

หลักสูตรความมั่นคงปลอดภัยระบบคลาวด์

กิจกรรมพัฒนาบุคลากรและวิทยากรด้านความมั่นคง
ปลอดภัยไซเบอร์ตามมาตรฐานสากลให้กับประชาชน



สำนักงานคณะกรรมการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ

สารบัญ

บทที่ 1 ความเข้าใจเกี่ยวกับสถาปัตยกรรมระบบคลาวด์ (Understanding Cloud Architecture Concepts).....	1
หน่วยที่ 1 การทำความเข้าใจแนวคิดและรูปแบบการให้บริการของระบบคลาวด์ (Recognize Cloud Concepts and Service Models)	1
หน่วยที่ 2 การทำความเข้าใจคำศัพท์พื้นฐานเกี่ยวกับระบบคลาวด์ (Recognize Cloud Terms).....	16
หน่วยที่ 3 เทคโนโลยีในระบบคลาวด์ (Technologies in the Cloud)	21
หน่วยที่ 4 การแก้ไขปัญหาในสภาพแวดล้อมระบบคลาวด์ (Troubleshoot the Cloud Environment)	26
บทที่ 2 การวางแผนบริการระบบคลาวด์ (Planning Cloud Services)	31
หน่วยที่ 1 การออกแบบแบบคลาวด์เนทีฟ (Cloud-native Design).....	31
หน่วยที่ 2 รูปแบบการปรับใช้ในคลาวด์ (Cloud Deployment Models)	37
หน่วยที่ 3 กลยุทธ์การปรับใช้คลาวด์ (Cloud Deployment Strategies).....	49
บทที่ 3 การจัดเตรียมและย้ายทรัพยากรไปยังคลาวด์ (Provisioning and Migrating Cloud Resources)	57
หน่วยที่ 1 การจัดเตรียมทรัพยากรในคลาวด์ (Provision Cloud Resources).....	57
หน่วยที่ 2 ประเด็นสำคัญของการย้ายระบบไปยังคลาวด์ (Aspects of Cloud Migration)	66
หน่วยที่ 3 ข้อควรพิจารณาในการย้ายระบบขึ้นคลาวด์ (Cloud Migration Considerations).....	71
บทที่ 4 การเปรียบเทียบระบบจัดเก็บข้อมูลบนคลาวด์ (Comparing Cloud Storage).....	78
หน่วยที่ 1 ทรัพยากรและเทคโนโลยีในการจัดเก็บข้อมูล (Storage Resources and Technologies) ...	78
บทที่ 5 การแก้ไขปัญหาการปรับใช้และต้นทุน (Troubleshooting Deployment Issues and Cost)	91
หน่วยที่ 1 การแก้ไขปัญหาการปรับใช้และการย้ายระบบ (Troubleshoot Deployment and Migration Issues)	91
หน่วยที่ 2 ข้อควรพิจารณาเกี่ยวกับค่าใช้จ่ายในการใช้งานระบบคลาวด์ (Cost Considerations Related to Cloud Usage)	101
บทที่ 6 การใช้เทคโนโลยีเวอร์ชวลไลเซชันและฐานข้อมูล (Using Virtualization and Databases)	110
หน่วยที่ 1 แนวคิดเกี่ยวกับเวอร์ชวลไลเซชัน (Virtualization Concepts)	110
หน่วยที่ 2 แนวคิดเกี่ยวกับการใช้คอนเทนเนอร์ (Containerization Concepts).....	120
หน่วยที่ 3 แนวคิดพื้นฐานเกี่ยวกับฐานข้อมูล (Database Concepts).....	128

สารบัญ (ต่อ)

บทที่ 7 การเข้าใจระบบเครือข่ายบนคลาวด์ (Comprehending Cloud Networking).....	135
หน่วยที่ 1 แนวคิดเกี่ยวกับระบบเครือข่ายบนคลาวด์ (Cloud Networking Concepts).....	135
หน่วยที่ 2 พังก์ชันเครือข่าย (Network Functions)	145
หน่วยที่ 3 องค์ประกอบของเครือข่าย (Network Components)	149
หน่วยที่ 4 บริการเครือข่าย (Network Services)	158
หน่วยที่ 5 บริการเครือข่ายในระบบคลาวด์ (Cloud Network Services)	165
หน่วยที่ 6 การแก้ไขปัญหาเครือข่าย (Troubleshoot Network Issues).....	168
บทที่ 8 การทำงานแบบอัตโนมัติของทรัพยากรคลาวด์ (Automating Cloud Resources).....	182
หน่วยที่ 1 การปรับใช้และกำหนดค่าทรัพยากรคลาวด์โดยใช้โค้ด (Deploy and Configure Cloud Resources using Code).....	182
หน่วยที่ 2 การผสานรวมอย่างต่อเนื่องและการปรับใช้แบบต่อเนื่อง (Continuous Integration / Continuous Deployment Pipelines).....	188
หน่วยที่ 3 เครื่องมือที่ใช้ในสภาพแวดล้อม DevOps (Tools Used in DevOps Environments)	193
หน่วยที่ 4 แนวคิดเกี่ยวกับการควบคุมเวอร์ชันของซอฟต์แวร์ (Source Control Concepts)	197
หน่วยที่ 5 การเขียนสคริปต์ (Scripting)	203
หน่วยที่ 6 การบูรณาการระบบ (Integration of Systems)	205
บทที่ 9 การบริหารจัดการด้านความมั่นคงปลอดภัยในระบบคลาวด์ (Implementing Security Management).....	209
หน่วยที่ 1 แนวคิดเรื่องการบริหารจัดการอัตลักษณ์และการเข้าใช้ (Identity and Access Management: IAM)	210
หน่วยที่ 2 การควบคุมความปลอดภัยในระบบคลาวด์ (Security Controls in the Cloud).....	215
หน่วยที่ 3 แนวคิดเกี่ยวกับการจัดการช่องโหว (Vulnerability Management Concepts)	219
หน่วยที่ 4 การเฝ้าระวังกิจกรรมที่น่าสงสัยเพื่อรับการแจ้งเตือนที่ไว (Monitoring Suspicious Activities to Identify Common Attacks)	220
บทที่ 10 ความเข้าใจด้านการปฏิบัติตามข้อกำหนดด้านความมั่นคงปลอดภัย และการแก้ไขปัญหา (Comprehending Security Compliance and Troubleshooting).....	227
หน่วยที่ 1 ประเด็นด้านการปฏิบัติตามข้อกำหนดและข้อบังคับ (Aspects of Compliance and Regulation)	227
หน่วยที่ 2 แนวปฏิบัติที่ดีที่สุดด้านความมั่นคงปลอดภัย (Security Best Practices)	231
หน่วยที่ 3 การแก้ไขปัญหาด้านความมั่นคงปลอดภัย (Troubleshooting Security Issues)	239

สารบัญ (ต่อ)

บทที่ 11 การดำเนินการด้านประสิทธิภาพและการตรวจสอบระบบ (Implementing Performance and Monitoring).....	244
หน่วยที่ 1 การเพิ่มประสิทธิภาพการประมวลผลงานโดยใช้ทรัพยากระบบคลาวด์ (Optimize Workloads Using Cloud Resources)	244
หน่วยที่ 2 การกำหนดแนวทางการปรับขนาดที่เหมาะสม (Configure Appropriate Scaling Approaches)	251
หน่วยที่ 3 การกำหนดค่าทรัพยากรเพื่อให้เกิดการสังเกตการณ์ (Configure Resources to Achieve Observability).....	253
หน่วยที่ 4 การจัดการวงจรชีวิตของทรัพยากรบนคลาวด์ (Managing the Life Cycle of Cloud Resources)	259
บทที่ 12 การจัดการการกู้คืนจากภัยพิบัติและความต้องเนื่องทางธุรกิจ (Managing Disaster Recovery and Business Continuity)	262
หน่วยที่ 1 วิธีการสำรองและกู้คืนข้อมูล (Backup and Recovery Methods)	262
หน่วยที่ 2 แนวคิดเกี่ยวกับความพร้อมให้บริการของระบบ (Service Availability Concepts)	267
บทที่ 13 การประยุกต์ใช้คลาวด์คอมพิวติ้งภายใต้กรอบกฎหมายและข้อบังคับของไทย (Applying Cloud Computing within the Thai Legal and Regulatory Framework)	271
หน่วยที่ 1 ภาพรวมกฎหมายดิจิทัลของไทยที่เกี่ยวข้องกับระบบคลาวด์ (Overview of Thai Digital Laws Relevant to Cloud Computing).....	271
หน่วยที่ 2 การปฏิบัติตาม PDPA ในสภาพแวดล้อมคลาวด์ (PDPA Compliance in a Cloud Environment).....	275
หน่วยที่ 3 การดำเนินการตาม พ.ร.บ. ความมั่นคงปลอดภัยไซเบอร์ (Implementing the Cybersecurity Act)	279
หน่วยที่ 4 กรณีศึกษาและการดำเนินการอย่างเป็นขั้นตอน (Case Studies and Step-by-Step Implementation).....	281
บทที่ 14 หลักการเป็นวิทยากรที่ดี (Principles of Effective Instruction)	285
หน่วยที่ 1 คุณลักษณะและจรรยาบรรณของวิทยากร (Qualities and Ethics of an Instructor)	286
หน่วยที่ 2 การเตรียมความพร้อมก่อนการบรรยาย (Pre-Instruction Preparation)	288
หน่วยที่ 3 เทคนิคการนำเสนอและการสร้างปฏิสัมพันธ์ (Presentation and Interaction Techniques)	291
หน่วยที่ 4 การวัดผลและประเมินผลการเรียนรู้ (Measuring and Evaluating Learning).....	293
หน่วยที่ 5 การพัฒนาตนเองอย่างต่อเนื่อง (Continuous Professional Development)	295

สารบัญ (ต่อ)

คำศัพท์ (Glossary) 298

สารบัญรูปภาพ

ภาพที่ 1 Platform as a Service	5
ภาพที่ 2 Infrastructure as a Service	6
ภาพที่ 3 Software as a Service (SaaS)	12
ภาพที่ 4 Role of Cloud Managed Service Providers	14
ภาพที่ 5 Shared Responsibility Model	16
ภาพที่ 6 Virtualization Review	18
ภาพที่ 7 Service Discovery	35
ภาพที่ 8 Function Chain	36
ภาพที่ 9 Fan-out/Fan-in	36
ภาพที่ 10 Cloud Components	38
ภาพที่ 11 Public Cloud	39
ภาพที่ 12 Private Cloud	41
ภาพที่ 13 Community Cloud	42
ภาพที่ 14 Hybrid Cloud	44
ภาพที่ 15 Multi-cloud	45
ภาพที่ 16 Separate IT Environments	50
ภาพที่ 17 Application Release Models	52
ภาพที่ 18 Blue-Green Deployment Strategy	53
ภาพที่ 19 Canary Deployment Strategy	54
ภาพที่ 20 Migrate Data to the Cloud	83
ภาพที่ 21 AWS M4 Instances	89
ภาพที่ 22 AWS D2 Instances	89
ภาพที่ 23 Resource Tagging	106
ภาพที่ 24 Rightsizing	107
ภาพที่ 25 Stand-Alone Virtualization	112
ภาพที่ 26 Virtual Machine Configuration File Templates	114
ภาพที่ 27 Clustering Virtualization	115
ภาพที่ 28 Types of Storage	119
ภาพที่ 29 Stand-Alone Containerization	122
ภาพที่ 30 Storage Types	127
ภาพที่ 31 Network Flow Diagram	137

สารบัญรูปภาพ (ต่อ)

ภาพที่ 32 Site-to-Site VPN	140
ภาพที่ 33 Point-to-Site VPN	141
ภาพที่ 34 IPsec Security	143
ภาพที่ 35 Web Application Firewalls (WAFs)	148
ภาพที่ 36 Single Network with Subnets	151
ภาพที่ 37 Multiple Virtual Networks Configured as Peers	152
ภาพที่ 38 Multiple Virtual Networks in a Hub-and-Spoke Topology	153
ภาพที่ 39 Microsegmentation	154
ภาพที่ 40 Virtual LAN Technologies	158
ภาพที่ 41 Software-Defined Networking	160
ภาพที่ 42 Content Delivery Network	164
ภาพที่ 43 Dynamic Host Configuration Protocol	166
ภาพที่ 44 Compute Resources for Containers	247
ภาพที่ 45 Log File Analysis	256

บทที่ 1

ความเข้าใจเกี่ยวกับสถาปัตยกรรมระบบคลาวด์

(Understanding Cloud Architecture Concepts)

บทนำของบทเรียน

บทเรียนนี้ครอบคลุมแนวคิดพื้นฐานและคำศัพท์ที่สำคัญเกี่ยวกับระบบคลาวด์ โดยมีวัตถุประสงค์เพื่อให้ผู้ที่เกี่ยวข้องกับการบริหารจัดการระบบคลาวด์สามารถสื่อสารได้อย่างชัดเจนและมีประสิทธิภาพภายในเนื้อหาจะมีการให้คำจำกัดความของรูปแบบการให้บริการคลาวด์ต่างๆ ตลอดจนแนวคิดเรื่องโมเดลความรับผิดชอบร่วมกัน (Shared Responsibility Model) นอกจากนี้ ยังกล่าวถึงแนวโน้มเทคโนโลยีคลาวด์ที่กำลังพัฒนา และนำเสนอแนวทางการวิเคราะห์และแก้ไขปัญหา (CompTIA Troubleshooting Methodology) ซึ่งเป็นเครื่องมือสำคัญสำหรับผู้ดูแลระบบคลาวด์

วัตถุประสงค์ของบทเรียน

เมื่อจบการเรียนรู้ในบทเรียนนี้ ผู้เรียนจะสามารถ:

- เข้าใจแนวคิดพื้นฐานของระบบคลาวด์
- ตระหนักรถึงการพัฒนาและวิวัฒนาการของเทคโนโลยีคลาวด์
- เข้าใจแนวทางการวิเคราะห์และแก้ไขปัญหาอย่างเป็นระบบ (Troubleshooting Methodology)

หน่วยที่ 1 การทำความเข้าใจแนวคิดและรูปแบบการให้บริการของระบบคลาวด์ (Recognize Cloud Concepts and Service Models)

เพื่อให้สามารถเข้าใจบริการคลาวด์และผู้ให้บริการระบบคลาวด์ได้อย่างถูกต้อง จำเป็นต้องมีความรู้พื้นฐานที่ชัดเจนเกี่ยวกับแนวคิดและคำศัพท์ที่ใช้ในบริบทนี้ บทเรียนนี้จะช่วยสร้างความเข้าใจเกี่ยวกับแนวคิดพื้นฐานของระบบคลาวด์และคำศัพท์ที่เกี่ยวข้อง เนื่องจากผู้ให้บริการแต่ละรายมักใช้ชื่อเรียกบริการที่แตกต่างกัน ทำให้ผู้ใช้งานเกิดความสับสนได้ง่าย

คำว่า “คลาวด์” อาจมีความหมายแตกต่างกันไปในแต่ละบริบท ดังนั้น เพื่อให้เกิดความเข้าใจที่เป็นมาตรฐาน บทเรียนนี้จึงอ้างอิงนิยามของระบบคลาวด์ตามที่กำหนดโดยสถาบันมาตรฐานและเทคโนโลยีแห่งชาติ (NIST)

จากนั้นจะอธิบายถึงรูปแบบการให้บริการหลักของระบบคลาวด์ 3 รูปแบบ ซึ่งเป็นพื้นฐานสำคัญในการเลือกใช้งาน ได้แก่ IaaS, PaaS และ SaaS ต่อด้วยการจำแนกรูปแบบการนำระบบคลาวด์ไปใช้งาน (Cloud Deployment Models) ซึ่งครอบคลุมการใช้งานทั้งในรูปแบบสาธารณะ ส่วนบุคคล และแบบผสม

ท้ายบทเรียนจะกล่าวถึงแนวคิดเรื่องความรับผิดชอบร่วมกันด้านความปลอดภัย (Shared Responsibility Model) ซึ่งเป็นหลักการสำคัญในการจัดการความปลอดภัยของข้อมูลและระบบในสภาพแวดล้อมคลาวด์

1.1 รูปแบบการให้บริการคลาวด์ (Cloud Service Models)

1.1.1 คลาวด์คืออะไร

คำว่า “คลาวด์” หรือ “การประมวลผลแบบคลาวด์” (Cloud Computing) อาจสร้างความสับสนให้กับผู้เริ่มต้น เนื่องจากมีบริการหลากหลายรูปแบบจากผู้ให้บริการที่ใช้ชื่อเรียกแตกต่างกัน อย่างไรก็ตาม สถาบันมาตรฐานและเทคโนโลยีแห่งชาติของสหรัฐอเมริกา (National Institute of Standards and Technology: NIST) ได้ให้คำจำกัดความที่ชัดเจนของคุณลักษณะที่สำคัญของระบบคลาวด์ไว้ 5 ประการ ซึ่งช่วยให้สามารถเข้าใจลักษณะของบริการคลาวด์ได้อย่างเป็นระบบ ดังนี้

1. การให้บริการด้วยตนเองตามต้องการ (On-demand Self-Service)

ผู้ใช้งานสามารถเปิดใช้หรือปรับเปลี่ยนทรัพยากรต่าง ๆ เช่น พื้นที่จัดเก็บข้อมูล กำลังการประมวลผล หรือบริการฐานข้อมูลได้ด้วยตนเองในทันที โดยไม่จำเป็นต้องผ่านเจ้าหน้าที่ของผู้ให้บริการระบบคลาวด์

2. การเข้าถึงได้ผ่านเครือข่ายที่หลากหลาย (Broad Network Access)

ผู้ใช้งานสามารถเข้าถึงบริการคลาวด์ผ่านเครือข่ายอินเทอร์เน็ตได้จากอุปกรณ์ต่าง ๆ ที่รองรับ เช่น คอมพิวเตอร์ แท็บเล็ต หรือโทรศัพท์มือถือ โดยไม่จำกัดสถานที่และเวลา

3. การจัดสรรทรัพยากรแบบรวมศูนย์ (Resource Pooling)

ผู้ให้บริการจะรวมทรัพยากรทั้งหมดไว้ในระบบกลาง และกระจายการใช้งานให้กับผู้ใช้แต่ละราย ในลักษณะของการใช้ทรัพยากรร่วมกัน โดยที่ผู้ใช้ไม่สามารถเห็นรายละเอียดของระบบภายในภาพที่ให้บริการอยู่เบื้องหลัง

4. ความยืดหยุ่นและการปรับขนาดทรัพยากรอย่างรวดเร็ว (Rapid Elasticity)

ระบบสามารถขยายหรือลดปริมาณทรัพยากรได้อย่างรวดเร็วตามความต้องการใช้งานจริง ไม่ว่าจะเป็นแบบอัตโนมัติหรือโดยการควบคุมของผู้ดูแลระบบ

5. การวัดผลและคิดค่าบริการตามการใช้งาน (Measured Service)

ทรัพยากรที่ใช้งานจะถูกตรวจสอบและวัดผลอย่างต่อเนื่อง ทำให้สามารถควบคุม ตรวจสอบ และเรียกเก็บค่าบริการตามปริมาณการใช้งานจริงได้อย่างแม่นยำ

1.1.2 การให้บริการด้วยตนเองตามต้องการ (On-Demand Self-Service)

ผู้ใช้งานสามารถจัดสรรทรัพยากรต่าง ๆ ได้ด้วยตนเองตามความต้องการ โดยไม่จำเป็นต้องติดต่อผู้ให้บริการโดยตรง ทรัพยากรเหล่านี้อาจรวมถึงหน่วยประมวลผลเพิ่มเติม พื้นที่จัดเก็บข้อมูล การสร้างเว็บไซต์ หรือการเปิดใช้งานฐานข้อมูล ผู้ใช้สามารถเพิ่มหรือลดการใช้งานได้ตามความจำเป็นอย่างยืดหยุ่นและรวดเร็ว

1.1.3 การเข้าถึงผ่านเครือข่ายที่หลากหลาย (Broad Network Access)

ผู้ใช้งานสามารถเข้าถึงบริการคลาวด์ได้ผ่านเครือข่าย ไม่ว่าจะเป็นเครือข่ายภายในองค์กร (On-premises Network) หรือเครือข่ายอินเทอร์เน็ต โดยสามารถใช้ผ่านอุปกรณ์หลากหลายประเภท เช่น คอมพิวเตอร์ สมาร์ตโฟน หรือแท็บเล็ต ซึ่งช่วยให้บริการคลาวด์สามารถเข้าถึงได้ทุกที่ทั่วโลก

1.1.4 การรวมทรัพยากรเพื่อใช้งานร่วมกัน (Resource Pooling)

ผู้ให้บริการระบบคลาวด์จะจัดสรรทรัพยากรต่าง ๆ เช่น พลังประมวลผล พื้นที่จัดเก็บข้อมูล และระบบเครือข่าย ให้กับผู้ใช้งานหลายรายผ่านการรวมทรัพยากรไว้ในระบบคลาส (Multi-tenant Model) โดยมีการจัดสรรแบบไดนามิกตามความต้องการ ผู้ใช้ไม่สามารถระบุได้ว่าทรัพยากรที่ใช้งานอยู่มีต้นทางจากที่ใด และในการใช้งานครั้งถัดไป ตำแหน่งของทรัพยากรอาจเปลี่ยนแปลงไปโดยสิ้นเชิง

1.1.5 ความยืดหยุ่นและการปรับขนาดอย่างรวดเร็ว (Rapid Elasticity)

ระบบคลาวด์สามารถขยายหรือลดขนาดของทรัพยากรได้อย่างรวดเร็วและทันต่อความต้องการ ซึ่งหมายความว่าระบบคลาวด์สามารถตัดสินใจเพิ่มหรือลดทรัพยากรตามความต้องการของผู้ใช้โดยอัตโนมัติ ไม่ต้องมีการ干预จากมนุษย์ ทำให้สามารถตอบสนองความต้องการของผู้ใช้ได้ในเวลาอันสั้น ช่วยลดต้นทุนและเพิ่มประสิทธิภาพ

1.1.6 การวัดผลและคิดค่าบริการตามการใช้งานจริง (Measured Service)

ผู้ให้บริการระบบคลาวด์มีระบบตรวจวัดและควบคุมปริมาณการใช้งานของผู้ใช้แต่ละรายอย่างแม่นยำ เช่น การใช้งานหน่วยประมวลผล พื้นที่จัดเก็บ หรือแบนด์วิดธ์ การวัดผลดังกล่าวช่วยให้สามารถบริหารจัดการทรัพยากรอย่างมีประสิทธิภาพ และสามารถเรียกเก็บค่าบริการตามปริมาณการใช้งานจริงของแต่ละผู้ใช้ได้อย่างยุติธรรมและโปร่งใส

คุณลักษณะทั้งห้านี้ถือเป็นหัวใจสำคัญของระบบคลาวด์ ซึ่งเมื่อผสานรวมเข้าด้วยกัน จะกลายเป็นพื้นฐานสำหรับการพัฒนารูปแบบการให้บริการที่หลากหลาย ไม่ว่าจะเป็นในระดับโครงสร้างพื้นฐาน แพลตฟอร์ม หรือแอปพลิเคชัน ทั้งนี้เพื่อให้ตอบโจทย์ความต้องการของผู้ใช้งานในทุกระดับได้อย่างมีประสิทธิภาพ

1.2 การเปรียบเทียบรูปแบบการให้บริการระบบคลาวด์ (Compare Cloud Service Models)

หนึ่งในแนวคิดพื้นฐานที่เข้าใจได้ง่ายที่สุดของการใช้บริการระบบคลาวด์ คือ การ “ถ่ายโอนภาระความรับผิดชอบ” จากผู้ใช้ไปยังผู้ให้บริการ ในระบบไอทีแบบเดิมที่ใช้สถาปัตยกรรมลูกข่าย-แม่ข่าย (Client–Server Model) องค์กรมักต้องรับผิดชอบในการจัดซื้อฮาร์ดแวร์ที่มีราคาสูงและซับซ้อน รวมถึงการจ้างผู้เชี่ยวชาญมาดูและระบบเซิร์ฟเวอร์ พื้นที่จัดเก็บข้อมูล และระบบเครือข่ายต่าง ๆ อีกทั้งยังไม่สามารถปรับขนาดระบบให้เหมาะสมกับความต้องการทางธุรกิจที่เปลี่ยนแปลงได้อย่างยืดหยุ่น

ในทางตรงกันข้าม ระบบคลาวด์เปิดโอกาสให้องค์กรสามารถถ่ายโอนความรับผิดชอบบางส่วนหรือทั้งหมดไปยังผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) ได้ส่งผลให้องค์กรสามารถมุ่งเน้นไปที่การใช้งานมากกว่าการจัดการโครงสร้างพื้นฐาน

รูปแบบการให้บริการระบบคลาวด์สามารถจำแนกเบื้องต้นออกเป็น 3 ประเภทหลักตามระดับของความรับผิดชอบที่ถ่ายโอน อย่างไรก็ตาม เมื่อเทคโนโลยีพัฒนาไปอย่างต่อเนื่อง มีการนิยามรูปแบบบริการเพิ่มเติมที่หลากหลายขึ้น โดยรูปแบบที่สำคัญ 5 ประการที่สะท้อนลักษณะการให้บริการคลาวด์ได้อย่างชัดเจน ได้แก่

1.2.1 ซอฟต์แวร์เป็นบริการ (Software as a Service: SaaS)

ผู้ใช้สามารถเข้าถึงและใช้งานซอฟต์แวร์ได้โดยตรงผ่านเครือข่าย โดยไม่ต้องติดตั้งหรือดูแลระบบเอง ผู้ให้บริการคลาวด์จะรับผิดชอบทั้งหมด ตั้งแต่ฮาร์ดแวร์ ระบบปฏิบัติการ ไปจนถึงการติดตั้ง การอัปเดต และการแก้ไขข้อบกพร่องของซอฟต์แวร์ ตัวอย่างของบริการประเภทนี้ ได้แก่ Google Workspace, Microsoft 365 และ Salesforce

1.2.2 แพลตฟอร์มเป็นบริการ (Platform as a Service: PaaS)

ผู้ให้บริการจะจัดเตรียมโครงสร้างพื้นฐานและสภาพแวดล้อมสำหรับการพัฒนาแอปพลิเคชัน เช่น เครื่องมือพัฒนา ระบบฐานข้อมูล และ Web Server ในขณะที่ผู้ใช้งานมีหน้าที่จัดการโค้ดและเนื้อหาภายในแพลตฟอร์ม ตัวอย่างเช่น Google App Engine และ Heroku

Hosted Applications			
Infrastructure Software			
Operating Systems			
Virtualization			
Servers and Server Hardware			
Networking			
Data Center Electrical and Mechanical Operations			

PaaS provides infrastructure software, operating systems, virtualization, servers and server hardware, networking, and data center electrical and mechanical operations. (Image © 123RF.com.)

ภาพที่ 1 Platform as a Service

1.2.3 โครงสร้างพื้นฐานเป็นบริการ (Infrastructure as a Service: IaaS)

ผู้ให้บริการจัดเตรียมเฉพาะทรัพยากรด้านฮาร์ดแวร์ เช่น เซิร์ฟเวอร์ หน่วยประมวลผล พื้นที่จัดเก็บ และเครื่อข่าย ในขณะที่ผู้ใช้มีหน้าที่ดูแลระบบปฏิบัติการ การติดตั้งซอฟต์แวร์ และการดูแลแอปพลิเคชันต่าง ๆ ที่อยู่บนระบบนั้น ตัวอย่างของ IaaS ได้แก่ Amazon EC2, Microsoft Azure VM และ Google Compute Engine

Infrastructure as a Service (IaaS)

Hosted Applications			
Infrastructure Software			
Operating Systems			
Virtualization			
Servers and Server Hardware			
Networking			
Data Center Electrical and Mechanical Operations			

IaaS provides virtualization, servers and server hardware, networking, and data center electrical and mechanical operations to clients. (Image © 123RF.com.)

ภาพที่ 2 Infrastructure as a Service

1.2.4 ฟังก์ชันเป็นบริการ (Function as a Service: FaaS)

เป็นรูปแบบที่นักพัฒนาสามารถเขียนและรันโค้ดได้โดยไม่ต้องดูแลระบบโครงสร้างพื้นฐานใด ๆ เป็นอย่างหลัง ระบบจะทำงานตามเหตุการณ์หรือคำสั่งที่กำหนดไว้ มักถูกใช้ในการพัฒนาไมโครเซอร์วิส (Microservices) และมักเรียกอีกชื่อว่า “Serverless” ตัวอย่างเช่น AWS Lambda และ Google Cloud Functions

1.2.5 บริการทุกอย่างเป็นบริการ (Anything as a Service: XaaS)

เป็นคำรวมที่ใช้อธิบายบริการระบบคลาวด์รูปแบบต่าง ๆ ที่ขยายออกไปนอกเหนือจากรูปแบบหลัก เช่น ฐานข้อมูลเป็นบริการ (Database as a Service: DBaaS), เดสก์ท็อปเป็นบริการ (Desktop as a Service:

DaaS) และคอนเทนเนอร์เป็นบริการ (Containers as a Service: CaaS) ซึ่งส่วนท่อนถึงการนำโซลูชันด้านเทคโนโลยีไปให้บริการผ่านระบบคลาวด์ในลักษณะที่ครอบคลุมทุกมิติ

1.3 ซอฟต์แวร์เป็นบริการ (Software as a Service: SaaS)

ซอฟต์แวร์เป็นบริการ หรือ SaaS (Software as a Service) เป็นรูปแบบการให้บริการที่ผู้ใช้งานสามารถเข้าถึงและใช้งานซอฟต์แวร์ที่ติดตั้งอยู่บนระบบของผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) ได้โดยตรง โดยไม่ต้องดำเนินการติดตั้งหรือดูแลรักษาด้วยตนเอง ความรับผิดชอบทั้งหมดเกี่ยวกับการติดตั้ง การตั้งค่า การบำรุงรักษา การอัปเดต และการแก้ไขข้อบกพร่องของซอฟต์แวร์จะอยู่ภายใต้การดูแลของผู้ให้บริการ

ซอฟต์แวร์ในรูปแบบ SaaS มักสามารถใช้งานได้จากอุปกรณ์หลากหลายประเภท ไม่ว่าจะเป็นคอมพิวเตอร์ตั้งโต๊ะ โน้ตบุ๊ก แท็บเล็ต หรือสมาร์ตโฟน ทั้งนี้ผู้ใช้จะได้รับสิทธิ์ใช้งานผ่านการเช่ารายเดือนหรือรายปี (Subscription Model) ตามจำนวนผู้ใช้หรือจำนวนไฟล์เอกสารที่เลือกใช้ ทำให้สามารถบริหารต้นทุนด้านซอฟต์แวร์ได้อย่างมีประสิทธิภาพยิ่งขึ้น

หนึ่งในข้อได้เปรียบที่สำคัญของ SaaS คือ ช่วยลดค่าใช้จ่ายในการติดตั้งเริ่มต้น (Initial Deployment Cost) ลดเวลาในการเริ่มใช้งาน และช่วยลดต้นทุนรวมตลอดอายุการใช้งานของระบบ (Total Cost of Ownership) เนื่องจากองค์กรไม่ต้องรับภาระในการบำรุงรักษาซอฟต์แวร์เอง นอก จากนี้ SaaS ยังมีอัตราการใช้งานที่สูงมากในปัจจุบัน โดยเฉพาะในระบบคลาวด์ผ่านอินเทอร์เน็ต

อีกจุดเด่นของ SaaS คือ ความเป็นอิสระจากระบบปฏิบัติการ (Operating System Agnostic) ซึ่งหมายความว่าองค์กรไม่จำเป็นต้องกังวลเกี่ยวกับแพลตฟอร์มที่พนักงานแต่ละคนใช้งาน ตัวอย่างเช่น ในองค์กรที่มีพนักงานบางส่วนใช้ระบบปฏิบัติการ Windows บางส่วนใช้ macOS และอีกกลุ่มหนึ่งใช้ Linux ก็ยังสามารถใช้งานซอฟต์แวร์บนคลาวด์ที่รองรับทุกรอบปีได้ เช่น Dropbox ที่เป็นบริการพื้นที่จัดเก็บข้อมูลแบบออนไลน์ ซึ่งสามารถทำงานได้บนทุกระบบปฏิบัติการข้างต้น หากเป็นการติดตั้งระบบภายในองค์กรแบบดั้งเดิม (On-premises) การสร้างโซลูชันจัดเก็บข้อมูลที่รวมศูนย์ และสามารถควบคุมนโยบายการใช้งานได้อย่างยืดหยุ่นในหลายระบบปฏิบัติการนั้นจะเป็นเรื่องที่ซับซ้อนและมีต้นทุนสูงกว่ามาก

ตัวอย่างบริการ SaaS ที่ใช้งานทั่วไป

- Microsoft Office 365
- Google Apps (เช่น Google Docs, Google Sheets)
- WebEx (ระบบประชุมออนไลน์)
- Dropbox (พื้นที่เก็บข้อมูลบนคลาวด์)
- Netflix (บริการสตรีมมิงเนื้อหา)

กลุ่มเป้าหมายของ SaaS

- ผู้ใช้งานปลายทาง (End-users) ที่ต้องการเข้าถึงซอฟต์แวร์อย่างรวดเร็ว โดยไม่จำเป็นต้องมีความรู้ด้านเทคนิคในการติดตั้งหรือดูแลระบบ

1.4 แพลตฟอร์มเป็นบริการ (Platform as a Service: PaaS)

บริการแพลตฟอร์มเป็นบริการ หรือ PaaS (Platform as a Service) คือ รูปแบบการให้บริการที่ผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) จัดเตรียมโครงสร้างพื้นฐานที่จำเป็นสำหรับการพัฒนาและอพลิเคชันไว้ให้ครบถ้วน ซึ่งประกอบด้วยฮาร์ดแวร์ ระบบปฏิบัติการ และเครื่องมือที่เกี่ยวข้อง โดยผู้ใช้งานสามารถใช้แพลตฟอร์มดังกล่าวในการจัดการข้อมูล พัฒนา และปรับใช้อപลิเคชันได้ทันที โดยไม่จำเป็นต้องติดตั้งหรือตั้งค่าระบบเอง

ผู้ให้บริการจะรับผิดชอบในการดูแลรักษาฮาร์ดแวร์ การสนับสนุนระบบปฏิบัติการ และการบำรุงรักษาแพลตฟอร์มทั้งหมด ขณะที่ผู้ใช้งานจะใช้แพลตฟอร์มตามความต้องการทางธุรกิจของตนเอง ซึ่งช่วยให้สามารถมุ่งเน้นไปที่การพัฒนาและอพลิเคชันหรือบริการต่าง ๆ ได้โดยไม่ต้องกังวลเกี่ยวกับโครงสร้างพื้นฐานที่อยู่เบื้องหลัง

PaaS มักได้รับความนิยมในหมู่นักพัฒนาโปรแกรม (Developers) และผู้ดูแลฐานข้อมูล (Database Administrators: DBAs) เนื่องจากช่วยลดภาระในการจัดหาและดูแลระบบที่ใช้ในการพัฒนา ตัวอย่างเช่น นักพัฒนามาไม่จำเป็นต้องจัดหาเซิร์ฟเวอร์ ติดตั้งระบบปฏิบัติการ หรือจัดตั้งฐานข้อมูลด้วยตนเองอีกด้วย เพรากระบวนการเหล่านั้นถูกถ่ายโอนความรับผิดชอบไปยังผู้ให้บริการระบบคลาวด์แล้ว

นอกจากนี้ PaaS ยังสามารถปรับขนาดได้อย่างรวดเร็วตามความต้องการ (Scalability) และมีกรอบรับสภาพแวดล้อมการพัฒนาและภาษาการเขียนโปรแกรมหลายรูปแบบ อีกทั้งยังสามารถพัฒนาและอพลิเคชันสำหรับอุปกรณ์ได้หลากหลาย ไม่ว่าจะเป็นโทรศัพท์มือถือ แท็บเล็ต คอมพิวเตอร์ สำหรับผู้ใช้งาน หรือแม้กระทั่งเซิร์ฟเวอร์

ตัวอย่างบริการ PaaS ที่ใช้งานทั่วไป

- Google App Engine
- Heroku
- AWS Elastic Beanstalk
- Salesforce Platform

กลุ่มเป้าหมายของ PaaS

- นักพัฒนาโปรแกรม (Developers)
- ผู้ดูแลฐานข้อมูล (Database Administrators: DBAs)

1.5 โครงสร้างพื้นฐานเป็นบริการ (Infrastructure as a Service: IaaS)

โครงสร้างพื้นฐานเป็นบริการ หรือ IaaS (Infrastructure as a Service) คือรูปแบบการให้บริการที่ช่วยให้องค์กรสามารถถ่ายโอนภาระด้านการดูแลฮาร์ดแวร์ไปยังผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) ได้อย่างสมบูรณ์ โดยผู้ใช้งานจะได้รับทรัพยากรพื้นฐานในรูปแบบของเครื่องเสมือน (Virtual Machines) ที่ไฮส์ตอร์ยู่บันโครงสร้างพื้นฐานของผู้ให้บริการ

ในรูปแบบ IaaS ผู้ใช้งานมีหน้าที่ติดตั้งระบบปฏิบัติการด้วยตนเอง เช่น Microsoft Windows หรือ Red Hat Enterprise Linux (RHEL) รวมถึงการกำหนดค่า ดูแลอัปเดต และติดตั้งซอฟต์แวร์ที่จำเป็นทั้งหมด ซึ่งให้ความยืดหยุ่นสูงในการเลือกแพลตฟอร์มที่เหมาะสมกับความต้องการขององค์กร

IaaS ถือเป็นทางเลือกที่มีความคุ้มค่าทางเศรษฐศาสตร์ เนื่องจากช่วยลดต้นทุนในการจัดซื้อฮาร์ดแวร์ และใบอนุญาต (Licensing) ที่มักเกิดขึ้นในระบบไฮเบนเดิม ซึ่งองค์กรต้องประมาณการความต้องการของเซิร์ฟเวอร์ล่วงหน้าในแต่ละรอบงบประมาณ การพิจารณาด้านความจุของบริการก็ทำได้ยาก โดยเฉพาะในอุตสาหกรรมที่มีความต้องการผันผวนตลอดทั้งปี เช่น ธุรกิจค้าปลีกในช่วงเทศกาล แต่ด้วยรูปแบบการคิดค่าบริการแบบจ่ายตามการใช้งานจริง (Pay-as-you-go) ของ IaaS องค์กรสามารถควบคุมต้นทุนได้อย่างมีประสิทธิภาพ จ่ายเฉพาะเท่าที่ใช้งานจริง โดยไม่ต้องแบกรับค่าใช้จ่ายส่วนเกิน

นอกจากนี้ เชิร์ฟเวอร์ที่ติดตั้งภายในองค์กรยังต้องอาศัยโครงสร้างพื้นฐานจำนวนมาก ไม่ว่าจะเป็น พื้นที่ติดตั้ง ระบบไฟฟ้า ระบบประbayความร้อน และมาตรการรักษาความปลอดภัยทางกายภาพ ซึ่งส่งผลให้เกิดค่าใช้จ่ายในการดำเนินงาน (Operating Expenditures: OpEx) ที่สูง การใช้บริการ IaaS จึงช่วยลดภาระเหล่านี้ได้อย่างมีนัยสำคัญ

ตัวอย่างบริการ IaaS ที่ใช้กันอย่างแพร่หลาย

- Amazon Web Services (AWS) Elastic Compute Cloud (EC2)
- Microsoft Azure
- Rackspace
- Digital Ocean

กลุ่มเป้าหมายของ IaaS

- ผู้ดูแลระบบสารสนเทศ (IT Administrators) ที่ต้องการควบคุมการติดตั้งและบริหารจัดการระบบเอง ในขณะที่ลดภาระด้านโครงสร้างพื้นฐาน

1.6 พังก์ชันเป็นบริการ (Function as a Service: FaaS)

พังก์ชันเป็นบริการ หรือ FaaS (Function as a Service) เป็นรูปแบบการให้บริการที่ช่วยให้นักพัฒนาสามารถเขียนและรันโค้ดได้อย่างง่ายดาย โดยไม่ต้องจัดการหรือดูแลโครงสร้างพื้นฐานด้านการประมวลผล ที่อยู่เบื้องหลัง FaaS มักถูกเรียกว่า “Serverless” เนื่องจากผู้ใช้ไม่ต้องจัดการกับเซิร์ฟเวอร์โดยตรง ถึงแม้ว่า การประมวลผลจะยังคงเกิดขึ้นบนเซิร์ฟเวอร์จริงก็ตาม

FaaS เหมาะอย่างยิ่งสำหรับการสร้างและใช้งานไมโครเซอร์วิส (Microservices) เช่น การประมวลผลข้อมูล การวิเคราะห์แบบเรียลไทม์ หรือการจัดการข้อมูลจากอุปกรณ์ Internet of Things (IoT) โดยระบบจะเรียกใช้โค้ดโดยอัตโนมัติเมื่อมีเหตุการณ์หรือคำสั่งที่กำหนดไว้ (Event-driven Execution)

แม้คำว่า “Serverless” จะสื่อว่าไม่มีเซิร์ฟเวอร์ แต่ในความเป็นจริง ระบบยังคงใช้ทรัพยากรการประมวลผลอยู่ เพียงแต่จะมีการจัดสรรทรัพยากรตามความจำเป็นเฉพาะช่วงเวลาที่แอปพลิเคชันทำงานเท่านั้น เมื่อแอปพลิเคชันหยุดทำงาน ระบบจะคืนทรัพยากรกลับโดยอัตโนมัติ ส่งผลให้การเรียกเก็บค่าบริการจะห้อนตามการใช้งานจริง

สิ่งที่ทำให้ FaaS โดดเด่น คือ นักพัฒนาไม่จำเป็นต้องวางแผนเรื่องขนาดทรัพยากร ไม่ต้องติดตั้งระบบไม่ต้องดูแลเซิร์ฟเวอร์ และไม่ต้องค่อยตรวจสอบสถานะระบบอีกต่อไป ทำให้สามารถมุ่งเน้นไปที่การพัฒนาและปรับปรุงฟังก์ชันของซอฟต์แวร์ได้อย่างเต็มที่

ตัวอย่างบริการ FaaS ที่ใช้กันอย่างแพร่หลาย

- Amazon Web Services (AWS) Lambda
- Google Cloud Functions (บน Google Cloud Platform)
- Azure Functions (จาก Microsoft)

กลุ่มเป้าหมายของ FaaS

- นักพัฒนาซอฟต์แวร์ (Developers) ที่ต้องการสร้างระบบแบบอัตโนมัติและยืดหยุ่น โดยไม่ต้องจัดการโครงสร้างพื้นฐาน

1.7 ทุกสิ่งเป็นบริการ (Anything as a Service: XaaS)

รูปแบบการให้บริการ “ทุกสิ่งเป็นบริการ” หรือ XaaS (Anything as a Service) โดยที่ “X” หมายถึง บริการใด ๆ ที่สามารถให้บริการผ่านระบบคลาวด์ เป็นแนวคิดที่ขยายมาจากรูปแบบการให้บริการคลาวด์ อีก 1 เช่น SaaS, PaaS และ IaaS โดยมุ่งเน้นการนำความคล่องตัว (Agility) ความรวดเร็วในการปรับใช้ (Deployment Speed) และการลดต้นทุนเริ่มต้น (Reduced Capital Expenditure) มาใช้กับบริการด้านเทคโนโลยีสารสนเทศอีก 1 ที่หลากหลายยิ่งขึ้น

บริการในรูปแบบ XaaS อาจครอบคลุมตั้งแต่อีเมล ระบบปฏิบัติการเดสก์ท็อป การเข้าถึงจากระยะไกล ไปจนถึงบริการด้านความปลอดภัยทางไซเบอร์ องค์กรต่าง ๆ โดยเฉพาะบริษัทเทคโนโลยีที่มีแนวคิดสร้างสรรค์ ได้ปรับตัวและพัฒนาบริการในรูปแบบนี้เพื่อตอบโจทย์ความต้องการของผู้ใช้งานในยุคดิจิทัล

สิ่งที่ทำให้ XaaS โดยเด่นคือการจัดโครงสร้างการให้บริการในรูปแบบ “จ่ายตามการใช้งานจริง” (Pay-as-you-go) ซึ่งทำให้องค์กรสามารถลดต้นทุนการลงทุนล่วงหน้า (CapEx) และเพิ่มความยืดหยุ่นในการปรับขนาดระบบให้เหมาะสมกับการเปลี่ยนแปลงของธุรกิจได้อย่างมีประสิทธิภาพ

ตัวอย่างบริการ XaaS ที่พบในชีวิตประจำวัน

- **Netflix:** บริการสตรีมมิ่งสื่อบันเทิงในรูปแบบสมัครสมาชิก
- **Dropbox:** บริการพื้นที่จัดเก็บข้อมูลบนคลาวด์
- **Uber:** บริการร่วมเดินทางผ่านแอปพลิเคชันที่ปรับใช้โมเดล XaaS กับอุตสาหกรรมขนส่ง

1.8 ผู้ให้บริการระบบคลาวด์รายใหญ่ที่สุด (Largest Cloud Service Providers)

ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers: CSPs) คือองค์กรที่ให้บริการทรัพยากรคอมพิวเตอร์ผ่านระบบคลาวด์ในรูปแบบการสมัครใช้งาน (Subscription-Based) โดยทรัพยากรที่ให้บริการอาจอยู่ในระดับโครงสร้างพื้นฐาน เช่น ฮาร์ดแวร์และเครื่องเสมือนในรูปแบบ Infrastructure as a Service (IaaS) หรืออยู่ในรูปแบบของซอฟต์แวร์สำเร็จรูปที่ให้บริการผ่านระบบคลาวด์ เช่น Microsoft Office 365 ซึ่งอยู่ในกลุ่ม Software as a Service (SaaS)

Software as a Service (SaaS)	Hosted Applications			
Infrastructure Software				
Operating Systems				
Virtualization				
Servers and Server Hardware				
Networking				
Data Center Electrical and Mechanical Operations				

SaaS provides hosted applications and all the corresponding software and hardware infrastructure related to using that application. (Image © 123RF.com.)

ภาพที่ 3 Software as a Service (SaaS)

ในปัจจุบัน ผู้ให้บริการหลัก 3 รายที่ครองส่วนแบ่งตลาดของบริการระบบคลาวด์ในระดับโลก ได้แก่ Amazon Web Services (AWS), Microsoft Azure และ Google Cloud Platform (GCP) ทั้งสามรายให้บริการที่หลากหลาย มีโครงสร้างพื้นฐานขนาดใหญ่ทั่วโลก และแข่งขันกันโดยตรงในตลาด ทำให้มีบริการที่คล้ายคลึงกัน แม้จะใช้ชื่อเรียกที่แตกต่างกัน

1.8.1 Amazon Web Services (AWS)

Amazon Web Services หรือ AWS เป็นผู้ให้บริการระบบคลาวด์รายใหญ่ของโลก โดยมีบริการคลาวด์มากกว่า 200 รายการ และดำเนินงานบนโครงสร้างพื้นฐานระดับโลกที่ประกอบด้วยมากกว่า 77 โซน

ให้บริการ (Availability Zones) ในกว่า 24 ภูมิภาค (Regions) ทั่วโลก จุดเด่นของ AWS คือสามารถเริ่มต้นใช้งานได้อย่างรวดเร็ว พร้อมต้นทุนเริ่มต้นต่ำ และรองรับการขยายตัวของระบบอย่างยืดหยุ่น

AWS พัฒนาขึ้นโดยมีโครงสร้างหลักอยู่บนระบบปฏิบัติการ Linux และมีบริการสำคัญที่ได้รับความนิยม ได้แก่

- **S3 (Simple Storage Service):** บริการพื้นที่จัดเก็บข้อมูลบนคลาวด์ (Storage as a Service: STaaS)
- **EC2 (Elastic Compute Cloud):** บริการโครงสร้างพื้นฐานเป็นบริการ (IaaS)
- **Lambda:** บริการฟังก์ชันแบบไร้เซิร์ฟเวอร์ (Serverless Computing)
- **Glacier:** บริการจัดเก็บข้อมูลระยะยาวและระบบจัดเก็บคลาวด์
- **SNS (Simple Notification Service):** บริการส่งข้อความจากผู้เผยแพร่ไปยังผู้ติดตาม
- **CloudFront:** บริการสำหรับเว็บไซต์แบบ@dynamic และการส่งเนื้อหา

1.8.2 Microsoft Azure

Microsoft Azure เป็นอีกหนึ่งผู้ให้บริการระบบคลาวด์ชั้นนำที่มีบริการมากกว่า 200 รายการ และมีศูนย์ข้อมูลทั่วโลกแบ่งเป็นภูมิภาคและโซนให้บริการเช่นเดียวกับ AWS จุดเด่นของ Azure คือการรวมระบบกับผลิตภัณฑ์ของ Microsoft ได้อย่างราบรื่น การเริ่มต้นใช้งานที่รวดเร็ว และต้นทุนการลงทุนเบื้องต้นที่ต่ำ Azure รองรับทั้งระบบปฏิบัติการ Windows และ Linux

บริการสำคัญของ Microsoft Azure ได้แก่

- **Azure Virtual Machines:** บริการ IaaS
- **Azure Disk Storage:** บริการ STaaS
- **Azure Visual Studio:** เครื่องมือสำหรับการพัฒนาแอปพลิเคชัน
- **Azure Functions:** บริการแบบไร้เซิร์ฟเวอร์ (Serverless Computing)
- **Azure Backup:** บริการสำรองและกู้คืนข้อมูล
- **Azure SQL:** บริการฐานข้อมูลเชิงสัมพันธ์
- **Azure Cosmos DB:** บริการฐานข้อมูล NoSQL
- **Microsoft Entra ID (Azure Active Directory):** บริการจัดการตัวตนและสิทธิ์การเข้าถึง (Identity and Access Management: IAM)

1.8.3 Google Cloud Platform (GCP)

Google Cloud Platform หรือ GCP เป็นผู้ให้บริการคลาวด์ที่มีบริการหลากหลายเช่นเดียวกับ AWS และ Azure โดยมีโครงสร้างพื้นฐานระดับโลกที่ประกอบด้วยศูนย์ข้อมูลในมากกว่า 73 โซนให้บริการ และ

24 ກົມງານ ພຣ້ອມທີ່ມີແນວທາງການໃຫ້ບໍລິການທີ່ເນັ້ນຄວາມຮຽນເຮົາໃນການເຮັດໃຈໜັງແລະຕັ້ນທຸນເຮັ່ມຕົ້ນທີ່ຕໍ່າ GCP ໃຫ້ບໍລິການທີ່ດ້ານການປະມາລັດ ການຈັດເກີບຂໍ້ມູນ ແລະຮູ້ານຂໍ້ມູນ
ບໍລິການເດັ່ນຂອງ GCP ໄດ້ແກ່

- **Cloud Storage:** ບໍລິການ STaaS
- **Compute Engine:** ບໍລິການ IaaS
- **App Engine:** ແພລົມສໍາຮັບການພັນນາແອປພລິເຂັ້ນ (PaaS)
- **Cloud SQL:** ບໍລິການຮູ້ານຂໍ້ມູນເຊີງສັນພັນ
- **Firestore:** ບໍລິການຮູ້ານຂໍ້ມູນ NoSQL
- **BigQuery:** ບໍລິການປະມາລັດຂໍ້ມູນຂາດໃໝ່ (Big Data)

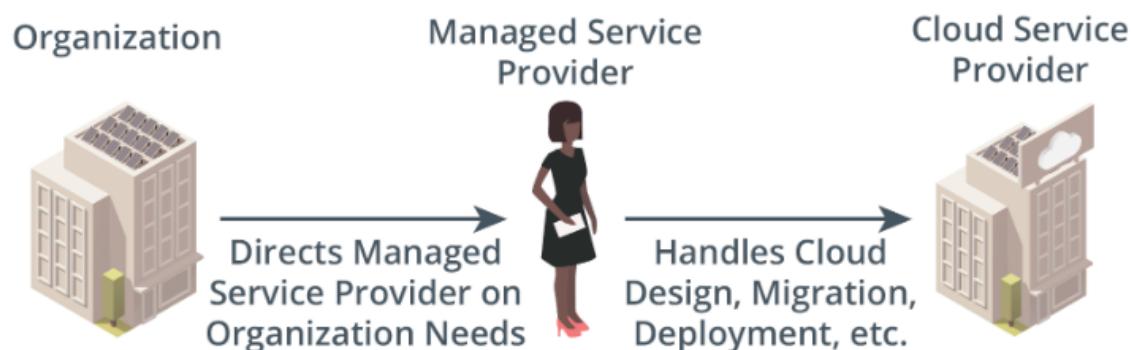
1.9 ບຫບາຫຂອງຜູ້ໃຫ້ບໍລິການຄລາວດໍແບບມີການຈັດການ (Role of Cloud Managed Service Providers: MSPs)

ການດຳເນີນການດ້ານຮະບບຄລາວດໍກາຍໃນອົງກົງ ໂດຍແນພະໃນອົງກົງຮັນາດເລັກແລະຂນາດກລາງ ອາຈ ປະສົບຄວາມທ້າທາຍໃນການຈັດຫາບຸຄລາກທີ່ມີຄວາມເຊື່ອວ່າງຈຸດຮູ້ອົງກົງຮັນາດທຸກໆດ້ານຂອງເຖິງໂຄໂລຢືກຄລາວດໍ ຜົ່ງມີຄວາມຊັບຊັນແລະເປີ່ມີຢັນແປງຍ່າງຮຽນເຮົາ

ເພື່ອແກ້ໄຂປ່ຽນທາດັກລ່າວ ອົງກົງສາມາຮັດເລືອກໃຫ້ບໍລິການຈາກຜູ້ໃຫ້ບໍລິການກາຍນອກທີ່ເວີກວ່າ ຜູ້ໃຫ້ບໍລິການຮະບບຄລາວດໍແບບມີການຈັດການ (Managed Service Providers: MSPs) ຜົ່ງເປັນບຸຄຄລ໌ ອົງກົງນິຕິບຸຄຄລ໌ ອົງກົງທີ່ໄໝເຂົ້າຕົ້ນກັບຜູ້ໃຫ້ບໍລິການຄລາວດໍຮາຍໃໝ່ (Cloud Service Providers: CSPs) ໂດຍອົງກົງສາມາຮັດວ່າ ຈ້າງ MSP ເຫັນນີ້ໄໝໜ່າຍດຳເນີນການດ້ານຕ່າງໆ ໃໝ່ ຂໍ້ມູນ ການອອກແບບຮະບບຄລາວດໍ ການໂຍກຍ້າຍຮະບບ (Migration) ການຕິດຕັ້ງຮະບບ (Deployment) ຕລອດຈົນການບໍລິການຈັດການແລະດູແລະຮະບບຄລາວດໍຍ່າງຕ່ອງ

ຜູ້ໃຫ້ບໍລິການຄລາວດໍຮາຍໃໝ່ບ່າງຮາຍ ເໜີ Amazon Web Services (AWS) ກໍມີບໍລິການ AWS Managed Services ຜົ່ງເປັນການໃຫ້ບໍລິການຈັດການຮະບບກາຍໃນແພລົມສໍາຮັບການ

ການໃຫ້ບໍລິການ MSP ມີແນວໂນ້ມຊ່ວຍໃຫ້ອົງກົງຮັນາດຕັ້ນທຸນໃນການດຳເນີນການ (Operating Expenditures: OpEx) ແລະເພີ່ມປະສິທິກັບໂດຍຮົມຂອງຮະບບໄດ້ຍ່າງຊັດເຈນ



Using an MSP outsources many business tasks. (Images © 123RF.com.)

ກາພທີ່ 4 Role of Cloud Managed Service Providers

บริการที่ MSP มักให้บริการประกอบด้วย:

- การจัดทำรายงานและตรวจสอบสถานะระบบ (Reporting and Monitoring)
- กระบวนการสำรองและกู้คืนข้อมูล (Backup and Recovery)
- การทดสอบประสิทธิภาพของระบบ (Performance Testing)

ประโยชน์ที่องค์กรจะได้รับจาก MSP ได้แก่:

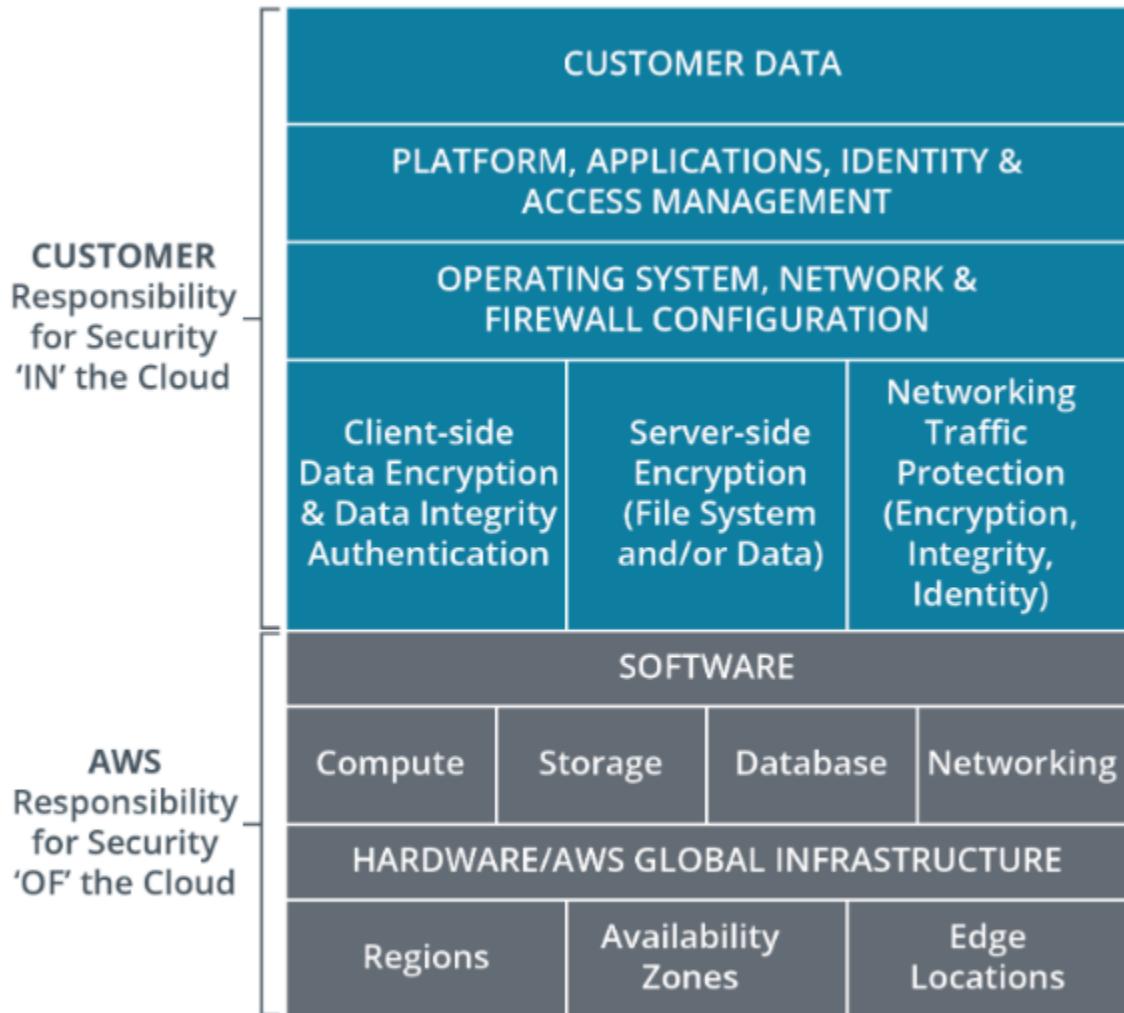
- ได้รับความเชี่ยวชาญในระดับที่อาจเกินกว่าศักยภาพของทีมภายในองค์กร เช่น ด้านความมั่นคงปลอดภัยไซเบอร์ การกู้คืนระบบจากภัยพิบัติ (Disaster Recovery) การปฏิบัติตามกฎหมายและมาตรฐาน (Compliance) และการนำเทคโนโลยีใหม่มาใช้
- ทีมงานด้าน IT และนักพัฒนาภายในองค์กรสามารถมุ่งเน้นไปที่โครงการอื่น ๆ ที่สำคัญกว่า โดยไม่ต้องแบกรับภาระงานประจำด้านการจัดการระบบคลาวด์

1.10 โมเดลความรับผิดชอบร่วมกัน (Shared Responsibility Model)

ในระบบคลาวด์ การรักษาความมั่นคงปลอดภัยไม่ได้เป็นหน้าที่ของฝ่ายใดฝ่ายหนึ่งโดยสมบูรณ์ แต่เป็นรูปแบบของ ความรับผิดชอบร่วมกัน (Shared Responsibility Model) ระหว่าง ผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) และ ผู้ใช้งานระบบคลาวด์ (Cloud Consumer) ซึ่งแต่ละฝ่ายจะมีบทบาทหน้าที่เฉพาะในด้านการรักษาความปลอดภัยของระบบ

โดยทั่วไป ผู้ให้บริการคลาวด์จะรับผิดชอบด้านความมั่นคงปลอดภัยทางกายภาพของศูนย์ข้อมูล เช่น การควบคุมการเข้าถึงทางกายภาพของอาคาร การรักษาความปลอดภัยของเครือข่ายโครงสร้างพื้นฐาน และการแยกข้อมูลของผู้ใช้แต่ละรายไม่ให้ปะปนกัน

ในขณะที่ผู้ใช้งานระบบคลาวด์จะต้องรับผิดชอบในส่วนของ การรักษาความปลอดภัยของข้อมูล ตนเอง รวมถึงการควบคุมการเข้าถึงของผู้ใช้ การกำหนดสิทธิ์ใช้งานข้อมูล และการเข้ารหัสข้อมูล (Encryption) เพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต



The shared security model divides responsibility between the CSP and the consumer.

ภาพที่ 5 Shared Responsibility Model

หน่วยที่ 2 การทำความเข้าใจคำศัพท์พื้นฐานเกี่ยวกับระบบคลาวด์ (Recognize Cloud Terms)

ในการนำระบบคลาวด์มาใช้งาน ไม่ว่าจะเลือกใช้ผู้ให้บริการรายใด องค์กรจำเป็นต้องเข้าใจคำศัพท์ และแนวคิดพื้นฐานบางประการที่เกี่ยวข้องกับกระบวนการให้บริการ เช่น การสมัครใช้งาน (Subscription), การจัดสรรทรัพยากร (Provisioning), เครื่องเสมือน (Virtual Machines), คอนเทนเนอร์ (Containers) และ การปรับขนาดระบบ (Scaling)

บทเรียนนี้จะอธิบายคำศัพท์สำคัญเหล่านี้ รวมถึงคำอื่น ๆ ที่เกี่ยวข้อง เพื่อช่วยให้ผู้เรียนสามารถเข้าใจระบบคลาวด์ในบริบทของการออกแบบและใช้งานจริงได้อย่างมีประสิทธิภาพ พร้อมทั้งเตรียมความพร้อมสำหรับการเรียนรู้เทคโนโลยีคลาวด์ที่กำลังพัฒนาอย่างต่อเนื่องในยุคปัจจุบัน

2.1 บริการแบบสมัครสมาชิก (Subscription Services)

โมเดลการชำระเงินแบบสมัครสมาชิก เป็นรูปแบบการเรียกเก็บค่าบริการแบบต่อเนื่องตามรอบระยะเวลาที่กำหนด เช่น รายเดือนหรือรายปี โดยมักไม่มีข้อผูกพันในระยะยาว ผู้ใช้งานสามารถรีเซ็ตต้นใช้งานบริการได้ทันทีที่สมัครสมาชิก และโดยทั่วไปสามารถยกเลิกบริการได้ทุกเมื่อตามความต้องการ

ค่าบริการในรูปแบบนี้มักจะมีโครงสร้างราคาที่จุใจให้เลือกสมัครระยะยาว เนื่องจากการสมัครใช้งานในระยะยาว เช่น รายปี มักมีราคาถูกกว่าการสมัครใช้งานแบบระยะสั้น เช่น รายเดือน

ตัวอย่างที่เห็นได้ชัดของบริการแบบสมัครสมาชิกคือ Microsoft Office 365 ซึ่งเปิดให้ผู้ใช้เลือกสมัครใช้บริการในรูปแบบรายเดือนหรือรายปี โดยผู้ใช้งานสามารถเข้าถึงซอฟต์แวร์และฟีเจอร์ต่าง ๆ ได้ตลอดระยะเวลาของการเป็นสมาชิก

2.2 การจัดการตัวตนผู้ใช้งาน (Identity Management)

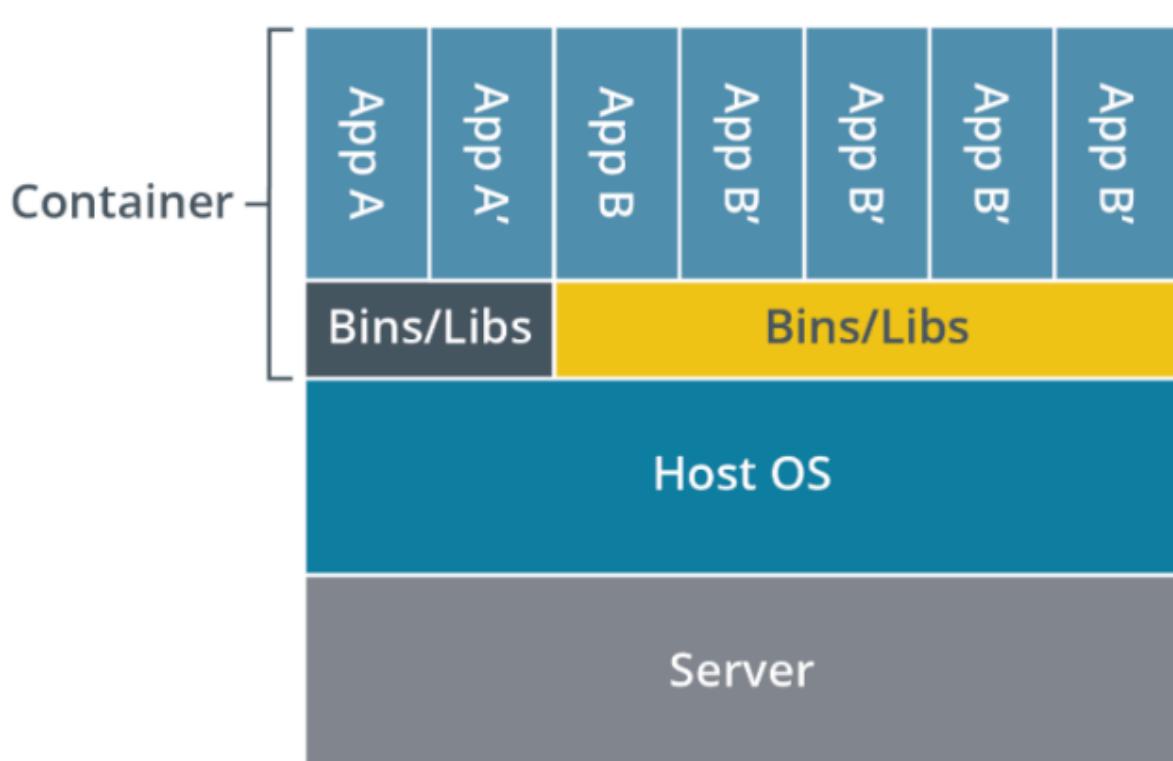
การจัดการตัวตนผู้ใช้งาน (Identity Management) คือกระบวนการในการระบุตัวตนของผู้ใช้ และควบคุมการเข้าถึงทรัพยากรภายในระบบ โดยทั่วไป ระบบจะมีการสร้างบัญชีผู้ใช้ (User Account) สำหรับแต่ละบุคคล และทำการกำหนดสิทธิ์ในการเข้าถึง (Access Rights) รวมถึงข้อจำกัดต่าง ๆ ผ่านระบบควบคุมการเข้าถึง (Access Control) เช่น ระบบสิทธิ์การใช้งาน (Permissions)

เพื่อให้การบริหารจัดการเป็นไปอย่างมีประสิทธิภาพ บัญชีผู้ใช้อาจถูกจัดกลุ่ม (User Groups) ตามบทบาทหน้าที่หรือแผนก ซึ่งช่วยให้การกำหนดสิทธิ์สามารถทำได้เป็นระบบมากขึ้น อย่างไรก็ตาม สิ่งสำคัญคือกระบวนการทั้งหมดจะต้องชัดเจนว่าผู้ใช้แต่ละรายสามารถเข้าถึงทรัพยากรใดได้บ้าง และในระดับใดเพื่อรักษาความปลอดภัยและป้องกันการเข้าถึงข้อมูลหรือระบบโดยไม่ได้รับอนุญาต

2.3 ทบทวนแนวคิดเกี่ยวกับเวอร์ชวลไอลเซชัน (Virtualization Review)

เวอร์ชวลไอลเซชัน (Virtualization) ถือเป็นแนวคิดพื้นฐานที่สำคัญของระบบคลาวด์ โดยเป็นเทคโนโลยีที่ช่วยแยกระบบปฏิบัติการ บริการ และแอปพลิเคชัน ออกจากข้อจำกัดของฮาร์ดแวร์จริง ทำให้สามารถนำไปใช้งานบนแพลตฟอร์มที่จัดสรรทรัพยากรด้านการประมวลผล พื้นที่จัดเก็บข้อมูล และเครือข่ายได้อย่างยืดหยุ่น ระบบคลาวด์จะไม่สามารถเกิดขึ้นได้หากปราศจากเทคโนโลยีเวอร์ชวลไอลเซชันนี้

ปัจจุบันมีโซลูชันด้านเวอร์ชวลไอลเซชันที่หลากหลาย ไม่ว่าจะเป็น เครื่องเสมือน (Virtual Machines: VMs), คอนเทนเนอร์ (Containers) และ เครือข่ายเสมือน (Virtual Networks) ซึ่งต่างก็มีบทบาทเฉพาะในการออกแบบระบบคลาวด์



Containers for each application, sharing bins, libraries, and a single OS.

ภาพที่ 6 Virtualization Review

2.3.1 เครื่องเสมือน (Virtual Machines: VMs)

เทคโนโลยีเวอร์ชัลไลเซชันในระดับฮาร์ดแวร์ช่วยให้สามารถจัดสรรทรัพยากรให้กับเครื่องเสมือนหนึ่งเครื่องหรือหลายเครื่องได้อย่างอิสระ โดยในแต่ละ VM จะสามารถติดตั้งระบบปฏิบัติการและแอปพลิเคชันได้เหมือนกับเครื่องจริง อีกทั้งยังสามารถทำงานบนเครือข่ายในฐานะโหนด (Node) ปกติที่ให้บริการต่าง ๆ เช่น ฐานข้อมูล ระบบพิสูจน์ตัวตน หรือระบบจัดเก็บข้อมูล

เครื่องเสมือนมีความสามารถในการเข้าถึงทรัพยากรของฮาร์ดแวร์ได้มากกว่า และสามารถกำหนดโครงสร้างให้มีความซ้ำซ้อน (Redundancy) เพื่อเพิ่มความพร้อมใช้งานสูง (High Availability) ได้ จึงเป็นองค์ประกอบสำคัญของบริการคลาวด์ในรูปแบบโครงสร้างพื้นฐานเป็นบริการ (Infrastructure as a Service: IaaS) ตัวอย่างเช่น Amazon Web Services (AWS) Elastic Compute Cloud (EC2) และ Microsoft Azure Virtual Machines

2.3.2 คอนเทนเนอร์ (Containers)

คอนเทนเนอร์ (Containerization) เป็นรูปแบบหนึ่งของเวอร์ชัลไลเซชันเข่นกัน แต่มีความแตกต่างจาก VMs อย่างมีนัยสำคัญ โดยคอนเทนเนอร์จะทำงานที่ระดับของระบบปฏิบัติการ (OS Layer) แทนที่จะจำลองทั้งฮาร์ดแวร์เหมือน VM

แต่ละคอนเทนเนอร์จะบรรจุเพียงแอปพลิเคชันเดียว พร้อมกับไลบรารีและส่วนประกอบที่จำเป็นสำหรับการทำงาน ซึ่งการมุ่งเน้นเฉพาะเจาะจงในฟังก์ชันเดียวทำให้คอนเทนเนอร์เหมาะสมอย่างยิ่งกับสถาปัตยกรรมแบบไมโครเซอร์วิส (Microservices)

คอนเทนเนอร์มีน้ำหนักเบา (Lightweight) ใช้ทรัพยากรน้อย และมักใช้ระบบปฏิบัติการเดียวกัน (โดยทั่วไปคือ Linux) ซึ่งช่วยให้สามารถใช้งานได้อย่างมีประสิทธิภาพสูงในสภาพแวดล้อมคลาวด์ ผู้ให้บริการระบบคลาวด์รายใหญ่ เช่น Google Cloud Platform (GCP), Microsoft Azure, และ Amazon Web Services (AWS) ต่างก็มีบริการจัดการคอนเทนเนอร์ในรูปแบบคลาวด์ เช่น Google Kubernetes Engine (GKE), Azure Kubernetes Service (AKS), และ Amazon Elastic Kubernetes Service (EKS)

2.3.3 เครือข่ายเสมือน (Virtual Networking)

เวอร์ชัลไลเซชันในระบบคลาวด์ไม่ได้จำกัดอยู่แค่ในระดับของเครื่องคอมพิวเตอร์หรือเซิร์ฟเวอร์เท่านั้น แต่ยังรวมถึงฟังก์ชันด้านเครือข่ายด้วย แนวคิดนี้ทำให้ผู้ดูแลระบบสามารถสร้างและจัดการองค์ประกอบของเครือข่ายในรูปแบบเสมือนได้ เช่น การสร้างซับเน็ตเสมือน การกำหนดค่าเราเตอร์หรือสวิตช์เสมือน ไปจนถึงการออกแบบระบบเครือข่ายคลาวด์เสมือนภายในโครงสร้างพื้นฐานของคลาวด์เอง

การใช้เครือข่ายเสมือนในระบบคลาวด์ช่วยให้สามารถจัดการระบบเครือข่ายได้อย่างยืดหยุ่น ปรับเปลี่ยนได้ตามความต้องการ และลดข้อจำกัดที่มักพบในระบบเครือข่ายแบบดั้งเดิมที่ต้องพึ่งพาอุปกรณ์ฮาร์ดแวร์จริง

2.3.4 แม่แบบ (Templates)

ในกระบวนการสร้างเครื่องเสมือน (Virtual Machine) โดยเฉพาะในระบบคลาวด์ การใช้แม่แบบ (Template) ถือเป็นวิธีที่มีประสิทธิภาพในการลดความซับซ้อน ลดข้อผิดพลาด และควบคุมค่าใช้จ่าย ที่อาจเกิดจากการกำหนดค่าด้วยตนเอง แม่แบบคือการกำหนดค่าล่วงหน้าสำหรับเครื่องเสมือนที่มีการออกแบบมาอย่างเหมาะสม ทั้งในด้านของหน่วยประมวลผล พื้นที่จัดเก็บ ระบบปฏิบัติการ ซอฟต์แวร์ที่จำเป็น และการตั้งค่าด้านความปลอดภัย

ตัวอย่างเช่น ในระบบคลาวด์ส่วนตัวขององค์กร อาจมีนโยบายเปิดให้ทีมพัฒนาสามารถสร้างเครื่องเสมือนได้เองตามแนวคิดคลาวด์แบบบริการตนเอง หากปล่อยให้กำหนดค่าด้วยตนเองทั้งหมด อาจเกิดข้อผิดพลาดที่ส่งผลกระทบต่อความปลอดภัยหรือต้นทุน แต่หากมีการจัดเตรียมแม่แบบที่เหมาะสมไว้ล่วงหน้า นักพัฒนา ก็เพียงเลือกแม่แบบที่ตรงกับความต้องการ ระบบก็จะสร้างเครื่องเสมือนจากการตั้งค่าที่กำหนดไว้โดยอัตโนมัติ

ผู้ให้บริการระบบคลาวด์รายใหญ่ เช่น AWS, Azure และ GCP ที่ใช้แนวทางเดียวกัน โดยจัดเตรียมแม่แบบไว้สำหรับลูกค้า เพื่อให้สามารถเลือกใช้งานเครื่องเสมือนที่มีการทำงานค่ามาตรฐานได้อย่างสะดวกยืดหยุ่น และสอดคล้องกับแนวทางการใช้งานที่ปลอดภัยและมีประสิทธิภาพ

2.4 การปรับใช้ทรัพยากรระบบคลาวด์ (Deploy Cloud Resources)

การจัดเตรียมทรัพยากรระบบคลาวด์สามารถดำเนินการได้ตามความต้องการ (On-Demand Provisioning) และยังสามารถปรับขนาดของทรัพยากรให้เพิ่มขึ้นหรือลดลงตามความต้องการได้ในกระบวนการที่เรียกว่า การปรับขนาดอัตโนมัติ (Auto-Scaling) ผู้ดูแลระบบมักจะรวมทรัพยากรที่มีลักษณะเป็นเชิงตระกูล เช่น หน่วยประมวลผล พื้นที่จัดเก็บข้อมูล และการเชื่อมต่อเครือข่าย ให้เป็นหน่วยเดียวกันในกระบวนการที่เรียกว่า การบูรณาการขั้นสูง (Hyperconvergence) เพื่อให้การจัดการง่ายขึ้น กระบวนการอัตโนมัติยังสามารถยืนยันการปรับใช้งานเหล่านี้ได้ผ่านการตรวจสอบหลังการปรับใช้ (Post-Deployment Validation)

2.4.1 การจัดเตรียมทรัพยากร (Provisioning)

การจัดเตรียมทรัพยากร (Provisioning) เป็นหนึ่งในขั้นตอนสำคัญของการปรับใช้บริการระบบคลาวด์ โดยหมายถึงการจัดสรรทรัพยากรระบบคลาวด์ในโครงสร้างพื้นฐานขององค์กร โดยกระบวนการนี้จะดำเนินการตามวัตถุประสงค์ นโยบาย และกระบวนการที่กำหนดไว้ล่วงหน้า

การจัดเตรียมทรัพยากรสามารถดำเนินการผ่านอินเทอร์เฟซบนเว็บหรือผ่านบรรทัดคำสั่ง (CLI) ซึ่งโดยทั่วไปแล้วจะอยู่ในรูปแบบการบริการตนเอง (Self-Service) ตามลักษณะเฉพาะของระบบคลาวด์ที่ระบุไว้โดยสถาบันมาตรฐานและเทคโนโลยีแห่งชาติ (NIST)

ในกระบวนการปรับใช้งานทั้งหมด การจัดเตรียมทรัพยากรจะเกิดขึ้น ก่อน การกำหนดค่าของเซิร์ฟเวอร์ บริการ ผู้ใช้ หรือเครือข่าย และมักจะมีการกำหนดการควบคุมการเข้าถึงเป็นส่วนหนึ่งของขั้นตอนนี้ด้วย

2.4.2 การปรับขนาดอัตโนมัติ (Auto-Scaling)

การปรับขนาดอัตโนมัติเป็นการใช้ประโยชน์จากการปรับใช้งานแบบอัตโนมัติและการจำลองเสมือน (Virtualization) เพื่อให้ทรัพยากรตรงกับความต้องการใช้งานในแต่ละช่วงเวลา องค์กรจะจ่ายค่าบริการเฉพาะทรัพยากรที่ใช้งานจริง ซึ่งช่วยในการควบคุมค่าใช้จ่าย โดยเฉพาะอย่างยิ่งเมื่อปริมาณการใช้งานมีลักษณะเปลี่ยนแปลงตามฤดูกาลหรือไม่สามารถคาดการณ์ได้ล่วงหน้า

การปรับขนาดอัตโนมัติสามารถเกิดได้สองรูปแบบคือ

- การปรับขนาดขึ้น (Scale Up): เพิ่มขนาดของทรัพยากรในเซิร์ฟเวอร์เดียว เช่น เพิ่ม RAM หรือ CPU
- การปรับขนาดออก (Scale Out): เพิ่มจำนวนของเซิร์ฟเวอร์เสมือนเพื่อกระจายภาระงาน

เมื่อความต้องการลดลง ระบบจะลดขนาดของทรัพยากรลงเพื่อประหยัดค่าใช้จ่าย

2.4.3 การบูรณาการขั้นสูง (Hyperconverged)

โซลูชันระบบคลาวด์แบบบูรณาการขั้นสูง (*Hyperconverged Cloud Solutions*) คือ การรวมทรัพยากรคอมพิวเตอร์ พื้นที่จัดเก็บข้อมูล และเครือข่าย เข้าด้วยกันเป็นองค์ประกอบเดียว โดยมีเป้าหมายเพื่อลดความซับซ้อนในการจัดการและเพิ่มความสามารถในการขยายระบบ

ระบบแบบ *Hyperconverged* จะจัดการทรัพยากรเหล่านี้เป็นหน่วยรวมกัน ในขณะที่ระบบแบบ *Converged* ยังคงสามารถแยกองค์ประกอบออกจากกันได้ ซึ่งจะทำให้การจัดการมีความซับซ้อนมากกว่า เพราะต้องดูแลแยกกันในแต่ละด้าน (คอมพิวเตอร์ / จัดเก็บข้อมูล / เครือข่าย)

2.4.4 การตรวจสอบหลังการปรับใช้ (Post-Deployment Validation)

การตรวจสอบหลังการปรับใช้เป็นกระบวนการเพื่อยืนยันว่าบริการหรือแอปพลิเคชันที่ปรับใช้ไปแล้วสามารถให้บริการได้ตามระดับที่กำหนดไว้ (Service Level Agreements – SLA) กระบวนการนี้อาจรวมถึงการทดสอบย้อนกลับ (Regression Testing) หรือการทดสอบการทำงานของระบบ (Functionality Testing)

เพื่อความมีประสิทธิภาพและความสม่ำเสมอ การดำเนินการตรวจสอบหลังการปรับใช้ควรเป็นแบบอัตโนมัติให้มากที่สุด

2.5 แอปพลิเคชันบนระบบคลาวด์ (Cloud Applications)

แอปพลิเคชันบนระบบคลาวด์ เป็นซอฟต์แวร์ที่ถูกติดตั้งและประมวลผลบนระบบคลาวด์ แทนที่จะทำงานอยู่บนเครื่องคอมพิวเตอร์หรือเซิร์ฟเวอร์ของผู้ใช้โดยตรง ผู้ใช้งานสามารถเข้าถึงแอปพลิเคชันเหล่านี้ผ่านเครือข่ายอินเทอร์เน็ต ไม่ว่าจะเป็นคลาวด์แบบส่วนตัวหรือสาธารณะ

ข้อได้เปรียบสำคัญของแอปพลิเคชันบนระบบคลาวด์ คือ ผู้ใช้จะได้รับประสบการณ์การใช้งานที่สอดคล้องกัน ไม่ว่าจะใช้งานผ่านระบบปฏิบัติการที่แตกต่างกัน เช่น Windows, Linux, macOS หรืออุปกรณ์เคลื่อนที่ เช่น Android และ iOS

หน่วยที่ 3 เทคโนโลยีในระบบคลาวด์ (Technologies in the Cloud)

ในปัจจุบัน เทคโนโลยี ปัญญาประดิษฐ์ (Artificial Intelligence: AI) และ การเรียนรู้ของเครื่อง (Machine Learning: ML) มีบทบาทสำคัญในภาคธุรกิจระดับองค์กร และทั้งสองเทคโนโลยีนี้มีการพึ่งพาโครงสร้างพื้นฐานของระบบคลาวด์อย่างมาก

บทเรียนนี้จะกล่าวถึงแนวคิดพื้นฐานของ AI และ ML รวมถึงการประยุกต์ใช้เทคโนโลยีเหล่านี้ร่วมกับบริการคลาวด์ ซึ่งมีจุดมุ่งหมายเพื่อเพิ่มประสิทธิภาพการดำเนินงาน การตัดสินใจโดยใช้ข้อมูล และการสร้างนวัตกรรมในระดับองค์กร

3.1 ปัญญาประดิษฐ์ (Artificial Intelligence: AI)

ปัญญาประดิษฐ์ (AI) คือศาสตร์ที่มุ่งเน้นการจำลองความสามารถของมนุษย์ในการใช้สติปัญญา โดยการประมวลผลข้อมูลทั้งที่มีโครงสร้าง กึ่งมีโครงสร้าง และไม่มีโครงสร้าง เพื่อนำมาใช้ในการแก้ไขปัญหา ที่ซับซ้อน AI อาศัยชุดของกฎเกณฑ์ในการจัดการและวิเคราะห์ข้อมูล และถือเป็นศาสตร์ที่กว้างกว่า องค์ประกอบอย่างของมันคือการเรียนรู้ของเครื่อง (Machine Learning: ML)

AI ได้รับความนิยมและมีการพัฒนาอย่างต่อเนื่อง โดยหลายอุตสาหกรรมต่างนำ AI เข้ามาร่วมใช้ใน กระบวนการทำงานประจำวัน ตัวอย่างประเภทและการประยุกต์ใช้ AI ได้แก่

- การรู้จำข้อความ (Text Recognition): ตรวจจับและดึงข้อมูลจากข้อความหรือภาพเพื่อนำไป วิเคราะห์หรือจัดเก็บ
- การแปลข้อความ (Text Translation): วิเคราะห์และแปลข้อความจำนวนมาก เช่น หนังสือ บทความ เอกสาร ฯลฯ ระหว่างภาษาต่างๆ เรียกอีกอย่างว่า Machine Translation
- การแปลงเสียงเป็นข้อความ และข้อความเป็นเสียง (Voice-to-Text/Text-to-Voice): แปลงข้อมูล ระหว่างเสียงพูดและข้อความ โดยมักผ่านกับระบบแปลงภาษา
- การรู้จำภาพ (Visual Recognition): วิเคราะห์ภาพหรือวิดีโอเพื่อรับและจัดประเภทวัตถุหรือบุคคล
- การวิเคราะห์อารมณ์ (Sentiment Analysis): หรือที่รู้จักกันว่า Opinion Mining เป็นการประมวลผล ภาษาหรือข้อความเพื่อหาความคิดเห็น ความรู้สึก เช่น ระดับการสนับสนุนแบรนด์หรือความนิยม สินค้า
- ปัญญาประดิษฐ์เชิงสร้างสรรค์ (Generative AI): สร้างเนื้อหาต่างๆ เช่น ข้อความ โค้ดคอมพิวเตอร์ หรือภาพจากคำสั่ง (prompt) โดยมักใช้การเรียนรู้เชิงลึก (Deep Learning: DL)

Generative AI นับเป็นหนึ่งในเทคโนโลยีที่ได้รับความนิยมอย่างสูงในหลายภาคอุตสาหกรรม เช่น การสร้างโค้ดอัตโนมัติ การค้นคว้าเนื้อหา การสร้างสื่อ และแซทบอท โดยสามารถสร้างเพลง เขียนเรื่องราว หรือสร้างภาพได้หากมีข้อมูลเพียงพอ ซึ่งการประมวลผลข้อมูลและการฝึกโมเดล AI ส่วนใหญ่อาศัยระบบคลาวด์เป็นหลัก

การทำงานของ Generative AI แบ่งออกได้เป็น 2 ขั้นตอนหลัก:

- Training (การฝึกฝน): ใช้ชุดข้อมูลขนาดใหญ่จากแหล่งต่างๆ เช่น หนังสือ ฐานข้อมูล หรือภาพ เพื่อฝึกโมเดลให้เรียนรู้และนำไปใช้ในขั้น inferencing
- Inferencing (การประมวลผลหรือใช้งานจริง): เป็นขั้นตอนที่โมเดลที่ผ่านการฝึกแล้วจะถูกนำมาใช้ สร้างเนื้อหาใหม่

โมเดลภาษาขนาดใหญ่ (Large Language Models: LLMs) คือ คลังความรู้ที่ผ่านการฝึกมาแล้ว โดยใช้เทคนิคการเรียนรู้เชิงลึก มีความเชี่ยวชาญในเรื่องของภาษา การวิเคราะห์ข้อความ และการโต้ตอบแบบมนุษย์ ตัวอย่างของ LLM ได้แก่

