName	K8S-Autoscaling-Node	ชื่อ Autoscaling Group ของ Node
AdjustmentType	ChangeInCapacity	เป็นการบอกว่าจะทำอย่างไรเมื่อมีการ Execute policy โดยในนี้ให้เพิ่มหรือลดจำนวน instance ตามที่ระบุในข้อถัดไป
ScalingAdjustment	1	ระบุจำนวนเครื่องที่ต้องการให้มีการเปลี่ยนแปลง โดยกำหนดให้เพิ่ม 1 เครื่อง
Cooldown	300 (seconds)	ระยะเวลาที่ต้องรอจนกว่า Policy ต่อไปจะสามารถ Execute ได้

จากนั้นจึงสร้าง CloudWatch Alarm เพื่อเป็นตัว Trigger policy ผ่าน CloudFormation เทมเพลต โดยใส่ค่า Properties เพื่อสั่งให้เพิ่มเครื่อง instance เมื่อ CPU Utilization มากกว่าหรือเท่ากับค่า Threshold ที่ระบุไว้ ดังตารางที่ 3.4

ตารางที่ 3.4 ค่า Properties ของ Alarm

ชื่อพารามิเตอร์	ค่าพารามิเตอร์	หน้าที่
AlarmName	K8S-Autoscaling-ScaleOut-Alarm	ชื่อของ Alarm ที่เราจะสร้าง
Namespace	ContainerInsights	Namespace ของเมตริกที่นำมาใช้กับ Alam
MetricName	node_cpu_utilization	ชื่อเมตริกที่อยู่ใน Namespace ที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Statistic	Average	ค่าทางสถิติของข้อมูลที่ต้องการ ในที่นี้ให้เป็นค่าเฉลี่ย
ComparisonOperator	GreaterThanOrEqualToThreshold	การเปรียบเทียบเชิงคณิตศาสตร์ ระหว่างค่าจากเมตริกกับ Threshold ในที่นี้ให้มากกว่าหรือเท่ากับ Threshold ก็จะ Alarm
Threshold	50	ค่าที่ต้องการนำมาเปรียบเทียบ
EvaluationPeriods	1	จำนวนช่วงระยะเวลาที่จะนำข้อมูลมาเปรียบเทียบแต่ละครั้ง
Period	300 (seconds)	ช่วงระยะเวลาเก็บข้อมูลในแต่ละรอบ
AlarmActions	ScaleOutPolicy (ARN)	Policy ที่ต้องการจะ Execute เมื่อเกิด Alarm
Dimensions: Name	ClusterName	กำหนดให้ Alarm ทำงานที่ Policy ของ Cluster
Dimensions: Value	K8S-Autoscaling	ระบุชื่อ Cluster

ในส่วนของการลดเครื่อง (Scale in) นั้นให้ทำเหมือนเดิม แต่ให้ทำการแก้ไขค่า ScalingAdjustment ของ Scaling Policy จาก 1 เป็น -1 และแก้ไข Properties ของ Alarm ดังนี้

ค่าComparisonOperator จากเดิมคือ GreaterThanOrEqualToThreshold แก้ไขเป็น LessThanOrEqualToThreshold

ค่า Threshold: จากเดิมคือ 50 แก้ไขเป็น 30

ค่า AlarmActions: จาก ScaleOutPolicy (ARN) แก้ไขเป็น ScaleInPolicy (ARN)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.5 สร้างฟังก์ชั่นสั่งให้ไป Execute policy