- GPT (Generative Pre-trained Transformer) โดย OpenAI ซึ่งเป็นพื้นฐานของ ChatGPT
- Microsoft Turing Natural Language Generation (NLG)
- Google PaLM (Pathways Language Model)

เทคโนโลยีคลาวด์สามารถสนับสนุน AI ได้อย่างมีประสิทธิภาพ ผู้ให้บริการคลาวด์รายใหญ่ เช่น Microsoft, AWS และ Google ได้พัฒนาและนำเสนอผลิตภัณฑ์ AI ที่หลากหลาย ตัวอย่างบริการ AI บนคลาวด์ ได้แก่

Microsoft Azure:

- Azure CycleCloud: จัดเตรียมและบริหารโครงสร้าง High Performance Computing (HPC)
- Azure Machine Learning: เครื่องมือเรียนรู้ของเครื่องที่ใช้ AI
- Azure AI Studio: สภาพแวดล้อมสำหรับพัฒนา Generative AI

Amazon Web Services (AWS):

- Amazon Q: ผู้ช่วยธุรกิจที่ใช้ AI
- Amazon Polly: บริการแปลงข้อความเป็นเสียง
- Amazon Rekognition: วิเคราะห์ข้อมูลจากภาพและวิดีโอ

Google Cloud Platform:

- Vertex AI Studio: สภาพแวดล้อมสำหรับสร้าง Generative AI
- Vertex AI Notebooks: เครื่องมือสำหรับงานด้านข้อมูลวิทยาศาสตร์
- Dialogflow: สภาพแวดล้อมสำหรับการสร้างแพทช์และระบบสนทนา

3.2 การเรียนรู้ของเครื่อง (Machine Learning: ML)

ML คือ สาขาย่อยของ AI ที่มุ่งเน้นให้ระบบสามารถเรียนรู้จากข้อมูลและประสบการณ์ เพื่อนำไปใช้ในการทำนายผลลัพธ์โดยไม่ต้องตั้งโปรแกรมอย่างชัดเจนให้ผลลัพธ์ใดๆ ระบบ ML จะเรียนรู้จากข้อมูลที่ได้รับและตัดสินใจโดยอัตโนมัติด้วยการมีปฏิสัมพันธ์จากมนุษย์ให้น้อยที่สุด ตัวอย่างการประยุกต์ใช้ ML เช่น

- การพยากรณ์การจราจร
- ระบบแนะนำสินค้า
- การวินิจฉัยทางการแพทย์
- กลยุทธ์ในการแข่งขันกีฬา

3.2.1 การเรียนรู้เชิงลึก (Deep Learning: DL)

DL คือสาขาอยู่ที่ลึกกว่าของ ML ซึ่งเลียนแบบกระบวนการเรียนรู้ของสมองมนุษย์ โดยสามารถวิเคราะห์ข้อมูลที่ไม่โครงสร้างได้อย่างแม่นยำมากขึ้น เมื่อจะใช้เวลาและข้อมูลในการฝึกมากกว่า แต่ผลลัพธ์ก็มีความแม่นยำสูงขึ้น ตัวอย่างการใช้งานของ DL ได้แก่

- เกมคอมพิวเตอร์
- การประมวลผลภาษา
- การเกษตร

3.2.2 เครื่องมืออุดนิยมในระบบคลาวด์สำหรับ ML/DL

- **Keras:** API สำหรับสร้างโครงข่ายประสาทเทียม ใช้งานร่วมกับ TensorFlow
- **TensorFlow:** ไลบรารีโอเพนซอร์สจาก Google สำหรับสร้างและจัดการ AI และ ML
- **PyTorch:** ไลบรารีโอเพนซอร์สของภาษา Python ใช้สำหรับสร้างโมเดลการเรียนรู้ของเครื่องโดยเฉพาะงานด้านภาษาธรรมชาติ

3.3 อินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT)

Internet of Things (IoT) หมายถึง การผสมผสานระหว่างอุปกรณ์อัจฉริยะและการเชื่อมต่อเครือข่ายที่ทำให้อุปกรณ์เหล่านั้นสามารถติดต่อและวิเคราะห์ข้อมูลได้อย่างมีประสิทธิภาพ อุปกรณ์ IoT อาจประกอบด้วยซอฟต์แวร์ เช่นเซอร์ และระบบกลไกที่สามารถแลกเปลี่ยนข้อมูลและคำสั่งผ่านทางอินเทอร์เน็ตหรือเครือข่ายภายใน

IoT เกิดขึ้นได้จากปัจจัยสนับสนุน ได้แก่ การเชื่อมต่อเครือข่ายที่ครอบคลุมทั่วโลก ต้นทุนของเซนเซอร์ที่ต่ำลง และแพลตฟอร์มการจัดการแบบคลาวด์ที่มีประสิทธิภาพ การใช้งานของผลิตภัณฑ์ IoT ที่พึ่งได้ทั่วไป ได้แก่

- บ้านอัจฉริยะ (Smart Homes)
- ระบบติดตามสุขภาพ (Medical Monitoring)
- การจัดการด้านเกษตรกรรม (Agriculture Management)
- การจัดการพลังงาน (Energy Management)
- การผลิตในภาคอุตสาหกรรม (Manufacturing/Industrial Production)

3.3.1 บทบาทของคลาวด์ในระบบ IoT

เทคโนโลยีคลาวด์เป็นกลไกสำคัญที่ทำให้ IoT ทำงานได้อย่างมีประสิทธิภาพ เนื่องจากอุปกรณ์ IoT สามารถสร้างข้อมูลจำนวนมากมหาศาล ซึ่งบริการคลาวด์สามารถนำไปจัดเก็บและประมวลผลเพื่อวิเคราะห์ข้อมูลได้อย่างรวดเร็ว

ระบบคลาวด์ยังช่วยรวมศูนย์ข้อมูลจากอุปกรณ์ IoT ที่กระจายอยู่ในหลายพื้นที่ทางภูมิศาสตร์ อีกทั้งคุณสมบัติต่างๆ เช่น การปรับขนาดอัตโนมัติ (scaling) และการใช้ทรัพยากร่วมกัน (resource pooling) ทำให้สามารถบริหารจัดการข้อมูลขนาดใหญ่ได้อย่างยืดหยุ่น

- อุปกรณ์ IoT ทำหน้าที่ สร้างข้อมูล (Generate Data)
- บริการคลาวด์ทำหน้าที่ ใช้ข้อมูล (Use Data)

3.3.2 องค์ประกอบของ IoT

องค์ประกอบสำคัญที่ทำให้การสื่อสารของ IoT กับระบบคลาวด์เกิดขึ้นได้ ได้แก่

- **เซนเซอร์ (Sensors):** ทำหน้าที่ตรวจจับการเปลี่ยนแปลงของสิ่งแวดล้อม เช่น ตำแหน่ง อุณหภูมิ ความชื้น หรือสารเคมี
- **เครือข่ายการสื่อสาร (Network Communication):** ใช้โปรโตคอลในการส่งข้อมูลจากอุปกรณ์ไปยังจุดเก็บหรือประมวลผล

รูปแบบของการสื่อสารแบ่งได้เป็น 3 ระดับหลัก:

1. ระดับท้องถิ่น (Local): การสื่อสารระหว่างอุปกรณ์ IoT และเซนเซอร์ในพื้นที่เดียวกัน
2. เกตเวย์ (Gateway): เป็นตัวกลางเชื่อมอุปกรณ์ IoT หลายชิ้นกับเครือข่ายอินเทอร์เน็ต
3. เซิร์ฟเวอร์ส่วนกลาง (Central Server): ทำหน้าที่รับข้อมูลและวิเคราะห์จากอุปกรณ์ IoT

อุปสรรคหรือข้อจำกัดในการสื่อสารของ IoT ประกอบด้วย:

- ความสามารถของอุปกรณ์
- ข้อจำกัดด้านพลังงาน
- ขอบเขตและสื่อในการส่งข้อมูล
- ข้อกำหนดด้านความมั่นคงปลอดภัย

ตัวอย่างของเซนเซอร์ที่ใช้ในระบบ IoT:

- ตรวจวัดคุณภาพอากาศ
- ตัวตรวจจับความเร่ง (Accelerometer)
- เซนเซอร์อุณหภูมิ
- เซนเซอร์ชีวภาพ
- เซนเซอร์น้ำ/ความชื้น

ตัวอย่างการใช้งาน: ในภาคเกษตร อาจมีการติดตั้งระบบบดน้ำอัจฉริยะที่ใช้เซนเซอร์ตรวจวัดระดับน้ำในดิน และวิเคราะห์ข้อมูลเพื่อปรับปรุงมาสน้ำที่รดให้เหมาะสมโดยอัตโนมัติ

3.3.3 โปรโตคอลในการส่งข้อมูลของ IoT

โปรโตคอลสำหรับการส่งข้อมูลใน IoT ถูกจัดเป็นชั้นตามโมเดล OSI โดยแบ่งเป็น 3 ชั้น ได้แก่:

- ชั้นการประยุกต์ (Application Layer): เชื่อมโยงกับแอปพลิเคชันที่ใช้ข้อมูลจาก IoT
- ชั้นเครือข่าย (Network Layer): รับผิดชอบการส่งข้อมูลและรักษาความปลอดภัย
- ชั้นการรับรู้/เซนเซอร์ (Perception/Sensor Layer): ทำหน้าที่รวบรวมข้อมูลจากอุปกรณ์

ตัวอย่างโปรโตคอลที่ใช้ใน IoT:

- Wi-Fi: โปรโตคอลไร้สายที่พบทั่วไป แต่ใช้พลังงานค่อนข้างมาก
- Bluetooth: ส่งข้อมูลระยะใกล้ ใช้พลังงานน้อย แต่มีข้อจำกัดด้านความปลอดภัย
- HTTP: ใช้การสื่อสารแบบ synchronous เมามากับบางกรณีของ IoT เท่านั้น
- NB-IoT (Narrow-Band IoT): โปรโตคอลไร้สายแบบใหม่ ใช้พลังงานต่ำ เมามากับการส่งข้อมูลปริมาณน้อย
- AMQP (Advanced Message Queuing Protocol): โปรโตคอลแบบ asynchronous ที่สามารถจัดคิวข้อความอย่างมีเสถียรภาพ เมามากับงานที่ต้องการความพร้อมใช้งานสูง

3.3.4 บทบาทของ Gateway และ Edge Computing

Gateway ทำหน้าที่เป็นตัวกลางในการแปลงโปรโตคอลของอุปกรณ์ IoT ให้สามารถสื่อสารกับระบบคลาวด์ได้อย่างเหมาะสม โดยเฉพาะในกรณีที่อุปกรณ์ IoT ไม่ได้ใช้โปรโตคอล IP ซึ่งเป็นพื้นฐานของการทำงานอินเทอร์เน็ตและบริการคลาวด์

Edge Computing คือแนวทางที่ให้การประมวลผล (หรือเตรียมข้อมูลก่อนประมวลผล) เกิดขึ้นใกล้กับอุปกรณ์ IoT แทนที่จะส่งข้อมูลไปยังคลาวด์โดยตรง

ประโยชน์ของ Edge Computing:

- ลดการใช้งานแบนด์วิดท์
- ลดระยะเวลาไฟล์ (Latency)
- ช่วยให้สามารถตัดสินใจแบบเรียลไทม์ได้อย่างมีประสิทธิภาพมากขึ้น

หน่วยที่ 4 การแก้ไขปัญหาในสภาพแวดล้อมระบบคลาวด์ (Troubleshoot the Cloud Environment)

หนึ่งในทักษะและหน้าที่หลักของผู้ดูแลระบบคลาวด์ (Cloud Administrator) คือ การวิเคราะห์และแก้ไขปัญหาที่เกี่ยวข้องกับบริการระบบคลาวด์และการเข้าถึงข้อมูล ซึ่งการดำเนินการแก้ไขปัญหาอย่างมีประสิทธิภาพ จำเป็นต้องอาศัย ระเบียบวิธีการแก้ไขปัญหา (troubleshooting methodology) ที่เป็นระบบ

ผู้ดูแลระบบควรทราบก่อนว่า วิธีการในการวิเคราะห์และแก้ไขปัญหาอาจแตกต่างกันไปตาม:

- สถานการณ์เฉพาะหน้า
- ระดับทักษะของผู้ดำเนินการ
- ประสบการณ์ในการทำงานกับระบบคลาวด์

ประเภทของปัญหาที่อาจต้องวิเคราะห์ในระบบคลาวด์ ได้แก่:

- ปัญหาในการปรับใช้ระบบ (Deployment Issues)
- ปัญหาเกี่ยวกับเครือข่าย (Network Problems)
- ปัญหาด้านความมั่นคงปลอดภัย (Security Concerns)

ผู้ดูแลระบบจะต้องสามารถ ประยุกต์ใช้ระเบียบวิธีการแก้ไขปัญหา อย่างเหมาะสมกับบริบทที่เกิดขึ้นจริงในสภาพแวดล้อมระบบคลาวด์ เพื่อให้สามารถค้นหาสาเหตุของปัญหาได้อย่างรวดเร็วและแม่นยำ พร้อมทั้งดำเนินการแก้ไขให้ระบบกลับมาทำงานได้ตามปกติ

4.1 ระเบียบวิธีการแก้ไขปัญหา (Troubleshooting Methodology)

กระบวนการแก้ไขปัญหาในระบบคลาวด์สามารถดำเนินการตามขั้นตอนพื้นฐานต่อไปนี้:

1. ระบุปัญหา
2. กำหนดขอบเขตของปัญหา
3. ตั้งสมมติฐานของสาเหตุที่เป็นไปได้หรือพิจารณาความผิดปกติที่เห็นได้ชัด
4. ทดสอบสมมติฐานเพื่อรับสุภาพที่แท้จริง
5. จัดทำแผนปฏิบัติการ
6. ดำเนินการตามแผนหรือส่งต่อปัญหา (Escalate)
7. ตรวจสอบการทำงานของระบบโดยรวม
8. วางแผนการป้องกันเพื่อลบเลี้ยงปัญหาในอนาคต
9. วิเคราะห์สาเหตุรากของปัญหา (Root Cause Analysis)
10. บันทึกผลการตรวจสอบ การดำเนินการ และผลลัพธ์อย่างเป็นระบบตลอดกระบวนการ

4.1.1 การระบุปัญหา

ขั้นตอนแรกในการแก้ไขปัญหาคือการระบุว่าปัญหาคืออะไร ซึ่งอาจมาจากผู้ใช้งานที่รายงานเข้ามา ระบบตรวจสอบอัตโนมัติ ไฟล์บันทึกเหตุการณ์ (Log Files) หรือการแจ้งเตือนจากแดชบอร์ด ทั้งนี้หลักสำคัญที่ควรตั้งข้อสังเกตก็คือ:

“หากระบบใช้งานได้เมื่อวาน แต่วันนี้ไม่สามารถใช้งานได้ มีอะไรเปลี่ยนแปลง?”

ผู้ดูแลระบบควรพิจารณาการเปลี่ยนแปลงทั้งทางด้านโครงสร้างพื้นฐานไอทีและสิ่งแวดล้อม เช่น การปรับปรุงระบบไฟฟ้า ระบบปรับอากาศ หรือแม้แต่การเปลี่ยนแปลงจากผู้ให้บริการระบบคลาวด์ของ

4.1.2 การกำหนดขอบเขตของปัญหา

เมื่อทราบว่าปัญหาคืออะไรแล้ว ต้องประเมินขอบเขตของผลกระทบ เช่น ปัญหาเกิดกับผู้ใช้งานรายเดียวหรือหลายราย เกิดขึ้นเฉพาะที่สาขาเดียวหรือหลายสาขา รวมถึงต้องพิจารณาว่าเป็นปัญหาที่เกี่ยวกับเครือข่ายหรือซอฟต์แวร์ เช่น การตั้งค่าผิดพลาด การจำลองสถานการณ์ (Reproduce) จะช่วยให้เข้าใจปัญหาชัดเจนยิ่งขึ้น

ควรทำการสำรวจข้อมูลหากมีความเสี่ยงที่จะเกิดความเสียหาย

4.1.3 การตั้งสมมติฐานของสาเหตุที่เป็นไปได้

อย่าคิดว่าปัญหาทุกอย่างจะซับซ้อนเสมอไป ควรเริ่มต้นจากสิ่งพื้นฐาน เช่น ตรวจสอบว่าบริการนั้นเริ่มทำงานหรือไม่ สมมติฐานควรตั้งจากข้อมูลและบริบทของระบบที่เกี่ยวข้อง และหากมีองค์ประกอบร่วมที่เกี่ยวข้องกับปัญหานั้นพยายามบริการ เช่น การอัปเดตระบบ หรือการปรับค่าใด ๆ ก็ควรนำมาพิจารณา

4.1.4 การทดสอบสมมติฐาน

เมื่อได้สมมติฐานแล้ว ควรดำเนินการทดสอบเพื่อยืนยันความถูกต้อง หากสมมติฐานถูกต้องก็สามารถเข้าสู่แผนปฏิการได้ทันที หากไม่ถูกต้องจะต้องตั้งสมมติฐานใหม่และทำการทดสอบอีกครั้ง

4.1.5 การจัดทำแผนปฏิบัติการ

แผนการแก้ไขควรคำนึงถึงผลกระทบ เช่น การหยุดให้บริการหรือความเสี่ยงของการสูญเสียของข้อมูล ควรแจ้งผู้ใช้งานที่เกี่ยวข้องล่วงหน้า และระบุระยะเวลาการแก้ไขอย่างชัดเจน

4.1.6 การดำเนินการหรือส่งต่อปัญหา

ในกรณีที่สามารถแก้ไขได้เอง ควรปฏิบัติตามแผนที่วางไว้อย่างเคร่งครัด และทำการทดสอบผลลัพธ์ ในแต่ละขั้นตอนก่อนดำเนินการต่อ หากไม่สามารถดำเนินการได้ด้วยตนเอง ควรส่งต่อให้ทีมงาน ผู้ให้บริการ หรือผู้เชี่ยวชาญในเรื่องนั้น

4.1.7 การตรวจสอบการทำงานของระบบโดยรวม

เมื่อดำเนินการแก้ไขปัญหาแล้ว ควรตรวจสอบว่าระบบกลับมาทำงานได้อย่างสมบูรณ์ตามระดับ การให้บริการ (Service Level) ที่กำหนดไว้

4.1.8 การวางแผนการป้องกัน

ควรปรับปรุงระบบเพื่อป้องกันปัญหาในอนาคต เช่น การเพิ่มระบบสำรองข้อมูล การใช้ RAID หรือระบบ High Availability เพื่อลดความเสี่ยง

4.1.9 การวิเคราะห์สาเหตุรากของปัญหา (Root Cause Analysis)

หลังจากระบบกลับมาทำงานได้ ควรทำการวิเคราะห์สาเหตุที่แท้จริงเพื่อป้องกันการเกิดซ้ำ และปรับปรุงกระบวนการหรือเทคโนโลยีที่เกี่ยวข้องให้มีความมั่นคงยิ่งขึ้น

4.1.10 การบันทึกผลการดำเนินงาน

การบันทึกขั้นตอน ปัญหา สมมติฐาน และผลการดำเนินการอย่างละเอียดในแต่ละขั้นตอนจะช่วยให้สามารถนำข้อมูลกลับมาใช้ในการปรับปรุงระบบในอนาคตได้อย่างมีประสิทธิภาพ อีกทั้งยังเป็นแนวทางที่ดีในงานบริการ Service Desk ที่ต้องปิด Ticket ด้วยข้อมูลที่ครบถ้วน

4.2 นโยบาย ขั้นตอน และผลกระทบขององค์กร

ในการดำเนินการแก้ไขปัญหาเกี่ยวกับการใช้งานระบบคลาวด์ ไม่ว่าจะเป็นการติดตั้งระบบ เครือข่าย หรือความปลอดภัย ผู้ดูแลระบบควรพิจารณานโยบายขององค์กรอย่างรอบคอบ โดยเฉพาะนโยบายที่เกี่ยวข้อง กับขั้นตอนการปฏิบัติงานมาตรฐาน (Standard Operating Procedures: SOPs) ซึ่งเป็นแนวทางที่ชัดเจน ในการดำเนินงานแต่ละด้าน และต้องปฏิบัติตามอย่างเคร่งครัดในการแก้ไขปัญหา

ตัวอย่างเช่น การบริหารจัดการสิทธิ์ผู้ใช้งาน (Identity and Access Management: IAM) ต้องเป็นไปตามนโยบายขององค์กรอย่างระมัดระวัง เพื่อหลีกเลี่ยงการให้สิทธิ์มากเกินไปหรือน้อยเกินไป ซึ่งอาจก่อให้เกิด ความเสี่ยงหรือข้อจำกัดในการเข้าถึงทรัพยากรระบบคลาวด์

นอกจากนี้ นโยบายด้านคลาวด์ยังครอบคลุมถึงกระบวนการส่งต่อกรณีปัญหา (Case Escalation) ไปยัง ฝ่ายสนับสนุนของผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) ซึ่งอาจมีค่าใช้จ่ายเพิ่มเติม ดังนั้น ผู้ดูแลระบบจึงต้องมีความเข้าใจภาพรวมของสภาพแวดล้อมระบบคลาวด์อย่างครบถ้วนก่อนที่จะดำเนินการ ส่งต่อปัญหา

อีกประดิษฐ์สำคัญคือ ข้อตกลงระดับการให้บริการ (Service-Level Agreements: SLAs) ซึ่งอาจมี บทลงโทษกรณีที่องค์กรไม่สามารถให้บริการได้ตามข้อตกลง อย่างไรก็ตาม ผู้ให้บริการระบบคลาวด์เอง ก็มี SLA เพื่อรับประกันความพร้อมใช้งานเช่นกัน หากเกิดเหตุข้องหารีไม่สามารถเข้าถึงระบบได้ และเป็น ความรับผิดชอบของ CSP องค์กรผู้ใช้บริการอาจได้รับการลดค่าบริการหรือการชดเชยในรูปแบบอื่น ตามข้อตกลงที่กำหนดไว้

4.3 ทบทวนกระบวนการแก้ไขปัญหา

แม้ว่าการแก้ไขปัญหาในสภาพแวดล้อมระบบคลาวด์จะใช้แนวทางเดียวกับระบบภายในองค์กร (On-Premises) หรือระบบทางกายภาพทั่วไป แต่ในระบบคลาวด์จะมีตัวแปรเพิ่มเติมที่ต้องพิจารณา เช่น

- ปัญหา Downtime อาจเกี่ยวข้องกับการเชื่อมต่อเครือข่ายระหว่างผู้ใช้และระบบคลาวด์ หรืออาจเกิดจากความล้มเหลวในระบบของผู้ให้บริการคลาวด์เอง
- ความผิดพลาดในการตั้งค่าการควบคุมการเข้าถึง (Access Control Misconfigurations) ที่ทำให้ผู้ใช้งานไม่สามารถเข้าถึงฟังก์ชันที่ควรใช้งานได้
- ปัญหาการติดตั้งระบบ (Deployment Issues) อาจเกิดจากการใช้ทรัพยากรมากเกินไป ข้อจำกัดของระบบ การเลิกใช้งานฟีเจอร์บางอย่าง (Feature Deprecation) หรือความพร้อมใช้งานของบริการในแต่ละภูมิภาค

อย่างไรก็ตาม หลักการในการระบุปัญหา ระบุขอบเขตของปัญหา วิเคราะห์หาสาเหตุ และดำเนินการแก้ไข ยังคงเป็นแนวทางหลักที่สามารถนำมาใช้ได้เสมอ

สุดท้าย อย่าลืมความสำคัญของการบันทึกผลการตรวจสอบและดำเนินการเพื่อป้องกันการเกิดปัญหาแบบเดียวกันในอนาคต การเรียนรู้จากเหตุการณ์ที่เกิดขึ้นเป็นหัวใจสำคัญในการบริหารจัดการระบบคลาวด์อย่างมีประสิทธิภาพ

สรุปความเข้าใจเกี่ยวกับแนวคิดสถาปัตยกรรมระบบคลาวด์

รูปแบบหลักของบริการระบบคลาวด์มีอยู่ 5 ประเภท ได้แก่ Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Function as a Service (FaaS) และ Anything as a Service (XaaS) โดยยังมีรูปแบบย่อยอื่น ๆ ที่พัฒนาขึ้นจากแนวคิดพื้นฐานเหล่านี้เพื่อตอบสนองต่อความต้องการที่แตกต่างกัน หนึ่งในประเด็นสำคัญของแต่ละโมเดล คือ ความรับผิดชอบด้านความปลอดภัย ซึ่งจะถูกแบ่งออกระหว่างผู้ให้บริการระบบคลาวด์และผู้ใช้บริการตามขอบเขตของบริการที่เลือกใช้

บริการคลาวด์สนับสนุนเทคโนโลยีหลากหลายรูปแบบ ไม่ว่าจะเป็นการเรียนรู้ของเครื่อง (Machine Learning), ปัญญาประดิษฐ์ (Artificial Intelligence) และเทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) ซึ่งล้วนเป็นองค์ประกอบสำคัญของการประมวลผลข้อมูลยุคใหม่

ผู้ดูแลระบบคลาวด์มีหน้าที่ในการทำความเข้าใจ ออกแบบ และนำบริการคลาวด์ไปใช้งานอย่างเหมาะสม และในกรณีที่เกิดปัญหา ยังต้องสามารถแก้ไขและบริหารจัดการทรัพยากรคลาวด์ได้อย่างมีประสิทธิภาพ โดยเฉพาะการใช้แนวทางแก้ไขปัญหาที่เป็นระบบตาม วิธีการแก้ไขปัญหาของ CompTIA (CompTIA Troubleshooting Methodology) ซึ่งเป็นขั้นตอนพื้นฐานที่ช่วยให้การจัดการปัญหามีความชัดเจนและตรวจสอบได้

บทที่ 2

การวางแผนบริการระบบคลาวด์ (Planning Cloud Services)

บทนำของบทเรียน (Module Introduction)

บริการระบบคลาวด์ได้เปลี่ยนแปลงรูปแบบการให้บริการแอปพลิเคชันอย่างสิ้นเชิง นักพัฒนาสามารถเข้าถึงทางเลือกใหม่ ๆ ในการนำแอปพลิเคชันเข้าสู่ระบบปฏิบัติการผ่านคลาวด์ ซึ่งความสามารถที่เพิ่มขึ้นนี้ ส่งผลให้แนวคิดด้านการออกแบบแอปพลิเคชันและบริการต้องมีการปรับเปลี่ยนให้สอดคล้องกับเทคโนโลยี ดังกล่าว

หนึ่งในประเด็นสำคัญของการนำระบบคลาวด์มาใช้งานคือ การตัดสินใจว่าจะใช้บริการใดที่ได้ ผู้ให้บริการระบบคลาวด์แบบสาธารณะ (Public Cloud Providers) เช่น Amazon Web Services (AWS), Azure และ Google Cloud Platform (GCP) ให้บริการโดยใช้ทรัพยากร่วมกันระหว่างลูกค้าหลายราย บางองค์กรเลือกที่จะพัฒนาและดูแลศูนย์ข้อมูลของตนเอง (Private Data Centers) เพื่อบริหารจัดการระบบคลาวด์ ในขณะที่บางองค์กรใช้รูปแบบผสมผสาน (Hybrid Cloud)

เมื่อองค์กรตัดสินใจว่าจะใช้บริการแอปพลิเคชันใดที่ได้แล้ว ขั้นตอนต่อไปคือการกำหนดว่าจะนำแอปพลิเคชันเข้าสู่ระบบอย่างไร และจะอัปเกรดบริการเหล่านั้นอย่างไร โดยรูปแบบการปรับใช้งาน (Deployment Models) จะช่วยเป็นแนวทางในการบริหารจัดการระบบคลาวด์

วัตถุประสงค์ของบทเรียน (Module Objectives)

ในบทเรียนนี้ ผู้เรียนจะได้:

- ศึกษาแนวคิดและองค์ประกอบของการออกแบบระบบคลาวด์โดยเฉพาะ (Cloud-Native Design Concepts and Components)
- ทำความเข้าใจเกี่ยวกับรูปแบบการปรับใช้ระบบคลาวด์ (Cloud Deployment Models)
- ทำความเข้าใจเกี่ยวกับกลยุทธ์การปรับใช้ระบบคลาวด์ (Cloud Deployment Strategies)

หน่วยที่ 1 การออกแบบระบบคลาวด์เนทีฟ (Cloud-native Design)

การออกแบบระบบคลาวด์เนทีฟ (Cloud-native design) หมายถึง การพัฒนาแอปพลิเคชันโดยอาศัยเทคโนโลยีของระบบคลาวด์เป็นหลัก เพื่อเพิ่มความยืดหยุ่นและประสิทธิภาพในการใช้งาน แอปพลิเคชันที่ออกแบบในลักษณะนี้ สามารถปรับขยาย (scale) ได้ง่าย รองรับการใช้งานในสภาพแวดล้อมที่เปลี่ยนแปลงรวดเร็ว และตอบสนองต่อความต้องการของผู้ใช้ได้อย่างมีประสิทธิภาพ

บทเรียนนี้ จะกล่าวถึงข้อดี และข้อเสียของการออกแบบระบบคลาวด์เนทีฟ พร้อมทั้งยกตัวอย่างประกอบ และอธิบายบทบาทของไมโครเซอร์วิส (Microservices) ซึ่งเป็นองค์ประกอบสำคัญของการออกแบบระบบคลาวด์เนทีฟ

1.1 แนวคิดการออกแบบแบบคลาวด์เนทีฟ (Cloud-native Design Concepts)

แนวคิดการออกแบบแบบคลาวด์เนทีฟ (Cloud-native design) คือแนวทางใหม่ในการออกแบบแอปพลิเคชันที่มุ่งเน้นการใช้ประโยชน์สูงสุดจากบริการระบบคลาวด์ โดยมุ่งลดข้อจำกัดที่เกิดจากแนวทางการพัฒนาแอปพลิเคชันแบบเดิม

เนื่องจากโมเดลการให้บริการของคลาวด์มีความแตกต่างจากแพลตฟอร์มแบบเดิม การออกแบบแอปพลิเคชันที่ทำงานในสภาพแวดล้อมคลาวด์จึงต้องมีแนวทางที่เหมาะสมกับความยืดหยุ่นและลักษณะเฉพาะของคลาวด์

แอปพลิเคชันแบบเดิมมักเป็นระบบแบบรวมศูนย์ (monolithic) ที่มีโครงสร้างตายตัวและพึ่งพาระบบปฏิบัติการหรือโครงสร้างพื้นฐานเช่น การอัปเดตระบบเหล่านี้มักมาในรูปแบบเวอร์ชันใหญ่ที่มีการเปลี่ยนแปลงอย่างมากทั้งในเรื่องของข้อกำหนด คุณสมบัติ และการเพิ่มพาระบบอื่น ซึ่งเป็นอุปสรรคต่อการเปลี่ยนแปลงและพัฒนาอย่างรวดเร็ว

ข้อเสียของการพัฒนาแบบเดิม ได้แก่

- ปรับขนาด (scale) ได้ยาก
- ไม่สามารถปรับให้ทำงานบนแพลตฟอร์มที่หลากหลายได้ง่าย
- การอัปเดตหรือแก้ไขทำได้ยาก
- ผู้ใช้อาจต้องเรียนรู้ใหม่มากเมื่อมีการเปลี่ยนเวอร์ชันใหญ่

1.2 ข้อดีและข้อเสียของการออกแบบแบบคลาวด์เนทีฟ (Cloud-native Design Advantages and Disadvantages)

การออกแบบแบบคลาวด์เนทีฟ (Cloud-native design) เป็นแนวทางที่ใช้ประโยชน์จากกลไกการให้บริการของระบบคลาวด์อย่างเต็มที่ โดยเน้นการสร้างแอปพลิเคชันให้มีความยืดหยุ่น แทนที่จะสร้างโปรแกรมเป็นหน่วยเดียวแบบผูกมัดกันแน่น แอปพลิเคชันแบบคลาวด์เนทีฟจะถูกออกแบบเป็นส่วนย่อยที่แยกจากกัน (loosely coupled) ซึ่งช่วยให้สามารถพัฒนา ปรับเปลี่ยน และปรับขนาด (scale) ได้ง่ายขึ้น ทั้งยังรองรับรูปแบบการให้บริการต่าง ๆ ของระบบคลาวด์

ข้อดีของการออกแบบแบบคลาวด์เนทีฟ

- ปรับขนาดระบบได้ดี (Improved scalability)
- พัฒนาและเปลี่ยนแปลงได้รวดเร็ว (Quicker changes/innovation)
- รองรับการทำงานข้ามแพลตฟอร์ม (Platform portability)
- ลดต้นทุนโดยรวม (Lower cost)
- รองรับความพร้อมใช้งานสูง (Higher availability)
- ปฏิบัติตามข้อกำหนดได้ดีขึ้น (Improved compliance)
- เพิ่มความมั่นคงปลอดภัย (Increased security)

ข้อเสียของการออกแบบแบบคลาวด์เนทีฟ

- อาจต้องมีการเปลี่ยนแปลงโครงสร้างพื้นฐานที่มีอยู่ (Cost infrastructure changes)
- ต้องมีขั้นตอนการพัฒนาที่ชัดเจนเพื่อให้ได้ประสิทธิภาพสูงสุด (Development processes must exist)
- ต้องใช้ทักษะทางเทคนิคของนักพัฒนาที่เหมาะสม (Appropriate developer technical skills)
- อาจเจอแรงต้านทางวัฒนธรรมในองค์กร (Cultural resistance/buy-in)

ควรตระหนักว่า แนวทางคลาวด์เนทีฟไม่ได้ “ดีกว่า” ในทุกรูป การออกแบบแบบดั้งเดิม (monolithic) ยังคงเหมาะสมสำหรับแอปพลิเคชันบางประเภท โดยเฉพาะอย่างยิ่งแอปที่ทำงานบนระบบเดี่ยวในสถานที่ (on-premises) การเลือกใช้แนวทางคลาวด์เนทีฟควรขึ้นอยู่กับลักษณะของงานและเป้าหมายของการใช้งาน

เทคโนโลยีที่เกี่ยวข้องกับการออกแบบแบบคลาวด์เนทีฟ ได้แก่

- DevOps และหลักการ CI/CD (Continuous Integration and Continuous Delivery)
- การใช้คอนเทนเนอร์ (Containerization)
- การจัดการแบบอัตโนมัติด้วย Orchestration
- แนวคิดไมโครเซอร์วิส (Microservices)
- พัฟ์ชันแบบไม่ใช้เซิร์ฟเวอร์ (Function as a Service หรือ Serverless application)

หมายเหตุ: อย่าสับสนระหว่างแนวคิด Cloud-native กับ Cloud-first เพราะ Cloud-first คือแนวโน้มโดยทั่วไป ระดับองค์กรที่มุ่งเน้นให้การพัฒนาแอปใหม่เป็นคลาวด์เนทีฟเสมอ เว้นแต่ว่าไม่สามารถทำได้เท่านั้นจึงเลือกใช้แอปแบบดั้งเดิม

1.3 สภาพแวดล้อมแบบไร้เซิร์ฟเวอร์ (Serverless Environments)

สถาปัตยกรรมแบบไร้เซิร์ฟเวอร์ (Serverless architecture) คือแนวทางการพัฒนาแอปพลิเคชันที่ยกเลิกความจำเป็นในการจัดการฮาร์ดแวร์ ระบบปฏิบัติการ และการบำรุงรักษาบริการของนักพัฒนาโดยตรง คำว่า “ไร้เซิร์ฟเวอร์” ไม่ได้หมายถึงว่าไม่มีเซิร์ฟเวอร์เลย แต่หมายถึงว่า ผู้ใช้ไม่ต้องรับผิดชอบในการจัดการเซิร์ฟเวอร์เหล่านั้นเอง ระบบจะใช้การจัดการทรัพยากรแบบอัตโนมัติผ่านบริการของผู้ให้บริการคลาวด์ เช่น AWS, Azure, หรือ Google Cloud โดยบริการนี้เรียกว่า Function as a Service (FaaS)

ระบบแบบไร้เซิร์ฟเวอร์สามารถเริ่มต้นการทำงานของเซิร์ฟเวอร์เมื่อมีความต้องการ และสามารถปรับขนาดขึ้นหรือลงตามปริมาณการใช้งานได้แบบอัตโนมัติ

1.3.1 ไมโครเซอร์วิส (Microservices)

แอปพลิเคชันแบบคลาวด์ หรือบริการที่บริหารจัดการโดยคลาวด์ มักจะไม่ถูกพัฒนาเป็นโปรแกรมขนาดใหญ่เพียงหน่วยเดียว (monolithic) แต่จะถูกแบ่งออกเป็นชุดของ คอมโพenenต์ย่อย ๆ ที่ทำงานร่วมกันซึ่งคอมโพenenต์เหล่านี้เรียกว่า **ไมโครเซอร์วิส (Microservices)**

ไมโครเซอร์วิสคือองค์ประกอบย่อยอิสระที่ทำหน้าที่เฉพาะเจาะจงแต่ละด้าน โดยมีลักษณะเด่นดังนี้:

- มีขนาดเล็ก (Small size components)
- แยกจากกันโดยสมบูรณ์ (Independent pieces)
- ทำหน้าที่เดียว (Single function)
- เป็นระบบปิดในตัวเอง (Self-contained)
- สามารถขยายขนาดได้ง่าย (Scalable)

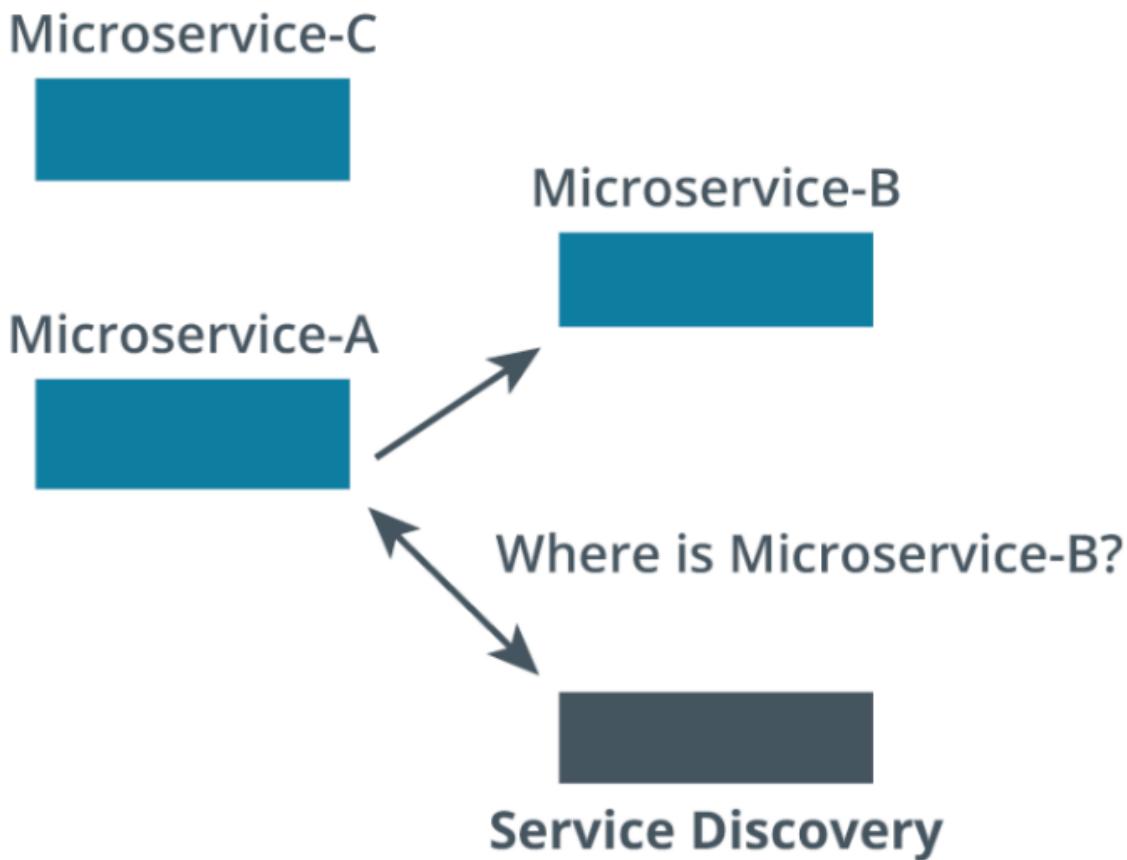
ข้อดีของไมโครเซอร์วิส

- ขยายระบบได้ง่ายขึ้น (Easier scalability)
- ใช้เวลาในการพัฒนาน้อยลง (Quicker build times)
- สามารถปรับปรุงบริการเฉพาะจุดได้สะดวก (Easier service improvement)

ไมโครเซอร์วิสจะไม่กำหนดล่วงหน้าว่าองค์ประกอบอื่นๆ อยู่ที่ใด ทำให้สามารถเชื่อมต่อกันได้อย่าง “หลวม” (Loosely Coupled) ซึ่งเป็นแนวทางสำคัญของการออกแบบแบบคลาวด์เนทีฟ (Cloud-native architecture) และมักใช้ควบคู่กับสภาพแวดล้อมแบบไร้เซิร์ฟเวอร์ เพื่อให้สามารถส่งมอบบริการได้รวดเร็ว มีความยืดหยุ่น และปรับขนาดได้ตามความต้องการ

1.4 การค้นหาบริการ (Service Discovery)

ในสถาปัตยกรรมไมโครเซอร์วิส (Microservices Architecture) แต่ละบริการจะทำงานอย่างอิสระ และให้ฟังก์ชันเฉพาะของตนเอง เมื่อนำมาทำงานร่วมกัน จึงจะเกิดเป็นแอปพลิเคชันที่สมบูรณ์ หนึ่งในข้อได้เปรียบสำคัญของระบบคลาวด์คือ ความสามารถในการปรับขนาด (Scalability) ซึ่งทำให้จำนวนของไมโครเซอร์วิสสามารถเพิ่มหรือลดลงได้ตามภาระงานแบบไดนามิก



Microservice-A uses service discovery to find Microservice-B.

ภาพที่ 7 Service Discovery

ดังนั้น ไมโครเซอร์วิสนี้จึง ไม่สามารถระบุพิกัด (hard-coded) ของอีกไมโครเซอร์วิสได้ล่วงหน้า เพราะจะทำให้เกิดการเชื่อมโยงแน่น (tightly coupled) ซึ่งขัดต่อหลักการของการออกแบบแบบ *loosely coupled* ที่เป็นหัวใจของแนวคิดคลาวด์เนทีฟ

การ ค้นหาบริการ (Service Discovery) คือวิธีที่ไมโครเซอร์วิสที่ถูกสร้างขึ้นแบบไดนามิก จะสามารถค้นหาและเชื่อมต่อกับบริการอื่น ๆ ได้

ตัวอย่าง:

- Microservice-A คือผู้บริโภคบริการ (Service Consumer)
- Microservice-B คือผู้ให้บริการ (Service Provider)

ผู้ให้บริการจะลงทะเบียนตนเองใน Service Registry พร้อมระบุความสามารถที่มี ส่วนผู้บริโภค จะค้นหาผ่าน Service Registry นี้ เพื่อเชื่อมต่อกับบริการที่ต้องการ

Service Discovery มีความสำคัญมากในระบบไมโครเซอร์วิส เพราะช่วยให้เกิดแนวทางการออกแบบ ขั้นสูง เช่น:

- Function Chains (โซ่อุปทาน)
- Fan-out/Fan-in (การกระจายและรวมผลพังก์ชัน)

1.5 รูปแบบการออกแบบฟังก์ชัน (Function Design Patterns)

ในระบบ ไรเซอร์ฟเวอร์ (Serverless) การทำงานจะขับเคลื่อนโดย ฟังก์ชัน (Functions) ซึ่งหมายถึง โปรแกรมย่อที่ทำหน้าที่เพียงอย่างเดียว ถูกเรียกใช้งานเมื่อมีเหตุการณ์เกิดขึ้น (event-triggered) ฟังก์ชัน ไม่มีการเก็บสถานะ-data (stateless) และจะเริ่มต้นหรือถูกทำลายโดยอัตโนมัติตามความจำเป็น โดยทั่วไปแล้วฟังก์ชันเหล่านี้จะใช้ในสองรูปแบบหลัก:

1. Function Chain (ใช้ฟังก์ชันแบบเรียงลำดับ)

- ฟังก์ชันแต่ละตัวทำงานต่อเนื่องกันทีละตัว
- เช่น: ฟังก์ชัน A → B → C



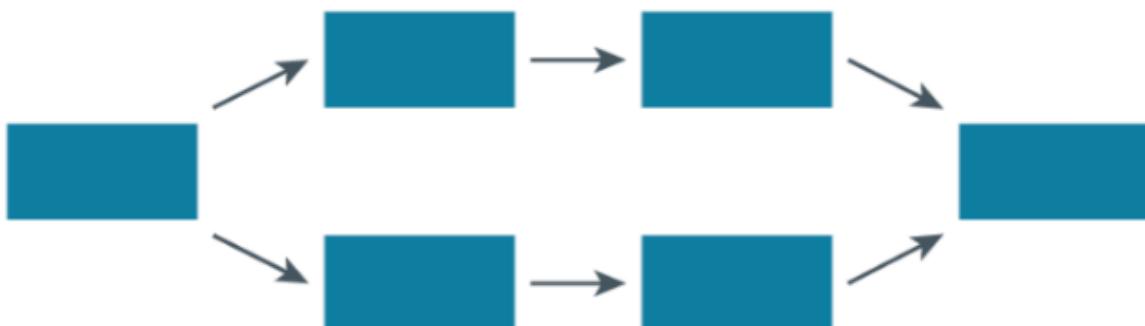
A function chain with functions executing serially.

ภาพที่ 8 Function Chain

2. Fan-out / Fan-in

- ฟังก์ชันหลายตัวทำงานพร้อมกัน (fan-out)
- ผลลัพธ์จะถูกรวบเข้าด้วยกัน (fan-in)
- เช่น: A → [B, C] ทำงานพร้อมกัน แล้วรวมผลลัพธ์ D

รูปแบบเหล่านี้ช่วยเพิ่มความเร็ว ความยืดหยุ่น และการจัดการโหลดในสภาพแวดล้อมแบบคลาวด์ เนทีฟและไรเซอร์ฟเวอร์ได้อย่างมีประสิทธิภาพ



A fan-out/fan-in design with functions executing simultaneously.

ภาพที่ 9 Fan-out/Fan-in

หน่วยที่ 2 รูปแบบการปรับใช้ในคลาวด์ (Cloud Deployment Models)

รูปแบบการปรับใช้คลาวด์หมายถึงตำแหน่งที่ตั้งทางกายภาพและเจ้าของโครงสร้างพื้นฐานของผู้ให้บริการคลาวด์ ซึ่งมีผลต่อการควบคุม ทรัพยากร และรูปแบบการใช้งานของบริการคลาวด์นั้น ๆ

ผู้ให้บริการคลาวด์สาธารณะ เช่น Amazon, Microsoft และ Google เปิดให้เข้าถึงทรัพยากรของศูนย์ข้อมูลผ่านระบบการสมัครใช้งานแบบรายเดือนหรือรายปี (subscription-based) โดยลูกค้าสามารถใช้ทรัพยากรได้ตามต้องการโดยไม่ต้องดูแลโครงสร้างพื้นฐานเอง

ในขณะที่คลาวด์ส่วนตัวจะถูกเป็นเจ้าของและบริหารจัดการโดยองค์กรโดยองค์กรหนึ่ง ซึ่งจัดสรรบริการคลาวด์ให้กับหน่วยงานภายในของตนเอง เพื่อควบคุมความปลอดภัยและการเข้าถึงข้อมูลได้อย่างเข้มงวด

นอกจากนี้ยังมีการออกแบบในรูปแบบผสมผสานระหว่างคลาวด์สาธารณะและคลาวด์ส่วนตัว เพื่อให้เกิดความยืดหยุ่นในการใช้งาน และเพิ่มประสิทธิภาพในการจัดการทรัพยากรให้เหมาะสมกับลักษณะงานหรือบริการที่แตกต่างกันในแต่ละองค์กร

2.1 องค์ประกอบของระบบคลาวด์ (Cloud Components)

ระบบบริการคลาวด์ประกอบด้วยองค์ประกอบหลัก 3 ส่วน ได้แก่:

1. แพลตฟอร์มของผู้ใช้งาน (Client Platform)

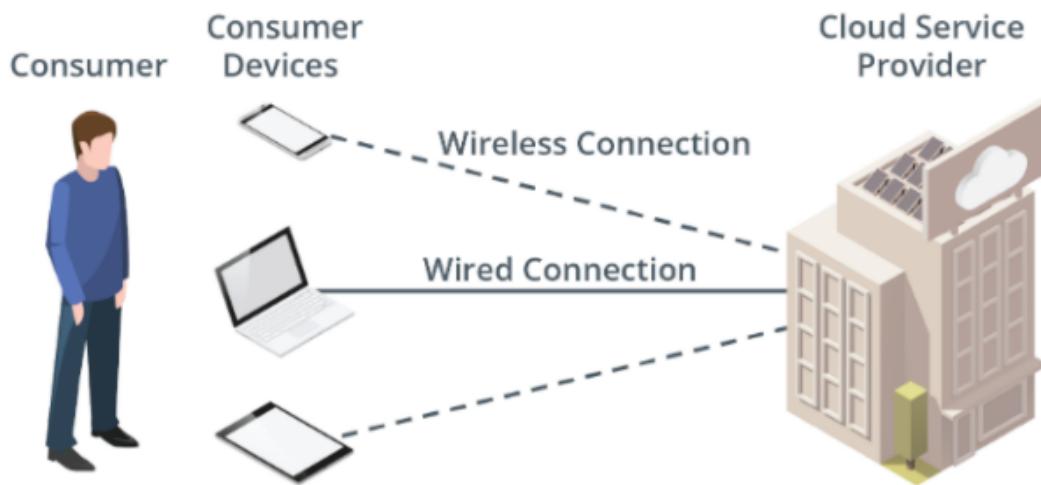
คืออุปกรณ์หรือระบบที่ผู้ใช้นำมาเข้าถึงบริการคลาวด์ เช่น คอมพิวเตอร์ แท็บเล็ต สมาร์ตโฟน หรืออุปกรณ์ IoT

2. ศูนย์ข้อมูลของผู้ให้บริการคลาวด์ (Data Center of Cloud Service Provider)

เป็นสถานที่ที่ให้บริการคลาวด์ทั้งหมด โดยมีความสามารถในการเข้าถึงพลังงานและอินเทอร์เน็ตที่เชื่อถือได้ มีระบบสำรอง (Redundancy) และมีความปลอดภัยทางกายภาพสูง

3. เครือข่ายที่เชื่อมต่อ (Network Connection)

เป็นเส้นทางที่เชื่อมโยงระหว่างศูนย์ข้อมูลของผู้ให้บริการคลาวด์กับอุปกรณ์ของผู้ใช้งาน อาจเป็นเครือข่ายภายในองค์กร อินเทอร์เน็ต หรือเครือข่ายโทรศัพท์มือถือ



The three components of cloud services are the client device, the data center, and the network that links them together. (Images © 123RF.com)

ภาพที่ 10 Cloud Components

2.1.2 ตารางบทบาทขององค์ประกอบคลาวด์

องค์ประกอบ	บทบาท
ผู้ให้บริการคลาวด์ (Cloud Service Provider - CSP)	โอล์ส์บริการคลาวด์ในศูนย์ข้อมูล
ผู้ใช้งาน (Client)	ใช้เป็นช่องทางเข้าถึงบริการคลาวด์
เครือข่าย (Network)	ทำหน้าที่เป็นเส้นทางการสื่อสารระหว่างผู้ใช้และคลาวด์

ตาราง 1 บทบาทขององค์ประกอบคลาวด์

ลักษณะของบริการคลาวด์และรูปแบบการติดตั้ง

- Public Cloud (คลาวด์สาธารณะ):** ให้บริการโดยผู้ให้บริการภายนอก เช่น Microsoft, Amazon, Google โดยลูกค้าหลายรายใช้ทรัพยากร่วมกัน
- Private Cloud (คลาวด์ส่วนตัว):** องค์กรเดียวเป็นเจ้าของและดำเนินการผ่านศูนย์ข้อมูลของตนเอง
- Community Cloud (คลาวด์ชุมชน):** องค์กรที่มีลักษณะความต้องการคล้ายกันรวมกลุ่มกันใช้คลาวด์ร่วมกัน
- Hybrid Cloud (คลาวด์แบบผสม):** ผสมผสานระหว่าง public, private หรือ community cloud
- Multi-Cloud (มัลติคลาวด์):** ใช้บริการจากผู้ให้บริการคลาวด์หลายรายพร้อมกัน เพื่อลดความเสี่ยงจากการผูกขาด ลดค่าใช้จ่าย และเพิ่มความหลากหลายในการให้บริการ

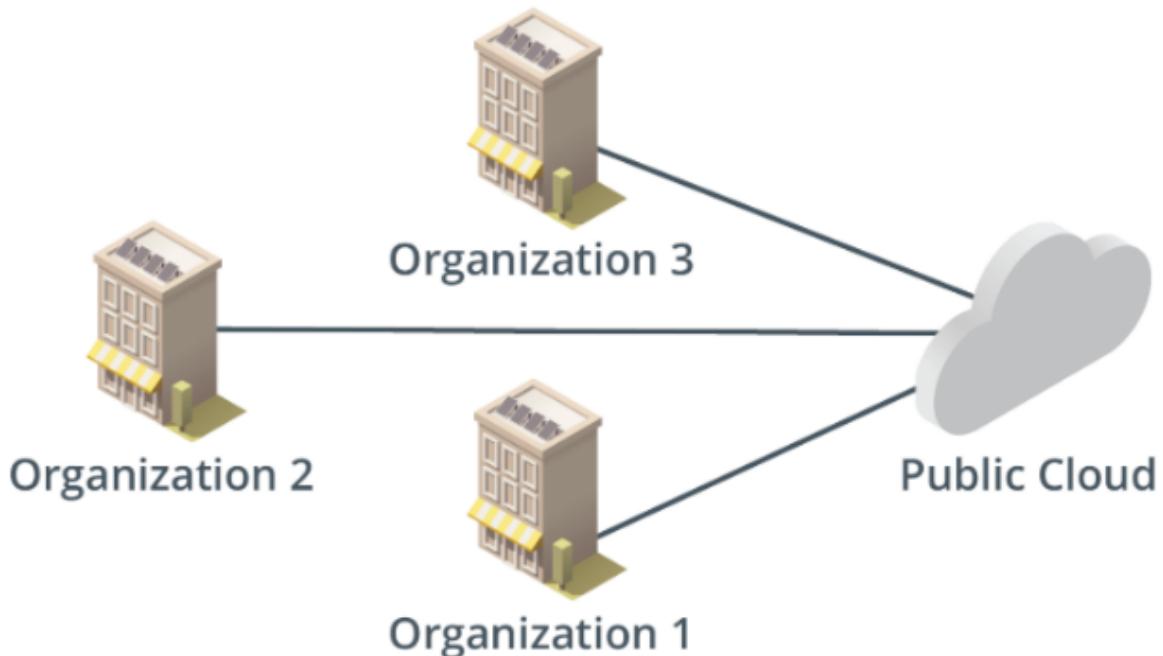
ระบบคลาวด์ในปัจจุบันรองรับแพลตฟอร์มและระบบปฏิบัติการที่หลากหลาย เช่น Windows, macOS, Linux, iOS และ Android ทำให้ผู้ใช้สามารถเข้าถึงบริการได้จากทุกที่ และอุปกรณ์ทุกประเภท ที่มีการเชื่อมต่อเครือข่าย

2.2 คลาวด์สาธารณะ (Public Cloud)

คลาวด์สาธารณะเป็นบริการคลาวด์ที่องค์กรต่าง ๆ ซึ่งไม่เกี่ยวข้องกันสามารถเข้ามาใช้งานร่วมกันได้ ผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) เช่น Amazon, Microsoft หรือ Google จะเปิดให้ลูกค้าเข้าถึงทรัพยากรในศูนย์ข้อมูลของตนผ่านระบบการสมัครใช้งานแบบรายเดือนหรือรายปี

ลูกค้าที่ใช้งานคลาวด์สาธารณะจะใช้ทรัพยากร่วมกับผู้ใช้รายอื่น ซึ่งระบบของผู้ให้บริการจะบริหารจัดการและจัดสรรทรัพยากรให้โดยอัตโนมัติตามปริมาณการใช้งานจริงในขณะนั้น ผู้ใช้งานจึงไม่จำเป็นต้องทราบว่าข้อมูลของตนถูกเก็บไว้ที่ตำแหน่งใดภายในศูนย์ข้อมูล

รูปแบบคลาวด์นี้ถือเป็นภาพจำของคนที่ไปเมื่อนึกถึง “คลาวด์คอมพิวติ้ง” โดยเฉพาะในกรณีของธุรกิจที่ต้องการความยืดหยุ่นและลดต้นทุนด้านโครงสร้างพื้นฐาน



Public cloud environments are shared among many unrelated organizations. (Images © 123RF.com.)

ภาพที่ 11 Public Cloud

ข้อดีของการใช้คลาวด์สาธารณะ

คลาวด์สาธารณะหมายความว่าอย่างยิ่งสำหรับธุรกิจขนาดเล็กและขนาดกลางที่ต้องการระบบที่มีความน่าเชื่อถือสามารถปรับขยายได้ และมีค่าใช้จ่ายที่ไม่สูงนัก

- ไม่ต้องลงทุนสร้างและดูแลศูนย์ข้อมูลเอง

- เนมาสำหรับใช้ในขั้นตอนการพัฒนา การทดสอบ และการใช้งานจริง
- รองรับการใช้งานที่หลากหลาย เช่น การจัดเก็บข้อมูล การประมวลผล และระบบเครือข่าย
- ช่วยให้ธุรกิจสามารถปรับตัวตามบริมาณงานที่เปลี่ยนแปลงได้อย่างมีประสิทธิภาพ

เทคโนโลยีเบื้องหลังคลาวด์สาธารณะ

คลาวด์สาธารณะทำงานโดยอาศัยเทคโนโลยีหลายอย่างผสมผสานกันเพื่อให้บริการได้อย่างเต็มประสิทธิภาพ ตัวอย่างเช่น:

- **การจำลองเสมือน (Virtualization)**
ช่วยให้สามารถสร้างเครื่องเซิร์ฟเวอร์เสมือนบนฮาร์ดแวร์จริงได้หลายเครื่อง เช่น
 - Microsoft Hyper-V
 - Amazon Linux AMI
 - VMware ESXi
- **ระบบจัดเก็บข้อมูล (Storage)**
มีหลายรูปแบบตามลักษณะการใช้งาน เช่น
 - เก็บวัตถุ: Azure Blob Storage, Amazon S3
 - เก็บแบบบล็อก: Azure Disk Storage, Amazon EBS
 - เก็บแบบไฟล์: Azure Files, Amazon EFS
- **ระบบเครือข่าย (Networking)**
สำหรับเชื่อมโยงระบบและบริการ เช่น
 - Azure Virtual Network หรือ Amazon VPC
 - VPN สำหรับสร้างเครือข่ายส่วนตัว
 - Software-defined Networking (SDN) สำหรับควบคุมเครือข่ายแบบยึดหยุ่น
- **การโฮสต์แอปพลิเคชัน (Application Hosting)**
ตัวอย่างการนำคลาวด์มาใช้โฮสต์เว็บไซต์หรือระบบ เช่น
 - เว็บแอปพลิเคชันด้วย JavaScript
 - WordPress
 - LAMP Stack (Linux, Apache, MySQL, PHP)

2.3 คลาวด์ส่วนตัว (Private Cloud)

คลาวด์ส่วนตัวเป็นรูปแบบการให้บริการคลาวด์ที่จำกัดการใช้งานเฉพาะองค์กรที่เป็นเจ้าของระบบเท่านั้น โดยองค์กรจะลงทุนสร้างศูนย์ข้อมูล (Data Center) ภายในบริษัทเอง พร้อมใช้เทคโนโลยีการจำลองเสมือน (Virtualization) และให้บริการทรัพยากรคลาวด์ภายในองค์กร ผ่านแคตตาล็อกของบริการที่ผู้ใช้ภายในสามารถเข้าถึงได้



A private cloud model is available only to the enterprise that owns it. (Image © 123RF.com.)

ภาพที่ 12 Private Cloud

แม้ระบบจะติดตั้งอยู่ภายในองค์กร แต่ก็สามารถใช้ประโยชน์จากเทคโนโลยีคลาวด์ได้อย่างเต็มที่ เช่น การปรับขยายได้ตามต้องการ (scalability), การให้บริการตามการใช้งานจริง (on-demand), และการจัดสรรทรัพยากรอัตโนมัติ

เหตุผลที่องค์กรเลือกใช้คลาวด์ส่วนตัว

- ต้องการ ควบคุมความปลอดภัยและข้อมูล อย่างใกล้ชิด
- ไม่สามารถใช้คลาวด์สาธารณะได้ เนื่องจากข้อกำหนดทางกฎหมาย กฎระเบียบ หรือข้อบังคับภายในอุตสาหกรรม
- เหมาะกับองค์กรที่ต้องรับมือกับ ข้อมูลสำคัญหรืออ่อนไหว เช่น
 - สถาบันการเงิน
 - โรงพยาบาลและหน่วยบริการสุขภาพ
 - หน่วยงานภาครัฐ
 - องค์กรที่ต้องปฏิบัติตามข้อกำหนดด้านความปลอดภัยและการปกป้องข้อมูล (เช่น HIPAA, GDPR)

เทคโนโลยีที่ใช้ในคลาวด์ส่วนตัว

แม้จะอยู่ภายในองค์กร แต่ระบบคลาวด์ส่วนตัวก็มักใช้เทคโนโลยีแบบเดียวกับคลาวด์สาธารณะ เช่น

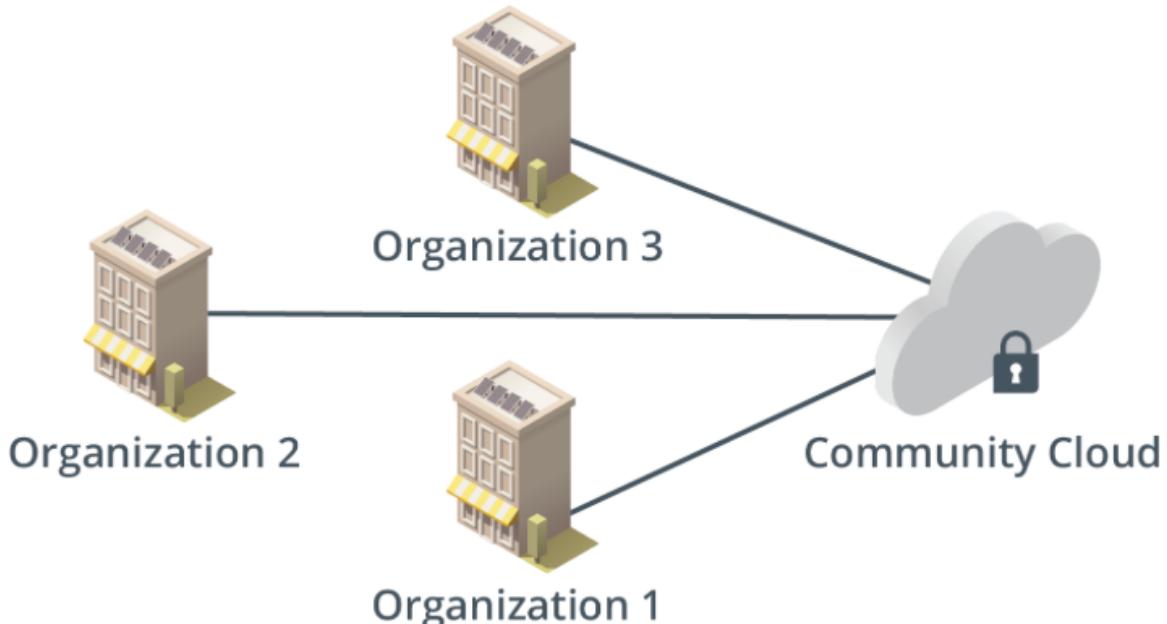
- การจำลองเสมือน (Virtualization):
 - Microsoft Hyper-V
 - VMware vSphere

- OpenStack
- ระบบจัดเก็บข้อมูลภายในองค์กร (Storage):
 - Storage Area Network (SAN) ภายใน
 - Network Attached Storage (NAS) ภายใน
- การโฮสต์แอปพลิเคชัน (Application Hosting):
 - เว็บแอปพลิเคชันที่ใช้ JavaScript
 - WordPress
 - LAMP Stack (Linux, Apache, MySQL, PHP)

2.4 คลาวด์ชุมชน (Community Cloud)

คลาวด์ชุมชนเป็นรูปแบบของระบบคลาวด์ที่ถูกออกแบบมาเพื่อให้บริการเฉพาะกลุ่มขององค์กรที่มีลักษณะการดำเนินธุรกิจหรือข้อกำหนดด้านความปลอดภัยที่คล้ายคลึงกัน โดยสมาชิกในกลุ่มนี้จะสามารถเข้าถึงทรัพยากรของระบบคลาวด์ได้เท่านั้น บุคคลภายนอกจะไม่สามารถใช้งานร่วมได้

การจัดการระบบคลาวด์ชุมชนอาจดำเนินการโดยองค์กรใดองค์กรหนึ่งในกลุ่ม หรืออาจเป็นการร่วมกันบริหารจัดการของหลายองค์กร หรืออาจว่าจ้างบุคคลภายนอกเป็นผู้ดูแลระบบก็ได้



*Community cloud environments are shared among only a specific group of related organizations.
(Images © 123RF.com.)*

ภาพที่ 13 Community Cloud

คุณสมบัติและเทคโนโลยีที่เกี่ยวข้อง

คลาวด์ชุมชนมักอิงจากเทคโนโลยีคลาวด์ทั่วไปที่รองรับการใช้งานแบบหลายผู้เช่า (Multitenant) โดยมีการควบคุมสิทธิ์การเข้าถึงอย่างรัดกุมผ่านระบบจัดการตัวตนและสิทธิ์ (Identity and Access Management: IAM)

ตัวอย่างเทคโนโลยีที่ใช้ได้แก่:

- ระบบรองรับหลายผู้เช่า เช่น OpenStack, Digital Ocean
- บริการตรวจสอบสิทธิ์ร่วม (Federated Authentication Services)
- เครื่องมือซอฟต์แวร์และความร่วมมือ เช่น ระบบฐานข้อมูล, ระบบไฟล์, เครื่องมือสื่อสาร ฯลฯ

กรณีการใช้งาน (Use Cases)

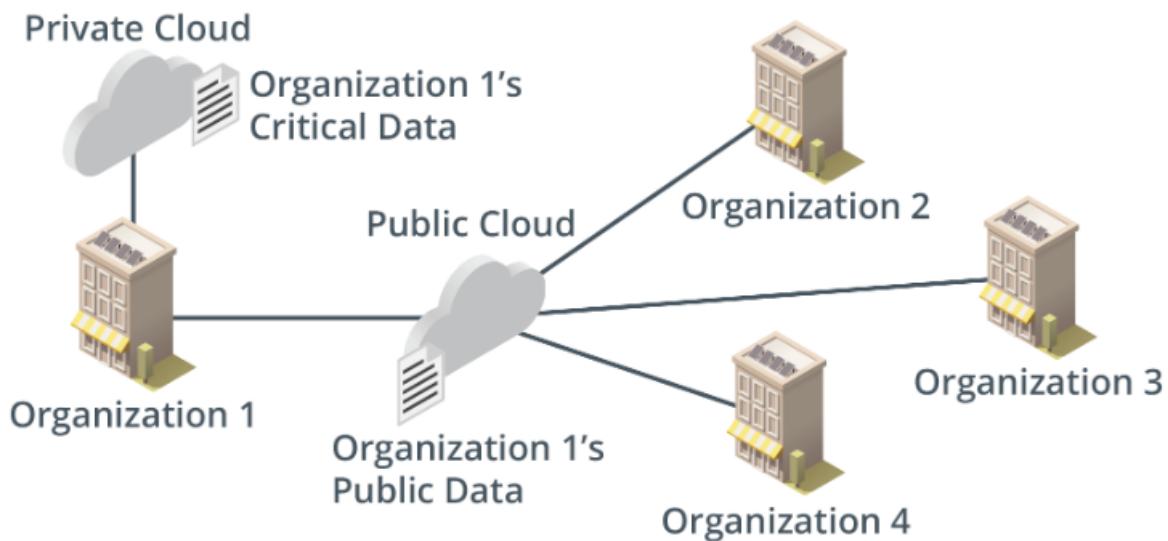
คลาวด์ชุมชนเหมาะสมสำหรับองค์กรที่ต้องการแบ่งปันทรัพยากรระหว่างกันในลักษณะร่วมมือหรือมีข้อกำหนดร่วมกัน เช่น:

- หน่วยงานด้านสาธารณสุข ที่ต้องการแลกเปลี่ยนข้อมูลเวชระเบียนของผู้ป่วยระหว่างกัน
- สถาบันวิจัย ที่ดำเนินโครงการศึกษาร่วมกัน
- สถาบันการศึกษา ที่แบ่งปันข้อมูลงานวิจัยระหว่างมหาวิทยาลัย

2.5 คลาวด์แบบผสม (Hybrid Cloud)

คลาวด์แบบผสมคือการผสมผสานรูปแบบการให้บริการคลาวด์ทั้งแบบสาธารณะ (Public Cloud), ส่วนตัว (Private Cloud) และ/หรือคลาวด์ชุมชน (Community Cloud) เข้าด้วยกัน เพื่อให้สามารถใช้งานได้อย่างยืดหยุ่นและตรงตามความต้องการขององค์กร

ตัวอย่างเช่น องค์กรหนึ่งอาจใช้บริการบางส่วนผ่านระบบคลาวด์สาธารณะของผู้ให้บริการ (Cloud Service Provider: CSP) เช่น การเก็บข้อมูลทั่วไปหรือให้บริการเว็บแอปพลิเคชันที่ไม่ต้องการความปลอดภัยสูงขนาดเดียวกัน องค์กรอาจเก็บข้อมูลที่มีความอ่อนไหวไว้ภายในศูนย์ข้อมูลของตนเองผ่านระบบคลาวด์ส่วนตัว เพื่อควบคุมความปลอดภัยและการเข้าถึงได้อย่างมีประสิทธิภาพ



Hybrid clouds are any combination of the other three models. (Image © 123RF.com.)

ภาพที่ 14 Hybrid Cloud

ข้อดีและการใช้งาน

คลาวด์แบบผสมหมายความอย่างยิ่งกับองค์กรที่มีข้อกำหนดด้านความปลอดภัยหลายระดับ เช่น:

- **ข้อมูลสำคัญ** เช่น ข้อมูลลูกค้า หรือข้อมูลภายในองค์กร จะถูกเก็บไว้ในคลาวด์ส่วนตัว เพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต
- **ข้อมูลทั่วไป** หรือบริการที่เน้นความคุ้มค่า อาจโฮสต์อยู่ในคลาวด์สาธารณะ เพื่อประหยัดต้นทุนและขยายขนาดได้จ่าย

อีกหนึ่งกรณีการใช้งานที่สำคัญคือ การกู้คืนจากภัยพิบัติ (Disaster Recovery) เช่น ธุรกิจขนาดเล็กอาจเก็บข้อมูลการทำงานประจำวันไว้ในเซิร์ฟเวอร์ภายในองค์กร แต่จะสำรองข้อมูลขึ้นคลาวด์ในช่วงเวลากลางคืน เพื่อให้สามารถกู้คืนได้จ่ายหากเกิดเหตุการณ์ไม่คาดคิด

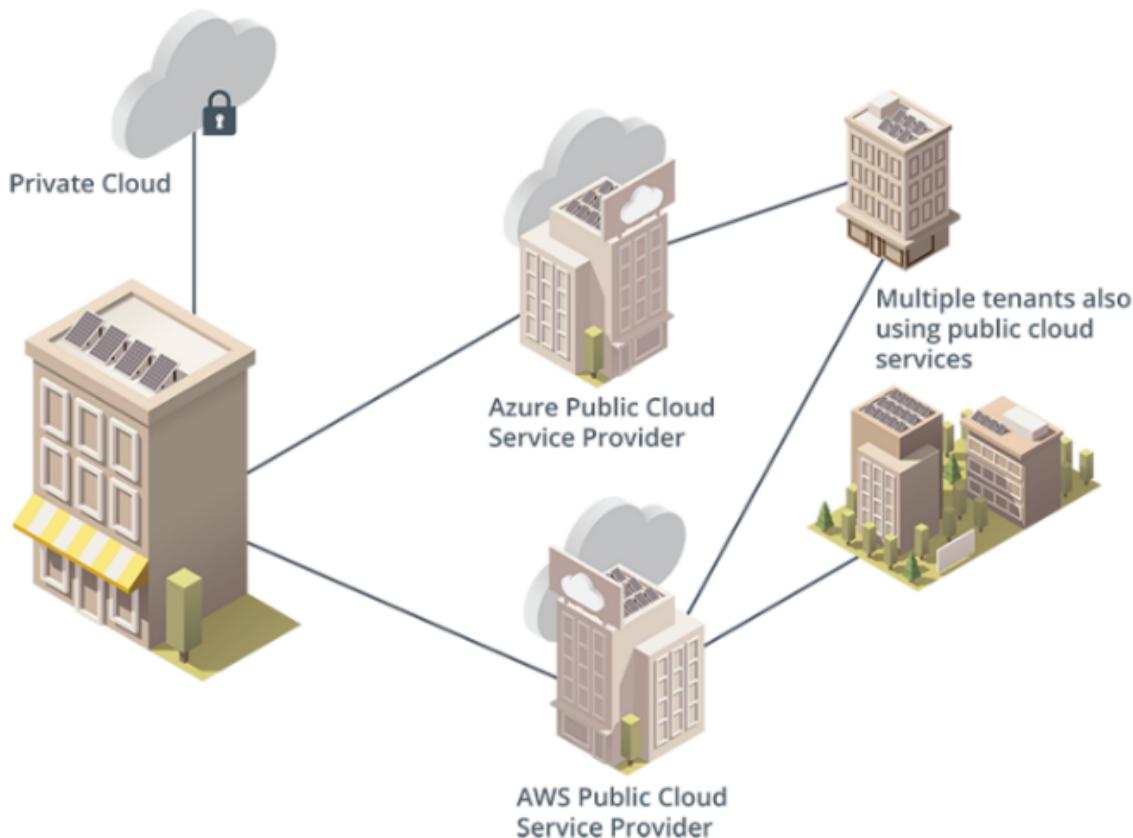
บริการที่เกี่ยวข้องกับ Hybrid Cloud

การใช้งานคลาวด์แบบผสมต้องอาศัยบริการในการโอนย้ายและผสานข้อมูลระหว่างระบบในองค์กร และระบบคลาวด์ภายนอก ตัวอย่างบริการ เช่น:

- บริการย้ายและประมวลผลข้อมูล (Data Transfer & Integration Services)
 - AWS DataSync
 - Azure Data Factory
 - Google Cloud Dataflow
- บริการเครือข่ายเชื่อมต่อเฉพาะ (Dedicated Network Connectivity)
 - AWS Direct Connect
 - Azure ExpressRoute
 - Google Cloud Interconnect

2.6 มัลติคลาวด์ (Multi-Cloud)

มัลติคลาวด์คือการที่องค์กรเลือกใช้งานบริการคลาวด์จากผู้ให้บริการมากกว่าหนึ่งรายพร้อมกัน โดยทั่วไปมักเป็นผู้ให้บริการสาธารณูปายใหญ่ เช่น Amazon Web Services (AWS), Microsoft Azure และ Google Cloud Platform (GCP)



ภาพที่ 15 Multi-cloud

รูปแบบนี้เปิดโอกาสให้องค์กรสามารถเลือกใช้บริการที่เหมาะสมที่สุดในแต่ละด้านจากผู้ให้บริการต่าง ๆ ช่วยเพิ่มความยืดหยุ่น ลดต้นทุน และลดการพึ่งพาผู้ให้บริการรายเดียวที่มีมากเกินไป
ข้อดีของมัลติคลาวด์

- ลดการผูกขาดกับผู้ให้บริการรายเดียว (Vendor Lock-In)
- เลือกใช้บริการที่ดีที่สุดจากแต่ละผู้ให้บริการ
- ควบคุมตำแหน่งทางภูมิศาสตร์ของข้อมูลได้ดีขึ้น
- รองรับการบริหารความเสี่ยงและการกู้คืนจากภัยพิบัติ (Disaster Recovery)

- สามารถรวมบริการจาก คลาวด์ส่วนตัว และ ทรัพยากรในองค์กร เข้าด้วยกันได้ เทคโนโลยีสนับสนุน

การใช้งานมัลติคลาวด์ให้เกิดประสิทธิภาพสูงสุดมักจะอาศัยเครื่องมือ DevOps และเครื่องมือจัดการ อัตโนมัติ (Orchestration) เพื่อให้สามารถ:

- ปรับใช้ระบบที่ใช้ คอนเทนเนอร์ (Container) ได้อย่างรวดเร็วและสม่ำเสมอ
- ทำงานกับแอปพลิเคชันแบบ เซิร์ฟเวอร์เลส (Serverless)
- กระจายเนื้อหาด้วย เครือข่าย CDN และบริการคลาวด์อื่น ๆ

ข้อควรระวัง: มัลติคลาวด์ vs ไฮบริดคลาวด์

อย่าสับสนระหว่าง “มัลติคลาวด์” กับ “ไฮบริดคลาวด์”:

- ไฮบริดคลาวด์ (Hybrid Cloud)** คือการผสมผสานระหว่างระบบในองค์กร (On-Premises) กับบริการคลาวด์ของผู้ให้บริการเพียงรายเดียว
- มัลติคลาวด์ (Multi-Cloud)** คือการใช้บริการจากผู้ให้บริการคลาวด์หลายรายพร้อมกัน เช่น ใช้ Microsoft Azure สำหรับแอปพลิเคชัน และใช้ AWS สำหรับบริการฐานข้อมูล

2.7 คลาวด์ส่วนตัวเสมือน (Virtual Private Cloud: VPC)

คลาวด์ส่วนตัวเสมือน หรือ VPC คือ บริการที่ผู้ให้บริการคลาวด์สาธารณะ (Cloud Service Provider – CSP) จัดสรรทรัพยากรคลาวด์ไว้ในพื้นที่เฉพาะที่แยกออกจากทรัพยากรที่องค์กรอื่นใช้งานร่วมกัน โดยเป็นการแบ่งส่วน เครือข่ายแบบตรรกะ (Logical Isolation) ทำให้แม่จะอยู่ในโครงสร้างพื้นฐานของคลาวด์สาธารณะ แต่ทรัพยากร เหล่านั้นจะถือว่าเป็น “ส่วนตัว” สำหรับองค์กรผู้ใช้งาน และไม่มีการแบ่งใช้กับผู้อื่น องค์กรที่ใช้ VPC จะต้องรับผิดชอบในการบริหารจัดการคลาวด์ทั้งหมดด้วยตนเอง

ความแตกต่างระหว่าง VPC กับคลาวด์ส่วนตัว (Private Cloud)

เนื่องจากทั้ง VPC และ Private Cloud ต่างก็ให้บริการในลักษณะ “ส่วนตัว” อาจเกิดความสับสน ได้ง่าย จึงคร่าวๆ ความเข้าใจความแตกต่างที่สำคัญดังนี้:

- VPC**
 - เป็นการแยกส่วนของโครงสร้างพื้นฐานแบบ ตรรกะ (Logical Isolation)
 - อยู่ภายในโครงสร้างพื้นฐานของผู้ให้บริการคลาวด์สาธารณะ
 - รองรับการปรับขนาด (Scalability) ได้ดีตามความสามารถของ CSP
- Private Cloud**
 - เป็นการแยกส่วนแบบ กายภาพ (Physical Isolation)
 - อาจตั้งอยู่ในดาต้าเซ็นเตอร์ภายในขององค์กร, ดาต้าเซ็นเตอร์ที่ใช้ร่วมกันในชุมชน, หรือของ ผู้ให้บริการ
 - มีข้อจำกัดด้านการขยายตัวตามขนาดของฮาร์ดแวร์ที่มีอยู่

2.8 การใช้งานร่วมกันหลายผู้เช่า (Multitenancy)

แนวคิดของ Multitenancy เป็นพื้นฐานของการให้บริการคลาวด์แบบสาธารณะ โดยหมายถึง การที่ผู้ใช้งานหลายราย (เรียกว่า ผู้เช่า หรือ Tenants) ใช้ทรัพยากร่วมกัน ซึ่งทรัพยากรเหล่านี้เป็นของ ผู้ให้บริการคลาวด์ (Cloud Service Provider – CSP) และอยู่ภายใต้การบริหารจัดการของผู้ให้บริการเพียงรายเดียว

แนวคิดนี้ตรงข้ามกับการให้บริการแบบ คลาวด์ส่วนตัวเสมือน (VPC) ที่ออกแบบให้แต่ละองค์กร แยกทรัพยากรออกจากกันอย่างสิ้นเชิง ในทางกลับกัน Multitenancy ช่วยให้เกิดประสิทธิภาพด้านต้นทุน จากการใช้ทรัพยากร่วมกัน ซึ่งเป็นหนึ่งในข้อได้เปรียบหลักของการใช้คลาวด์สาธารณะ อย่างไรก็ตาม Multitenancy ก็นำมาซึ่งประเด็นที่ควรพิจารณาเพิ่มเติม เช่น:

- **ปัญหาด้านประสิทธิภาพการทำงาน (Performance):**

เนื่องจากทรัพย์สินแบ่งใช้ระหว่างองค์กร หากองค์กรหนึ่งมีการใช้งานเพิ่มขึ้นอย่างรวดเร็ว อาจส่งผลกระทบต่อประสิทธิภาพขององค์กรอื่นในระบบเดียวกัน

- **ข้อกังวลด้านความมั่นคงปลอดภัย (Security):**

การแบ่งใช้ทรัพยากร่วมกัน เช่น หน่วยประมวลผลกลาง (CPU) หรือหน่วยความจำ อาจนำไปสู่ ความเสี่ยงในการเข้าถึงข้อมูลที่ละเอียดอ่อนของผู้เช่ารายอื่นโดยไม่ได้ตั้งใจ เช่น ผ่านข้อมูลที่ค้างอยู่ใน แคชหรือในหน่วยความจำระบบ

เพื่อจัดการกับความเสี่ยงเหล่านี้ ผู้ให้บริการคลาวด์จะกำหนดข้อตกลงระดับการให้บริการ (Service Level Agreement – SLA) ซึ่งเป็นข้อตกลงระหว่างผู้ให้บริการกับลูกค้า โดยองค์กรควรตรวจสอบ SLA อย่างละเอียด เพื่อให้แน่ใจว่าข้อกำหนดด้านความมั่นคงปลอดภัย ความพร้อมใช้งาน ความสามารถในการปรับขนาด และประสิทธิภาพตรงตามความต้องการขององค์กร

ไม่เพียงแต่คลาวด์สาธารณะเท่านั้นที่ใช้แนวคิด Multitenancy โดยเดือน ฯ เช่น คลาวด์แบบไฮบริด (Hybrid Cloud), มัลติคลาวด์ (Multi-cloud) และ คลาวด์ชุมชน (Community Cloud) ก็ล้วน ออกแบบโดยใช้แนวคิดนี้ร่วมด้วย

เทคโนโลยีสำคัญที่สนับสนุน Multitenancy ได้แก่:

- การจำลองเสมือน (Virtualization)
- การใช้คอนเทนเนอร์ (Containerization)
- ระบบเครือข่ายเสมือน (Virtual Networking)

ซึ่งล้วนมีบทบาทสำคัญในการบริหารจัดการระบบที่ผู้ใช้งานหลายรายใช้ร่วมกันอย่างมีประสิทธิภาพ

2.9 การรู้จำบริการคลาวด์ (Recognize Cloud Services)

มีบริการคลาวด์หลากหลายรูปแบบที่สามารถใช้งานได้ในโมเดลการให้บริการคลาวด์ประเภทต่าง ๆ โดยมักมีบริการในรูปแบบ Public Cloud เช่น ของที่ Private และ Hybrid Cloud มักเป็นตัวเลือกเพิ่มเติม องค์กรมักตัดสินใจตามความต้องการว่าต้องการดูแลศูนย์ข้อมูลของตนเอง (หรือใช้แบบ Hybrid/Community

Cloud) หรือไม่ หากไม่จำเป็นต้องดูแลเอง องค์กรก็สามารถเลือกใช้บริการต่อไปนี้ในสภาพแวดล้อมแบบสาธารณะ (Public Cloud) ได้

Google Workspace

เดิมชื่อ G Suite หรือ Google Apps เป็นบริการแบบ Software as a Service (SaaS) ที่ให้บริการด้านการประมวลผลคำ ตารางคำนวณ ซอฟต์แวร์นำเสนอด้วย และบริการอื่น ๆ สำหรับการทำงานร่วมกันออนไลน์ Office 365

เป็นชุดซอฟต์แวร์สำนักงานของ Microsoft ในรูปแบบ SaaS ให้ความยืดหยุ่นสูง ใช้งานได้บนหลายแพลตฟอร์ม ติดตั้งและดูแลง่าย

Microsoft Fabric

เป็นโซลูชันด้าน Business Intelligence ของ Microsoft ในรูปแบบ SaaS ที่มีการใช้ปัญญาประดิษฐ์ (AI) เพื่อการวิเคราะห์ข้อมูลและสร้างคุณค่าสูงสุดจากข้อมูลทางธุรกิจ

Salesforce

เป็นโซลูชัน CRM (Customer Relationship Management) ในรูปแบบ SaaS ที่ได้รับความนิยมสูงรองรับการบริหารลูกค้า ด้านการตลาด การขาย และการบริการ

Dropbox

เป็นโซลูชันแบบ Storage as a Service (STaaS) ที่รองรับการจัดเก็บและเข้าถึงข้อมูลออนไลน์ข้ามแพลตฟอร์มที่ยังคงได้รับความนิยม

Digital Ocean

เป็นผู้ให้บริการคลาวด์ที่เน้นกลุ่มนักพัฒนา ให้บริการทรัพยากรที่สามารถปรับขนาดและติดตั้งได้อย่างรวดเร็ว เป็นตัวเลือกยอดนิยมสำหรับโฮสต์เว็บแอปพลิเคชัน ด้วยความเรียบง่ายในการใช้งาน

Rackspace

ให้บริการเซิร์ฟเวอร์บนคลาวด์ ฐานข้อมูล ระบบกระจายโหลด การจัดเก็บข้อมูล และบริการอื่น ๆ จุดเด่น คือ การบริการสนับสนุนที่เรียกว่า “Fanatical Support” ซึ่งได้รับการยอมรับอย่างสูง

Red Hat Cloud Suite

เป็นบริการคลาวด์จาก Red Hat ซึ่งเดิมเป็นผู้พัฒนา Linux สำหรับองค์กร โดย Red Hat Cloud Suite ประกอบด้วย 4 ผลิตภัณฑ์หลัก ได้แก่

- OpenStack Platform สำหรับสร้างคลาวด์แบบสาธารณะและส่วนตัว
- Virtualization สำหรับเสมือนจริงเซิร์ฟเวอร์คลาวด์
- Satellite สำหรับจัดการบริการคลาวด์
- OpenShift สำหรับบริหารจัดการ Kubernetes และคอนเทนเนอร์

หน่วยที่ 3 กลยุทธ์การปรับใช้คลาวด์ (Cloud Deployment Strategies)

เนื่องจากบริการคลาวด์มีความยืดหยุ่นและสามารถปรับขนาดได้สูง ตัวเลือกในการปรับใช้จึงมีความหลากหลายมากขึ้น นอกจากนี้ แนวทางการพัฒนาแบบ DevOps และการออกแบบแอปพลิเคชันด้วยสถาปัตยกรรมแบบไมโครเซอร์วิสในลักษณะ Cloud-Native ยังช่วยให้สามารถอัปเกรดแอปพลิเคชันได้อย่างรวดเร็วโดยไม่จำเป็นต้องใช้การอัปเดตเวอร์ชันใหม่ ซึ่งมักสร้างภาระกับองค์กรขนาดใหญ่

บทเรียนนี้จะกล่าวถึงกลยุทธ์การปรับใช้ (Deployment Strategies) ที่เกี่ยวข้องกับบริการและแอปพลิเคชันบนคลาวด์ ซึ่งรวมถึงแนวทางแบบ Blue-Green, Canary, Rolling และ In-Place ซึ่งแต่ละวิธีมีจุดเด่น จุดด้อย และความเหมาะสมกับสถานการณ์ที่แตกต่างกันไป

3.1 กลยุทธ์การปรับใช้ (Deployment Strategies)

เป็นสิ่งสำคัญที่ต้องจัดแนวบริการด้านไอทีทั้งหมดให้สอดคล้องกับเป้าหมายเชิงกลยุทธ์ขององค์กร โดยบริการคลาวด์เป็นหนึ่งในองค์ประกอบของแก็ตต้าลีกบริการไอทีที่ใหญ่กว่า การเข้าใจความต้องการของธุรกิจ สภาพแวดล้อมทางไอทีภายในองค์กร และเทคนิคการทดสอบที่หลากหลาย จะช่วยให้สามารถปรับใช้บริการคลาวด์ได้อย่างมีประสิทธิภาพ

การดำเนินการปรับใช้บริการคลาวด์ควรดำเนินไปอย่างเป็นลำดับ มีการบันทึกเป็นระบบ และคำนึงถึงงบประมาณ ความสำคัญของแต่ละงาน และประสบการณ์ขององค์กรกับระบบคลาวด์ การปรับใช้ที่ประสบความสำเร็จสูงสุดมักเป็นผลจากการจัดการที่ตอบโจทย์เป้าหมายของธุรกิจ

ก่อนเริ่มการปรับใช้หรือการโยกย้ายระบบไปยังคลาวด์ ควรรวบรวมความต้องการจากหน่วยงานต่างๆ และพิจารณาว่าบริการคลาวด์จะสามารถช่วยให้องค์กรบรรลุเป้าหมายทางธุรกิจได้อย่างไร หัวข้อที่ควรพิจารณา ได้แก่ ฮาร์ดแวร์ ซอฟต์แวร์ การบูรณาการระบบ งบประมาณ ข้อกำหนดการปฏิบัติตามกฎหมาย ระดับการให้บริการ (SLA) ความปลอดภัย และเครือข่าย

การเข้าใจความต้องการของธุรกิจหมายถึงการมีส่วนร่วมของทุกฝ่ายภายในองค์กรในขั้นตอนการวางแผน การใช้งานคลาวด์

ประเด็นหนึ่งที่สำคัญ คือ หลายองค์กรกำหนดนโยบาย “Cloud First” คือ ให้พิจารณาการใช้ระบบคลาวด์สำหรับบริการใหม่ทุกประเภทเป็นอันดับแรก อย่างไรก็ตาม “Cloud First” ไม่ได้หมายถึง “Cloud Only” เพราะไม่ใช่ทุกบริการหรือแอปพลิเคชันจะเหมาะสมกับคลาวด์ โดยเฉพาะแอปพลิเคชันแบบเดิมที่มีข้อจำกัด ความต้องการของผู้ใช้และธุรกิจ

ผู้ใช้ภายในองค์กรและฝ่ายบริหารอาจมีมุมมองความต้องการที่แตกต่างกัน เช่น ผู้ใช้อาจต้องการอินเทอร์เฟซที่คุ้นเคย ประสิทธิภาพในการใช้งาน ความสามารถในการเข้าถึงจากหลายแพลตฟอร์ม (Windows, macOS, Linux, iOS, Android) และการสนับสนุนทางเทคนิคที่ดี

ในขณะที่ฝ่ายบริหารขององค์กรอาจให้ความสำคัญกับประเด็นเรื่องต้นทุน (ทั้ง CapEx และ OpEx) ความสามารถในการบูรณาการกับระบบเดิม ความสามารถในการขยายระบบ (Scalability) และการปฏิบัติตามกฎหมายหรือมาตรฐาน (Compliance)

ความกังวลของผู้ใช้:

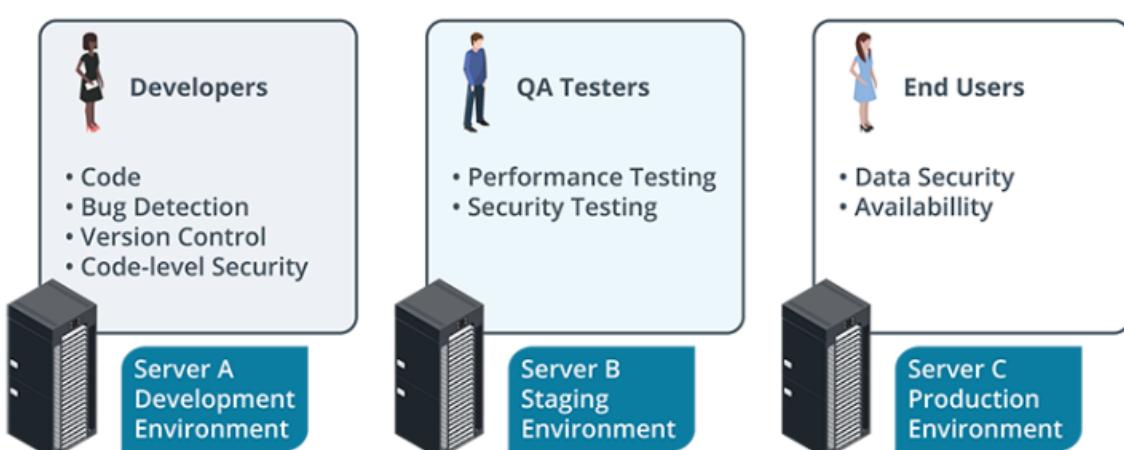
- อินเทอร์เฟซที่ใช้งานง่ายและคุ้นเคย
- พังก์ชันการทำงาน
- การสนับสนุนทางเทคนิค
- ความเสถียรและประสิทธิภาพของระบบ

ความกังวลของฝ่ายบริหาร:

- ต้นทุน (ทั้ง CapEx และ OpEx)
- การผสานรวมกับระบบเดิม
- การปฏิบัติตามข้อกำหนดทางกฎหมาย
- การสนับสนุนและความสามารถในการปรับขยายระบบ

3.2 การแยกสภาพแวดล้อมทางไอที (Separate IT Environments)

ผู้ดูแลระบบมักจะแยกสภาพแวดล้อมทางไอทีออกจากกัน เพื่อแยกระบบที่ใช้งานจริง (Production) ออกจากกระบวนการพัฒนา (Development) ตัวอย่างที่พบบ่อย ได้แก่ สภาพแวดล้อมการพัฒนา (Development Environment), สภาพแวดล้อมทดสอบก่อนใช้งานจริงหรือสเตจจิ่ง (Staging Environment) และสภาพแวดล้อมการใช้งานจริง (Production Environment) ซึ่งแต่ละสภาพแวดล้อมอาจมีรูปแบบแตกต่างกันไปตามลักษณะขององค์กร และใช้ทรัพยากรไม่เท่ากัน (ส่งผลต่อค่าใช้จ่ายโดยตรง)



Development, staging, and production environments. (Images © 123RF.com)

ภาพที่ 16 Separate IT Environments

สภาพแวดล้อมการพัฒนา (Development Environment)

เป็นพื้นที่ที่นักพัฒนาใช้ในการเขียนโค้ด ตรวจหาบັກ จัดการเวอร์ชันของซอฟต์แวร์ และใช้มาตราการด้านความปลอดภัยระดับโค้ด ในระบบคลาวด์ อาจผสมผสานการใช้บริการแบบแพลตฟอร์ม (PaaS) เพื่อการพัฒนาและโครงสร้างพื้นฐาน (IaaS) เพื่อการทดสอบ

สภาพแวดล้อมสเตจจิงหรือทดสอบ (Staging Environment)

มักเรียกอีกชื่อหนึ่งว่า Quality Assurance (QA) เป็นพื้นที่สำหรับทีมทดสอบในการตรวจสอบความถูกต้องของแอปพลิเคชันและบริการคลาวด์ การทดสอบอาจครอบคลุมทั้งด้านความปลอดภัยและประสิทธิภาพ และสามารถดำเนินการในรูปแบบอัตโนมัติหรือแม่นวนລ (หรือทั้งสองแบบ)

ในระบบคลาวด์ สภาพแวดล้อมสเตจจิงมักใช้บริการแบบ IaaS ซึ่งอาจต้องขยายขนาดมากกว่าปกติ ในช่วงทดสอบการทำงานแบบเต็มระบบ ทำให้ค่าใช้จ่ายในสเตจจิงอาจไม่สูงเท่าที่ตั้งทุนในระบบจริง

สภาพแวดล้อมใช้งานจริง (Production Environment)

เป็นพื้นที่ที่ให้บริการแก่ผู้ใช้งานจริง โดยมีมาตรการด้านความปลอดภัยและความพร้อมใช้งานของระบบที่สูงกว่าสภาพแวดล้อมอื่น ๆ หากสภาพแวดล้อมนี้อยู่บนระบบคลาวด์ จะต้องมีระบบมอนิเตอร์และความสามารถในการปรับขยายตามการใช้งานจริง (Scalability)

การจัดการสภาพแวดล้อมทั้งสามนี้ช่วยให้องค์กรสามารถแยกความรับผิดชอบและลดความเสี่ยงจากข้อผิดพลาดในการพัฒนาหรือทดสอบที่อาจส่งผลต่อผู้ใช้งานจริง โดยระบบคลาวด์มีความยืดหยุ่นสูงในด้าน การให้บริการด้วยตนเอง (Self-Service) และ การปรับขนาดตามต้องการ (Scalability) ทำให้สามารถจัดสรรทรัพยากรให้เหมาะสมกับแต่ละสภาพแวดล้อมได้ในราคากลางๆ ทั้งยังสนับสนุนกระบวนการปรับใช้แอปพลิเคชันใหม่หรือการโยกย้ายระบบจาก On-premises ไปยัง Cloud ได้อย่างมีประสิทธิภาพ

3.3 รูปแบบการเผยแพร่แอปพลิเคชัน (Application Release Models)

มีรูปแบบการเผยแพร่และปรับใช้ซอฟต์แวร์หลายประเภทที่ช่วยให้ผู้ดูแลระบบสามารถติดตั้งและจัดการแอปพลิเคชันได้อย่างถูกต้องและมีประสิทธิภาพ ไม่ได้จำกัดเฉพาะแอปพลิเคชันสำหรับผู้ใช้งานทั่วไป เช่น ชุดโปรแกรมสำนักงานหรือเว็บเบราว์เซอร์เท่านั้น แต่ยังครอบคลุมไปถึงแอปพลิเคชันระดับองค์กร เช่น ฐานข้อมูล และบริการเครือข่าย



Automated deployment example

การปรับใช้ซอฟต์แวร์อย่างมีการจัดการสามารถช่วยลดความผิดพลาด เช่น การกำหนดค่าผิดพลาด (Misconfiguration), ความไม่เข้ากันของระบบ (Incompatibility), หรือการใช้ฟีเจอร์ที่ถูกยกเลิก (Deprecated Features) นอกจากนี้ยังช่วยจัดสรรทรัพยากรได้อย่างเหมาะสม กำหนดสิทธิ์ในการเข้าถึงอย่างถูกต้อง ปรับขนาดระบบให้พอดี และลดปัญหาในขั้นตอนการแก้ไขปัญหา (Troubleshooting)

หากไม่มีการควบคุมที่ดี อาจทำให้ระบบเกิด Downtime หรือแอปพลิเคชันทำงานได้ไม่เต็มประสิทธิภาพ ส่งผลเสียต่อธุรกิจโดยตรง เครื่องมือสำคัญที่นิยมใช้ในการจัดการการปรับใช้ซอฟต์แวร์ ได้แก่ Ansible, Docker, Git, Kubernetes และ Terraform

ระบบอัตโนมัติ (Automation) เป็นองค์ประกอบสำคัญของการเผยแพร่แอปพลิเคชัน โดยช่วยลดข้อผิดพลาดจากมนุษย์ เพิ่มความยืดหยุ่น และสร้างรูปแบบการทำงานที่สม่ำเสมอ การใช้งานร่วมกับแนวทาง DevOps และกระบวนการ Continuous Integration / Continuous Deployment (CI/CD) จะช่วยให้การปรับใช้มีความน่าเชื่อถือยิ่งขึ้น

ตัวอย่างของระบบอัตโนมัติ:

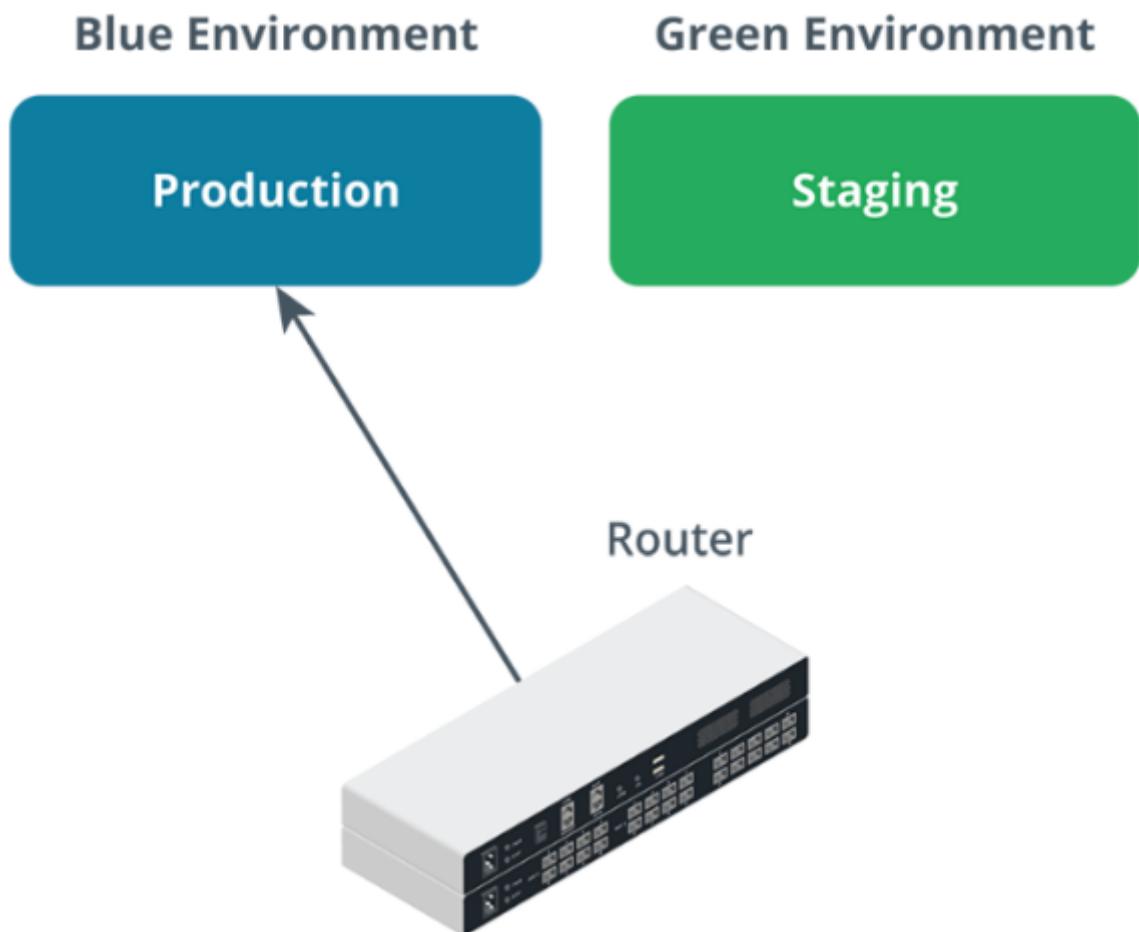
- โค้ดต้นฉบับใน Repository ถูกส่งไปยังบริการจัดเก็บของผู้ให้บริการคลาวด์
 - จากนั้นผ่านเข้าสู่ Pipeline เพื่อประเมินผล
 - และสุดท้ายถูกปรับใช้ลงบน Virtual Server
- ระบบอัตโนมัติเหล่านี้มักอาศัยไฟล์กำหนดค่าที่เขียนในภาษาอย่าง YAML (Yet Another Markup Language) หรือ JSON (JavaScript Object Notation)

รูปแบบการเผยแพร่แอปพลิเคชันที่ใช้กันทั่วไปมี 4 แบบ ได้แก่:

- รูปแบบ Blue-Green
- รูปแบบ Canary
- รูปแบบ Rolling
- รูปแบบ In-place

3.4 กลยุทธ์การปรับใช้แบบ Blue-Green (Blue-Green Deployment Strategy)

แนวทาง Blue-Green Deployment เป็นรูปแบบหนึ่งของการแยกสภาพแวดล้อมการพัฒนา ทดสอบ และใช้งานจริงออกจากกัน โดยจะมีการสร้างสภาพแวดล้อมที่เหมือนกันสองชุด คือ “Blue” และ “Green” ซึ่งในขณะใดขณะหนึ่งจะมีเพียงชุดเดียวเท่านั้นที่ทำงานที่เป็นสภาพแวดล้อมการผลิต (Production)



*The blue-green release model environment has one production environment and one for staging.
(Images © 123RF.com)*

ภาพที่ 18 Blue-Green Deployment Strategy

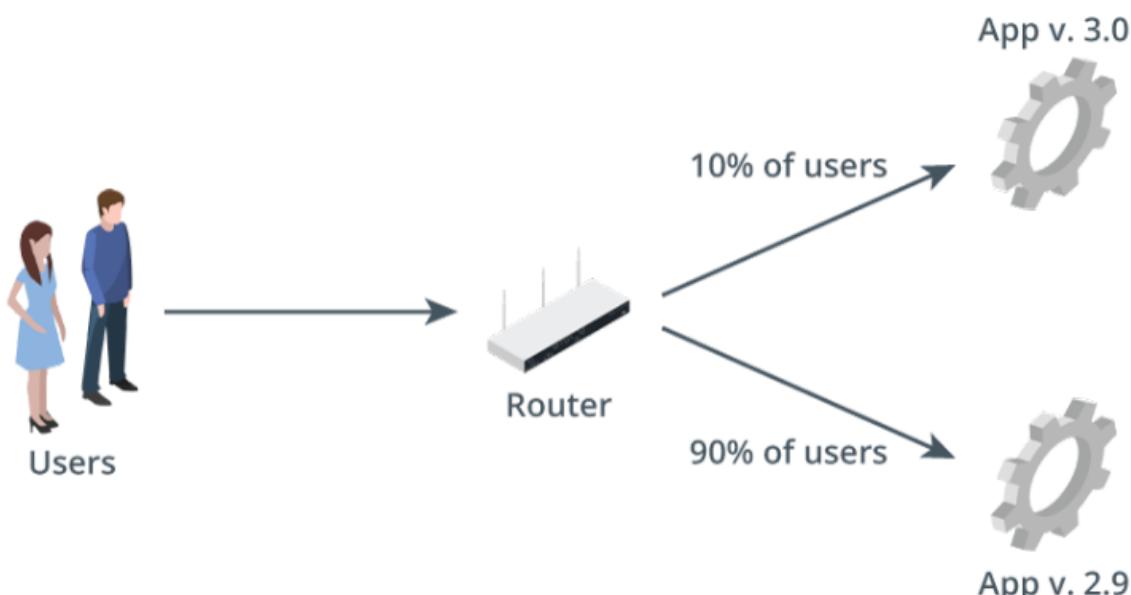
สภาพแวดล้อมที่ไม่ได้ใช้งานจะถูกใช้เป็นพื้นที่สำหรับการเตรียมความพร้อมของซอฟต์แวร์เวอร์ชันถัดไป เมื่อการทดสอบและการตรวจสอบคุณภาพ (Quality Assurance หรือ QA) เสร็จสิ้นแล้ว ระบบจะสลับให้ผู้ใช้ทั้งหมดไปใช้งานในสภาพแวดล้อมใหม่นั้นทันที

เป้าหมายหลักของรูปแบบการปรับใช้นี้คือ การลดเวลา Downtime ให้น้อยที่สุดหรือเป็นศูนย์ ตัวอย่าง: หากมีแอปพลิเคชันเว็บที่ให้บริการอยู่ในสภาพแวดล้อม “Blue” มาเป็นเวลาหนึ่งปี และมีการพัฒนาเวอร์ชันใหม่ขึ้นในสภาพแวดล้อม “Green” หลังจากการทดสอบเรียบร้อยแล้ว ระบบจะเปลี่ยนเส้นทางการเข้าถึงของผู้ใช้ทั้งหมดไปยังสภาพแวดล้อม “Green” โดยทันที และสภาพแวดล้อม “Blue” จะกลับเป็นสำรองและรอใช้ในการปรับใช้ครั้งถัดไป

การสลับกันระหว่างสองสภาพแวดล้อมนี้จะดำเนินไปต่อทางจารชีวิตของแอปพลิเคชัน ทำให้สามารถปรับใช้เวอร์ชันใหม่ได้อย่างราบรื่น ปลอดภัย และไม่รบกวนการใช้งานของผู้ใช้

3.5 กลยุทธ์การปรับใช้แบบ Canary (Canary Deployment Strategy)

Canary Deployment เป็นแนวทางที่คล้ายกับ Blue-Green แต่มีความแตกต่างตรงที่ผู้ใช้งานจะถูกย้ายจากเวอร์ชันเดิมไปยังเวอร์ชันใหม่ อย่างค่อยเป็นค่อยไป ไม่ใช่ทั้งหมดในคราวเดียว โดยในระยะแรกจะมีเพียงกลุ่มผู้ใช้งานน้อยที่ได้รับการเข้าถึงซอฟต์แวร์เวอร์ชันใหม่ เพื่อทำการทดสอบและให้ข้อเสนอแนะก่อนที่จะขยายการใช้งานไปยังผู้ใช้ทั้งหมด



Canary deployments direct some users to the new application and others to the older version.

ภาพที่ 19 Canary Deployment Strategy

Canary Deployment เป็นส่วนหนึ่งของแนวทาง Continuous Integration และ Continuous Delivery (CI/CD) ซึ่งช่วยลด Downtime ได้อย่างมาก เนื่องจากการเปลี่ยนแปลงจะมีผลต่อผู้ใช้งานเพียงกลุ่มเล็ก ๆ ในแต่ละช่วงเวลา จึงง่ายต่อการย้อนกลับ (Rollback) หากเกิดปัญหา ตัวอย่าง: หากมีผู้ใช้งาน 100 คน ระบบอาจเลือกให้ 10 คนแรกใช้งานแอปพลิเคชันเวอร์ชันใหม่ (เช่น 3.0) ในขณะที่อีก 90 คนยังคงใช้เวอร์ชันเดิม (2.9) หากมีปัญหา ระบบจะง่ายต่อการย้อนกลับ (Rollback) หากเกิดปัญหา ต่อไป

Canary Deployment อาจทำได้ทั้งแบบ Rolling Deployment หรือแบบ Side-by-Side ขึ้นอยู่กับลักษณะระบบ

3.6 กลยุทธ์การปรับใช้แบบ Rolling (Rolling Deployment Strategy)

Rolling Deployment เป็นแนวทางที่อัปเดตซอฟต์แวร์หรือบริการ แบบค่อยเป็นค่อยไปตามลำดับ โดยจะทำการปรับใช้ทีละเครื่อง หรือทีละกลุ่มย่อยของเซิร์ฟเวอร์ จากนั้นจึงทำการทดสอบและวิเคราะห์ เพื่อให้มั่นใจว่าไม่มีปัญหา ก่อนจะดำเนินการกับเครื่องถัดไป

แนวทางนี้ช่วยลด Downtime ได้ดี และหลีกเลี่ยงความเสี่ยงจากการล้มเหลวทั้งหมดในครั้งเดียว หากมีปัญหาเกิดขึ้น ระบบสามารถหยุดกระบวนการอัปเดตไว้ชั่วคราวและย้อนกลับได้ทันที

มักใช้ในระบบแบบ Cluster โดยจะอัปเดตโหนด (Node) ทีละตัว ขณะที่โหนดอื่นยังให้บริการ ตามปกติ การเปลี่ยนเส้นทาง trafic จากโหนดบานานเชอร์หรือคลัสเตอร์เมเนเจอร์ จะถูกจัดการอย่างระมัดระวัง เพื่อให้มั่นใจว่า trafic จะค่อย ๆ วิ่งไปยังระบบที่อัปเดตแล้ว

ข้อดีของ Rolling Deployment ได้แก่:

- ง่ายต่อการย้อนกลับหากพบปัญหา
- ลดผลกระทบต่อผู้ใช้งาน
- มีการตรวจสอบสถานะระหว่างกระบวนการ
- ให้การควบคุมที่แม่นยำมากยิ่งขึ้น

แม้ว่า Rolling Deployment จะสามารถใช้ผู้ใช้งานเป็น “Canary Tester” ได้เช่นกัน แต่โดยหลักการแล้วถือเป็นกลยุทธ์ที่แยกต่างหากจาก Canary Deployment

3.7 กลยุทธ์การปรับใช้แบบ In-Place (In-Place Deployment Strategy)

In-Place Deployment เป็นแนวทางที่มุ่งเน้นการปรับใช้แอปพลิเคชัน โดยไม่เปลี่ยนแปลงโครงสร้างพื้นฐานของระบบ ที่มิໄວที่จะหยุดการทำงานของแอปพลิเคชันเวอร์ชันเดิม จากนั้นจึงติดตั้งเวอร์ชันใหม่ในตำแหน่งเดียวกัน และเปิดให้ผู้ใช้งานเข้าใช้บริการอีกครั้ง

แนวทางนี้ ไม่จำเป็นต้องลงทุนเพิ่มในด้านโครงสร้างพื้นฐาน เช่น การตั้งค่าเซิร์ฟเวอร์หรือระบบเครือข่ายเพิ่มเติม เพราะระบบเดิมยังคงถูกใช้งานอยู่

ข้อแตกต่างที่สำคัญจากกลยุทธ์ Canary คือ ใน In-Place Deployment ไม่มีการทดสอบจากกลุ่มผู้ใช้งานย่อยก่อน การติดตั้งและการใช้งานเกิดขึ้นทันทีหลังจากแอปพลิเคชันใหม่ถูกนำมาใช้งาน ข้อควรระวัง:

- In-Place Deployment จะส่งผลให้ระบบมี Downtime เนื่องจากต้องหยุดบริการก่อนติดตั้งเวอร์ชันใหม่
- จำเป็นต้องวางแผน Downtime อย่างรอบคอบ และแจ้งผู้ใช้งานล่วงหน้าเพื่อหลีกเลี่ยงผลกระทบต่อธุรกิจ

เหมาะสมสำหรับระบบที่สามารถหยุดชั่วคราวได้โดยไม่กระทบต่อการให้บริการอย่างมีนัยสำคัญ หรือระบบภายในที่ไม่ได้ให้บริการแบบ 24 ชั่วโมง

สรุปการวางแผนบริการคลาวด์ (Summary: Planning Cloud Services)

การออกแบบระบบให้รองรับคลาวด์ (Cloud-native design) ถือเป็นองค์ประกอบสำคัญในการบริหารจัดการแอปพลิเคชันยุคใหม่ โดยอาศัยคุณสมบัติของคลาวด์ที่มีการกระจายตัว (distributed), คิดค่าบริการตามการใช้งาน (metered) และสามารถปรับขยายได้ตามต้องการ (scalable) ส่งผลให้มีพัฒนาสามารถสร้างโปรแกรมที่มีโครงสร้างแบบแยกส่วน (loosely coupled microservices) ซึ่งปรับขยาย อัปเดต และปรับใช้งานได้อย่างยืดหยุ่น

องค์กรที่ใช้บริการคลาวด์จำเป็นต้องตัดสินใจเลือกรูปแบบการติดตั้ง ไม่ว่าจะเป็นการใช้คลาวด์สาธารณะ (public cloud), ศูนย์ข้อมูลภายในองค์กร (private cloud) หรือรูปแบบผสมผสาน (hybrid cloud) ซึ่งแต่ละรูปแบบมีผลกระทบต่อค่าใช้จ่าย ความมั่นคงปลอดภัย และความพร้อมใช้งาน

นอกจากนี้ ทีมพัฒนายังสามารถเลือกกลยุทธ์การปรับใช้แอปพลิเคชันที่เหมาะสม เพื่อให้การเปิดตัวบริการหรือฟีเจอร์ใหม่เกิดขึ้นได้อย่างราบรื่น ไม่เสียดุด และไม่กระทบผู้ใช้ เช่น กลยุทธ์แบบ Blue-Green, Canary, Rolling และ In-Place ซึ่งช่วยให้องค์กรสามารถให้บริการแอปพลิเคชันที่ทันสมัย ปลอดภัย และเชื่อถือได้สูงสุด

บทที่ 3

การจัดเตรียมและย้ายทรัพยากรไปยังคลาวด์ (Provisioning and Migrating Cloud Resources)

บทนำของบทเรียน

การติดตั้งทรัพยากรในระบบคลาวด์มักเริ่มต้นจากการใช้เครื่องเสมือน (Virtual Machines: VMs) ที่อยู่บนคลาวด์ ซึ่งระบบเหล่านี้ต้องการองค์ประกอบด้านการประมวลผล เช่น หน่วยประมวลผลกลาง (CPU), หน่วยประมวลผลกราฟิก (GPU), หน่วยความจำ (Memory), ระบบจัดเก็บข้อมูล (Storage) และการเชื่อมต่อเครือข่าย (Network Support) โดยการกำหนดค่าของระบบจะขึ้นอยู่กับความต้องการด้านประสิทธิภาพ การปฏิบัติตามข้อกำหนด (Compliance) และความปลอดภัย รวมถึงงบประมาณที่มีอยู่

เส้นทางการย้ายข้อมูลมาตຽานมักเริ่มจากการย้ายเซิร์ฟเวอร์จริงหรือเสมือนที่มีอยู่ในองค์กรไปยังเครื่องเสมือนในคลาวด์ (Cloud VMs) ในบางกรณี องค์กรอาจย้ายทรัพยากรระหว่างผู้ให้บริการคลาวด์ เพื่อเพิ่มประสิทธิภาพด้านต้นทุนหรือฟีเจอร์ ส่วนการย้ายแอปพลิเคชันอาจง่ายเพียงแค่นำโปรแกรมไปติดตั้งใหม่ในระบบคลาวด์ (Rehosting) หรืออาจถึงขั้นเขียนใหม่ทั้งหมด หรือยุติการใช้งานแอปพลิเคชันนั้น

วัตถุประสงค์ของบทเรียนนี้

คุณจะได้เรียนรู้เกี่ยวกับการ:

- จัดเตรียมทรัพยากรด้านการประมวลผล การจัดเก็บข้อมูล และเครือข่ายในคลาวด์
- ย้ายทรัพยากรไปยังระบบคลาวด์และระหว่างผู้ให้บริการคลาวด์

หน่วยที่ 1 การจัดเตรียมทรัพยากรในคลาวด์ (Provision Cloud Resources)

การเริ่มต้นติดตั้งทรัพยากรในระบบคลาวด์ต้องมาจากการทำความเข้าใจบริการและแอปพลิเคชันที่ต้องการสนับสนุนก่อน โดยทรัพยากรเหล่านี้จะมีข้อกำหนดเฉพาะด้านการจัดสรร เช่น การประมวลผล (Compute), การจัดเก็บข้อมูล (Storage), ระบบเครือข่าย (Networking) และสเปกตั้งต้นอื่น ๆ ที่เกี่ยวข้อง

บทเรียนนี้จะอธิบายคำศัพท์สำคัญที่เกี่ยวข้องกับกระบวนการจัดเตรียมทรัพยากรในคลาวด์ วิเคราะห์ตัวแปรที่ใช้ในการเลือกองค์ประกอบที่เหมาะสม และอธิบายขั้นตอนการติดตั้งทรัพยากร

1.1 ทรัพยากรประมวลผล (Compute Resources)

ทรัพยากรประมวลผลหมายถึงการจัดสรรหน่วยประมวลผลกลาง (CPU), หน่วยความจำ (Memory), ที่จัดเก็บข้อมูล (Storage) และการเชื่อมต่อเครือข่าย (Networking) โดยผู้ดูแลระบบจะสร้างโซลูชันการประมวลผลตามความต้องการเฉพาะ เช่น การตั้งค่าเครื่องเสมือน (VM) เพื่อรับแอปพลิเคชันหนึ่ง

ซึ่งต้องกำหนดสเปกของ CPU หรือ GPU, ขนาดของหน่วยความจำ, การจัดเก็บ และการเชื่อมต่อเครือข่าย ให้เพียงพอเพื่อให้บรรลุเป้าหมายด้านประสิทธิภาพ ความพร้อมใช้งาน และความปลอดภัย ซึ่งกระบวนการเหล่านี้เรียกว่า **การจัดสรรทรัพยากร (Resource Allocation)**

ในอดีต ผู้ดูแลระบบมักจะต้องสร้างเซิร์ฟเวอร์ตามข้อกำหนดเฉพาะ แต่ในปัจจุบันการใช้เทคโนโลยีเสมือนจริง (Virtualization) ทำให้สามารถปรับขยายทรัพยากรได้อย่างง่ายดาย ทั้งแบบปรับด้วยตนเอง และอัตโนมัติ

CPU ที่กำหนดให้ VM เรียกว่า **vCPU (Virtual CPU)** ส่วนการประมวลผลภาพหรือกราฟิกจะใช้ GPU หน่วยประมวลผลและหน่วยความจำมีความสัมพันธ์กันใกล้ชิด ขณะที่การจัดเก็บข้อมูลและเครือข่ายมักจัดเป็นองค์ประกอบแยกต่างหาก

ความต้องการของระบบจะถูกจัดตามลักษณะการใช้งาน เช่น บางแอปพลิเคชันต้องการประมวลผล และหน่วยความจำจำนวนมาก (Compute-Intensive) ขณะที่บางแอปเน้นการถ่ายโอนข้อมูลหรือการเข้าถึง迪สก์มากกว่า (Data-Intensive) ผู้ดูแลระบบจึงจำเป็นต้องเข้าใจลักษณะเฉพาะของแอปพลิเคชันในการออกแบบ VM

ผู้ดูแลระบบคลาวด์ต้องมีสิทธิ์ที่เหมาะสมในการสร้างและแก้ไขทรัพยากรบนคลาวด์ เพราะการตั้งค่าผิดพลาดอาจนำไปสู่ปัญหาการให้บริการหรือความไม่พร้อมใช้งาน

1.2 การจัดการเทคโนโลยีเสมือนจริง (Manage Virtualization)

การใช้เซิร์ฟเวอร์เสมือนคือแกนหลักของบริการ Infrastructure as a Service (IaaS) ซึ่งเริ่มต้นจาก Hypervisor ที่ทำหน้าที่เป็นชั้นจัดการระหว่างฮาร์ดแวร์ของเซิร์ฟเวอร์จริงกับ VM ที่กำหนดค่าไว้โดย Hypervisor มี 2 ประเภทคือ

- **Type 1 Hypervisor:** ทำงานโดยตรงบนฮาร์ดแวร์ของเซิร์ฟเวอร์ โดยไม่ต้องมีระบบปฏิบัติการกลาง เช่น Microsoft Hyper-V, VMware ESXi, KVM
- **Type 2 Hypervisor:** ทำงานเป็นแอปพลิเคชันบนระบบปฏิบัติการของไอดีท์ เช่น Oracle VirtualBox, VMware Workstation

ผู้ให้บริการระบบคลาวด์มักใช้ Type 1 Hypervisor บนเซิร์ฟเวอร์จริง (Bare Metal) ซึ่งให้ประสิทธิภาพสูงและเข้าถึงฮาร์ดแวร์ได้โดยตรง

การใช้ทรัพยากรเกินจริง (Oversubscription) เป็นอีกเทคนิคนึงที่ช่วยให้สามารถจัดสรรทรัพยากรเกินจากที่ฮาร์ดแวร์มีอยู่จริงได้ เช่น หากเซิร์ฟเวอร์มี RAM 64 GB อาจตั้งค่า VM ไว้ 10 เครื่องโดยแต่ละเครื่องมี RAM 8 GB รวม 80 GB ได้ หากทุกเครื่องไม่ใช้ RAM เต็มพร้อมกัน ระบบก็ยังทำงานได้โดยไม่ส่งผลกระทบ

1.3 การจัดสรรทรัพยากรประมวลผล (Provision Compute Resources)

ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers – CSPs) มักมีชุดค่ากำหนดล่วงหน้าสำหรับเครื่องเสมือน (Virtual Machine – VM) เพื่อให้การติดตั้งและใช้งานเป็นไปอย่างสะดวกรวดเร็ว โดยเครื่อง VM เหล่านี้มีการกำหนด CPU, GPU, หน่วยความจำ, พื้นที่เก็บข้อมูล และการเชื่อมต่อเครือข่ายที่เหมาะสมกับบทบาทเฉพาะของงาน และถึงแม้สามารถปรับแต่งได้ แต่โดยปกติค่ามาตรฐานก็มักเพียงพอแล้วสำหรับการใช้งานทั่วไป

ตัวอย่างเช่น AWS (Amazon Web Services) ให้บริการประเภทของอินสแตนซ์ที่หลากหลายตามลักษณะงาน ได้แก่

- **General Purpose:** สมดุลระหว่าง CPU, หน่วยความจำ และเครือข่าย เหมาะกับการใช้งานทั่วไป
- **Compute Optimized:** เหมาะสำหรับงานที่ต้องการการประมวลผลสูง เช่น Web Server หรือ Game Server
- **Memory Optimized:** เหมาะสำหรับการประมวลผลข้อมูลขนาดใหญ่ในหน่วยความจำ เช่น Big Data
- **Accelerated Computing:** ใช้งานร่วมกับ GPU หรืออุปกรณ์เร่งการประมวลผล เช่น AI หรือการประมวลผลภาพ
- **Storage Optimized:** เหมาะสำหรับงานที่เน้นการอ่านเขียนข้อมูลสูง เช่น Database
- **HPC Optimized:** ใช้ในงานประมวลผลประสิทธิภาพสูง เช่น Simulation หรือ Machine Learning ขั้นสูง

การเลือกอินสแตนซ์ต้องพิจารณาให้สอดคล้องกับความต้องการของแอปพลิเคชันและงบประมาณ เช่น อินสแตนซ์ C6g, C6gd, หรือ C6gn เหมาะสมกับงานประมวลผลสูง ขณะที่ I3 หรือ I3en เหมาะกับงานที่ต้องใช้ I/O หนัก เช่น ฐานข้อมูล

ผู้ดูแลระบบสามารถปรับขนาดทรัพยากรได้ภายหลัง แต่โดยทั่วไปต้องหยุดการทำงานของอินสแตนซ์ ก่อนปรับ โดยอาจมีเหตุผลในการปรับขนาด เช่น การคาดการณ์ผิดพลาด การเติบโตเกินคาด หรือการลดต้นทุน ซึ่งการตั้งค่าผิดพลาดอาจส่งผลให้ต้นทุนสูงขึ้น ประสิทธิภาพลดลง หรือเกิด Downtime ได้

1.3.1 การประมวลผลแบบหลายเธรด (Multithreading)

หนึ่งในปัจจัยสำคัญของการประเมินทรัพยากรคือการประมวลผลแบบหลายเธรด ซึ่งช่วยให้ระบบสามารถแบ่งงานไปยังหลายคอร์ของ CPU ได้พร้อมกัน โดยการจัดสรร vCPU ให้ VM จะคำนวณจาก:

$$vCPU = \text{จำนวนคอร์} \times \text{จำนวนเธรดต่อคอร์}$$

ตัวอย่างเช่น อินสแตนซ์ f1.2xlarge มี vCPU 8 เพราะมี 4 คอร์ และแต่ละคอร์รองรับ 2 เธรด ส่วน f1.4xlarge มี vCPU 16 จาก 8 คอร์ \times 2 เธรด

ดังนั้น การเลือกประเภทอินสแตนซ์และกำหนดค่าทรัพยากรให้เหมาะสมจึงเป็นสิ่งจำเป็น เพื่อให้แอปพลิเคชันทำงานได้มีประสิทธิภาพ คุ้มค่า และสอดคล้องกับเป้าหมายขององค์กร

1.4 การจัดสรรทรัพยากรหน่วยประมวลผลกราฟิก (Provision Graphics Processing Unit Resources)

ในบางกรณี งานประมวลผลสามารถถ่ายจากหน่วยประมวลผลกลาง (CPU) ไปยังตัวประมวลผลร่วม หรือฮาร์ดแวร์เร่งการประมวลผลอื่นได้ หนึ่งในตัวเลือกยอดนิยมคือการใช้หน่วยประมวลผลกราฟิก (GPU) ซึ่งเหมาะสมอย่างยิ่งกับเครื่องเสมือน (VM) ที่ใช้ในการสตรีมเกม การทำงานกราฟิกระยะไกล หรือการพัฒนาโมเดล 3 มิติ

ผู้ให้บริการคลาวด์หลัก เช่น Amazon Web Services (AWS), Microsoft Azure และ Google Cloud Platform (GCP) ต่างรองรับการใช้ GPU ของ NVIDIA

1.4.1 vGPU แบบใช้ร่วมกัน (Shared vGPUs)

คล้ายกับ CPU ที่สามารถจำลองเป็น vCPU สำหรับ VM, GPU ก็สามารถแบ่งให้ใช้งานในรูปแบบ Virtual GPU (vGPU) ได้ เช่นกัน การใช้ vGPU เหมาะกับงานที่ต้องใช้การประมวลผลกราฟิกจำนวนมาก เช่น การแสดงผลภาพ การเรนเดอร์ และการเรียนรู้ของเครื่อง (Machine Learning)

1.4.2 GPU แบบ Pass-through (Pass-through GPUs)

ในแบบ Pass-through จะมอบการ์ดกราฟิกทั้งใบให้กับ VM เดียวโดยเฉพาะ ส่งผลให้ประสิทธิภาพของ VM คงที่ และสูงมาก แม้การจัดการฮาร์ดแวร์จะซับซ้อนขึ้น ตัวอย่างเช่น AWS Marketplace มีอินสแตนซ์ VM สำหรับงานกราฟิกที่ติดตั้งล่วงหน้าพร้อม NVIDIA GPU ให้เลือก เช่น แพลตฟอร์ม Ubuntu Linux สำหรับเกมบนคลาวด์

ผู้ดูแลระบบสามารถเลือกอินสแตนซ์ VM ที่รองรับ GPU ได้ เมื่อกับการเลือกอินสแตนซ์ที่เหมาะสมกับหน่วยความจำหรือการจัดเก็บข้อมูล

1.4.3 การจัดการความเร็วสัญญาณนาฬิกา (Manage Clock Speed)

ความเร็วสัญญาณนาฬิกา (Clock Speed) เป็นหนึ่งในตัวชี้วัดประสิทธิภาพของ CPU การปรับความเร็วสัญญาณนาฬิกาให้เหมาะสมกับประเภทของงาน เช่น การประมวลผลทางวิทยาศาสตร์หรือการเรนเดอร์ภาพขึ้นสูง จะช่วยให้ใช้งานทรัพยากรได้มีประสิทธิภาพมากขึ้น

การพัฒนาแอปพลิเคชันสำหรับงานประมวลผลสมรรถนะสูง (High-Performance Computing – HPC) อาจทำให้ต้องใช้ CPU เพิ่มขึ้นตามปริมาณงานที่เพิ่มขึ้น ดังนั้นการจัดสรรทรัพยากร GPU อย่างเหมาะสมจึงเป็นสิ่งสำคัญในการรองรับการใช้งานลักษณะนี้

1.5 ข้อกำหนดด้านการประมวลผล (Compute Requirements)

ทรัพยากรการประมวลผลมักประกอบด้วยการกำหนดค่าหน่วยประมวลผลกลาง (CPU) และหน่วยความจำ (RAM) เป็นหลัก ในบางบริบท คำว่า “การประมวลผล (Compute)” ยังครอบคลุมถึงการจัดเก็บข้อมูลและเครือข่ายด้วย เนื่องจากหน่วยความจำมีความสัมพันธ์ใกล้ชิดกับการตั้งค่า CPU จึงมักถูกจัดการร่วมกัน

เช่นเดียวกับการกำหนดค่า Virtual CPU (vCPU) ผู้ให้บริการคลาวด์ (CSPs) จะจัดสรรหน่วยความจำ (RAM) ให้กับอินสแตนซ์ VM แต่ละประเภท การเลือกใช้อินสแตนซ์ประเภทที่เน้นหน่วยความจำ (Memory-Optimized Instances) เป็นวิธีง่ายที่สุดในการตั้งค่า VM สำหรับแอปพลิเคชันหรือบริการที่ใช้ RAM มาก

ตัวอย่าง เช่น Microsoft Azure และ Amazon Web Services (AWS) Elastic Compute Cloud (EC2) ต่างมีอินสแตนซ์ที่ออกแบบมาเพื่อรองรับงานที่ต้องการหน่วยความจำสูง เช่น Azure Mv2-series ซึ่งเหมาะสมกับเวิร์กโหลดที่ใช้หน่วยความจำอย่างเข้มข้น

ผู้ดูแลระบบจะต้องปิดเครื่องอินสแตนซ์ก่อนจึงจะสามารถเปลี่ยนประเภทของอินสแตนซ์ เพื่อปรับขนาดของ RAM หรือ vCPU ได้ใน AWS Management Console การเปลี่ยนแปลงนี้อาจเพิ่มหรือลดปริมาณทรัพยากรที่จัดสรรให้ VM ขึ้นอยู่กับข้อมูลประสิทธิภาพที่ได้รับเมื่อเวลาผ่านไปหรือจากการเปลี่ยนแปลงของความต้องการทางธุรกิจ ทั้งนี้ยังต้องอยู่ภายใต้ขีดจำกัดของทรัพยากรที่กำหนดไว้สำหรับแต่ละ VM

1.6 ข้อกำหนดด้านการจัดเก็บข้อมูล (Storage Requirements)

ผู้ดูแลระบบคลาวด์มีหน้าที่ในการจัดสรรทรัพยากรการจัดเก็บข้อมูล โดยข้อมูลในระบบคลาวด์สามารถจัดเก็บได้ในรูปแบบที่แตกต่างกัน เช่น การจัดเก็บแบบไฟล์ (File), แบบบล็อก (Block), และแบบอ็อบเจกต์ (Object) ซึ่งแต่ละวิธีมีข้อดีและข้อจำกัดต่างกัน การเลือกใช้จึงขึ้นอยู่กับลักษณะของข้อมูลและวิธีการเข้าถึงข้อมูลนั้น ผู้ดูแลระบบจำเป็นต้องมีสิทธิ์ที่เหมาะสมในการจัดสรรและปรับขนาดความจุของพื้นที่จัดเก็บ

ข้อมูลแต่ละประเภทมีความต้องการในการเข้าถึงที่ต่างกัน บางข้อมูลต้องสามารถเข้าถึงได้อย่างรวดเร็ว โดยไม่มีความหน่วง (Latency) ในขณะที่บางข้อมูลถูกเก็บไว้ในรูปแบบของการเก็บถาวร (Archived) เพื่อตอบสนองต่อข้อกำหนดด้านการเก็บรักษาข้อมูลในอุตสาหกรรมต่างๆ

1.6.1 I/O คืออะไร?

คำว่า I/O (Input/Output) หมายถึงการกระทำในการอ่านหรือเขียนข้อมูลเพียงครั้งเดียว ของสื่อจัดเก็บข้อมูล ค่าที่ใช้วัดประสิทธิภาพของการจัดเก็บคือ IOPS (Input/Output Operations per Second) หรือจำนวนคำสั่งอ่าน/เขียนต่อวินาที ซึ่งสัมพันธ์กับค่าความหน่วงเวลา (Latency) ทั้งสองค่านี้ใช้ในการประเมินประสิทธิภาพของระบบจัดเก็บข้อมูล

เทคโนโลยีการจัดเก็บแต่ละแบบมีความสามารถในการรองรับ I/O และ Latency ที่ต่างกัน เช่นเดียวกับบริการอื่น ๆ การเลือกใช้งานต้องมีการพิจารณาเรื่องต้นทุนและประสิทธิภาพอย่างเหมาะสม

1.6.2 รายละเอียดของ Solid-State Drive (SSD) และ Hard Disk Drive (HDD)

อินสแตนซ์ของ VM ต้องมีพื้นที่จัดเก็บข้อมูลเช่นเดียวกับคอมพิวเตอร์ทั่วไป โดยสามารถเลือกใช้ได้ทั้ง SSD และ HDD แบบตั้งเดิม การเลือกใช้อุปกรณ์จัดเก็บข้อมูลนั้นขึ้นอยู่กับวัตถุประสงค์ของ VM และบริการที่ใช้งาน เพื่อเพิ่มประสิทธิภาพและความจุ พร้อมทั้งควบคุมต้นทุน นอกจากนี้ยังอาจมีการใช้เทคนิคการทำซ้ำข้อมูล (Redundancy) เพื่อความมั่นใจด้านความพร้อมใช้งาน

1.6.3 Bucket คืออะไร?

ในระบบคลาวด์ “Bucket” คือทรัพยากรที่ใช้ในการจัดเก็บข้อมูล โดยมีคุณสมบัติที่ส่งผลต่อความสามารถในการใช้งาน ต้นทุน ฟีเจอร์ และการกำหนดค่า ผู้ดูแลระบบสามารถควบคุมสิทธิ์ในการเข้าถึง Bucket ได้

คุณสมบัติทั่วไปของ Bucket ในระบบคลาวด์ เช่น:

- มีชื่อที่ไม่ซ้ำกันในระดับโลก
- ถูกจัดเก็บในตำแหน่งทางภูมิศาสตร์ที่ระบุไว้ (ใกล้กับแหล่งที่ใช้ข้อมูล)
- คิดค่าบริการตามประเภทของพื้นที่จัดเก็บ ความจุ ความเร็วในการเข้าถึง การถ่ายโอนข้อมูลออก ฯลฯ
- ปรับขนาดได้ตามความต้องการ
- รองรับคุณสมบัติตามข้อกำหนด เช่น การเก็บรักษาข้อมูล ความปลอดภัย การเข้าถึง ฯลฯ

Bucket ถูกใช้งานในบริการและแอปพลิเคชันคลาวด์ต่าง ๆ เช่น:

- ไฟล์เรื้อรังแบบ Static
- แอปพลิเคชันที่สร้างข้อมูลเพื่อจัดเก็บ
- ไฟล์บันทึก (Log) จาก Load Balancer หรือเว็บไซต์
- ข้อมูลด้านการวิเคราะห์หรือสินค้าคงคลังที่ดึงจาก Bucket อื่น

แม้ว่าการจัดเก็บข้อมูลจะเป็นส่วนหนึ่งของการจัดสรรทรัพยากรการประมวลผล แต่ด้วยลักษณะที่เฉพาะตัวและมีความซับซ้อน จึงเหมาะสมที่จะมีการอธิบายอย่างละเอียดในบทเรียนถัดไปเกี่ยวกับการตั้งค่าและการปรับแต่งระบบจัดเก็บข้อมูล

1.7 ข้อกำหนดด้านเครือข่าย (Network Requirements)

ระบบเครือข่ายในระบบคลาวด์มีลักษณะคล้ายคลึงกับเครือข่ายแบบตั้งเดิมในองค์กร (On-premises) โดยมีการสร้างและควบคุมเส้นทางการสื่อสารระหว่างบริการและผู้ใช้งาน เครือข่ายเสมือน (Virtual Network)

มีข้อดีหลายประการ เช่น เดียวกับเครื่องเสมือน (VM) ได้แก่ ความสามารถในการขยายระบบ ปรับแต่งได้ง่าย และการควบคุมต้นทุน

บริการคลาวด์สามารถสนับสนุนเครือข่ายเสมือนเข้ากับเครือข่ายทางกายภาพในองค์กร หรือสร้างเครือข่ายเสมือนแบบสมมูลรูณ์ในระบบคลาวด์ ผู้ดูแลระบบเครือข่ายสามารถแบ่งแยกเครือข่ายออกเป็นส่วน ๆ โดยใช้ VLAN (Virtual LAN) พร้อมด้วยการทำหน้าต่างสวิตช์และเราเตอร์ รวมถึงมีหน้าที่บริหารจัดการไฟร์wall (Firewall), ไฟร์wall สำหรับเว็บแอปพลิเคชัน (Web Application Firewall), เครือข่ายจัดส่งคอนเทนต์ (CDN) และการเชื่อมต่อ VPN (Virtual Private Network)

มีตัวเลือกมากมายสำหรับการทำหน้าต่างเครือข่ายเมื่อต้องจัดสรรบริการคลาวด์ โดยเครื่อง VM ในคลาวด์จะใช้การเดาเชื่อมต่อเครือข่ายเสมือน (Virtual NIC) หนึ่งใบหรือมากกว่า เพื่อสื่อสารกับทรัพยากรอื่น ๆ การ์ดเหล่านี้จะมีการทำหน้าต่าง IP ความสามารถในการกำหนดเส้นทาง (Routing) และฟังก์ชันอื่น ๆ

ผู้ดูแลระบบสามารถสร้างเครือข่ายแบบเปิดสาธารณะ สำหรับเว็บเซอร์วิสหรือแอปพลิเคชัน ที่ให้บริการแก่ลูกค้า ตลอดจนสร้างเครือข่ายภายในสำหรับพนักงานหรือระบบภายในองค์กรที่อยู่ในคลาวด์ บางส่วนหรือทั้งหมด ซึ่งเครือข่ายเหล่านี้มักมีความซับซ้อน ประกอบด้วยการกระจายโหลด (Load Balancing), การรองผ่านไฟร์wall, การกำหนดเส้นทาง, การตั้งค่า IPv4 หรือ IPv6 และการจำลองข้อมูล (Replication)

การตั้งค่าเครือข่ายสามารถกำหนดได้ตั้งแต่ขั้นตอนเริ่มต้นของการติดตั้ง VM และสามารถปรับแต่งได้ภายหลังเพื่อให้ระบบทำงานได้อย่างเหมาะสม

1.8 การติดตั้งเครื่องเสมือน (Deploy a Virtual Machine)

เมื่อเข้าใจองค์ประกอบต่างๆ แล้ว คุณสามารถเริ่มจัดสรร VM ในระบบคลาวด์ได้ ขั้นตอนจะต่างกันไปตามผู้ให้บริการคลาวด์และประเภทของ VM ที่คุณต้องการสร้าง โดยต้องคำนึงถึงสิทธิ์และการควบคุม การเข้าถึงในการบริหารจัดการอินสแตนซ์ และอาจสามารถจัดสรรทรัพยากรกันจากที่ระบบรองรับได้ในบางกรณี ขั้นตอนทั่วไปของการติดตั้งอินสแตนซ์ประเภท t2.micro บน AWS มีดังนี้:

1. ไปที่หน้าแดชบอร์ด EC2 และเลือก “Launch Instance”
2. ตั้งชื่ออินสแตนซ์ในหน้าต่าง “Launch an instance”
3. เลือกระบบปฏิบัติการ เช่น Linux, macOS หรือ Windows
4. เลือกประเภทอินสแตนซ์ที่กำหนด CPU และหน่วยความจำ เช่น แบบ General Purpose
5. กำหนด Key Pair สำหรับการยืนยันตัวตน การตั้งค่าเครือข่าย และข้อกำหนดด้านพื้นที่จัดเก็บข้อมูล

ผู้ให้บริการคลาวด์ เช่น VMware, Oracle และ Azure มีตัวเลือกอินสแตนซ์หลายขนาดที่ปรับให้เหมาะสมกับวัตถุประสงค์เฉพาะ

1.9 การจำลองเสมือนแบบไฮเปอร์คอนเวอร์เจนซ์ (Hyperconverged Virtualization)

การจำลองเสมือนแบบบูรณาการ (Converged Virtualization) คือการรวมหน่วยประมวลผลกลาง (CPU), หน่วยความจำ, พื้นที่เก็บข้อมูล และการกำหนดค่าเครือข่ายเข้าไว้ด้วยกันในระบบเดียว แต่ยังสามารถแยกองค์ประกอบทั้งสิ่งออกจากกันได้

ในขณะที่ การจำลองเสมือนแบบไฮเปอร์คอนเวอร์เจนซ์ (Hyperconverged Virtualization) จะรวมทั้งสิ่งระบบย่อยเข้าด้วยกันอย่างแนบแน่น ไม่สามารถแยกออกจากกันได้ การปรับใช้งานแบบไฮเปอร์คอนเวอร์เจนซ์มักใช้อาร์ดแวร์ทั่วไป (off-the-shelf) ซึ่งหมายความว่าการสร้างคลาวด์ส่วนตัวและดาต้าเซ็นเตอร์ขององค์กร

โซลูชันไฮเปอร์คอนเวอร์เจนซ์ทำให้การติดตั้งและดูแลระบบง่ายขึ้น เพราะโครงสร้างทั้งหมดสามารถจัดการได้เป็นหน่วยเดียว ซึ่งง่ายกว่าการกำหนดค่า CPU, GPU, หน่วยความจำ, พื้นที่เก็บข้อมูล และเครือข่ายแยกกัน

ผู้ให้บริการระบบโครงสร้างพื้นฐานแบบไฮเปอร์คอนเวอร์เจนซ์ (HCI) ที่เป็นที่รู้จัก เช่น

- StarWind Hyperconvergence appliances
- Nutanix ที่ใช้สถาปัตยกรรมซอฟต์แวร์เพื่อระบบไฮบริด
- VMware HCI ที่ทำงานร่วมกับ vSphere
- Dell VxRail ระบบไฮร์ดแวร์

สิ่งสำคัญคือต้องเข้าใจว่าไฮเปอร์คอนเวอร์เจนซ์คือการรวม CPU, หน่วยความจำ, พื้นที่เก็บข้อมูล และเทคโนโลยีเครือข่ายเข้าด้วยกัน ไม่ใช่การตั้งค่าแยกของแต่ละองค์ประกอบเหล่านี้

การจัดสรรทรัพยากรระบบคลาวด์มักเน้นไปที่การปรับใช้ทรัพยากรเสมือน แต่ยังมีประเด็นอื่น ๆ ที่ต้องพิจารณาเมื่อวางแผนและปรับใช้บริการคลาวด์

1.10 ข้อกำหนดด้านประสิทธิภาพ (Performance Requirements)

บริการและแอปพลิเคชันบนคลาวด์หลายประเภทมีข้อกำหนดด้านประสิทธิภาพขั้นต่ำ และผู้ใช้งานทั้งภายในและภายนอกองค์กรต่างก็มีความคาดหวังในเรื่องนี้ ตัวอย่างข้อกำหนด เช่น

- ความเร็วในการเข้าถึงทรัพยากร: การกำหนดค่า compute, storage และ network มีผลต่อความเร็วในการเข้าถึงข้อมูล
- การเลือกใช้ประเภท instance ให้เหมาะสมกับความต้องการของระบบ
- การจัดวางข้อมูลและบริการให้อยู่ใกล้ผู้ใช้งาน: เช่น การใช้ CDN เพื่อให้บริการได้เร็วขึ้น หรือใช้ edge computing
- ค่าความหน่วงของเครือข่าย (Latency): ส่งผลต่อความเร็วในการใช้งานแอปพลิเคชัน
- การใช้ load balancer หรือการแยกเซกเมนต์เครือข่ายเพื่อลด latency
- การอ่าน/เขียนข้อมูล (Storage I/O): เลือกระบบเก็บข้อมูลให้เหมาะสมกับประเภทการใช้งาน

- การเข้าถึงฐานข้อมูล: เลือก instance ที่เหมาะสมสำหรับการใช้สัมภาระฐานข้อมูลเพื่อประสิทธิภาพสูงสุด

1.11 ข้อกำหนดด้านการปฏิบัติตามข้อกำหนด (Compliance Requirements)

องค์กรต้องเข้าใจและปฏิบัติตามข้อกำหนดต่าง ๆ ที่เกี่ยวข้องกับข้อมูลและการใช้งานคลาวด์ โดยข้อกำหนดจะแตกต่างกันไปตามประเภทข้อมูล พื้นที่ และอุตสาหกรรม เช่น

- กฎหมาย: ตรวจสอบข้อกฎหมายที่เกี่ยวข้องกับความเป็นส่วนตัวและการจัดเก็บข้อมูล
- ข้อกำหนดทางกฎหมายเช่น HIPAA, Sarbanes-Oxley, GDPR, PCI DSS
- ข้อกำหนดเฉพาะประเทศ เช่น FedRAMP, ITAR, EAR ซึ่งใช้กับองค์กรที่ดำเนินการระหว่างประเทศ
- กฎหมายขององค์กร: เช่นนโยบายภายในหรือพันธกิจขององค์กร
- ข้อกำหนดท้องถิ่น: อาจมีข้อกำหนดเกี่ยวกับพัฒนา ผลิต หรือเสียง สำหรับดาต้าเซ็นเตอร์ของคลาวด์ส่วนตัว

ผู้ให้บริการคลาวด์อาจจำกัดบริการบางอย่างในบางภูมิภาค ดังนั้นควรศึกษาให้เข้าใจเกี่ยวกับความพร้อมให้บริการในแต่ละพื้นที่ด้วย

1.12 ข้อกำหนดด้านความปลอดภัย (Security Requirements)

ธุรกิจทุกประเภทต้องเผชิญกับความท้าทายด้านความปลอดภัยไซเบอร์ เช่น การโจมตีแบบปฏิเสธการให้บริการ (DDoS), แรนซัมแวร์ และการขโมยข้อมูลระบุตัวตน กลยุทธ์ที่ใช้ปกป้องทรัพยากรในองค์กรแบบ on-premises ส่วนใหญ่สามารถประยุกต์ใช้กับทรัพยากรบนคลาวด์ได้ เช่นกัน ตัวอย่างข้อกำหนดด้านความปลอดภัย เช่น

- ข้อมูลภายในขององค์กร: เช่น การโอนย้ายข้อมูล การเงิน และกระบวนการทางธุรกิจ ต้องได้รับการปกป้อง
- ทรัพย์สินทางปัญญา: เช่น แบบจำลองผลิตภัณฑ์และแอปพลิเคชันที่พัฒนาเอง ต้องป้องกันไม่ให้รั่วไหล
- ความเป็นส่วนตัวของลูกค้า: เช่น ชื่อผู้ใช้ ข้อมูลติดต่อ ประวัติการซื้อ ข้อมูลบัตรเครดิต และข้อมูลสุขภาพ
- ความเป็นส่วนตัวของพนักงาน: เช่น ป้องกันการรั่วไหลของข้อมูลส่วนตัว การโจมตีแบบฟิชชิ่ง และ social engineering

องค์กรอาจเลือกใช้รูปแบบการให้บริการคลาวด์แตกต่างกันตามระดับความต้องการด้านความปลอดภัย เช่น เลือกใช้คลาวด์แบบส่วนตัว (Private Cloud) หรือแบบผสม (Hybrid Cloud) เพื่อควบคุมข้อมูลได้มากขึ้น หากกังวลเรื่องความเป็นส่วนตัวของลูกค้าและพนักงาน อาจเลือกผู้ให้บริการคลาวด์ที่ผ่านการรับรองมาตรฐานความปลอดภัยและความเป็นส่วนตัว

ข้อกำหนดความปลอดภัยบางอย่างยังขึ้นอยู่กับระดับภัยมิภा�คหรือประเทศด้วย ผู้ให้บริการคลาวด์บางรายอาจจำกัดบริการบางประเภทในพื้นที่ที่มีข้อจำกัดด้านกฎหมายหรือข้อบังคับ

1.13 ข้อกำหนดด้านความพร้อมใช้งาน (Availability Requirements)

ผู้ใช้งานในปัจจุบันคาดหวังว่าระบบจะพร้อมใช้งานตลอดเวลา การสร้างความพร้อมใช้งานสูง (High Availability) มีค่าใช้จ่ายที่อาจคุ้มค่าหรือไม่ขึ้นอยู่กับความสำคัญของทรัพยากร ตัวอย่างข้อกำหนด เช่น

- ความพร้อมใช้งานสำหรับลูกค้า: ลูกค้าคาดหวังว่าจะสามารถเข้าถึงเว็บไซต์ ฐานข้อมูล หรือบริการสนับสนุนได้ตลอดเวลา
- ความพร้อมใช้งานสำหรับพนักงาน: การหยุดชะงักของระบบอาจกระทบต่อประสิทธิภาพการทำงานขององค์กร
- ความพร้อมใช้งานของบริการภายนอก: ระบบไม่ควรชอร์วิสหรือเชิร์ฟเวอร์ที่ทำงานร่วมกันต้องมีความพร้อมใช้งานสูงเสมอ

ผู้ให้บริการคลาวด์มีการอัปเดตและปรับปรุงบริการอย่างสม่ำเสมอ รวมถึงการยกเลิกบริการเก่าซึ่งอาจมีผลกระทบต่อแผนการปรับใช้ของคุณ

ควรวางแผนสำรองไว้สำหรับเหตุการณ์ที่อาจเกิดขึ้น เช่น ไฟดับหรืออินเทอร์เน็ตล้ม รวมถึงการวางแผนกู้คืนระบบ (Disaster Recovery) และแผนความต่อเนื่องทางธุรกิจ (Business Continuity Plan)

1.14 ข้อกำหนดด้านค่าใช้จ่าย (Cost Requirements)

ค่าใช้จ่ายในการปรับใช้และดูแลรักษาระบบคลาวด์มีความสำคัญเช่นเดียวกับด้านอื่น ๆ ของไอที ความแตกต่างของระบบคลาวด์อยู่ที่โครงสร้างต้นทุนที่เน้นการใช้งานแบบสมัครสมาชิกแทนการซื้ออาร์ดแวร์ หรือซอฟต์แวร์ถาวร ตัวอย่างข้อควรพิจารณา ได้แก่

- ค่าใช้จ่ายแบบลงทุน (CapEx) เทียบกับค่าใช้จ่ายในการดำเนินงาน (OpEx): ระบบคลาวด์เปลี่ยนแปลงรูปแบบการใช้จ่ายโดยให้ยืดหยุ่นตามการใช้งานจริง
- การเพิ่มประสิทธิภาพ: การปรับแต่งทรัพยากรให้เหมาะสมช่วยควบคุมงบประมาณ
- การรายงาน: การตรวจวัดและตรวจสอบการใช้ทรัพยากรช่วยให้องค์กรเข้าใจการใช้จ่ายและวางแผนงบประมาณได้แม่นยำมากขึ้น

หน่วยที่ 2 ประเด็นสำคัญของการย้ายระบบไปยังคลาวด์ (Aspects of Cloud Migration)

การย้ายระบบไปยังคลาวด์ (Cloud Migrations) เป็นกระบวนการที่มีความซับซ้อนสูง จึงแนะนำให้ใช้วิธีการแบบ เป็นลำดับขั้น (phased approach) เพื่อลดความเสี่ยงและจัดการการเปลี่ยนแปลงอย่างเป็นระบบ ในบทเรียนนี้ คุณจะได้เรียนรู้เกี่ยวกับประเภทของการย้ายระบบต่าง ๆ ได้แก่:

- **Rehosting (Lift and Shift):** การย้ายระบบแบบยกทั้งระบบเดิมขึ้นคลาวด์โดยไม่ปรับแก้ใดๆ หรือสถาปัตยกรรม
- **Re-platforming (Lift, Tinker, and Shift):** การย้ายระบบพร้อมกับปรับปรุงบางส่วน เช่น เปลี่ยนฐานข้อมูลเป็นแบบที่คลาวด์จัดการให้
- **Refactoring (Rip and Replace):** การเขียนแอปพลิเคชันใหม่ทั้งหมดเพื่อให้เหมาะสมกับสภาพแวดล้อมของคลาวด์โดยเฉพาะ

คุณจะได้ศึกษาเพิ่มเติมเกี่ยวกับ:

- การย้ายจาก เครื่องจริงไปยังเครื่องเสมือน (P2V: Physical to Virtual)
- การย้ายระหว่าง เครื่องเสมือนด้วยกัน (V2V: Virtual to Virtual)
- การย้ายจาก เครื่องเสมือนไปยังเครื่องจริง (V2P: Virtual to Physical)

รวมถึงประเด็นที่ต้องพิจารณาในการย้ายระบบจัดเก็บข้อมูล (Storage Migration) และฐานข้อมูล (Database Migration)

การย้ายแอปพลิเคชันและบริการขึ้นสู่คลาวด์สามารถเกิดขึ้นได้ในหลายรูปแบบ โดยมักจะดำเนินการเป็น รายแอปพลิเคชันหรือรายบริการ ไม่จำเป็นต้องย้ายระบบทั้งหมดขององค์กรในคราวเดียว ทั้งนี้มี 7 รูปแบบหลักของการย้ายแอปพลิเคชันขึ้นคลาวด์ ซึ่งจะอธิบายเพิ่มเติมในหัวข้อถัดไป

2.1 ประเภทของการย้ายระบบขึ้นคลาวด์ (Migration Types)

เมื่อกล่าวถึงการย้ายระบบขึ้นคลาวด์ (Cloud Migration) โดยทั่วไปเรามักนึกถึงการย้ายข้อมูลบริการ แอปพลิเคชัน และองค์ประกอบต่าง ๆ ของโครงสร้างพื้นฐานด้านไอทีไปยังระบบคลาวด์ การย้ายระบบเหล่านี้สามารถจำแนกได้เป็น 3 ประเภทหลัก ดังนี้:

2.1.1 การย้ายจากระบบภายในองค์กรขึ้นคลาวด์ (On-Premises-to-Cloud Migration)

เป็นรูปแบบที่พบบ่อยที่สุดในการย้ายระบบ โดยเป็นการโอนย้ายทรัพยากรจากระบบที่ติดตั้งภายในองค์กรไปยังทรัพยากรที่จัดสรรจากผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) การย้ายนี้อาจรวมถึงการย้ายแอปพลิเคชัน บริการ และเซิร์ฟเวอร์บางส่วนหรือทั้งหมดไปยังศูนย์ข้อมูลของผู้ให้บริการ นอกเหนือนี้ยังรวมถึงการย้ายจากระบบภายในองค์กรไปยังคลาวด์แบบส่วนตัว (Private Cloud) หรือแบบชุมชน (Community Cloud)

2.1.2 การย้ายจากคลาวด์หนึ่งไปยังอีกคลาวด์หนึ่ง (Cloud-to-Cloud Migration)

เกิดขึ้นเมื่อองค์กรต้องการรวมบริการคลาวด์ที่เคยใช้จากหลายผู้ให้บริการให้เหลือเพียงรายเดียว หรือเมื่อต้องการย้ายออกจากผู้ให้บริการเดิม ตัวอย่างเช่น องค์กรอาจย้ายแอปพลิเคชันและบริการทั้งหมดจาก Microsoft Azure และ AWS ไปยัง Google Cloud Platform (GCP)

ขั้นตอนโดยทั่วไปของกระบวนการนี้ ได้แก่:

1. สร้างบัญชีและสิทธิ์ในการเข้าถึง (IAM) ที่จำเป็น
2. สร้างการเชื่อมต่อ VPN ระหว่างผู้ให้บริการเดิมและ GCP
3. กำหนดการเข้าถึงเครือข่ายสำหรับ AWS
4. สร้างส่วนขยายคลาวด์ที่จำเป็น
5. ติดตั้งเครื่องมือย้ายระบบสำหรับ VM ของ Linux หรือ Windows
6. ทดสอบกระบวนการย้าย
7. ดำเนินการย้ายแบบเป็นกลุ่ม และทดสอบในแต่ละกลุ่ม

ผู้ให้บริการคลาวด์หลายรายมีเครื่องมือช่วยในการย้ายข้อมูล เช่น GCP มีบริการ **Migrate for Compute Engine** เพื่ออำนวยความสะดวกในการย้ายจาก AWS หรือ Azure ไปยัง GCP

อีกสถานการณ์หนึ่งคือการสร้าง **Multi-cloud Environment** ซึ่งเป็นการใช้งานบริการคลาวด์จากผู้ให้บริการสองรายขึ้นไป โดยมีการบริหารจัดการแบบรวมศูนย์ ตัวอย่างเช่น องค์กรใช้ AWS สำหรับเว็บแอปพลิเคชัน และใช้ Azure สำหรับแพลตฟอร์มนักพัฒนา (PaaS) หรืออาจเลือกใช้ AWS และ Azure สำหรับฐานข้อมูลร่วมกัน เพื่อป้องกันปัญหาความล้มเหลวจากผู้ให้บริการรายเดียว

ในทางตรงข้าม องค์กรบางแห่งอาจเลือกแนวทางที่ตrong กันข้ามด้วยการกำหนดให้ทั้งองค์กรใช้ผู้ให้บริการคลาวด์รายเดียว เพื่อความง่ายในการสนับสนุนและมาตรฐานเดียวกัน

หมายเหตุ: Multi-cloud แตกต่างจาก Hybrid Cloud โดยที่ Multi-cloud คือการใช้บริการจากผู้ให้บริการคลาวด์สาธารณะ (Public Cloud) ตั้งแต่สองรายขึ้นไป ส่วน Hybrid Cloud คือการผสมผสานระหว่างคลาวด์สาธารณะ คลาวด์ส่วนตัว และ/หรือระบบแบบดั้งเดิม (Legacy Systems)

2.1.3 การย้ายจากคลาวด์กลับไปยังระบบภายในองค์กร (Cloud-to-On-Premises)

ในบางกรณี องค์กรอาจตัดสินใจย้ายข้อมูลและบริการกลับจากคลาวด์มายังระบบภายในองค์กร สาเหตุอาจมาจากข้อกังวลเรื่องความปลอดภัย กฎหมายเบียบใหม่ หรือค่าใช้จ่ายที่สูงกว่าที่คาดไว้ หรือองค์กรอาจพบว่าการดำเนินธุรกิจในระบบคลาวด์ไม่ก่อให้เกิดประโยชน์เท่าที่ควร จึงเลือกที่จะออกแบบระบบใหม่ให้เหมาะสมกับความต้องการภายในองค์กรมากขึ้น

2.2 ระยะของการย้ายระบบขึ้นคลาวด์ (Phases of Cloud Migration)

กระบวนการย้ายระบบขึ้นคลาวด์ประกอบด้วย 4 ระยะหลัก ได้แก่ การประเมิน การวางแผน การดำเนินการ และการเพิ่มประสิทธิภาพ

- **การประเมิน (Assessment):** ทำความเข้าใจว่าแอปพลิเคชันและบริการใดที่มีอยู่ และสิ่งใดที่เหมาะสมจะย้ายขึ้นคลาวด์
- **การวางแผน (Planning):** การวางแผนเชิงกลยุทธ์สำหรับกระบวนการย้ายระบบ

- การดำเนินการ (Implementation): การถ่ายโอนข้อมูล บริการ และเซิร์ฟเวอร์ชั้นคลาوار์ด โดยมักดำเนินการเป็นลำดับขั้น
- การเพิ่มประสิทธิภาพและความมั่นคง (Optimization and Security): ปรับปรุงบริการให้ทำงานได้อย่างมีประสิทธิภาพ ปลอดภัย และคุ้มค่า

องค์กรส่วนใหญ่มักย้ายระบบเป็นช่วง ๆ โดยอาจรวมถึงการค้นพบทรัพยากร ตรวจสอบความเหมาะสมกับคลาوار์ด การแมปบริการกับโซลูชันคลาوار์ด การย้ายเซิร์ฟเวอร์และแอปพลิเคชัน และการดูแลหลังจากย้ายระบบแล้ว

2.3 การย้ายระบบเสมือน (Virtualization Migrations)

การย้ายระบบไปยังสภาพแวดล้อมเสมือน (ทั้งในองค์กรหรือบนคลาوار์ด) มีความซับซ้อนตามขนาดระบบปฏิบัติการ และแอปพลิเคชัน โดยมี 3 ประเภทหลัก ดังนี้:

- การย้ายจากเครื่องจริงไปยังเครื่องเสมือน (P2V: Physical to Virtual)
- การย้ายจากเครื่องเสมือนไปยังเครื่องจริง (V2P: Virtual to Physical)
- การย้ายระหว่างเครื่องเสมือน (V2V: Virtual to Virtual)

กระบวนการทั่วไปจะผ่าน 4 ขั้นตอน: การประเมิน การวางแผน การติดตั้ง และการเพิ่มประสิทธิภาพ

2.3.1 การย้ายจากเครื่องจริงไปยังเครื่องเสมือน (P2V)

เป็นรูปแบบที่พบบ่อย โดยเฉพาะเมื่อองค์กรเริ่มใช้คลาوار์ดหรือเสมือนจริงในระบบภายในองค์กรของประเทศ:

- แบบแมนนวล: ผู้ดูแลระบบติดตั้ง OS และแอปพลิเคชันบน VM และคัดลอกข้อมูลเอง
- แบบกึ่งอัตโนมัติ: ใช้เครื่องมือช่วยบางส่วน เช่น การกำหนดสเปก
- แบบอัตโนมัติ: เครื่องมือจัดการกระบวนการทั้งหมด

ขั้นตอนทั่วไป:

- บันทึกภาพระบบ (snapshot)
- ผู้จัดการ VM จัดสรรทรัพยากร
- โหลด snapshot ไปยัง VM ใหม่

ตัวอย่างเครื่องมือที่ใช้:

- GCP:** Migrate for Compute Engine
- Azure:** Azure Migrate: Server Migration
- AWS:** AWS Server Migration Service

ข้อกำหนดเบื้องต้น:

- รองรับระบบปฏิบัติการ เช่น RHEL, Ubuntu, Windows Server
- ต้องอัปเดตแพตช์ OS
- ปิดโปรแกรมแอนติไวรัส
- เปิดใช้งาน RDP หรือ SSH
- ต้องเตรียมเรื่องลิขสิทธิ์ให้พร้อม

2.3.2 การย้ายระหว่างเครื่องเสมือน (V2V)

การย้าย VM ระหว่างแพลตฟอร์มเสมือน เช่น จาก VMware ไป Hyper-V หรือระหว่าง CSP

เหตุผลในการทำ V2V:

- ใช้แพลตฟอร์มต่างกันระหว่างการพัฒนาและการใช้งานจริง
- เปลี่ยนผู้ให้บริการ hypervisor
- ทดสอบ VM บนหลายแพลตฟอร์ม
- ย้ายระหว่าง CSP ที่ใช้ hypervisor ต่างกัน

V2V ไม่จำเป็นต้องเป็นงานในคลาวด์เสมอไป อาจเกิดในระบบเสมือนภายในองค์กรด้วยเช่นกัน

2.4 กลยุทธ์การย้ายแอปพลิเคชันขึ้นคลาวด์ (Application Migration Strategies)

ในการย้ายระบบขึ้นคลาวด์ มีรูปแบบการย้ายที่หลากหลาย โดยจำแนกได้เป็น 8 ประเภท ซึ่งแต่ละแบบ มีผลต่อกระบวนการย้ายและผลกระทบต่อธุรกิจต่างกัน ดังนี้:

1. Rehost (ย้ายโดยไม่ปรับแก้ - Lift and Shift)

- ย้ายแอปพลิเคชันขึ้นคลาวด์โดยไม่มีการแก้ไขใด ๆ
- แอปพลิเคชันต้องพร้อมสำหรับคลาวด์ (เช่น ทำงานบน VM และทนต่อการสูญเสียเครือข่ายได้)
- เป็นวิธีที่รวดเร็วและง่ายที่สุด แต่หมายความว่าต้องปรับเปลี่ยนการรับคลาวด์ตั้งแต่ต้น

2. Replatform (ย้ายพร้อมการปรับเล็กน้อย - Lift, Tinker, and Shift)

- ปรับแต่งแอปพลิเคชันบางส่วนเพื่อให้รองรับคุณสมบัติของคลาวด์
- ไม่ต้องพัฒนาใหม่ทั้งหมด แต่ต้องปรับโครงสร้างบางส่วน เช่น การปรับฐานข้อมูลหรือ API
- หมายความว่าต้องปรับเปลี่ยนการรับคลาวด์อย่างน้อยหน่อย

3. Refactor (รื้อและสร้างใหม่ - Rip and Replace)

- ปรับโครงสร้างและรีฟาร์คแอปพลิเคชันใหม่ทั้งหมดเพื่อให้ทันสมัยและรองรับคลาวด์เต็มรูปแบบ
- ต้องใช้เวลาพัฒนาและต้นทุนสูง
- แอปที่ได้รับการ refactor จะทำงานร่วมกับบริการคลาวด์อื่น ๆ ได้ดีขึ้น

4. Rearchitect (ออกแบบใหม่ทั้งหมด)

- เปลี่ยนสถาปัตยกรรมของแอปทั้งหมด เช่น เปลี่ยนไปใช้ microservices หรือ container-based
- ต้องวิเคราะห์โค้ดเดิมอย่างละเอียดและอัปเดตให้เหมาะสม
- ใช้เวลานานและซับซ้อน แต่ช่วยเพิ่มความสามารถในการขยายระบบ

5. Repurchase (เปลี่ยนไปใช้ระบบใหม่ - Drop and Shop)

- เลิกใช้แอปเก่า แล้วซื้อแอปใหม่ที่พร้อมใช้งานบนคลาวด์ เช่น เปลี่ยนไปใช้ SaaS
- เหมาะสำหรับระบบเก่าที่ไม่สามารถรับคลาวด์หรือ virtualized environment ได้
- มักคุ้มค่ากว่า refactor ในระยะยาว

6. Retire (เลิกใช้โดยไม่แทนที่)

- ยุติการใช้งานแอปพลิเคชันเก่าที่ไม่จำเป็นหรือไม่มีหน่วยงานใดใช้งานแล้ว
- เป็นโอกาสดีในกระบวนการย้ายคลาวด์ที่จะกำจัดแอปที่ไม่จำเป็น

7. Retain (เก็บไว้ใช้งานในระบบเดิม)

- แอปบางตัวอาจยังคงอยู่ในระบบภายใน เช่น แอปที่มีข้อกำหนดด้านความปลอดภัยสูง
- อาจวางแผนย้ายในอนาคต แต่ยังไม่รวมในแผนการย้ายปัจจุบัน

8. Hybrid (ผสมผสาน)

- ส่วนใหญ่องค์กรจะใช้หลายกลยุทธ์ผสมกัน
- เช่น บางแอปใช้ lift-and-shift, บางแอป refactor, บางแอป repurchase และบางแอป retire
- ต้องวางแผนอย่างรอบคอบเพื่อให้เหมาะสมกับลักษณะของแต่ละระบบ

หน่วยที่ 3 ข้อควรพิจารณาในการย้ายระบบขึ้นคลาวด์ (Cloud Migration Considerations)

การย้ายบริการ แอปพลิเคชัน หรือแพลตฟอร์มไปยังคลาวด์จำเป็นต้องมีการวางแผนอย่างรอบคอบ โดยต้องคำนึงถึงหลายปัจจัยเพื่อให้การเปลี่ยนผ่านเป็นไปอย่างราบรื่นและมีประสิทธิภาพ ปัจจัยสำคัญที่ควรพิจารณา ก่อนดำเนินการ

3.1 ข้อควรพิจารณาเกี่ยวกับความเข้ากันได้ของแพลตฟอร์ม (Platform Compatibility Considerations)

ผู้ให้บริการคลาวด์ส่วนใหญ่มีตัวเลือกหลากหลายในการไฮส忒์เครื่องเสมือน เว็บไซต์ แอปพลิเคชัน ฐานข้อมูล และทรัพยากรอื่น ๆ อย่างไรก็ตาม ควรตรวจสอบให้แน่ชัดว่าแพลตฟอร์มที่เลือกตรงตามข้อกำหนดที่ต้องการหรือไม่

รายการที่ควรตรวจสอบ:

- ทรัพยากรคอมพิวเตอร์ของเครื่องเสมือน (VM)
- ความจุและความเร็วในการเข้าถึงของพื้นที่จัดเก็บข้อมูล
- บริการเครือข่ายเสมือน

- ความสามารถในการเข้าถึงตามภูมิภาค เพื่อให้บริการใกล้กับผู้ใช้งาน
- การควบคุมด้านความปลอดภัย

ผู้ให้บริการคลาวด์จะมีเอกสารรายละเอียดเกี่ยวกับแพลตฟอร์มเพื่อช่วยให้ลูกค้าเลือกบริการที่เหมาะสม สามารถสอบถามข้อมูลเพิ่มเติมจากฝ่ายขายหรือทีมสนับสนุนของผู้ให้บริการได้

3.2 ข้อควรพิจารณาเรื่องต้นทุน (Cost Considerations)

ต้นทุนเป็นปัจจัยสำคัญในการเลือกใช้บริการคลาวด์ ควรวางแผนร่วมกับผู้ให้บริการเพื่อทำความเข้าใจโครงสร้างค่าใช้จ่ายและแนวทางการเพิ่มประสิทธิภาพงบประมาณ แนวทางการเพิ่มประสิทธิภาพต้นทุน:

- เปรียบเทียบการใช้งานแบบ On-demand กับ Spot Instances
- ใช้ตัวเลือกการปรับขนาดอัตโนมัติเมื่อความต้องการพุ่งสูง
- ปรับขนาดทรัพยากรให้เหมาะสมกับงาน (Rightsizing)

3.3 ข้อควรพิจารณาเกี่ยวกับความพร้อมของบริการ (Service Availability Considerations)

เมื่อย้ายระบบไปยังคลาวด์ ควรตรวจสอบว่าบริการที่ต้องการมีให้บริการในภูมิภาคที่ต้องการหรือไม่ บางบริการอาจไม่พร้อมใช้งานในบางพื้นที่เนื่องจากข้อจำกัดทางการเมือง เศรษฐกิจ หรือโครงสร้างพื้นฐาน ตัวอย่าง:

- ก่อนย้ายระบบ: แอปพลิเคชันโอลิมปิกในศูนย์ข้อมูลเพียงแห่งเดียวในอเมริกาเหนือ ผู้ใช้งานทั่วโลกเข้าถึงจากที่เดียว
- หลังย้ายระบบ: แอปพลิเคชันโอลิมปิกในศูนย์ข้อมูลหลายแห่งทั่วโลก ผู้ใช้งานสามารถเข้าถึงจากศูนย์ข้อมูลที่ใกล้ที่สุด

โอลิมปิกในศูนย์ข้อมูลใดก็ได้ที่ต้องการจะสามารถใช้งานที่สูงขึ้น มีความยืดหยุ่นในการใช้ระบบสำรองและการทำงานต่อเนื่องในกรณีที่ศูนย์ข้อมูลใดล้ม

3.4 ข้อควรพิจารณาเกี่ยวกับการผูกติดกับผู้ให้บริการ (Vendor Lock-in Considerations)

การผูกติดกับผู้ให้บริการเกิดขึ้นเมื่อค่าใช้จ่ายหรือความยุ่งยากในการเปลี่ยนผู้ให้บริการสูงเกินไป เช่น การย้ายฐานข้อมูลขนาดใหญ่ หรือระบบที่พึ่งพาเครื่องมือเฉพาะของผู้ให้บริการ ตัวอย่างความเสี่ยง:

- ความยากในการย้ายฐานข้อมูลขนาดใหญ่ หรือระบบที่พึ่งพาเครื่องมือเฉพาะของผู้ให้บริการ
- เครื่องมือและ API เฉพาะของผู้ให้บริการ
- รูปแบบการจัดเก็บข้อมูลที่ไม่เป็นมาตรฐาน

แนวทางลดความเสี่ยง:

- เลือกผู้ให้บริการที่รองรับมาตรฐานเปิด
- วางแผนการย้ายข้อมูลล่วงหน้า
- ใช้เครื่องมือจัดการที่ไม่ผูกกับผู้ให้บริการรายได้รายหนึ่ง
- สำรวจข้อมูลอย่างเป็นระบบและเก็บไว้ภายในองค์กร

มาตรฐานอุตสาหกรรมที่ช่วยลด Vendor Lock-in เช่น:

- OASIS
- CAMP (Cloud Application Management for Platforms)
- TOSCA (Topology and Orchestration Specification for Cloud Applications)

3.5 ข้อควรพิจารณาด้านการประมวลผล (Compute Considerations)

ทรัพยากรการประมวลผล (Compute Resources) ประกอบด้วยหน่วยประมวลผลกลาง (CPU), หน่วยประมวลผลกราฟิก (GPU) และหน่วยความจำ (RAM) ซึ่งเป็นส่วนสำคัญที่ทำให้เครื่องเสมือน (VM) สามารถประมวลผลข้อมูลได้ตามความต้องการของแอปพลิเคชัน