เขียนฟังก์ชั่นที่รันบน AWS Lambda ให้รับค่า Autoscaling Group Name และ Autoscaling Group Policy และไป Execute policy ดังกล่าวทันทีเมื่อมีการเรียกใช้ฟังก์ชั่นนี้ ในกรณีที่เราต้องการเพิ่ม Instance โดยไม่ต้องการรอ Alarm ซึ่งกรณีนี้จะใช้กับการเพิ่มจำนวนเครื่องโหนดล่วงหน้าตามเวลาที่กำหนดจากการทำนาย Traffic ในขั้นตอนต่อไป

3.2.6 การทำนาย Traffic

ในการทำนาย Traffic ที่จะเกิดขึ้นในอนาคต ผู้จัดทำได้เลือกใช้ไลบรารี่ Pandas ของ Python เป็นเครื่องมือในการวิเคราะห์ จัดเตรียมข้อมูล และทำนายด้วยโมเดล ARIMA (AutoRegressive Integrated Moving Average)

สาเหตุที่เลือกใช้โมเดลนี้เนื่องจาก โมเดลที่ได้ศึกษามานั้นมีอยู่สองแบบคือ ETS (Exponential Smoothing) และ ARIMA โดยโมเดล ETS นั้นเหมาะกับข้อมูลที่เป็น Non-stationary นั่นคือข้อมูลไม่มั่นคง ไม่มีความเป็นแนวโน้ม

โมเดล ARIMA เป็นโมเดลที่ใช้กันอย่างแพร่หลายในการทำนายข้อมูลที่เป็น Time series และใช้ได้ดีกับข้อมูลที่เป็น Stationary ก็คือมีความมั่นคง ซึ่งข้อมูลของ Traffic ของเราก็ค่อนข้างที่จะเป็นเช่นนั้น แต่ขณะเดียวกันก็สามารถใช้กับข้อมูลที่เป็น Non-stationary ได้ด้วย ซึ่งจะอธิบายการใช้งานในลำดับถัดไป

เริ่มต้นเราจะอ่านข้อมูลจากไฟล์ csv ออกมา แล้วแปลง String เวลาให้อยู่ในฟอร์แมท datetime ของ pandas แล้วนำตัว datetime นั้นมาตั้งเป็น index แล้วสั่งแสดงผลออกมาดังภาพที่ 3.11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'2019-11-22 02:50:00', '2019-11-22 02:55:00',
'2019-11-22 03:00:00', '2019-11-22 03:05:00',
'2019-11-22 03:10:00', '2019-11-22 03:15:00',
'2019-11-22 03:20:00', '2019-11-22 03:25:00',
'2019-11-22 03:30:00', '2019-11-22 03:35:00'],
```

ภาพที่ 3.11 Datetime index

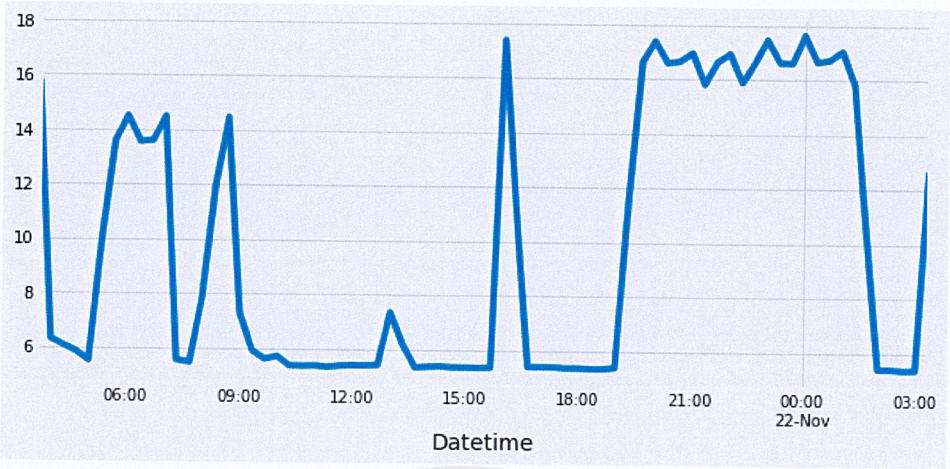
จากนั้นทำการ Resample ให้ข้อมูลอยู่ในช่วงเวลาที่เราต้องการและเหมาะสมกับการนำไปทำนายต่อไป โดยจากข้อมูลเริ่มต้นเราเก็บค่าเฉลี่ยการใช้ CPU ทุก 10 นาที เราจะเอามาทำการ Resample ให้ได้ค่าเฉลี่ยของการใช้งานทุก 20 นาที จะได้ตัวอย่างข้อมูลตามภาพที่ 3.12

```
                        ...
2019-11-22 02:00:00    5.384781
2019-11-22 02:20:00    5.385811
2019-11-22 02:40:00    5.346506
2019-11-22 03:00:00    5.360441
2019-11-22 03:20:00   12.674476
```

ภาพที่ 3.12 ตัวอย่างข้อมูลหลังจาก Resample

เมื่อนำข้อมูลใหม่มาพลอตกราฟจะได้กราฟดังภาพที่ 3.13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.13 กราฟการใช้งาน CPU เฉลี่ยทุกๆ 20 นาที

จากนั้นก็นำข้อมูลที่ได้มาทำนาย Traffic โดยใช้โมเดลที่ชื่อว่า ARIMA มาคำนวณ โมเดลนี้ใช้สำหรับการทำนายข้อมูลแบบ Time series ข้อมูลที่นำมาทำนายนั้นจะต้องเป็น Stationary (คงที่) นั่นคือ ไม่มี Trend หรือ Seasonality ซึ่งหากข้อมูลของเราเป็น Non-Stationary ก็ต้องทำให้เป็น Stationary ก่อน ด้วยการ Differencing หรือการหาความแตกต่างระหว่างค่าปัจจุบันกับค่าก่อนหน้า โดยโมเดล ARIMA จะเขียนอยู่ในรูปแบบของ ARIMA(p,d,q) โดยที่ค่า p, d, q คือ

AR (p) = จำนวนลำดับของ AutoRegressive

I (d) = จำนวนครั้งการ Differencing ของข้อมูล

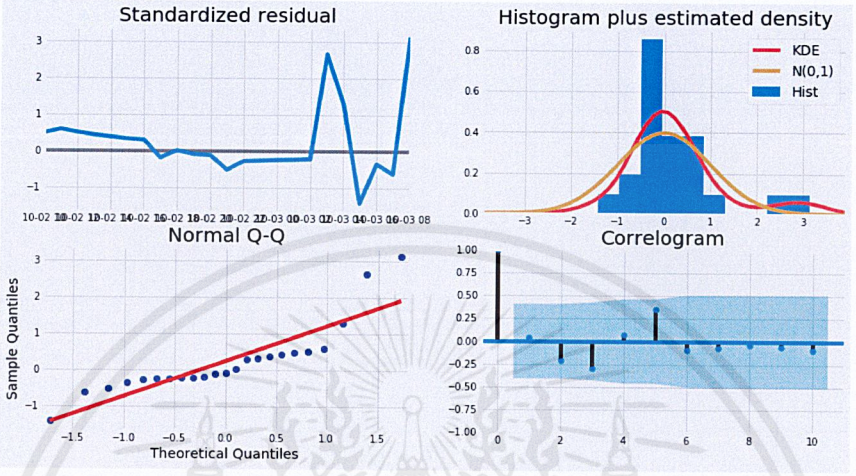
MA (q) = จำนวนลำดับของ Moving Average

โดยวิธีที่เราจะหาค่า p, d, q ที่เหมาะสมเพื่อมาคำนวณนั้นหาจากโมเดล AIC (Akaike Information Criterion) ซึ่งโมเดลที่ได้ค่า AIC ต่ำที่สุดจะสามารถนำมาทำนายค่าได้ดีที่สุด โดยสร้างลูปวนหาค่า p, d, q ที่อยู่ระหว่าง 0 ถึง 1 ที่ทำให้ได้ค่า AIC น้อยที่สุดและนำค่าของ p, d, q นั้นมาใช้ทำนายโมเดล ARIMA โดยค่า p, d, q ที่หาได้จากข้อมูลที่มีอยู่คือ (1, 1, 1)