ประสิทธิภาพของ VM จะขึ้นอยู่กับลักษณะการใช้งานของโปรแกรม จึงควรศึกษาคุณสมบัติหรือคำแนะนำของผู้พัฒนาโปรแกรมว่าควรกำหนดค่าอย่างไรเมื่อใช้งานในระบบคลาวด์ สิ่งที่ควรคำนึงถึง:

- เลือกรูปแบบการคิดค่าบริการ เช่น แบบเฉพาะเครื่อง (dedicated), แบบชั่วคราว (spot), หรือแบบตามการใช้งานจริง (on-demand)
- ใช้ระบบแท็ก (tagging) เพื่อควบคุมและติดตามการใช้งาน
- ตรวจสอบความสามารถในการปรับขนาดของเครื่อง VM ได้ในภายหลัง

ผู้ให้บริการคลาวด์ เช่น AWS, Azure และ Google Cloud Platform ต่างมีชุดค่ากำหนด (Instance Types) ที่เตรียมไว้สำหรับวัตถุประสงค์การใช้งานที่หลากหลาย รวมถึงการย้ายระบบแบบอัตโนมัติ หรือกีงอัตโนมัติ

3.6 ข้อควรพิจารณาด้านการจัดเก็บข้อมูล (Storage Considerations)

การย้ายข้อมูลจำนวนมากไปยังระบบคลาวด์เป็นกระบวนการที่ท้าทาย อาจใช้เวลานานและมีค่าใช้จ่ายสูง โดยเฉพาะหากข้อมูลต้องสามารถเข้าถึงได้ตลอดเวลา

ปัจจัยที่ควรพิจารณาในการย้ายข้อมูล:

- ต้นทุน: เวลาในการดูแลระบบ, เวลาหยุดทำงาน, ค่าการใช้แบนด์วิเดอร์
- เวลา: ระยะเวลาในการย้าย, การหยุดให้บริการ

- การย้ายแบบออนไลน์ที่ียบกับอฟไลน์: ส่งผลต่อความพร้อมใช้งานของระบบ
- เครื่องมือที่ใช้: ต้องเลือกให้เหมาะสมกับชนิดของข้อมูลและวิธีการย้าย
- ความปลอดภัย: ต้องคงความปลอดภัยและความเป็นส่วนตัวของข้อมูลตลอดกระบวนการ
การรักษาความถูกต้องของข้อมูล (Data Integrity) เป็นสิ่งสำคัญ ควรใช้วิธีการส่งข้อมูลที่สามารถตรวจสอบได้ เช่น ใช้การทำแฮช (hashing) เพื่อยืนยันว่าไฟล์ที่ส่งและรับเหมือนกันทุกประการ

หากข้อมูลมีขนาดใหญ่มาก และอินเทอร์เน็ตไม่สามารถรองรับได้ การใช้เครื่องมือถ่ายโอนแบบอฟไลน์ เช่น Google Transfer Appliance อาจเป็นทางเลือกที่ดี โดยอุปกรณ์จะถูกส่งมาอย่างค์กรของคุณ หลังจากโอนข้อมูลลงอุปกรณ์เรียบร้อยแล้วจึงส่งกลับไปยัง Google เพื่ออัปโหลดเข้าสู่ระบบคลาวด์ ประเภทของข้อมูลในการจัดเก็บ:

- Block Storage:** เร็วและมีประสิทธิภาพ เหมาะสมสำหรับระบบที่ต้องการความเร็วสูง เช่น ฐานข้อมูล
- File Storage:** ใช้งานง่าย ราคาถูก แต่ไม่เหมาะสมสำหรับการปรับขนาด เหมาะสมกับการจัดเก็บไฟล์ทั่วไป เช่น NFS หรือ CIFS
- Object Storage:** ประหยัดและปรับขนาดได้ดี เหมาะสมสำหรับข้อมูลที่ไม่ต้องเขียนบ่อย เช่น รูปภาพหรือข้อมูลสำรอง แต่ไม่เหมาะสมกับฐานข้อมูลที่มีการเขียนข้อมูลบ่อย ๆ

เครื่องมือช่วยในการย้ายข้อมูล:

- Google Cloud Platform: *Storage Transfer Service*
- Amazon Web Services (AWS): *Storage Gateway, Direct Connect*

3.7 ข้อควรพิจารณาด้านระบบเครือข่าย (Networking Considerations)

การวางแผนระบบเครือข่ายสำหรับการย้ายไปสู่ระบบคลาวด์ควรเริ่มต้นจากการประเมินความพร้อมในการเข้าถึงอินเทอร์เน็ตของพนักงาน เนื่องจากการใช้งานระบบต่าง ๆ จะเปลี่ยนจากการเข้าถึงผ่านเซิร์ฟเวอร์ภายในองค์กรมาเป็นการเชื่อมต่อกับบริการที่อยู่บนคลาวด์ ซึ่งจะเพิ่มภาระทางไฟเบอร์ออฟฟิศเครือข่ายอินเทอร์เน็ตขององค์กรอย่างมาก

ประเด็นสำคัญที่ต้องคำนึงถึง:

- แบนด์วิดท์ (Bandwidth):** ควรจัดให้มีแบนด์วิดท์เพียงพอ มีความหน่วงต่ำ เพื่อรองรับการเชื่อมต่อที่มีประสิทธิภาพ
- ความพร้อมใช้งาน (Availability):** ควรมีการเชื่อมต่ออินเทอร์เน็ตที่มีความเสถียรตลอด 24 ชั่วโมง และพิจารณาการมีผู้ให้บริการอินเทอร์เน็ต (ISP) สำรองเพื่อลดความเสี่ยงจากจุดล้มเหลวเพียงจุดเดียว
- ความปลอดภัย (Security):** เนื่องจากข้อมูลที่จะถูกส่งผ่านเครือข่าย ควรใช้การเข้ารหัสเครือข่ายที่แข็งแกร่ง เช่น VPN หรือ TLS เพื่อป้องกันการดักฟังข้อมูล

การออกแบบเครือข่ายเสมือน (Virtual Network) บนคลาวด์

เมื่อย้ายบริการไปยังคลาวด์ ควรออกแบบโครงสร้างเครือข่ายเสมือน (Virtual Network) เพื่อให้สามารถเข้ามายังระหว่างระบบที่เกี่ยวข้องได้อย่างมีประสิทธิภาพและปลอดภัย

- **VPC (Virtual Private Cloud):** สร้างเครือข่ายเสมือนแบบจำกัดสิทธิ์ เช่น แยกเครือข่ายของนักพัฒนาออกจากเครือข่ายของระบบใช้งานจริง
- **การควบคุมการเข้าถึง (Access Control):** ออกแบบระบบการกำหนดสิทธิ์เพื่อบริหารจัดการการเข้าถึงเครือข่ายคลาวด์ให้เหมาะสม โดยแยกหน้าที่ระหว่างผู้ดูแลเครือข่ายและผู้ดูแลระบบหรือแอปพลิเคชัน
- **การกำหนดหมายเลข IP (IP Addressing):** วางแผนระบบ IP Address สำหรับอุปกรณ์และบริการในคลาวด์ เพื่อให้สามารถควบคุมและบริหารจัดการการเข้ามายังต่อได้อย่างมีประสิทธิภาพ

การเข้าถึงระบบจากภายนอก

เพื่อให้ลูกค้าหรือผู้ใช้งานภายนอกสามารถเข้าถึงเว็บไซต์หรือบริการต่าง ๆ ที่อยู่บนคลาวด์ได้ ควรพิจารณาในประเด็นดังต่อไปนี้:

- **การเข้าถึงจากสาธารณะ (Public-facing Access):** ตรวจสอบให้แน่ใจว่าทรัพยากรในคลาวด์ที่ต้องให้ลูกค้าเข้าถึงสามารถเข้าถึงได้จริง
- **การควบคุมความปลอดภัย (Security Controls):** กำหนดสิทธิ์อย่างรอบคอบว่าลูกค้าสามารถเข้าถึงข้อมูลหรือบริการใดได้บ้าง โดยจำกัดการเข้าถึงเฉพาะข้อมูลที่อนุญาตเท่านั้น

3.8 ข้อควรพิจารณาด้านการจัดการ (Management Considerations)

การโยกย้ายระบบไปยังคลาวด์ไม่ใช่เพียงเรื่องทางเทคนิคหรือการดูแลระบบเท่านั้น แต่ยังมีปัจจัยด้านการบริหารจัดการอื่น ๆ ที่ต้องคำนึงถึง ซึ่งส่งผลต่อความสำเร็จของการย้ายระบบ

3.8.1 การบริหารจัดการภาระงาน (Overhead Considerations)

หลังจากย้ายระบบไปยังคลาวด์ รูปแบบการดูแลระบบและทักษะของผู้ดูแลระบบจะเปลี่ยนไป ผู้ดูแลต้องเรียนรู้:

- คำศัพท์เฉพาะและแนวคิดใหม่ ๆ ของระบบคลาวด์
- การใช้งานคอนโซลหรือแดชบอร์ดที่แตกต่างจากเดิม
- การพิจารณาอบรมเพิ่มเติมจากผู้ให้บริการคลาวด์หรือบริษัทที่อบรมภายนอก
- การใช้ผู้ให้บริการ Managed Service ซึ่งควรระวังว่าที่ที่มีอยู่ที่ยังปรับตัว

3.8.2 สิ่งแวดล้อม พลังงาน และการระบายความร้อน (Environmental, Power, and Cooling Considerations)

การย้ายระบบจากศูนย์ข้อมูลภายในองค์กรไปยังคลาวด์ส่งผลให้:

- ความต้องการใช้พลังงานและการระบายความร้อนลดลง
- ค่าใช้จ่ายด้านพลังงานลดลง
- ลดผลกระทบต่อสิ่งแวดล้อมและสภาพภูมิอากาศ

ข้อดีจากการใช้บริการคลาวด์ที่ส่งผลต่อสิ่งแวดล้อม:

- ผู้ให้บริการคลาวด์ใช้เซิร์ฟเวอร์ที่มีประสิทธิภาพสูงและใช้งานเต็มที่
- ผู้ให้บริการมักใช้พลังงานหมุนเวียนในศูนย์ข้อมูล
- รองรับการทำงานระยะไกล ลดการเดินทาง
- โครงสร้างพื้นฐานรวมศูนย์ช่วยลดภาระทางสิ่งแวดล้อม

ข้อควรพิจารณาเพิ่มเติม:

- ปริมาณการใช้งานอินเทอร์เน็ตจะเพิ่มขึ้น

3.8.3 ข้อกำหนดด้านกฎหมายและการปฏิบัติตาม (Regulatory and Compliance Considerations)

การย้ายข้อมูลไปยังคลาวด์อาจทำให้สูญเสียการควบคุมว่า “ข้อมูลเก็บไว้ที่ใด” ซึ่งอาจขัดต่อข้อกำหนดทางกฎหมายของแต่ละประเทศหรืออุตสาหกรรม:

- ข้อกำหนดเฉพาะท้องถิ่นอาจแตกต่างกันในแต่ละพื้นที่
- Data Sovereignty:** ข้อมูลต้องอยู่ภายใต้กฎหมายของประเทศที่จัดเก็บหรือรวบรวมข้อมูล
- Data Residency:** ข้อมูลต้องถูกจัดเก็บในสถานที่ที่กำหนดไว้ตามข้อกำหนด

ถึงแม้ผู้ให้บริการคลาวด์จะมีเครื่องมือและแนวทางช่วยในด้าน compliance แต่ความรับผิดชอบยังคงอยู่ที่องค์กรผู้ใช้ในการวิจัย ตรวจสอบ และควบคุมการจัดเก็บและจัดการข้อมูลให้ถูกต้องตามกฎหมาย

สรุปการเตรียมใช้งานและการย้ายทรัพยากรขึ้นคลาวด์ (Summary: Provisioning and Migrating Cloud Resources)

การวางแผนใช้งานและการย้ายทรัพยากรขึ้นคลาวด์ต้องคำนึงถึงปัจจัยด้านต้นทุน ประสิทธิภาพ ความปลอดภัย และการปฏิบัติตามข้อกำหนดต่าง ๆ (compliance)

- องค์กรสามารถเลือกใช้ทรัพยากรที่หลากหลายบนคลาวด์ เช่น เครื่องเสมือน (VMs) หรือ แอปพลิเคชัน ซึ่งอาจต้องพัฒนาใหม่หรือย้ายจากระบบเดิมขึ้นคลาวด์
- ไม่ใช่ทุกแอปพลิเคชันในระบบเดิมจะเหมาะสมกับการนำขึ้นคลาวด์ ในระหว่างการย้ายระบบ ต้องมีการวางแผนโครงสร้างพื้นฐานและควบคุมการเข้าถึงอย่างรอบคอบ ก่อนเริ่มการย้ายข้อมูลหรือบริการ เพื่อหลีกเลี่ยงปัญหาการใช้งานหลังจากย้ายเสร็จ

ปัญหา Vendor Lock-in

- อาจเกิดขึ้นเมื่อไม่สามารถดึงข้อมูลด้านการกำหนดค่าความปลอดภัยหรือโครงสร้างพื้นฐานของมาได้ง่าย เช่น กฎไฟร์วอลล์ที่ไม่สามารถนำไปใช้ในระบบใหม่ได้
- ควรเลือกผู้ให้บริการคลาวด์ที่สนับสนุน มาตรฐานเปิด (open standards) และมี แผนออกจากระบบ (exit strategy) อย่างชัดเจน

การควบคุมการเข้าถึง (Access Control)

- การย้ายระบบขึ้นคลาวด์เป็นโอกาสที่ดีในการทบทวนนโยบายการเข้าถึง และปรับให้สอดคล้องกับหลัก Least Privilege (ให้น้อยที่สุดเท่าที่จำเป็น)

เครื่องมือช่วยย้ายระบบ

- ผู้ให้บริการคลาวด์มักมีเครื่องมือและทรัพยากรช่วยในการย้ายระบบและติดตั้ง เพื่อให้ผู้ใช้งานสามารถใช้งานคลาวด์ได้ง่ายขึ้น ท่ามกลางการแข่งขันสูงในตลาด

บทที่ 4

การเปรียบเทียบระบบจัดเก็บข้อมูลบนคลาวด์ (Comparing Cloud Storage)

บทนำของบทเรียน (Module Introduction)

การจัดเก็บข้อมูลเป็นองค์ประกอบที่สำคัญอย่างยิ่งในการใช้งานระบบคลาวด์ โดยเฉพาะในด้านประสิทธิภาพ และต้นทุน ผู้ให้บริการคลาวด์มีเทคโนโลยีการจัดเก็บข้อมูลที่หลากหลายให้เลือก ดังนั้นการทำความเข้าใจ และเลือกใช้ให้เหมาะสมจึงเป็นสิ่งจำเป็น

วัตถุประสงค์ของบทเรียน (Module Objectives)

ในบทเรียนนี้ ผู้เรียนจะได้:

- ศึกษาประเภทของการจัดเก็บข้อมูลและระดับของการให้บริการ (Storage Types and Tiers)
- วิเคราะห์ผลกระทบของประสิทธิภาพและต้นทุนของการจัดเก็บข้อมูล (Performance and Cost Implications)

หน่วยที่ 1 ทรัพยากรและเทคโนโลยีในการจัดเก็บข้อมูล (Storage Resources and Technologies)

การจัดเก็บข้อมูลบนคลาวด์ได้เปลี่ยนแปลงแนวทางการจัดการข้อมูลขององค์กรไปอย่างมาก โดยมอบ ความยืดหยุ่น ความสามารถในการปรับขนาด (scalability) ประสิทธิภาพ และการควบคุมต้นทุนที่ดีกว่าเดิม บทเรียนนี้จะกล่าวถึง:

- ตัวเลือกการจัดเก็บข้อมูล (Storage Options)
- ประเภทของดิสก์ที่ใช้ (Disk Types)
- การจัดเก็บข้อมูลแบบเป็นชั้น (Tiered Storage)
- ผลกระทบต่อประสิทธิภาพและต้นทุนที่เกี่ยวข้อง (Performance and Cost Implications)

1.1 ประเภทของดิสก์ (Disk Types)

ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers – CSPs) มีการนำเสนอระดับการจัดการพื้นที่ จัดเก็บข้อมูลที่หลากหลาย ซึ่งระดับต่าง ๆ เหล่านี้หรือที่เรียกว่า “ชั้นการจัดเก็บ” (Storage Tiers) เป็นการสะท้อนความสัมพันธ์ระหว่างต้นทุน ความจุ และความเร็วในการเข้าถึงข้อมูล โดยเทคโนโลยีที่ใช้มีทั้ง แบบ Solid-State Drive (SSD) และ Hard Disk Drive (HDD)

ตัวอย่างหนึ่งของการจัดการพื้นที่จัดเก็บคือบริการ Elastic Block Storage (EBS) จาก Amazon Web Services (AWS) ซึ่งแบ่งออกเป็นดิสก์ประเภท SSD และ HDD พร้อมทั้งรองรับการจำลองข้อมูลซ้ำ (Replication) ภายใน Availability Zone เพื่อเพิ่มความมั่นคงและความพร้อมใช้งานในเชิงภูมิศาสตร์

ในการออกแบบระบบคลาวด์ใหม่ มักจะมีการประเมินปริมาณข้อมูลต่ำกว่าความเป็นจริง โดยเฉพาะอย่างยิ่งเมื่อคำนึงถึงข้อมูลจำนวนมากที่สะสมอยู่ในระบบเดิมในองค์กร ดังนั้นควรมีการตรวจสอบข้อมูลอย่างละเอียด และจัดหมวดหมู่ข้อมูลตามความถี่ในการเข้าถึงจากผู้ใช้และแอปพลิเคชัน

1.1.1 Solid State Drives (SSD)

SSDs ใช้หน่วยความจำแฟลชในการเข้าถึงข้อมูลได้อย่างรวดเร็วมาก ปัจจุบันกลายเป็นมาตรฐานในเครื่องของผู้ใช้งานทั่วไป และกำลังได้รับความนิยมเพิ่มขึ้นในเซิร์ฟเวอร์ แม้ว่าจะมีราคาสูงกว่าฮาร์ดดิสก์แบบงานหมุน แต่ความเร็วที่ได้ก็มีคุ้มค่าต่อการลงทุน

ตัวอย่าง SSD จาก AWS EBS:

- **io1 และ io2:** ประสิทธิภาพสูงสุด เหมาะสำหรับงานที่ต้องการประมวลผลธุกรรม เช่น MSSQL และ IBM DB2
- **gp2 และ gp3:** สมดุลระหว่างราคาและประสิทธิภาพ ใช้ได้กับ virtual desktops, สภาพแวดล้อม test/dev และงานเกม

1.1.2 Hard Disk Drives (HDD)

HDD เป็นมาตรฐานการเก็บข้อมูลในระบบคอมพิวเตอร์และเซิร์ฟเวอร์มานาน ด้วยความน่าเชื่อถือและราคาที่ประหยัดกว่า เหมาะสำหรับองค์กรที่ต้องจัดเก็บข้อมูลขนาดใหญ่ระดับหลายเทราไบต์ขึ้นไป

ตัวอย่าง HDD จาก AWS EBS:

- **st1:** เหมาะสำหรับข้อมูลที่มีการเข้าถึงบ่อย และต้องการการรับส่งข้อมูลต่อเนื่องสูง (throughput)
- **sc1:** สำหรับการจัดเก็บข้อมูลระยะยาวที่ไม่ค่อยถูกเรียกใช้งาน เช่น ข้อมูลสำรองหรือเอกสารเก่า

1.1.3 Hybrid Disks

ดิสก์แบบไฮบริดเป็นการผสมผสานระหว่าง SSD และ HDD เพื่อให้ได้ทั้งประสิทธิภาพและต้นทุนที่เหมาะสม โดยข้อมูลที่เข้าถึงบ่อยจะถูกเก็บไว้ในหน่วยความจำแฟลช ส่วนข้อมูลที่ไม่ค่อยใช้จะถูกเก็บไว้ในดิสก์แบบงานหมุน

1.1.4 การจัดเก็บข้อมูลระยะยาว (Long-Term Storage)

ข้อมูลที่ถูกจัดเก็บระยะยาว เช่น ในบริการ Azure Blob Storage – Archive Tier จะใช้เวลาหลายชั่วโมงในการค้นหา หมายความว่าข้อมูลที่ถูกเก็บไว้ตามกำหนดการปฏิบัติตามกฎหมาย (compliance) หรือสำรองในกรณีฉุกเฉินที่คาดว่าจะไม่จำเป็นต้องเรียกใช้บ่อย

1.2 ประเภทของระบบจัดเก็บข้อมูล (Storage Types)

ระบบจัดเก็บข้อมูลในระบบคลาวด์แบ่งออกเป็น 3 ประเภทหลัก ได้แก่ Block Storage, File Storage และ Object Storage โดยผู้ให้บริการคลาวด์จะออกแบบการจัดเก็บให้เหมาะสมกับลักษณะการใช้งานแต่ละประเภท ผู้ใช้จำเป็นต้องเลือกประเภทการจัดเก็บที่เหมาะสมกับลักษณะข้อมูลที่ต้องจัดการ

1.2.1 Block Storage (การจัดเก็บแบบบล็อก)

Block Storage เป็นการแบ่งข้อมูลออกเป็นหน่วยเล็ก ๆ ที่เรียกว่า “บล็อก” โดยไม่ขึ้นกับระบบไฟล์ของเซิร์ฟเวอร์ และบล็อกข้อมูลแต่ละบล็อกสามารถกระจายไปเก็บไว้บนอุปกรณ์หลายตัวได้ การจัดเก็บแบบบล็อกจะเน้นการจัดระเบียบข้อมูลให้เกิดประสิทธิภาพสูงสุดต่อข้อมูลเอง ต่างจาก File Storage ที่เน้นการจัดระเบียบตามโครงสร้างระบบไฟล์

Block Storage มีความเร็วสูงและเหมาะสมสำหรับข้อมูลขนาดใหญ่ที่มีการแก้ไขบ่อย เช่น ฐานข้อมูลอย่างไรก็ตาม การจัดเก็บแบบนี้มีต้นทุนที่สูงและการจัดการที่ซับซ้อน โดยมักใช้ร่วมกับระบบ Storage Area Network (SAN) ที่ต้องใช้ต้นทุนและทักษะเชิงเทคนิคสูง ตัวอย่างของ Block Storage ได้แก่:

- AWS Elastic Block Storage (EBS)
- Azure Blob Storage
- GCP Persistent Disk

แม้ว่าชื่อ Azure Blob จะดูเหมือน Object Storage แต่ในการใช้งานเชิงเทคนิคกับ VM สามารถกำหนดพฤติกรรมให้ทำหน้าที่เหมือน Block Storage ได้ ระบบอย่าง AWS EBS จะจัดองข้อมูลช้าภายใน Availability Zone เพื่อเพิ่มความทนทานและพร้อมใช้งาน โดยมีข้อจำกัดว่า EC2 instance ต้องอยู่ในโซนเดียวกับ EBS

1.2.2 File Storage (การจัดเก็บแบบไฟล์)

File Storage เป็นรูปแบบการจัดเก็บที่ผู้ใช้งานทั่วไปคุ้นเคย เช่น การบันทึกเอกสาร รูปภาพ หรือสเปรดชีต โดยไฟล์ถูกจัดเก็บแยกกันเป็นรายชื่อในระบบไฟล์ เช่น NTFS หรือ ext4 ระบบ NAS (Network Attached Storage) ก็ใช้การจัดเก็บแบบนี้เช่นกัน

ข้อมูลเมตาของไฟล์ เช่น ตำแหน่ง เวลาเข้าถึง และสิทธิ์ จะถูกจัดเก็บไว้ในระบบไฟล์ ทำให้สามารถเข้าถึงและจัดการได้อย่างมีประสิทธิภาพ File Storage เป็นตัวเลือกที่มีต้นทุนต่ำ เหมาะสำหรับข้อมูลขนาด

เลือกถึงปานกลาง และใช้ร่วมกับระบบแชร์ไฟล์ เช่น NFS และ CIFS ตัวอย่างของ File Storage บนคลาวด์ได้แก่:

- AWS Elastic File System (EFS)
- Azure File Storage
- GCP Filestore

1.2.3 Object Storage (การจัดเก็บแบบออบเจกต์)

Object Storage เป็นการจัดเก็บข้อมูลในรูปแบบของวัตถุ (Object) ซึ่งแต่ละวัตถุจะประกอบด้วยข้อมูลหลักและเมตาดาทาโดยละเอียด ใช้สำหรับจัดเก็บข้อมูลขนาดใหญ่ เช่น วิดีโอ รูปภาพ หรือไฟล์ที่ไม่ต้องเปลี่ยนแปลงบ่อย

จุดเด่นของ Object Storage คือการสเกลที่ง่าย ต้นทุนต่ำ และเหมาะสมสำหรับข้อมูลที่เขียนครั้งเดียวแล้วอ่านซ้ำบ่อย ๆ อย่างไรก็ตาม ประสิทธิภาพในการเขียนจะไม่ดีนัก จึงไม่เหมาะสมกับข้อมูลแบบฐานข้อมูล ตัวอย่างของ Object Storage ได้แก่:

- AWS S3
- Azure Blob Storage
- GCP Cloud Storage

1.2.4 Buckets (ถังจัดเก็บข้อมูล) ใน Object Storage

Bucket คือหน่วยจัดเก็บหลักสำหรับ Object Storage ผู้ใช้สามารถสร้าง Bucket ได้ตามภูมิภาคที่ต้องการ และเก็บวัตถุต่าง ๆ ไว้ในนั้น Bucket ต้องมีชื่อที่ไม่ซ้ำกันในระดับโลก และสามารถกำหนดคุณสมบัติการจัดเก็บ เช่น AWS S3 Standard สำหรับข้อมูลที่ใช้งานบ่อย หรือ AWS S3 Glacier สำหรับการเก็บถาวร

ตัวอย่างการสร้าง Bucket ใน AWS มีขั้นตอนดังนี้:

1. เข้าสู่เมนู Storage
2. เลือก S3
3. เลือก Buckets
4. คลิก Create bucket

1.2.5 ความมั่นคงปลอดภัยในระบบคลาวด์แบบ Multi-Tenant

ในโครงสร้างพื้นฐานระบบคลาวด์แบบสาธารณะที่รองรับผู้ใช้หลายราย (Multi-tenant) มีข้อกังวลเกี่ยวกับความเป็นส่วนตัวและความปลอดภัยของข้อมูล ดังนั้น ระบบจึงมีการใช้ Tenant ID เพื่อระบุผู้ใช้อย่างเฉพาะเจาะจงและใช้ควบคุมการเข้าถึงทรัพยากร

Tenant ID จะผูกกับ namespace ซึ่งใช้สร้าง Bucket และบริหารจัดการข้อมูลในระดับองค์กรต่อไป

1.3 การจัดเก็บข้อมูลแบบแบ่งระดับ (Tiered Storage)

การจัดเก็บข้อมูลแบบแบ่งระดับ (Tiered Storage) เป็นแนวทางที่ผู้ดูแลระบบใช้เพื่อปรับสมดุลระหว่างต้นทุนและประสิทธิภาพ โดยพิจารณาจากความถี่ในการเข้าถึงข้อมูลและระยะเวลาในการจัดเก็บข้อมูล ซึ่งการแบ่งระดับนี้มักเรียกตามลำดับความถี่ในการเข้าถึงข้อมูล ได้แก่ Hot, Warm, Cold และ Archive

ระดับ Hot หมายถึงข้อมูลที่มีการเข้าถึงบ่อย ส่วนระดับ Cold และ Archive เป็นข้อมูลที่เข้าถึงน้อย หรือแทบไม่ถูกเรียกใช้เลย โดย Archive จะเป็นข้อมูลที่มีการจัดเก็บในระยะยาวและไม่คาดว่าจะถูกเรียกใช้งานบ่อย จึงเหมาะสมสำหรับข้อมูลสำรองหรือข้อมูลตามข้อกำหนดด้านกฎหมายเป็นตัวอย่างอัตราค่าบริการของ Azure Blob Storage ในแบบจ่ายตามการใช้งาน (Pay-as-you-go):

- ระดับ Premium: \$0.15/GB
- ระดับ Hot: \$0.018/GB
- ระดับ Cool: \$0.01/GB
- ระดับ Cold: \$0.00036/GB
- ระดับ Archive: \$0.00099/GB

ข้อมูลในระดับ Cool จะถูกจัดประเภทเมื่อมีอายุอย่างน้อย 30 วัน ส่วนระดับ Archive ต้องมีอายุอย่างน้อย 180 วัน และจัดเก็บแบบอฟไลน์ ทั้งนี้ Microsoft ใช้คำศัพท์ที่สอดคล้องกับมาตรฐานในอุตสาหกรรม เช่น Hot, Cool และ Archive

1.3.1 การเพิ่มพื้นที่จัดเก็บข้อมูลให้กับเครื่อง VM

กระบวนการเพิ่มพื้นที่จัดเก็บข้อมูลให้กับเครื่อง VM (Virtual Machine) จะแตกต่างกันเล็กน้อยขึ้นอยู่กับผู้ให้บริการคลาวด์ แต่มีลักษณะใกล้เคียงกัน โดยทั่วไปมีขั้นตอนดังนี้:

- เลือกเครื่อง VM ที่ต้องการเพิ่มพื้นที่จัดเก็บ
- เลือกว่าจะสร้างดิสก์ใหม่หรือเชื่อมต่อดิสก์ที่มีอยู่
- หากสร้างดิสก์ใหม่ ให้เลือกประเภทดิสก์ เช่น SSD หรือ HDD พร้อมกำหนดความจุ และตรวจสอบว่าค่าการทำงานตรงกับความต้องการ

เมื่อดำเนินการเสร็จสิ้น หน้าจอบริหารจัดการจะปรากฏดิสก์ที่เพิ่มใหม่ในคุณสมบัติของ VM

1.4 การย้ายข้อมูลขึ้นคลาวด์ (Migrate Data to the Cloud)

ผู้ให้บริการคลาวด์แต่ละรายมีเครื่องมือหลากหลายสำหรับการย้ายข้อมูลขนาดใหญ่จากระบบภายในองค์กรไปยังคลาวด์ โดยแบ่งออกเป็นสองแนวทางหลัก ได้แก่ การถ่ายโอนผ่านเครือข่ายอินเทอร์เน็ต (รวมถึงการเชื่อมต่อโดยตรงที่มีความปลอดภัย) และการถ่ายโอนแบบอฟไลน์ผ่านอุปกรณ์จัดเก็บข้อมูลทางกายภาพ ตัวอย่างการเชื่อมต่อโดยตรงและเครื่องมือจากผู้ให้บริการคลาวด์:

	GCP	Azure	AWS
Direct Connection Options	<ul style="list-style-type: none"> • Direct Peering (public Internet) • Cloud Interconnect (avoids public Internet) • Storage Transfer Service (on-premises/cloud-to-cloud) 	<ul style="list-style-type: none"> • ExpressRoute (avoids public Internet) • Migrate (all-in-one migration service) 	<ul style="list-style-type: none"> • Cloud Data Migration • Direct Connect (avoids public Internet)
Physical device Options	<ul style="list-style-type: none"> • Transfer Appliance 	<ul style="list-style-type: none"> • DataBox 	<ul style="list-style-type: none"> • Snow Family

ภาพที่ 20 Migrate Data to the Cloud

- **Google Cloud Platform (GCP):** Direct Peering (เชื่อมต่อผ่านอินเทอร์เน็ต), Cloud Interconnect (หลีกเลี่ยงอินเทอร์เน็ตเพื่อความปลอดภัย), Storage Transfer Service (ถ่ายโอนหรือสำรองข้อมูลระหว่างระบบ)
- **Microsoft Azure:** ExpressRoute (เชื่อมต่อโดยไม่ผ่านอินเทอร์เน็ต), Azure Migrate (เครื่องมือย้ายข้อมูลแบบครบวงจรสำหรับเซิร์ฟเวอร์ ฐานข้อมูล และเว็บแอป)
- **Amazon Web Services (AWS):** Cloud Data Migration, AWS Direct Connect (หลีกเลี่ยงการใช้สาธารณูปโภคอินเทอร์เน็ต)

สำหรับพื้นที่ที่มีการเข้ามต่ออินเทอร์เน็ตจำกัด อาจใช้การย้ายข้อมูลแบบอพลайнผ่านอุปกรณ์จริง เช่น:

- GCP: Transfer Appliance
- Azure: DataBox
- AWS: Snow Family

1.5 ผลกระทบด้านประสิทธิภาพ (Performance Implications)

ประสิทธิภาพของระบบจัดเก็บข้อมูลในคลาวด์ขึ้นอยู่กับปัจจัยหลายประการ ได้แก่ จำนวนการดำเนินการอ่าน/เขียนต่อวินาที (IOPS), ความเร็วในการส่งผ่านข้อมูล (Throughput), เวลาແ Pang (Latency) และความลึกของคิวคำสั่ง (Queue Depth) ซึ่งทั้งหมดนี้ควรได้รับการพิจารณาแบบองค์รวมในฐานะระบบเดียวกัน เพื่อประเมินความสามารถของสื่อจัดเก็บข้อมูล

IOPS (Input/Output Operations Per Second) คือการวัดจำนวนการดำเนินการอ่านหรือเขียนข้อมูลที่สามารถทำได้ต่อวินาที โดยวัดตั้งแต่ต้นจนจบของแต่ละคำสั่ง IOPS จะขึ้นอยู่กับเทคโนโลยีของดิสก์ เช่น SSD หรือ HDD โดยทั่วไป SSD จะให้ค่า IOPS ที่สูงกว่า ซึ่งหมายความว่าต้องทำการพิจารณาแบบองค์รวมในฐานะระบบเดียวกัน เพื่อประเมินความสามารถของสื่อจัดเก็บข้อมูลที่ส่งผลต่อประสิทธิภาพ เพราะต้องพิจารณาร่วมกับ Throughput ด้วย

Throughput หมายถึงความเร็วในการถ่ายโอนข้อมูล มีหน่วยวัดเป็นเมกะไบต์ต่อวินาที (MB/sec) ซึ่งใช้วัดจำนวนข้อมูลที่อ่านหรือเขียนได้ในแต่ละวินาที Throughput มีความสำคัญอย่างมากสำหรับการจัดการไฟล์ขนาดใหญ่ เช่น การประมวลผลบันทึกหรือการเก็บข้อมูลขนาดใหญ่

ตัวอย่างเช่น Amazon Web Services (AWS) มีดิสก์ประเภท Throughput-Optimized HDD (เช่น st1) และ Cold HDD (เช่น sc1) สำหรับงานที่ไม่ต้องการเข้าถึงข้อมูลบ่อย โดย st1 เหมาะกับงาน Big Data หรือคลังข้อมูล ส่วน sc1 เหมาะกับข้อมูลที่เน้นความคุ้มค่ามากกว่าประสิทธิภาพ

- st1: ขนาด 125 GiB – 16 TiB, IOPS สูงสุด 500, Throughput สูงสุด 500 MB/s
- sc1: ขนาด 125 GiB – 16 TiB, IOPS สูงสุด 250, Throughput สูงสุด 250 MB/s

การเลือกใช้ SSD หรือ HDD ควรพิจารณาจากลักษณะงาน เพื่อให้สมดุลระหว่างต้นทุน กับประสิทธิภาพ เช่น หากงานมีคำสั่งอ่าน/เขียนจำนวนมากในแต่ละวินาที ให้เลือก SSD ที่ปรับแต่ง IOPS ได้ในทางกลับกัน หากเป็นงานใหญ่ที่ต้องการความเร็วในการอ่าน/เขียนไฟล์ใหญ่ ให้เลือก HDD แบบ throughput-optimized

ผู้ให้บริการอย่าง Microsoft Azure ก็มีการจัดระดับประสิทธิภาพของดิสก์ โดยใช้เกณฑ์ของ IOPS และ Throughput ที่คล้ายกับ AWS

Latency หมายถึงเวลาที่ข้อมูลใช้เดินทางจากต้นทางไปยังปลายทาง เวลาแฝงส่งผลโดยตรง ต่อประสบการณ์ของผู้ใช้งาน เช่น ความล่าช้าในการเข้าถึงไฟล์ Latency จะได้รับผลกระทบจากหลายปัจจัย เช่น ประสิทธิภาพของสื่อจัดเก็บข้อมูล และความหนาแน่นของคิวคำสั่ง ตัวอย่างของปัจจัยที่ส่งผลต่อ Storage Latency ได้แก่:

- ความเร็วในการอ่าน/เขียนของดิสก์
- ความลึกของคิวคำสั่ง
- ตำแหน่งทางภูมิศาสตร์ของดิสก์เมื่อเทียบกับผู้ใช้

Queue Depth คือจำนวนคำสั่งอ่านหรือเขียนที่รอดำเนินการในคิว หากค่าดังกล่าวสูงเกินไป แสดงว่าดิสก์กำลังทำงานหนัก และอาจจำเป็นต้องขยายขนาดหรือเพิ่มจำนวนดิสก์เพื่อรับ荷重任ได้ดีขึ้น

1.5.1 การวิเคราะห์ประสิทธิภาพ (Analyzing Performance)

การจัดเก็บข้อมูลในคลาวด์ต้องอาศัยการตรวจสอบและปรับแต่งอย่างสม่ำเสมอเพื่อให้ได้ทั้งประสิทธิภาพ และความคุ้มค่า โดยปัจจัยที่ต้องพิจารณา มีทั้งการออกแบบของแอปพลิเคชัน, ความต้องการของระบบ, งบประมาณ และประเภทของดิสก์ที่เลือกใช้

Google Cloud Platform (GCP) มีเครื่องมือชื่อ Cloud Monitoring ที่ช่วยวิเคราะห์ประสิทธิภาพของการจัดเก็บข้อมูลแบบเชิงลึก ตัวอย่างข้อมูลที่สามารถตรวจสอบได้จาก Cloud Monitoring ได้แก่:

- ค่าเฉลี่ย IOPS สำหรับการอ่าน
- ค่า Latency ของการอ่านและเขียน

- ค่าเฉลี่ยของ Queue Depth
- ค่าการอ่านและเขียนสูงสุด
- ปริมาณข้อมูลที่อ่านและเขียนในหน่วยไบต์

การใช้เครื่องมือวิเคราะห์ประสิทธิภาพเหล่านี้เป็นส่วนสำคัญของบทบาทผู้ดูแลระบบคลาวด์ในการบริหารจัดการทรัพยากรจัดเก็บข้อมูลอย่างมีประสิทธิภาพ

1.6 การให้บริการจัดเก็บข้อมูลแบบบริการ (Storage as a Service: STaaS)

บทเรียนนี้มุ่งเน้นที่การจัดเก็บข้อมูลในระบบคลาวด์ที่รวมอยู่ในระบบที่กว้างกว่า อย่างไรก็ตาม การจัดเก็บข้อมูลไม่ได้เป็นเพียงข้อกำหนดสำหรับบริการคลาวด์อื่นเท่านั้น แต่ยังเป็นโซลูชันในตัวเองด้วย บริการจัดเก็บข้อมูลแบบสแตนเดอร์โอลน (Standalone Storage as a Service: STaaS) จึงกลายเป็นความต้องการที่สำคัญทั้งในระดับองค์กรและผู้ใช้ทั่วไป เนื่องจากให้ประโยชน์ทั้งในด้านประสิทธิภาพและต้นทุน ตัวอย่างของ STaaS ได้แก่:

- Microsoft OneDrive
- Dropbox
- Box
- Google Drive

ประโยชน์ของ STaaS

- ประหยัดต้นทุน หากบริหารจัดการอย่างเหมาะสม
- รองรับการถูกคืนข้อมูลเมื่อเกิดภัยพิบัติ
- รองรับการขยายตัวของข้อมูล (Scalability)
- เข้าถึงได้จากอุปกรณ์หลากหลายในทุกที่
- ง่ายต่อการแชร์และทำงานร่วมกัน

ข้อเสียที่อาจเกิดขึ้นจาก STaaS

- ความมั่นคงปลอดภัยและความเป็นส่วนตัวของข้อมูลที่เก็บไว้
- ความมั่นคงปลอดภัยของข้อมูลระหว่างการส่งผ่าน
- ค่าบริการจัดเก็บข้อมูลอาจเพิ่มขึ้นเมื่อใช้มากขึ้น
- การควบคุมและการเก็บรักษาข้อมูล
- ความพร้อมใช้งานของระบบ (High Availability)
- ปัญหา Downtime ของระบบ
- การขึ้นอยู่กับผู้ให้บริการ (Vendor Lock-in)

1.7 การจัดเก็บข้อมูลแบบกำหนดด้วยซอฟต์แวร์ (Software-defined Storage: SDS)

SDS เป็นองค์ประกอบสำคัญของระบบจัดเก็บข้อมูลระดับองค์กร โดยแยกระบบจัดการออกจากฮาร์ดแวร์ที่ใช้จัดเก็บข้อมูล SDS ทำหน้าที่เป็นตัวกลางระหว่างการร้องขอข้อมูลกับฮาร์ดแวร์จัดเก็บข้อมูล โดยซอฟต์แวร์จะจัดการและรวมฮาร์ดแวร์ที่หลากหลาย เช่น SAN, NAS และระบบจัดเก็บข้อมูลแบบตั้งเดิม ให้สามารถทำงานร่วมกันได้อย่างยืดหยุ่น

ลักษณะเด่นของ SDS ได้แก่:

- การทำงานแบบอัตโนมัติ (Automation)
- การจัดสรรทรัพยากรตามนโยบาย (Policy-based provisioning)
- การจัดการระบบตามนโยบาย (Policy-based management)

คุณสมบัติของการจัดการตามนโยบายใน SDS

- การบีบอัดข้อมูล (Compression)
- การลดความซ้ำซ้อนของข้อมูล (Deduplication)
- การจัดสรรพื้นที่จัดเก็บแบบหนาและบาง (Thick and Thin Provisioning)
- การทำสำเนาข้อมูล (Replication)

SDS อาจเป็นส่วนหนึ่งของบริการจากผู้ให้บริการคลาวด์ (CSP) หรือสามารถติดตั้งใช้งานได้ภายในศูนย์ข้อมูลขององค์กรโดยไม่ต้องพึ่งเทคโนโลยีการจัดการคลาวด์

1.7.1 การบีบอัดข้อมูล (Compression)

การบีบอัดข้อมูลคือการลดขนาดของข้อมูลให้อยู่ในรูปแบบที่ใช้บิตน้อยลง ช่วยลดพื้นที่ในการจัดเก็บโดยเฉพาะสำหรับข้อมูลที่ไม่ถูกเข้าถึงบ่อยนัก อย่างไรก็ตาม ข้อมูลที่ถูกบีบอัดต้องใช้ทรัพยากรของ CPU ในการถอดรหัสเมื่อเรียกใช้งาน นอกจากนี้ ยังช่วยลดปริมาณข้อมูลที่ต้องส่งผ่านเครือข่าย

1.7.2 การลดความซ้ำซ้อนของข้อมูล (Deduplication)

การลดความซ้ำซ้อนคือการลบส่วนข้อมูลที่ซ้ำกันออก โดยเก็บไว้เพียงชุดเดียว และแทนที่ส่วนที่ซ้ำกันด้วยตัวชี้ (pointer) ไปยังข้อมูลต้นฉบับ วิธีนี้ช่วยประหยัดพื้นที่ในการจัดเก็บอย่างมาก โดยเฉพาะกับไฟล์ประเภทที่ซ้ำกันสูง เช่น รูปภาพระบบปฏิบัติการหรือไฟล์ติดตั้ง VM โดย AWS ระบุว่าสามารถลดพื้นที่ได้ถึง 50–60% ในกรณีข้อมูลทั่วไป

1.7.3 การจัดสรรพื้นที่แบบบางและหนา (Thin vs Thick Provisioning)

Thin Provisioning คือการกำหนดพื้นที่สูงสุด เช่น 100 GB แต่ในความเป็นจริงจะถูกใช้เท่าที่จำเป็น เช่น 50 GB โดยจะเรียกเก็บค่าใช้จ่ายเฉพาะพื้นที่ที่ใช้งานจริง

Thick Provisioning คือการจองพื้นที่ทั้งหมดล่วงหน้า เช่น 100 GB แม้ว่าจะใช้งานจริงเพียง 50 GB ก็ตาม ค่าใช้จ่ายจะคิดตามขนาดที่จองไว้ทั้งหมดทันที

Thin provisioning ช่วยประหยัดต้นทุนแต่เมื่อความเสี่ยงต่อประสิทธิภาพหากต้องขยายพื้นที่ในช่วงที่มีการใช้งานสูง

1.7.4 การทำสำเนาข้อมูล (Replication)

Replication คือการสร้างสำเนาข้อมูลจากต้นฉบับไปยังตำแหน่งอื่น เพื่อให้สามารถใช้งานได้อย่างต่อเนื่องเมื่อเกิดความผิดพลาดในตำแหน่งใดตำแหน่งหนึ่ง

ประเภทของการทำ Replication ได้แก่:

- **Storage replication:** ทำซ้ำระหว่างโครงสร้างจัดเก็บข้อมูล
- **VM replication:** ทำซ้ำเครื่องเสมือนระหว่างไซส์
- **CDN replication:** กระจายข้อมูลไปยังศูนย์ข้อมูลใกล้ผู้ใช้เพื่อความรวดเร็ว
- **Distributed file system replication:** ทำสำเนาข้อมูลไปยังเซิร์ฟเวอร์ที่ใกล้ผู้ใช้เป้าหมาย
- **Availability zone replication:** ทำซ้ำข้อมูลในหลายศูนย์ข้อมูลในภูมิภาคเดียวกัน เพื่อให้ใช้งานได้หากศูนย์ข้อมูลใดล้ม

1.8 ผลกระทบด้านต้นทุน (Cost Implications)

บริการจัดเก็บข้อมูลในระบบคลาวด์มีทางเลือกมากmany ที่ช่วยลดต้นทุนได้อย่างมีนัยสำคัญ อย่างไรก็ตาม เพื่อให้สามารถใช้ประโยชน์ด้านต้นทุนได้สูงสุด ผู้ใช้งานจำเป็นต้องเข้าใจปัจจัยต่างๆ ที่ส่งผลต่อต้นทุนของการใช้บริการจัดเก็บข้อมูลในคลาวด์

ปัจจัยที่ควรพิจารณา ได้แก่:

- **ค่าธรรมเนียมการดึงข้อมูล (Egress fees):** การอัปโหลดข้อมูลเข้าสู่ระบบคลาวด์มักไม่เสียค่าใช้จ่าย แต่จะมีค่าธรรมเนียมในการดึงข้อมูลออกมายังผู้ใช้งาน ซึ่งไม่ได้หมายถึงการลบข้อมูลออกจากคลาวด์ แต่รวมถึงการเรียกดูหรือดาวน์โหลดด้วย
- **การถ่ายโอนหรือซิงโครไนซ์ข้อมูลที่ไม่จำเป็น:** การเคลื่อนย้ายข้อมูลระหว่างบริการจัดเก็บข้อมูลบนคลาวด์ต่างๆ อาจมีค่าใช้จ่ายเกิดขึ้น
- **ความต้องการของข้อมูล:** ควรเข้าใจความต้องการของข้อมูลในด้านความมั่นคงปลอดภัย การจัดเก็บ และการเข้าถึง เพื่อเลือกโซลูชันที่เหมาะสม
- **วงจรชีวิตของข้อมูล (Data lifecycle):** การปรับรูปแบบการจัดเก็บข้อมูลตามอายุการใช้งาน เช่น การย้ายข้อมูลที่ไม่ค่อยถูกเรียกใช้งานไปยังระบบจัดเก็บที่ช้ากว่าแต่ราคาถูกกว่า จะช่วยประหยัดต้นทุนได้มาก
- **การจัดเก็บข้อมูลแบบแบ่งชั้น (Tiered storage):** ควรออกแบบการจัดเก็บข้อมูลให้เหมาะสม กับลักษณะการเข้าถึง โดยใช้ชั้นร้อน (Hot Tier) สำหรับข้อมูลที่ถูกเข้าถึงบ่อย และชั้นเย็น (Cold/Archive Tier) สำหรับข้อมูลที่ไม่ค่อยถูกใช้

- ความแตกต่างด้านต้นทุนระหว่างภูมิภาค: ควรศึกษาและเปรียบเทียบราคากลางจัดเก็บข้อมูลในแต่ละภูมิภาค

1.8.1 ค่าธรรมเนียมการดึงข้อมูล (Egress Fees)

ผู้ให้บริการคลาวด์มักอนุญาตให้ผู้ใช้อัปโหลดข้อมูลเข้าสู่ระบบได้โดยไม่คิดค่าบริการ เพื่อจูงใจให้ใช้พื้นที่จัดเก็บในระบบของตน อย่างไรก็ตาม การนำข้อมูลออกจากระบบจะมีค่าใช้จ่าย เช่น:

- พนักงานขององค์กรดาวน์โหลดไฟล์หรือรายงานจากคลาวด์
- ลูกค้าเรียกดูข้อมูลจากระบบคลาวด์มายังอุปกรณ์ของตน
- การย้ายข้อมูลจากประเทการจัดเก็บข้อมูลหนึ่งไปยังอีกประเทหนึ่ง

ประโยชน์ด้านต้นทุนของคลาวด์เมื่อเปรียบเทียบกับระบบภายในองค์กร

- ลดต้นทุนハードแวร์จัดเก็บข้อมูล
- ลดการใช้พลังงาน
- ลดความจำเป็นในการดูแลและบำรุงรักษา
- เพิ่มขีดความสามารถในการขยายตัว (Scalability)
- เพิ่มความยืดหยุ่นในการกู้คืนจากภัยพิบัติ

1.8.2 การจัดการโควตาการใช้งานของผู้ใช้ (User Quotas)

การตั้งโควตาสำหรับผู้ใช้งานเป็นวิธีที่ช่วยควบคุมต้นทุนและพื้นที่จัดเก็บ โดยสามารถตั้งขีดจำกัดในหลายระดับ เช่น:

- ไฟล์ระบบจัดเก็บข้อมูล: เช่น NTFS ของ Microsoft สามารถตั้งโควตาสำหรับผู้ใช้แต่ละคนหรือกลุ่มได้
- อุปกรณ์จัดเก็บข้อมูล: พาร์ทิชันของดิสก์ใน Windows สามารถตั้งโควตาเพื่อจำกัดการใช้พื้นที่
- ระบบคลาวด์: รองรับการตั้งโควตาสำหรับผู้ใช้งานเพื่อจำกัดการใช้พื้นที่รวม

ผู้ให้บริการคลาวด์มักใช้คำว่า “Service Quota” เพื่อควบคุมการใช้ทรัพยากร เช่น ความจุที่แต่ละ Subscription, VPC, VM หรือองค์ประกอบอื่นๆ จะสามารถใช้งานได้ ดังนั้นควรระวังไม่ให้สับสนระหว่าง quota ระดับผู้ใช้กับ quota ระดับบริการ

1.8.3 การใช้ SDS สำหรับโควตา

ระบบจัดเก็บข้อมูลแบบกำหนดด้วยซอฟต์แวร์ (Software-defined Storage: SDS) สามารถนำมาใช้ตั้งโควตาในระดับไฟล์ระบบได้ โดยมีทั้งโควตาแบบอ่อน (Soft) และแบบแข็ง (Hard)

- โควตาแบบอ่อน: ไม่มีการบังคับใช้จริง แต่จะแจ้งเตือนผู้ใช้หรือผู้ดูแลระบบ หรือบันทึกลงใน IoT

- គុវតាបែបផ្លើង៖ ប៉ាក់បំិចទេរិន ដូចមួយសាមរភាពតុកកៅបីអូមូលធិំពិំមេពិំមេ ដីជានកវារៈមិនធម៌ពីនេះទៀត ហើយទីនេះ គុវតាបែបផ្លើង។

ในบางกรณี CSP อาจกำหนดข้อจำกัดด้านการใช้พื้นที่จัดเก็บไว้ตามประเภทของการสมัครใช้งาน (Subscription) โดยผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้หากไม่อัปเกรดประเภทการใช้งาน

1.9 ทรัพยากรัจดเก็บข้อมูลแบบไฮเปอร์คอนเวอร์จ (Hyperconverged Storage Resources)

การจัดเก็บข้อมูลแบบไฮเปอร์คอนเวอร์จ (Hyperconverged Storage) คือการผสมรวมระหว่างทรัพยากรประมวลผล (Virtual CPU หรือ vCPU และหน่วยความจำ) เข้ากับความสามารถในการจัดเก็บข้อมูลกล่าวอีกนัยหนึ่งคือ แนวคิด “Software-defined everything” ได้ถูกนำมาใช้ในระดับการรวมทรัพยากรประมวลผลและการจัดเก็บข้อมูลเข้าไว้ด้วยกันผ่านซอฟต์แวร์

ในระบบคลาวด์ เช่น Amazon Web Services (AWS), Google Cloud Platform (GCP), และ Microsoft Azure การกำหนดค่าของอินสแตนซ์ (Instance) จะรวมถึงตัวเลือกการจัดเก็บข้อมูลไว้เป็นส่วนหนึ่งของการตั้งค่าโดยรวม

ตัวอย่างเช่น:

AWS M4 Instances (กลุ่ม General-Purpose):

- m4.large: vCPU = 2, RAM = 8 GiB, ใช้ EBS เท่านั้น, แบนด์วิดธ์ของ EBS = 450 Mbps
 - m4.xlarge: vCPU = 4, RAM = 16 GiB, ใช้ EBS เท่านั้น, แบนด์วิดธ์ของ EBS = 750 Mbps

Instance	vCPU	Mem (GiB)	Storage	Dedicated EBS Bandwidth (Mbps)
m4.large	2	8	EBS-only	450
m4.xlarge	4	16	EBS-only	750

ภาพที่ 21 AWS M4 Instances

AWS D2 Instances (กลุ่ม Storage-Optimized):

- d2.xlarge: vCPU = 4, RAM = 30.5 GiB, พื้นที่จัดเก็บ = 3x2TB HDD
 - d2.2xlarge: vCPU = 8, RAM = 61 GiB, พื้นที่จัดเก็บ = 6x2TB HDD

Instance	vCPU	Mem (GiB)	Storage
d2.xlarge	4	30.5	3x2TB HDD
d2.2xlarge	8	61	6x2TB HDD

ภาพที่ 22 AWS D2 Instances

อินสแตนซ์กลุ่ม D2 นี้เหมาะสมสำหรับงานที่ต้องการความจุสูงและประหยัดต้นทุนด้านการจัดเก็บข้อมูล โดยมีพื้นที่จัดเก็บแบบฮาร์ดดิสก์ท้องถิ่น (Local HDD)

GCP มีตัวเลือกสำหรับ SSD แบบท้องถิน (Local SSD) เพื่อการจัดเก็บข้อมูลแบบล็อก (Block Storage) ที่มีประสิทธิภาพสูง และคล้ายกับ AWS ตรงที่ GCP มีบริการจัดเก็บข้อมูลหลากหลายรูปแบบที่ถูกปรับแต่งมาให้เหมาะสมกับภาระงานที่คาดว่าจะเกิดขึ้น

สรุปการเปรียบเทียบการจัดเก็บข้อมูลบนคลาวด์ (Summary Comparing Cloud Storage)

Storage as a Service (STaaS) และ Software-defined Storage (SDS) มอบทางเลือกมากมายให้แก่องค์กรในการจัดเก็บข้อมูลบนคลาวด์ ทั้งนี้ขึ้นอยู่กับประเภทของดิสก์และระดับชั้น (tier) ที่เลือกใช้งาน ข้อมูลอาจถูกจัดเก็บใกล้กับผู้ใช้งานมากขึ้น เข้าถึงได้รวดเร็วขึ้น และสามารถคืนได้อย่างมีประสิทธิภาพยิ่งขึ้นกว่าที่เคย

อย่างไรก็ตาม การจัดเก็บข้อมูลบนคลาวด์มีข้อพิจารณาที่แตกต่างจากการจัดเก็บข้อมูลแบบเดิม เช่น การจัดเก็บแบบต่อเข้ากับเครื่องโดยตรง (Direct-attached Storage) หรือเครือข่ายพื้นที่จัดเก็บข้อมูล (Storage Area Network: SAN) ซึ่งรวมถึงค่าธรรมเนียมการถ่ายโอนข้อมูลออก (egress fee) โครงสร้างประสิทธิภาพตามระดับชั้น (tiered performance structure) และตัวเลือกของประเภทดิสก์ที่หลากหลาย ทั้งหมดนี้ล้วนส่งผลต่อโซลูชันการจัดเก็บข้อมูลบนคลาวด์ของคุณ

บทที่ 5

การแก้ไขปัญหาการปรับใช้และต้นทุน (Troubleshooting Deployment Issues and Cost)

บทนำของบทเรียน (Module Introduction)

การจัดสรรทรัพยากร (Resource provisioning) ไม่ว่าจะเป็นการปรับใช้ครั้งแรกหรือเป็นส่วนหนึ่งของการย้ายระบบ อาจไม่เป็นไปอย่างราบรื่นเสมอไป อีกทั้งบริการจากผู้ให้บริการคลาวด์ก็มีการเปลี่ยนแปลงอยู่ตลอดเวลา ซึ่งอาจทำให้ฟีเจอร์บางอย่างไม่สามารถใช้งานได้ตามที่เคย หรืออาจไม่มีให้ใช้งานอีกต่อไป ส่งผลให้คุณต้องเผชิญกับปัญหาด้านความไม่เข้ากันของระบบ การกำหนดค่าที่ผิดพลาด ข้อจำกัดของทรัพยากร และความพร้อมใช้งานของบริการ

ประเด็นเหล่านี้ยังส่งผลกระทบต่อค่าใช้จ่ายของบริการคลาวด์อีกด้วย กล่าวคือ บริการต่าง ๆ ต้องมีความพร้อมใช้งานและมีประสิทธิภาพ แต่ก็ไม่ควรถูกจัดสรรเกินความจำเป็นจนสิ้นเปลืองเวลาและบประมาณ การปรับใช้ระบบคลาวด์จึงต้องเชื่อมโยงกับรูปแบบต้นทุนและระบบการเรียกเก็บเงินอย่างใกล้ชิด

บทเรียนนี้จะนำเสนอแนวคิดและเครื่องมือที่จำเป็นในการวิเคราะห์และแก้ไขปัญหาที่เกิดขึ้นระหว่างการปรับใช้และการย้ายระบบ โดยจะพิจารณาปัญหาที่เกิดขึ้นบ่อยและแนวทางการแก้ไขที่เป็นไปได้

วัตถุประสงค์ของบทเรียน (Module Objectives)

ในบทเรียนนี้ คุณจะได้:

- แก้ไขปัญหาการปรับใช้และการย้ายระบบคลาวด์
- เข้าใจประเด็นด้านต้นทุนของบริการคลาวด์

หน่วยที่ 1 การแก้ไขปัญหาการปรับใช้และการย้ายระบบ (Troubleshoot Deployment and Migration Issues)

การแก้ไขปัญหาที่เกี่ยวข้องกับการปรับใช้และการย้ายระบบไปยังคลาวด์จำเป็นต้องพิจารณาหลายปัจจัย เช่น ความเข้ากันได้ของระบบ การจัดสรรทรัพยากร การควบคุมการเข้าถึง และฟีเจอร์ที่ล้าสมัย คุณต้องเตรียมรับมือกับปัญหาเหล่านี้ในระหว่างขั้นตอนการวางแผนและดำเนินการเปลี่ยนแปลงทรัพยากรในระบบคลาวด์

1.1 การแก้ไขปัญหาการปรับใช้ระบบ (Troubleshoot Deployment Issues)

บทเรียนนี้นำเสนอประเด็นปัญหาที่อาจเกิดขึ้นพร้อมแนวทางการตรวจสอบ ซึ่งจัดหมวดหมู่เพื่อให้ง่ายต่อการวิเคราะห์ ปัญหาที่กล่าวถึงจะระบุແ่อมุนที่ควรตรวจสอบ โดยสอดคล้องกับ แนวทางการแก้ปัญหาของ CompTIA (CompTIA Troubleshooting Methodology) ที่ได้กล่าวไว้ก่อนหน้านี้ ซึ่งขั้นตอนเหล่านี้

จะช่วยให้คุณสามารถระบุขอบเขตและสาเหตุที่น่าจะเป็นไปได้ของปัญหา ก่อนเข้าสู่การวางแผนและการดำเนินการแก้ไข

รายการด้านล่างคือขั้นตอนพื้นฐานในกระบวนการแก้ไขปัญหา:

- ระบุปัญหา
- กำหนดขอบเขตของปัญหา
- ตั้งสมมุติฐานของสาเหตุที่เป็นไปได้ หรือพิจารณาความซัดเจน
- ทดสอบสมมุติฐานเพื่อระบุสาเหตุที่แท้จริง
- วางแผนดำเนินการ
- ดำเนินการแก้ไขหรือส่งต่อปัญหา
- ตรวจสอบให้แน่ใจว่าระบบทำงานได้อย่างสมบูรณ์
- ดำเนินมาตรการป้องกันในอนาคต
- วิเคราะห์สาเหตุที่แท้จริง (Root Cause Analysis)
- บันทึกผลการวิเคราะห์ การดำเนินการ และผลลัพธ์ตลอดกระบวนการ

1.2 ปัญหาความเข้ากันได้ของระบบ (Compatibility Problems)

ปัญหาที่เกิดขึ้นระหว่างการปรับใช้หรือย้ายระบบ (Deployment และ Migration) อาจมีสาเหตุมาจาก ความไม่เข้ากันของระบบ ซึ่งหนึ่งในจุดที่ควรตรวจสอบก่อนคือ การตั้งค่าการจำลองระบบ (Instance Virtualization Settings) โดยอาจเกี่ยวข้องกับแพลตฟอร์มการจำลอง, สถาปัตยกรรมของหน่วยประมวลผล, เครือข่าย และคุณลักษณะด้านการปรับแต่งประสิทธิภาพ

หากปัญหาคือ “การย้ายอินสแตนซ์ล้มเหลวนี้องจากปัญหาความเข้ากันโดยรวม” ควรตรวจสอบสิ่งต่อไปนี้:

- ประเภทหรือรุ่นของแพลตฟอร์มการจำลองระบบไม่ตรงกัน
- ขนาดหรือความเร็วของชนิดที่เก็บข้อมูลไม่ตรงกัน
- สถาปัตยกรรมของหน่วยประมวลผลไม่เข้ากัน เช่น ARM, Intel, หรือ AMD
- ประเภทของการเพิ่มประสิทธิภาพของอินสแตนซ์ไม่ตรงกัน เช่น สำหรับการประมวลผลหน่วยความจำ ประสิทธิภาพสูง หรือการจัดเก็บ
- ไลบรารีหรือคอมโพเนนต์ที่เป็นของบุคคลที่สามหรือที่สร้างขึ้นเองไม่สามารถทำงานร่วมกันได้

หากปัญหาคือ “อินสแตนซ์ไม่สามารถถือสารได้หลังการปรับใช้หรือย้ายระบบ” ควรตรวจสอบสิ่งต่อไปนี้:

- ไม่ได้ตั้งค่าภูมิการรับส่งข้อมูลขาเข้าและขาออกสำหรับแอปพลิเคชันหรือบริการ
- ไม่ได้ตั้งค่าการใช้งานเครือข่ายสำหรับไลบรารีหรือคอมโพเนนต์จากบุคคลที่สามหรือที่พัฒนาเอง
- ไม่ได้ตั้งค่าระบบเครือข่ายในแพลตฟอร์มการจำลองอย่างถูกต้อง

1.2.1 โควต้าในการให้บริการ (Service Quotas)

ผู้ให้บริการคลาวด์จะตั้งโควต้า (Quota) เพื่อกำหนดขีดจำกัดสูงสุดของการดำเนินการหรือการใช้ทรัพยากรในบัญชีลูกค้า โดยโควต้าบางรายการสามารถร้องขอเพิ่มได้ ส่วนบางรายการเป็นแบบตายตัว

หากปัญหาคือ “ไม่สามารถดำเนินการบางอย่างได้ ทั้งที่มีสิทธิ์แล้ว” ให้ตรวจสอบว่าทรัพยากรอยู่ภายใต้ข้อจำกัดของโควต้าหรือไม่

หากปัญหาคือ “ไม่สามารถใช้ทรัพยากรได้ แม้จะเคยใช้งานประเภทเดียวกันในอดีต” ให้พิจารณาดังนี้:

- ตรวจสอบคุณสมบัติของทรัพยากรว่ามีข้อจำกัดจากโควต้าหรือไม่
- พิจารณาว่าผู้ให้บริการกำหนดโควต้าจากภูมิภาค บทบาท ทรัพยากร หรือระดับบัญชีผู้ใช้ หากปัญหาคือ “กำลังร้องขอเพิ่มโควต้าผ่านแพลตฟอร์ม” ให้พิจารณาดังนี้:
 - ตรวจสอบว่าโควต้าถูกกำหนดในระดับใด (ภูมิภาค, บทบาท, ทรัพยากร, หรือบัญชี)
 - ตรวจสอบคอนโซลผู้ดูแลระบบเพื่อดูรายละเอียดว่าโควต้าถูกนำไปใช้ในลักษณะใด

1.2.2 การจำกัดอัตราการเรียกใช้ API (API Throttling)

แอปพลิเคชันอาจเข้าถึงหรือดึงข้อมูลผ่าน API ที่ผู้ให้บริการกำหนดไว้ อย่างไรก็ตาม ผู้ให้บริการอาจควบคุมอัตราการเข้าถึงเพื่อป้องกันไม่ให้ระบบลันหรือเพื่อควบคุมต้นทุน

เมื่อแอปพลิเคชันในคลาวด์ของคุณเติบโต อาจมีการเรียกใช้ API จำนวนมากในองค์กร, ผู้ให้บริการสาธารณชน หรือพันธมิตรทางธุรกิจ ซึ่งอาจทำให้เกิดการจำกัดอัตรา

หากปัญหาคือ “ประสิทธิภาพของแอปพลิเคชันต่ำกว่าที่คาดไว้” ให้พิจารณาดังนี้:

- ตรวจสอบการใช้ทรัพยากร เช่น CPU, หน่วยความจำ, ที่เก็บข้อมูล และเครือข่าย
- ตรวจสอบความสามารถของเครือข่ายระหว่างผู้ให้บริการและผู้ใช้งาน
- ตรวจสอบว่า API ที่เรียกใช้อยู่มีการจำกัดอัตราหรือไม่ และพิจารณาร้องขอเพิ่มหากมีเหตุผลควร

1.3 การกำหนดทรัพยากรณิดพลาด (Resource Allocation Misconfiguration)

การประเมินการจัดสรรทรัพยากร (Resource Allocation) เป็นสิ่งสำคัญในการรับรองว่าองค์กรมีบริการที่เพียงพอให้กับผู้ใช้ในรูปแบบที่คุ้มค่าที่สุด หากมีการจัดสรรทรัพยากรมากเกินหรือไม่เพียงพออาจส่งผลให้เกิดความซับซ้อนในระบบคลาวด์และส่งผลกระทบต่อประสบการณ์ของผู้ใช้ อีกทั้งองค์กรอาจต้องเสียค่าใช้จ่ายสำหรับทรัพยากรที่ไม่ได้ใช้งานจริง

1.3.1 การจัดสรรหน่วยประมวลผลกลาง (CPU Allocation)

หากปัญหาคือ CPU ถูกใช้งานเกินกำลัง (รองรับภาระงานไม่ไหว)

ให้พิจารณาสิ่งต่อไปนี้:

- ตรวจสอบเครื่องมือมอนิเตอร์ภายในระบบปฏิบัติการของ VM เพื่อดูการใช้งานที่สูง
- ตรวจสอบรายงานการใช้งานจากคอนโซลบริหารคลาวด์
- ตรวจสอบข้อจำกัดทางลิขสิทธิ์หรือไลเซนส์

หากปัญหาคือ CPU ใช้งานต่ำเกินไป (สามารถลดขนาด CPU ได้โดยไม่กระทบงาน)

ให้พิจารณาสิ่งต่อไปนี้:

- ตรวจสอบเครื่องมือมอนิเตอร์ใน OS สำหรับการใช้งานที่ต่ำ
- ตรวจสอบรายงานการใช้งานในคอนโซลคลาวด์
- เปรียบเทียบประเภทอินสแตนซ์กับภาระงานจริง

1.3.2 การจัดสรรหน่วยประมวลผลกราฟิก (GPU Allocation)

มีหลักการคล้ายกับ CPU ทั้งในกรณีที่ใช้งานเกินหรือน้อยเกินไป โดยควรตรวจสอบ:

- เครื่องมือมอนิเตอร์ของระบบ
- รายงานการใช้งานจากคอนโซลบริหารคลาวด์
- ความเหมาะสมระหว่างประเภทอินสแตนซ์กับภาระงานจริง

1.3.3 การจัดสรรหน่วยความจำ (Memory Allocation)

หากหน่วยความจำถูกจัดสรรมากเกินไป

- ตรวจสอบว่าการใช้งานจริงต่ำ
- ดูรายงานจากคอนโซลคลาวด์
- ตรวจสอบว่าขนาดอินสแตนซ์เกินความจำเป็น

หากหน่วยความจำจัดสรรไม่เพียงพอ

- ผู้ใช้อาจร้องเรียนเรื่องความล่าช้า
- ตรวจสอบว่ามีการใช้งานสูงหรือมีการสลับหน่วยความจำเมื่อไหร่บ่อย
- รายงานการอ่าน/เขียนผิดปกติอาจแสดงถึงปัญหานี้

1.3.4 การจัดสรรพื้นที่จัดเก็บและ I/O (Storage I/O and Capacity)

หาก I/O ช้าเกินไป

- ตรวจสอบว่าผู้ใช้ร้องเรียนเรื่องความเร็ว
- ดูรายงานการใช้งานจากคอนโซล
- เปรียบเทียบประเภทอินสแตนซ์กับภาระงานจริง

หาก I/O ต่ำกว่าปกติ (มากเกินความจำเป็น)

- ตรวจสอบจากรายงานการใช้งาน

หากความจุของดิสก์ไม่เพียงพอ

- อาจเกิดข้อผิดพลาดในการเขียนข้อมูล
- ตรวจสอบว่ามีการเก็บข้อมูลประเภทใด (ควรแยกข้อมูลสำคัญกับข้อมูลเก็บถาวร)
- พิจารณาใช้การบีบอัดหรือการลบข้อมูลช้าๆ ่อน

หากดิสก์มีความจุมากเกินไป

- ตรวจสอบรายงานความจุ
- ตรวจสอบใบแจ้งหนี้ว่ามีการคิดค่าบริการกับดิสก์ที่ไม่จำเป็นหรือไม่

1.3.5 การจัดสรรเครือข่าย (Network Allocation)

หากแบบดิวิดท์ไม่เพียงพอ

- ตรวจสอบจากคำอธิบายผู้ใช้
- ตรวจสอบว่าไม่มีการใช้งานเครือข่ายที่ไม่จำเป็น
- ตรวจสอบความเร็วอินเทอร์เน็ตจากผู้ให้บริการ
- ตรวจสอบการแบ่งเครือข่ายภายในองค์กรให้เหมาะสม

หากเกิดปัญหาความหน่วง (Latency)

- พิจารณาว่าข้อมูลถูกเก็บไว้ใกล้ผู้ใช้หรือไม่ เช่น Edge, CDN
- ตรวจสอบอุปกรณ์เครือข่าย เช่น Proxy, Firewall
- ตรวจสอบการจราจรในเครือข่าย และการแบ่งเครือข่ายที่เหมาะสม

1.3.6 แดชบอร์ดการตรวจสอบ

ผู้ให้บริการคลาวด์มีแดชบอร์ดที่แสดงสถานะการใช้งานของแต่ละอินสแตนซ์ เช่น

- การใช้ CPU
- ปริมาณรับส่งข้อมูลเครือข่าย
- การอ่าน/เขียนดิสก์
- การใช้งานเครดิต CPU
- และการตรวจสอบอื่น ๆ เพื่อประเมินการใช้งานและปรับขนาดให้เหมาะสมกับภาระงาน

1.4 การกำหนดค่าอินสแตนซ์และการทดสอบประสิทธิภาพ (Instance Configuration and Performance Testing)

อินสแตนซ์ของเครื่องเสมือน (Virtual Machine: VM) เป็นองค์ประกอบแรก ๆ ที่ควรทดสอบหลังจากการปรับใช้หรือลงมือย้ายระบบมายังคลาวด์ เพื่อยืนยันว่า VM เหล่านี้สามารถให้บริการได้อย่างมีเสถียรภาพ และมีประสิทธิภาพตามที่แอปพลิเคชันต้องการ

ปัญหาทั่วไปที่พบบ่อยคือเรื่องของการปรับขนาดอินสแตนซ์ (Instance Sizing) ซึ่งเกี่ยวข้องกับการจัดสรรทรัพยากรคอมพิวเตอร์ พื้นที่จัดเก็บ และเครือข่าย หากขนาดไม่เหมาะสม อาจส่งผลต่อทั้งด้านการทำงานและต้นทุนการใช้งานคลาวด์

หากปัญหาคือ “การกำหนดค่าอินสแตนซ์ผิดพลาดเนื่องจากการกำหนดขนาด”

ควรพิจารณา:

- ปัญหาการจัดขนาดทำให้มีการใช้งานทรัพยากรมากหรือน้อยเกินไป
- มีการใช้สคริปต์อัตโนมัติแทนการตั้งค่าด้วยมือ (การตั้งค่าด้วยมืออาจก่อให้เกิดข้อผิดพลาด)
- ทรัพยากรเครือข่ายหรือแบบดิวิดท์ถูกจัดขนาดไม่เหมาะสมสำหรับอินสแตนซ์

หากปัญหาคือ “การย้ายระบบและการบูรณาการกับผู้ให้บริการหรือแพลตฟอร์มล้มเหลว”

ควรพิจารณา:

- ปัญหาการจัดขนาดทำให้ทรัพยากรถูกใช้งานเกินหรือน้อยเกินไป
- ปัญหาการจัดขนาดทำให้ไม่สามารถย้าย VM หรือบริการจากระบบทภายนอกค์กรได้
- ปัญหาการจัดขนาดทำให้ไม่สามารถสร้างหรือใช้งานทรัพยากรจัดเก็บข้อมูลได้
- ข้อจำกัดของสเปคทำให้ไม่สามารถใช้ทรัพยากรเครือข่ายได้

หากปัญหาคือ “อินสแตนซ์ถูกกำหนดค่าไม่ถูกต้อง”

ควรพิจารณา:

- ขั้นตอนการตั้งค่าด้วยมือมีความถูกต้องหรือไม่
- สคริปต์ที่ใช้มีความถูกต้องหรือไม่
- เทมเพลตของ VM หรือคอนเทนเนอร์ถูกตั้งค่าถูกต้องหรือไม่
- Tag ของอินสแตนซ์มีความถูกต้องเพื่อให้สามารถตั้งค่าไฟร์วอลล์และเส้นทางเครือข่ายได้
- เครื่องมืออัตโนมัติเช่น Ansible, Chef หรือ Puppet ทำงานหรือกำหนดค่าไม่ถูกต้องหรือไม่

หากปัญหาคือ “การย้ายระบบและการบูรณาการกับผู้ให้บริการหรือแพลตฟอร์มล้มเหลว”

ควรพิจารณา:

- แอปพลิเคชันเก่าอาจไม่สามารถทำงานได้บนคลาวด์
- เส้นทางการย้ายระบบต้องผ่านการตรวจสอบเงื่อนไขเบื้องต้นทั้งหมด
- การย้ายระบบแบบอัตโนมัติหรือกิ่งอัตโนมัติมีการกำหนดค่าไม่ถูกต้อง
- กระบวนการย้ายระบบแบบ manual ต้องทำให้ครบถ้วนทุกขั้นตอน

- ตรวจสอบความเข้ากันได้ของเวอร์ชันระบบปฏิบัติการ (OS)
- เปิดใช้งานการจัดการระยะไกล เช่น SSH หรือ RDP และวิธีอย่าง
- การย้ายระบบจากคลาวด์หนึ่งไปยังอีกคลาวด์หนึ่งรองรับโดยบริการหรือแอปพลิเคชันหรือไม่
- ระบบใช้บริการ ซอฟต์แวร์ หรือรูปแบบข้อมูลที่เป็นกรรมสิทธิ์ที่อาจขัดขวางการย้ายข้อมูลหรือไม่
- ตรวจสอบความเข้ากันได้ของฐานข้อมูล (เช่น relational กับ relational, non-relational กับ non-relational)

กรณีผู้ให้บริการมีการขายเกินความสามารถ (Oversubscription)

ผู้ให้บริการคลาวด์บางรายอาจจัดสรรทรัพยากรมากกว่าที่โครงสร้างพื้นฐานของตนรองรับ โดยคาดว่าผู้ใช้งานจำนวนมากจะไม่ใช้ทรัพยากรตามที่ขอไว้จริง ส่งผลให้เกิดปัญหาประสิทธิภาพและความพร้อมใช้งานลดลง หากสนใจว่าเป็นกรณีนี้ ควร:

- ตรวจสอบการมอนิเตอร์ประสิทธิภาพและ log ต่าง ๆ
- เปรียบเทียบผลลัพธ์กับข้อตกลงระดับบริการ (SLA) เพื่อดูว่ามีการเบี่ยงเบนหรือไม่
- วิเคราะห์การใช้บริการเพื่อดูว่ามีจุดใดที่เกิดจากการ oversubscription

หากปัญหาคือ ผู้ให้บริการมีการ oversubscription

ควรดำเนินการดังนี้:

- ติดต่อฝ่ายสนับสนุนของผู้ให้บริการคลาวด์เพื่อตรวจสอบ
- ขอทรัพยากรแบบเฉพาะสำหรับบริการที่สำคัญ
- พิจารณาถ่ายบางบริการกลับไปไว้ในระบบภายในหรือคลาวด์แบบ private

1.5 การกำหนดสิทธิ์และสิทธิพิเศษผิดพลาด (Permissions and Privilege Misconfiguration)

การเข้าถึงทรัพยากร (Privileges) ถูกจัดการผ่านการกำหนดบทบาท (Role) ให้กับหลักความปลอดภัย เช่น ผู้ใช้หรือกลุ่ม โดยบทบาทนั้นจะระบุถึงสิทธิ์ (Permissions) ที่เกี่ยวข้อง และขอบเขตของการควบคุม ที่สามารถดำเนินการได้

การแก้ไขปัญหาการเข้าถึงและสิทธิ์มักเกี่ยวข้องกับการกำหนดสิทธิ์ผิดพลาด (Authorization) หรือ ปัญหาการยืนยันตัวตน (Authentication)

บทบาท (Role) คือ ระดับการเข้าถึงที่ถูกกำหนดไว้ล่วงหน้า ผู้ให้บริการคลาวด์มักจัดเตรียมบทบาท พื้นฐานไว้ให้เพื่อความสะดวกในการบริหารจัดการ แต่ผู้ใช้สามารถสร้างบทบาทที่กำหนดเองได้ หากบทบาท ที่ไม่มีต่อไปโดยที่ไม่สามารถต้องการขององค์กร

1.5.1 บทบาท (Roles)

บทบาท ทำหน้าที่กำหนดระดับการเข้าถึงของผู้ใช้ต่อแดชบอร์ด อินสแตนซ์ บริการ และพีเจอร์อื่น ๆ บนคลาวด์ โดยแต่ละบทบาทจะมีชุดของสิทธิ์ (Permissions) ที่แนบอยู่ และสามารถใช้บทบาทที่มีอยู่แล้ว หรือสร้างบทบาทใหม่เฉพาะกิจได้ตามต้องการ

ปัญหาการกำหนดสิทธิ์ไม่ครบถ้วน (Missing or Incomplete Privileges)

เป็นกรณีที่ผู้ใช้ควรจะสามารถเข้าถึงทรัพยากรได้แต่กลับไม่สามารถทำได้

หากปัญหาคือ ผู้ใช้หรือกลุ่มผู้ใช้มาตรฐานไม่สามารถเข้าถึงทรัพยากรที่ควรเข้าถึงได้ ควรพิจารณา:

- บัญชีผู้ใช้อยู่ในกลุ่มหรือบทบาทที่เหมาะสมแล้วหรือยัง
- บทบาทที่เข้มโงยิงกับทรัพย์สินต้องหรือไม่
- สิทธิ์ที่แนบกับบทบาทครบถ้วนหรือไม่
- บัญชีผู้ใช้มีการเข้มโงยิงกับ Subscription ที่เหมาะสมหรือไม่

1.5.2 ปัญหาการยกระดับสิทธิ์ (Privilege Escalation Issues)

เป็นกรณีที่ผู้ดูแลระบบไม่สามารถเข้าสู่ระบบหรือดำเนินการในระดับผู้ดูแลได้

หากปัญหาคือ ผู้ดูแลระบบไม่สามารถเข้าสู่ระบบอินสแตนซ์ VM ด้วยสิทธิ์ผู้ดูแลได้ ควรพิจารณา:

- บทบาทที่กำหนดไว้ออนุญาตให้เข้าสู่ระบบหรือแค่ให้จัดการ VM ภายนอก
- ตรวจสอบว่าบทบาทนั้นมีสิทธิ์ “Administrator Login” หรือไม่ (เช่น Azure Virtual Machine Administrator Login)
- หากบทบาทนั้non อนุญาตให้จัดการ VM แต่ไม่สามารถเข้าสู่ระบบได้ เช่น Azure Virtual Machine Contributor
- หรือบทบาทนั้non อนุญาตให้เข้าสู่ระบบได้แต่ในระดับผู้ใช้ทั่วไป เช่น Azure Virtual Machine User Login

หากปัญหาคือ ไม่สามารถยกระดับสิทธิ์ในอินสแตนซ์ VM ได้ ควรพิจารณา:

- ผู้ใช้รู้รหัสผ่านของผู้ดูแลหรือ root หรือไม่
- ผู้ใช้มีสิทธิ์ sudo ใน Linux หรือไม่
- ผู้ใช้มีสิทธิ์ Runas ใน Windows หรือไม่
- ผู้ใช้อยู่ในกลุ่ม local administrator ที่ถูกต้องหรือไม่ เช่น Administrators ใน Windows หรือ wheel ใน Linux

1.6 การแก้ไขปัญหาระบบบริการและแอปพลิเคชัน (Troubleshoot Services and Applications)

บริการและแอปพลิเคชันถือเป็นหัวใจสำคัญของโซลูชันด้าน IT เนื่องจากเป็นองค์ประกอบที่ผู้ใช้และลูกค้าต้องโต้ตอบอยู่ทุกวัน จึงต้องมีความเสถียรและเชื่อถือได้ หลังการปรับใช้ (Deployment) หรือการย้าย

ระบบ (Migration) อาจพบปัญหาต่าง ๆ เช่น นิยามของคอมโพเนนต์ที่ล้าสมัย, ฟีเจอร์ที่ถูกยกเลิก, แอปพลิเคชันที่ไม่สามารถใช้งานร่วมกับระบบคลาวด์, ปัญหาด้านภูมิภาค, บริการบางอย่างไม่พร้อมใช้งานในบางพื้นที่, หรือ API ที่ถูกจำกัดความถี่การเรียกใช้งาน (Throttling) ซึ่งควรตรวจสอบประเด็นต่อไปนี้ในการวิเคราะห์และแก้ไข

1.6.1 ความไม่เข้ากันหรือการเลิกใช้งานของบริการหรือแอปพลิเคชัน

หากปัญหาคือ คอมโพเนนต์ของซอฟต์แวร์ล้าสมัยหรือถูกเลิกใช้งาน ควรพิจารณา:

- คอมโพเนนต์ของแอปพลิเคชันมีช่องโหว่ที่เป็นที่รู้จักหรือไม่
- ผู้นำที่มีความสามารถในการปรับปรุง (refactor) หรือออกแบบใหม่ (rearchitect) ให้สามารถทำงานร่วมกับบริการคลาวด์ได้หรือไม่
- แอปพลิเคชันยังไม่ได้รับการปรับปรุง (refactor) หรือออกแบบใหม่ (rearchitect) ให้สามารถทำงานร่วมกับบริการคลาวด์ได้หรือไม่

หากปัญหาคือ บริการไม่สามารถใช้งานได้อีกต่อไป ควรพิจารณา:

- บริการดังกล่าวถูกเลิกให้บริการโดยผู้ให้บริการคลาวด์หรือไม่
- บริการดังกล่าวถูกควบรวมกับฟังก์ชันอื่นหรือไม่
- บริการนั้นไม่ปลอดภัยอีกต่อไปหรือไม่

1.6.2 แนวทางการแก้ไขปัญหาบริการและแอปพลิเคชันในระบบคลาวด์

หากแอปพลิเคชันถูกนำไปใช้ในคลาวด์โดยไม่ได้มีการวิเคราะห์หรือทดสอบที่เพียงพอ อาจจำเป็นต้องดำเนินการตามแนวทางต่อไปนี้:

- อัปเดตซอฟต์แวร์เป็นเวอร์ชันล่าสุด
- ตรวจสอบให้แน่ใจว่าแอปพลิเคชันสามารถใช้งานร่วมกับระบบคลาวด์ได้
- พิจารณาปรับปรุง (Refactor), ออกแบบใหม่ (Rearchitect) หรือเปลี่ยนแอปพลิเคชันเดิมที่ไม่รองรับคลาวด์

1.6.3 ปัญหาเกี่ยวกับทรัพยากรโครงสร้างพื้นฐาน (Infrastructure Resources)

หากปัญหาคือ ทรัพยากรโครงสร้างพื้นฐาน เช่น CPU, หน่วยความจำ, ที่เก็บข้อมูล หรือเครือข่าย ไม่สามารถใช้งานได้อีกต่อไป ควรพิจารณา:

- ผู้นำที่มีความสามารถในการปรับปรุง (refactor) หรือออกแบบใหม่ (rearchitect) ให้สามารถทำงานร่วมกับระบบคลาวด์
- ฮาร์ดแวร์ถูกแทนที่ด้วยตัวเลือกที่มีประสิทธิภาพหรือคุ้มค่ากว่า

1.6.4 แนวทางแก้ไขปัญหาเมื่อแอปพลิเคชันต้องการฮาร์ดแวร์เฉพาะทาง

บางแอปพลิเคชันอาจต้องใช้ฮาร์ดแวร์พิเศษที่ไม่สามารถจำลองได้ง่ายในระบบคลาวด์ แนวทางที่ควรพิจารณา ได้แก่:

- ย้ายบริการหรือแอปพลิเคชันกลับไปยังอาร์ดแวร์ที่สนับสนุนในองค์กร (On-Premises)
- ย้ายระบบไปยังอาร์ดแวร์ที่ยังรองรับได้หากเป็นไปได้
- ศึกษาทางเลือกในการจำลอง (Virtualization) แบบอื่น
- ปรับแต่งอาร์ดแวร์ให้เหมาะสมกับแอปพลิเคชัน
- ใช้เทคโนโลยีคลาวด์ เช่น Load Balancer หรือ Web Application Firewall เพื่อช่วยให้แอปพลิเคชันเดิมสามารถใช้งานได้อย่างปลอดภัยและมีประสิทธิภาพในระบบใหม่

1.7 การยกเลิกการใช้งานของฟังก์ชัน (Deprecation of Functionality)

หนึ่งในสาเหตุที่ทรัพยากรหรือบริการไม่สามารถใช้งานได้อีกต่อไป คือการที่ฟังก์ชันบางอย่างถูกยกเลิกการใช้งาน (Deprecated) ระบบคลาวด์มีการพัฒนาอย่างรวดเร็ว และผู้ให้บริการต่างแข่งขันกันเพื่อเสนอความสามารถที่ทันสมัย ปลอดภัย และมีประสิทธิภาพสูงที่สุด ส่งผลให้บริการบางรายการจะถูกยกเลิกหรือถูกรวบเข้ากับบริการอื่น เพื่อให้บริการของผู้ให้บริการมีความทันสมัย มีประสิทธิภาพ และคุ้มค่ามากขึ้น

1.8 การหยุดให้บริการของระบบคลาวด์ (Cloud Service Outages)

การหยุดให้บริการอาจเกิดขึ้นได้ในหลายระดับ ตั้งแต่โครงสร้างพื้นฐานในพื้นที่ เช่น ไฟฟ้าดับหรืออินเทอร์เน็ตล่ม ศูนย์ข้อมูลขององค์กรเอง พื้นที่ระดับภูมิภาค ไปจนถึงผู้ให้บริการระบบคลาวด์ ผู้ดูแลระบบ จำเป็นต้องประเมินขอบเขตของเหตุขัดข้องและวางแผนเพื่อลดผลกระทบจากเหตุการณ์ลักษณะเดียวกันในอนาคต