เมื่อได้ค่า p, d, q มาแล้ว ต่อมาให้นำค่านั้นมาคำนวณหา Model diagnostic เพื่อทำให้แน่ใจว่าโมเดลที่เราได้มานั้นสามารถใช้ทำนายได้ และตรวจสอบว่าโมเดลนั้นไม่มีพฤติกรรมที่ผิดปกติ ซึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ควรมีลักษณะการกระจายตัวแบบปกติ (Normal distribution) โดยกราฟที่ได้จากการหา Model diagnostic จากค่า p, d, q เท่ากับ (1, 1, 1) ได้ผลลัพธ์ตามภาพที่ 3.14



ภาพที่ 3.14 ผลการรัน Model diagnostic

จากกราฟที่ได้ เราสามารถนำมาวิเคราะห์ว่าข้อมูลของเรากระจายตัวแบบปกติจาก

กราฟ Standardized residual ไม่มีพฤติกรรรมที่เป็น Seasonality และกราฟ Correlogram มีค่า lag ที่ต่ำ

กราฟ Histogram เส้น KDE อยู่ใกล้เคียงกับเส้น N(0, 1) ซึ่งเป็นเส้นแสดง Normal distribution ที่ค่าเฉลี่ยอยู่ที่ 0

กราฟ Q-Q แสดงลำดับการกระจายตัวของความคลาดเคลื่อน(จุดสีฟ้า) ที่มีแนวโน้มเป็นไปตามค่าการกระจายตัวแบบปกติ(เส้นสีแดง)

จากข้อมูลข้างต้นสรุปได้ว่าโมเดลของเรามีการกระจายตัวแบบปกติ สามารถใช้ทำนายได้ จากนั้นเราสามารถตรวจสอบโมเดลที่ใช้ทำนายได้โดยนำมาทำนาย Traffic เทียบกับ Traffic จริงๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.7 สร้าง CloudWatch Event Rule

เมื่อได้เวลาที่ต้องการที่จะเพิ่มเครื่องมาแล้ว ต่อมาให้ทำการสร้าง CloudWatch Event Rule เพื่อไป Trigger Lambda function ที่เราเขียนไว้ก่อนหน้านี้ให้ไป Execute policy เพื่อเพิ่มเครื่องโหนดอีก 1 เครื่อง โดยเราจะสร้าง Rule ผ่านการเขียนฟังก์ชั่นคำสั่ง Command line ซึ่งมีอยู่ 2 คำสั่งหลักๆ คือ PutRule และ PutTarget

การกำหนดเวลาที่จะให้ทำการ Trigger Lambda นั้นจะใช้นำเวลาที่ได้มาใส่ cron ที่มีรูปแบบคือ cron(Minutes Hours Day-of-month Month Day-of-week Year) โดยเวลาเป็น UTC เช่น เวลาที่ต้องการคือ 11/10/2019 15.00 UTC+7 จะได้ cron(0, 8, 11, 10, ?, 2019) ส่วนการกำหนดอินพุทให้เรียกผ่านไฟล์ JSON

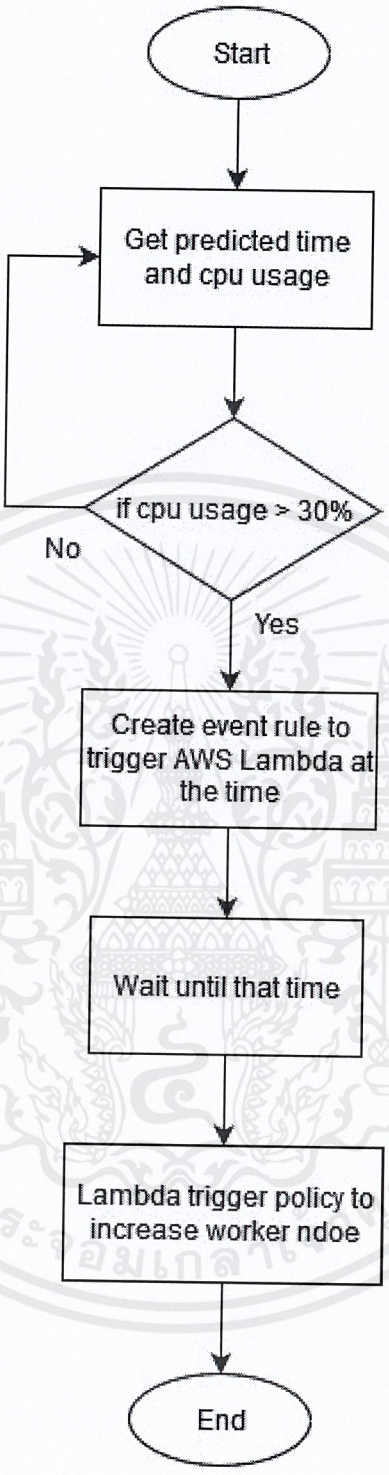
จะได้คำสั่ง PutRule และ PutTarget ดังนี้

aws events put-rule --name <<RuleName>> --schedule-expression cron(0, 8, 11, 10, ?, 2019)

aws events put-targets --rule <<RuleName>> --targets \"file://targets.json\"

ก็จะได้ EventRule ที่มีค่าเวลาที่กำหนดไว้เพื่อรอไป trigger ฟังก์ชั่นของ Lamda โดยมีอินพุทเป็น policy และ node group ที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.15 ขั้นตอนการเพิ่มเครื่องโหนดหลังจากได้ค่าที่ทำนาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.8 Deploy Horizontal Pod Autoscaler

Horizontal Pod Autoscaler หรือ HPA เป็นการสเกลจำนวนพอด โดยวัดจากการใช้งานของเมตริกที่สนใจ ดังนั้นคลัสเตอร์ของเราจะต้องทำการติดตั้ง metrics-server ที่เป็นตัวเก็บเมตริกของพอดก่อน โดยทำการโคลน metrics-server มาจาก Github ของ Kubernetes แล้วจึง Deploy

git clone https://github.com/kubernetes-incubator/metrics-server/

kubectl create -f deploy/1.8+/

จากนั้นติดตั้ง HPA ด้วยคำสั่ง

kubectl autoscale deployment/<<autoscaler-demo-app>> --cpu-percent=50 --min=1 --max=10

โดยที่ <<autoscaler-demo-app>> คือชื่อ deployment ของแอปพลิเคชั่นที่ต้องการที่จะทำการสเกล ส่วน cpu-percent คือการกำหนด threshold ของ metric ที่ต้องการ และค่า min และ max คือจำนวนพอดต่ำสุดและสูงสุดที่สามารถมีได้

เมื่อติดตั้งเสร็จแล้วสามารถตรวจสอบได้โดยใช้คำสั่ง kubectl get hpa

เมื่อทีการใช้ CPU เฉลี่ยเกินทาร์เก็ต 50% ก็จะเพิ่มพอดของแอปพลิเคชั่นนั้นจนกว่าการใช้งานจะลดลงต่ำกว่าทาร์เก็ตเป็นเวลา 5 นาที จึงจะทำการลดจำนวนพอดลง