หากปัญหาคือการตรวจสอบว่ามีการหยุดให้บริการบางส่วนหรือทั้งหมด ควรพิจารณา:

- ตรวจสอบว่ามีไฟฟ้าหรืออินเทอร์เน็ตในพื้นที่หรือภูมิภาคดับหรือไม่
- แยกแยะพื้นที่ที่ได้รับผลกระทบและขอบเขตของปัญหา
- ตรวจสอบระบบสำรองและระบบที่มีความทนทานต่อความผิดพลาดว่ายังทำงานได้หรือไม่
- ตรวจสอบว่ามีการขยายระบบไปยังภูมิภาคหรือศูนย์ข้อมูลอื่นเพื่อรับความต้องการแล้วหรือไม่
- ตรวจสอบข้อตกลงระดับการให้บริการ (SLA) เพื่อดูผลกระทบจากเหตุการณ์หยุดให้บริการ

หากปัญหาคือการวางแผนเพื่อลดผลกระทบจากการหยุดให้บริการในอนาคต ควรพิจารณา:

- ทบทวนกระบวนการทำงาน บันทึกเหตุการณ์ และรายงานจากผู้ใช้เพื่อประเมินผลกระทบ
- ระบุและแก้ไขปัญหาในระบบสำรองและระบบที่มีความทนทานต่อความผิดพลาด
- ตรวจสอบว่าเหตุขัดข้องเกิดจากผู้ให้บริการไฟฟ้า ผู้ให้บริการอินเทอร์เน็ต ระบบในองค์กร หรือผู้ให้บริการระบบคลาวด์
- วางแผนเพิ่มความซ้ำซ้อนให้กับระบบหรือโครงสร้างพื้นฐานที่ล้มเหลว

1.9 การให้บริการตามภูมิภาค (Regional Service Availability)

ผู้ให้บริการระบบคลาวด์ตั้งศูนย์ข้อมูลในหลากหลายภูมิภาคทั่วโลก โดยแต่ละแห่งมีขีดความสามารถ และข้อจำกัดด้านกฎหมายท้องถิ่นที่แตกต่างกัน ทำให้บริการบางประเภทไม่สามารถใช้งานได้ในบางพื้นที่ เหตุผลที่บริการระบบคลาวด์อาจไม่สามารถให้บริการในบางภูมิภาค ได้แก่:

- ข้อจำกัดด้านโครงสร้างพื้นฐาน เช่น ไฟฟ้า อินเทอร์เน็ต สถานที่ตั้ง หรือความมั่นคงทางกายภาพ
- ประเด็นทางกฎหมายศาสตร์ เช่น ข้อจำกัดจากกฎหมายท้องถิ่นที่ส่งผลกระทบต่อการให้บริการ

ผู้ให้บริการระบบคลาวด์จะมีรายการบริการที่สามารถใช้งานได้ในแต่ละภูมิภาคอย่างชัดเจน

หากหลังจากการปรับใช้หรือการย้ายข้อมูลแล้วพบว่าบริการบางอย่างไม่สามารถใช้งานในภูมิภาคนั้นได้ ควรพิจารณา:

- ตรวจสอบรายการบริการที่รองรับในภูมิภาคนั้น ก่อนการติดตั้งหรือย้ายระบบ
- ตรวจสอบว่ามีภูมิภาคใดล้าสุดที่สามารถให้บริการได้หรือไม่ และพิจารณาใช้บริการจากภูมิภาคนั้นแทน

หน่วยที่ 2 ข้อควรพิจารณาเกี่ยวกับค่าใช้จ่ายในการใช้งานระบบคลาวด์ (Cost Considerations Related to Cloud Usage)

การใช้งานระบบคลาวด์อย่างมีประสิทธิภาพจำเป็นต้องมีการบริหารจัดการต้นทุนอย่างจริงจัง โดยขั้นตอนแรกคือการทำความเข้าใจบริการแบบสมัครสมาชิก (subscription services) และรูปแบบการคิดค่าบริการ (billing models) ซึ่งเป็นพื้นฐานที่กำหนดวิธีการเลือกและตั้งค่าทรัพยากรระบบคลาวด์ขององค์กร

หนึ่งในประโยชน์สำคัญของการนำระบบคลาวด์มาใช้งานหรือการย้ายระบบไปยังคลาวด์ คือความสามารถในการควบคุมและจัดการต้นทุนได้อย่างมีประสิทธิภาพ ผู้ให้บริการระบบคลาวด์มีเครื่องมือและฟีเจอร์มากมายที่ช่วยให้คุณสามารถเลือกการตั้งค่าที่เหมาะสมกับการใช้งานได้ โดยการใช้ทรัพยากระบบทราเวล (metered) และติดแท็ก (tagged) เพื่อบุนเดอร์ธุรกิจที่ใช้งานแต่ละรายการ

หัวข้อนี้กล่าวถึงความสัมพันธ์ระหว่างการจัดสรรทรัพยากรกับต้นทุน รวมถึงการใช้เครื่องมือคำนวณราคา (pricing calculators) เพื่อประมาณค่าใช้จ่าย การตรวจสอบต้นทุน (cost monitoring) การติดแท็กเพื่อจัดการค่าใช้จ่าย และการปรับขนาดทรัพยากรให้เหมาะสม (rightsizing) โดยคำนึงถึงงบประมาณที่มีอยู่

2.1 ต้นทุนของบริการแบบสมัครสมาชิก (Subscription Services Costs)

บริการแบบสมัครสมาชิก (Subscription Services) คือรูปแบบของการอนุญาตใช้งานซอฟต์แวร์หรือบริการที่ผู้ใช้หรือองค์กรต้องจ่ายค่าธรรมเนียมตามช่วงเวลาที่กำหนดไว้ เช่น รายเดือนหรือรายปี เพื่อให้สามารถเข้าถึงบริการหรือทรัพยากรต่าง ๆ ได้ตลอดระยะเวลาของการสมัครใช้งาน โดยรูปแบบการคิดค่าบริการแบบ “จ่ายตามตัวตนผู้ใช้” (pay-per-identity) เป็นแนวทางที่พึ่งได้บ่อยในบริการคลาวด์

บริการแบบสมัครสมาชิกสอดคล้องกับลักษณะของระบบคลาวด์ที่ให้บริการตามความต้องการ (on-demand self-service) เพราะผู้บริโภคสามารถเลือกสมัครหรือยกเลิกบริการได้อย่างยืดหยุ่น

ตัวอย่างเช่น Microsoft Office 365 เป็นบริการแบบ Software as a Service (SaaS) ที่ต้องชำระค่าบริการรายเดือนหรือรายปี (โดยมักมีส่วนลดสำหรับการชำระรายปี) เพื่อเข้าถึงแอปพลิเคชันอย่าง Word, Excel, Outlook ฯลฯ รุ่นสำหรับองค์กรจะมีค่าใช้จ่ายตามจำนวนผู้ใช้งานต่อเดือน พร้อมเพิ่มสิทธิ์ในการใช้งานบริการคลาวด์อื่น ๆ เช่น Microsoft Exchange, Teams และ SharePoint

อีกตัวอย่างคือ การสมัครใช้งานแบบรายปีของ Amazon Web Services (AWS) Elastic Compute Cloud (EC2) ซึ่งเป็นบริการเครื่องเสมือน (VM) ที่ประกอบด้วยการจัดสรรทรัพยากรหารดแวร์ไว้ล่วงหน้า เช่น CPU หน่วยความจำ และเครือข่าย โดยคุณสามารถเลือกซื้อแบบสมัครสมาชิกรายปีสำหรับอินสแตนซ์เหล่านี้เพื่อควบคุมค่าใช้จ่ายและทำให้การวางแผนด้านการเงินคาดการณ์ได้ง่ายกว่ารูปแบบการจ่ายตามการใช้งานจริง (pay-as-you-go)

การบริหารจัดการการสมัครใช้งานสามารถทำผ่านอินเทอร์เฟซแบบเว็บที่แสดงรายละเอียดของบริการ ต้นทุน และข้อตกลงการใช้งาน ตัวอย่างเช่น Google Cloud Platform (GCP) มี Cloud Billing Account สำหรับจัดการค่าใช้จ่ายและการชำระเงิน โดยรวมถึงเครื่องมือในการประมาณและวิเคราะห์ค่าใช้จ่าย

การบริหารการสมัครใช้งานมักเป็นความรับผิดชอบร่วมกันระหว่างผู้ดูแลระบบคลาวด์และฝ่ายการเงิน เพราะค่าใช้จ่ายจากบริการคลาวด์จะถูกรวบอยู่ในงบประมาณโดยรวมขององค์กร

การเปรียบเทียบราคาของบริการแบบสมัครสมาชิกนั้นอาจมีความซับซ้อน เนื่องจากผู้ให้บริการแต่ละรายมีการตั้งราคา การรวมบริการ และส่วนลดในรูปแบบที่แตกต่างกัน อีกทั้งราคายังเปลี่ยนแปลงบ่อยทำให้การเปรียบเทียบระยะยาวทำได้ยาก

แนวทางการเปรียบเทียบต้นทุนบริการคลาวด์ ได้แก่:

- การศึกษาเปรียบเทียบจากบุคคลที่สาม (Third-party studies)
- ใช้บริการผู้ให้บริการคลาวด์แบบจัดการ (Managed CSPs)
- วิเคราะห์และเปรียบเทียบต้นทุนด้วยตนเอง
- เปรียบเทียบต้นทุนการสมัครสมาชิกกับองค์กรที่มีลักษณะคล้ายกัน

2.2 เครื่องคำนวณราคาและการวิเคราะห์ต้นทุน (Price Calculators and Cost Analysis)

ผู้ให้บริการคลาวด์รายใหญ่ (Cloud Service Providers - CSPs) ต่างก็มีเครื่องมือคำนวณราคาผ่านเว็บไซต์ที่ออกแบบมาเพื่อช่วยให้ผู้ดูแลระบบและผู้มีอำนาจตัดสินใจในองค์กรสามารถประเมินต้นทุนการใช้บริการคลาวด์ได้อย่างแม่นยำ เครื่องคำนวณราคนี้จะคำนวณค่าบริการจากการตั้งค่าหรือการสมัครใช้งานที่แตกต่างกัน พร้อมแยกต้นทุนออกเป็นหมวดหมู่เพื่อให้เข้าใจได้ง่ายว่าการปรับเปลี่ยนทรัพยากรแต่ละอย่าง

ส่งผลต่อค่าใช้จ่ายในการดำเนินงาน (Operating Expenses – OpEx) อาย่างไร โดยสามารถสร้างการประมาณราคาได้จากห้ายบริการของระบบคลาวด์

ตัวอย่างเช่น GCP Price Calculator ของ Google Cloud Platform ที่แสดงผลสรุปต้นทุนโดยรวมพร้อมรายละเอียดแยกตามอินสแตนซ์ (compute engine) และยอดรวมทั้งหมดในหน้าหลัก ผู้ใช้สามารถดาวน์โหลดหรือทำสำเนารายการประมาณราคайдี

ในกรณีที่องค์กรใช้บริการผู้ให้บริการคลาวด์แบบมีการจัดการ (Managed CSPs) พวกเขาก็จะช่วยดูแลการบริหารระบบคลาวด์บางส่วนหรือทั้งหมดโดยใช้ผู้เชี่ยวชาญเฉพาะด้าน และอาจให้คำแนะนำในการเลือกแพ็กเกจหรือรูปแบบการสมัครใช้งานที่เหมาะสมกับงบประมาณขององค์กรได้

ตัวอย่างส่วนลดที่ CSPs เสนอให้แก่ผู้ใช้งาน ได้แก่:

- AWS (Amazon Web Services): Reserved Instances (การจองอินสแตนซ์ล่วงหน้าเพื่อรับส่วนลด)
- Azure: Azure Reservations (การจองบริการล่วงหน้าเพื่อรับส่วนลด)
- Google Cloud Platform (GCP): Committed-use discounts (ส่วนลดเมื่อให้คำมั่นว่าจะใช้บริการในปริมาณที่แน่นอนเป็นระยะเวลา)

ผู้บริโภคบริการคลาวด์ควรตระหนักว่า ตลาดบริการคลาวด์มีการแข่งขันสูงมาก ผู้ให้บริการจึงออกแบบโครงสร้างราคาที่ดึงดูดใจ โดยห้ายองค์กรในปัจจุบันใช้แนวทาง มัลติคลาวด์ (Multi-cloud) คือ การใช้บริการคลาวด์จากห้ายผู้ให้บริการควบคู่กับแพลตฟอร์มคลาวด์ส่วนตัว ซึ่งช่วยให้มีความยืดหยุ่นทั้งในเรื่องต้นทุนและประสิทธิภาพของระบบ

2.3 รูปแบบการเรียกเก็บค่าบริการระบบคลาวด์ (Cloud Billing Models)

ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers - CSPs) เสนอทรัพยากรการประมาณผลในหลากหลายรูปแบบ ซึ่งแต่ละรูปแบบมีผลต่อค่าใช้จ่ายขององค์กรอย่างมีนัยสำคัญ การบริหารจัดการอย่างรอบคอบจึงเป็นสิ่งจำเป็นเพื่อให้ได้ผลตอบแทนสูงสุดจากการลงทุนในระบบคลาวด์

ตัวอย่างรูปแบบของอินสแตนซ์ (Instance) และโมเดลการเรียกเก็บเงิน ได้แก่:

- Dedicated Host with VM Instance (ไฮสต์เดิร์ฟสำหรับ VM):

ผู้ให้บริการจะจัดสรรเซิร์ฟเวอร์จริงให้ใช้งานโดยเฉพาะ สามารถรัน VM ได้หลายตัว ราคาขึ้นอยู่กับสเปกของเซิร์ฟเวอร์ (CPU, หน่วยความจำ, พื้นที่เก็บข้อมูล ฯลฯ) เหมาะสมสำหรับองค์กรที่มีข้อกำหนดด้านการอนุญาตสิทธิ์การใช้งาน (Licensing) หรือการปฏิบัติตามกฎหมาย (Compliance)

- On-Demand Instance (จ่ายตามการใช้งานจริง):

จ่ายค่าบริการตามระยะเวลาการใช้งานแบบวินาที ไม่มีข้อผูกพันระยะยาว เหมาะกับวิธีก็荷ดที่ใช้งานชั่วคราวหรือไม่แน่นอน

- **Reserved Instance** (อินสแตนซ์แบบสำรอง):
ผู้ให้บริการสำรองทรัพยากรให้ใช้งานล่วงหน้า ซึ่งสามารถช่วยลดต้นทุนได้ เนื่องจากมีการรับเงินโดยที่มีรูปแบบการใช้งานที่คาดการณ์ได้ในระยะยาว
- **Spot Instance** (อินสแตนซ์ราคาพิเศษตามความว่าง):
เข้าถึงทรัพยากรคอมพิวต์ที่ไม่ได้ใช้งานในเดี๋ยวนี้ของผู้ให้บริการในขณะนั้น ซึ่งมีราคาถูกมาก แต่ไม่รับประกันว่าจะมีเสมอ เนื่องจากงานที่สามารถยืดหยุ่นหรือขาดช่วงได้โดยไม่ส่งผลเสีย

ตัวอย่างการใช้งานตามสถานการณ์:

- **Dedicated Host:**
องค์กรในอุตสาหกรรมที่ถูกควบคุมอย่างเข้มงวดและต้องการจัดการสิทธิ์การใช้งาน OS หรือแอปพลิเคชัน ด้วยตนเอง จึงเลือกใช้เซิร์ฟเวอร์เฉพาะที่ไม่ได้แบ่งใช้กับผู้อื่น และมีภาระการดูแลอย่างต่อเนื่องให้แก่ผู้ให้บริการคลาวด์
- **On-Demand Instance:**
ที่ไม่ต้องการสร้างสภาพแวดล้อมการเรียนรู้ โดยการใช้งานแปรผันตั้งแต่ 0-20 ชั่วโมงต่อเดือน จึงเหมาะสมกับโมเดลจ่ายตามการใช้งานจริง
- **Reserved Instance:**
องค์กรมีการประมวลผลข้อมูลขนาดใหญ่ทุกวันที่ 20 ของเดือน และสามารถวางแผนล่วงหน้าได้ จึงเลือกใช้อินสแตนซ์แบบสำรองเพื่อประหยัดค่าใช้จ่าย
- **Spot Instance:**
งานเรนเดอร์ภาพที่ไม่มีข้อจำกัดเรื่องเวลา และสามารถรองรับการทำงานที่ยืดหยุ่นสูง จึงเลือกใช้ Spot Instance ที่มีราคาประหยัดมากกว่า Dedicated Instance

2.4 การจัดสรรทรัพยากร (Resource Allocation)

ในการพูดถึงการจัดสรรทรัพยากรบนคลาวด์ (Cloud Resource Allocation) มักเน้นที่การเพิ่มประสิทธิภาพในการทำงาน (Performance Optimization) เป็นหลัก อย่างไรก็ตาม การเพิ่มประสิทธิภาพด้านต้นทุน (Cost Optimization) ก็มีความสำคัญไม่แพ้กัน

การจัดสรรทรัพยากรไม่เหมาะสม ไม่ว่าจะเป็นการจัดสรรมากเกินไปหรือไม่เพียงพอ ทั้งในส่วนของการประมวลผล (Compute), การจัดเก็บข้อมูล (Storage), และเครือข่าย (Network) ย่อมส่งผลกระทบต่อประสิทธิภาพของผู้ใช้และประสิทธิภาพของบริการโดยรวม ดังนั้น องค์กรจึงต้องบริหารการใช้ทรัพยากรอย่างคุ้มค่าและมีประสิทธิภาพสูงสุด มีแนวทางสำคัญในการจัดการและวิเคราะห์ปัญหาด้านต้นทุน ได้แก่

การวัดผลการใช้ทรัพยากร (Resource Metering)

ช่วยให้มองเห็นและควบคุมได้ว่าแต่ละทรัพยากรถูกใช้อย่างไร เมื่อเปรียบเทียบกับต้นทุน ช่วยในการติดตาม การใช้และปรับทรัพยากรให้เหมาะสมกับการใช้งานจริง

การติดแท็ก (Tagging)

ช่วยให้เห็นภาพการใช้ทรัพยากรของแต่ละหน่วยงานธุรกิจ (Business Units) ชัดเจนขึ้น เช่น แท็กว่า VM เครื่องใดเป็นของแผนกพัฒนา หรือแผนกการตลาด เป็นต้น

ข้อมูลเหล่านี้สามารถนำมาใช้เพื่อวิเคราะห์ต้นทุนและจัดการระบบการคิดค่าบริการรายหน่วยงาน (Showback/Chargeback)

การปรับขนาดทรัพยากรให้เหมาะสม (Rightsizing)

การใช้บริการคลาวด์มีความต้องการที่เปลี่ยนแปลงไปตามเวลา เช่น การเติบโตของธุรกิจหรือช่วงเวลา ตามฤดูกาล (Seasonal Cycles)

ดังนั้น การกำหนดขนาดของอินสแตนซ์หรือแอปพลิเคชันเพียงครั้งเดียวแล้วปล่อยไว้ไม่เพียงพอ

Rightsizing คือกระบวนการต่อเนื่องในการวิเคราะห์การใช้งานจริง เพื่อนำมาปรับขนาดทรัพยากร ให้สมดุลระหว่างประสิทธิภาพและต้นทุน

2.5 การวัดผลการใช้ทรัพยากร (Resource Metering)

ลักษณะเฉพาะของคลาวด์ที่สามารถวัดผลการใช้ทรัพยากรได้ คือปัจจัยสำคัญที่ทำให้รูปแบบการคิดค่าบริการตามการใช้งานจริงเป็นไปได้ ผู้ให้บริการคลาวด์จะติดตามการใช้ทรัพยากรของผู้ใช้งาน และคิดค่าบริการ ตามปริมาณที่ใช้จริง ซึ่งองค์กรสามารถเลือกวิธีเข้าถึงทรัพยากรได้ เช่น การใช้แบบตามต้องการ (On-demand), แบบจองล่วงหน้า (Reserved) ฯลฯ ซึ่งล้วนมีผลต่อค่าใช้จ่าย

องค์กรยังสามารถเข้าถึงข้อมูลการใช้งานของตนเอง เพื่อใช้ในการปรับแต่งทรัพยากรให้เหมาะสม กับลำดับความสำคัญและความต้องการด้านประสิทธิภาพของแต่ละบริการ นอกจากนี้ยังสามารถกำหนดแท็ก ให้กับทรัพยากรแต่ละรายการ เพื่อให่ง่ายต่อการระบุและจัดการอย่างมีประสิทธิภาพสูงสุด

รายงานการวัดผลการใช้ทรัพยากรจะแสดงเป็นข้อมูลที่นำไปปฏิบัติได้ โดยสามารถบอกได้ว่า ควรเป็นผู้ใช้บริการใด และใช้ทรัพยากรใดไปเท่าใด ข้อมูลเหล่านี้จะช่วยให้ผู้ดูแลระบบสามารถกำหนดค่าและจัดสรรทรัพยากรให้เหมาะสมกับงานของแต่ละหน่วยงาน

การกำหนดแท็กให้กับทรัพยากรยังช่วยให้สามารถดำเนินกระบวนการปรับขนาด (Rightsizing) ได้อย่างมีประสิทธิภาพ เพื่อลดต้นทุน โดยเมื่อมีการกำหนดขนาดของอินสแตนซ์ของเครื่องเสมือน ได้อย่างเหมาะสมแล้ว ระบบสามารถปรับขยาย (Auto-scale) ให้รองรับการใช้งานที่เพิ่มขึ้นได้โดยอัตโนมัติ ซึ่งจะช่วยเพิ่มประสิทธิภาพได้อีกดับ

2.6 การติดแท็กทรัพยากร (Resource Tagging)

การแท็กหมายถึงการกำหนดป้ายกำกับ (Labels) ให้กับทรัพยากรต่าง ๆ ในระบบคลาวด์ โดยไม่ควร สับสนกับการใช้แท็กเพื่อจุดประสงค์ด้านความปลอดภัย เช่น ป้ายกำกับเพื่อควบคุมการเข้าถึง เพราะแท็ก ในบริบทนี้ถูกใช้เพื่อการกำหนดคุณสมบัติ (Governance) และการจัดการต้นทุน (Cost Management) แท็กจะถูก นำมาใช้ในการสร้างรายงานค่าใช้จ่ายและการใช้งานในเครื่องมืออย่างเช่น AWS Billing and Cost Management Console

The screenshot shows the Microsoft Azure portal interface for managing tags on a resource named 'Test-Server01 | Tags'. At the top, there's a search bar and a user profile for 'SISupport-Developer... SISUPPORT (SISUPPORT.COM)'. Below the search bar, it says 'Dashboard > Test-Server01'. The main area displays the tag configuration for 'Test-Server01' (Virtual machine). It shows two tags: 'BusinessUnit : DevDept' and 'Environment : Dev'. A note at the bottom states: 'Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. Tag names are case insensitive, but tag values are case sensitive.' Another note says: 'Do not enter names or values that could make your resources less secure or that contain personal/sensitive information because tag data will be replicated globally.' There are also 'Delete all' and 'Edit' buttons.

This VM has two resource tags - DevDept and Dev. (Screenshot courtesy of Amazon.)

ภาพที่ 23 Resource Tagging

ตัวอย่างเช่น: คุณอาจกำหนดแท็กให้กับอินสแตนซ์ EC2 จำนวน 12 เครื่องว่าเป็นของฝ่ายพัฒนา ซอฟต์แวร์ (Development Team) ซึ่งข้อมูลการใช้งานของอินสแตนซ์ทั้ง 12 เครื่องนี้จะถูกติดตาม และรวบรวมไว้ในรายงาน เพื่อให้ทีมสามารถประเมินว่าอินสแตนซ์ใดยังจำเป็นอยู่ และอินสแตนซ์ใดอาจไม่ถูก ใช้งานแล้ว (สามารถลบทิ้งเพื่อลดค่าใช้จ่าย) หรืออินสแตนซ์ใดอาจมีสเปคต่ำเกินไปและควรปรับเพิ่มทรัพยากร การใช้งานแท็กเพื่อจัดการต้นทุน

องค์กรสามารถใช้ข้อมูลจากแท็กเพื่อดำเนินการ การเรียกเก็บเงินคืน (Chargeback) ไปยัง หน่วยธุรกิจที่ใช้งานทรัพยากร เช่น แผนกไอทีเรียกเก็บค่าใช้บริการ EC2 จากทีมพัฒนาเพื่อนำไปครอบคลุม ต้นทุนโดยรวมของการใช้คลาวด์

ในกรณีของ Showback จะใช้รายงานการใช้งานทรัพยากรโดยไม่คิดค่าใช้จ่ายจริงกับหน่วยธุรกิจ เช่น ทีมพัฒนาได้รับรายงานประจำปีที่แสดงรายละเอียดการใช้งานทรัพยากร ซึ่งช่วยให้เข้าใจการบริหาร ต้นทุนในอนาคตได้ดียิ่งขึ้น

2.7 การปรับขนาดให้เหมาะสม (Rightsizing)

บริการคลาวด์ช่วยให้การปรับแต่งทรัพยากรคอมพิวต์ให้เหมาะสมกับปริมาณงาน (Workload) เป็นเรื่องง่าย ซึ่งในอดีตเมื่อใช้เซิร์ฟเวอร์แบบตั้งเดิม การจัดสรรทรัพยากร เช่น หน่วยประมวลผลกลาง (CPU), หน่วยความจำ (Memory), ที่เก็บข้อมูล (Storage) และเครือข่าย (Network) มักเป็นแบบตายตัว ผู้ดูแลระบบ จึงมักเลือกสเปคสูงเกินจริงเพื่อรับการเติบโตในอนาคต ส่งผลให้เกิดการใช้ทรัพยากรไม่เต็มประสิทธิภาพ และสิ้นเปลืองงบประมาณโดยไม่จำเป็น

Viewing 207 of 695 available instances						
Instance name	On-Demand hourly rate	vCPU	Memory	Storage	Network performance	
t2.nano	\$0.0058	1	0.5 GiB	EBS Only	Low	
t2.micro	\$0.0116	1	1 GiB	EBS Only	Low to Moderate	
t2.small	\$0.023	1	2 GiB	EBS Only	Low to Moderate	
t2.medium	\$0.0464	2	4 GiB	EBS Only	Low to Moderate	
t2.large	\$0.0928	2	8 GiB	EBS Only	Low to Moderate	
t2.xlarge	\$0.1856	4	16 GiB	EBS Only	Moderate	
t2.2xlarge	\$0.3712	8	32 GiB	EBS Only	Moderate	

AWS general purpose on-demand instances. Note the price versus performance information.
(Screenshot courtesy of Amazon.)

ภาพที่ 24 Rightsizing

แม้การใช้ระบบเวอร์ชวลไลเซชัน (Virtualization) ภายในองค์กรจะช่วยให้สามารถเพิ่มหรือลดทรัพยากรได้สะดวกขึ้น แต่การปรับขนาดในคลาวด์ทำได้ง่ายยิ่งกว่า ผู้ดูแลสามารถเพิ่มหรือลดทรัพยากรของเครื่องเสมือนได้ตามความต้องการจริง

ความหมายของ Rightsizing:

Rightsizing คือการปรับความสามารถของเครื่องเสมือน (VM) ให้เหมาะสมกับปริมาณงาน โดยมีเป้าหมายเพื่อไม่จัดสรรทรัพยากรมากเกินไป (ซึ่งสิ้นเปลือง) หรือ น้อยเกินไป (ซึ่งกระทบต่อประสิทธิภาพ) แม้ระบบคลาวด์จะสามารถปรับขยาย (Scaling) ได้ตามความต้องการ แต่การ Rightsizing เน้นการปรับแต่งให้เหมาะสมที่สุดก่อนการขยาย เพื่อควบคุมต้นทุนได้อย่างมีประสิทธิภาพ

ตัวอย่าง: การ Rightsizing ฐานข้อมูล

บริษัทแห่งหนึ่งย้ายระบบฐานข้อมูลจากเซิร์ฟเวอร์จริงในองค์กร (On-Premises) ไปยังคลาวด์โดยไม่ได้เก็บบันทึกประวัติภาพของเซิร์ฟเวอร์เดิม จึงเลือกกำหนดค่าทรัพยากร VM เท่ากับเครื่องเก่าเพื่อความปลอดภัย

ต่อมา เมื่อใช้งานไประยะหนึ่ง ผู้ดูแลระบบพบว่าการใช้ CPU และ RAM ต่ำกว่าความสามารถของเครื่องอย่างมาก จึงแนะนำให้ลดสเปกของ VM ลงเพื่อ省ท้อนความต้องการใช้งานจริง ซึ่งช่วยลดค่าใช้จ่ายของบริษัทได้ทันที

ตัวอย่าง: กระบวนการ Rightsizing ในองค์กรขนาดเล็ก

บริษัทขนาดเล็กแห่งหนึ่งย้ายระบบไอทีเกือบทั้งหมดไปยังคลาวด์ โดยมีลักษณะของงานและความต้องการของลูกค้าเปลี่ยนแปลงอยู่ตลอดทั้งปี ทีมไอทีจึงทราบว่า Rightsizing เป็นกระบวนการที่ต้องทำต่อเนื่อง และได้มอบหมายให้ผู้ดูแลระบบคลาวด์ทำการวิเคราะห์ข้อมูลการใช้งานและต้นทุนเป็นรายเดือน เพื่อให้บริการคลาวด์มีประสิทธิภาพและประหยัดต้นทุนที่สุด

2.8 การแก้ไขปัญหาการเรียกเก็บเงิน (TROUBLESHOOT BILLING)

ปัญหาการเรียกเก็บเงินในระบบคลาวด์อาจเกิดจากข้อกำหนดของผู้ให้บริการคลาวด์ การจัดการทรัพยากรที่ไม่เหมาะสม หรือความเข้าใจคลาดเคลื่อนเกี่ยวกับการใช้งานทรัพยากรคลาวด์ ดังนั้น การตรวจสอบแต่ละกรณีควรคำนึงถึงประเด็นต่าง ๆ ต่อไปนี้

หนึ่งในแนวทางป้องกันปัญหาค่าใช้จ่ายไม่คาดคิด คือ การตั้งค่าการเฝ้าระวัง (Monitoring) และการแจ้งเตือน (Alarm) เกี่ยวกับการใช้งบประมาณและทรัพยากร ตัวอย่างเช่น ระบบ AWS CloudWatch สามารถตั้ง Billing Alarm เพื่อแจ้งเตือนเมื่อใช้เกินงบประมาณที่กำหนด

แนวทางการตรวจสอบเมื่อเกิดปัญหา

- หากปัญหาคือ ไม่สามารถรับบริการสนับสนุนด้านเทคนิคหรือบริการไม่เพียงพอ:
 - ตรวจสอบรายละเอียดในข้อตกลงระดับการให้บริการ (Service Level Agreement - SLA)
 - ตรวจสอบรูปแบบการอนุญาตให้ใช้สิทธิ์ (Licensing) โดยเฉพาะในกรณีของบริการประเภท SaaS
- หากปัญหาคือ บริการที่ได้รับมีความพร้อมใช้งานต่ำกว่าที่ต้องการ:
 - ตรวจสอบ SLA ของบริการ
 - ตรวจสอบเงื่อนไขของ Licensing
 - ตรวจสอบข้อจำกัดในการเรียกใช้งานผ่าน API (Application Programming Interface)
- หากปัญหาคือ การอนุญาตสิทธิ์การใช้งาน (Licensing) ผิดพลาด:
 - ตรวจสอบเดชบอร์ดของระบบบัญชี/การเงินของผู้ให้บริการคลาวด์
 - ตรวจสอบข้อตกลงสิทธิ์การใช้งานร่วมกับฝ่ายบัญชี (อาจไม่ได้อยู่ในความรับผิดชอบของฝ่าย IT)
- หากปัญหาคือ ค่าใช้จ่ายสูงหรือการเรียกเก็บเงินผิดปกติ:

- ตรวจสอบแอปพลิเคชันหรืออินสแตนซ์ที่ทำงานเป็นเวลานานโดยไม่จำเป็น
- ตรวจสอบการเข้าถึงทรัพยากรคลาวด์ที่ไม่มีการควบคุม
- ตรวจสอบอินสแตนซ์ที่ไม่ได้ใช้งานแต่ยังคงมีการคิดค่าใช้จ่าย
- ตรวจสอบเดชบอร์ดการเรียกเก็บเงินของผู้ให้บริการ
- ตรวจสอบข้อตกลงด้านการอนุญาตสิทธิ์กับฝ่ายบัญชี

สรุปการแก้ไขปัญหาในการปรับใช้ระบบและการควบคุมต้นทุน (Summary: Troubleshooting Deployment Issues and Cost)

บทเรียนนี้เน้นให้ความเข้าใจเกี่ยวกับปัญหาที่อาจเกิดขึ้นในการจัดเตรียมทรัพยากร (Provisioning) และการย้ายระบบ (Migration) รวมถึงแนะนำวิธีตรวจสอบสาเหตุของปัญหาที่เป็นไปได้ โดยเน้นความสัมพันธ์ระหว่างอาการของปัญหาและทรัพยากรที่ควรตรวจสอบเพื่อวิเคราะห์อย่างเป็นระบบ

ในด้านการบริหารจัดการต้นทุน การใช้บริการคลาวด์ได้เปลี่ยนรูปแบบจากการลงทุนในハードแวร์固定 (CapEx) ไปสู่การชำระเงินแบบสมัครสมาชิก (Subscription) ซึ่งคิดค่าบริการตามการใช้งานจริง (OpEx) การบริหารค่าใช้จ่ายอย่างมีประสิทธิภาพจึงจำเป็นต้องมีความเข้าใจเกี่ยวกับ:

- ไม่เดลการเรียกเก็บเงินของบริการคลาวด์ เช่น แบบจองล่วงหน้า (Reserved), แบบจ่ายตามการใช้งาน (Pay-as-you-go), หรือแบบ Spot Instances
- การปรับแต่งการใช้ทรัพยากร (Resource Optimization) เช่น การวัดการใช้ (Metering), การติดป้าย (Tagging) และการปรับขนาดให้เหมาะสม (Rightsizing) อย่างต่อเนื่องเพื่อควบคุมต้นทุนและรักษาประสิทธิภาพ

การเข้าใจโครงสร้างเหล่านี้จะช่วยให้สามารถวางแผน ติดตาม และปรับปรุงการใช้งานระบบคลาวด์ให้คุ้มค่าที่สุด ทั้งในด้านการทำงานและการเงิน

บทที่ 6

การใช้เทคโนโลยีเวอร์ชวลไลเซชันและฐานข้อมูล (Using Virtualization and Databases)

บทนำบทเรียน (Module Introduction)

Infrastructure as a Service (IaaS) เป็นหนึ่งในรูปแบบบริการคลาวด์ที่สำคัญ โดยมีฐานะอยู่บนเทคโนโลยีเวอร์ชูลไลเซชัน (Virtualization) ซึ่งเป็นการจำลองฮาร์ดแวร์ให้สามารถรันระบบปฏิบัติการหลายชุดบนเครื่องเดียวกันได้ ปัจจุบันยังมีเทคโนโลยีใหม่ที่เรียกว่า คอนเทนเนอร์ไรเซชัน (Containerization) ซึ่งพัฒนาต่อยอดจากเวอร์ชูลไลเซชันและหมายความว่ามีความสัมภาระต่ำและแยกส่วน (Loosely Coupled) และกระบวนการ DevOps

บริการฐานข้อมูลที่ใช้ส่วนใหญ่บนคลาวด์เป็นหนึ่งในบริการหลักของผู้ให้บริการคลาวด์ โดยผู้ใช้สามารถเลือกใช้ฐานข้อมูลแบบเชิงสัมพันธ์ (Relational – SQL) หรือไม่เชิงสัมพันธ์ (Non-relational – NoSQL) ได้ทั้งในรูปแบบที่จัดการด้วยตนเองหรือใช้บริการฐานข้อมูลแบบมีการจัดการโดยผู้ให้บริการ (Managed Database Services)

วัตถุประสงค์ของบทเรียน (Module Objectives)

ในบทเรียนนี้ คุณจะได้เรียนรู้เกี่ยวกับ:

- ความเข้าใจเกี่ยวกับแนวคิดของเวอร์ชูลไลเซชัน
- ความเข้าใจเกี่ยวกับแนวคิดของคอนเทนเนอร์ไรเซชัน
- ความแตกต่างระหว่างฐานข้อมูลเชิงสัมพันธ์และไม่เชิงสัมพันธ์

หน่วยที่ 1 แนวคิดเกี่ยวกับเวอร์ชูลไลเซชัน (Virtualization Concepts)

เครื่องเวอร์ชูล (Virtual Machines หรือ VMs) อาศัยซอฟต์แวร์ตัวกลางที่เรียกว่า Hypervisor เพื่อทำหน้าที่จำลองฮาร์ดแวร์เสมือน (Virtualized Hardware) ในขณะที่ คอนเทนเนอร์ (Containers) จะใช้แนวทางของการจำลองระบบปฏิบัติการ (Virtualized Operating System) แทน เครื่องเวอร์ชูลเป็นองค์ประกอบสำคัญของการปรับใช้ระบบ IT ในปัจจุบัน เนื่องจากสามารถ:

- ใช้ทรัพยากรได้อย่างมีประสิทธิภาพ
- ขยายระบบได้ง่าย
- รองรับการทำงานแบบ High Availability

ตัวอย่างการใช้งานที่พบบ่อยของ Virtual Machines ได้แก่:

- เซิร์ฟเวอร์ในระบบงานจริง (Production Servers)
- เซิร์ฟเวอร์สำหรับการพัฒนา (Development Servers)

- สภาพแวดล้อมทดสอบ (Test Environments)
- ห้องทดลองสำหรับการเรียนรู้ (Learning Labs)

ระบบคลาวด์ส่วนใหญ่ถูกสร้างขึ้นบนเทคโนโลยีเวอร์ชัลไลเซชัน โดยเปิดให้ผู้ใช้สามารถรัน VM บนศูนย์ข้อมูลของผู้ให้บริการคลาวด์ แทนที่จะใช้ฮาร์ดแวร์ขององค์กรเอง ซึ่งช่วยลดภาระในการดูแลรักษา อุปกรณ์ขององค์กรลงได้อย่างมาก

1.1 แนวคิดเกี่ยวกับ Virtualization (การจำลองระบบ)

การจำลองระบบเป็นพื้นฐานสำคัญของบริการคลาวด์ (Cloud Services) โดยมีจุดประสงค์เพื่อแบ่งทรัพยากรของฮาร์ดแวร์จริง (Physical Hardware) ให้สามารถใช้งานร่วมกันได้อย่างมีประสิทธิภาพผ่านเครื่องเสมือน (Virtual Machines – VMs)

หลักการของ Virtualization:

- ผู้ดูแลระบบ (Administrator) จะแบ่งทรัพยากรของเซิร์ฟเวอร์จริง เช่น CPU, หน่วยความจำ, และพื้นที่จัดเก็บข้อมูล เพื่อสร้าง VM หลายเครื่อง
- แต่ละ VM จะมีระบบปฏิบัติการของตัวเอง เช่น Windows Server หรือ Red Hat Enterprise Linux
- ระบบปฏิบัติการภายใน VM สามารถเข้าสู่บริการมาตรฐาน เช่น Web Server, ฐานข้อมูล, DNS, หรือ DHCP

หน้าที่ของผู้ดูแลระบบ (Sysadmin):

- ยังคงต้องติดตั้ง ปรับแต่ง ดูแล อัปเดต และรักษาความปลอดภัยของระบบปฏิบัติการและแอปพลิเคชัน ในแต่ละ VM
- หากไม่ใช่บริการ IaaS องค์กรยังต้องรับผิดชอบเรื่องฮาร์ดแวร์และความปลอดภัยทางกายภาพเอง

Hypervisor คืออะไร?

Hypervisor เป็นซอฟต์แวร์ที่ทำหน้าที่จำลองทรัพยากรหาร์ดแวร์ให้ VM ใช้งาน และควบคุมการเข้าถึงฮาร์ดแวร์ของแต่ละ VM

ความสามารถของ Hypervisor:

- จัดการการแบ่งทรัพยากรระหว่าง VM ต่าง ๆ
- รองรับการสร้าง Snapshot เพื่อย้อนสถานะของ VM ได้
- รองรับการโคลน (Replication) เพื่อความพร้อมใช้งานสูง (High Availability)
- สามารถขยายหรือลดขนาดทรัพยากรของ VM ตามภาระงาน (Workload) ได้

ข้อดีของการจำลองระบบในคลาวด์:

- ผู้ให้บริการคลาวด์จะเป็นผู้ดูแลฮาร์ดแวร์ ทำให้องค์กรไม่ต้องลงทุนด้านอุปกรณ์
- ลดภาระงานด้านการบำรุงรักษาเครื่องเซิร์ฟเวอร์
- รองรับการขยายตัวของระบบได้ง่ายและคุ้มค่า
- ลดความไม่แน่นอนในการวางแผนทรัพยากรลงหน้า

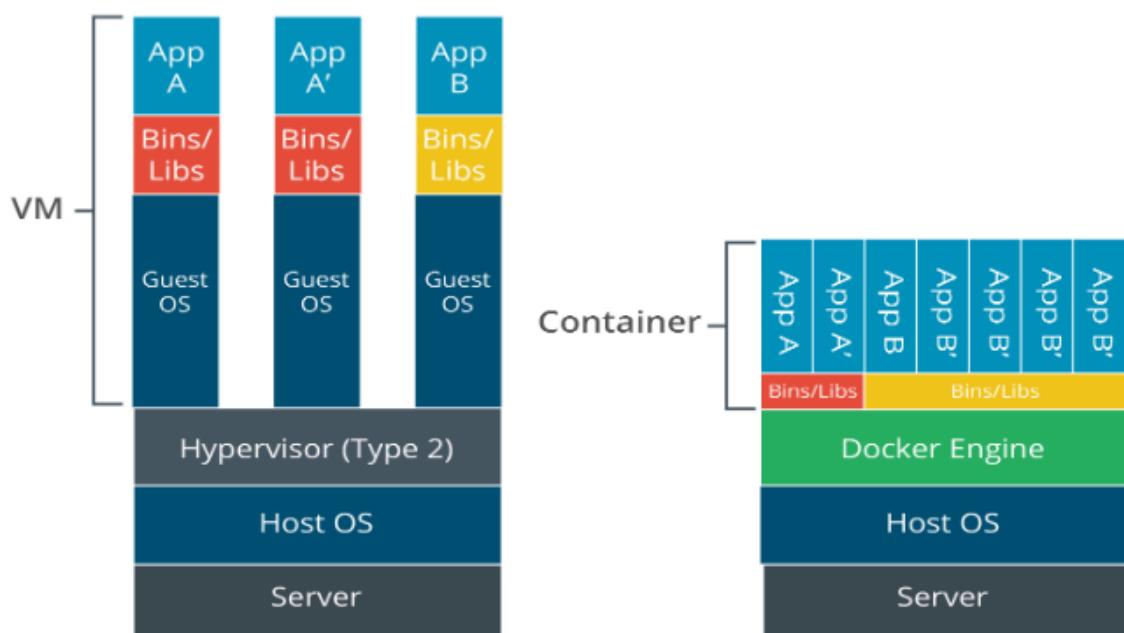
IaaS (Infrastructure as a Service):

เป็นโมเดลการให้บริการคลาวด์แบบพื้นฐานที่สำคัญที่สุด ซึ่งให้สิทธิผู้ดูแลระบบสามารถเข้าถึงและจัดการ VM ได้โดยตรง โดยไม่ต้องดูแลฮาร์ดแวร์เบื้องหลัง ทำให้องค์กรสามารถควบคุมและปรับแต่งระบบได้เต็มที่ พร้อมประเมินจากความยืดหยุ่นและความพร้อมใช้งานสูงของคลาวด์

1.2 การจำลองระบบแบบสแตนด์อโลน (Stand-alone Virtualization)

การจำลองระบบแบบสแตนด์อโลนหมายถึงการกำหนดค่าของเครื่องเสมือน (VM) ให้เหมือนกับเครื่องเซิร์ฟเวอร์จริง (Physical System) โดยผู้ดูแลระบบจะต้องดำเนินการจัดสรรทรัพยากร เช่น หน่วยความจำ (Memory), ที่จัดเก็บข้อมูล (Storage), หน่วยประมวลผล (CPU), และเครือข่าย (Networking) ให้กับ VM เพื่อให้สามารถเข้ามายังเครื่องที่ต้องการได้ เช่นเดียวกับเซิร์ฟเวอร์ Linux หรือเวิร์กสเตชันทั่วไป

Container vs. VMs



Comparing virtual machines and containers. Note the virtual machine operating systems and hypervisor layer.

ภาพที่ 25 Stand-Alone Virtualization

การจัดการ VM ผ่าน Virtualization Console:

ผู้ดูแลระบบสามารถใช้คอนโซลเพื่อ:

- สร้าง (Create), จัดการ (Manage), และลบ (Remove) VM
- ตั้งค่าข้อ, CPU, Memory, Storage, และเครือข่าย
- ตรวจสอบประสิทธิภาพและเวลาใช้งาน (Uptime)

- ปรับขนาด VM ให้สอดคล้องกับโหลดของงาน

ตัวอย่าง: ค่อนขอลของ Microsoft Azure สำหรับการจัดการ VM จะแสดงข้อมูลสำคัญ เช่น สถานะ, ขนาด, ที่อยู่ IP, ระบบปฏิบัติการ, กลุ่มทรัพยากร และเครือข่ายย่อย

วงจรการจัดการของ VM (VM Management Lifecycle):

- Create:** สร้าง VM โดยอิงจากความต้องการใช้งานและซอฟต์แวร์ที่เกี่ยวข้อง
- Maintain:** ดูแลและอัปเดตระบบปฏิบัติการรวมถึงแอปพลิเคชัน
- Monitor:** ตรวจสอบค่าใช้จ่ายและประสิทธิภาพของ VM อย่างต่อเนื่อง
- Delete:** ลบ VM เมื่อไม่จำเป็นต้องใช้งานอีกต่อไป

1.3 การปรับใช้เครื่องเสมือนบนคลาวด์ (Deploy Virtual Machines in the Cloud)

การสร้างและปรับใช้เครื่องเสมือน (VM) บนระบบคลาวด์นั้นมีขั้นตอนที่คล้ายคลึงกับการใช้ซอฟต์แวร์จำลองระบบแบบโลคลัล (Local Virtualization Software) โดยผู้ดูแลระบบต้องกำหนดชื่อเครื่อง, ตั้งค่าฮาร์ดแวร์ และเลือกค่าที่เกี่ยวข้องกับคลาวด์ เช่น เขต (Region), วิธีการยืนยันตัวตน (Authentication), ตัวระบุการคิดค่าบริการ (Metering Identifier) และตัวเลือกเฉพาะอื่น ๆ

ข้อพิจารณา ก่อนปรับใช้ VM:

ควรพิจารณาความต้องการด้านการประมวลผล, ที่จัดเก็บข้อมูล, ประสิทธิภาพ และงบประมาณ เพื่อให้การกำหนดค่าทรัพยากรมีความเหมาะสม ไม่สิ้นเปลืองเกินความจำเป็น

ประเภทของ VM ตามวัตถุประสงค์:

- General purpose (ทั่วไป)
- Compute optimized (เน้นการประมวลผล)
- Memory optimized (เน้นหน่วยความจำ)
- Storage optimized (เน้นการจัดเก็บข้อมูล)

ตัวเลือกประเภทของที่จัดเก็บข้อมูล:

- Premium SSD (Solid-state Drive ความเร็วสูง)
- Standard SSD
- Standard HDD (Hard Disk Drive มาตรฐาน)

ขั้นตอนการสร้าง VM บนระบบคลาวด์ (เช่น Microsoft Azure):

- เปิดหน้า Virtual Machines Console และเลือก Create เพื่อสร้างเครื่องใหม่
- กำหนด ชื่อ VM, เลือก ภูมิภาค (Region), และ Availability Zone
- เลือก ภาพระบบปฏิบัติการ (OS Image) เช่น Ubuntu Linux หรือ Windows Server 2022
- เลือก ขนาดของ VM (เช่น จำนวน vCPU และ RAM)

5. ตั้งค่าบัญชีผู้ดูแลระบบ เช่น การยืนยันตัวตนแบบ SSH Key
 6. เลือกประเภทที่จัดเก็บข้อมูล (SSD หรือ HDD)
 7. ตั้งค่าเครือข่าย เช่น เครือข่ายเสมือน, IP Address และการเปิดพอร์ต
 8. กำหนด Tags สำหรับการตรวจสอบการใช้งานและการเรียกเก็บเงิน
 9. คลิก Create เพื่อปรับใช้เครื่องเสมือน

ໝາຍເຫດ:

- ระบบจะแสดงประมาณการราคา (Price Estimate) สำหรับการทำหนดค่าที่เลือก
 - คุณสามารถดาวน์โหลด Template เพื่อใช้สร้าง VM แบบเดียวกันในอนาคตแบบอัตโนมัติได้

1.4 แม่แบบไฟล์การกำหนดค่าเครื่องเสมือน (Virtual Machine Configuration File Templates)

ไฟล์การกำหนดค่า (Configuration Files) คือไฟล์ที่ใช้กำหนดพารามิเตอร์ต่าง ๆ ของเครื่องเสมือน (VM) โดยระบุรายละเอียดเกี่ยวกับการจัดสรรทรัพยากร่วมกัน เช่น

- ข้อมูลหน่วยประมวลผลกลาง (CPU)
 - ปริมาณหน่วยความจำ (RAM)
 - ตัวเลือกเครือข่าย (Network Options)
 - ประเภทและขนาดของที่จัดเก็บข้อมูล (Storage)

```
1} template.json (deleted) X
C:\Users\Administrator\AppData\Local\Temp>1>Temp1_ExportedTemplate-515Support-Devlod38619419 (1):
24
25
26
27
28
29
30
31
32     },
33     "properties": {
34         "hardwareProfile": {
35             "vmSize": "Standard_D2s_v3"
36         },
37         "additionalCapabilities": {
38             "hibernationEnabled": false
39         },
40         "storageProfile": {
41             "imageReference": {
42                 "publisher": "canonical",
43                 "offer": "0001-com-ubuntu-server-focal",
44                 "sku": "20_04-lts-gen2",
45                 "version": "latest"
46             },
47             "osDisk": {
48                 "osType": "Linux",
49                 "name": "[concat(parameters('virtualMachines_Test_Server01_r",
50                 "createOption": "FromImage",
51                 "caching": "ReadWrite",
52                 "managedDisk": {
53                     "storageAccountType": "StandardSSD_LRS",
54                     "id": "[parameters('disks_Test_Server01_disk1_1ccdc9a5f8
55                 },
56                 "deleteOption": "Delete",
57                 "diskSizeGB": 30
58             }
59         }
60     }
61 }
```

Partial sample of a VM configuration file. Note the VM size, operating system (OS) information, and disk configurations. (Screenshot courtesy of Amazon.)

ภาพที่ 26 Virtual Machine Configuration File Templates

ประโยชน์ของไฟล์กำหนดค่า:

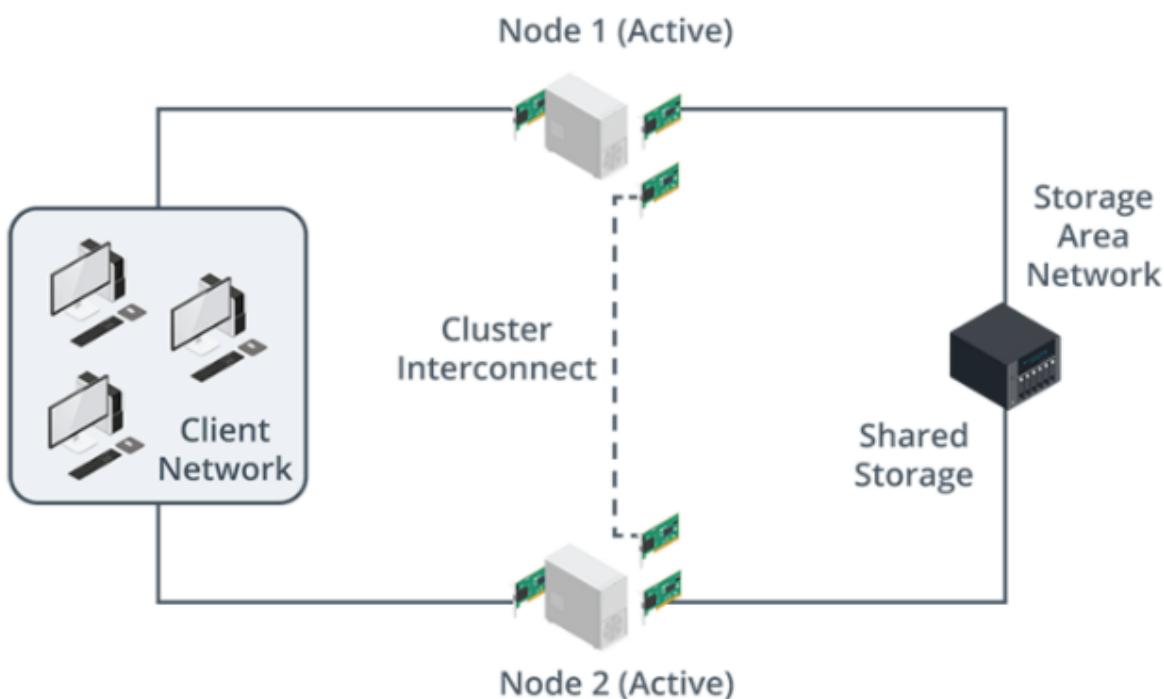
ไฟล์เหล่านี้สามารถใช้เป็น แม่แบบ (Template) เพื่อสร้าง VM ที่มีการตั้งค่าล่วงหน้าเหมือนกันได้ ซึ่งหมายความว่า การขยายระบบ (Scalability) ในสภาพแวดล้อมคลาวด์

ข้อควรพิจารณา:

VM ในขั้นตอนแรกเป็นอุปกรณ์เดี่ยว (Stand-alone Virtual Devices) เช่น การซื้อเซิร์ฟเวอร์เครื่องเดียวเพื่อให้บริการ เว็บไซต์ แต่ระบบแบบสแตนด์อโลนไม่ว่าจะเป็นระบบภายในภาพหรือเสมือนล้วนเป็น จุดที่เกิดความล้มเหลวได้ง่าย (Single Point of Failure) หากไม่มีการวางแผนด้านความทนทานหรือความพร้อมใช้งานสูง (High Availability)

1.5 การจำลองระบบแบบคลัสเตอร์ (Clustering Virtualization)

การจัดระบบแบบคลัสเตอร์ (Cluster) คือการรวมกลุ่มคอมพิวเตอร์หลายเครื่องให้ทำงานร่วมกัน เป็นหนึ่งหน่วย โดยแต่ละเครื่องในคลัสเตอร์เรียกว่า “โหนด (Node)” จุดประสงค์หลักคือเพื่อสร้างระบบ ที่มี ความทนทานต่อความล้มเหลว (Fault Tolerance) และเพิ่ม ประสิทธิภาพ (Performance) โดยกระจายภาระงาน (Workload) ให้แต่ละโหนดในคลัสเตอร์รับผิดชอบร่วมกัน



A server cluster. (Images © 123RF.com)

ภาพที่ 27 Clustering Virtualization

1.5.1 คลัสเตอร์เสมือน (Virtual Machine Clusters in the Cloud)

ผู้ให้บริการคลาวด์ (Cloud Service Providers – CSPs) เช่น AWS, Azure, และ GCP มีบริการคลัสเตอร์ที่พัฒนามาจากแนวคิดคลัสเตอร์แบบดั้งเดิมในระบบกายภาพ โดยปรับปรุงด้วยความสามารถของเวอร์ชวลайเซชัน (Virtualization)

- **AWS ECS Clusters:** ประกอบด้วย Tasks และ Services ที่สามารถรันบน VM, Container, หรือแม้กระทั่ง Serverless
- **Communication:** คลัสเตอร์ใช้ Virtual Private Cloud (VPC) เป็นเครือข่ายส่วนตัวในการสื่อสารระหว่างโนํา
- **การจัดการความสามารถ (Capacity Provider):** ควบคุมการขยายตัว (Scalability) ของคลัสเตอร์ได้

1.5.2 การตั้งค่าความสัมพันธ์ของโนํา (Host Affinity vs Anti-affinity)

- **Affinity:** บังคับให้ VM หลายตัวในคลัสเตอร์รันอยู่บน ไอดีเดียวกัน เพื่อเพิ่มประสิทธิภาพในการสื่อสาร (เช่น ใช้ host bus)
- **Anti-affinity:** บังคับให้ VM กระจายอยู่บน ไอดีต่างกัน เพื่อลดความเสี่ยงหากไอดีเครื่องหนึ่งล่ม จะยังมี VM ตัวอื่น ๆ ที่ทำงานได้อยู่

หมายเหตุ: การเลือกใช้ Affinity หรือ Anti-affinity ต้องพิจารณาร่วมกับเป้าหมายด้าน High Availability (HA) และ Performance

1.5.3 ฮาร์ดแวร์พาสทรู (Hardware Pass-Through)

โดยทั่วไป VM จะสื่อสารกับอุปกรณ์ผ่านชั้น Hypervisor แต่ในงานที่ต้องการประสิทธิภาพสูง เช่น:

- การประมวลผลกราฟิก (GPU-intensive)
- การเข้าถึงสื่อบันทึกข้อมูลโดยตรง (Storage access)
- การจัดการโหลดเครือข่าย (Network Load Balancing)

ผู้ดูแลระบบ (Admin) อาจต้องตั้งค่าให้ ข้าม hypervisor และเชื่อมต่อฮาร์ดแวร์โดยตรง เพื่อให้การทำงานมีความเร็วและประสิทธิภาพสูงขึ้น

1.6 การโคลนเครื่องเสมือน (Virtual Machine Cloning)

การโคลนเครื่องเสมือน (VM) เป็นกระบวนการที่ผู้ดูแลระบบสามารถสร้างสำเนาเครื่องเสมือนที่มีการกำหนดค่าและข้อมูลเหมือนกับต้นฉบับ ณ ขณะเวลาที่ทำการโคลน การโคลนนี้ช่วยให้สามารถนำ VM ที่มีการติดตั้งและตั้งค่าเรียบร้อยแล้วมาใช้งานได้อย่างรวดเร็ว โดยไม่ต้องเริ่มต้นจากศูนย์

ประโยชน์ของการโคลน VM

- การปรับใช้งานได้อย่างรวดเร็ว (Rapid Deployment): เหมาะสำหรับสถานการณ์ที่ต้องเปิดใช้งานหลาย VM ที่มีลักษณะคล้ายกัน
- การสร้างระบบสำรอง (Redundancy): สามารถใช้สำเนาเพื่อการกู้คืนข้อมูลหรือลดภัยด้วยเวลา
- การตั้งค่าระบบคลัสเตอร์ (Clustering): โคลน VM เพื่อให้ทำงานร่วมกันในระบบคลัสเตอร์
- การสำรองข้อมูล (Backup): ใช้ VM โคลนเป็นสำรองในกรณีฉุกเฉิน

ความแตกต่าง: Cloning vs Templates

- Cloning: เป็นการสร้างสำเนาของ VM พร้อมข้อมูลและการตั้งค่าทั้งหมด ณ เวลาหนึ่ง
- Template Deployment: ใช้แม่แบบ (Template) ที่สร้างไว้ล่วงหน้า เพื่อสร้าง VM ใหม่ที่มีการกำหนดค่ามาตรฐานเบื้องต้น

ตัวเลือกจากผู้ให้บริการคลาวด์

ผู้ให้บริการระบบคลาวด์ เช่น AWS, Microsoft Azure และ Google Cloud Platform (GCP) มีพิจารณาสำหรับการโคลน VM ได้แก่:

- การสร้าง Image หรือ Snapshot
- การใช้ Templates หรือ Machine Images
- การเปิดใช้งาน VM จาก Image เดิมหลายเครื่องพร้อมกัน

1.7 ประเภทของเครือข่าย (Network Types)

ในสภาพแวดล้อมระบบคลาวด์และการจำลองเสมือน (Virtualization) การจัดการเครือข่ายมีความซับซ้อนมากกว่าระบบเครือข่ายแบบดั้งเดิม (On-premises) โดยแบ่งออกเป็นประเภทต่าง ๆ ดังนี้:

1.7.1 เครือข่ายมาตรฐาน (Standard Networks)

- ประกอบด้วยสายสื่อสาร, สวิตช์ Layer 2, เรเเตอร์ Layer 3 และการ์ดเครือข่าย (NIC)
- สามารถจัดการทรัพย์ฟิก แบ่งส่วนเครือข่าย (Segmentation) และกรองข้อมูล
- มีข้อจำกัดด้านระยะทาง, ชนิดของสายสื่อสาร, และปัญหาทางกายภาพ
- เหมาะสมสำหรับ LAN ภายในองค์กร

1.7.2 เครือข่ายซ้อนทับ (Overlay Networks)

- เป็นเครือข่ายเสมือนที่แยกการจัดการจากโครงสร้างจริง เช่น สาย, สวิตช์, เรเเตอร์
- ช่วยให้สามารถสร้างเครือข่ายตรรกะ (logical networks) ได้โดยไม่ขึ้นกับเครือข่ายจริง
- ใช้ได้ในระบบคอนเทนเนอร์และคลัสเตอร์

ข้อดี:

- ลดภาระในการจัดการ
- ขยายระบบได้ง่าย (Scalable)
- เพิ่มความปลอดภัยผ่านการแยกส่วน
- ยืดหยุ่นต่อการปรับแต่งและประสิทธิภาพสูง

ข้อเสีย:

- ซับซ้อน อาจทำให้เกิดข้อผิดพลาดในการกำหนดค่า
- แก้ไขปัญหาได้ยากกว่า

โปรโตคอลที่ใช้:

- VXLAN (Virtual Extensible LAN)
- GRE (Generic Routing Encapsulation)
- Network Virtualization Overlays

หมายเหตุ: Overlay Network ต่างจาก SDN (Software-Defined Networking) ซึ่งแยก Control Plane ออกจาก Data Plane

1.7.3 เครือข่ายของเครื่องเสมือน (Virtual Machine Networks)

- ให้การเชื่อมต่อกับ VM ผ่าน Virtual Switch
- VM สามารถทำงานได้เหมือนเซิร์ฟเวอร์จริง เช่น Web Server, DHCP Server, Git Server
- ระบบเสมือนสามารถมีหลาย Virtual Switch เพื่อแยกกลุ่ม VM ได้

1.7.4 ประเภทของเครือข่ายในเครื่องเสมือน (VM Network Types)

1. External

- เข้าถึงเครือข่ายจริงผ่าน NIC ของ Host ได้
- เหมาะสมสำหรับ VM ที่ต้องเชื่อมต่อกับภายนอก เช่น เว็บเซิร์ฟเวอร์

2. Internal

- เชื่อมต่อได้เฉพาะกับ Host และ VM อื่น ๆ บน Host เดียวเท่านั้น
- ไม่มีการเชื่อมต่อภายนอก

3. Private

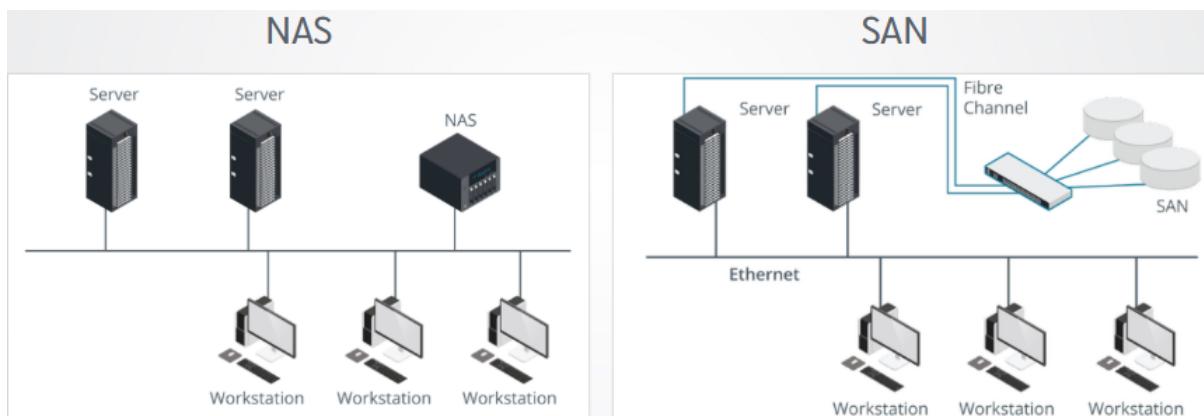
- เชื่อมต่อได้เฉพาะระหว่าง VM ด้วยกันเท่านั้น
- ไม่มีการเข้าถึง Host หรือเครือข่ายอื่น

เครือข่ายเหล่านี้สามารถปรับเปลี่ยนได้ภายหลังจากการสร้าง VM โดยเลือก switch และ NIC ที่เหมาะสม

1.8 ประเภทของระบบจัดเก็บข้อมูล (Types of Storage)

เซิร์ฟเวอร์ทั้งที่อยู่ภายในองค์กรและในระบบคลาวด์สามารถใช้งานโครงสร้างพื้นฐานด้านการจัดเก็บข้อมูลได้หลากหลายรูปแบบ ไม่ว่าจะเป็นในลักษณะของเครื่องจริงหรือเครื่องเสมือน ทั้งในแบบเดียว (Stand-alone) หรือคลัสเตอร์ (Cluster) แม้ว่าเครื่องคอมพิวเตอร์จำนวนมากจะใช้โครงสร้างภายใน แต่การจัดเก็บข้อมูลผ่านเครือข่ายก็เป็นอีกทางเลือกหนึ่งที่ตอบโจทย์ประสิทธิภาพที่สูงกว่า

บทเรียนนี้จะอธิบายรูปแบบของการจัดเก็บข้อมูล ได้แก่ การจัดเก็บในเครื่อง, การจัดเก็บผ่านอุปกรณ์ NAS และเครือข่าย SAN



ภาพที่ 28 Types of Storage

1.8.1 Local Storage (การจัดเก็บข้อมูลภายในเครื่อง)

คอมพิวเตอร์ตั้งโต๊ะทั่วไปและเซิร์ฟเวอร์หลายเครื่องยังคงใช้ Direct-Attached Storage (DAS) ซึ่งโดยมากเป็น Solid-State Drives (SSD) หรือ Hard Disk Drives (HDD) การจัดเก็บข้อมูลแบบนี้ติดตั้งและกำหนดค่าได้ง่าย แต่การขยายระบบเป็นไปได้ยาก และไม่สามารถแบ่งปันกับบริการอื่นได้โดยตรง

แม้จะติดตั้งง่ายและเหมาะสมกับระบบขนาดเล็ก แต่หลายกรณียังจำเป็นต้องใช้วิธีการเข้าถึงพื้นที่จัดเก็บที่มีความยืดหยุ่นและประสิทธิภาพมากขึ้น เช่น ในเซิร์ฟเวอร์ท้องถิ่น, คลัสเตอร์, และระบบคลาวด์

1.8.2 Network Attached Storage (NAS)

อุปกรณ์ NAS คือกลุ่มของฮาร์ดไดร์ฟที่มีการ์ดเครือข่าย (NIC) ในตัว ซึ่งสามารถใช้เพิ่มพื้นที่จัดเก็บข้อมูลได้อย่างรวดเร็วและง่ายดาย NAS บางรุ่นยังสามารถติดตั้งในรูปแบบ RAID เพื่อปกป้องข้อมูลจากความเสียหายได้อีกด้วย NAS มักพบได้ทั้งในเครือข่ายภายในองค์กรและในระบบคลาวด์ส่วนตัว (Private Cloud)

อย่างไรก็ตาม การเพิ่มอุปกรณ์ NAS เข้าไปในเครือข่าย ก็หมายถึงการเพิ่มโหลดใหม่เข้าไปด้วยซึ่งจะใช้หมายเลข IP อย่างน้อยหนึ่งหมายเลข และต้องพึงพาความสามารถของแบบดิจิตอลท์เครือข่าย หากเครือข่ายเดิมมีปัญหาหรือถูกใช้งานหนักอยู่แล้ว การเพิ่ม NAS อาจทำให้เกิดความล่าช้าในการเข้าถึง

ข้อมูลของผู้ใช้และแอปพลิเคชันได้ แต่หากเครือข่ายมีแบบวิดท์เพียงพอ NAS จะเป็นทางเลือกที่ง่าย และประหยัดสำหรับการเพิ่มพื้นที่จัดเก็บข้อมูล

NAS ทำหน้าที่เหมือนเซิร์ฟเวอร์ไฟล์พื้นฐาน โดยจะแชร์ไฟล์เดอร์ให้กับคลาวน์ผ่านโปรโตคอล:

- **Network File System (NFS)** สำหรับระบบ Unix/Linux
- **Server Message Block (SMB)** สำหรับระบบ Windows

แม้ว่า NAS จะติดตั้งง่ายและมีราคาถูก แต่ประสิทธิภาพของมันก็ขึ้นอยู่กับประสิทธิภาพโดยรวมของเครือข่าย ดังนั้น NAS อาจไม่เหมาะสมกับการเป็นโซลูชันการจัดเก็บข้อมูลหลักในระดับองค์กร

1.8.3 Storage Area Network (SAN)

อุปกรณ์ Storage Area Network (SAN) มอบความสามารถในการขยายตัว ความทนทานต่อความล้มเหลว และประสิทธิภาพในการเข้าถึงข้อมูลที่สูงกว่า NAS อย่างมีนัยสำคัญ แต่ก็มีความซับซ้อนและมีต้นทุนสูงกว่า เช่นกัน SAN มักพบในดาต้าเซ็นเตอร์ของระบบคลาวด์แบบส่วนตัว (Private Cloud)

แม้ทางเทคนิคแล้ว SAN จะหมายถึงโครงข่ายสนับสนุนที่เชื่อมต่อระหว่างเซิร์ฟเวอร์และอุปกรณ์จัดเก็บข้อมูล แต่โซลูชัน SAN ที่สมบูรณ์มักประกอบด้วยองค์ประกอบหลักสามประการ (นอกเหนือจากเครื่องลูกปั่ย):

- เซิร์ฟเวอร์หนึ่งเครื่องหรือมากกว่าที่ทำหน้าที่บริหารการเข้าถึงข้อมูล
- เครือข่ายเฉพาะสำหรับการสื่อสารระหว่างเซิร์ฟเวอร์กับระบบจัดเก็บข้อมูล
- โครงสร้างพื้นฐานการจัดเก็บข้อมูล ซึ่งประกอบด้วยไดรฟ์จำนวนมาก, ตัวควบคุม, และองค์ประกอบอื่นๆ ที่เชื่อมต่อกับเครือข่าย SAN

เซิร์ฟเวอร์อาจเป็นระบบปฏิบัติการ Windows Server ที่ติดตั้งการ์ด Host Bus Adapter (HBA) เพื่อเชื่อมต่อกับระบบจัดเก็บข้อมูล โครงข่ายเฉพาะสำหรับ SAN นี้มักถูกออกแบบให้มีความซับซ้อนสูง เพื่อให้การเข้าถึงข้อมูลมีความพร้อมใช้งานอย่างต่อเนื่อง

ในส่วนของโครงสร้างการจัดเก็บข้อมูล จะอยู่ในกล่องแยกต่างหากจากเซิร์ฟเวอร์ และมักเป็นแรร์ หรืออาร์เรย์ของดิสก์ที่มีประสิทธิภาพสูง SAN ช่วยให้สามารถเชื่อมต่อกับอุปกรณ์จัดเก็บข้อมูลประเภทต่าง ๆ ได้ เช่น เทปหรืออุปติคัลเมดี้ และสามารถเชื่อมโยงข้อมูลข้ามสถานที่ตั้งได้ เช่น ศูนย์ข้อมูลระยะไกล

หน่วยที่ 2 แนวคิดเกี่ยวกับการใช้คอนเทนเนอร์ (Containerization Concepts)

เครื่องเสมือน (Virtual Machines หรือ VMs) เป็นพื้นฐานสำหรับการใช้งานระบบ Infrastructure as a Service (IaaS) ส่วนใหญ่ อย่างไรก็ตาม “การใช้คอนเทนเนอร์ (Containerization)” ก็เป็นอีกรูปแบบหนึ่งของการจำลองเสมือน (Virtualization) ที่ให้ประโยชน์มากมายสำหรับนักพัฒนาซอฟต์แวร์ที่ต้องการสร้างแอปพลิเคชันสมัยใหม่ที่สามารถปรับขนาดได้ รองรับระบบคลาวด์ และสนับสนุนการพัฒนาที่ต่อเนื่อง

บทเรียนนี้จะกล่าวถึงทางเลือกในการใช้คอนเทนเนอร์ในระบบคลาวด์ (Cloud Containerization Options)

2.1 การใช้คอนเทนเนอร์ (Containerization)

คอนเทนเนอร์ (Containers) เป็นแนวทางใหม่ในการออกแบบซอฟต์แวร์ การเพิ่มความพร้อมใช้งาน (Availability) และการปรับขนาด (Scalability) ของแอปพลิเคชัน โดยแอปพลิเคชันจะมีความยืดหยุ่นมากขึ้น และสามารถพัฒนาและปรับปรุงได้อย่างต่อเนื่อง ด้วยสถาปัตยกรรมแบบหลวม (Loosely Coupled Architecture) ซึ่งอิงจากการจำลองเสมือนในรูปแบบคอนเทนเนอร์

บทเรียนนี้จะครอบคลุมแนวคิดเกี่ยวกับคอนเทนเนอร์ (Container Concepts) การใช้คอนเทนเนอร์แบบสแตนเดอร์ดออลอน (Stand-alone) และแบบอยู่บนคลาวด์ (Cloud-based containerization) ที่เก็บคอนเทนเนอร์ (Container Registries) และแนวปฏิบัติในการบริหารจัดการ

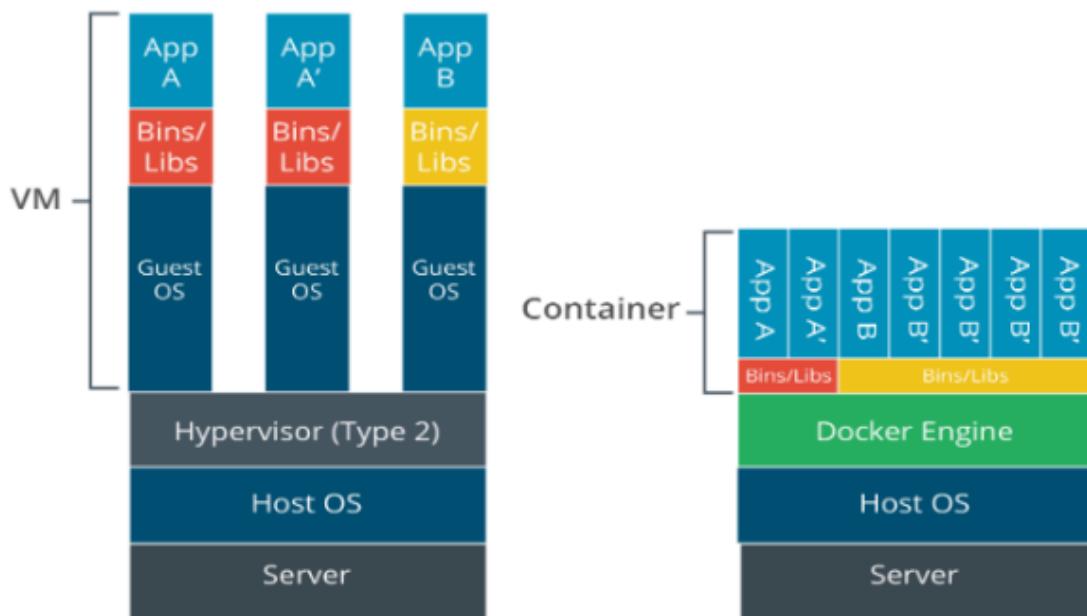
การดูแลโครงสร้างพื้นฐานของคอนเทนเนอร์ (Container Infrastructures) เป็นบทบาทสำคัญของผู้ดูแลระบบคลาวด์ (Cloud Administrator)

โดยทั่วไปแล้ว คอนเทนเนอร์มักเป็นส่วนหนึ่งของกลยุทธ์การจัดการเวิร์กโหลดในระดับอัตโนมัติ (Workload Orchestration Strategy) ซึ่งเนื้อหาเกี่ยวกับระบบอัตโนมัติ (automation) และการจัดระเบียบการทำงาน (Orchestration) จะกล่าวถึงในบทเรียนถัดไป

2.2 การใช้คอนเทนเนอร์แบบสแตนเดอร์ดออลอน (Stand-Alone Containerization)

คอนเทนเนอร์เป็นรูปแบบของการจำลองเสมือน (Virtualization) ที่แตกต่างจากเครื่องเสมือน (Virtual Machine – VM) โดยคอนเทนเนอร์จะเป็นชุดซอฟต์แวร์แบบสมมูลรูณ์และสามารถพกพาได้ ซึ่งรวมถึงโค้ดของแอปพลิเคชัน รันไทม์ ไลบรารี การตั้งค่าต่าง ๆ และองค์ประกอบอื่น ๆ ที่จำเป็นต่อการทำงานของซอฟต์แวร์ทั้งหมดไว้ในแพ็คเกจเดียว คอนเทนเนอร์สามารถนำไปใช้งานบนแพลตฟอร์มใดก็ได้ที่มีเอนจินของคอนเทนเนอร์ (Container Engine) รวมถึงโครงสร้างพื้นฐานของคลาวด์

Container vs. VMs



Comparing virtual machines and containers. Note the container engine with multiple containers.