3.2.9 Deploy Cluster Autoscaling

Cluster Autoscaling (CA) คือการสเกลจำนวนโหนดในคลัสเตอร์เมื่อมีจำนวนพอดที่อยู่ในสถานะ Pending เกินกว่าเวลาที่กำหนด หมายความว่าจำนวนเครื่องที่มีอยู่นั้นมีทรัพยากรไม่พอที่จะรันพอดทั้งหมดได้ ก็จะทำการเพิ่มเครื่องขึ้นมาจนกว่าทุกพอดจะอยู่ในสถานะ Running โดยมีวิธีการดังนี้

สร้าง Policy ใน AWS IAM ชื่อ ClusterAutoScaler แล้วนำไปติดกับ Role ของโหนดที่มีชื่อว่า NodeInstanceRole โดยกำหนด Policy ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeTags",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:TerminateInstanceInAutoScalingGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

ภาพที่ 3.16 ClusterAutoScaler policy

ดาวน์โหลดไฟล์ CA จาก Github ของ Kubernetes

wget https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml

แก้ไข Configuration ของไฟล์ให้เรียบร้อยและสร้าง CA Deployment โดยใช้คำสั่ง

kubectl apply -f cluster-autoscaler-autodiscover.yaml

ตรวจสอบ logs โดยใช้คำสั่ง

kubectl logs -f deployment/cluster-autoscaler -n kube-system

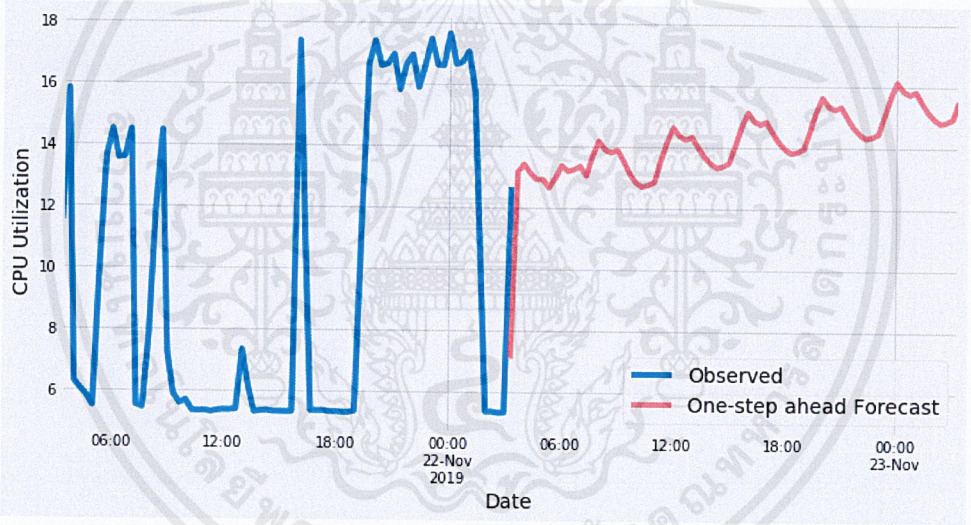
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

ผลการวิจัย

4.1 ผลการทำนาย Traffic

ผลการทำนาย Traffic จากการใช้โมเดล ARIMA ของข้อมูลระหว่างวันที่ 21 พฤศจิกายน เวลา 3:30 น. ถึงวันที่ 22 พฤศจิกายน เวลา 3:30 น. และทำนายตั้งแต่วันที่ 22 พฤศจิกายน เวลา 3:30 น. ถึงวันที่ 23 พฤศจิกายน เวลา 3:30 น. ดังภาพที่ 4.1



ภาพที่ 4.1 ตรวจสอบการทำนาย Traffic

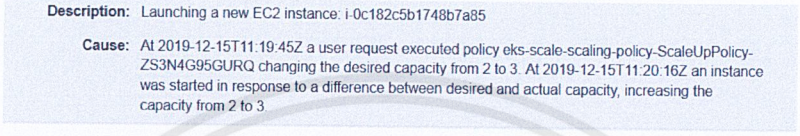
จากภาพที่ 4.1 สีฟ้าคือกราฟของ Traffic ที่เราสร้างขึ้นมา สีชมพูคือกราฟของ Traffic ที่ทำนายได้จากการใช้โมเดล ARIMA