ภาพที่ 29 Stand-Alone Containerization

คอนเทนเนอร์มีขนาดเบา ใช้ระบบปฏิบัติการเดียวกันร่วมกัน (โดยทั่วไปเป็น Linux) และถูกออกแบบมาเพื่อให้ทำหน้าที่เฉพาะด้านเพียงหนึ่งเดียว

2.2.1 เริ่มต้นใช้งานคอนเทนเนอร์

เอนจินของคอนเทนเนอร์ (Container engine) จะเป็นตัวรันคอนเทนเนอร์หนึ่งตัวหรือมากกว่าบนระบบปฏิบัติการเดียวกัน การเลือกและติดตั้ง Container Engine บนเซิร์ฟเวอร์เป็นขั้นตอนแรกของการใช้งานคอนเทนเนอร์แบบโอลคัล ในคลาวด์ ผู้ให้บริการจะมีแพลตฟอร์ม Container แบบจัดการให้ (Managed Container Platform)

Container engine ที่รองรับมาตรฐานของ Open Container Initiative (OCI) ได้แก่:

- Docker
- Hyper-V และ Windows Containers
- Kubernetes
- Podman

คอนเทนเนอร์สามารถติดตั้งได้ทั้งบนเซิร์ฟเวอร์จริง (Physical Servers), ระบบในองค์กร (On-premises), หรือโครงสร้างพื้นฐานคลาวด์ อีกทั้งยังสามารถใช้บนเวิร์กสเตชันส่วนบุคคลได้ การใช้คอนเทนเนอร์ในคลาวด์

จะได้รับประโยชน์ด้านการปรับขนาด (Scalability), ความพร้อมใช้งานสูง (High Availability), และการใช้งานที่รวดเร็ว

2.2.2 เข้าใจองค์ประกอบหลักของระบบคอนเทนเนอร์

ระบบคอนเทนเนอร์ประกอบด้วย 3 ส่วนหลัก:

- ไฟล์กำหนดค่า (Configuration File)
- อิมเมจของคอนเทนเนอร์ (Container Image)
- คอนเทนเนอร์ที่กำลังทำงาน (Running Container)

ตัวอย่างโดยใช้ Docker:

- Dockerfile: ไฟล์ข้อความที่ใช้ระบุคำสั่งสำหรับสร้างอิมเมจ เช่น โค้ดไลบรารี เครื่องมือ และ dependencies ทั้งหมด (คล้ายกับสเปคหรือคู่มือการติดตั้ง)
- Container Image: สร้างจาก Dockerfile โดยใช้คำสั่ง docker build เพื่อสร้างเทมเพลตของคอนเทนเนอร์
- Running Container: คอนเทนเนอร์ที่กำลังทำงานคืออินสแตนซ์ของอิมเมจ ใช้คำสั่ง docker run ในการรัน

คุณสมบัติสำคัญของคอนเทนเนอร์

- คอนเทนเนอร์เป็นแบบ stateless หมายความว่าการเปลี่ยนแปลงใด ๆ ภายในคอนเทนเนอร์จะไม่คงอยู่ เมื่อหยุดการทำงาน
- หากต้องการให้การเปลี่ยนแปลงถาวร จะต้องปรับเปลี่ยนที่ไฟล์กำหนดค่า (Dockerfile) และสร้างอิมเมจใหม่

2.3 การใช้งานคอนเทนเนอร์ (Deploy Containers)

การเริ่มต้นใช้งานคอนเทนเนอร์บนระบบภายใน (local system) เป็นเรื่องที่ง่าย ขั้นแรกคือการเลือกและติดตั้งเอนจินของคอนเทนเนอร์ เช่น Docker ซึ่งเป็นตัวเลือกยอดนิยมในหมู่นักพัฒนา เมื่อติดตั้งเสร็จแล้ว ให้ดึง (pull) อิมเมจหนึ่งหรือหลายอิมเมจจาก container registry เช่น Docker Hub โดยอิมเมจแต่ละรายการ มักจะให้ความสามารถเฉพาะทางบางอย่าง

ตัวอย่างอิมเมจยอดนิยม:

- Docker hello-world (อิมเมจทดสอบ)
- Docker Alpine Linux (ระบบปฏิบัติการเบา)
- Docker Python (อิมเมจทางการของ Python)

Container engine เช่น Docker มีคำสั่งจำนวนมากสำหรับการสร้าง ลบ เริ่ม หยุด และจัดการคอนเทนเนอร์ เพื่อสร้างสภาพแวดล้อมที่แข็งแกร่ง มีเครือข่าย และเชื่อมโยงกันได้อย่างยืดหยุ่น

คำสั่ง docker

คำสั่งหลักในการจัดการ Docker คือ docker โดยมีรูปแบบ:

`docker [subcommand] {options} {arguments}`

คำสั่งย่อย (subcommands) ที่พบบ่อย:

คำสั่งย่อย (Subcommand)	คำอธิบาย (Description)
info	แสดงข้อมูลระบบโดยรวม
port	แสดงพอร์ตที่แมปไว้ของคอนเทนเนอร์
pull	ดึงอิมเมจจากรีจิสทรี
push	อัปโหลดอิมเมจขึ้นรีจิสทรี
ps	แสดงรายการคอนเทนเนอร์ที่ทำงานอยู่
rm	ลบคอนเทนเนอร์
rmi	ลบอิมเมจ
run	รันคำสั่งในคอนเทนเนอร์

ตาราง 2 คำสั่งย่อย

2.4 ระบบจัดเก็บอิมเมจของคอนเทนเนอร์ (Image Registries)

Image registry หรือระบบจัดเก็บอิมเมจของคอนเทนเนอร์ คือแหล่งเก็บข้อมูลของอิมเมจคอนเทนเนอร์ที่แอปพลิเคชันและนักพัฒนาต้องการใช้งาน โดยระบบนี้ไม่เพียงแค่เก็บไฟล์เท่านั้น แต่ยังเป็นศูนย์กลางในการรักษาความปลอดภัยและแบ่งปันอิมเมจให้กับทีมพัฒนาและระบบต่าง ๆ ได้อย่างมีประสิทธิภาพ

อย่างไรก็ตาม ความสามารถของ registry มีมากกว่าการจัดเก็บข้อมูลทั่วไป กล่าวคือสามารถสนับสนุนรวมเข้ากับกระบวนการ DevOps และระบบ CI/CD (Continuous Integration/Continuous Deployment) ได้โดยตรง ซึ่งหมายความว่า ระบบอัตโนมัติที่ใช้สร้างและปรับใช้บริการต่าง ๆ สามารถดึงอิมเมจจาก registry ได้ทันทีแบบเรียลไทม์ ซึ่งเป็นหัวใจสำคัญของแนวคิด CI/CD สมัยใหม่

ประเภทของ Container Image Registries

1. Public Container Registries (ระบบสาธารณะ)

- เริ่มต้นใช้งานได้ง่าย
- เหมาะสำหรับผู้เริ่มต้นหรือการทดลอง
- ตัวอย่างเช่น Docker Hub
- ข้อดีคือสะดวก ประหยัดต้นทุน และสามารถเข้าถึงอิมเมจจำนวนมากจากชุมชนผู้พัฒนา

2. Private Container Registries (ระบบส่วนตัว)

- เหมาะกับองค์กรที่มีระบบอัตโนมัติหรือโครงสร้าง CI/CD ที่ซับซ้อน
- ให้การควบคุมที่เข้มงวดกว่าในด้านการจัดการอิมเมจและความปลอดภัย
- สามารถตั้งอยู่ได้ทั้งในระบบ on-premises, private cloud, หรือ public cloud
- เหมาะกับการใช้งานในระดับองค์กร

ข้อดีของระบบ Image Registry โดยรวม

- Scalability (สามารถขยายระบบได้ตามต้องการ)
- Security (มีความปลอดภัยสูง)
- Management Control (ควบคุมการจัดการอิมเมจได้ง่าย)
- Automation/Orchestration Integration (รองรับการผสานกับระบบอัตโนมัติและการจัดการบริการ) ประโยชน์เหล่านี้จะสามารถนำมาใช้ได้มากน้อยเพียงใด ขึ้นอยู่กับประเภทของ registry ที่ใช้งาน (public หรือ private)

ตัวอย่างผู้ให้บริการ Container Image Registry บนระบบคลาวด์:

- Amazon Elastic Container Registry (ECR)
- Azure Container Registry
- Google Artifact Registry (ซึ่งมาแทน Google Container Registry ที่เลิกใช้แล้ว)

2.5 การใช้งานคอนเทนเนอร์บนระบบคลาวด์ (Containers in the Cloud)

การไฮสต์บริการคอนเทนเนอร์บนเครื่องเวิร์กสเตชันหรือเซิร์ฟเวอร์ภายในองค์กร (on-premises) เป็นทางเลือกหนึ่งที่ใช้งานได้จริง แต่มีข้อจำกัดในเรื่อง การขยายระบบ (scalability) และ ความพร้อมใช้งานสูง (high availability) ซึ่งอาจทำได้ยากเมื่อระบบเติบโตขึ้น

การย้ายบริการคอนเทนเนอร์ไปยัง ระบบคลาวด์ ช่วยให้คุณสามารถใช้ประโยชน์จากข้อดีของคลาวด์ เช่น:

- การขยายระบบได้ง่าย (scalability)
- บริการแบบจ่ายตามการใช้งาน (metered services)
- ความพร้อมใช้งานสูง (availability)
- ลดภาระการบริหารจัดการระบบ (reduced management overhead)

ตัวอย่างบริการคอนเทนเนอร์จากผู้ให้บริการคลาวด์:

- Amazon Elastic Container Service (ECS):

ใช้สำหรับรัน, ตรวจสอบ และปรับขนาดแอปพลิเคชัน พร้อมการผสานรวมกับบริการอื่นของ AWS

- Azure Kubernetes Service (AKS):

ใช้สำหรับรันและปรับขนาดคอนเทนเนอร์บนแพลตฟอร์ม Kubernetes

- Google Compute Engine:

ใช้สำหรับรันและจัดการคอนเทนเนอร์บนแพลตฟอร์มต่าง ๆ เช่น Kubernetes, Docker และอื่น ๆ

ผู้ให้บริการคลาวด์เหล่านี้ยังมีบริการด้านคอนเทนเนอร์อื่น ๆ อีกมากมาย

การปรับใช้งานคอนเทนเนอร์ในระบบคลาวด์ (Deploy containers in the cloud)

การบริหารจัดการคอนเทนเนอร์เป็นองค์ประกอบสำคัญของระบบคลาวด์ เช่นเดียวกับการจัดการเครื่องเสมือน (Virtual Machine) และในหลายองค์กร การใช้งานคอนเทนเนอร์อาจเป็นแรงผลักดันหลักของการเปลี่ยนผ่านสู่ระบบคลาวด์

ผู้ให้บริการคลาวด์มี คอนโซลการจัดการคอนเทนเนอร์ ที่ออกแบบมาโดยเฉพาะ ซึ่งช่วยให้ผู้ดูแลระบบสามารถสร้างและควบคุมอินสแตนซ์ของคอนเทนเนอร์ได้อย่างง่ายดาย

ขั้นตอนทั่วไปในการปรับใช้คอนเทนเนอร์ด้วย Azure Container Instances Console:

1. เลือก **Create container instance** (สร้างอินสแตนซ์ของคอนเทนเนอร์)
2. ตั้งชื่ออินสแตนซ์ เช่น container07
3. เลือกภูมิภาคที่ใช้โฮสต์ (hosting region) และ **Availability Zone** (เขตความพร้อมใช้งาน) — ควรเลือกพื้นที่ใกล้คุณ
4. เลือกแหล่งที่มาของอิมเมจจาก **Image Source Registry** เช่น Quickstart images และเลือกอิมเมจ เช่น helloworld:latest (Linux)
5. เลือกขนาดของอิมเมจที่ต้องการใช้งาน เช่น จำนวนคอร์ของ CPU และขนาดของหน่วยความจำ (Memory)
6. กำหนดค่าการเชื่อมต่อเครือข่าย เช่น การใช้ Virtual Network และ Subnet
7. เพิ่มแท็ก (Tags) เพื่อใช้ในการตรวจสอบการใช้งานและการเรียกเก็บค่าบริการ (Metering and Billing)

2.6 การจัดการคอนเทนเนอร์ (Container Management)

คอนเทนเนอร์มีความยืดหยุ่นในการกำหนดค่า เช่น ด้าน **เครือข่าย (Networking)** และ **การจัดเก็บข้อมูล (Storage)** ซึ่งช่วยให้ออปเพลิเคชันและบริการที่ทำงานภายใต้คอนเทนเนอร์สามารถเชื่อมต่อกับระบบเครือข่ายทั่วไป และเก็บรักษาข้อมูลได้นอกเหนือจากช่วงเวลาที่คอนเทนเนอร์ทำงานอยู่

นักพัฒนาแอปเพลิเคชันมักออกแบบระบบให้มีความเชื่อมโยงกันแบบหลวม ๆ (Loosely Coupled Architecture) โดยใช้คอนเทนเนอร์เพื่อให้บริการมีความยืดหยุ่นมากขึ้น การทำ อัตโนมัติ (Automation) จะช่วยนำการจัดการคอนเทนเนอร์เข้าสู่ระบบ การจัดการแบบออร์คेसเตรต (Orchestrated Workloads) ที่มีขนาดใหญ่

2.6.1 การแมปพอร์ต (Port Mapping)

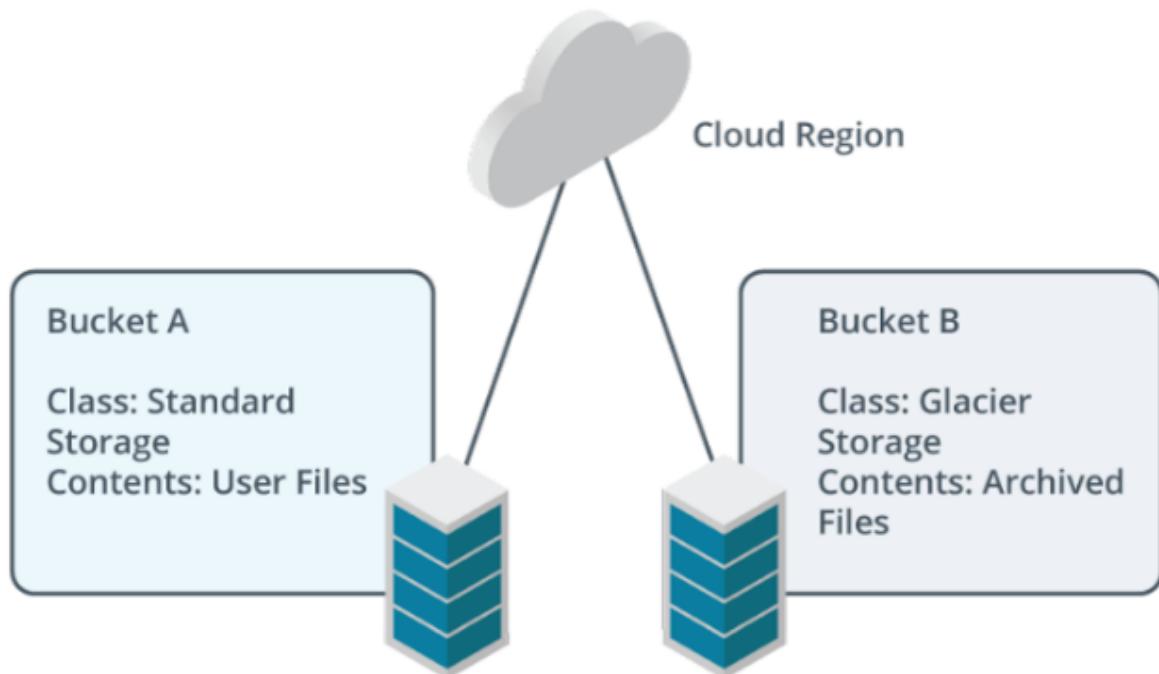
โดยทั่วไป คอนเทนเนอร์จะเปิดใช้งานเครือข่ายแบบพื้นฐาน แต่ในค่าเริ่มต้นจะสามารถสื่อสารได้เฉพาะกับคอนเทนเนอร์อื่นภายในเครือข่ายเดียวกันเท่านั้น ไม่สามารถเข้าถึงบริการบนโฮสต์หรือเครือข่ายภายนอกได้โดยตรง

การแมปพอร์ต (Port Mapping) จะช่วยเชื่อมพอร์ตของคอนเทนเนอร์เข้ากับพอร์ตของไฮสต์ ทำให้บริการภายในคอนเทนเนอร์สามารถเข้าถึงได้จากภายนอก เช่น หากคุณรันเว็บเซิร์ฟเวอร์ nginx บนคอนเทนเนอร์ ระบบจะเข้าถึงได้จากคอนเทนเนอร์อินภายนอกในท่านี้ แต่หากกำหนดค่าแมปพอร์ต ผู้ใช้ภายนอก ก็สามารถดูเว็บไซต์ที่รันในคอนเทนเนอร์นั้นได้

ในระบบคลาวด์ ค่าการแมปพอร์ตจะถูกกำหนดไว้ในไฟล์คอนฟิกของคอนเทนเนอร์ เช่นเดียวกัน และสามารถจัดการโดยอัตโนมัติได้เป็นส่วนหนึ่งของการออร์เคสเตรต

2.6.2 ประเภทของพื้นที่จัดเก็บข้อมูล (Storage Types)

การจัดเก็บข้อมูลของคอนเทนเนอร์อาจมีความซับซ้อน คอนเทนเนอร์โดยพื้นฐานจะใช้ พื้นที่จัดเก็บ ข้อมูลแบบชั่วคราว (Ephemeral Storage) ซึ่งจะหายไปเมื่อคอนเทนเนอร์ถูกลบหรือล้มเหลว อย่างไรก็ตาม แอปพลิเคชันบางประเภทจำเป็นต้องเก็บข้อมูลไว้นอกช่วงชีวิตของคอนเทนเนอร์ จึงมีการใช้งานพื้นที่จัดเก็บข้อมูล 2 แบบ:



Storage buckets with different classes and stored contents. (Images © 123RF.com)

ภาพที่ 30 Storage Types

พื้นที่จัดเก็บแบบชั่วคราว (Ephemeral Storage)

- ข้อมูลจะสูญหายเมื่อคอนเทนเนอร์หยุดทำงานหรือถูกลบ
- ใช้ในกรณีที่ไม่จำเป็นต้องเก็บข้อมูลระยะยาว เช่น:
 - การแกะข้อมูล
 - การบันทึก Log ของคอนเทนเนอร์

○ พื้นที่ชั่วคราวอื่น ๆ

พื้นที่จัดเก็บแบบถาวร (Persistent Storage)

- ข้อมูลจะยังคงอยู่แม้คอนเทนเนอร์จะหยุดทำงานหรือถูกลบ
- เหมาะสมสำหรับแอปพลิเคชันแบบ Stateful ที่ต้องเก็บข้อมูลระยะยาว
- ตัวอย่างเช่น การกำหนด Persistent Volume บนระบบคลาวด์ผ่าน Amazon EFS เพื่อใช้ร่วมกับคลัสเตอร์ของคอนเทนเนอร์ที่รันบน Amazon EKS

2.6.3 การออร์คेसเตրต (Orchestration)

การออร์คेसเตรตงานของคอนเทนเนอร์ (Container Workload Orchestration) ช่วยให้การจัดการระบบที่มีขนาดใหญ่เป็นไปได้อย่างมีประสิทธิภาพ โดยเฉพาะสำหรับองค์กรที่ใช้แนวทาง DevOps ข้อดีของการออร์คेसเตรต ได้แก่:

- การปรับใช้งานที่มีประสิทธิภาพและยืดหยุ่น
- การจัดการที่ง่ายและเป็นระบบอัตโนมัติ
- การบริหารจัดการตารางการทำงาน (Scheduling) และความพร้อมใช้งาน
- ความสามารถในการขยายระบบได้มากขึ้น
- การตรวจสอบการทำงานของคอนเทนเนอร์

เมื่องานเต็บโตก็จะต้องใช้คอนเทนเนอร์จำนวนมาก ระบบออร์คेसเตรตจะกลายเป็นสิ่งจำเป็นการทำงานแบบอัตโนมัตินี้ยังรวมถึงการกำหนดพอร์ตและการเลือกประเภทการจัดเก็บข้อมูลให้เหมาะสมกับแต่ละแอปพลิเคชัน

หน่วยที่ 3 แนวคิดพื้นฐานเกี่ยวกับฐานข้อมูล (Database Concepts)

ผู้ดูแลระบบคลาวด์ (Cloud Administrators) อาจมีหน้าที่ช่วยเหลือหรือแม้แต่เป็นผู้กำหนดค่าระบบฐานข้อมูลในองค์กร โดยการทำความเข้าใจพื้นฐานเกี่ยวกับฐานข้อมูลจึงเป็นสิ่งสำคัญ โดยเริ่มจาก การเข้าใจโครงสร้างของฐานข้อมูลสองประเภทหลัก ได้แก่ ฐานข้อมูลแบบมีความสัมพันธ์ (Relational Database) และ ฐานข้อมูลแบบไม่มีความสัมพันธ์ (Non-relational Database)

3.1 ประเภทของฐานข้อมูล (Database Types)

ข้อมูลมักถูกจัดเก็บไว้ในระบบฐานข้อมูล ซึ่งสามารถแบ่งออกได้เป็น 2 ประเภทหลัก โดยแต่ละประเภท มีลักษณะเฉพาะและการใช้งานที่แตกต่างกัน ดังนี้:

- ฐานข้อมูลเชิงสัมพันธ์ (Relational):

ใช้โครงสร้างแบบตาราง ซึ่งประกอบด้วยคอลัมน์และแถว เพื่อจัดระเบียบข้อมูลที่มีโครงสร้างอย่างชัดเจน (Structured Data)

- **ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (Non-relational):**
ใช้การออกแบบพิเศษหลากหลายรูปแบบ เพื่อรับข้อมูลที่ไม่มีโครงสร้างแน่นอน (Unstructured Data)
เช่น ข้อมูลแบบ JSON, เอกสาร, หรือกราฟข้อมูล

บริบทในการใช้งานฐานข้อมูล

ฐานข้อมูลสามารถมีขนาดใหญ่และซับซ้อนได้มาก เช่น

- การเก็บข้อมูลลูกค้า
- ข้อมูลคลังสินค้า
- ข้อมูลธุรกิจภายในองค์กร

ลองนึกถึงฐานข้อมูลของผู้ให้บริการค้าปลีกออนไลน์หรือผู้ให้บริการเครือข่ายโทรศัพท์มือถือ ที่ต้องจัดเก็บและเข้าถึงข้อมูลลูกค้าจำนวนมากมหาศาลอยู่ตลอดเวลา

ผู้ใช้งาน เช่น ลูกค้าและพนักงาน สามารถดึงข้อมูลจากฐานข้อมูลได้ผ่านการใช้คำสั่ง query ซึ่งเป็นคำสั่งที่ใช้เรียกดูข้อมูลเฉพาะตามที่ต้องการ

แนวโน้มขององค์กรในปัจจุบัน

หลายองค์กรได้เปลี่ยนแนวทางจากการจัดเก็บฐานข้อมูลในระบบภายในองค์กร (On-premises) มาเป็นการใช้บริการฐานข้อมูลในระบบคลาวด์ (Cloud-hosted Databases) ซึ่งมีข้อดี เช่น:

- ลดภาระในการดูแลโครงสร้างของฐานข้อมูล
- ได้รับประโยชน์จากการมีอัตราการขยายตัวของระบบคลาวด์
- ยังคงสามารถควบคุมข้อมูลได้ตามความต้องการ

แนวทางในการใช้งานฐานข้อมูลในระบบคลาวด์

1. Hosted Database Services:

องค์กรอับโหลดข้อมูลไปยังระบบฐานข้อมูลที่ให้บริการโดยผู้ให้บริการคลาวด์ โดยไม่ต้องดูแลโครงสร้างฐานข้อมูลเอง

2. Self-Managed Databases on IaaS:

องค์กรสามารถสร้างและบริหารจัดการโครงสร้างฐานข้อมูลเอง บนแพลตฟอร์มคลาวด์ประเภท Infrastructure as a Service (IaaS)

3.2 ฐานข้อมูลเชิงสัมพันธ์ (Relational Databases)

ฐานข้อมูลประเภทแรกที่กล่าวถึงคือ ฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลประเภทนี้มีการใช้งานมาอย่างยาวนาน ได้รับการยอมรับว่า มีความปลอดภัย, เข้าใจง่าย, และมีเสถียรภาพ

องค์กรของคุณอาจได้รับประโยชน์จากการใช้ฐานข้อมูลเชิงสัมพันธ์ในระบบคลาวด์

3.2.1 ลักษณะของฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลเชิงสัมพันธ์ใช้ ตาราง (Table) ในการจัดเก็บข้อมูล โดยใช้ คีย์ (Key) เพื่อเชื่อมโยงข้อมูลระหว่างตาราง

- ตาราง (Table): โครงสร้างหลักในการจัดเก็บข้อมูล
- คอลัมน์ (Column): แสดงประเภทของข้อมูลหรือคุณลักษณะ เช่น ชื่อ, ที่อยู่, เบอร์โทรศัพท์
- แถว (Row): แสดงข้อมูลแต่ละรายการ เช่น รายชื่อลูกค้าแต่ละคน

ตัวอย่างตารางลูกค้า:

Customer ID	Name	Address	Phone Number	Email Address
1000	Kai Garcia	33 First Street	(123) 456-7890	kai.garcia@example.com
1001	22 Second Street	(123) 890-4567		
1002	11 Third Street	(987) 654-3210		

ตาราง 3 ตัวอย่างฐานข้อมูลลูกค้า

คีย์ในฐานข้อมูลเชิงสัมพันธ์

- Primary Key: เป็นคีย์หลักที่กำหนดให้ระบุข้อมูลแต่ละແറาไว้ไม่ซ้ำกัน เช่น Customer ID
- Foreign Key: เป็นคีย์ที่ใช้เชื่อมโยงไปยัง Primary Key ของตารางอื่น

3.2.2 การสืบค้นข้อมูล (Query)

ฐานข้อมูลเชิงสัมพันธ์ใช้ภาษา SQL (Structured Query Language) ในการสืบค้นข้อมูล เช่น คำสั่งด้านล่างนี้ใช้ดึงข้อมูลหัวลูกค้าและชื่อจากตาราง customers:

```
SELECT Customer_ID, Name
```

```
FROM customers;
```

- บรรทัด SELECT ระบุชื่อคอลัมน์ที่ต้องการดึงข้อมูล
- บรรทัด FROM ระบุตารางที่ต้องการใช้งาน

ตัวอย่างบริการฐานข้อมูลเชิงสัมพันธ์จากผู้ให้บริการคลาวด์

- Microsoft Azure: MS SQL Server, PostgreSQL, MySQL, MariaDB

- Amazon Web Services (AWS): Aurora, RDS, Redshift
- Google Cloud Platform (GCP): Bare Metal Solution, Cloud SQL, Cloud Spanner

ข้อดีของฐานข้อมูลเชิงสัมพันธ์

- เหมาะสำหรับข้อมูลที่มีโครงสร้าง (Structured Data)
- รองรับคำสั่ง query ที่ซับซ้อนได้
- รักษาความถูกต้องและความสอดคล้องของข้อมูล (Data Integrity & Consistency)

3.3 ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ (Non-relational Databases)

ไม่ใช่ข้อมูลทุกประเภทที่สามารถจัดเก็บในรูปแบบตารางและเข้มข้นด้วยคีย์ได้ง่าย

การใช้งาน Big Data อย่างแพร่หลายในสภาพแวดล้อมธุรกิจปัจจุบัน ทำให้ต้องมีวิธีการจัดเก็บข้อมูลที่แตกต่างออกไป

ระบบฐานข้อมูล NoSQL หรือ ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ ถูกออกแบบมาเพื่อจัดเก็บ ข้อมูลที่ไม่มีโครงสร้าง (Unstructured Data) โดยแยกประเภทตามลักษณะข้อมูล

การออกแบบลักษณะนี้ทำให้เกิดความ ยืดหยุ่นสูง, สามารถปรับขนาดของแอปพลิเคชันได้ง่าย, และ รองรับ การเติบโตของข้อมูลอย่างรวดเร็ว

คุณลักษณะเหล่านี้ทำให้ เหมาะสำหรับการใช้งานระบบคลาวด์

ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ประเภทหลัก

มีฐานข้อมูลแบบไม่ใช่เชิงสัมพันธ์อยู่ 4 ประเภทหลัก ดังนี้:

- Document-oriented: เก็บข้อมูลในรูปแบบเอกสาร เช่น JSON, BSON
- Key-value: จัดเก็บข้อมูลเป็นคู่คีย์และค่า (Key-Value Pairs)
- Wide column: ใช้คอลัมน์แบบยืดหยุ่น เก็บข้อมูลในตารางที่สามารถเปลี่ยนแปลงโครงสร้างได้
- Graph stores: ออกแบบเพื่อจัดเก็บและวิเคราะห์ความสัมพันธ์ระหว่างข้อมูล เช่น โหนดและเส้น เชื่อมโยง

ตัวอย่างการใช้งาน

สมมุติว่าองค์กรของคุณต้องการจัดเก็บข้อมูลสินค้า ที่ได้จากโพสต์ของลูกค้าในโซเชียลมีเดีย

ข้อมูลเหล่านี้อาจประกอบด้วย ไฟล์ผู้ใช้, โพสต์, การกดไลค์, ความคิดเห็น, รูปภาพ, วิดีโอ หรือไฟล์เสียง

ในกรณีนี้ ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ จะเหมาะสมมากกว่าฐานข้อมูลเชิงสัมพันธ์ในการจัดเก็บข้อมูลที่หลากหลายและไม่มีโครงสร้างตายตัว

อีกตัวอย่างหนึ่งคือ บริษัทที่เก็บข้อมูลจากอุปกรณ์ IoT ในภาคเกษตรกรรม เช่น อุณหภูมิ, ความชื้น, วันที่, ระดับความชุ่มชื้นในดิน และข้อมูลจากเซนเซอร์อื่น ๆ

ข้อมูลเหล่านี้มีโครงสร้างที่คาดเดาได้ยาก การใช้ ฐานข้อมูลที่ไม่ใช่เชิงสัมพันธ์ จะมีความสามารถมากกว่า

ตัวอย่างบริการฐานข้อมูล NoSQL จากผู้ให้บริการระบบคลาวด์

- Microsoft Azure: Azure Cosmos DB
- Amazon Web Services (AWS): DynamoDB, Neptune, DocumentDB
- Google Cloud Platform (GCP): Cloud Bigtable, Firestore, MongoDB

3.4 รูปแบบการปรับใช้ฐานข้อมูล (Database Deployment Options)

บริการคลาวด์มีตัวเลือกในการปรับใช้ฐานข้อมูลหลักอยู่ 2 รูปแบบ ได้แก่

- ฐานข้อมูลที่องค์กรจัดการเอง (Self-managed)
- ฐานข้อมูลที่ผู้ให้บริการคลาวด์จัดการให้ (Provider-managed)

องค์กรอาจเลือกใช้ทั้งสองรูปแบบควบคู่กัน ขึ้นอยู่กับความต้องการ ความสามารถภายในองค์กร และความเหมาะสมของข้อมูลแต่ละประเภท

3.4.1 ฐานข้อมูลที่องค์กรจัดการเอง (Self-managed Databases)

ฐานข้อมูลแบบ Self-managed หมายถึงฐานข้อมูลที่ทีมภายในองค์กรเป็นผู้รับผิดชอบ ออกแบบ, ติดตั้ง, บริหารจัดการ, บำรุงรักษา และขยายระบบ เองทั้งหมด

ผู้ดูแลฐานข้อมูล (Database Administrators) ต้องรับผิดชอบทุกขั้นตอนของระบบฐานข้อมูล

ข้อดีของฐานข้อมูลแบบ Self-managed:

- ควบคุมข้อมูลได้เต็มที่
- มีความยืดหยุ่นในการออกแบบ
- หลีกเลี่ยงการพึ่งพาผู้ให้บริการ (Vendor lock-in)
- เลือกเทคโนโลยีและโครงสร้างฐานข้อมูลได้ตามต้องการ

ข้อจำกัดของฐานข้อมูลแบบ Self-managed:

- ต้องใช้เวลาและทรัพยากรจำนวนมาก
- ต้องมีทักษะทางเทคนิคที่เพียงพอ
- ต้องมีความมุ่งมั่นและความพร้อมในการดูแลระบบระยะยาว

องค์กรสามารถเลือกปรับใช้ฐานข้อมูลแบบนี้ได้ทั้งในสถานที่ (On-premises), ระบบคลาวด์ส่วนตัว (Private Cloud), หรือแม้แต่ในระบบคลาวด์สาธารณะ (Public Cloud) โดยที่ยังต้องดูแลฐานข้อมูลเองทั้งหมด แม้จะใช้โครงสร้างพื้นฐานของคลาวด์

3.4.2 ฐานข้อมูลที่ผู้ให้บริการคลาวด์จัดการให้ (Provider-managed Databases หรือ Database as a Service – DBaaS)

ฐานข้อมูลแบบ Provider-managed คือฐานข้อมูลที่ให้บริการผ่านระบบคลาวด์ โดยผู้ให้บริการ เป็นผู้ดูแลโครงสร้างพื้นฐาน การติดตั้ง การบำรุงรักษา และการอัปเดตระบบ แนวทางนี้ช่วยลดภาระของทีมไอทีภายในองค์กร และอาจช่วยลดค่าใช้จ่ายระยะยาวได้

ข้อดีของฐานข้อมูลแบบ Provider-managed:

- มีโครงสร้างระบบที่เป็นระเบียบ
- ไม่ต้องดูแลระบบเอง / มีทีมซัพพอร์ต
- รองรับระบบอัตโนมัติ (Automation)
- ช่วยลดต้นทุน และมีโมเดลค่าบริการแบบจ่ายตามการใช้งาน
- ขยายระบบได้ง่ายตามความต้องการ

ข้อจำกัดของฐานข้อมูลแบบ Provider-managed:

- มีความเสี่ยงจากการพึ่งพาผู้ให้บริการ (Vendor lock-in)
- มีข้อกังวลเรื่องความมั่นคงปลอดภัยบนคลาวด์สาธารณะ
- ปรับแต่งระบบได้น้อยกว่ารูปแบบ Self-managed
- อาจมีปัญหาด้านเขตอำนาจของข้อมูล (Data Sovereignty)

บริการฐานข้อมูลแบบ Provider-managed นี้เรียกว่า Database as a Service (DBaaS)

แนวคิดนี้คล้ายกับบริการ SaaS (Software as a Service) หรือ PaaS (Platform as a Service) ซึ่งมีเป้าหมายเพื่อช่วยลดภาระงานของทีมไอที ลดค่าใช้จ่ายในด้านทรัพยากร และช่วยให้สามารถจัดสรรกำลังคนไปพัฒนาบริการอื่น ๆ ได้มากขึ้น

3.5 การย้ายฐานข้อมูล (Migrate Databases)

การย้ายฐานข้อมูลถือเป็นความท้าทายที่แตกต่างจากการย้ายระบบเสมือน (Virtualized Systems) หรือข้อมูลเดิมทั่วไป เนื่องจากฐานข้อมูลมักมีความเชื่อมโยงกับการให้บริการที่ต่อเนื่องและมีความซับซ้อนสูง ด้วยเหตุนี้ ผู้ให้บริการคลาวด์ส่วนใหญ่ (Cloud Service Providers: CSPs) จึงมีโซลูชันเฉพาะสำหรับการย้ายฐานข้อมูล

ปัจจัยสำคัญที่สุดในการย้ายฐานข้อมูลคือ “ความพร้อมใช้งาน” (Availability)

ผู้ให้บริการคลาวด์มักมีระบบ replication ระหว่างกระบวนการย้ายข้อมูล กล่าวคือ การเปลี่ยนแปลงใด ๆ ที่เกิดขึ้นกับฐานข้อมูลต้นทาง จะถูกทำซ้ำไปยังฐานข้อมูลปลายทางโดยอัตโนมัติ ระบบนี้ช่วยให้การย้ายข้อมูลเกิด downtime น้อยที่สุด หรือแทบไม่มีเลย

โดยทั่วไป ควรย้ายแอปพลิเคชันและฐานข้อมูล Back-end พร้อมกัน การเก็บฐานข้อมูลไว้ภายในองค์กร (On-premises) ขณะที่แอปพลิเคชันทำงานอยู่บนคลาวด์นั้นไม่ใช่แนวทางที่นิยม เนื่องจากอาจเกิดปัญหาเรื่อง Latency และความไม่สอดคล้องของระบบ

การย้ายข้อมูลแบบตรงและข้ามบริการ (Direct and Cross-Service Migrations)

ตัวอย่างเช่น บริการ AWS Database Migration Service (DMS) รองรับทั้งการย้ายข้อมูลแบบตรง และแบบข้ามบริการ

- การย้ายข้อมูลแบบตรง (Direct migrations):

คือการย้ายข้อมูลระหว่างฐานข้อมูลที่มีโครงสร้างเดียวกัน เช่น

จาก Microsoft SQL Server ไปยัง Microsoft SQL Server (ระหว่างผู้ให้บริการ หรือจาก private cloud)

- การย้ายข้อมูลข้ามบริการ (Cross-service migrations):

คือการย้ายข้อมูลระหว่างฐานข้อมูลต่างระบบหรือแตกต่างกัน เช่น

จาก Oracle ไปยัง Amazon Aurora ซึ่งอาจต้องมีการแปลงข้อมูลหรือ Schema Conversion

สรุปการใช้เทคโนโลยีเสมือนจริงและฐานข้อมูล (Summary: Using Virtualization and Databases)

เครื่องเสมือน (Virtual Machines) และคอนเทนเนอร์ (Containers) ถือเป็นกลไกสำคัญและรากรฐานของโซลูชันซอฟต์แวร์บนระบบคลาวด์ในยุคปัจจุบัน

ผู้ดูแลระบบคลาวด์ (Cloud Administrators) มักมีบทบาทในการบริหารจัดการบริการเหล่านี้ในชีวิตประจำวัน โดยโครงสร้างพื้นฐานแบบเสมือนจริงจะต้องได้รับการปรับแต่งให้สอดคล้องกับข้อกำหนดด้านต้นทุน ประสิทธิภาพ และคุณลักษณะตามที่องค์กรต้องการ

ผู้ให้บริการคลาวด์ (Cloud Service Providers) มีทางเลือกมากมายในด้านการให้บริการระบบเสมือน เครื่องมือบริหารจัดการ และโซลูชันการตรวจสอบ (Monitoring Solutions) เพื่อรองรับการดำเนินงานที่สำคัญขององค์กร

หนึ่งในบริการที่สามารถใช้ประโยชน์อยู่บนเครื่องเสมือนได้คือ ฐานข้อมูล (Databases) ซึ่งองค์กรใช้ในการจัดเก็บและดึงข้อมูลสำคัญทางธุรกิจ เช่น ข้อมูลสินค้าคงคลัง ข้อมูลลูกค้า และทรัพยากร่วยในการออกแบบฐานข้อมูลสามารถแบ่งออกเป็น 2 แนวทางหลัก ได้แก่:

- ฐานข้อมูลเชิงสัมพันธ์ (Relational Databases)
- ฐานข้อมูลแบบไม่ใช้ความสัมพันธ์ (Non-relational Databases)

ความแตกต่างระหว่างสองประเภทนี้อยู่ที่รูปแบบการจัดเก็บข้อมูล

ผู้ดูแลฐานข้อมูล (Database Administrators) ต้องสามารถวิเคราะห์และเลือกใช้ฐานข้อมูลให้เหมาะสมกับประเภทของข้อมูลที่จัดเก็บ ขณะเดียวกัน ผู้ดูแลระบบคลาวด์มักจะมีส่วนร่วมในการตัดสินใจดังกล่าว เนื่องจากระบบคลาวด์ในปัจจุบันมีตัวเลือกฐานข้อมูลให้เลือกใช้อย่างหลากหลาย

บทที่ 7

การเข้าใจระบบเครือข่ายบนคลาวด์ (Comprehending Cloud Networking)

บทนำของบทเรียน (Module Introduction)

ระบบเครือข่ายบนคลาวด์ (Cloud Networking) มีคุณลักษณะหลายประการที่คล้ายคลึงกับการสนับสนุนระบบเครือข่ายแบบดั้งเดิมในองค์กร (On-Premises)

บทเรียนนี้จะอธิบายถึงแนวคิด องค์ประกอบ และบริการที่เกี่ยวข้องกับเครือข่ายบนคลาวด์ พร้อมทั้งกล่าวถึงการบูรณาการระหว่างเครือข่ายบนคลาวด์และเครือข่ายภายในองค์กร โดยใช้การเชื่อมต่อแบบตรง (Direct Connections), เครือข่ายส่วนตัวเสมือน (VPNs) และเครือข่ายที่กำหนดโดยซอฟต์แวร์ (Software-Defined Networking หรือ SDN)

การแก้ไขปัญหาเครือข่ายบนคลาวด์ยังคงใช้แนวทางที่คล้ายคลึงกับการแก้ไขปัญหาในเครือข่ายแบบภายในส่วนที่สองของบทเรียนนี้ กล่าวถึงทางเลือกในการแก้ไขปัญหาเฉพาะสถานการณ์ต่าง ๆ

วัตถุประสงค์ของบทเรียน (Module Objectives)

เมื่อเรียนจบบทเรียนนี้ คุณจะสามารถ:

- เข้าใจบริการและองค์ประกอบต่าง ๆ ของระบบเครือข่ายบนคลาวด์
- ทบทวนเทคนิคและเครื่องมือในการแก้ไขปัญหาเครือข่ายบนคลาวด์

หน่วยที่ 1 แนวคิดเกี่ยวกับระบบเครือข่ายบนคลาวด์ (Cloud Networking Concepts)

หนึ่งในประเด็นสำคัญของระบบเครือข่ายบนคลาวด์ คือ การเชื่อมต่อระหว่างผู้ใช้งาน ระบบเครือข่ายภายในองค์กร (On-Premises Network) และองค์ประกอบเครือข่ายต่าง ๆ บนคลาวด์ ซึ่งอาจมีอิสระอยู่บนผู้ให้บริการคลาวด์หลากหลายราย ความเชื่อมต่อเหล่านี้จำเป็นต้องมีความน่าเชื่อถือและมีความปลอดภัยสูง

อุปกรณ์เสมือน (Virtual Devices) ที่หลากหลาย เช่น ตัวกระจายโหลด (Load Balancer), เร��เตอร์ (Router), ไฟร์วอลล์ (Firewall), เครือข่ายเสมือนแบบ LAN (VLAN) และอุปกรณ์อื่น ๆ มีบทบาทในการสร้างความสามารถในการเชื่อมต่อของระบบคลาวด์ นอกจากนี้ บริการเครือข่ายที่สำคัญ เช่น การแปลงชื่อ (Name Resolution), การจัดการหมายเลข IP (IP Address Management) และการจัดการเครือข่ายผ่านซอฟต์แวร์ (Software-Based Network Management) ยังสนับสนุนการเชื่อมต่อเหล่านี้อีกด้วย

การเชื่อมต่อระบบภายในองค์กรและตำแหน่งที่ตั้งระยะไกลกับบริการคลาวด์จะต้องเริ่มต้นจากการพิจารณาประเด็นด้านความปลอดภัยและความเป็นส่วนตัว จากนั้นจึงพิจารณาประสิทธิภาพในการให้บริการองค์กรต่าง ๆ อาจใช้การเชื่อมต่อเฉพาะ (Dedicated Connections) หรือเครือข่ายส่วนตัวเสมือน (VPN)

เพื่อสร้างลิงก์แบบส่วนตัว โดยผู้ดูแลระบบยังคงใช้เครื่องมืออย่าง Secure Shell (SSH) เพื่อช่วยรักษาความมั่นคงปลอดภัยของระบบ

บพเรียนนี้จะกล่าวถึงหัวข้อข้างต้นอย่างครบถ้วน เพื่อเตรียมความพร้อมสำหรับบพเรียนถัดไปในเรื่องการแก้ไขปัญหาเครือข่ายบนคลาวด์

1.1 แนวคิดระบบเครือข่าย (Networking Concepts)

ระบบเครือข่ายบนคลาวด์มีความคล้ายคลึงกับระบบเครือข่ายทางกายภาพแบบดั้งเดิมในหลายประการ โดยข้อมูลจะไหลไปตามเส้นทางที่กำหนดไว้ล่วงหน้า การจำลององค์ประกอบเครือข่ายแบบเสมือน (Virtualized Components) ของอุปกรณ์มาตรฐาน เช่น สวิตซ์ เร��เตอร์ หรือไฟร์วอลล์ ช่วยให้ผู้ดูแลระบบสามารถควบคุมการไหลของทรัพฟิก และมั่นใจได้ว่าเฉพาะผู้ที่ได้รับอนุญาตเท่านั้นจึงจะสามารถเข้าถึงบริการเครือข่ายต่าง ๆ ได้

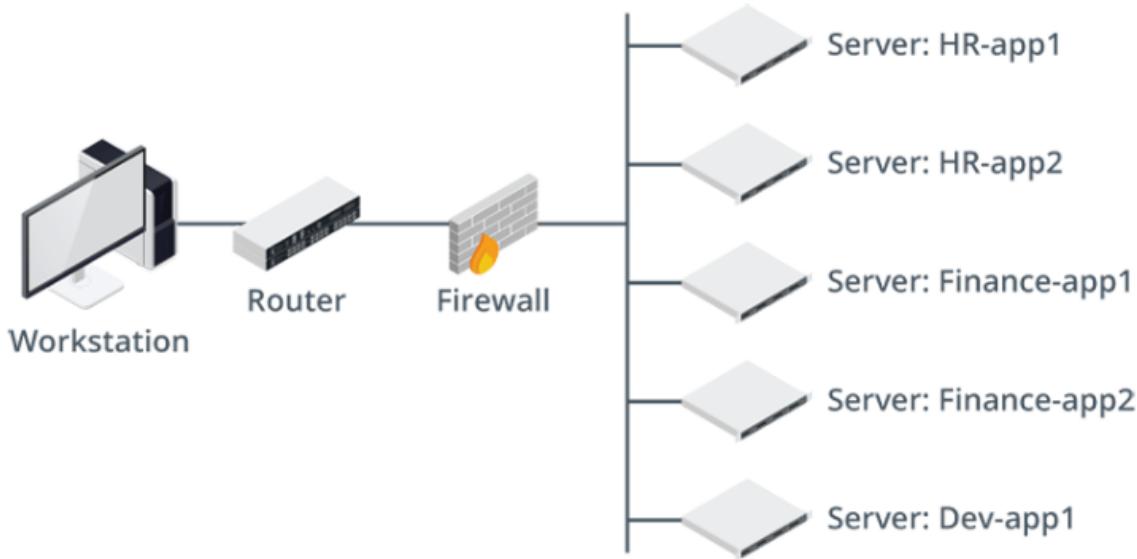
บพเรียนนี้จะกล่าวถึงประเด็นสำคัญเกี่ยวกับการเชื่อมต่อไปยังระบบคลาวด์ หน้าที่ของเครือข่าย อุปกรณ์เสมือน และบริการเครือข่ายที่เกี่ยวข้อง นอกจากนี้ยังอธิบายแนวคิดที่สำคัญของคลาวด์ เช่น:

- การกระจายโหลด (Load Balancing)
- เครือข่ายเสมือนส่วนตัว (Virtual Private Clouds: VPC)
- เครือข่ายที่กำหนดโดยซอฟต์แวร์ (Software-Defined Networking: SDN)
- เครือข่ายกระจายเนื้อหา (Content Delivery Networks: CDN)

สุดท้าย บพเรียนยังเน้นการเปรียบเทียบระหว่างการเชื่อมต่อระบบคลาวด์แบบสาธารณะ (Public Cloud Connections) และแบบส่วนตัว (Private Cloud Connections) เพื่อให้ผู้เรียนเข้าใจข้อดีข้อเสียของแต่ละรูปแบบอย่างชัดเจน

1.2 แผนภาพการไหลของเครือข่าย (Network Flow Diagram)

แผนภาพการไหลของเครือข่ายเป็นเครื่องมือที่ช่วยให้ผู้ใช้งานสามารถมองเห็นและเข้าใจได้ว่า ข้อมูล มีการเคลื่อนผ่านโครงสร้างพื้นฐานของระบบเครือข่ายอย่างไร โดยครอบคลุมถึงโหนดทั้งภายในและภายนอก อุปกรณ์เครือข่าย (เช่น เร��เตอร์) ตลอดจนบริการบนคลาวด์ต่าง ๆ



This network flow diagram displays the relationships between compute environments. Note that all environments are in a single segment. (Images © 123RF.com)

ภาพที่ 31 Network Flow Diagram

การมองเห็นองค์ประกอบเครือข่ายผ่านแผนภาพประเภทนี้มีความสำคัญอย่างยิ่งในบริบทของระบบคลาวด์ ซึ่งมักมีการผสานรวมระหว่างเครือข่ายในองค์กร (on-premises) กับระบบของผู้ให้บริการคลาวด์รายหนึ่งหรือหลายราย

แผนภาพการไหลของเครือข่ายมีประโยชน์ในหลายกรณี ได้แก่:

- การปฏิบัติตามข้อกำหนดและการตรวจสอบ (Compliance and Auditing)
- การรักษาความมั่นคงปลอดภัย (Security)
- การแก้ไขปัญหา (Troubleshooting)
- การขยายบริการ (Scaling Services)
- การออกแบบบริการเครือข่าย (Designing Network Services)
- การลดความเสี่ยง (Risk Mitigation)
- การอัปเดตเครือข่าย (Network Updates)

องค์ประกอบสำคัญที่ควร pragmat ในแผนภาพการไหลของเครือข่าย ได้แก่:

- ส่วนของเครือข่ายแบบใช้สายและไร้สาย
- ขอบเขตของเครือข่าย (เครือข่ายภายใน ภายนอก และ DMZ)
- อุปกรณ์ป้องกันภัย (เช่น เร้าเตอร์ ไฟร์วอลล์ และอุปกรณ์ควบคุมการเข้าถึงเครือข่าย หรือ NAC)
- เครือข่ายที่เชื่อมต่อได้และไม่ได้รับความเชื่อมต่อ
- การเชื่อมต่อคลาวด์แบบสาธารณะและแบบส่วนตัว
- โหนดทั้งหมดที่เชื่อมต่อกับระบบ เช่น เครื่องของผู้ใช้งาน การจัดการระยะไกล และผู้ปริโภคปลายทาง

1.3 การเชื่อมต่อระบบคลาวด์แบบสาธารณะและส่วนตัว (Public and Private Cloud Connections)

บริการระบบคลาวด์ก่อให้เกิดการเปลี่ยนแปลงที่สำคัญต่อระบบเครือข่าย โดยมีความจำเป็นต้องใช้การเชื่อมต่อที่มีความเร็วสูง มีความพร้อมใช้งานสูง และมีความปลอดภัยระหว่างเครือข่ายในองค์กร (on-premises) กับสภาพแวดล้อมของผู้ให้บริการระบบคลาวด์

การรักษาความปลอดภัยของการเชื่อมต่อนี้จำเป็นต้องอาศัยโปรโตคอลการเข้ารหัส (Encryption Protocols) ซึ่งการเชื่อมต่อประเภทนี้เรียกว่า เครือข่ายเสมือนส่วนตัว หรือ Virtual Private Network (VPN) นอกจากนี้ บางองค์กรอาจเลือกใช้การเชื่อมต่อเฉพาะทาง (dedicated connections) เพื่อเพิ่มการควบคุมและความน่าเชื่อถือ

หัวข้อถัดไปในบทเรียนนี้จะกล่าวถึงการเชื่อมต่อแบบส่วนตัวโดยใช้เทคโนโลยี VPN, โปรโตคอล SSH, IPsec และการเชื่อมต่อแบบเฉพาะ ทั้งนี้เพื่อช่วยปกป้องข้อมูลของผู้ดูแลระบบและผู้ใช้งานที่กำลังส่งผ่านระหว่างระบบในองค์กร ที่บ้าน หรือปลายทางบนคลาวด์

1.4 แนวคิดเกี่ยวกับเครือข่ายเสมือนส่วนตัว (Virtual Private Network Concepts)

ผู้ดูแลระบบ แอปพลิเคชัน และผู้ใช้งานปลายทางอาจต้องเชื่อมต่อระหว่างทรัพยากรในองค์กร กับบริการคลาวด์ ดังนั้นจึงต้องมีมาตรการป้องกันไม่ให้การเชื่อมต่อนี้ถูกดักฟังหรือดัดแปลง

การรับส่งข้อมูลผ่านเครือข่าย ไม่ว่าจะเป็นภายในระบบหรือตามเส้นทางสาธารณะ เช่น อินเทอร์เน็ต มักตกเป็นเป้าหมายได้ง่ายจากผู้ไม่หวงดี อีกทั้งการรับส่งข้อมูลตามปกติในรูปแบบ Internet Protocol (IP) มักไม่ถูกเข้ารหัส ทำให้สามารถอ่านข้อมูลได้หากถูกสกัดกั้น

ตัวอย่างสถานการณ์:

- ผู้ใช้งานทำงานจากห้องพักในโรงแรมและเชื่อมต่อกับศูนย์ข้อมูลของบริษัท หากผู้ไม่ประสงค์ดีอยู่บนเครือข่ายโรงแรมเดียวกัน อาจดักจับข้อมูล HTTP ได้ง่าย
- ผู้ดูแลระบบเข้าถึงทรัพยากรของคลาวด์ ซึ่งอาจมีข้อมูลสำคัญถูกส่งผ่านเครือข่ายระหว่างคอมพิวเตอร์ในองค์กรกับศูนย์ข้อมูลของผู้ให้บริการคลาวด์

ทั้งสองกรณีมีความเสี่ยงต่อ การดักจับแพ็กเก็ต (packet sniffing) ซึ่งเป็นเทคนิคการแอบอ่านข้อมูลที่ส่งผ่านเครือข่าย

การเข้ารหัสข้อมูลที่ส่งผ่านเครือข่าย ช่วยรักษาความลับของข้อมูลไม่ให้ถูกอ่านได้โดยง่าย เมื่อจะถูกดักจับไว้ก็ตาม

โปรโตคอล VPN tunneling ช่วยเข้ารหัสข้อมูลตั้งแต่ก่อนออกจากเครื่องต้นทาง และถอดรหัสเมื่อถึงปลายทาง ซึ่งแม้ข้อมูลจะถูกสกัดกั้นระหว่างทาง แต่ก็ไม่สามารถอ่านเนื้อหาได้อย่างง่ายดาย ตัวอย่างโปรโตคอลสำหรับการเข้ารหัสการสื่อสารทางเครือข่าย ได้แก่:

- **IPsec:** ໂປຣໂຕຄອລທ່ວ່ມ້ນຂໍ້ມູລທີ່ສາມາດເຂົ້າຮ້າສໂປຣໂຕຄອລຮະດັບແອປພລິເຄັນໄດ້ທຸກຮູບແບບ ແລະມັກໃຊ້ກັບໂຈຸລູ້ນ VPN
- **Secure Shell (SSH):** ໂປຣໂຕຄອລສໍາຫຼັບການບໍລິຫານຮະຍະໄກລທີ່ມີການປັບປຸງກັນ ມັກໃຊ້ງານໃນຮະບບ Linux ແລະ macOS ແຕ່ກີ່ສາມາດໃຊ້ງານນີ້ Windows ໄດ້ເຂັ້ນກັນ
- **L2TP/IPsec:** ໂປຣໂຕຄອລ VPN ທົ່ວໄປທີ່ໃຊ້ IPsec ໃນການເຂົ້າຮ້າສ
- **PPTP:** ໂປຣໂຕຄອລຮູ່ເກົ່າທີ່ຄວຣໂລິກເລື່ອງເນື່ອງຈາກມີໜ້ອງໂຫວ່ດ້ານຄວາມປລອດວັຍ
- **OpenVPN:** ໂອເພນໂຫວ່ດທີ່ມີຄວາມປລອດວັຍສູງ ແຕ່ມີຄວາມເຮົວຕໍ່ກວ່າຕົວອື່ນເລັກນ້ອຍ

1.5 ກາຣອອກແບບເຄື່ອງຂ່າຍເສມືອນສ່ວນຕົວ (Virtual Private Network Designs)

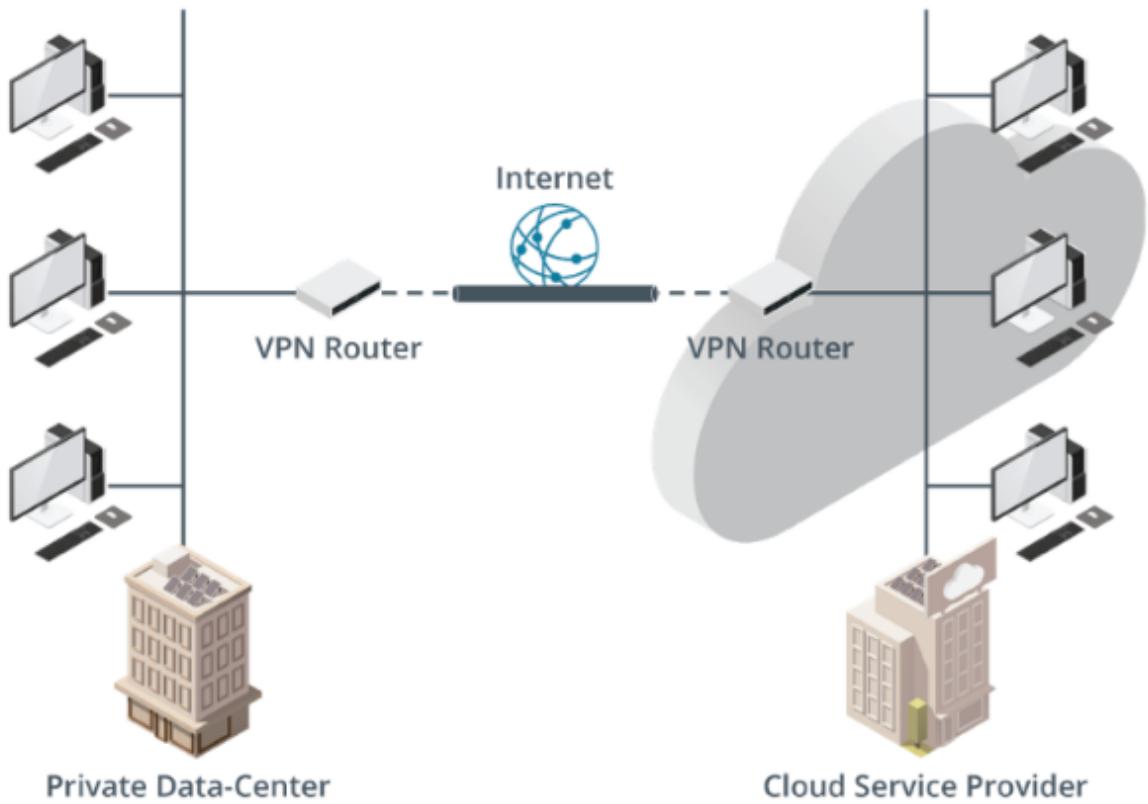
ເຄື່ອງຂ່າຍເສມືອນສ່ວນຕົວ ອ່ານວິທີ່ VPN (Virtual Private Network) ເປັນໂຈຸລູ້ນທີ່ພັບໄດ້ທົ່ວໄປໃນຮະບບ ເຄື່ອງຂ່າຍປ່າຈຸບັນ ໂດຍມີສາດາປັຕຍກຣມໜັກຍູ່ສອງຮູບແບບ ໄດ້ແກ່ VPN ແບບໄຊ໌ຕ່ອໄຫ່ (Site-to-Site VPN) ແລະ VPN ແບບຈຸດຕ່ອໄຫ່ (Point-to-Site VPN)

1.5.1 VPN ແບບໄຊ໌ຕ່ອໄຫ່ (Site-to-Site VPN)

VPN ແບບໄຊ໌ຕ່ອໄຫ່ເປັນການເຂື່ອມຕ່ອແບບ tunneled ຮະຫວ່າງສອງໄຫ່ທີ່ເປັນຮະບບເຄື່ອງຂ່າຍຂາດໃຫຍ່ ເຊັ່ນສໍານັກງານຫຼືອສູນຍົງຂໍ້ມູລ

ຕ້ວຍຢ່າງຂອງການເຂື່ອມຕ່ອແບບໄຊ໌ຕ່ອໄຫ່ ໄດ້ແກ່:

- ຈາກສໍານັກງານໃຫຍ່ໄປຍັງສໍານັກງານສາຂາ (ໃຊ້ໃນເຄື່ອງຂ່າຍອິນທຣາເນັ້ນ)
- ຈາກສູນຍົງຂໍ້ມູລເອກະນຳໃໝ່ໄປຍັງອີກແໜ່ງໜຶ່ງ (ໃນຮະບບຄລາວດໍສ່ວນຕົວ)
- ຈາກສູນຍົງຂໍ້ມູລເອກະນຳໃໝ່ຜູ້ໃຫ້ບໍລິການຮະບບຄລາວດໍສາຮາຮັນໜຶ່ງຮ່າຍຫຼືອມາກກວ່າ (ໃຊ້ໃນຮະບບ ຄລາວດໍແບບໄຂປຣິດ)



A VPN connection from the data center to a CSP. (Images © 123RF.com)

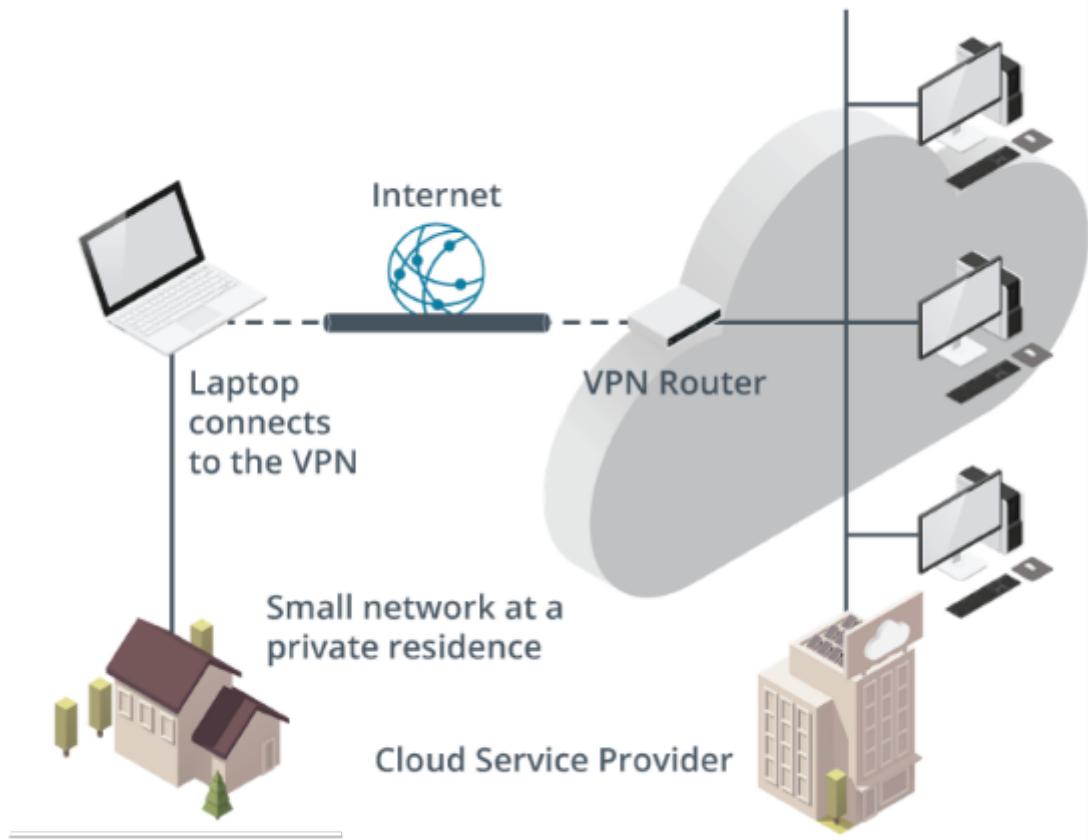
ภาพที่ 32 Site-to-Site VPN

1.5.2 VPN แบบจุดต่อไซต์ (Point-to-Site VPN)

VPN แบบจุดต่อไซต์ ซึ่งบางครั้งเรียกว่า VPN แบบเข้าถึงจากระยะไกล (Remote Access VPN) เป็นการเชื่อมต่อระหว่างอุปกรณ์เดี่ยว (เช่น คอมพิวเตอร์ของผู้ใช้งาน) ไปยังเครือข่ายขององค์กรหรือระบบคลาวด์ ในบางบริบท อาจเรียกว่า VPN แบบจุดต่อจุด (Point-to-Point VPN) เนื่องจากเป็นการเชื่อมต่อระหว่างอุปกรณ์เครือข่ายเพียงสองจุด

ตัวอย่างของ VPN แบบจุดต่อไซต์:

- เวิร์กสเตชันที่บ้านเชื่อมต่อกับศูนย์ข้อมูลขององค์กร (ใช้งานจากระยะไกล)
- เวิร์กสเตชันที่บ้านเชื่อมต่อกับบริการคลาวด์สาธารณะ (เข้าถึงคลาวด์จากระยะไกล)
- แล็ปท็อปที่ใช้งานระหว่างเดินทางเชื่อมต่อกับศูนย์ข้อมูลหรือผู้ให้บริการคลาวด์



Home-to-cloud service provider point-to-site VPN. (Images © 123RF.com)

ภาพที่ 33 Point-to-Site VPN

1.6 Secure Shell (SSH)

การเชื่อมต่อกับเครื่องเสมือน (Virtual Machines: VMs) บนระบบคลาวด์มักใช้โปรโตคอล Secure Shell (SSH) ซึ่งเป็นช่องทางการเชื่อมต่อแบบเข้ารหัสที่ให้การเข้าถึงผ่านบรรทัดคำสั่ง (command-line) ไปยังเซิร์ฟเวอร์ระยะไกล โดยมักนิยมใช้ในระบบปฏิบัติการ Linux และเนื่องจากเครื่อง VM ส่วนใหญ่ที่ให้บริการผ่านระบบคลาวด์มักติดตั้ง Linux เป็นหลัก SSH จึงกลายเป็นเครื่องมือมาตรฐานสำหรับการจัดการ VM

ลองจินตนาการถึงผู้ดูแลระบบคลาวด์ที่ทำงานจากบ้าน และต้องเชื่อมต่อเข้ากับเครื่อง VM ขององค์กรที่อยู่ตัววิ่งกับผู้ให้บริการระบบคลาวด์ เส้นทางการเชื่อมต่อนั้นมีจุดที่ความปลอดภัยถูกละเมิดได้หลายจุดดังนี้:

1. ผู้ดูแลระบบทำการล็อกอินเข้าสู่เครื่อง Linux เวิร์กสเตชันของตนเอง และร้องขอการเชื่อมต่อไปยัง VM ที่อยู่บนคลาวด์
2. การเชื่อมต่อวิ่งผ่านเครือข่ายขององค์กรไปยังผู้ให้บริการอินเทอร์เน็ต (ISP)
3. การเชื่อมต่อเดินทางผ่านโครงสร้างพื้นฐานของอินเทอร์เน็ต
4. การเชื่อมต่อเข้าสู่โครงข่ายของผู้ให้บริการคลาวด์และศูนย์ข้อมูล

5. การเชื่อมต่อเข้าสู่เครือข่ายเสมือน (VPC) และไปยังเครื่อง VM เป้าหมาย
SSH จะทำการเข้ารหัสข้อมูลทั้งหมดตั้งแต่ก่อนที่ข้อมูลจะออกจากแล็ปท็อปของผู้ดูแลระบบ
และจะถอดรหัสเมื่อข้อมูลเข้าไปยัง VM ปลายทางในระบบคลาวด์

สิ่งที่ SSH ปกป้องอย่างปลอดภัยจากต้นทางถึงปลายทาง ได้แก่:

- รหัสผ่านของผู้ดูแลระบบ
- ข้อมูลธุรกิจที่เป็นความลับ
- การสื่อสารผ่านอีเมลและช่องทางอื่น ๆ
- ไฟล์ที่ถูกโอนย้าย (file transfers)

1.7 การพิสูจน์ตัวตนแบบใช้กุญแจใน Secure Shell (SSH Key-Based Authentication)

วิธีมาตรฐานในการพิสูจน์ตัวตนต่อเซิร์ฟเวอร์ SSH ในปัจจุบันมักใช้ การพิสูจน์ตัวตนด้วยกุญแจ (Key-Based Authentication) แทนการใช้รหัสผ่านแบบดั้งเดิม การพิสูจน์ตัวตนด้วยกุญแจนี้เริ่มจาก การสร้างคู่กุญแจสาธารณะ (Public Key) และกุญแจส่วนตัว (Private Key) ซึ่งมีความสัมพันธ์ทางคณิตศาสตร์ ระหว่างกัน

เมื่ออุปกรณ์ของคุณสามารถเชื่อมต่อไปยังเซิร์ฟเวอร์ SSH ระยะไกล ระบบจะตรวจสอบว่า มีกุญแจที่ตรงกันหรือไม่ หากตรวจสอบแล้วถูกต้อง ระบบจะอนุญาตให้เข้าสู่ระบบได้ วิธีการนี้มีความปลอดภัยมากกว่าการใช้รหัสผ่าน เนื่องจากสามารถป้องกันการโจมตีแบบ brute force และการคาดเดารหัสผ่านได้ดีกว่า

ขั้นตอนพื้นฐานในการตั้งค่าการพิสูจน์ตัวตนด้วยกุญแจ SSH มีดังนี้:

- สร้างคู่กุญแจแบบสมมาตร (asymmetric key pair) บนอุปกรณ์ของคุณ

\$ ssh-keygen

- ส่งกุญแจสาธารณะไปยังเครื่อง VM ที่เป็นระบบปฏิบัติการ Linux บนคลาวด์:

\$ ssh-copy-id {remote Linux cloud VM}

- ทดสอบการเชื่อมต่อโดยการเข้าสู่ระบบระยะไกล จะสังเกตได้ว่าไม่มีการขอรหัสผ่าน เพราะระบบใช้กระบวนการแลกเปลี่ยนกุญแจแบบเงียบ (silent key exchange) ซึ่งจัดการโดยซอฟต์แวร์ SSH:

\$ ssh {remote Linux cloud VM}

- ใช้การเชื่อมต่อ SSH ที่ปลอดภัยสำหรับการทำงานทั่วไป เช่น การตั้งค่าบริการ การจัดการข้อมูล หรือการดูแลเว็บไซต์ เป็นต้น

แม้ขั้นตอนข้างต้นจะเป็นภาพรวม แต่รายละเอียดอาจแตกต่างกันขึ้นอยู่กับการตั้งค่าเฉพาะ แอปพลิเคชันที่ใช้งาน และเงื่อนไขด้านความปลอดภัย อย่างไรก็ตาม คอนโซลของผู้ให้บริการคลาวด์มักมีระบบช่วยจัดการขั้นตอนเหล่านี้ให้อัตโนมัติ และบางรายยังให้บริการจัดเก็บกุญแจ SSH และข้อมูลลับอื่น ๆ ผ่านระบบ Key Vault

1.7.1 การจัดการโคลເອນຕໍ່ SSH

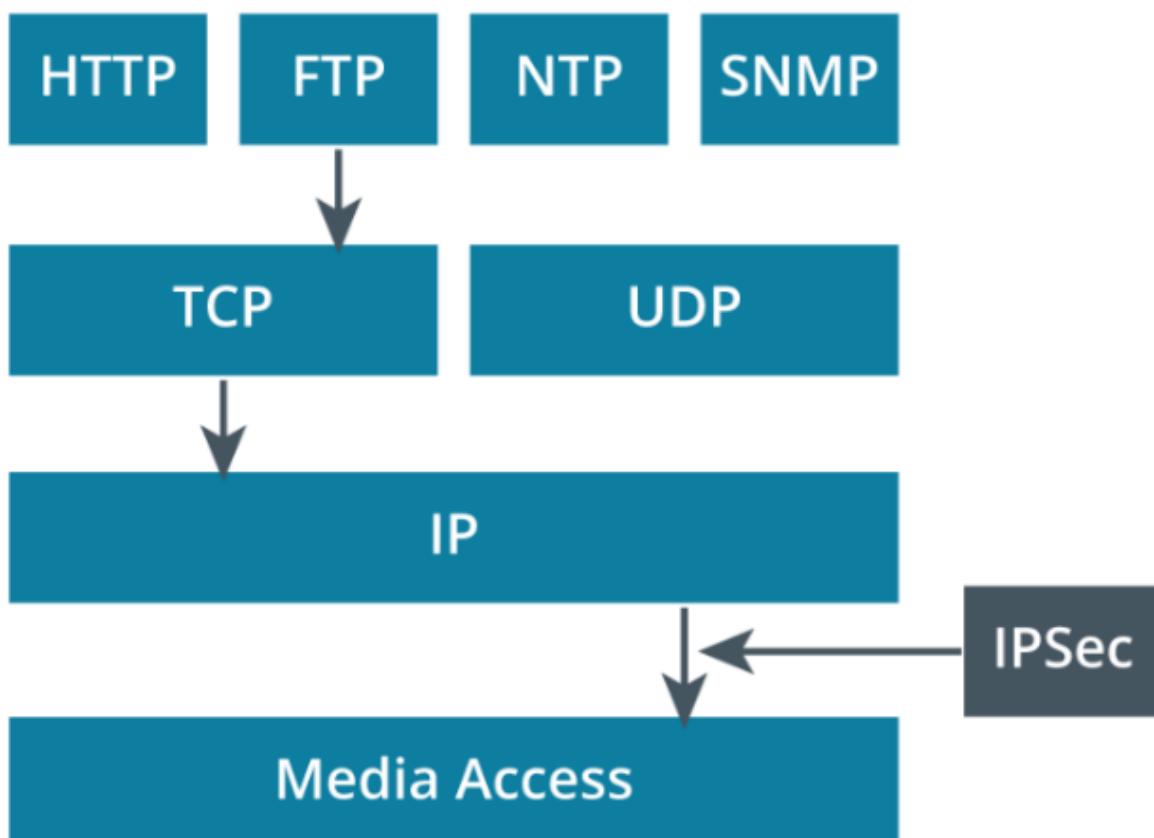
ຜູ້ດູແລະບບຄລາວດັກໃໝ່ SSH ເພື່ອເຂົ້ມຕ່ອງກັບເຄີ່ອງ VM ບນຄລາວດີສໍາຮັບການຕັ້ງຄ່າແລະບົງຫາຮະບບຣະຍະໄກລ ໂດຍເຄີ່ອງ VM ທີ່ໃຊ້ຮບບປົງປັຕິການ Linux ມັກພື້ນພາ SSH ເປັນໜັກ

ໃນຂະໜາດທີ່ ເຄີ່ອງໂຄລເອນຕໍ່ທີ່ໃໝ່ Linux ແລະ macOS ມັກພຽມໃໝ່ງານ SSH ໂດຍໄມ່ຕ້ອງຕິດຕັ້ງອະໄຣເພີ່ມເຕີມ ຮະບບ Windows ຈາກຕ້ອງມີການຕັ້ງຄ່າປີເສີ່ງ ເຊັ່ນ ການຕິດຕັ້ງໂປຣແກຣມ PuTTY ປຶ້ງເປັນໂຄລເອນຕໍ່ SSH ທີ່ນີ້ຍືມໃໝ່ກັນໃນ Windows

ນອກຈາກນີ້ ຜູ້ດູແລະບບຍັງສາມາດໃໝ່ ຄອນໂສລຂອງຜູ້ໃຫ້ບົງຫາຮະບບຄລາວດີ ເພື່ອເຂົ້າຖືງເຄີ່ອງ VM ຜ່ານ SSH ໄດ້ໂດຍຕຽງ ຊຶ່ງໜ່ວຍລົດຄວາມຢູ່ຢາກໃນການຕັ້ງຄ່າບນອຸປະກອນຂອງໂຄລເອນຕໍ່

1.8 ກາຮັກໝາຄວາມມັນຄົງປລອດກັຍດ້ວຍ IPsec (IPsec Security)

ໃນການເຂົ້າຮ້າສ້າງມູນຂອງຮະບບເຄີ່ອງຢ່າຍແບບດັ່ງເດີມ ການເຂົ້າຮ້າສ້າງດຳເນີນການໃນ **ໜັນແອປພລິເຄັ້ນ** (Application Layer) ປຶ້ງໜ່າຍຄວາມວ່າ ທັງຝ່າໄໂຄລເອນຕໍ່ແລະແອປພລິເຄັ້ນຂອງຮະບບເຄີ່ອງຢ່າຍທີ່ປ່າຍທັງສອງ ຜຶ້ງຂອງການເຂົ້ມຕ່ອງກັບການຮັບການເຂົ້າຮ້າສ້າງຮູບແບບເດືອກກັນ ປຶ້ງຈາກສ້າງຂ້ອງຈັກດ້ານຄວາມເຂົ້າກັນໄດ້ ຮະຫວ່າງແອປພລິເຄັ້ນ



The TCP/IP stack with IPsec and the FTP protocol encrypted.

ກາພທີ 34 IPsec Security

IPsec (Internet Protocol Security) เป็นวิธีการเข้ารหัสที่แตกต่าง โดยมีลักษณะเด่นคือ ทำงานในชั้น Network Layer ซึ่งเป็นชั้นเดียวกับที่จัดการเรื่องหมายเลข IP ใน TCP/IP Stack ข้อมูลทั้งหมดที่ส่งผ่านเครือข่ายต้องผ่านชั้นนี้โดยไม่ขึ้นกับแอปพลิเคชันที่ใช้งาน จึงสามารถทำการเข้ารหัสและถอดรหัสได้โดยไม่ต้องให้แอปพลิเคชันรับรู้ ส่งผลให้มีปัญหาความเข้ากันได้ของแอปพลิเคชัน

ตัวอย่างเช่น ผู้ดูแลระบบสามารถกำหนดให้ IPsec เข้ารหัสเฉพาะข้อมูลที่ส่งผ่าน FTP แต่ไม่เข้ารหัสข้อมูล SMTP ได้ ข้อมูลที่ถูกเข้ารหัสจะถูกถอดรหัสเมื่อถึงปลายทางก่อนเข้าสู่ชั้นแอปพลิเคชัน

IPsec ถูกใช้งานร่วมกับ VPN อย่างแพร่หลาย เช่นในโซลูชันที่ใช้ร่วมกับ L2TP และยังเป็นเครื่องมือสำคัญที่ผู้ดูแลระบบคลาวด์ใช้เพื่อเข้ารหัสการสื่อสารในระบบคลาวด์แบบส่วนตัว และการเชื่อมต่อ VPN ที่มีข้อมูลคลาวด์อยู่ภายใน

โครงสร้าง TCP/IP Stack ที่มีการใช้ IPsec:

- ชั้นบน: HTTP, FTP, NTP, SNMP
- ชั้นต่อมาก: TCP และ UDP
- ชั้นถัดไป: IP
- ชั้นล่างสุด: Media Access
- IPsec จะทำการเข้ารหัสระหว่าง IP กับ Media Access Layer โดยไม่ให้แอปพลิเคชันด้านบนรับรู้

1.8.1 โหมดการทำงานของ IPsec

IPsec มีรูปแบบการทำงานหลัก 2 แบบ ได้แก่:

- โหมด Transport (Transport Mode):** เข้ารหัสเฉพาะส่วน Payload ของแพ็กเก็ต โดยไม่เข้ารหัสส่วนหัว หมายความว่า การเข้ารหัสจะไม่ครอบคลุมต่อระหว่างเครื่องต่อเครื่อง (Host-to-Host) ใช้แบบดิจิตที่น้อยกว่า
- โหมด Tunnel (Tunnel Mode):** เข้ารหัสทั้งแพ็กเก็ต (ส่วนหัวและ Payload) หมายความว่า การเข้ารหัสจะครอบคลุมต่อระหว่างเราเตอร์ (Router-to-Router)

ผู้ดูแลระบบสามารถตั้งค่า IPsec ให้เข้ารหัสเฉพาะบางประเภทของการสื่อสาร หรือจะเข้ารหัสทั้งหมดก็ได้ตามนโยบายขององค์กร

1.9 การเชื่อมต่อเฉพาะทาง (Dedicated Connections)

เมื่อองค์กรย้ายระบบต่าง ๆ ไปยังคลาวด์มากขึ้น ผู้ดูแลระบบและผู้บริหารจะต้องพิจารณาผลกระทบจากการใช้โครงสร้างพื้นฐานเครือข่ายสาธารณะสำหรับวัตถุประสงค์ทางธุรกิจ

ข้อกังวลของการใช้เครือข่ายสาธารณะ (Public Internet):

- ความเร็วไม่สม่ำเสมอ
- ความน่าเชื่อถือต่ำ
- ความปลอดภัยต่ำ

- ขาดการสนับสนุนเฉพาะด้าน

แนวทางการแก้ไข

การใช้ การเชื่อมต่อเฉพาะทาง (Dedicated Connection) ระหว่างเครือข่ายภายในขององค์กร กับระบบคลาวด์ของผู้ให้บริการ ช่วยลดข้อจำกัดข้างต้น พร้อมสร้างข้อได้เปรียบดังนี้:

- ความหน่วงต่ำและประสิทธิภาพที่ดีขึ้น
- รองรับตามข้อตกลงระดับการให้บริการ (SLA)
- เพิ่มความมั่นคงปลอดภัยและการควบคุม
- สามารถรวมเข้ากับเครือข่ายชั้นที่ 2 (Layer 2) หรือชั้นที่ 3 (Layer 3) เพื่อการกำหนดเส้นทาง และการกรองข้อมูล
- รองรับแบบดีวิดท์สูง

การเชื่อมต่อเฉพาะทางมักได้รับการติดตั้งและดูแลโดยผู้ให้บริการคลาวด์ และมีการรวมค่าใช้จ่ายไว้ในใบเรียกเก็บเงินรวมกับบริการอื่น ๆ ของระบบคลาวด์

ตัวอย่างบริการการเชื่อมต่อเฉพาะทางของผู้ให้บริการคลาวด์หลัก:

- Google Cloud Platform (GCP): Dedicated Interconnect
- Amazon Web Services (AWS): Direct Connect
- Microsoft Azure: ExpressRoute

หน่วยที่ 2 พิงก์ชันเครือข่าย (Network Functions)

ในระบบคลาวด์ อุปกรณ์เครือข่ายเสมือน (Virtual Devices) เช่น ตัวกระจายโหลด (Load Balancer) และ ไฟร์วอลล์ (Firewall) มีบทบาทสำคัญในการรักษาความมั่นคงปลอดภัยและความพร้อมใช้งาน ของบริการเครือข่าย โดยทำหน้าที่เสมือนอุปกรณ์เครือข่ายจริงที่คุ้นเคยในระบบเครือข่ายภายใน (On-Premises)

Load Balancer จะทำหน้าที่กระจายปริมาณการรับส่งข้อมูลไปยังทรัพยากรในระบบคลาวด์ ตามเงื่อนไขที่กำหนด เพื่อให้การประมวลผลในแต่ละเซิร์ฟเวอร์มีความสมดุล และช่วยลดปัญหาความล่าช้า หรือประสิทธิภาพที่ตกต่ำ อีกทั้งยังเพิ่มความสามารถในการเข้าถึงบริการหรือแอปพลิเคชัน

Firewall ทำหน้าที่ควบคุมและกรองการเข้าถึงเครือข่าย โดยอาศัยกฎเกณฑ์ที่กำหนดไว้ (Rule-based Access) เพื่อป้องกันการเข้าถึงที่ไม่ได้รับอนุญาตไปยังทรัพยากรในระบบคลาวด์ ช่วยให้ระบบมีความปลอดภัย จากภัยคุกคามทางไซเบอร์

อุปกรณ์เสมือนเหล่านี้ยังสามารถสนับสนุนการทำงานเข้ากับระบบเครือข่ายภายในขององค์กร เพื่อสร้าง เป็นโซลูชันเครือข่ายแบบผสมผสาน (Hybrid Network) ที่ประกอบด้วยทั้งทรัพยากรแบบดั้งเดิม และทรัพยากรบนระบบคลาวด์

2.1 ตัวกระจายโหลด (Load Balancers)

ตัวกระจายโหลดมีหน้าที่จัดการปริมาณการรับส่งข้อมูลที่เข้าสู่อินสแตนซ์ของเครื่องเสมือน (VM) และบริการอื่น ๆ เช่น กรณีระบบที่ใช้แนวทางการปรับใช้งานแบบ Blue-Green Deployment ตัวกระจายโหลดจะถูกตั้งค่าให้ส่งข้อมูลไปยังระบบที่อยู่ในสถานะการผลิต (Production) ปัจจุบัน องค์ประกอบการตั้งค่าที่พบบ่อยของ Load Balancer:

- ชื่อ (Name)
- ประเภท (Type)
- การเชื่อมโยงกับเครือข่ายย่อย (Subnet Association)
- การเป็นสมาชิกของกลุ่มความปลอดภัย (Security Group)

เมื่อกำหนดค่าตัวกระจายโหลดเรียบร้อยแล้ว จะสามารถเชื่อมโยงอินสแตนซ์ที่เกี่ยวข้องเข้ากับตัวกระจายโหลดดังกล่าวได้

วิธีการกระจายโหลดที่ใช้กันทั่วไป

- Round Robin – แจกจ่ายปริมาณงานแบบเรียงลำดับไปยังเซิร์ฟเวอร์โดยไม่คำนึงถึงภาระงานปัจจุบัน
- Static Algorithm – แจกจ่ายปริมาณงานอย่างเท่าเทียม หมายความว่าระบบที่ปริมาณงานไม่เปลี่ยนแปลงมากนัก
- Dynamic Algorithm – มอบหมายปริมาณงานไปยังเซิร์ฟเวอร์ที่มีภาระงานน้อยที่สุด หมายความว่าระบบที่มีภาระงานเปลี่ยนแปลงตลอดเวลา

ประเภทของ Load Balancer:

- Application Load Balancer (ALB):
ทำงานที่ระดับเลเยอร์ 7 (Application Layer) ของโมเดล OSI หมายความว่ามันจัดการปริมาณงานในเว็บแอปพลิเคชัน เช่น ระบบสั่งซื้อออนไลน์ที่มีระบบจัดการสินค้า ตักร้าสินค้า และการชำระเงิน การตัดสินใจกระจายโหลดอาจขึ้นกับประเภทของเนื้อหา เช่น รูปภาพหรือไฟล์เสียง
- Network Load Balancer (NLB):
ทำงานที่ระดับเลเยอร์ 4 (Transport Layer) ใช้สำหรับจัดสรรสมดุลการเชื่อมต่อเครือข่ายผ่านอุปกรณ์เสมือน เช่น Virtual Router เพื่อรักษาประสิทธิภาพของการเชื่อมต่อ
- Gateway Load Balancer:
ออกแบบมาเพื่อปรับขนาดของอุปกรณ์เครือข่าย เช่น Firewall, Web Application Firewall (WAF), ระบบตรวจจับ/ป้องกันการบุกรุก (IDS/IPS) ตามความต้องการของเครือข่าย

2.2 ไฟร์วอลล์ระบบคลาวด์ (Cloud Firewalls)

ไฟร์วอลล์ถือเป็นองค์ประกอบสำคัญในการปกป้องเครือข่ายและโหนด (Nodes) โดยทำหน้าที่วิเคราะห์рафฟิกและใช้กฎ (Rules) ในการอนุญาตหรือปฏิเสธการเข้ามายังต่อเครือข่าย

โดยทั่วไปไฟร์วอลล์จะตั้งค่าเริ่มต้นให้ บล็อกการรับส่งข้อมูลทั้งหมด จากนั้นผู้ดูแลระบบจะระบุกฎที่อนุญาตเฉพาะบริการหรือแอปพลิเคชันที่ต้องการ เช่น อนุญาตเฉพาะอีเมลและบริการเว็บ

ประเภทของไฟร์วอลล์:

- **Stateful Firewall:** ตรวจสอบการสื่อสารแบบสมบูรณ์ เช่น การเข้ามายังต่อ TCP ลักษณะของрафฟิกและสถานะการเข้ามายังต่อ
- **Stateless Firewall:** ตรวจสอบแพ็กเกตแบบรายตัว โดยไม่จดจำสถานะการสื่อสาร ทำงานเร็วกว่าแต่มีความสามารถในการวิเคราะห์เชิงลึกน้อยกว่า

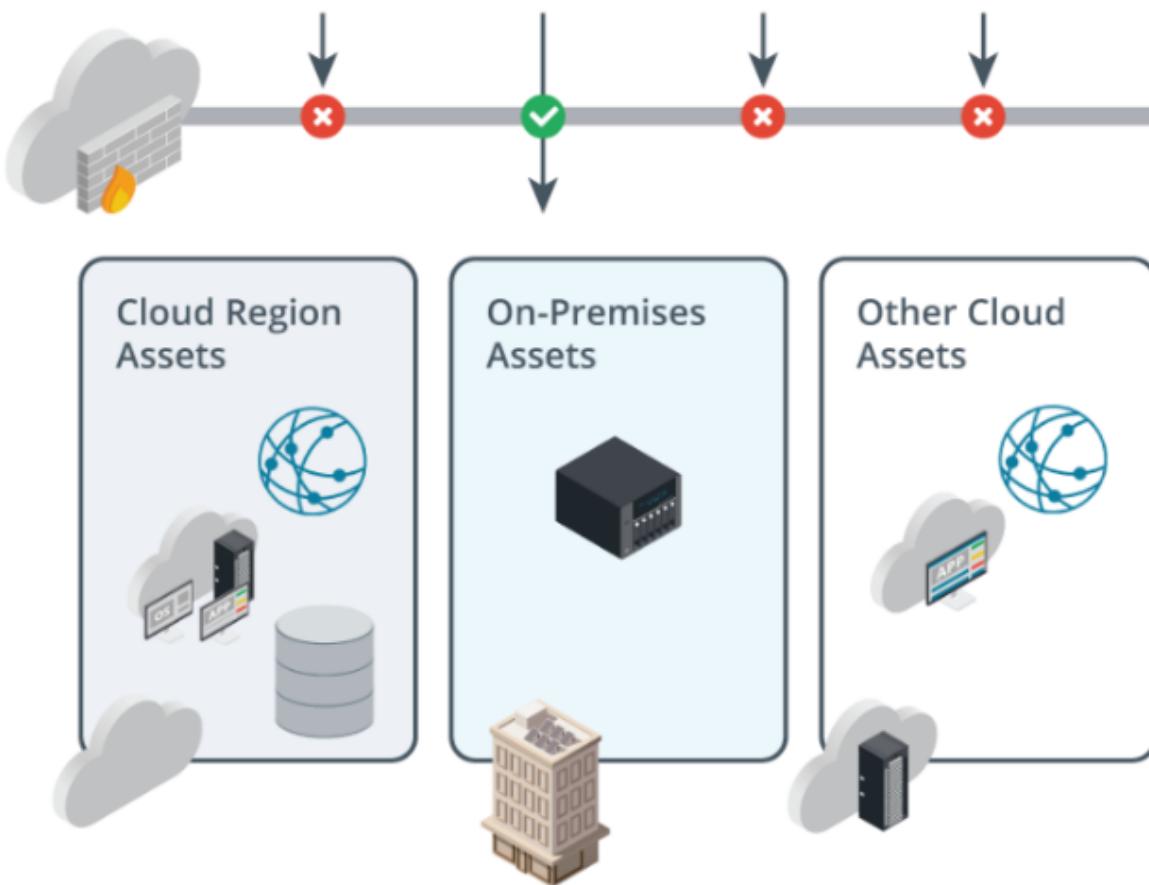
ตำแหน่งที่พบบ่อยของไฟร์วอลล์:

- เครื่องผู้ใช้งานในระบบภายใน
- เซิร์ฟเวอร์ในสถานที่ (ทั้งจริงและเสมือน)
- ขอบเขตของเครือข่ายองค์กรที่เข้ามายังต่อ กับอินเทอร์เน็ต
- จุดเข้ามายังต่อ กับผู้ให้บริการคลาวด์
- ภายใน VPC ท้องค์กรตั้งค่าเอง
- ภายในอินสแตนซ์ VM บนคลาวด์

2.3 ไฟร์วอลล์สำหรับเว็บแอปพลิเคชัน (Web Application Firewalls – WAF)

WAF ทำงานที่ระดับ เลเยอร์ 7 (Application Layer) เพื่อป้องกันช่องโหว่ที่เกิดขึ้นในระดับแอปพลิเคชัน เช่น:

- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- SQL Injection
- Distributed Denial of Service (DDoS)



A web application firewall protects a company's cloud and on-premises assets. (Images © 123RF.com)

ภาพที่ 35 Web Application Firewalls (WAFs)

WAF ทำหน้าที่เหมือนไฟร์วอลล์แบบเสมือน (Virtual Patch) ซึ่งสามารถปักป้องระบบได้แม้ยังไม่ได้อัปเดต โค้ดของแอปพลิเคชันจริง

ตัวอย่างบริการ WAF จากผู้ให้บริการคลาวด์:

- AWS WAF: ปักป้อง CloudFront, API Gateway, Load Balancers, และ AppSync โดยใช้กฎที่กำหนดเองหรือกฎสำเร็จรูปจาก AWS Marketplace
- Azure WAF: ทำงานร่วมกับ Azure Application Gateway, Front Door และ Azure CDN โดยใช้กฎจาก OWASP
- Google Cloud Armor: ใช้เกณฑ์ต่างๆ เช่น IP, ประเทศ, ขนาดของคำร้องขอ เพื่อปักป้องจาก DDoS และภัยคุกคามอื่นๆ

ตัวอย่างเกณฑ์การกำหนดกฎ (Rule Criteria):

- ที่อยู่ IP ต้นทาง
- ประเทศต้นทาง

- คำในคำขอ (Matched String)
- ขนาดของคำขอ
- โค้ดที่น่าสงสัย

การตั้งค่า WAF โดยทั่วไป:

1. เข้าสู่ระบบคอนโซลการจัดการคลาวด์
2. เลือกเมนู WAF
3. สร้างกฎ
4. ผูกกฎเข้ากับทรัพยากรที่ต้องการ
5. จัดลำดับความสำคัญของกฎ หากมีมากกว่าหนึ่ง

การใช้งาน WAF เป็นแบบจ่ายตามการใช้จริง (Pay-per-Use) โดยคิดจากจำนวนกฎและปริมาณคำขอที่ตรวจสอบ

หน่วยที่ 3 องค์ประกอบของเครือข่าย (Network Components)

ระบบเครือข่ายบนคลาวด์ (Cloud Networks) อาศัยอุปกรณ์เครือข่ายเสมือน (Virtualized Network Devices) ซึ่งเป็นการจำลองการทำงานของอุปกรณ์เครือข่ายทางกายภาพ เช่น สวิตช์ชั้นที่ 2 (Layer 2 Switch), เรตออร์ชั้นที่ 3 (Layer 3 Router) และการ์ดเชื่อมต่อเครือข่าย (Network Interface Card: NIC)

ผู้ดูแลระบบเครือข่ายบนคลาวด์มักใช้แนวทาง การแบ่งส่วนเครือข่าย (Network Segmentation) เพื่อแยกเครือข่ายออกเป็นส่วนย่อย เพื่อวัตถุประสงค์ด้านความปลอดภัยและประสิทธิภาพ เมื่อเทียบกับที่ผู้ดูแลระบบเครือข่ายในระบบกายภาพใช้

เมื่อเทคโนโลยีคลาวด์พัฒนาอย่างต่อเนื่อง องค์ประกอบของเครือข่ายจำนวนมากก็ถูกแปลงให้เป็นแบบเสมือนมากยิ่งขึ้นซึ่งรวมถึงอุปกรณ์ต่อไปนี้:

- **สวิตช์ (Switches):**
ทำงานที่ระดับชั้นที่ 2 ของโมเดล OSI โดยอาศัยที่อยู่ MAC ในการจัดการการสื่อสาร สามารถปรับแต่งการกำหนดค่าได้หลากหลาย และใช้ในการจัดการ เครือข่ายเสมือน (VLANs)
- **เซกเมนต์ (Segments):**
หมายถึงการแบ่งส่วนของเครือข่ายเพื่อเพิ่มความยืดหยุ่นในการควบคุมปริมาณข้อมูลและการจัดการ สิทธิ์การเข้าถึง อาจอยู่ในรูปแบบของ IP Subnets, VLANs, หรือ Virtual Private Clouds (VPCs)
- **เรตออร์ (Routers):**
ทำหน้าที่กำหนดเส้นทางการรับส่งข้อมูลโดยอ้างอิงที่อยู่ IP และเชื่อมโยงเครือข่ายที่มีขนาดใหญ่ หรือเครือข่ายที่แยกจากกัน

3.1 สวิตช์ (Switches)

สวิตช์ทำงานในเลเยอร์ 2 ของโมเดล OSI โดยมีหน้าที่จัดการการเชื่อมต่อและส่งต่อข้อมูลตามหมายเลข MAC address (ทั้งแบบกা�ยกาพหรือเสมือน) โดยจะทำหน้าที่กำหนดเส้นทางหรือกรองการจราจรระหว่างอุปกรณ์ต่าง ๆ

ในเครือข่ายแบบกากาพ สวิตช์จะเชื่อมต่ออุปกรณ์ในระดับอุปกรณ์ (Device-Level Connectivity) ทำหน้าที่เป็นตัวรวมสัญญาณเพื่อเชื่อมโยงระบบที่อยู่ใกล้กัน สวิตช์สามารถเชื่อมโยงกันเองได้เพื่อเพิ่มจำนวนพอร์ตหรือครอบคลุมระยะทางที่มากขึ้น

ตัวอย่าง: บริษัทที่มีสำนักงาน 3 ชั้น อาจติดตั้งสวิตช์แต่ละตัวในแต่ละชั้นเพื่อต่อเขื่อมคอมพิวเตอร์ในชั้นนั้น ๆ และใช้สายเชื่อมแนวตั้งระหว่างชั้น (เรียกว่า trunk) เพื่อเชื่อมโยงทั้งสามสวิตช์เข้าด้วยกัน ทำให้ทุกระบบสามารถเชื่อมต่อกันได้

อย่างไรก็ตาม สวิตช์จำนวนมากในปัจจุบันมีฟังก์ชันเพิ่มเติมเพื่อรองรับการแบ่งเครือข่ายอย (Segmentation) เพื่อเพิ่มความปลอดภัยและประสิทธิภาพ โดยจะทำการติดแท็กข้อมูล (Tagging) และจำกัดให้การสื่อสารเกิดขึ้นเฉพาะในเซกเมนต์ที่กำหนด เช่น การติดแท็ก VLAN ด้วยมาตรฐาน IEEE 802.1Q