เมื่อนำค่าที่ได้จากการทำนายมาคำนวณหาค่าความผิดพลาด Mean Square Error = 9.96

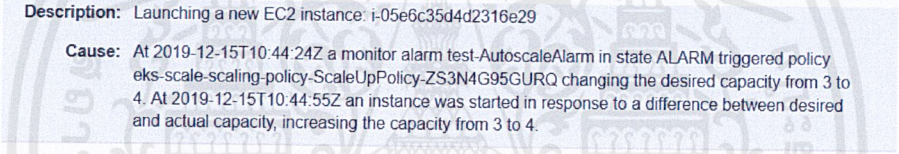
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ผลการเพิ่มเครื่อง worker node

เมื่อนำเวลาที่ต้องการให้ทำการเพิ่มเครื่องโหนดมาสร้างอีเวนท์และสั่งให้ lambda ฟังก์ชั่นทำงาน หากเพิ่มเครื่องสำเร็จก็จะได้ผลลัพธ์ดังภาพที่ 4.2 แต่หากเป็นการเพิ่มเครื่องอัตโนมัติโดยวัดจาก CPU Utilization ในขณะนั้นก็จะได้ผลลัพธ์ดังภาพที่ 4.3

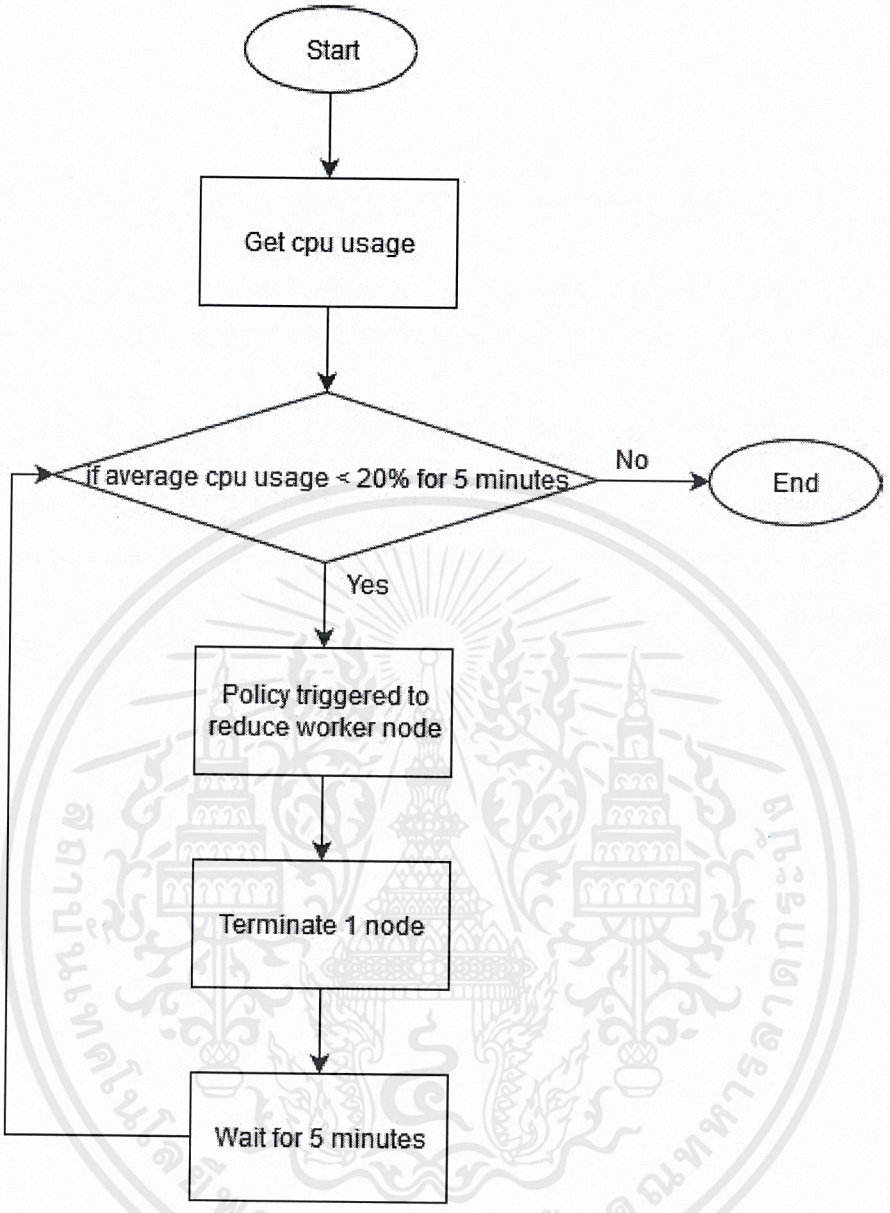


ภาพที่ 4.2 ผลการเพิ่มเครื่องโหนดจากการกำหนดเวลาเอง



ภาพที่ 4.3 ผลการเพิ่มเครื่องเมื่อ CPU Utilization ถึงค่าที่กำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.4 ขั้นตอนการลดเครื่องโหนดเมื่อมีการใช้งาน CPU ต่ำกว่าที่กำหนดไว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากผลการวิจัย ระบบสามารถทำการเพิ่มเจำนวนพอดเมื่อมีการใช้งาน CPU ในพอดมากกว่าระดับที่กำหนดได้ และเมื่อการใช้พอดเกินกว่าที่โหนดจะรับได้ก็สามารถเพิ่มจำนวนโหนดได้จนพอดทุกพอดสามารถรันได้หมด สามารถทำนายการใช้ CPU ของเครื่องโหนดได้ โดยดูแนวโน้มจากการใช้งาน CPU ของโหนดที่ผ่านมา

5.2 ปัญหาอุปสรรคและข้อเสนอแนะ

พอดที่รันอยู่บน kuernetes นั้นต้องทำการ config ไว้ด้วยว่าหากมีการเพิ่มจำนวนพอดให้เพิ่มออกไปยังเครื่องโหนดที่ไม่เคยมีพอดนั้นก่อน เพราะหากเพิ่มมาที่โหนดเครื่องเดิมนั้น การเพิ่มเครื่องโหนดมาก็ไม่ได้ช่วยเรื่องการทำงานแต่อย่างใด

การเพิ่มจำนวนพอดนั้นสามารถทำได้อย่างไม่มีปัญหา แต่ในการลดจำนวนพอดเมื่อการใช้งานกลับมาเป็นปกตินั้นเราไม่สามารถกำหนด threashold ได้ว่าต้องการให้ CPU อยู่ระดับใดจึงทำการลดจำนวนพอด แต่เป็นค่า default ของ kubernetes ที่กำหนดให้เมื่อค่าเฉลี่ยของการใช้งาน CPU ในเวลา 5 นาที มีค่าต่ำกว่า threashold ของการเพิ่มจำนวนพอด จะทำการลดจำนวนพอดลงให้เองโดยอัตโนมัติ ทั้งนี้เป็นเพราะว่าหากต้องการแก้ไขค่า threashold และเวลาที่ใช้เก็บค่าสำหรับการลดจำนวนพอดจะต้องเข้าไปแก้ไขในไฟล์ manifest ของ control plane ของ kubernetes ซึ่งส่วนนี้ AWS เป็นผู้จัดการทั้งหมดและยังไม่มีฟังก์ชั่นที่เปิดให้เราเข้าไปแก้ไขส่วนใดส่วนหนึ่งได้ในตอนนี้ แต่ถ้าหากเราใช้ kubernetes ที่เราเป็นคน manage เองทั้งหมดก็สามารถเข้าไปแก้ไขส่วนนี้ได้เลยในไฟล์ manifest

ในการทำนายการใช้งาน CPU โหลดนั้น ทราฟฟิคที่นำมาทำนายเป็นทราฟฟิคที่เรา generate ขึ้นมาเอง ทำให้อาจจะดูมีแนวโน้มที่สามารถนำมาทำนายได้ดี ส่วนทราฟฟิคของแอปพลิเคชั่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่มีอยู่นั้น ส่วนมากโดยปกติมักจะโหลดที่ค่อนข้างคงที่ ซึ่งหากนำมาทำนายก็อาจจะไม่ได้ส่งผลให้ได้การใช้งานจนถึงระดับที่จะต้องทำการเพิ่มเครื่องโหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

[1] Amazon Documentation, 2019, docs.aws.amazon.com/. Accessed 3 Aug. 2019.

[2] Kubernetes Documentation. Kubernetes.Io, 2019, kubernetes.io/docs/home/. Accessed 20 Aug. 2019.

[3] Get Started with Docker | Docker. Docker, 2019, www.docker.com/get-started. Accessed 18 Aug. 2019.

[4] Hyndman, Rob J, and George Athanasopoulos. Forecasting: Principles and Practice. Otexts.Com, 2016, otexts.com/fpp2/. Accessed 15 Nov. 2019.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้