3.1.1 สวิตช์เสมือน (Virtual Switches)

สวิตช์เสมือนเป็นองค์ประกอบสำคัญในสภาพแวดล้อมเครือข่ายเสมือน ซึ่งทำงานในลักษณะเดียวกับสวิตช์จริงแต่ใช้ซอฟต์แวร์ภายในไฮเปอร์ไวเซอร์ (Hypervisor) โดยจะช่วยให้เครื่องเสมือน (VMs) ที่อยู่บนไฮสต์เดียวกันหรือข้ามไฮสต์สามารถสื่อสารกันได้ร่วงกับอยู่ในเครือข่ายจริง

ในสภาพแวดล้อมที่ซับซ้อนมากขึ้น สวิตช์เสมือนอาจรวมเข้ากับสวิตช์จริงเพื่อให้การเชื่อมต่อระหว่างเซกเมนต์ของเครือข่ายจริงและเสมือนเป็นไปอย่างไรเรอยต่อ ซึ่งอาจมีไฟเบอร์เสริม เช่น VLAN, การจำลองพอร์ต (Port Mirroring) และการจัดลำดับคุณภาพบริการ (QoS) เพื่อบริหารจัดการトラฟิกอย่างมีประสิทธิภาพ

3.2 การแบ่งเครือข่ายและชับเน็ต (Network Segmentation and Subnetting)

การแบ่งเครือข่าย (Segmentation) ช่วยเพิ่มความปลอดภัยของข้อมูลในเครือข่ายโดยแยกการรับส่งข้อมูลให้อยู่ภายในส่วนต่าง ๆ ของเครือข่ายแน่นอน แนวคิดนี้ใช้ได้ทั้งกับเครือข่ายจริงและเสมือน ไม่ว่าจะเป็นระบบภายในองค์กรหรือเครือข่ายบุคคลาเวอร์

การแบ่งเครือข่ายสามารถแยกสิ่งแวดล้อมการทำงานออกจากกัน เช่น แยกส่วนพัฒนา (Development), ทดสอบ (Test) และใช้งานจริง (Production) หรือแยกตามประเภทของข้อมูล เช่น ข้อมูลทางการเงินกับข้อมูลการตลาด

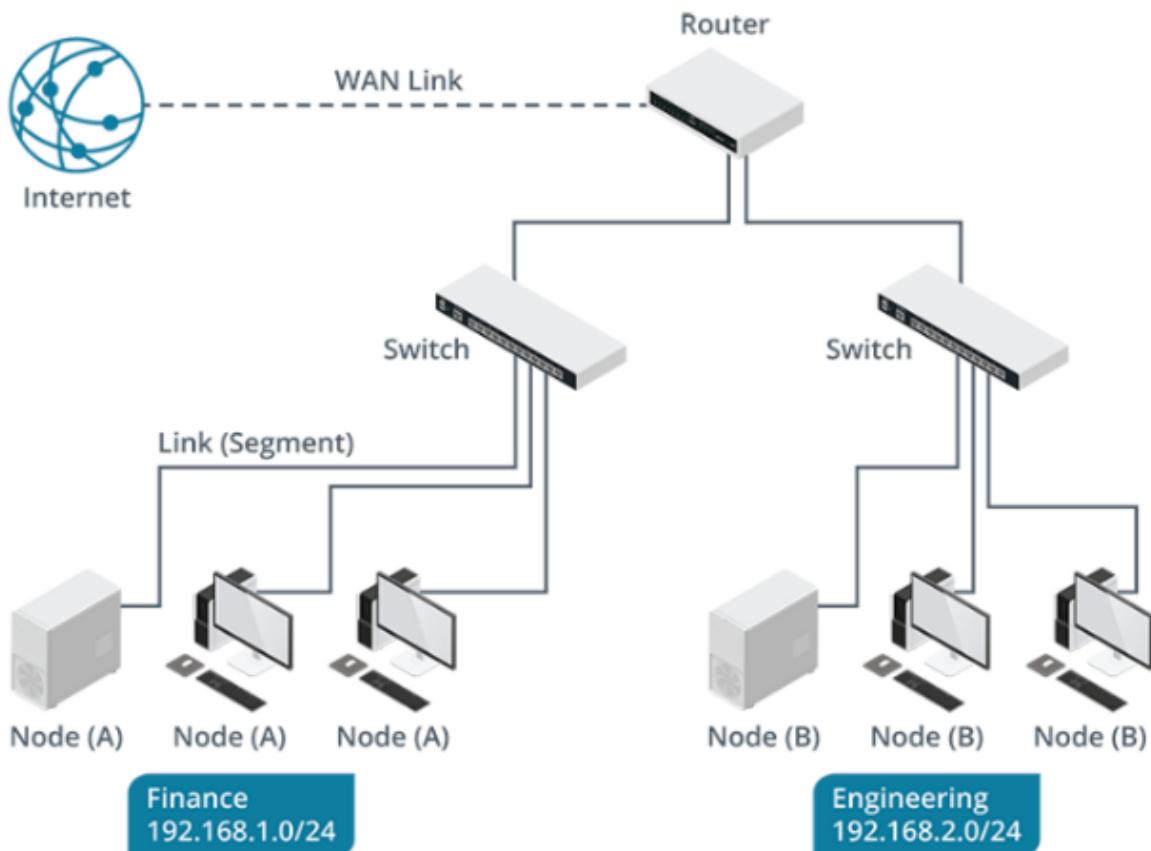
ไมโครซอฟท์ให้รูปแบบการออกแบบการแบ่งเครือข่ายพื้นฐานไว้ 3 แบบ ดังนี้:

- เครือข่ายเสมือนเดียวที่มีหลายชั้บเน็ต

- เครือข่ายเสมือนหลายเครือข่ายที่เชื่อมกันแบบเพียร์
- เครือข่ายเสมือนหลายเครือข่ายในรูปแบบศูนย์กลาง-รัศมี (Hub-and-Spoke)

3.2.1 เครือข่ายเดียวที่มีหลายชั้บเน็ต (Single Virtual Network with Subnets)

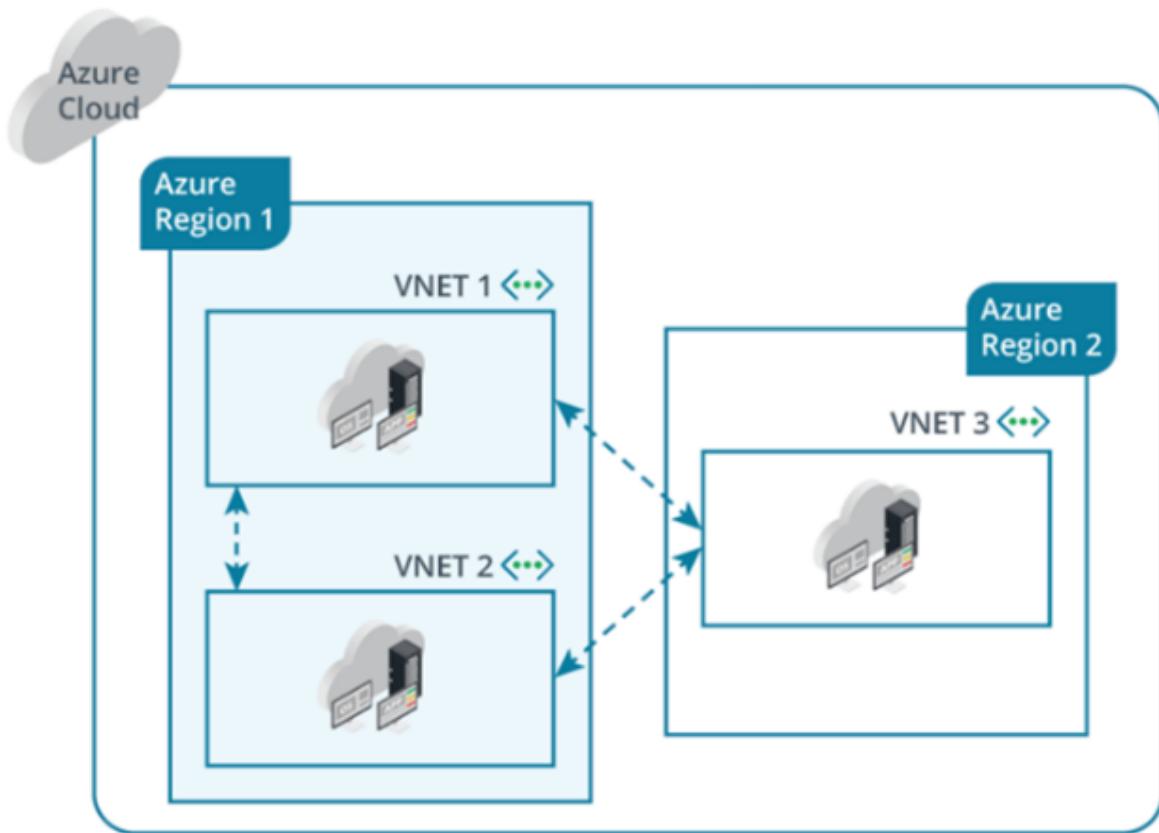
ในรูปแบบพื้นฐานนี้ ผู้ดูแลระบบคลาวด์จะจัดการเครือข่ายเสมือนเดียวที่ประกอบด้วยหนึ่ง หรือหลายชั้บเน็ต โดยแต่ละชั้บเน็ตจะมีการจำกัดการรับส่งข้อมูลเฉพาะภายในชั้บเน็ตนั้น เว้นแต่จะมีการตั้งค่าพิเศษเพิ่มเติม เช่น แบ่งเป็นแผนก “การเงิน” และ “วิศวกรรม” แยกกันในเครือข่ายเดียวกัน



ภาพที่ 36 Single Network with Subnets

3.2.2 เครือข่ายเสมือนหลายเครือข่ายแบบเพียร์ (Multiple Virtual Networks Configured as Peers)

ในสถานการณ์ที่ชั้บชั้นมากขึ้น (และพบได้จริงบ่อย) จะมีเครือข่ายเสมือนหลายเครือข่ายเชื่อมต่อกันแบบเพียร์ (peer-to-peer) และสามารถสื่อสารกันได้อย่างโปร่งใส แต่ละเครือข่ายเสมือนมีชั้บเน็ตของตนเอง และสามารถอยู่ในเขตภูมิภาค (Region) ของ Azure ที่แตกต่างกันได้

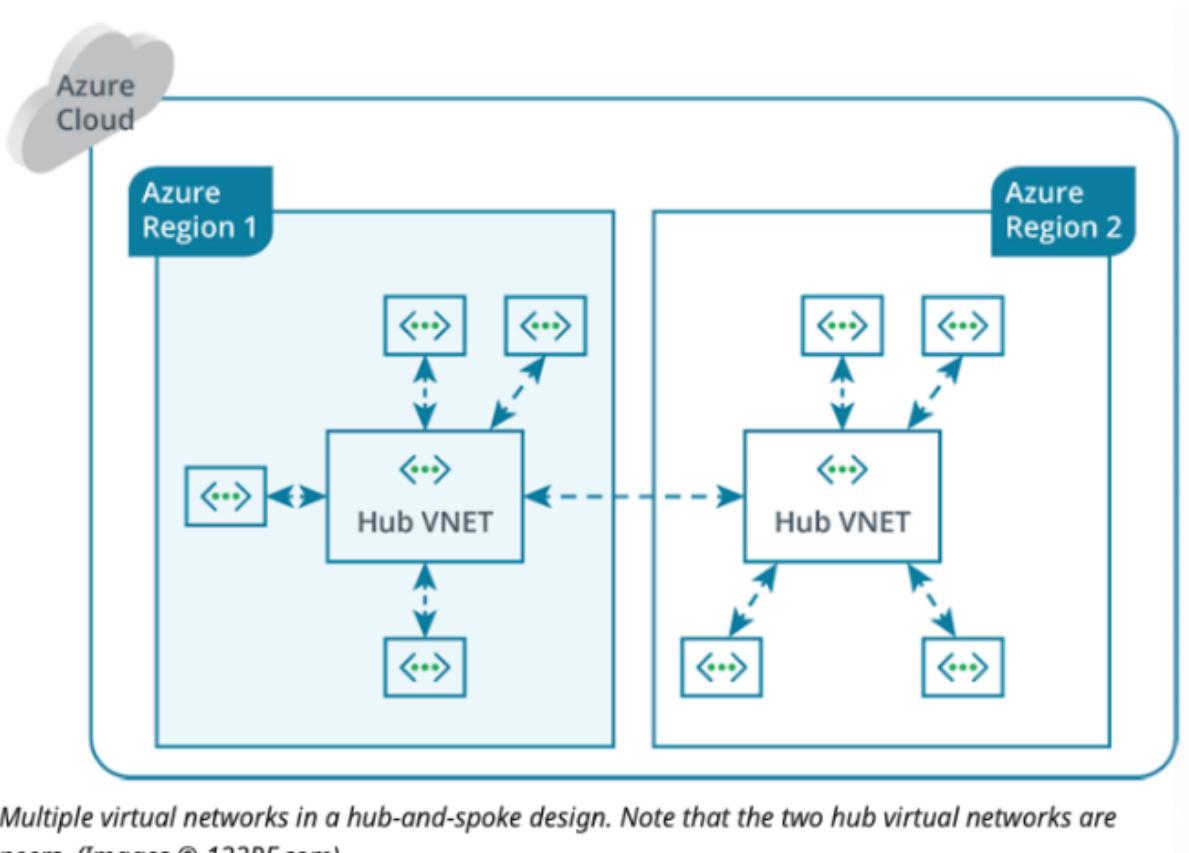


Three virtual networks configured as peers of each other. (Images © 123RF.com)

ภาพที่ 37 Multiple Virtual Networks Configured as Peers

3.2.3 เครือข่ายเสมือนแบบศูนย์กลาง-รัศมี (Multiple Virtual Networks in a Hub-and-Spoke Topology)

ในกรณีนี้ มีเครือข่ายเสมือนหลายชุดในแต่ละ Region ของ Azure และเครือข่ายภายในแต่ละ Region จะเชื่อมต่อกันผ่านเครือข่ายศูนย์กลาง (Hub VNET) พร้อมกับมีการตั้งค่าให้ Hub ของแต่ละ Region เชื่อมต่อกันเองด้วย ซึ่งช่วยให้สามารถขยายระบบเครือข่ายในแต่ละภูมิภาค และรองรับการสื่อสารข้ามภูมิภาค ได้อย่างมีประสิทธิภาพ



Multiple virtual networks in a hub-and-spoke design. Note that the two hub virtual networks are peers. (Images © 123RF.com)

ภาพที่ 38 Multiple Virtual Networks in a Hub-and-Spoke Topology

การตั้งค่าใน Azure และ AWS:

ผู้ดูแลระบบสามารถสร้างเครือข่ายเสมือนใหม่ผ่านพอร์ทัลของ Azure จากนั้นระบุชื่อชับเน็ตและระบุหมายเลขเครือข่าย (Network ID) ซึ่งกระบวนการคล้ายคลึงกันใน AWS Management Console
ชับเน็ต (Subnets)

การแบ่งเครือข่าย (Segmentation) คือการแยกเครือข่ายออกเป็นหลายส่วน โดยในเครือข่าย TCP/IP จะใช้ Network ID เพื่อแยกเครือข่ายออกจากกัน อุปกรณ์ต่าง ๆ จะมี IP Address ซึ่งบางส่วนจะระบุ Network ID เพื่อบรุ๊ว่าข้อมูลควรอยู่ในเซกเมนต์ใด

ตัวอย่าง:

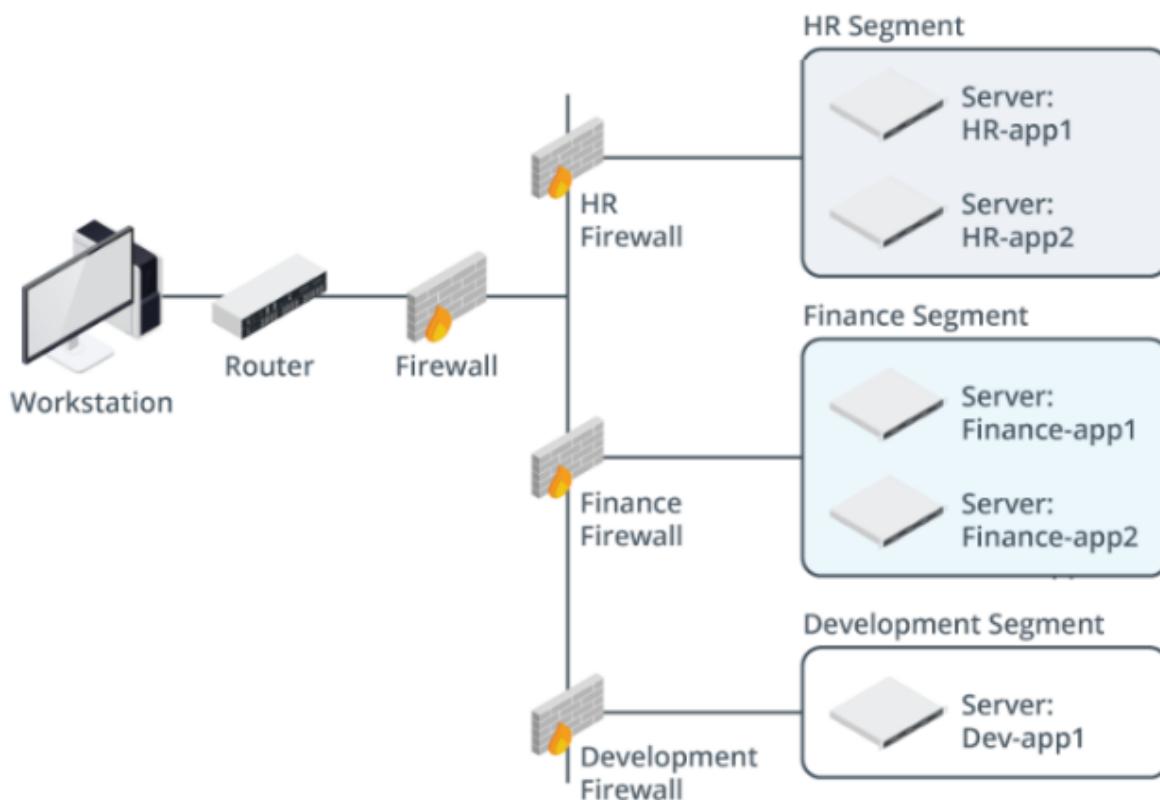
- Segment A ใช้ Network ID 192.168.1.0/24
- Segment B ใช้ Network ID 192.168.2.0/24

IP Address ลักษณะนี้ใช้ได้ทั้งในเครือข่ายจริง เครือข่ายเสมือนภายในองค์กร และเครือข่ายบนคลาวด์ เราเตอร์จะเข้มต่อเซกเมนต์เหล่านี้เข้าด้วยกัน และสามารถกำหนดให้อุปกรณ์หรือบล็อกการสื่อสารระหว่างเซกเมนต์ได้ เพื่อควบคุมความปลอดภัย ประสิทธิภาพ และการเข้มต่อ

3.3 ไมโครเซกเมนเทชัน (Microsegmentation)

การแบ่งเครือข่าย (Segmentation) จะทำในระดับเครือข่ายโดยแยกเซกเมนต์ออกจากกัน และป้องกันการเข้าถึงจากภายนอกที่ไม่ได้รับอนุญาต ซึ่งมักใช้ตัวระบุ เช่น IP Address หรือหมายเลขพอร์ต (Port Number)

การแบ่งเครือข่ายในระดับเครือข่าย (Network-level segmentation) มักใช้เพื่อแยกการรับส่งข้อมูลที่ขอบเครือข่ายหรือระหว่างลิงก์เฉพาะ (เช่น เรตเตอร์) อย่างไร้ตาม วิธีนี้ไม่สามารถควบคุมความปลอดภัยภายในแต่ละเซกเมนต์ได้โดยละเอียด ซึ่งอาจเป็นปัญหาโดยเฉพาะเมื่อแอปพลิเคชันหนึ่งมีบางส่วนทำงานในระบบภายใน (On-premises) และบางส่วนในคลาวด์



This network flow diagram displays how microsegmentation isolates various compute environments, such as HR, Finance, and Development applications. (Images © 123RF.com)

ภาพที่ 39 Microsegmentation

ไมโครเซกเมนเทชัน จึงเข้ามาเสริม โดยทำการแบ่งในระดับเวิร์กโหลด (Workload-level) แยกแอปพลิเคชัน และโครงสร้างพื้นฐานที่เกี่ยวข้องออกจากกันแบบละเอียดมากขึ้น การแบ่งนี้ทำในระดับแอปพลิเคชันแทนที่จะเป็นระดับเครือข่าย และใช้การจัดการด้วยนโยบายแบบ Software-defined networking (SDN) ทำให้มีความยืดหยุ่นและปรับขนาดได้ง่าย

องค์ประกอบสำคัญ 3 ประการของไมโครเซกเมนเทชัน ได้แก่:

- **การมองเห็น (Visibility):** ต้องสามารถมองเห็นและกำหนดค่าทั้งเวิร์กโฟล์วของแอปพลิเคชันได้ทั้งหมด

- ความละเอียด (Granularity): เวิร์กโฟล์วแต่ละชุดถูกแยกจากกันโดยสิ้นเชิง
- ความยืดหยุ่นแบบไดนามิก (Dynamic): การตั้งค่าจะเคลื่อนย้ายตามแอปพลิเคชันเมื่อมีการเปลี่ยนแปลงหรือขยายตัว

ประโยชน์ของไมโครเซ็กเมนเทชัน:

- ลดพื้นที่เสียงของการจูงดี
- เพิ่มความสามารถในการจำกัดความเสียหายจากการละเมิดความปลอดภัย
- เพิ่มความสามารถด้านการปฏิบัติตามกฎหมาย
- ง่ายต่อการบริหารจัดการด้วยนโยบาย

ตัวอย่างของไมโครเซ็กเมนเทชัน:

Amazon Web Services (AWS) ใช้ **Security Groups** เพื่อควบคุมการเข้าถึงและการจัดการตามสมาชิกซึ่งเปรียบเสมือนไฟร์wall สำหรับการรับส่งข้อมูลเข้าและออกจากอินสแตนซ์

3.4 ตารางเส้นทาง (Routing Tables)

เราเตอร์ทำงานในระดับ Layer 3 ของโมเดล OSI และทำหน้าที่ส่งต่อข้อมูลเครือข่ายไปยังเซ็กเมนต์ (subnet) ตาม Network ID ที่ฝังอยู่ใน IP Address โดยปกติแล้วเราเตอร์จะเชื่อมต่อกับแต่ละเซ็กเมนต์ผ่าน Network Interface Card (NIC) หลายตัว

ข้อมูลที่ส่งมาจากเซ็กเมนต์หนึ่งไปยังอีกเซ็กเมนต์จะเข้าเราเตอร์ทาง NIC หนึ่งและออกอีก NIC หนึ่งไปยังปลายทางตามตารางเส้นทาง

ความแตกต่างระหว่างการสวิตช์กับการเราร์ต:

- **สวิตช์ (Switching):** ใช้ MAC Address ที่ระดับ Layer 2 เพื่อเชื่อมต่ออุปกรณ์ภายในเซ็กเมนต์เดียวกัน
- **เราเตอร์ (Routing):** ใช้ IP Address ที่ระดับ Layer 3 เพื่อเชื่อมต่อระหว่างเซ็กเมนต์ต่าง ๆ

การอ่านตารางเส้นทาง (Interpreting Route Tables)

เราเตอร์จะเก็บข้อมูลว่ามี NIC ตัวใดเชื่อมกับเซ็กเมนต์ใดไว้ในตารางเส้นทาง

ในเครือข่ายที่ซับซ้อน จะมีเราเตอร์หลายตัว ซึ่งไม่ได้เชื่อมต่อโดยตรงกับทุกเซ็กเมนต์ แต่จะเชื่อมต่อกันเป็นลำดับ การส่งข้อมูลข้ามเครือข่ายเหล่านี้ต้องใช้ตารางเส้นทางเพื่อรับส่งทางลัดไปที่ใกล้จุดหมายที่สุด

การสร้างตารางเส้นทางมี 2 วิธี:

1. ตารางเส้นทางแบบกำหนดเอง (Static Routing Tables):

- ผู้ดูแลระบบต้องกรอกตารางด้วยตนเอง
- เหมาะกับเครือข่ายขนาดเล็กหรือมีโครงสร้างคงที่

○ ข้อควรระวัง:

- อาจเกิดข้อผิดพลาดจากการพิมพ์ผิดหรือเมื่อโครงสร้างเครือข่ายเปลี่ยน
- ยากต่อการปรับขยาย

2. ตารางเส้นทางแบบไดนามิก (Dynamic Routing Tables):

○ เรอาเตอร์จะดับองค์กรสามารถแก้ไขตารางเส้นทางกันได้

○ ข้อดี:

- บำรุงรักษาง่าย – ปรับปรุงอัตโนมัติ
- ขยายเครือข่ายได้ง่าย
- เหมาะสมกับเครือข่ายขนาดใหญ่ที่มีการเปลี่ยนแปลงบ่อย

○ ข้อควรระวัง:

- อาจมีปัญหาในการสื่อสารระหว่างเรอาเตอร์ หากเครือข่ายไม่เสถียร

โปรโตคอลการเราร์ต (Routing Protocols):

จะใช้ในการแลกเปลี่ยนข้อมูลตารางเส้นทางระหว่างเรอาเตอร์ในเครือข่าย ซึ่งมีหลายประเภทให้เลือกใช้งานตามความเหมาะสม

3.5 โปรโตคอล BGP (Border Gateway Protocol)

โปรโตคอลการเราร์ต (Routing Protocols) ทำหน้าที่ปรับปรุงและอัปเดตตารางเส้นทางแบบไดนามิก ตลอดโครงสร้างพื้นฐานเครือข่าย เพื่อให้สามารถปรับตัวตามสถานการณ์ต่าง ๆ เช่น การเชื่อมต่อที่ล้มเหลว ทรัพยากรใหม่ที่ถูกเพิ่มเข้ามา หรือความต้องการใช้งานเครือข่ายที่เปลี่ยนแปลง

โปรโตคอลการเราร์ตที่ใช้กันทั่วไป ได้แก่:

- OSPF (Open Shortest Path First)
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- BGP (Border Gateway Protocol)

BGP (Border Gateway Protocol) เป็นโปรโตคอลที่กำหนดการกำหนดค่าเรอาเตอร์แบบไดนามิก เพื่อแลกเปลี่ยนข้อมูลเกี่ยวกับเส้นทางที่สามารถรองรับได้ โดยใช้กลไกของ “ผู้ประกาศ” และ “ผู้รับ” (speakers และ responders) ในการส่งและรับข้อมูลการประกาศเส้นทาง (route advertisements) ซึ่งจะทำให้เรอาเตอร์แต่ละตัวอัปเดตตัวเองตามข้อมูลที่ได้รับ

BGP เลือกเส้นทางที่ดีที่สุดโดยอิงจากนโยบายเครือข่ายและเส้นทางที่มีอยู่ โดยพิจารณาจากปัจจัยต่าง ๆ เช่น น้ำหนักของเส้นทาง (weight), เส้นทาง (path), และค่าที่กำหนดเอง (custom values) อีกทั้งยังสามารถใช้กับทั้ง IPv4 และ IPv6 เพื่อให้สามารถปรับขยายและรองรับความปลอดภัยได้มากขึ้น

ระบบอิสระ (Autonomous Systems – AS)

เครือข่ายอินเทอร์เน็ตถูกแบ่งออกเป็นหลายส่วนย่อยที่เรียกว่า ระบบอิสระ (AS) ซึ่งมักจะถูกบริหารจัดการโดยองค์กรหรือผู้ให้บริการรายเดียวกัน เช่น ผู้ให้บริการอินเทอร์เน็ต (ISP) หน่วยงานรัฐ หรือองค์กรขนาดใหญ่

ข้อมูล เช่น เว็บหรืออีเมล มักต้องเดินทางข้ามจากระบบอิสระหนึ่งไปยังอีกระบบที่ส่งจากอุปกรณ์ในอังกฤษ (AS หนึ่ง) ไปยังอุปกรณ์ปลายทางในสหรัฐอเมริกา (AS อีกหนึ่งที่ควบคุมโดย ISP)

BGP จึงมีบทบาทสำคัญในการจัดการตารางเส้นทางระหว่างระบบอิสระเหล่านี้ โดยเฉพาะในรูปแบบที่เรียกว่า **external BGP (eBGP)**

ในขณะเดียวกัน BGP ยังสามารถใช้ภายในองค์กรขนาดใหญ่ ผู้ให้บริการ หรือแม้แต่ใน VPC (Virtual Private Cloud) ภายในคลาวด์ได้ โดยเรียกว่า **internal BGP (iBGP)**

ตัวอย่างบริการที่ใช้ BGP:

- Google Cloud Platform (GCP): มีบริการ Cloud Router ที่ใช้ BGP เพื่อเชื่อมต่อระบบภายใน (on-premises) เข้ากับเครือข่ายคลาวด์แบบเสมือน หรือเชื่อมต่อ VPC ภายใน GCP เข้าด้วยกัน
- AWS และ Azure ก็มีทางเลือกที่คล้ายคลึงกันในการจัดการเส้นทางเครือข่ายผ่าน BGP

3.6 การ์ดเครือข่ายเสมือน (Virtual Network Interface Cards – vNICs)

ในเซิร์ฟเวอร์แบบดั้งเดิม จะมี Network Interface Card (NIC) แบบhardtware ติดตั้งอยู่ เพื่อให้ระบบปฏิบัติการเข้าถึงเครือข่ายได้

ในระบบที่ใช้ Virtualization (การจำลองระบบ) ไฮสต์จะมี NIC แบบกายภาพหนึ่งตัวหรือมากกว่า และเครื่องเสมือน (Virtual Machine – VM) จะเข้าถึงเครือข่ายผ่าน vNIC (Virtual NIC) ซึ่งเชื่อมต่ออินสแตนซ์เข้ากับเครือข่ายและกำหนดวิธีการสื่อสารของเครื่องกับทรัพยากรอื่น

ค่าการกำหนดค่าทั่วไปของ vNIC ได้แก่:

- MAC Address
- Private IP Address และ Subnet Mask
- รองรับ IPv4 และ IPv6
- ตัวเลือกการกำหนด Public IP Address
- การเป็นสมาชิกของ Network Security Group
- การเชื่อมต่อกับ VPC หนึ่งตัวหรือมากกว่า

vNIC จะจัดเตรียมข้อมูลเครือข่ายให้กับ VM ที่กำลังทำงาน ตัวอย่างเช่น AWS Elastic Network Interface สามารถดูออกจากอินสแตนซ์หนึ่งและเชื่อมต่อกับอีกอินสแตนซ์หนึ่งได้ โดยยังคงรักษาการตั้งค่าเครือข่ายเดิมไว้

หน่วยที่ 4 บริการเครือข่าย (Network Services)

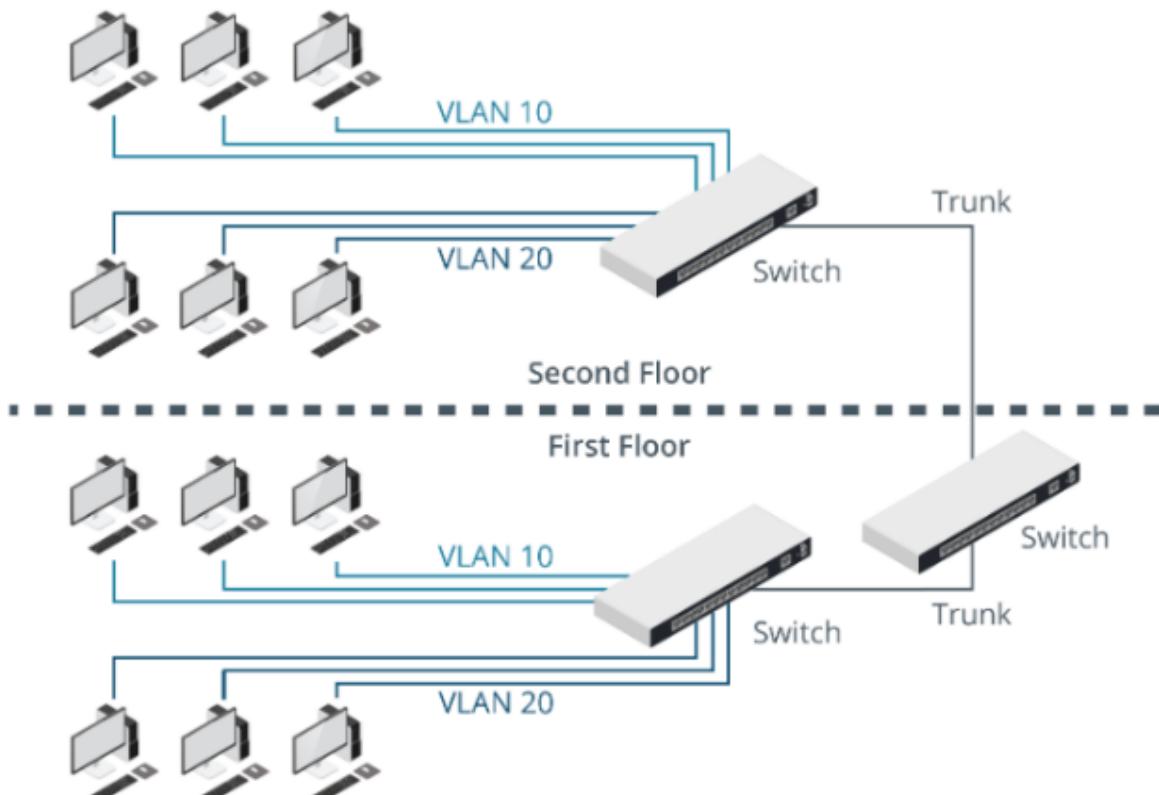
เครือข่ายบนคลาวด์ใช้ประโยชน์จากแนวคิด เครือข่ายที่กำหนดโดยซอฟต์แวร์ (Software-Defined Networking – SDN) เพื่อเพิ่มความสามารถในการปรับขยาย (Scalability) และประสิทธิภาพ (Efficiency) แนวคิดของระบบอัตโนมัติเหล่านี้เริ่มต้นจากการเข้าใจโครงสร้างของเครือข่ายคลาวด์ในส่วนต่าง ๆ เช่น เครือข่ายท้องถิ่นเสมือน (VLANs) และ ระบบเครือข่ายส่วนตัวเสมือน (VPCs)

เครือข่ายบนคลาวด์พึงพาการออกแบบโครงสร้างพื้นฐานและการปรับใช้ในหลายรูปแบบ เพื่อให้สามารถให้บริการได้อย่าง ปรับขยายได้ (scalable), เชื่อถือได้ (reliable) และ คุ้มค่า (cost-efficient)

โครงสร้างพื้นฐานนี้สามารถ ผสานเข้ากับระบบ VLAN และ Subnet ที่ใช้งานในองค์กร (on-premises) ได้อย่างราบรื่น และได้รับประโยชน์จากการจัดการผ่านระบบ SDN

4.1 เทคโนโลยีเครือข่าย VLAN และ VLAN (Virtual LAN Technologies)

การแบ่งเครือข่ายแบบพื้นฐานใช้เราเตอร์ชั้นที่ 3 (Layer 3) เพื่อเชื่อมต่อกับชั้บเน็ต IP ที่แยกจากกัน โหนดในเครือข่ายจะเชื่อมต่อกับสวิตซ์ และสามารถสื่อสารกับโหนดอื่น ๆ ที่เชื่อมต่อกับสวิตซ์เดียวกัน สวิตซ์จะเชื่อมต่อกับเราเตอร์ ซึ่งทำหน้าที่เป็นขอบเขตของชั้บเน็ต โดยเราเตอร์จะเชื่อมต่อหลายชั้บเน็ต และมีการตั้งกฎเพื่อจัดการการสื่อสารระหว่างชั้บเน็ตเหล่านั้น



Some nodes are in VLAN-10 and some are in VLAN-20. (Images © 123RF.com.)

ภาพที่ 40 Virtual LAN Technologies

อย่างไรก็ตาม การแบ่งเครือข่ายในระดับ Layer 3 มีข้อจำกัด เพราะต้องให้หน่วยโกล์กันในเชิงกายภาพ Virtual LAN (VLAN) ซึ่งขัดข้อจำกัดดังกล่าว โดยเพิ่มพอร์ตของสวิตช์ใน Layer 2 เข้าไปยัง VLAN ที่กำหนด และใช้กฎของพอร์ตเวอร์เพื่อแยกการรับส่งข้อมูลเฉพาะพอร์ตที่อยู่ใน VLAN เดียวกัน ซึ่งการแยกนี้จะเกิดขึ้นภายในสวิตช์ ไม่ใช่เราเตอร์ สวิตช์จะติดแท็กให้กับトラフィกที่ผ่านเข้ามาด้วยหมายเลขของ VLAN ผลลัพธ์คือสามารถแยกข้อมูลในแต่ละ VLAN ได้โดยไม่จำเป็นต้องสร้างเครือข่ายแยกทางกายภาพ VLAN รองรับได้สูงสุด 4094 เครือข่าย

เฉพาะอุปกรณ์ใน VLAN เดียวกันเท่านั้นที่สามารถสื่อสารกันได้โดยตรง หากต้องการสื่อสารข้าม VLAN จะต้องผ่านเราเตอร์ และสามารถใช้ไฟลเตอร์เพื่อควบคุมการเชื่อมต่อ

ตัวอย่าง

- พนักงานจากสองแผนกทำงานอยู่บนชั้น 1 และชั้น 2 บริษัทต้องการแยกトラフィกของแต่ละแผนก เพื่อความเป็นส่วนตัว
- ตัวเลือก 1: ใช้การแบ่งชั้บเน็ตแบบดั้งเดิม ต้องย้ายพนักงานทั้งแผนกไปอยู่ชั้นเดียวกัน ซึ่งไม่สะดวก
- ตัวเลือก 2: ใช้ VLAN แต่ละพอร์ตของสวิตช์จะถูกกำหนดให้อยู่ใน VLAN-10 หรือ VLAN-20 ซึ่งช่วยให้พนักงานไม่ต้องย้ายที่นั่ง และยังได้แยกトラフィกของแต่ละแผนกตามต้องการ VLAN ยังสามารถขยายไปยังทรัพยากรบนคลาวด์ได้ เช่น การกำหนดให้ VM ในคลาวด์อยู่ใน VLAN เดียวกันกับระบบภายในองค์กร ซึ่งอาจต้องใช้โปรโตคอลและการกำหนดค่าที่ซับซ้อนเพิ่มเติม

4.2 การตั้งค่า VXLAN (Virtual Extensible LAN Settings)

VXLAN มีความคล้ายกับ VLAN แต่แก้ปัญหาข้อจำกัดของ VLAN ได้ดีกว่า เช่น:

- รองรับเชิงเมต์เครือข่ายได้ ~16 ล้าน (VLAN รองรับ 4094)
- รองรับการ tunneling ระหว่าง VXLAN
- ใช้ประโยชน์จากการรวมลิงก์และ Layer 3 routing ได้ดีขึ้น
- สามารถรวมระบบในองค์กรกับคลาวด์ได้

ตัวอย่าง VXLAN:

- เชื่อมต่อเครือข่ายเดิมในองค์กรเข้ากับเครือข่ายในคลาวด์ โดยให้ทั้งสองฝ่ายใช้ IP เดิมได้ผ่าน tunnel แบบสองทาง (เช่น Azure ExpressRoute หรือ Site-to-Site VPN)

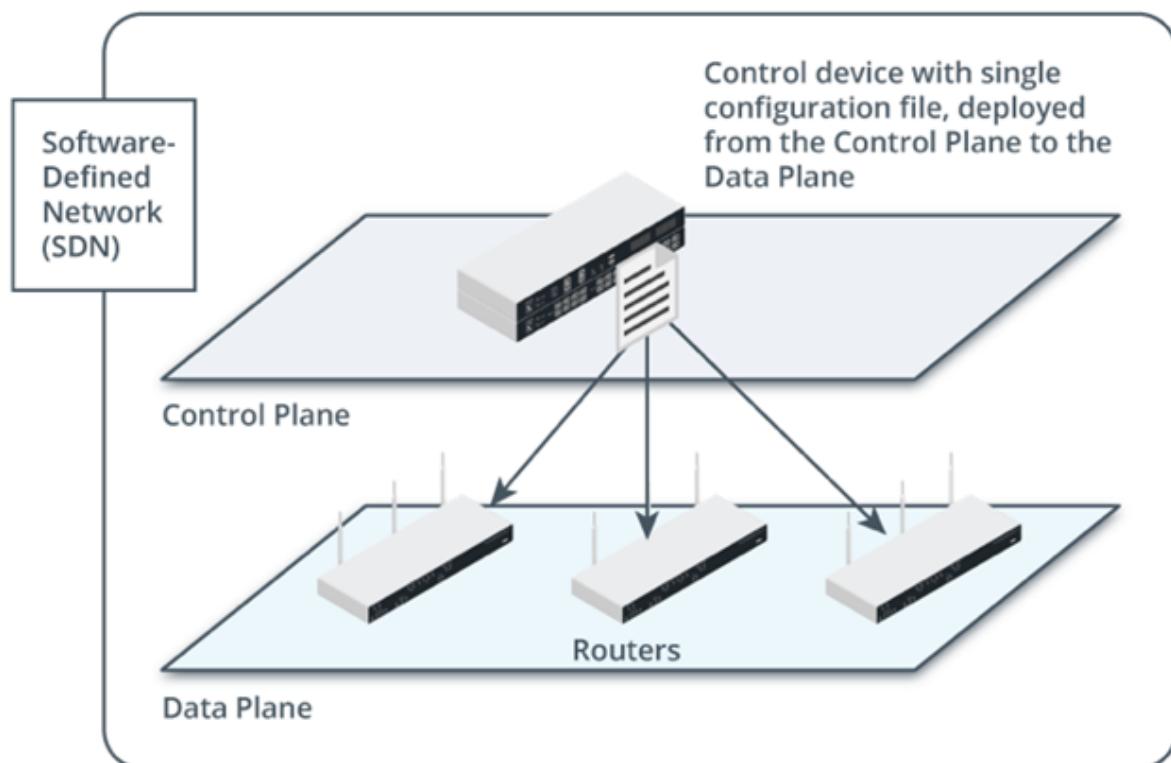
VXLAN ยังสามารถใช้ในกรณี การทำ Traffic Mirroring เช่น เพื่อส่งข้อมูลไปยังระบบวิเคราะห์ภัยคุกคาม หรือเครื่องมือวิเคราะห์เครือข่าย

ทางเลือกอื่นนอกจาก VXLAN:

- Microsoft ใช้ NVGRE
- อีกทางเลือกคือ STT
- ทั้ง 3 แบบไม่สามารถใช้ร่วมกันได้ ซึ่งเป็นปัญหาใน Multi-Cloud
- GENEVE เป็นมาตรฐานกลางที่ใช้กำหนดรูปแบบข้อมูล และสามารถใช้ร่วมกับทั้ง 3 แบบได้

4.3 เครือข่ายที่กำหนดโดยซอฟต์แวร์ (Software-Defined Networking - SDN)

เครือข่ายแบบดั้งเดิมมีการกระจายตัว ผู้ดูแลระบบต้องกำหนดค่าแต่ละสวิตช์หรือเราเตอร์แยกกัน การอัปเดตหรือสำรองข้อมูลทำได้ยาก



Data plane devices are managed by a control plane device. (Images © 123RF.com)

ภาพที่ 41 Software-Defined Networking

SDN แก้ปัญหาโดย:

- แยกการจัดการออกเป็น 2 ส่วน:
 - Data Plane: ควบคุมการส่งข้อมูล เช่น พิลเตอร์หรือส่งต่อ packet
 - Control Plane: จัดการคำสั่งควบคุมและนโยบายผ่าน software จากส่วนกลาง

Controller คือศูนย์กลางของ SDN ทำให้สามารถควบคุมสวิตช์, เราเตอร์, และ load balancer จากจุดเดียวได้โดยใช้ นโยบาย (policy) และการจัดการอัตโนมัติ

ข้อดีของ SDN:

- เห็นภาพรวมทั้งระบบ
- ลดภาระการจัดการแบบทีละอุปกรณ์
- เข้ากับโครงสร้างแบบ Virtualization
- ใช้ได้ทั้งระบบคลาวด์, hybrid, และ on-premises

เทคโนโลยี SDN ใน Microsoft Azure:

- **Network Controller:** ศูนย์จัดการเครือข่ายทั้งในองค์กรและบนคลาวด์ โดยใช้ API เชื่อมต่อกับส่วนต่าง ๆ
- **Network Function Virtualization:** เปลี่ยนบริการเครือข่ายให้เป็น VM เช่น firewall, NAT, load balancer
- **Virtual Switches:** จัดการด้านความปลอดภัยและการให้บริการ

4.4 คลาวด์ส่วนตัวเสมือน (Virtual Private Clouds - VPCs)

องค์กรจำนวนมากมีความกังวลเกี่ยวกับความปลอดภัยของข้อมูลและบริการที่โฮสต์อยู่ในระบบคลาวด์สาธารณะ วิธีหนึ่งในการจัดการกับความกังวลนี้ คือ การสร้างศูนย์ข้อมูลของตนเองและโฮสต์ระบบคลาวด์ส่วนตัว (Private Cloud) อย่างไรก็ตาม วิธีนี้มีค่าใช้จ่ายสูงและซับซ้อน

VPC เป็นทางเลือกที่ช่วยให้องค์กรสามารถมีความมั่นคงปลอดภัยยิ่งขึ้นในระบบคลาวด์สาธารณะ โดยที่นำไปใช้บริการคลาวด์แบบโครงสร้างพื้นฐานเป็นบริการ (Infrastructure as a Service – IaaS) จะมีการแยกข้อมูลลูกค้าโดยใช้ชับเน็ต (Subnets) และ VLAN แต่ VPC จะเพิ่มระดับการแยกและความเป็นส่วนตัวโดยจำลองโครงสร้างแบบ “ผู้เช่าเดียว” (Single-Tenant) ภายในสภาพแวดล้อมที่มีผู้เช่าหลายราย (Multi-Tenant) ของผู้ให้บริการคลาวด์ (Cloud Service Provider – CSP)

VPC ยังสามารถเพิ่มการเชื่อมต่อผ่านเครือข่ายส่วนตัวเสมือน (VPN) และใช้การควบคุมผ่าน SDN เพื่อจัดการโครงสร้างเครือข่ายได้อย่างปลอดภัย

ภายใน VPC องค์กรสามารถ:

- สร้างชับเน็ตและ VLAN
- ปรับใช้ VM, เว็บแอปพลิเคชัน และบริการอื่น ๆ
- ใช้ประโยชน์จากคุณสมบัติของคลาวด์ เช่น ความยืดหยุ่น, จ่ายตามการใช้งาน, ความพร้อมใช้งานสูง และความสะดวก

ขณะที่ยังคงรักษาความเป็นส่วนตัวของข้อมูล

องค์ประกอบใน SDN สำหรับสร้าง VPC อาจรวมถึง:

- Subnets
- VLANs
- VPNs
- Instances
- Web applications

4.5 การออกแบบ VPC แบบ Hub-and-Spoke

โครงสร้าง VPC แบบ Hub-and-Spoke ใช้เครือข่ายศูนย์กลาง (Hub) เพื่อประสิทธิภาพการส่วนกลาง เช่น Microsoft Entra ID (หรือ Azure AD เดิม) และระบบ DNS เครือข่าย Spoke จะเชื่อมต่อกับ Hub แต่ไม่เชื่อมต่อถึงกันเอง ทำให้สามารถควบคุมทรัพยากรจากศูนย์กลางได้

ตัวอย่างกรณีการใช้งาน:

- ต้องแยกการทำงานระหว่างระบบพัฒนา, ทดสอบ และระบบผลิตจริง
- มีบริการส่วนกลาง เช่น Active Directory และ DNS ที่ต้องใช้ร่วมกัน
- ต้องการใช้ทรัพยากรจำนวนมากเกินกว่าที่ Subscription เดียวรองรับ
- ต้องมีการมอบหมายสิทธิ์แยกตามแต่ละ Spoke
- ต้องมีการบริหารจัดการและวิเคราะห์ทรัพยากรแบบรวมศูนย์

VPC Peering

การ Peering เป็นการเชื่อมต่อเครือข่ายระหว่าง VPC สองชุด เช่น ในโครงสร้างแบบ Hub-and-Spoke โดยเครือข่ายเสมือนจะปราศเป็นเครือข่ายเดียวแก่ผู้ใช้งาน

ประโยชน์ของ Peering:

- เชื่อมต่อรวดเร็วระหว่าง VPC
- ใช้งานทรัพยากรร่วมกันได้
- ไม่ต้องตั้งค่าอุปกรณ์เสมือนเพิ่มเติม
- ใช้เพื่อเข้าถึงข้อมูลในคลาวด์ภายนอกบัญชีผู้ใช้อื่น

หมายเหตุ: Spoke VPC จะ Peering เฉพาะกับ Hub VPC เท่านั้น ไม่เชื่อมกับในรูปแบบ Hub-and-Spoke

4.6 การแปลงที่อยู่เครือข่าย (Network Address Translation - NAT) ใน VPC

NAT เป็นเทคนิคที่ใช้เพื่อเพิ่มประสิทธิภาพ การตรวจสอบ และความปลอดภัยของเครือข่าย โดยหลักการคือ การแมปที่อยู่ IP ภายในให้เชื่อมต่อกับที่อยู่ IP สาธารณะ ทำให้สามารถสื่อสารกับอินเทอร์เน็ต โดยไม่เปิดเผยที่อยู่ภายนอก

Client NAT

ในระบบองค์กร ตัวอย่างเช่น นักพัฒนาซอฟต์แวร์ใช้งานเครื่องภายนอกที่มี IP ส่วนตัว เมื่อเข้าถึง เว็บไซต์ภายนอก NAT จะแมปการเชื่อมต่อไปยัง IP สาธารณะขององค์กร ทำให้ชื่อนี้ข้อมูลของเครื่องภายนอก จำกบุคคลภายนอกและภัยคุกคาม

Cloud NAT

แนวคิดเดียวกันนำมาใช้ในระบบคลาวด์ โดย VM ภายนอก VPC สามารถเข้าถึงอินเทอร์เน็ตผ่าน NAT ได้ ซึ่งครอบคลุมทั้ง Source NAT (SNAT) และ Destination NAT (DNAT)

กรณีการใช้งาน Cloud NAT:

- การเข้าถึง Content Delivery Networks (CDNs)

- การจำลองข้อมูลระหว่าง VPC ใน Availability Zones ต่าง ๆ
- การเข้าถึงทรัพยากรสำหรับ AI/ML

บริการ NAT จากผู้ให้บริการคลาวด์ ได้แก่:

- AWS: NAT Gateways
- Azure: NAT Gateway
- GCP: Cloud NAT

4.7 เกตเวย์ทرانซิตในระบบคลาวด์ (Transit Gateway in the Cloud)

เมื่อมีการขยายการใช้งานระบบคลาวด์ในระดับองค์กร เกตเวย์ทرانซิต (Transit Gateway) จะทำหน้าที่เป็นศูนย์กลางในการจัดการและควบคุมการรับส่งข้อมูลและบริการระหว่างพื้นที่เครือข่ายต่าง ๆ โดยเกตเวย์นี้จะใช้กฎ (Rules) เพื่อกำหนดเส้นทางที่ข้อมูลเครือข่ายควรเดินทางไปยังจุดหมายปลายทางที่ถูกต้อง คล้ายกับการทำงานของเรตออร์ (Router)

ตัวอย่างของการเชื่อมต่อผ่าน Transit Gateway ได้แก่:

- Virtual Private Clouds (VPCs)
- เครือข่ายภายในองค์กรที่เชื่อมต่อผ่านเครือข่ายส่วนตัวเสมือน (VPN)
- เครือข่ายภายในองค์กรที่เชื่อมต่อผ่านสายเชื่อมเฉพาะ (Dedicated Connections)
- การเชื่อมต่อผ่าน Software-Defined WAN (SD-WAN)

ผู้ให้บริการคลาวด์ (Cloud Service Providers – CSPs) เช่น Amazon Web Services (AWS) จะเรียกเก็บค่าบริการสำหรับการใช้งาน Transit Gateway แบบรายชั่วโมง

ผู้ดูแลระบบสามารถเข้าถึงการกำหนดค่าของ Transit Gateway ได้หลายวิธี เช่น ผ่านคอนโซลของระบบคลาวด์, การเชื่อมต่อผ่าน Common Language Infrastructure (CLI), การใช้ Application Programming Interface (API) หรือวิธีการอื่น

4.8 เครือข่ายส่งเนื้อหา (Content Delivery Networks – CDNs)

เครือข่ายส่งเนื้อหา หรือ CDN เป็นโครงสร้างเครือข่ายแบบกระจาย (Distributed Network) ที่มีมาตั้งแต่ช่วงทศวรรษ 1990 โดยมีวัตถุประสงค์เพื่อเพิ่มความพร้อมใช้งาน (Availability) และประสิทธิภาพ (Performance) โดยใช้การแคช (Cache) เนื้อหาหรือบริการให้อยู่ใกล้กับผู้ใช้งานมากที่สุด



Content Delivery Network with data centers around the world. (Images © 123RF.com.)

ภาพที่ 42 Content Delivery Network

CDN จะทำหน้าที่:

- จัดเก็บข้อมูลหรือบริการที่มีการเรียกใช้งานบ่อยให้อยู่ในภูมิภาคใกล้ผู้ใช้
- รองรับข้อมูลประเภทเว็บแอปพลิเคชัน, ซอฟต์แวร์แบบ SaaS, เว็บไซต์, สคริปต์, ไฟล์ที่สามารถดาวน์โหลด และอื่น ๆ

ประโยชน์ของ CDN ได้แก่:

- ลดเวลาในการโหลดเว็บไซต์และบริการอื่น ๆ
- ลดความหน่วงเวลา (Latency) และการใช้แบนด์วิดท์
- เพิ่มความสามารถในการปรับขยายระบบ (Scalability) และรองรับการกระจายโหลด
- เพิ่มความพร้อมใช้งานด้วยระบบ Failover อัตโนมัติ
- เพิ่มความเร็วในการเข้าถึงข้อมูลและแอปพลิเคชัน
- รองรับการใช้งานแอปพลิเคชันที่ไม่ทนต่อความหน่วงได้ดียิ่งขึ้น

ตัวอย่าง:

สมมุติว่าองค์กรของคุณขายสินค้าออนไลน์ผ่านเว็บไซต์ที่ประกอบด้วยวิดีโอสาธิตการใช้งานสินค้า, ไฟล์ PDF คู่มือการใช้งาน และระบบสั่งซื้อสินค้า หากเว็บไซต์นั้นโอสถอยู่เพียงในสหรัฐอเมริกาฝั่งตะวันตก ลูกค้าจากประเทศอินเดีย ญี่ปุ่น หรืออสเตรเลีย อาจพบว่าเว็บไซต์โหลดช้า และเลือกซื้อจากที่อื่นแทน

แต่หากใช้ CDN ระบบจะทำการแคชเนื้อหาเว็บไซต์ไว้ที่เซิร์ฟเวอร์ในภูมิภาคเอเชียแปซิฟิก ทำให้การเข้าถึงเร็วขึ้นและลดภาระบนเว็บเซิร์ฟเวอร์หลัก อีกทั้งยังเพิ่มความทนทานในกรณีที่ศูนย์ข้อมูลหลักไม่สามารถให้บริการได้

CDN จะมีโหนดที่เรียกว่า Point of Presence (PoP) ซึ่งเป็นจุดแคชข้อมูลที่ผู้ให้บริการคลาวด์แล

ผู้ให้บริการคลาวด์รายใหญ่ที่มีโซลูชัน CDN ได้แก่:

- Microsoft Azure: Azure CDN
- Google Cloud Platform (GCP): Cloud CDN
- Amazon Web Services (AWS): CloudFront

หน่วยที่ 5 บริการเครือข่ายในระบบคลาวด์ (Cloud Network Services)

การบริหารจัดการบริการเครือข่ายในระบบคลาวด์จำเป็นต้องใช้แนวทางและแนวคิดเช่นเดียวกับการบริหารจัดการอุปกรณ์เครือข่ายแบบกายภาพในสถานที่ (On-premises Physical Network components) ซึ่งบริการเหล่านี้รวมถึง การจัดการที่อยู่ IP (IP Address Management), การแปลงชื่อโดเมน (Name Resolution), การกำหนดเวลาระบบ (Time Services) และการเลือกใช้โปรโตคอล (Protocol Selection)

เครือข่ายเสมือนในระบบคลาวด์สามารถกำหนดค่าให้ให้บริการเครือข่ายเหล่านี้ได้โดยตรง หรือสามารถสนับสนุนกับบริการเครือข่ายที่มีอยู่เดิมภายในองค์กรได้ บริการต่าง ๆ

5.1 การแปลงชื่อโดเมน (Name Resolution)

ระบบ Domain Name System (DNS) ให้บริการแปลงชื่อ (Name Resolution) เพื่ออำนวยความสะดวกแก่ผู้ใช้ในการเรียกใช้งานทรัพยากรบนเครือข่าย โดยแปลงชื่อที่เข้าใจง่ายสำหรับมนุษย์ (เช่น www.example.com) ให้เป็นที่อยู่ IP ที่ใช้ในเครือข่าย (เช่น 192.0.2.1) ซึ่งเป็นที่เข้าใจของระบบเครือข่าย

ในโครงสร้างพื้นฐานแบบดั้งเดิม (on-premises) องค์กรมักใช้ DNS server เพื่อจัดเก็บและจัดการฐานข้อมูลทรัพยากรที่มีชื่อและที่อยู่ IP เมื่อผู้ใช้พิมพ์ชื่อทรัพยากร เช่น ชื่อเว็บไซต์หรือเครื่องแม่ข่าย ระบบจะส่งคำขอไปยัง DNS เพื่อดึงที่อยู่ IP และเชื่อมต่อกับปลายทางตามที่อยู่ IP นั้น

ในระบบคลาวด์ ความสามารถนี้ยังคงจำเป็น ตัวอย่างเช่น เมื่อผู้ใช้งานต้องการเชื่อมต่อกับเซิร์ฟเวอร์ Linux (RHEL) ที่ติดตั้งใน AWS ผู้ใช้จะใช้ชื่อเครื่อง (hostname) เช่น rhel-server-7 แต่การเชื่อมต่อจริงเกิดขึ้นผ่านที่อยู่ IP เช่น 10.20.5.99 จึงต้องอาศัยบริการ DNS เสมอ

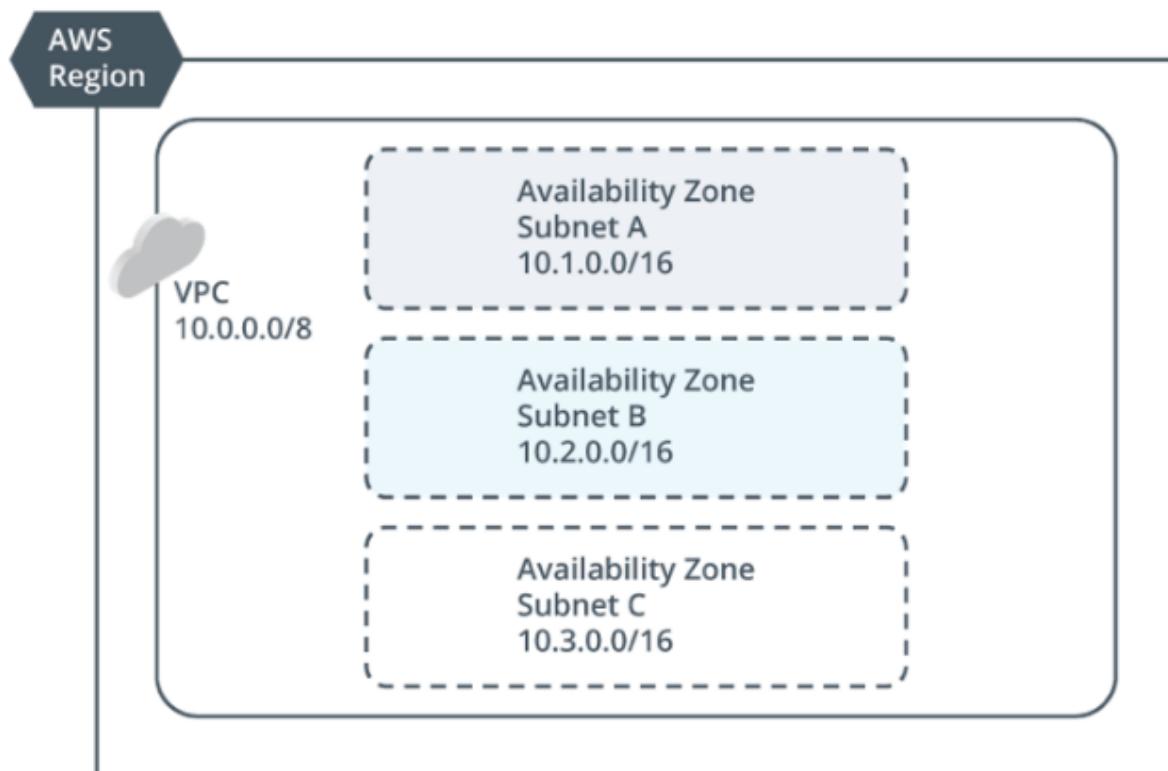
ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers - CSPs) เช่น AWS, Azure, และ GCP ต่างมีบริการ DNS ของตน เช่น:

- Google Cloud DNS มีคุณสมบัติ เช่น Zone Forwarding, DNSSEC, DNS Peering
- AWS Route 53 รองรับการลงทะเบียนโดเมน การแปลงชื่อ และตรวจสอบสถานะ
- Azure DNS จัดการผ่าน Azure Portal

การรักษาความมั่นคงปลอดภัยของ DNS: DNS ดั้งเดิมไม่มีการเข้ารหัส ทำให้เสี่ยงต่อการถูกดักฟังหรือดัดแปลงข้อมูล โดยสามารถเพิ่มความมั่นคงปลอดภัยได้ด้วย DNS over HTTPS (DoH), DNS over TLS (DoT), และ DNSSEC

5.2 ໂປຣໂຄລກຳນັດຄ່າທີ່ອູ້ໝີແບບໄດ້ນາມືກ (Dynamic Host Configuration Protocol – DHCP)
ອຸປະກນົມທຸກເຄື່ອງໃນເຄື່ອງຂ່າຍຕ້ອງມີການຕັ້ງຄ່າທີ່ອູ້ໝີ IP ຜຶ່ງປະກອບດ້ວຍ:

- Hostname ແລະ Fully Qualified Domain Name (FQDN)
- IP Address ແລະ Subnet Mask
- Default Gateway
- MAC Address



Cloud region with VPC further subdivided into subnets. (Images ©123RF.com)

ກາພທີ 43 Dynamic Host Configuration Protocol

ກາຮໍານັດຄ່າທີ່ອູ້ໝີ 2 ແບບກີ່ອ:

1. **ແບບ Static:** ຜູ້ຜູ້ແລະຮບຕັ້ງຄ່າດ້ວຍຕົນເອງ ແນະກັບອຸປະກນົມສຳຄັນ ເຊັ່ນ ເຊີ່ວິເວຼ່ວ
2. **ແບບ Dynamic (DHCP):** ອຸປະກນົມຮ້ອງຂອງຄ່າຄອນຟິກຈາກ DHCP Server ຈຶ່ງສະດວກຮັດເຮົວແລະລດ
ຄວາມຜິດພາດຈາກການຕັ້ງຄ່າດ້ວຍຕົນເອງ

ກຮະບວນການ DHCP ມີ 4 ຫັ້ນທອນ:

1. **DHCP Discover** – ເຄື່ອງລູກຂ່າຍສ່ວຍອອກປະກາສຂອທີ່ອູ້ໝີ IP
2. **DHCP Offer** – ເຊີ່ວິເວຼ່ວຕອບກັບດ້ວຍຂໍ້ອເສນອ

3. **DHCP Request** – เครื่องลูกข่ายตอบรับข้อเสนอ
4. **DHCP Acknowledgment** – เชิร์ฟเวอร์ยืนยันการจัดสรร IP

ใน AWS เมื่อสร้าง VPC จะต้องกำหนดช่วง IP และ Subnet โดยบริการ EC2 จะได้รับ IP อัตโนมัติผ่านการกำหนดค่า DHCP และสามารถรองรับทั้ง IPv4 และ IPv6

5.3 การติดตามที่อยู่ IP (IP Address Tracking)

การจัดสรรที่อยู่ไอพี (IP Address Allocation) จำเป็นต้องมีการติดตามและบริหารจัดการอย่างเป็นระบบซึ่งกระบวนการนี้มักเรียกว่า การจัดการที่อยู่ไอพี (IP Address Management – IPAM)

ขั้นตอนแรกในการดำเนินการคือการจัดทำ แผนผังเครือข่าย (Network Map) ซึ่งแสดงรายละเอียดของ คลาวด์ส่วนตัวเสมือน (Virtual Private Cloud – VPC) และ ชับเน็ต (Subnets) ที่เกี่ยวข้องภายใน VPC เอกสารดังกล่าวจะระบุช่วงของที่อยู่ไอพี (IP address ranges) ที่มีอยู่และตำแหน่งที่ใช้งานภายในโครงสร้างพื้นฐานระบบคลาวด์

หลังจากนั้น ควรจัดทำวิธีการติดตามการใช้ที่อยู่ไอพีภายในแต่ละชับเน็ต โดยทั่วไปแล้ว:

- การติดตามที่อยู่ไอพีแบบไนามิก (Dynamic IP Addresses) จะไม่จำเป็นต้องติดตามอย่างละเอียดเนื่องจากมีการจัดสรรโดยอัตโนมัติ
- แต่ การติดตามที่อยู่ไอพีแบบคงที่ (Static IP Addresses) ถือเป็นสิ่งสำคัญเพื่อหลีกเลี่ยงการจัดสรรซ้ำซ้อน (Duplicate Assignment)

ในระบบคลาวด์ เช่น Amazon EC2 หรือ Azure Virtual Machines จะมีระบบจัดการที่อยู่ไอพีที่คล้ายกับ เชิร์ฟเวอร์ DHCP ในระบบเครือข่ายทั่วไป

การติดตามที่อยู่ไอพีมีความสำคัญอย่างยิ่งในระบบที่เป็นแบบ ไฮบริด (Hybrid) หรือ มัลติคลาวด์ (Multi-cloud) ซึ่งมีทรัพยากรไอพีกระจายอยู่ทั้งในสภาพแวดล้อมแบบภายในภูมิภาคและภายนอกภูมิภาค ตัวอย่างเช่น:

- ใน AWS ผู้ดูแลระบบสามารถค้นหาทรัพยากรที่ใช้ที่อยู่ไอพีเฉพาะได้จากหน้าคอนโซล EC2 โดยดูที่หัวข้อ Network Interfaces
- ทั้ง Azure Marketplace และ AWS Marketplace มีเครื่องมือ IPAM จากผู้พัฒนารายอื่นให้เลือกใช้หลากหลาย

หมายเหตุ: การติดตามที่อยู่ไอพีไม่ได้จำกัดเฉพาะในระบบคลาวด์เท่านั้น ในระบบเครือข่ายแบบดั้งเดิม การจัดสรรและติดตามที่อยู่ไอพีก็เป็นเรื่องที่ต้องบริหารจัดการเช่นเดียวกัน

5.4 โปรโตคอลเวลาบนเครือข่าย (Network Time Protocol – NTP)

การบริหารจัดการเวลา (Time Management) ถือเป็นบริการพื้นฐานที่มีความสำคัญในระบบเครือข่าย โดย โปรโตคอลเวลาบนเครือข่าย (Network Time Protocol – NTP) ถูกใช้เพื่อให้เกิดการซิงโครไนซ์เวลา ระหว่างอุปกรณ์ต่าง ๆ บนเครือข่าย เช่น เครื่องลูกข่าย (Client) และเครื่องแม่ข่าย (Server)

หากอุปกรณ์ต่าง ๆ มีเวลาที่ไม่ตรงกันหรือเกิด “Time Drift” (ความคลาดเคลื่อนของเวลา) อาจส่งผล ให้เกิดปัญหาด้านการสื่อสาร การพิสูจน์ตัวตนล้มเหลว หรือระบบงานที่อ่อนไหวต่อเวลาทำงานผิดพลาด

ในระบบเครือข่ายแบบติดตั้งภายในองค์กร (On-premises) การซิงโครไนซ์เวลาจะถูกดำเนินการ โดยอัตโนมัติผ่านเซิร์ฟเวอร์ภายใน แต่ในสภาพแวดล้อมคลาวด์ เช่น:

- Elastic Compute Cloud (EC2) ของ Amazon Web Services (AWS) และ
- Azure Virtual Machines (VMs) ของ Microsoft Azure

เครื่องเสมือน (VMs) เหล่านี้จะซิงโครไนซ์เวลา กับ Host Machine ที่เป็นเครื่องจริง โดยอัตโนมัติ โดยเฉพาะ Azure VM จะรับเวลาอ้างอิงจาก Stratum 1 Time Servers ของ Microsoft

ผู้ดูแลระบบยังสามารถตั้งค่าให้เครื่อง VM เหล่านี้ซิงโครไนซ์เวลา กับแหล่งภายนอก เช่น โครงสร้าง พื้นฐานเวลา (Time Infrastructure) ขององค์กรเอง ตัวอย่างเช่น:

- AWS Linux VM สามารถใช้บริการ Chrony เพื่อบริหารจัดการเวลาได้
- สำหรับ Linux โดยทั่วไป จะมีการตั้งค่าผ่านไฟล์ */etc/ntp.conf**

Network Time Security (NTS)

NTS เป็นการเพิ่มความมั่นคงปลอดภัยให้กับการซิงโครไนซ์เวลาผ่าน NTP โดยใช้กระบวนการ แลกเปลี่ยนกุญแจ (Key Exchange) เพื่อรับข้อมูลเซิร์ฟเวอร์เวลาอย่างแม่นยำ จากนั้นใช้ฟังก์ชันของ NTP เพื่อรับข้อมูลเวลาและตรวจสอบแหล่งที่มาของข้อมูลดังกล่าว

ด้วยการใช้ NTS เครื่องลูกข่าย (Client) จะสามารถ เชื่อถือได้ว่าเซิร์ฟเวอร์เวลา มีตัวตนที่แท้จริง และข้อมูลเวลาที่ได้รับมานั้น มีความถูกต้องและไม่ถูกดัดแปลง

หน่วยที่ 6 การแก้ไขปัญหาเครือข่าย (Troubleshoot Network Issues)

การติดตั้งโครงสร้างพื้นฐานเครือข่าย (Network Infrastructure) เป็นเพียงจุดเริ่มต้นของการให้บริการ ระบบคลาวด์เท่านั้น หน้าที่ของผู้ดูแลระบบคลาวด์ (Cloud Administrator) ยังรวมถึงการแก้ไขปัญหาและ ปรับแต่งบริการเครือข่ายและองค์ประกอบต่าง ๆ อย่างต่อเนื่องด้วย

การแก้ไขปัญหาในระบบคลาวด์จำเป็นต้องดำเนินการในหลากหลายมิติ ได้แก่:

- การตรวจสอบการเชื่อมต่อ (Connectivity Verification)
- การยืนยันสถานะของบริการ (Service Availability)
- การตรวจสอบการตั้งค่าของอุปกรณ์ (Device Configuration)
- การประเมินประสิทธิภาพว่าเป็นไปตามวัตถุประสงค์ (Performance Objectives)

6.1 เครื่องมือมาตรฐานสำหรับการแก้ไขปัญหาเครือข่าย (Standard Network Troubleshooting Tools)

การแก้ไขปัญหาเครือข่ายควรเริ่มต้นจากการตรวจสอบพื้นฐาน โดยหนึ่งในขั้นตอนเริ่มต้นของการแก้ไขปัญหาในระบบคลาวด์ คือ การตรวจสอบการเชื่อมต่อ (Connectivity) ซึ่งสามารถใช้เครื่องมือเดียวกับที่ใช้ในเครือข่ายภายในองค์กร (On-premises) ได้แก่ ping และ traceroute

คำสั่ง Ping

คำสั่ง ping ใช้สำหรับทดสอบการเชื่อมต่อระหว่างโหนดสองจุดในเครือข่าย โดยทำงานในระดับ Network Layer ของโปรโตคอล TCP/IP ไม่ขึ้นอยู่กับการตั้งค่าของโปรแกรมใน Application Layer จึงสามารถใช้ตรวจสอบการเชื่อมต่อพื้นฐานได้

รูปแบบคำสั่ง:

```
ping server42
```

```
ping 192.168.2.200
```

ผลลัพธ์จากคำสั่ง ping ที่เป็นไปได้มีดังนี้:

- **สำเร็จ (Success):** ได้รับข้อความ REPLY แสดงว่าการเชื่อมต่อทำงานปกติ ทั้งสองโหนดสามารถส่งและรับข้อมูลได้
- **ล้มเหลว (Fail):** ได้รับข้อความ REQUEST TIMED OUT แสดงว่าเครื่องตั้งทางส่งคำขอได้ แต่ไม่สามารถรับการตอบกลับจากปลายทางได้ (อาจมีปัญหาที่อุปกรณ์ปลายทาง)
- **ล้มเหลว (Fail):** ได้รับข้อความ DESTINATION HOST UNREACHABLE แสดงว่าเครื่องตั้งทางไม่สามารถส่งคำขอได้เลย (อาจมีปัญหาที่อุปกรณ์ตั้งทาง)

เทคนิคเพิ่มเติม:

- หาก ping ด้วยชื่อโฮสต์ (hostname) สำเร็จ แสดงว่าทั้งการเชื่อมต่อและระบบการแปลงชื่อ (name resolution) ทำงานได้
- หาก ping ด้วยชื่อโฮสต์ล้มเหลว แต่ ping ด้วย IP สำเร็จ แสดงว่าระบบการแปลงชื่อ (DNS) มีปัญหา
- หากทั้ง ping ด้วยชื่อโฮสต์และ IP ล้มเหลว แสดงว่าการเชื่อมต่อเครือข่ายมีปัญหานั่นเอง

ในระบบ Linux และ macOS คำสั่ง ping จะทำงานต่อเนื่องโดยปริยาย ในขณะที่ Windows จะส่งคำขอ 4 ครั้ง (สามารถกำหนดจำนวนเองได้)

เส้นทางเครือข่าย (Network Path)

การเชื่อมต่อในเครือข่ายอาจมีความซับซ้อน ดังนั้นการตรวจสอบเส้นทางของข้อมูลจึงเป็นประโยชน์ โดยสามารถใช้คำสั่ง traceroute (ใน Linux/macOS) หรือ tracert (ใน Windows)

ตัวอย่างคำสั่ง:

```
traceroute www.comptia.org # Linux/macOS
```

```
tracert www.comptia.org # Windows
```

คำสั่งนี้จะแสดงรายการของเราเตอร์ (Router) ที่ข้อมูลเดินทางผ่านไปจนถึงปลายทาง ซึ่งสามารถใช้เปรียบเทียบกับแผนภาพโครงสร้างเครือข่ายเพื่อตรวจสอบว่าข้อมูลเดินทางถูกต้องหรือไม่ หากเส้นทางหยุดอยู่ที่เราเตอร์ตัวใดตัวหนึ่ง แสดงว่าปัญหาอาจอยู่ที่จุดนั้น รายการที่ควรตรวจสอบเมื่อการเชื่อมต่อล้มเหลวหรือ ping ไม่สำเร็จ:

- ตรวจสอบการตั้งค่า IP address ของเครื่องต้นทาง
- ตรวจสอบว่าเสียบสายเครือข่ายถูกต้องและไฟแสดงสถานะทำงาน
- ตรวจสอบสถานะของสวิตช์ที่เชื่อมต่ออยู่
- ตรวจสอบสถานะของเราเตอร์ที่เชื่อมต่อระหว่างเครือข่ายอยู่
- ตรวจสอบว่าบริการ DNS ทำงานอยู่หรือไม่
- ตรวจสอบการตั้งค่าของ Firewall หรืออุปกรณ์ที่กรองข้อมูล
- ตรวจสอบว่าอุปกรณ์ปลายทางเปิดใช้งานและเชื่อมต่อกับเครือข่ายหรือไม่

6.2 การให้บริการของระบบ DNS (DNS Service Availability)

เริ่มต้นการแก้ไขปัญหาโดยการตรวจสอบว่าเซิร์ฟเวอร์ Domain Name System (DNS) พร้อมใช้งานหรือไม่ ควรยืนยันว่าเซิร์ฟเวอร์ DNS เปิดเครื่องอยู่และมีการเปิดใช้งานบริการ DNS แล้ว จากนั้นใช้เครื่องมือตรวจสอบการเชื่อมต่อเครือข่ายพื้นฐาน เช่น ping เพื่อตรวจสอบว่าอุปกรณ์ของผู้ใช้สามารถเชื่อมต่อกับเซิร์ฟเวอร์ DNS ได้ผ่านเครือข่าย

ระเบียนทรัพยากรของ DNS (DNS Resource Records)

DNS ถือเป็นหนึ่งในบริการเครือข่ายที่สำคัญที่สุด เนื่องจากมุ่งเน้นใช้งานผ่านชื่อ (Name) แต่คอมพิวเตอร์ต้องอ้างอิงด้วยหมายเลข IP หากมีข้อผิดพลาดในระเบียนทรัพยากรที่เชื่อมโยงระหว่างชื่อและหมายเลข IP จะส่งผลให้เกิดปัญหาในการเชื่อมต่อ

โดยปกติแล้ว เครื่องลูกข่าย เช่น โทรศัพท์ แท็บเล็ต และคอมพิวเตอร์ของผู้ใช้มักจะลงทะเบียนชื่อและหมายเลข IP ของตนโดยอัตโนมัติ ซึ่งเรียกว่า Resource Records ขณะที่อุปกรณ์และบริการอื่น ๆ ในเครือข่ายมักจะใช้ระเบียนแบบคงที่ที่ผู้ดูแลระบบสร้างไว้

หากปัญหาคือไม่สามารถเข้าถึงเซิร์ฟเวอร์หรือบริการได้ ควรตรวจสอบดังนี้:

- การแปลงชื่อทำงานถูกต้องหรือไม่ (Name Resolution)
- เครื่องลูกข่ายสามารถเชื่อมต่อกับเซิร์ฟเวอร์ DNS และแปลงชื่อได้หรือไม่
- เซิร์ฟเวอร์ DNS มีระเบียนสำหรับทรัพยากรที่ต้องการหรือไม่
- ระเบียน DNS ถูกต้องหรือไม่

6.3 การให้บริการของ IP Address (IP Address Service Availability)

อุปกรณ์เครือข่ายจำเป็นต้องมีการกำหนดค่าหมายเลข IP ที่ถูกต้อง ซึ่งสามารถตั้งค่าได้แบบมั�วัลโดยผู้ดูแลระบบ หรือแบบอัตโนมัติโดยบริการ DHCP

การกำหนดค่าที่ไม่ถูกต้อง (Incorrect IP Configurations)

เครื่องลูกข่าย (Cloud Clients), เซิร์ฟเวอร์ และอุปกรณ์เครือข่ายทั้งแบบภายในและภายนอก ต้องมีค่ากำหนดเครือข่ายที่ถูกต้องเพื่อสื่อสารกันได้ โดยค่ากำหนดมาตรฐานได้แก่:

- หมายเลข IP (IPv4 หรือ IPv6)
- Subnet mask: แยกเครือข่ายออกจากโฉนด
- Default gateway: เส้นทางไปยังเครือข่ายอื่น
- Routing: เส้นทางการส่งข้อมูล

หากปัญหาเกิดจากการตั้งค่า IP ผิดพลาด ควรตรวจสอบ:

- หมายเลข IP หรือ subnet mask ผิด
- Default gateway พิมพ์ผิด (กรณีตั้งค่าด้วยมือ) หรือ DHCP กำหนดค่าผิด
- ไม่มีการตั้งค่า IP เลย ทำให้ระบบกำหนดค่า Automatic Private IP Address (APIPA) เป็น 169.254.x.x

คำสั่งสำหรับตรวจสอบ IP Address

Windows

ใช้คำสั่ง ipconfig ดังนี้:

คำสั่ง	คำอธิบาย
ipconfig	แสดงข้อมูลพื้นฐานของ IP เช่น IP address, subnet mask, default gateway
ipconfig /all	แสดงข้อมูลรายละเอียดเพิ่มเติม รวมถึง DHCP, DNS และอื่น ๆ
/release	ปล่อย IP ที่ได้จาก DHCP กลับคืนไปยังเซิร์ฟเวอร์ DHCP
/renew	ขอ IP ใหม่จาก DHCP
/displaydns	แสดงผลลัพธ์การแปลงชื่อที่เก็บอยู่ใน cache
/flushdns	ล้าง cache ของการแปลงชื่อเพื่อโหลดข้อมูลใหม่

ตาราง 4 คำสั่งสำหรับตรวจสอบ IP Address

Linux

ใช้คำสั่ง ip หรือ ifconfig (บางระบบ):

- แสดง IP ทั้งหมด: ip addr
- แสดง routing table: ip route
- แสดงข้อมูลการตั้งค่าเครือข่าย: ip link

6.4 ความพร้อมให้บริการของ DHCP (DHCP Service Availability)

การกำหนดค่าของเซิร์ฟเวอร์ DHCP (Dynamic Host Configuration Protocol) ที่ถูกต้องเป็นสิ่งสำคัญยิ่ง ต่อการสื่อสารในเครือข่าย หมายเลขอ IP ที่จัดสรรให้กับอุปกรณ์ต้องถูกต้อง และไม่ซ้ำกันระหว่างอุปกรณ์ต่าง ๆ

เซิร์ฟเวอร์ DHCP จะต้องมีหมายเลขอ IP เพียงพอสำหรับจำนวนอุปกรณ์ลูกข่ายที่รองรับ ซึ่งอาจมี การเปลี่ยนแปลงได้ ควรพิจารณาการกำหนดค่า *DHCP Failover* เพื่อให้หากเซิร์ฟเวอร์ DHCP หนึ่ง ไม่สามารถใช้งานได้ เซิร์ฟเวอร์อีกตัวสามารถทำหน้าที่แทนได้

รูปแบบการกำหนดค่า DHCP มีทั้งแบบ Active-Active (เพื่อการโหลดбалานซ์) และ Active-Standby (เพื่อสำรองในกรณีล้มเหลว)

Scope Exhaustion

Scope exhaustion คือสถานการณ์ที่เซิร์ฟเวอร์ DHCP ไม่มีหมายเลขอ IP เพียงพอใน scope เพื่อปล่อยให้กับอุปกรณ์ลูกข่าย Scope หมายถึงกลุ่มของ IP Address ที่พร้อมสำหรับการแจกจ่ายให้ลูกข่าย

ตัวอย่างเช่น scope ที่ใช้ IP class C มีเพียง 254 หมายเลขอ หากมีการแจกจ่ายครบหมดแล้ว จะไม่สามารถให้บริการแก่ลูกข่ายที่เข้ามาใหม่ได้

วิธีการตรวจสอบ:

- ตรวจสอบ scope ของ DHCP server
- ตรวจสอบ log file ของ DHCP เป็นประจำ
- เปรียบเทียบจำนวน client กับจำนวน IP ที่มีในเอกสารเครือข่าย
- ตรวจสอบว่ามีปัญหา IP ซ้ำ (duplicate IP) หรือไม่

Network Overlap

ห้ามมีการใช้ช่วง IP ซ้ำกันระหว่าง scopes ที่อยู่ในเซิร์ฟเวอร์ DHCP เดียวกันหรือข้ามเซิร์ฟเวอร์ เพราะจะก่อให้เกิดปัญหา IP ซ้ำ ส่งผลให้อุปกรณ์ไม่สามารถเชื่อมต่อเครือข่ายได้อย่างเสถียร

ตัวอย่างเช่น การตั้งค่า VPC สำหรับ Development, Testing และ Production จะต้องกำหนดช่วง IP ที่ไม่ทับซ้อนกัน

วิธีการตรวจสอบ:

- สังเกตข้อความผิดพลาดจาก VM ที่บ่งชี้ว่าเกิด IP ซ้ำ
- เอกสารเครือข่ายควรระบุ IP ranges ของแต่ละ DHCP scope
- เอกสารควรระบุหมายเลข IP แบบ static ทั้งหมดที่ตั้งค่าไว้บน router, server และอุปกรณ์อื่น ๆ

6.5 ความพร้อมของบริการ NTP (Network Time Protocol Service Availability)

การซิงโครไนซ์เวลาเป็นสิ่งสำคัญต่อความถูกต้องของการทำงานในระบบเครือข่าย โดยใช้โปรโตคอล NTP เพื่อให้เวลาของอุปกรณ์ต่าง ๆ ในระบบตรงกัน

ในระบบ VM มักจะดึงเวลาจาก host system โดยอัตโนมัติ แต่ก็สามารถกำหนดให้ดึงเวลาจากแหล่งภายนอกได้ เช่น time server ขององค์กร หรือแหล่งสาธารณะ หากเกิดปัญหาเวลาคลาดเคลื่อน ควรตรวจสอบดังนี้:

- ไฟล์กำหนดค่าเวลาใน host อาจผิดพลาด
- บริบูรณ์ NTS (Network Time Security) ถูกกำหนดไม่ถูกต้อง
- มี time drift ระหว่าง host กับ VM → ควรติดตั้ง guest extensions บน VM
- มี time drift ระหว่างระบบในองค์กรกับระบบคลาวด์ → ควรใช้ time source เดียวกัน

6.6 ปัญหาเกี่ยวกับ HTTP (Hypertext Transfer Protocol Issues)

HTTP เป็นโปรโตคอลหลักสำหรับการเข้าถึงเว็บไซต์ หากการเชื่อมต่อ HTTP ล้มเหลว ควรตรวจสอบประเด็นต่อไปนี้:

หากไม่สามารถเชื่อมต่อผ่าน HTTP ได้ ควรตรวจสอบว่า:

- เปิดพอร์ต 80 (TCP) บน firewall และหรือไม่
- เซิร์ฟเวอร์ที่ให้บริการเว็บทำงานอยู่หรือไม่

HTTP Status Codes ที่พบบ่อย

หมวดสถานะ	ความหมาย
4xx	ปัญหาฝั่งผู้ใช้
403 Forbidden	ผู้ใช้ไม่มีสิทธิ์เข้าถึงทรัพยากร
404 Not Found	ไม่พบทรัพยากรที่ร้องขอ
5xx	ปัญหาฝั่งเซิร์ฟเวอร์
500 Internal Server Error	เกิดข้อผิดพลาดที่ไม่ระบุชัดในเซิร์ฟเวอร์
502 Bad Gateway	เกตเวย์ได้รับคำตอบที่ไม่ถูกต้องจากเซิร์ฟเวอร์ต้นทาง

ตาราง 5 HTTP Status Codes ที่พบบ่อย

6.7 ปัญหาการกำหนดค่าเครือข่าย (Network Configuration Issues)

ข้อผิดพลาดด้านการกำหนดค่าเครือข่ายอาจเกิดขึ้นได้ทั้งในระบบจริงและระบบเสมือน เช่น:

- Security groups
- VPC (Virtual Private Cloud)
- อุปกรณ์เครือข่ายจริงและเสมือน
- อุปกรณ์ในองค์กร

สาเหตุที่พบบ่อย:

- ไม่มีการอัปเดต patch หรือ firmware

- ความผิดพลาดของมนุษย์ในการตั้งค่า
- การแบ่งส่วนเครือข่ายผิด (network segmentation)
- ไฟล์กำหนดค่าผิด
- ไม่มีระบบควบคุมเวอร์ชันของไฟล์กำหนดค่า

แนวทางการวิเคราะห์:

- ระบุปัญหาและขอบเขตของปัญหา
- อุปกรณ์ที่ได้รับผลกระทบอยู่ใน VPC เดียวกันหรือไม่
- อยู่ใน VLAN/Subnet เดียวกันหรือไม่
- อยู่หลัง firewall ตัวเดียวกันหรือไม่
- ปัญหาเกิดกับเครือข่ายภายใน (on-premises) หรือระบบคลาวด์ หรือทั้งสอง

6.8 ปัญหาความหน่วงของการกำหนดค่าเครือข่าย (Network Configuration Latency Issues)

Latency หรือ “ค่าความหน่วงเวลา” หมายถึงช่วงเวลาระหว่างการร้องขอ (Request) และการให้บริการ (Response) โดยเป็นตัวสะท้อนถึงความล่าช้าในการสื่อสารของเครือข่าย ซึ่งส่งผลกระทบต่อประสิทธิภาพและประสบการณ์ของผู้ใช้

แอปพลิเคชันที่มีความไวต่อความหน่วงสูง ได้แก่:

- แอปพลิเคชันวิเคราะห์ข้อมูลแบบเรียลไทม์ (Streaming analytics)
- เครื่องมือบริหารจัดการข้อมูล (Data management tools)
- API (Application Programming Interfaces)
- การทำงานของวิดีโอและภาพเคลื่อนไหว

แนวทางการตรวจสอบเมื่อเกิดปัญหาความหน่วง:

- ระยะทาง: ลดระยะห่างระหว่างผู้ใช้และบริการ
- จำนวน hops: ลดจำนวนอุปกรณ์เราเตอร์ที่ข้อมูลต้องผ่าน
- ปริมาณข้อมูล: พิจารณาปริมาณข้อมูลที่ถูกส่งผ่านเครือข่าย
- ประสิทธิภาพของเซิร์ฟเวอร์: ตรวจสอบว่าเซิร์ฟเวอร์มีทรัพยากรเพียงพอ เช่น การตั้งค่าเครือข่ายและดิสก์
- สื่อกลางของเครือข่าย: เช่น สายเคเบิล (Network cabling) มีผลต่อคุณภาพของเครือข่าย
- การจัดลำดับความสำคัญ: ตรวจสอบหรือกำหนดค่าการบริการคุณภาพ (QoS)

6.9 การแก้ไขปัญหาเกี่ยวกับคุณภาพของบริการ (Troubleshoot Quality of Service - QoS)

QoS (Quality of Service) คือการกำหนดลำดับความสำคัญให้กับทรัพยากรทางประเททในเครือข่าย เช่น VoIP (Voice over IP) เพื่อให้มั่นใจว่าทรัพยากรที่สำคัญจะไม่ถูกชะลอจากทรัพยากรอื่นที่ไม่สำคัญเท่า

หาก QoS ไม่ทำงานตามที่คาดหวัง ควรพิจารณา:

- เครือข่ายอาจเกิดภาวะคับคั่ง (Congestion) ส่งผลกระทบต่อ traffif ทุกชนิด ไม่เฉพาะเพียง traffif ที่ควรได้รับความสำคัญ
- QoS tag หรือ header อาจถูกกำหนดค่าผิดพลาด ทำให้การจัดลำดับความสำคัญล้มเหลว

6.10 ปัญหาแบบดีวิดท์และปริมาณ traffif (Network Bandwidth and Throughput Issues)

- Bandwidth (แบนด์ดีวิดท์):** หมายถึงปริมาณข้อมูลที่สามารถส่งผ่านเครือข่ายต่อหนึ่งหน่วยเวลา (เช่น Mbps)
- Throughput:** หมายถึงปริมาณข้อมูลจริงที่ส่งผ่านสำเร็จ ซึ่งได้รับผลกระทบจากการสูญเสียของ packet และปัญหาอื่น ๆ

หากเกิดปัญหาแบบดีวิดท์ ควรตรวจสอบ:

- ความล่าช้าของเครือข่าย (Latency)
- แอปพลิเคชันที่ใช้แบนด์ดีวิดท์สูงเกินไป
- การตั้งค่าอุปกรณ์เครือข่ายผิดพลาด
- อุปกรณ์ที่ประมวลผล traffif ไม่ทัน เช่น Router หรือ Proxy
- การกำหนดค่าของ Load Balancer (ประเภทและวิธีการกระจายโหลด)

6.10.1 การตั้งค่า MTU เพื่อเพิ่มประสิทธิภาพ (Setting Maximum Transmission Unit Sizes)

MTU (Maximum Transmission Unit) คือขนาดสูงสุดของข้อมูล (ในหน่วยไบต์) ที่สามารถส่งผ่านในแต่ละแพ็กเก็ตของ TCP/IP โดยไม่มีการแบ่งเพิ่มเติม

หากการตั้งค่า MTU ไม่เพิ่มประสิทธิภาพ อาจเกิดจาก:

- เครือข่ายมีข้อผิดพลาดจำนวนมาก ทำให้เกิดการ retransmit ของแพ็กเก็ตบ่อย ส่งผลต่อความมีประสิทธิภาพ
- อุปกรณ์เครือข่ายบางตัวไม่รองรับแพ็กเก็ตขนาดใหญ่ → ผู้ส่งอาจส่งแพ็กเก็ตใหญ่กว่าที่ผู้รับรองรับได้

คำสั่งที่ใช้ดู MTU บน Windows:

```
netsh interface ipv4 show subinterfaces
```

6.10.2 ปัญหาเกี่ยวกับการเชื่อมต่อและประสิทธิภาพของเครือข่าย (Network Connectivity and Performance Issues)

การทดสอบขั้นพื้นฐาน เช่น ping และ traceroute ช่วยตรวจสอบการเชื่อมต่อระหว่างโหนดเครือข่าย และระบบเส้นทางของ traffif ได้

หากการเชื่อมต่อล้มเหลว ควรพิจารณาสาเหตุดังนี้:

- ผู้ให้บริการคลาวด์ (CSP) เกิดระบบล้ม (Outage)

- ระบบป้องกันการบุกรุก (IPS) หยุดทำงาน
- อุปกรณ์เครือข่ายในองค์กรมีปัญหา
- ตรวจสอบ log file และระบบอนิเตอร์
- ตรวจสอบว่า instance tag ของระบบคลาวด์ถูกต้อง เพื่อให้สามารถกำหนด firewall และ routing ได้อย่างเหมาะสม

6.11 ปัญหาการกำหนดค่าอุปกรณ์ผิดพลาด (Device Misconfiguration Issues)

ข้อผิดพลาดในการกำหนดค่าของอุปกรณ์อาจเกิดขึ้นทั้งที่ผู้ใช้งาน (Client) หรือ เซิร์ฟเวอร์ (Server) รวมถึงอุปกรณ์เครือข่ายกลาง เช่น เรตเตอร์ (Router), สวิตช์ (Switch), โหลดбалานเซอร์ (Load Balancer) หรืออุปกรณ์เครือข่ายอื่น ๆ

การกำหนดค่า Proxy

Proxy ทำหน้าที่เป็นตัวกลางระหว่างผู้ใช้งานกับบริการบนอินเทอร์เน็ต หากผู้ใช้งานไม่สามารถเข้าถึง Proxy หรือ Proxy มีการตั้งค่าผิดพลาด จะทำให้ไม่สามารถเข้าถึงอินเทอร์เน็ตได้ หากปัญหาคือไม่สามารถเข้าถึงอินเทอร์เน็ตหรือคลาวด์:

- ตรวจสอบว่าผู้ใช้งานสามารถเข้าถึง Proxy ได้หรือไม่
- ตรวจสอบว่า Proxy ออนไลน์และสามารถเข้าถึงทรัพยากรอินเทอร์เน็ตได้
- ตรวจสอบไฟร์วอลล์และการตั้งค่า micro-segmentation ว่าถูกต้องหรือไม่

หากมีปัญหาการเชื่อมต่อแบบจำกัดกับเครือข่ายย่อย (microsegments):

- ตรวจสอบการตั้งค่า IP address
- ตรวจสอบกฎ ACL และลำดับของกฎ
- ตรวจสอบกลุ่มความปลอดภัย (Security Group) ที่เครื่องลูกข่ายเป็นสมาชิก

คำสั่งตรวจสอบการเชื่อมต่อใน Linux

- ss ใช้แสดงสถานะการเชื่อมต่อ TCP/UDP ปัจจุบันบนเซิร์ฟเวอร์
- ระบบเก่าอาจใช้ netstat (เลิกใช้แล้ว)

6.12 ปัญหาความไม่เข้ากันของโปรโตคอลและการเลิกใช้ (Protocol Incompatibility and Deprecation)

เครือข่ายจำเป็นต้องใช้โปรโตคอลที่ทั้งสองฝ่ายสามารถสื่อสารได้ตรงกัน เช่น Client–Server, VPN–VPN, Router–Router ปัญหา VLAN

โปรโตคอลของ VLAN บางประเภทไม่สามารถทำงานร่วมกันได้

- ไม่เข้ากัน: VXLAN, NVGRE, STT
- แนะนำ: ใช้ GENEVE เพื่อเชื่อมระหว่างโปรโตคอลที่แตกต่างกัน

HTTP กับ HTTPS

- ควรใช้ HTTPS เพื่อความปลอดภัยและหลีกเลี่ยงการโจมตีแบบ Man-in-the-Middle
- ปัญหาเกิดจาก Client ต้องการ HTTPS แต่ Server รองรับเฉพาะ HTTP หรือในทางกลับกัน

แนวทางแก้ไข:

- ยืนยันว่า Client รองรับและยอมรับใบเซอร์ฟิล์ม HTTPS
- ตั้งค่า Web Server ให้ใช้ใบเซอร์ฟิล์ม HTTPS ที่ถูกต้อง
- ตรวจสอบวันหมดอายุของใบเซอร์ฟิล์มและความถูกต้องของ CA

การเลิกใช้โปรโตคอล

- หลีกเลี่ยงการใช้ RIP เพราะไม่เหมาะสมกับระบบคลาวด์
- ใช้ OSPF หรือ BGP แทน
- อัปเกรดเราเตอร์เก่าที่ไม่รองรับโปรโตคอลใหม่

โปรโตคอลที่ไม่เข้ารหัส

หลีกเลี่ยงการใช้โปรโตคอลที่ไม่มีการเข้ารหัสในระบบคลาวด์ เช่น Telnet

แนะนำให้ใช้:

- HTTPS
- SSH
- SFTP
- RDP แบบเข้ารหัส

6.13 ปัญหาเกี่ยวกับการกำหนดค่าการ Routing (Routing Issues)

แนวทางการตรวจสอบ Routing

ใช้คำสั่ง ping และ traceroute (tracert ใน Windows) เพื่อวิเคราะห์เส้นทางของข้อมูลและจุดที่เกิดปัญหา

Routing Table ขาดข้อมูล (Missing Routes)

Routing Table อาจไม่ถูกตั้งค่าหรือไม่ได้รับข้อมูลจากโปรโตคอล Routing

ควรพิจารณา:

- ใช้โปรโตคอล Routing ที่เข้ากัน เช่น OSPF, BGP
- กรณีใช้ Static Route: ตรวจสอบว่าป้อนเส้นทางครบถ้วนหรือไม่
- Dynamic Route: ตรวจสอบว่าเราเตอร์อัปเดต Routing Table ถูกต้องหรือไม่

การกำหนดค่า Route ผิดพลาด

ประเภทของ Route:

- Static: ระบุปลายทางแบบคงที่
- Default: ใช้มีเมื่อไม่มี Route อื่นที่ตรง

- Dynamic: อัปเดตอัตโนมัติจาก Routing Protocol

ปัญหาอาจเกิดจาก:

- ข้อมูล Default Route ที่ได้จาก DHCP ผิด
- ข้อมูล Static Route พิมพ์ผิด
- Routing Table ที่ได้จากเราเตอร์อื่นมีข้อมูลผิด

ปัญหาการเชื่อมต่อ VPC

- ตรวจสอบว่าได้ตั้งค่า VPC Peering และ Routing ทั้งสองฝ่ายถูกต้อง
- ตรวจสอบ Security Group และการตั้งชื่อ

6.14 การแปลงที่อยู่เครือข่าย (Network Address Translation - NAT)

เนื่องจากที่อยู่ IP แบบ IPv4 มีจำนวนจำกัด NAT จึงถูกใช้เพื่อแปลง IP ภายในองค์กร (Private IP) ให้สามารถใช้งานอินเทอร์เน็ตได้โดยใช้ Public IP ร่วมกัน

ช่วง IP Address ที่ส่วนไว้สำหรับใช้งานภายใน (Private IP Address Ranges):

Class	IP Range
A	10.0.0.0/8
B	172.16.0.0/12
C	192.168.0.0/16

ตาราง 6 ช่วง IP Address ที่ส่วนไว้สำหรับใช้งานภายใน

ปัญหา NAT ทำให้ไม่สามารถเข้าถึงอินเทอร์เน็ตได้

- อุปกรณ์ที่ทำ NAT (โดยมากคือ Router) ของเล่น
- ตาราง NAT ไม่ถูกต้อง

กรณี VPN ใช้งานร่วมกับ NAT

- VPN ป้องกันการเปลี่ยนแปลงแพ็กเก็ต แต่ NAT จำเป็นต้องเปลี่ยน IP
- ต้องทำ NAT ก่อน ใช้ tunneling เพื่อป้องกันการชนกันของ IP และรักษาความลับของต้นทาง

แนวทางแก้ไข:

- ตรวจสอบโปรโตคอล NAT ที่ใช้งาน
- ตรวจสอบการกำหนดค่า NAT
- ใช้ NAT แบบเฉพาะสำหรับ VPN

6.15 ปัญหาการสวิตช์เครือข่าย (Switching Issues)

แม้ว่าการตั้งค่าพื้นฐานของอุปกรณ์สวิตช์ (Switch) มากจะเป็นแบบอัตโนมัติและใช้งานได้ง่ายแต่ในระบบเครือข่ายที่ซับซ้อนมากขึ้น การกำหนดค่าที่เกี่ยวข้องกับ VLAN และ Trunking มีความสำคัญอย่างยิ่งต่อการรับรองว่าเครือข่ายจะทำงานได้อย่างถูกต้อง ปลอดภัย และมีประสิทธิภาพ

การกำหนดค่า VLAN (VLAN Configurations)

VLAN (Virtual LAN) เป็นการแบ่งส่วนเครือข่ายแบบโลจิกในระดับเลเยอร์ 2 (Layer 2) ซึ่งมีประสิทธิภาพมากกว่าการแบ่งชั้บเน็ตในเลเยอร์ 3 การใช้ VLAN ช่วยแยกกลุ่มการใช้งานในเครือข่ายออกจากกัน ทำให้การควบคุมการเข้าถึงและความปลอดภัยดีขึ้น

- ปัญหาที่พบ: จำนวน VLAN ไม่เพียงพอ
 - ขีดจำกัดของ VLAN แบบดั้งเดิมคือ 4,094 หมายเลข
 - แนวทางแก้ไข: เปลี่ยนมาใช้ VXLAN ซึ่งสามารถรองรับได้ถึงประมาณ 16 ล้านหมายเลข

Tag ที่ตั้งค่าผิดพลาด (Misconfigured Tags)

Tag ใน VLAN ใช้ระบุว่าสวิตช์พอร์ตใดอยู่ใน VLAN ใด หากการตั้งค่า tag ผิดพลาด อาจทำให้พอร์ตที่ควรแยกออกจากกันกลับสามารถติดต่อกันได้

หากพบว่าอุปกรณ์ใน VLAN ต่างกันสามารถสื่อสารกันได้:

- ตรวจสอบการตั้งค่า tag ในแต่ละสวิตช์
- ตรวจสอบว่าแต่ละพอร์ตอยู่ใน VLAN ที่ถูกต้อง
- ตรวจสอบการตั้งค่า trunk ระหว่างสวิตช์

พอร์ต Access กับ Trunk (Access and Trunk Ports)

พอร์ตในสวิตช์มี 2 ประเภทหลัก:

1. Access Ports – ใช้เชื่อมต่อกับเครื่องลูกข่าย (เช่น คอมพิวเตอร์, เซิร์ฟเวอร์)
2. Trunk Ports – ใช้เชื่อมต่อระหว่างสวิตช์เพื่อส่ง流量 VLAN ผ่านพอร์ตเดียว

ข้อควรระวัง:

- ห้ามต่ออุปกรณ์ลูกข่ายเข้ากับ trunk port
- หลีกเลี่ยงการตั้งค่า trunk บนพอร์ตที่ใช้เชื่อมต่อ client

6.16 เครื่องมือจับแพ็คเก็ตและสแกนเครือข่าย (Packet Capture and Scanning Utilities)

เครื่องมือในกลุ่มนี้ใช้ในการวิเคราะห์ปัญหาเครือข่ายและตรวจสอบความปลอดภัย โดยช่วยให้ผู้ดูแลระบบสามารถตรวจสอบการรับส่งข้อมูลได้อย่างละเอียด

การจับแพ็คเก็ต (Packet Capture)

การดักจับแพ็คเก็ตในเครือข่ายสามารถใช้วิเคราะห์ปัญหา เช่น การเชื่อมต่อช้า การรับส่งข้อมูลผิดปกติ หรือการโจมตี

เครื่องมือยอดนิยม:

- **Wireshark:** มี GUI ใช้งานง่าย แสดงรายละเอียดทั้ง IP Address, Protocol และ Payload (เนื้อหาภายใน)
- **tcpdump:** ทำงานผ่าน Command Line, มักใช้ในระบบ Linux

ตัวอย่างการใช้งาน:

- ตรวจสอบว่าแพ็กเก็ตมีการส่งข้อมูลที่ลasseiyd อ่อน (เช่น รหัสผ่าน) แบบไม่เข้ารหัสหรือไม่
- วิเคราะห์ชนิดของโปรโตคอลที่ใช้งานภายในเครือข่าย

การสแกนเครือข่าย (Network Scanning)

ใช้เพื่อตรวจสอบอุปกรณ์ที่เชื่อมต่อในเครือข่ายว่ามีบริการใดเปิดใช้งานอยู่ เช่น Web, DNS, DHCP

เครื่องมือยอดนิยม:

- **Nmap:** ใช้สแกนพอร์ตและอุปกรณ์ในเครือข่าย เพื่อค้นหาบริการที่ไม่รู้จักหรืออุปกรณ์ที่อาจไม่ได้รับอนุญาต (Rogue Devices)

ประโยชน์:

- ตรวจสอบการตั้งค่าความปลอดภัย
- ค้นหาจุดอ่อนที่อาจถูกโจมตี
- ตรวจสอบว่าอุปกรณ์และบริการที่เชื่อมต่ออยู่ตรงกับที่ควรเป็นหรือไม่

สรุปความเข้าใจเรื่องระบบเครือข่ายในคลาวด์ (Comprehending Cloud Networking)

ระบบเครือข่ายในคลาวด์ (Cloud Networking) ถือเป็นองค์ประกอบสำคัญของโครงสร้างพื้นฐานคลาวด์ โดยอุปกรณ์เครือข่ายแบบดั้งเดิม เช่น เร��เตอร์, ไฟร์วอลล์, และสวิตช์ ได้ถูกจำลอง (Virtualized) ขึ้นมาให้รองรับการทำงานในระบบคลาวด์ ซึ่งมีบทบาทในการควบคุมрафฟิกของเครือข่าย การจัดการประสิทธิภาพเครือข่าย และการเชื่อมตอกับระบบภายในองค์กร (On-Premises)

ประเด็นสำคัญ:

- แนวคิดด้านเครือข่ายแบบดั้งเดิม ยังคงมีประโยชน์ในระบบคลาวด์ เช่น การแบ่งชั้บเน็ต, การใช้ ACL, VLAN ฯลฯ
- การแก้ไขปัญหาเครือข่าย (Troubleshooting) ยังคงใช้เครื่องมือที่คล้ายกับระบบแบบเดิม เช่น ping, traceroute, netstat, Wireshark
- ความปลอดภัยของการเชื่อมต่อ เป็นสิ่งจำเป็นในระบบคลาวด์ ซึ่งมักใช้ VPN และไฟร์วอลล์ในการป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต
- การแบ่งเครือข่าย (Segmentation) ช่วยจำกัดขอบเขตการเข้าถึงระหว่างบริการต่างๆ ในเครือข่าย
- เครือข่ายแบบ Software-Defined Networking (SDN) ทำให้การจัดการเครือข่ายง่ายขึ้น โดยเฉพาะในเรื่องการขยายระบบ (scaling) และการปรับใช้งานอย่างรวดเร็ว

สรุปได้ว่า การเข้าใจโครงสร้างและการทำงานของเครือข่ายในคลาวด์มีความจำเป็นอย่างยิ่งต่อการดูแลโครงสร้างพื้นฐานให้มีความปลอดภัย ยืดหยุ่น และมีประสิทธิภาพสูงในยุคการประมวลผลแบบกระจาย

บทที่ 8

การทำงานแบบอัตโนมัติของทรัพยากรคลาวด์ (Automating Cloud Resources)

บทนำของบทเรียน (Module Introduction)

แม้ว่างานส่วนใหญ่ในระบบคลาวด์สามารถดำเนินการด้วยตนเองได้ แต่การดำเนินการในลักษณะนี้ มักใช้เวลานาน มีความเสี่ยงที่จะเกิดข้อผิดพลาด และไม่เหมาะสมสำหรับการขยายระบบในระยะยาว ทั้งยังไม่คุ้มค่ากับต้นทุน การนำเครื่องมือทางเทคโนโลยีและไฟล์มาใช้เพื่อ ทำงานแบบอัตโนมัติ จึงเป็นทางเลือกที่ดีกว่า โดยช่วยเพิ่มความรวดเร็วในการดำเนินการและลดข้อผิดพลาด

บทเรียนนี้จะกล่าวถึงการใช้ การทำงานอัตโนมัติ (Automation) และ การประสานงานอัตโนมัติ (Orchestration) ในการปรับใช้ระบบ รวมถึงการใช้เครื่องมือแบบ DevOps ที่ช่วยให้การดำเนินงานเป็นไปอย่างต่อเนื่องและมีประสิทธิภาพ

วัตถุประสงค์ของบทเรียน (Module Objectives)

ในบทเรียนนี้ ผู้เรียนจะได้:

- ศึกษาแนวทางของ Automation และ Orchestration สำหรับการปรับใช้งานระบบโดยใช้โค้ด เป็นฐาน (Code-Based Deployments)
- เรียนรู้การใช้เครื่องมือแบบ DevOps เพื่อจัดการ CI/CD (Continuous Integration/Continuous Deployment) อย่างมีประสิทธิภาพ

หน่วยที่ 1 การปรับใช้และกำหนดค่าทรัพยากรคลาวด์โดยใช้โค้ด (Deploy and Configure Cloud Resources using Code)

ในปัจจุบัน เซิร์ฟเวอร์จริง (Physical Servers) และ เซิร์ฟเวอร์เสมือน (Virtual Servers) รวมถึง อุปกรณ์เครือข่ายอย่าง เร้าเตอร์ (Routers), ไฟร์วอลล์ (Firewalls) และอุปกรณ์อื่น ๆ ถูกปรับใช้และดูแลโดย บริการจัดการการกำหนดค่าด้วยซอฟต์แวร์ (Software Configuration Management Services) แทนการทำด้วยมือแบบเดิม

1.1 Infrastructure as Code (IaC) (โครงสร้างพื้นฐานในรูปแบบโค้ด)

การนำระบบคลาวด์มาใช้จำเป็นต้องรองรับการใช้งานแบบตามต้องการ (on-demand) และ สามารถขยายขนาดได้อย่างรวดเร็ว โดยรูปแบบการบริหารจัดการระบบเครือข่ายแบบดั้งเดิม เช่น การตั้งค่า เร้าเตอร์หรือไฟร์วอลล์ มักดำเนินการด้วยมือในแต่ละอุปกรณ์ ซึ่งทำให้เกิดความล่าช้า ความผิดพลาด และไม่มี ประสิทธิภาพ

เช่น หากต้องอัปเดตเราเตอร์ 3 ตัว ผู้ดูแลระบบจะต้องเข้าไปตั้งค่าที่ละเอียดด้วยตนเอง กระบวนการลักษณะนี้ไม่เหมาะสมในระบบขนาดใหญ่หรือคลาวด์ที่ต้องการความคล่องตัว โดยเฉพาะในกรณีที่เกี่ยวข้องกับเซิร์ฟเวอร์จริง เครื่องเสมือน (VMs) หรือคอนเทนเนอร์

ส่วนประกอบของโครงสร้างพื้นฐานและการสนับสนุน (Infrastructure Components and Integration)

ในปัจจุบัน สภาพแวดล้อมของ CI/CD (Continuous Integration/Continuous Deployment) ได้นำแนวคิด DevOps มาใช้เพื่อทำให้การปรับใช้และการตั้งค่าระบบเป็นไปโดยอัตโนมัติ ผู้ดูแลระบบจะกำหนดค่าที่ต้องการในรูปแบบโค้ด แล้วนำโค้ดนั้นไปใช้กับอุปกรณ์โครงสร้างพื้นฐานต่าง ๆ แนวทางนี้ช่วยแก้ปัญหาระดับ:

- การเบี่ยงเบนของค่าตั้งต้น (Configuration Drift)
- การจัดการแบบกระจาย (Decentralized Management) ที่ไม่สอดคล้องกัน

ข้อดีของการตั้งค่าผ่านโค้ด ได้แก่:

- ทดสอบได้ง่าย
- ปรับใช้ได้เร็ว
- ขยายขนาดขึ้นหรือลงได้อย่างแม่นยำ
- ทำซ้ำได้สม่ำเสมอ

เช่น นักพัฒนาสามารถสร้าง sandbox ที่เหมือนกับระบบจริงได้ในเวลาอันสั้นและแม่นยำ

หลักการสำคัญของ IaC

- การเปลี่ยนแปลงทั้งหมดต้องทำผ่านโค้ด
- ห้ามเปลี่ยนค่าด้วยมือในอุปกรณ์โดยตรง
- หลักการที่ใช้เรียกว่า:
“โค้ดคือความจริงสูงสุด” (The code is the source of truth)

1.2 Configuration as Code (CaC) (การตั้งค่าระบบในรูปแบบโค้ด)

CaC คืออะไร?

ในขณะที่ Infrastructure as Code (IaC) มุ่งเน้นที่การจัดเตรียมอุปกรณ์โครงสร้างพื้นฐาน (เช่น เซิร์ฟเวอร์ เราเตอร์ หรือไฟร์วอลล์) แบบเสมือนจริง CaC หรือ Configuration as Code จะเน้นไปที่ การตั้งค่าและควบคุมแอปพลิเคชัน โดยตรง ทั้งนี้ CaC เป็นส่วนหนึ่งของกระบวนการ CI/CD (Continuous Integration/Continuous Deployment) และใช้ระบบเก็บไฟล์แบบศูนย์กลาง (repository) เพื่อจัดเก็บไฟล์การตั้งค่าต่าง ๆ

ตัวอย่างการใช้งาน Configuration as Code

สมมติว่าคุณกำลังปรับใช้คลัสเตอร์เว็บเซิร์ฟเวอร์เพื่อใช้งานเว็บไซต์ของบริษัท:

- เซิร์ฟเวอร์แต่ละเครื่องต้องโหลดไฟล์ HTML, CSS, และ JavaScript ที่เหมือนกัน
- เซิร์ฟเวอร์ทั้งหมดต้องเชื่อมต่อกับฐานข้อมูลเดียวกันเพื่อจัดการข้อมูลสินค้าคงคลัง

- การตั้งค่าเหล่านี้จะเก็บไว้ในไฟล์และจัดการโดยอัตโนมัติผ่าน CaC
อีกตัวอย่างหนึ่ง เช่น หากคุณใช้ **Kubernetes cluster** เพื่อให้บริการแอปพลิเคชันใหม่ CaC จะช่วยให้คุณดูแลและอัปเดตการตั้งค่าของคลัสเตอร์ได้อย่างมั่นคง โดยใช้เฉพาะโค้ดที่ผ่านการทดสอบและอนุมัติแล้วเท่านั้น

ประโยชน์ของสภาพแวดล้อมที่ใช้ IaC และ CaC

การบริหารจัดการโครงสร้างพื้นฐานและแอปพลิเคชันด้วยโค้ดนั้นมอบข้อดีหลายประการ โดยเฉพาะอย่างยิ่งในระบบคลาวด์ที่ต้องการความสามารถในการขยายตัวสูงและความพร้อมใช้งานตลอดเวลา

1. ความสามารถในการทำซ้ำ (Repeatability)

การใช้โค้ดในการกำหนดค่าซ้ำๆ ให้สามารถนำการตั้งค่ากลับมาใช้ซ้ำได้ไม่จำกัดจำนวนครั้ง เช่น:

- ปรับใช้ระบบได้อย่างรวดเร็ว
- ได้ผลลัพธ์ที่เหมือนกันทุกครั้ง (consistency)
- รองรับการทำงานอัตโนมัติ
- ขยายระบบได้ง่าย
- ลดความผิดพลาดจากมนุษย์

ตัวอย่าง:

หากคุณจัดเก็บการตั้งค่าเว็บไซต์ไว้ในโค้ด เมื่อเกิดเหตุขัดข้อง คุณสามารถนำเว็บไซต์กลับมาได้ทันที หรือเพิ่มจำนวนอินสแตนซ์ของเว็บเซิร์ฟเวอร์ในช่วงที่มีผู้ใช้มาก โดยไม่ต้องตั้งค่าใหม่ทีละเครื่อง

2. การตรวจจับความเปลี่ยนแปลงของการตั้งค่า (Configuration Drift Detection)

Configuration Drift คือการที่อุปกรณ์หรือระบบเปลี่ยนแปลงออกจากค่าตั้งต้นที่กำหนดไว้ในโค้ด เช่น:

- ระบบปฏิบัติการไม่ตรงเวอร์ชัน
- แพตช์ของแอปพลิเคชันหรือไ/drเวอร์ไม่ตรงกัน
- ระบบ staging และ production มีค่าที่ต่างกัน

สาเหตุของ Configuration Drift:

- มีการเปลี่ยนแปลงโดยไม่วางแผน
- ไม่ได้บันทึกการเปลี่ยนแปลง
- มีการติดตั้ง hotfix โดยตรง
- ผู้ดูแลระบบปรับค่าด้วยตนเองโดยไม่แจ้ง

1.3 แนวทางปฏิบัติในการกำหนดเวอร์ชัน (Versioning Practices)

การติดตามและจัดการเวอร์ชันของซอฟต์แวร์เป็นสิ่งสำคัญ เพราะช่วยให้ทั้งนักพัฒนาและผู้ใช้เข้าใจคุณลักษณะ ความสามารถ และการปรับปรุงของแอปพลิเคชันในแต่ละเวอร์ชัน โดยการกำหนดเวอร์ชันควรเป็นส่วนหนึ่งของการบันการพัฒนาซอฟต์แวร์ตั้งแต่ต้น

แนวทางสำคัญในการกำหนดเวอร์ชันมีดังนี้:

- มีความสม่ำเสมอ: เลือกรูปแบบการกำหนดเวอร์ชันแบบใดแบบหนึ่งและยึดตามนั้นเพื่อความชัดเจน และไม่สับสน
- แสดงคุณสมบัติของเวอร์ชัน: แจ้งให้ทราบว่าในแต่ละเวอร์ชันมีคุณสมบัติใหม่หรือมีการแก้ไขข้อบกพร่องอะไรบ้าง
- อธิบายรูปแบบเวอร์ชัน: ให้ผู้ใช้และนักพัฒนาเข้าใจวิธีอ่านรหัสเวอร์ชัน

หนึ่งในรูปแบบที่ได้รับความนิยมคือ Semantic Versioning ซึ่งใช้รูปแบบ Major.Minor.Patch

เช่น 6.10.9 หมายถึง:

- 6 = เวอร์ชันหลัก (มีการเปลี่ยนแปลงครั้งใหญ่)
- 10 = เวอร์ชันย่อย (เพิ่มฟีเจอร์แบบไม่กระทบโครงสร้างเดิม)
- 9 = แพตช์ (แก้ไขข้อบกพร่อง)

ความปลอดภัย เป็นอีกเหตุผลสำคัญที่ต้องติดตามเวอร์ชันอย่างใกล้ชิด เพราะเวอร์ชันล่าสุดมักรวม การอัปเดตด้านความปลอดภัยที่สำคัญไว้แล้ว

หมายเหตุ: ในที่นี้ “แนวทางปฏิบัติในการกำหนดเวอร์ชัน” หมายถึงการตั้งชื่อและติดตามเวอร์ชันของแอปพลิเคชัน ไม่ใช่ระบบควบคุมเวอร์ชัน (version control system) อย่าง Git ซึ่งจะกล่าวถึงในบทเรียนถัดไป

1.4 ขั้นตอนการทดสอบระบบ (Testing Procedures)

การทดสอบเป็นองค์ประกอบสำคัญของการจัดการการตั้งค่าด้วยโค้ดและกระบวนการ CI/CD เนื่องจากช่วยให้การปรับใช้มีความแม่นยำและนำไปเชื่อถือ

การทดสอบในแต่ละขั้นตอนของ CI/CD มีเป้าหมายเพื่อ:

- ตรวจสอบความเปลี่ยนแปลงของโค้ด
- ตรวจพบข้อผิดพลาดตั้งแต่เนิน ๆ
- ส่งผลตอบกลับให้ทีมพัฒนาเพื่อให้แก้ไขได้ทันที

ประเภทของการทดสอบประกอบด้วย:

- Unit Test:** ทดสอบแต่ละส่วนย่อยของแอปพลิเคชันโดยแยกจากกัน
- Integration Test:** ทดสอบการทำงานร่วมกันระหว่างส่วนต่าง ๆ ของแอป
- System Test:** ทดสอบระบบโดยรวมทั้งระบบที่ถูกปรับใช้
- Acceptance Test:** ทดสอบเพื่อตรวจสอบว่าแอปพลิเคชันตรงตามความต้องการทางธุรกิจหรือไม่

ประโยชน์ของการทดสอบ:

ช่วยให้มั่นใจได้ว่าแอปพลิเคชันสามารถทำงานได้ตามที่ออกแบบไว้และบรรลุเป้าหมายหลักของการพัฒนา

1.5 แนวทางการจัดทำเอกสาร (Documentation Practices)

การจัดทำเอกสารถือเป็นองค์ประกอบสำคัญในการพัฒนาแอปพลิเคชัน และยิ่งมีความสำคัญมากยิ่งขึ้นในสภาพแวดล้อมที่ใช้แนวคิด Continuous Integration/Continuous Deployment (CI/CD)

การจัดทำเอกสารช่วยให้ทุกคนที่เกี่ยวข้องกับแอปพลิเคชัน ไม่ว่าจะเป็นนักพัฒนา ผู้ดูแลระบบ หรือผู้ใช้งาน สามารถเข้าใจแนวทางการพัฒนา การดูแลจัดการ และการใช้งานแอปพลิเคชันได้อย่างถูกต้อง

อย่างไรก็ตาม หลายองค์กรยังไม่บังคับใช้แนวปฏิบัติเกี่ยวกับการจัดทำเอกสารอย่างจริงจัง ส่งผลให้เกิดปัญหาหลายประการ เช่น การนำไปใช้งานที่ไม่สมบูรณ์ ประสิทธิภาพในการพัฒนาที่ลดลง และระดับความพึงพอใจของผู้ใช้งานที่ต่ำ

เหตุผลที่การจัดทำเอกสารมักไม่ถูกดูแลหรือถูกละเลย ได้แก่:

- เข้าใจว่าการเขียนเอกสารเป็นเรื่องเสียเวลา
- ขาดความเข้าใจในตัวซอฟต์แวร์หรือระบบ
- ไม่มีแรงจูงใจหรือกระบวนการที่เอื้อต่อการจัดทำเอกสาร
- เครื่องมือสำหรับการจัดทำเอกสารไม่มีประสิทธิภาพหรือใช้งานยาก
- ความยากลำบากในการเขียน โดยเฉพาะเมื่อจำเป็นต้องใช้หลายภาษา

การจัดการเอกสารจึงควรสนับสนุนเข้ากับกระบวนการพัฒนา CI/CD และควรกำหนดให้เป็นข้อกำหนดพื้นฐานในการพิจารณาอยู่รับซอฟต์แวร์

1.6 รูปแบบของโค้ด (Code Formats)

ในการใช้งาน Infrastructure as Code (IaC) และเครื่องมือ DevOps มักมีการใช้โครงสร้างไฟล์หลักอยู่สองประเภท คือ YAML (Yet Another Markup Language) และ JSON (JavaScript Object Notation) ซึ่งทั้งสองรูปแบบถูกออกแบบมาเพื่อใช้ถ่ายทอดข้อมูลให้คอมพิวเตอร์นำไปใช้ในการเตรียมข้อมูลหรือกำหนดค่าระบบ

ตัวอย่างเช่น ผู้ดูแลระบบคลาวด์อาจใช้ไฟล์ YAML เพื่อกำหนดให้ Ansible ติดตั้งและตั้งค่าเครื่องเสมือน (Virtual Machines)

ทั้ง YAML และ JSON มีไวยากรณ์ที่เข้มงวดแต่เข้าใจง่าย ซึ่งช่วยให้ผู้ดูแลระบบสามารถอ่านและแก้ไขได้สะดวก

1.6.1 การรู้จักไฟล์ YAML (Yet Another Markup Language)

Playbook ของ Ansible และแหล่งกำหนดค่าต่าง ๆ มักใช้รูปแบบ YAML เพราะมีโครงสร้างที่อ่านง่าย และเข้าใจง่าย ช่วยให้การกำหนดค่าของ Ansible ง่ายต่อการเขียนและดีบัก

หลักไวยากรณ์ของ YAML ที่สำคัญ:

- ไฟล์ YAML ต้องเริ่มต้นด้วยขีดสามตัว (---)
- โดยทั่วไปจะลงท้ายด้วยจุดสามตัว (...)

- ใช้โครงสร้างแบบคู่คี่/ค่า (key/value pairs)
- ใช้ลำดับรายการ (sequences) เพื่อแสดงข้อมูลแบบลิสต์
- ใช้การย่อองบรรทัด (indentation) และการเว้นบรรทัดเพื่อกำหนดโครงสร้างไฟล์
- ไม่ไว้อ่านว่าในเชิงข้อมูล แต่ต้องอ่านให้ถูกต้องตามลำดับโครงสร้าง
- ไม่รองรับแท็บ (tab) ให้ใช้อ่านว่า (spaces) เท่านั้น

YAML มีลักษณะโครงสร้างคล้ายภาษา Python จึงเป็นมิตรกับผู้ใช้ที่คุ้นเคยกับ Python
เครื่องมือที่รองรับไฟล์ YAML ได้แก่:

- Amazon Web Services (AWS) CloudFormation
- Ansible
- HashiCorp Terraform

ตัวอย่างการใช้งาน:

- ไฟล์ YAML เพื่อติดตั้ง Apache (httpd) และอัปเดตแพ็คเกจทั้งหมดผ่าน YUM บนระบบ Linux
- ไฟล์ YAML เพื่อสร้างเครื่องเสมือน Linux (Debian) สำหรับใช้ในคลัสเตอร์ของเซิร์ฟเวอร์

1.6.2 การรู้จักไฟล์ JSON (JavaScript Object Notation)

JSON เป็นรูปแบบข้อความ (text-based) สำหรับการส่งและจัดเก็บข้อมูล มีความเข้าใจง่ายและสามารถอ่านได้โดยตรง แม้ว่าจะมีโครงสร้างต่างจาก YAML

JSON ใช้วัตถุ (objects) และคุณสมบัติ (properties) ในการบรรยายข้อมูล เช่น ตัวอย่างการกำหนดคุณสมบัติของวัตถุ “apple”, “red”, “yummy”

จุดประสงค์ของ JSON:

ใช้สำหรับส่งข้อมูลระหว่างระบบ ซึ่งอาจใช้เพื่อกำหนดค่า (configuration) หรือการเติมข้อมูล (population)

JSON มีไวยากรณ์คล้ายกับภาษา JavaScript ซึ่งผู้ที่คุ้นเคยกับ JavaScript จะสามารถใช้งานได้ง่ายขึ้น

เนื่องจาก JSON เป็นข้อความล้วน (text-only) จึงสามารถใช้งานร่วมกับเครื่องมือหรือระบบต่าง ๆ ได้หลากหลาย

เครื่องมือที่รองรับ JSON ได้แก่:

- AWS CloudFormation
- Azure ARM Templates
- HashiCorp Terraform

กฎไวยากรณ์ทั่วไปของ JSON:

- ข้อมูลอยู่ในรูปแบบคู่ชื่อ/ค่า (name/value) ใช้โครงสร้าง: "field_name": "value"
- ใช้เครื่องหมายคอมมา (,) เพื่อแบ่งข้อมูล
- ใช้งานลีบปีกกา {} ครอบข้อมูลของวัตถุ (objects)

หน่วยที่ 2 การพัฒนาร่วมอย่างต่อเนื่องและการปรับใช้แบบต่อเนื่อง (Continuous Integration / Continuous Deployment Pipelines)

การทำงานอัตโนมัติและการจัดการระบบแบบออร์คेसเตอเรต (orchestration) สำหรับระบบคลาวด์ ทั้งแบบส่วนตัว (Private Cloud) และสาธารณะ (Public Cloud) มีบทบาทสำคัญในการสนับสนุนการปรับใช้ อินสแตนซ์ คอนเทนเนอร์ และแอปพลิเคชันให้สามารถดำเนินการได้อย่างรวดเร็วและสม่ำเสมอ

บทเรียนนี้จะนำเสนอเครื่องมืออัตโนมัติต่าง ๆ กลไกการควบคุมเวอร์ชัน (version control mechanisms) และแนวทางการเขียนสคริปต์ที่ช่วยให้ระบบสามารถพัฒนาและปรับใช้งานได้อย่างมีประสิทธิภาพ

2.1 การพัฒนาร่วมอย่างต่อเนื่องและการปรับใช้อย่างต่อเนื่อง (Continuous Integration / Continuous Deployment)

การปรับใช้ซอฟต์แวร์ในอัตโนมัติมักดำเนินการเป็นเวอร์ชันหลัก (major version) ซึ่งอาจใช้เวลาหลายปี และมีการเปลี่ยนแปลงจำนวนมากระหว่างแต่ละเวอร์ชัน การเผยแพร่ซอฟต์แวร์ใหม่ให้แก่ผู้ใช้งานจึงใช้เวลานานและมักต้องอาศัยการเรียนรู้จำนวนมากจากการเปลี่ยนแปลงของฟีเจอร์ใหม่

แนวคิด Continuous Integration / Continuous Deployment (CI/CD) ได้เข้ามาปรับเปลี่ยนรูปแบบดังกล่าว โดยเน้นการรวมโค้ดและการปรับใช้อย่างสม่ำเสมอด้วยระบบอัตโนมัติ กระบวนการพัฒนาและการปรับใช้จะถูกจัดลำดับให้อยู่ใน pipeline โดยแอปพลิเคชันจะถูกส่งผ่านขั้นตอนอัตโนมัติแบบเป็นลำดับ ตัวอย่างกระบวนการ:

นักพัฒนาทำการเพิ่มฟีเจอร์ใหม่ในแอปพลิเคชัน โค้ดใหม่จะถูกพัฒนาเข้ากับโค้ดฐานเดิม จากนั้นจะถูกสร้าง (build) โดยอัตโนมัติ มีการทดสอบอัตโนมัติเพื่อตรวจสอบว่าไม่มีฟังก์ชันเดิมได้เสียหาย จากนั้นเวอร์ชันใหม่จะถูกส่งไปยัง repository ซึ่งสามารถนำไปใช้งานต่อได้ทันที

กระบวนการนี้จะทำซ้ำทุกรอบที่มีการเพิ่มโค้ดใหม่เข้าระบบ

หมายเหตุ: คำว่า *Delivery* และ *Deployment* อาจใช้แทนกันได้ในบริบทนี้ โดย “Delivery” มักหมายถึง การนำซอฟต์แวร์ไปเก็บไว้ใน repository เพื่อให้ผู้ใช้งานดาวน์โหลด และ “Deployment” หมายถึง การที่ผู้ใช้นำซอฟต์แวร์ไปติดตั้งใช้งานจริง

CI/CD เชื่อมโยงอย่างใกล้ชิดกับแนวทาง DevOps และแนวคิด Infrastructure as Code (IaC) โดยเฉพาะในกรณีของการพัฒนาเว็บแอปพลิเคชันในระบบคลาวด์ ที่ต้องการความยืดหยุ่นและการปรับขนาดที่รวดเร็ว

2.2 แนวปฏิบัติด้านระบบอัตโนมัติ (Automation Practices)

ผู้ดูแลระบบคลาวด์ (หรือแม้กระทั่งผู้ดูแลเซิร์ฟเวอร์แบบดั้งเดิม) มักจะนำระบบอัตโนมัติมาใช้ในการจัดการงาน เช่น การปรับขนาดระบบ การปรับใช้แอปพลิเคชัน และการกำหนดค่าระบบ

โดยทั่วไป ระบบอัตโนมัติมักอ้างอิงถึง “งานเดียวที่สามารถทำได้” ไม่ใช่การจัดลำดับหลายขั้นตอน ต่อเนื่อง

ตัวอย่างงานที่ควรใช้ระบบอัตโนมัติ:

- การดำเนินงานประจำ: เช่น การจัดเก็บ log ไฟล์ การติดตั้งซอฟต์แวร์ หรือการจัดการบัญชีผู้ใช้
- การปรับใช้และการอัปเดต: เช่น การนำโปรแกรมหรือแอปเวอร์ชันใหม่เข้าสู่ระบบ
- การปรับขนาด (Scaling): เช่น การขยายสภาพแวดล้อม QA เพื่อทดสอบระบบ
- การปิดและรีสตาร์ต: เช่น การรีบูตระบบโดยอัตโนมัติผ่านตัวจัดการ agent ที่ทำงานหลังจากรีสตาร์ต

2.2.1 การสร้าง API ภายใน (Internal APIs)

API (Application Programming Interface) คือกลไกสำหรับให้เข้าถึงทรัพยากรต่าง ๆ ได้ นักพัฒนาในองค์กรสามารถใช้ API ภายในเพื่อเข้าถึงข้อมูลทางธุรกิจและเร่งกระบวนการพัฒนา โดยการรวม API เข้ากับ pipeline ของระบบอัตโนมัติจะช่วยเพิ่มความรวดเร็วในการนำทรัพยากรมาใช้งาน

2.2.2 การแก้ไขปัญหาการทำงานของระบบอัตโนมัติ (Troubleshooting Automation Issues)

ปัญหาและแนวทางแก้ไขที่พบบ่อย ได้แก่:

- งานอัตโนมัติล้มเหลว:
 - เวอร์ชันของ OS หรือแอปพลิเคชันในเครื่องเป้าหมายไม่ตรงกับที่รองรับ
 - เครื่องมือบริหารจัดการคอนฟิกเวอร์ชันไม่ตรง
 - ไฟล์คอนฟิกที่ใช้ไม่เข้ากันกับเครื่องมือ
- ฟีเจอร์ที่ถูกยกเลิก (Deprecated features):
 - คอนฟิกพารามิเตอร์ที่ไม่มีอยู่แล้ว
 - ไฟล์คอนฟิกเก่าอาจมีคำสั่งหรือพารามิเตอร์ที่เลิกใช้งานแล้ว
- การปรับใช้ล้มเหลวในขั้นตอนการตรวจสอบ (Validation):
 - ตรวจสอบว่า pipeline ดำเนินไปถึงขั้นตอนใดก่อนเกิดปัญหา
 - ตรวจสอบ log ของซอฟต์แวร์อัตโนมัติและระบบปฏิบัติการ
- เกิดข้อผิดพลาดด้านการพิสูจน์ตัวตน (Authentication Errors):
 - SSH key ไม่ตรง
 - การแมปบัญชีผู้ใช้กับ service account ผิดพลาด

2.3 แนวปฏิบัติด้าน Orchestration (Orchestration Practices)

แม้คำว่า “Automation” และ “Orchestration” มักถูกใช้แทนกัน แต่ทั้งสองมีความหมายต่างกัน อย่างมีนัยสำคัญในบริบทของ DevOps, CI/CD และการบริหารระบบคลาวด์

- *Automation* หมายถึง การตั้งค่ากระบวนการหนึ่งให้ทำงานโดยไม่ต้องอาศัยการมีส่วนร่วมของมนุษย์ เช่น การสำรองข้อมูลหรือการกรองอีเมล แม้จะสามารถใช้กับงานเด็กๆ ได้ แต่ส่วนใหญ่จะถูกใช้กับงานที่มีความซับซ้อนมากกว่า
- *Orchestration* คือ การรวมกระบวนการที่เป็น *Automation* หลายขั้นตอนเข้าไว้ด้วยกันเป็นลำดับ ขั้นตอนเดียว ซึ่งสามารถทำงานต่อเนื่องได้โดยอัตโนมัติ

ตัวอย่างขั้นตอนแบบ Automation สำหรับติดตั้ง Web Server ประกอบด้วย:

1. สร้าง instance ใน Virtual Private Cloud (VPC)
2. ติดตั้งระบบปฏิบัติการ
3. ตั้งค่าระบบปฏิบัติการ
4. ติดตั้ง web server
5. คัดลอกไฟล์เว็บไซต์
6. เริ่มต้นบริการเว็บเซิร์ฟเวอร์

แต่ในกรณี Orchestration กระบวนการทั้งหมดขึ้นจะถูกรวบเป็น “workflow” เดียวที่เมื่อเริ่มทำงานแล้ว ระบบจะเรียกใช้แต่ละขั้นโดยอัตโนมัติ ไม่ต้องส่งรันแต่ละขั้นตอนแยก

2.4 การจัดลำดับขั้นตอนของ Orchestration (Manage Orchestration Sequencing)

Orchestration workflow ต้องมีการจัดลำดับที่ถูกต้อง เช่น ระบบจะต้องติดตั้ง OS ให้เสร็จสิ้นก่อน จะติดตั้ง Web Server หรือไฟล์เว็บไซต์ได้ ผู้ดูแลระบบต้องกำหนดให้แต่ละขั้นสามารถตรวจสอบสถานะก่อน ดำเนินขั้นถัดไป เช่น ต้องมีการยืนยันว่า instance ถูกสร้างก่อนที่จะติดตั้งซอฟต์แวร์

งานต่าง ๆ ใน Workflow (Workflow Tasks)

CI/CD Workflow แบ่งออกเป็น 2 ส่วนคือ Continuous Integration และ Continuous Deployment

- *Continuous Integration (CI)* ครอบคลุมงานด้านการพัฒนา การรวมโค้ด และการทดสอบ โดยผลลัพธ์ จะถูกเก็บไว้ใน repository
- *Continuous Deployment (CD)* คือการนำโค้ดที่ผ่านการทดสอบไปติดตั้งในระบบเป้าหมายตามความต้องการ หรืออัตโนมัติ

กระบวนการใน Workflow ได้แก่:

- **Code Development:** เขียนโค้ดโปรแกรมหรือไฟล์ตั้งค่า (เช่น YAML, JSON)
- **Code Integration:** รวมโค้ดของแต่ละผู้พัฒนาเข้าเป็นโค้ดเดียวกัน
- **Code Testing:** ทดสอบโค้ดเพื่อให้แน่ใจว่าไม่มีข้อผิดพลาด (มักเป็น automated tests)
- **Code Repository:** เก็บโค้ดที่ผ่านการทดสอบไว้ที่ repository เพื่อควบคุมเวอร์ชันและความปลอดภัย
- **Code Deployment:** ใช้เครื่องมือเช่น Git ทำการ build และ deploy แอปพลิเคชันจาก repository ไปยัง production

2.5 การแก้ปัญหา Orchestration (Troubleshoot Orchestration Issues)

ปัญหาที่พบได้ใน Orchestration ได้แก่:

- ลำดับ workflow ผิดพลาด
- ระบบยังไม่พร้อมก่อนเริ่มงานถัดไป
- งาน automation ข้ามได้ขั้นหนึ่งล้มเหลว

แนวทางการวิเคราะห์ ได้แก่:

- ตรวจสอบ log ของ orchestration และ automation
- ตรวจสอบว่าขั้นตอนใดเป็นขั้นสุดท้ายที่ทำสำเร็จก่อนล้มเหลว
- ตรวจสอบปัญหาด้าน authentication หรือ configuration

2.6 ผลลัพธ์จากการดำเนิน CI/CD Workflow (Deployment Artifacts)

Artifacts คือผลลัพธ์จาก CI/CD Workflow ซึ่งอาจเป็นไฟล์หรือ image ที่พร้อมใช้งานใน production ได้แก่:

- VM Images:** ใช้ได้ สเกลได้ และเชื่อมต่อได้
- Container Images:** เช่น Dockerfile สำหรับสร้าง container ที่สามารถปรับแต่งได้ง่าย
- Application Packages:** ใช้ package manager เช่น rpm, deb, yum, dnf ติดตั้งบน Linux
- Tar/Zip Files:** รวมไฟล์หลายไฟล์เข้าด้วยกันเพื่อความสะดวกในการจัดการ
- Flat Files:** ไฟล์ข้อมูลเชิงตาราง (column/row) เช่น ข้อมูลลูกค้า

2.7 Code Repositories

Repository คือพื้นที่จัดเก็บโค้ดและไฟล์ที่เกี่ยวข้องกับ CI/CD, IaC และ DevOps โดยทำหน้าที่:

- เป็นศูนย์กลางการควบคุมเวอร์ชัน
- ใช่วร์มกันได้โดยนักพัฒนาและบริการอัตโนมัติ
- รองรับการใช้งานพิเจอร์เช่น GitHub Actions สำหรับ merge, scan, และ approval
- ใช้เก็บไฟล์หลากหลาย เช่น:
 - ไฟล์ configuration
 - โค้ดโปรแกรม
 - ไฟล์ image ของ VM และ container
 - เอกสารประกอบ
 - Artifacts อื่น ๆ

แบ่งออกเป็น:

- Public Repository:** เปิดสาธารณะ เหมาะกับ open source
- Private Repository:** จำกัดสิทธิ์การเข้าถึง อาจจัดเก็บภายในองค์กรหรือบนแพลตฟอร์มคลาวด์ เช่น GitHub

2.8 การจัดการโค้ดอย่างมั่นคงปลอดภัย (Secure Code Management)

ในอดีต การจัดการค่าคอนฟิกของระบบมักทำโดยใช้สคริปต์ เช่น Bash หรือ PowerShell และในปัจจุบันเครื่องมือแบบ Infrastructure as Code (IaC) อย่าง Ansible หรือ Puppet มีบทบาทสำคัญมากขึ้น แม้จะมีจุดประสงค์คล้ายกันคือปรับแต่งการตั้งค่าเซิร์ฟเวอร์ แต่เครื่องมือเหล่านี้จัดการและโต้ตอบกับระบบแตกต่างจากสคริปต์ทั่วไป

เพื่อให้การใช้งานเครื่องมือเหล่านี้มีความปลอดภัย ควรปฏิบัติตามแนวทางด้านความมั่นคงปลอดภัยดังนี้:

ห้ามฝั่งรหัสผ่านลงในสคริปต์ (No Hardcoded Passwords)

การเปลี่ยนแปลงค่าคอนฟิกจำนวนมากต้องการการยืนยันตัวตน เช่น รหัสผ่าน หรือ Token ซึ่งไม่ควรเก็บไว้ในสคริปต์ เพราะสคริปต์มักไม่ได้ถูกเข้ารหัส และสามารถถูกเปิดอ่านหรือแก้ไขได้ง่าย

ใช้การยืนยันตัวตนด้วยกุญแจ (Key-Based Authentication)

ควรใช้ SSH แบบ key-based แทนการใช้รหัสผ่าน เพื่อให้การทำงานอัตโนมัติดำเนินไปโดยไม่ต้องป้อนข้อมูลตัวอย่าง:

Ansible ใช้ SSH ในการเชื่อมต่อกับเครื่องเป้าหมาย หากระบบปลายทางใช้ password-based authentication ระบบจะหยุดรอการป้อนรหัสผ่าน ซึ่งทำให้ automation ไม่สามารถทำงานต่อได้

ด้วย key-based authentication เครื่องต้นทางจะสร้างกุญแจคู่ (public/private key) โดย:

- กุญแจ private ถูกเก็บไว้ที่เครื่องต้นทาง
- กุญแจ public ถูกคัดลอกไปยังเครื่องปลายทาง

ข้อดี:

- กระบวนการอัตโนมัติไม่หยุดชะงักจากการรอป้อนรหัสผ่าน
- มีความปลอดภัยมากกว่าการใช้รหัสผ่าน
- ไม่ต้องฝั่งรหัสผ่านในสคริปต์

ใช้ Password Vault (Password Vaults)

รหัสผ่านหรือข้อมูลลับที่จำเป็นในการ deploy ไม่ควรฝังอยู่ในสคริปต์ แต่ควรเก็บไว้ในระบบเก็บข้อมูลลับ (Vault)

เช่น: Ansible Vault ซึ่งใช้เก็บข้อมูลลับอย่างปลอดภัย และมีการป้องกันด้วยรหัสผ่านหนึ่งชุดสำหรับการเข้าถึง

ใช้บัญชีบริการแยกต่างหาก (Use of Individual Service Accounts)

บัญชีบริการ (Service Account) คือบัญชีที่ใช้โดยระบบหรือแอปพลิเคชัน แทนที่จะใช้บัญชีผู้ใช้งานทั่วไป บัญชีบริการควรมีสิทธิ์เฉพาะเท่าที่จำเป็น เช่น:

- รัน VM หรือ Container
- รัน workload บน instance
- ใช้กับบริการต่าง ๆ เช่น web server

ใช้มาตรฐานการเขียนโค้ดที่ปลอดภัย (Adopt a Secure Coding Standard)

ผู้ดูแลระบบควรมีกฎหรือมาตรฐานในการเขียนโค้ด เช่นเดียวกับนักพัฒนา เพื่อให้มั่นใจว่าโค้ดและการกำหนดค่าเป็นไปตามหลักความมั่นคงปลอดภัย ไม่ว่าจะใช้เครื่องมือใด เช่น Ansible หรือ Docker หลักการให้น้อยที่สุดเท่าที่จำเป็น (Principle of Least Privilege)

หลักการนี้คือให้สิทธิ์น้อยที่สุดเท่าที่จำเป็นในการทำงาน เช่น:

- หากผู้ใช้เพียงต้องการอ่านไฟล์ ให้สิทธิ์ “อ่าน” เท่านั้น ไม่ใช่ “เขียน” หรือ “ลบ”
- อนุญาตเฉพาะเมื่อมีการกำหนดสิทธิอย่างชัดเจน (default deny)

นอกจากนี้ ไฟล์สคริปต์และไฟล์ automation ควรถูกป้องกันด้วย permission ที่เข้มงวด เช่น:

- เฉพาะผู้มีสิทธิ์เท่านั้นจึงจะสามารถอ่าน เขียน หรือรันได้

หน่วยที่ 3 เครื่องมือที่ใช้ในสภาพแวดล้อม DevOps (Tools Used in DevOps Environments)

ผู้ดูแลระบบ DevOps ใช้เครื่องมือหลายประเภทเพื่อจัดการกับการพัฒนา การปรับใช้ และการตรวจสอบระบบอย่างมีประสิทธิภาพ เครื่องมือเหล่านี้มีบทบาทสำคัญในแนวทางปฏิบัติของ Infrastructure as Code (IaC), Automation, และ Orchestration ซึ่งช่วยให้การกำหนดค่าระบบและบริการเป็นไปโดยอัตโนมัติ และมีความสม่ำเสมอ

3.1 เครื่องมือในสภาพแวดล้อม DevOps (DevOps Tools)

ในบริบทของการพัฒนาและปรับใช้ซอฟต์แวร์สมัยใหม่ การนำแนวคิด DevOps มาใช้ได้กลายเป็นแนวทางสำคัญในการเพิ่มความรวดเร็วและความน่าเชื่อถือในการส่งมอบบริการทางดิจิทัล โดยหนึ่งในองค์ประกอบหลักของ DevOps คือการใช้เครื่องมือเฉพาะทางที่ช่วยอำนวยความสะดวกในการจัดการระบบ การติดตั้ง การกำหนดค่า และการตรวจสอบระบบอย่างต่อเนื่อง

การบริหารจัดการระบบใน DevOps อาศัยแนวทาง โครงสร้างพื้นฐานเป็นโค้ด (Infrastructure as Code: IaC) ควบคู่กับการทำงานแบบอัตโนมัติ (Automation) และการจัดลำดับขั้นตอนอัตโนมัติ (Orchestration) เพื่อให้มั่นใจได้ว่าระบบเครือข่าย อุปกรณ์ และบริการต่าง ๆ ถูกกำหนดค่าอย่างถูกต้องและสม่ำเสมอ ตัวอย่างเครื่องมือสำคัญใน DevOps

1. Ansible

Ansible เป็นเครื่องมือจัดการการกำหนดค่าระบบและการจัดลำดับขั้นตอนแบบอัตโนมัติ (Orchestration) ซึ่งมีจุดเด่นที่สามารถทำงานได้โดยไม่ต้องติดตั้ง Agent ลงในเครื่องเป้าหมาย (Agentless) โดยใช้การเชื่อมต่อผ่านโปรโตคอล SSH

Ansible อาศัยไฟล์ข้อความในรูปแบบ YAML ซึ่งเรียกว่า Ansible Playbooks สำหรับกำหนดค่า คอนฟิกของระบบ การประมวลผลคำสั่งใน Playbook นี้จะดำเนินการโดยใช้บทเรียนเฉพาะ เช่น บทเรียน สำหรับติดตั้งซอฟต์แวร์หรือจัดการระบบเครือข่าย

Ansible รองรับระบบปฏิบัติการหลากหลาย ได้แก่ Linux, Windows และ macOS และยังสามารถ เชื่อมต่อกับบริการคลาวด์ยอดนิยม เช่น Amazon Web Services (AWS), Microsoft Azure และ Google Cloud Platform ผ่านบทเรียนเฉพาะ (เช่น ec2_instance) เพื่อบริหารจัดการเครื่องเสมือนและทรัพยากรอื่น ๆ

Ansible สามารถใช้งานได้ทั้งในระบบคลาวด์แบบสาธารณะ (Public Cloud), คลาวด์ส่วนตัว (Private Cloud) หรือแบบผสมผสาน (Hybrid Cloud)

คุณสมบัติเด่นของ Ansible ได้แก่:

- การเตรียมระบบสำหรับการใช้งาน (System Provisioning)
- การปรับใช้แอปพลิเคชัน (Application Deployment)
- การจัดการการกำหนดค่า (Configuration Management)
- การทำงานร่วมกับเครื่องมือและกระบวนการจัดลำดับขั้นตอนอัตโนมัติ (Orchestration Integration)

ข้อควรระวังในการใช้งาน Ansible

แม้ว่า Ansible จะช่วยให้การจัดการระบบเป็นไปอย่างมีประสิทธิภาพและลดข้อผิดพลาดจากมนุษย์ได้ แต่ก็ยัง มีความเสี่ยงจากการกำหนดค่าที่ผิดพลาด เช่น:

- การกำหนดนโยบายไปยังกลุ่มเชิร์ฟเวอร์ผิด (Misapplied Policies)
- เครื่องปลายทางไม่อยู่ในกลุ่มที่ได้รับการกำหนดค่า
- ไฟล์กำหนดค่ามีข้อผิดพลาดด้านไวยากรณ์หรือการพิมพ์ (Syntax Errors)

การตรวจสอบและทดสอบโดยภายในก่อนการนำไปใช้งานจริง รวมถึงการใช้กระบวนการควบคุมเวอร์ชัน (Version Control) จึงมีความสำคัญอย่างยิ่งในการใช้งาน Ansible อย่างปลอดภัยและมีประสิทธิภาพ

3.2 Ansible Playbooks

แม้ว่า Ansible จะสามารถรันคำสั่งเฉพาะกิจ (ad hoc commands) ได้ แต่จุดแข็งที่แท้จริงของ Ansible อยู่ที่ความสามารถในการกำหนดค่าระบบอย่างเป็นระบบผ่านไฟล์ที่เรียกว่า Playbook ตัวอย่างคำสั่งเฉพาะกิจของ Ansible ที่ใช้รูปแบบเชิร์ฟเวอร์ Linux มีดังนี้:

```
$ ansible server42 -a "/sbin/reboot"
```

เขียนในรูปแบบ YAML (Yet Another Markup Language) ซึ่งเป็นภาษาที่อ่านง่าย เข้าใจง่าย ช่วยให้การเขียนและการแก้ไขกำหนดค่าต่าง ๆ ของระบบเป็นไปอย่างสะดวกและลดข้อผิดพลาด

Playbook เป็นเอกสาร YAML ที่ระบุภารกิจ (Jobs) หนึ่งหรือหลายอย่างที่ Ansible ดำเนินการ กับระบบปลายทาง โดยมีองค์ประกอบหลัก ดังนี้:

- **Tasks (ภารกิจย่อย):** งานที่เฉพาะเจาะจงซึ่งต้องดำเนินการ เช่น ติดตั้งแพ็คเกจ รีสตาร์ตเซอร์วิส ฯลฯ
- **Plays (กลุ่มภารกิจ):** กลุ่มของภารกิจที่เกี่ยวข้องกัน ซึ่งจะรันบนกลุ่มโฮสต์ที่กำหนด
- **Playbook:** เอกสารหลักที่รวมหนึ่งหรือหลาย Plays เข้าไว้ด้วยกัน

Playbook สามารถกำหนดให้ทำงานบนระบบเป้าหมายหลายระบบพร้อมกันได้ ความเป็นอัตโนมัติของ Playbook ทำให้มันมีความยืดหยุ่นและมีประสิทธิภาพมากกว่าการรันคำสั่งแบบ ad hoc

Ansible Modules เป็นองค์ประกอบที่ใช้สำหรับดำเนินงานแต่ละ Task โดยมีให้เลือกใช้งานหลายพันบทเรียน ครอบคลุมหมวดหมู่ต่าง ๆ เช่น เครือข่าย (Networking), การจัดการคลาวด์ (Cloud), ความมั่นคงปลอดภัย (Security), และการกำหนดค่าระบบ (Configuration)

3.3 การจัดการคอนเทนเนอร์ด้วย Docker (Docker Container Management)

คอนเทนเนอร์ (Container) เป็นแพลตฟอร์มที่ใช้ในการรันแอปพลิเคชันแบบแยกส่วน (Isolated Environment) และมีความยืดหยุ่นสูง ทั้งยังสามารถบริหารจัดการได้โดยใช้เครื่องมืออัตโนมัติและการจัดลำดับขั้นตอน (Orchestration) เช่นเดียวกับเซิร์ฟเวอร์จีริงหรืออินสแตนซ์ในระบบคลาวด์

การใช้งานระบบอัตโนมัติร่วมกับคอนเทนเนอร์ช่วยให้การปรับใช้และอัปเดตแอปพลิเคชันมีความรวดเร็ว สม่ำเสมอ และประหยัดค่าใช้จ่าย อีกทั้งยังสามารถขยายขนาดการให้บริการได้ตามความต้องการ

Docker เป็นเอนจินหลักที่ใช้ในการจัดการคอนเทนเนอร์ โดยมีความสามารถในการสร้าง รัน และจัดการ วงจรชีวิตของคอนเทนเนอร์ได้อย่างมีประสิทธิภาพ ทั้งนี้ยังสามารถทำงานร่วมกับเครื่องมือจัดการอื่น ๆ เช่น Docker Swarm และ Kubernetes

DockerHub

DockerHub เป็นคลังเก็บภาพคอนเทนเนอร์ออนไลน์ (Online Repository) ที่ช่วยให้องค์กรสามารถจัดเก็บและเรียกใช้งานภาพคอนเทนเนอร์ (Container Images) ได้อย่างเป็นระบบ โดยสามารถผนวกกระบวนการอัปเดตเข้ากับระบบอัตโนมัติ เพื่อให้มั่นใจว่าแต่ละอิมเมจจะเป็นเวอร์ชันล่าสุดเสมอ

Docker ยังสนับสนุน การสร้างอิมเมจอัตโนมัติ (Automated Builds) ซึ่งจะเริ่มต้นกระบวนการ build ใหม่โดยอัตโนมัติเมื่อมีคัดใหม่ถูกส่งเข้าคลังเก็บ (Repository)

Dockerfile

การสร้างคอนเทนเนอร์ใน Docker ดำเนินการผ่านไฟล์ข้อความที่เรียกว่า **Dockerfile** ซึ่งภายในจะประกอบด้วยคำสั่งที่ระบุวิธีการติดตั้งและกำหนดค่าระบบภายในคอนเทนเนอร์นั้น คำสั่งพื้นฐานที่ใช้สร้างอิมเมจจาก Dockerfile ได้แก่:

docker build imagename

การใช้ Dockerfile ช่วยให้การสร้างคอนเทนเนอร์เป็นไปอย่างรวดเร็ว ยืดหยุ่น และสามารถนำกลับมาใช้ซ้ำได้อย่างมีประสิทธิภาพ

3.4 การจัดลำดับขั้นตอนของคอนเทนเนอร์ด้วย Kubernetes (Kubernetes Container Orchestration)

Kubernetes เป็นเครื่องมือโอเพนซอร์สยอดนิยมที่ใช้สำหรับการจัดลำดับขั้นตอน (orchestration) ของคอนเทนเนอร์ โดยมีจุดเด่นด้านความยืดหยุ่น การรองรับปลั๊กอิน และโครงสร้างที่สามารถสนับสนุนร่วมกับระบบอื่นได้อย่างง่ายดาย

Kubernetes (หรือเรียกโดยย่อว่า K8s) สามารถสนับสนุนเข้ากับเวิร์กโฟลว์ของระบบ Continuous Integration/Continuous Deployment (CI/CD) ที่มีอยู่เดิมได้อย่างไร้รอยต่อ เพื่อเพิ่มความสามารถด้านอัตโนมัติในการจัดการคอนเทนเนอร์

Kubernetes ช่วยให้องค์กรสามารถจัดการแอปพลิเคชันที่อยู่ในรูปแบบคอนเทนเนอร์ได้โดยอัตโนมัติ ด้วยฟีเจอร์มาตรฐานที่จำเป็นต่อการให้บริการ เช่น:

- การปรับขนาดทรัพยากรโดยอัตโนมัติ (Auto-scaling)
- การกระจายโหลด (Load Balancing)
- การจัดการพื้นที่จัดเก็บของคอนเทนเนอร์ (Container Storage Management)
- การจัดการไฟล์กำหนดค่า (Configuration Files) รวมถึงข้อมูลลับ (Secrets)

ในฐานะที่เป็นกรอบการทำงานสำหรับแอปพลิเคชันที่ใช้คอนเทนเนอร์ Kubernetes จัดกลุ่มคอนเทนเนอร์ ให้เป็นหน่วยที่เรียกว่า Pods ซึ่งประกอบด้วยคอนเทนเนอร์หนึ่งตัวหรือมากกว่าที่ใช้ทรัพยากรร่วมกัน เช่น เครือข่ายและการจัดเก็บข้อมูล Pod จึงเป็นหน่วยพื้นฐานของการจัดการภายในระบบ Kubernetes

ด้วยความเป็นโอเพนซอร์ส Kubernetes ไม่ได้กำหนดโครงสร้างหรือเครื่องมือเฉพาะที่ต้องใช้ ทำให้องค์กรสามารถปรับแต่งให้สอดคล้องกับความต้องการได้ เช่น การเสริมระบบบันทึกเหตุการณ์ (Logging)

Kubernetes สามารถใช้งานร่วมกับระบบในองค์กร (On-Premises), ระบบคลาวด์ หรือโครงสร้างแบบผสมผสาน (Hybrid Environments) ได้อย่างมีประสิทธิภาพ

3.5 ชุดเครื่องมือ ELK Stack (Elasticsearch, Logstash, และ Kibana)

ชุดเครื่องมือ ELK Stack เป็นระบบที่ประกอบด้วย 3 ส่วนหลัก ได้แก่ Elasticsearch, Logstash, และ Kibana ซึ่งทำงานร่วมกันเพื่อร่วบรวม วิเคราะห์ และแสดงผลข้อมูลจากไฟล์บันทึกเหตุการณ์ (Log Files) ของระบบ

ชุดเครื่องมือนี้มีบทบาทสำคัญในการให้ข้อมูลเชิงลึกแก่นักพัฒนา ผู้ดูแลระบบ และผู้บริหาร เพื่อประกอบการตัดสินใจโดยอาศัยข้อมูลที่แม่นยำและเข้าใจง่าย

ส่วนประกอบของ ELK Stack:

- Elasticsearch:** ระบบค้นหาและวิเคราะห์ข้อมูลซึ่งมีพื้นฐานมาจาก Apache Lucene ใช้ในการจัดเก็บข้อมูลที่รวบรวมมาเพื่อให้สามารถเข้าถึงและวิเคราะห์ได้อย่างรวดเร็ว
- Logstash:** ระบบโอเพนซอร์สสำหรับรวบรวมข้อมูล log จากระบบต่าง ๆ เช่น เชิร์ฟเวอร์ แอปพลิเคชัน และเว็บไซต์ โดยสามารถใช้ปลั๊กอินได้มากกว่า 200 รายการ เพื่อปรับแต่งข้อมูลที่นำเข้าตามความต้องการ

- **Kibana:** เครื่องมือสำหรับแสดงผลข้อมูลที่จัดเก็บไว้ใน Elasticsearch โดยสามารถสร้างกราฟต่าง ๆ เช่น กราฟแท่ง เส้น วงกลม หรือแนวโน้ม เพื่อนำเสนอข้อมูลในรูปแบบที่เข้าใจง่ายผ่านแดชบอร์ด

ลำดับการทำงานของ ELK Stack

ข้อมูลจะถูกนำเข้าด้วย Logstash → จัดเก็บใน Elasticsearch → เรียกดูและวิเคราะห์ผ่าน Kibana

ประโยชน์ของ ELK Stack

ELK Stack มีบทบาทสำคัญในการติดตามและสนับสนุนการให้บริการในระบบคลาวด์ โดยมอบความสามารถในการสังเกตการณ์ (Observability) ซึ่งช่วยให้เข้าใจระดับการให้บริการในปัจจุบัน ย้อนดูแนวโน้มในอดีต และคาดการณ์ความต้องการในอนาคตได้อย่างแม่นยำ

ประโยชน์หลัก ได้แก่:

- การตรวจสอบและจัดการระบบ (Monitoring & Management)
- การวิเคราะห์แนวโน้มล่วงหน้า (Trend Prediction)
- การแก้ไขปัญหาอย่างมีประสิทธิภาพ (Troubleshooting)
- การวิเคราะห์ความปลอดภัยและสนับสนุนระบบ SIEM (Security Information and Event Management)

หน่วยที่ 4 แนวคิดเกี่ยวกับการควบคุมเวอร์ชันของซอฟต์แวร์ (Source Control Concepts)

การจัดการโค้ด (Code Management) ครอบคลุมถึงโค้ดของแอปพลิเคชันที่เขียนโดยนักพัฒนา การจัดทำโครงสร้างโค้ดในรูปแบบ **Infrastructure as Code (IaC)** สำหรับการปรับใช้งานในระบบ และ การดำเนินงานด้าน **Configuration as Code (CaC)** เพื่อกำหนดค่าการทำงานของระบบโดยใช้โค้ด

โค้ดที่พัฒนาอยู่เสมออยู่ในมีการเปลี่ยนแปลงหรือปรับปรุงให้ดีขึ้นอย่างต่อเนื่อง ดังนั้นจึงมีความจำเป็น อย่างยิ่งที่จะต้องมีระบบติดตามและระบุว่าโค้ดเวอร์ชันใดคือเวอร์ชันที่เป็นปัจจุบันหรือ “เวอร์ชันที่เชื่อถือได้” (Authoritative Version)

4.1 การควบคุมเวอร์ชันของซอฟต์แวร์ (Source Control)

ในการดำเนินงานด้านระบบอัตโนมัติ (Automation) และการจัดการกำหนดค่า (Configuration Management) นักอาชีวไฟล์ที่ใช้ในการกำหนดค่าและการจัดเตรียมทรัพยากร (Provisioning) ดังนั้น การบริหาร จัดการไฟล์เหล่านี้อย่างระมัดระวังจึงเป็นสิ่งจำเป็น

นักอาชีวไฟล์ที่ดูแลระบบ (System Administrators) นำแนวทางปฏิบัติของนักพัฒนา ซอฟต์แวร์มาใช้ โดยเฉพาะการใช้ Git ซึ่งเป็นบริการควบคุมเวอร์ชัน (Version Control Service)

เครื่องมืออย่าง Ansible, Docker และอื่น ๆ ล้วนใช้ไฟล์ที่อาจมีผู้ใช้งานหลายคนเข้าถึง แก้ไข และอัปเดต Git ช่วยติดตามการเปลี่ยนแปลงของไฟล์เหล่านี้ พร้อมจัดการเวอร์ชันผ่านคำสั่งที่เรียบง่ายแต่ทรงพลัง

4.2 การจัดการเวอร์ชัน (Version Management)

การควบคุมเวอร์ชันมีบทบาทสำคัญในการรักษาความถูกต้องของโค้ด และในการติดตามการเปลี่ยนแปลงที่อาจส่งผลต่อฟีเจอร์ ความเสถียร หรือประสิทธิภาพ

โดยเฉพาะอย่างยิ่งในสภาพแวดล้อมการพัฒนาแบบทำงานร่วมกัน (Collaborative Development) ซึ่งมีนักพัฒนาหลายคนทำงานพร้อมกัน บางการเปลี่ยนแปลงอาจเป็นฟีเจอร์ใหม่ ในขณะที่บางส่วนอาจเป็นแพตช์เพื่อแก้ไขช่องโหว่ด้านความปลอดภัยหรือเพิ่มความเสถียร

เครื่องมือควบคุมเวอร์ชันช่วยให้สามารถติดตามและจัดการการเปลี่ยนแปลงทั้งหมดได้อย่างเป็นระบบ โดย Git เป็นหนึ่งในเครื่องมือที่ได้รับความนิยมสูงสุด

Git ทำงานโดยการติดตามการเปลี่ยนแปลงของไฟล์ในช่วงเวลาต่าง ๆ โดยใช้แนวคิดของ **repository** (พื้นที่เก็บข้อมูลกลาง) เพื่อให้สามารถ *checkout* และ *checkin* ไฟล์ พร้อมรองรับการรวมโค้ด (merge) จากนักพัฒนาหลายคนแหล่ง

4.3 แนวคิดและกรณีใช้งานของ Git (Git Concepts and Use Cases)

ซอฟต์แวร์โอเพ่นซอร์ส การจัดการสคริปต์ และแนวทาง **Infrastructure as Code (IaC)** ล้วนเป็นแนวทางที่ใช้โครงสร้างบนไฟล์ การจัดการไฟล์อย่างถูกต้องจึงเป็นสิ่งจำเป็น

แม้ว่าจะมีเครื่องมือควบคุมเวอร์ชันแบบลิขสิทธิ์อยู่หลายตัว แต่ในปี ค.ศ. 2005 Linus Torvalds ผู้พัฒนา Linux ได้เปิดตัว Git ซึ่งเป็นระบบควบคุมเวอร์ชันแบบโอเพ่นซอร์สที่ทรงประสิทธิภาพ ปัจจุบัน Git ได้กลายเป็นมาตรฐานในวงการ

Git หมายอย่างยิ่งสำหรับโครงการขนาดใหญ่ที่มีการพัฒนาแบบกระจาย (Distributed Development) โดยเฉพาะเมื่อมีนักพัฒนาหลายคนทำงานพร้อมกันในหลายส่วนของแอปพลิเคชันเดียวกัน ด้วยความเร็ว ความถูกต้องของไฟล์ และความยืดหยุ่นสูง Git จึงเหมาะสมสำหรับการพัฒนาแบบไม่เป็นเส้นตรง (nonlinear) และการทำงานแบบร่วมมือ (collaborative)

กรณีใช้งานของ Git (Git Use Cases)

1. ไฟล์กำหนดค่าที่เป็นมาตรฐาน (Standardized Configuration Files)

คุณดูแลเครื่อง Linux VM จำนวน 15 เครื่อง ซึ่งต้องตั้งค่าบริการ SSH ให้อยู่ในรูปแบบเดียวกัน ทั้งหมด คุณสามารถเก็บไฟล์กำหนดค่าไว้ใน Git repository และเมื่อมีการเปลี่ยนแปลงใด ๆ ก็สามารถกระจายไฟล์ไปยังเครื่องทั้ง 15 เครื่องได้อย่างรวดเร็วและสม่ำเสมอ

2. Infrastructure as Code (IaC)

สามารถใช้ Git เป็นแหล่งเก็บไฟล์อัตโนมัติและไฟล์จัดการระบบเครือข่ายต่าง ๆ โดยเครื่อง Ansible Server สามารถดึงไฟล์เหล่านี้จาก Git repository ได้ ช่วยให้แน่ใจว่าไฟล์ที่ใช้การปรับใช้มีความถูกต้องและสามารถเข้าถึงได้แม้ในเครือข่ายที่แยกออกมา

3. สคริปต์ของผู้ดูแลระบบ (Scripts)

ผู้ดูแลระบบสามารถจัดเก็บสคริปต์ที่ต้นของพัฒนาขึ้นใน repository กลาง ช่วยส่งเสริมการแบ่งปัน และอัปเดตเครื่องมือสคริปต์ภายในทีม เพิ่มความร่วมมือและลดความซ้ำซ้อน

4. โครงการพัฒนาแอปพลิเคชัน (Development Projects)

Git ถูกออกแบบมาเพื่อรับทีมพัฒนาที่ทำงานร่วมกันในโปรเจกต์ โดยสามารถติดตามและควบคุม เวอร์ชันของซอฟต์แวร์ได้อย่างมีประสิทธิภาพ ทั้งทีมที่ทำงานในสถานที่เดียวกันและทีมที่กระจายตัวทั่วโลก

นอกจากนักพัฒนาแล้ว Git ยังเหมาะสมสำหรับผู้ดูแลระบบคลาวด์ที่ต้องจัดการกับไฟล์กำหนดค่า และทรัพยากรอื่น ๆ ที่ต้องมีการจัดเก็บและควบคุมเวอร์ชันอย่างเป็นระบบ

4.4 คลังเก็บของ Git (Git Repositories)

การเริ่มต้นใช้งาน Git

ในการเริ่มใช้งาน Git บนเครื่องทำงาน (Workstation) เพียงเครื่องเดียว ผู้ใช้งานสามารถติดตั้ง แพ็กเกจ git ผ่านตัวจัดการแพ็กเกจ (Package Manager) ที่ใช้งานอยู่ จากนั้นสร้างไดเรกทอรีสำหรับเก็บไฟล์ โครงการ และทำการ initialize หรือเริ่มต้นคลังเก็บ Git โดยใช้คำสั่ง git init เพื่อให้สามารถบริหารจัดการได้ และเวอร์ชันได้อย่างเป็นระบบ

คลังเก็บ Git (Git Repository)

องค์ประกอบสำคัญของ Git คือ **Repository** หรือคลังเก็บ ซึ่งเป็นพื้นที่เก็บข้อมูลที่รวบรวมเวอร์ชัน ของโค้ดและไฟล์ที่เกี่ยวข้อง การควบคุมเวอร์ชันจะเกิดขึ้นภายในไดเรกทอรีนี้ โดย repository จะอยู่บน เครื่องของนักพัฒนาแต่ละคน หรือจะตั้งเป็นคลังเก็บกลาง (Centralized Repository) ก็ได้ เช่น GitHub หรือ GitLab โดยไม่จำเป็นต้องมีการตั้งค่าคลังเก็บกลางเสมอไป

องค์กรหลายแห่งอาจเลือกใช้ repository แบบส่วนตัว (Private Repository) ทั้งในรูปแบบ On-Premises หรือแบบ Cloud-Based เช่น AWS CodeCommit หรือ Azure Repos

คำสั่งสำคัญของ Git (Git Commands)

คำสั่งย่อ	ความหมาย
config	ตั้งค่าต่าง ๆ เช่น ชื่อผู้ใช้งาน หรืออีเมล
init	สร้าง repository ใหม่หรืออินเทียล์ repository เดิม
clone	คัดลอก repository จากที่อื่นมาใช้งานในเครื่องตนเอง
add	เพิ่มไฟล์เข้าสู่ระบบติดตามเวอร์ชัน
commit	บันทึกการเปลี่ยนแปลง พร้อมข้อความอธิบาย
status	ตรวจสอบสถานะของไฟล์ที่เปลี่ยนแปลง
branch	จัดการสาขา (Branch) ของโครงการ
merge	รวมการเปลี่ยนแปลงจากสาขาหนึ่งเข้าสู่สาขาหลัก

คำสั่งย่อ	ความหมาย
rebase	รวมการเปลี่ยนแปลงในลักษณะลำดับต่อเนื่องเพื่อให้ Git history ดูสะอาดขึ้น
pull	ดึงการเปลี่ยนแปลงจาก repository อื่นเข้าสู่เครื่อง吨เอง
push	อัปโหลดการเปลี่ยนแปลงจากเครื่อง吨เองไปยัง repository ปลายทาง
log	แสดงบันทึกการเปลี่ยนแปลงที่ผ่านมา
checkout	สลับไปยังสาขาที่ต้องการทำงาน
tag	กำหนดป้ายเวอร์ชันใน Git เช่น v1.0
show	แสดงข้อมูลของสาขาหรือ tag ปัจจุบัน

ตาราง 7 คำสั่งสำคัญของ Git

4.5 การแตกสาขา (Branching) และการทำงานร่วมกัน (Collaboration) ด้วย Git

การแตกสาขา (Branching)

การใช้ไฟล์ Branch ของ Git ช่วยให้ผู้พัฒนาสามารถแยกแยะภาระการทำงาน (Workstream) ออกจากสาขาหลัก เพื่อทดลองหรือพัฒนาไฟล์ใหม่ได้โดยไม่กระทบกับโค้ดหลัก

ตัวอย่างสถานการณ์:

ในกรณีที่คุณทำงานกับคลังเก็บ (Repository) ที่สร้างโดยผู้อื่น คุณสามารถสร้างสาขาใหม่ขึ้นมาเพื่อทำการปรับปรุงโค้ด แล้วจึงรวม (Merge) กลับเข้าสู่สาขาหลัก (main)

- สร้างสาขาใหม่จากโค้ดต้นฉบับ:

\$ git branch newbranch

- ทำการเปลี่ยนแปลงในสาขา newbranch และรวมการเปลี่ยนแปลงกลับเข้าสู่ main:

\$ git merge newbranch

หลังจากรวมแล้ว คุณสามารถลบสาขา newbranch ได้ เนื่องจากการเปลี่ยนแปลงได้ถูกนำไปเป็นส่วนหนึ่งของสาขาหลักแล้ว

4.6 แผนผังการทำงานของ Git (Git Process Flow)

ในกระบวนการทำงานของ Git ผู้ใช้จะต้องกับส่วนต่าง ๆ ดังนี้:

- Working Directory:** พื้นที่ทำงานหลักของผู้ใช้
- Staging Area:** พื้นที่เตรียมไฟล์ก่อนทำการ commit
- Local Repository:** คลังเก็บเวอร์ชันที่อยู่ในเครื่องผู้ใช้
- Remote Repository:** คลังเก็บเวอร์ชันบนเซิร์ฟเวอร์หรือคลาวด์

โดยมีคำสั่งหลักที่ใช้ในกระบวนการ:

- git add: ย้ายไฟล์จาก Working Directory ไปยัง Staging Area
- git commit: บันทึกไฟล์จาก Staging Area ลงใน Local Repository
- git push: ส่งการเปลี่ยนแปลงจาก Local Repository ไปยัง Remote Repository
- git pull: ดึงการเปลี่ยนแปลงจาก Remote Repository มาอยู่ Local Repository
- git checkout: สลับไปยังสาขาอื่นภายในโครงการ

การจัดการอย่างมีประสิทธิภาพระหว่างส่วนเหล่านี้ทำให้สามารถควบคุมการพัฒนาโค้ดได้อย่างมีระเบียบ และสนับสนุนการทำงานแบบทีมอย่างเต็มรูปแบบ

4.7 Git Pull Request และไฟล์ที่เกี่ยวข้อง

Git Pull Request (คำร้องขอดึงโค้ด)

หนึ่งในแนวคิดสำคัญของ Git คือ “การทำงานร่วมกัน” (Collaboration) โดยนักพัฒนาสามารถใช้ Git Pull Request เพื่อขอให้ผู้ดูแลโครงการ (Project Maintainer) ตรวจสอบการแก้ไขหรือพัฒนาโค้ด และพิจารณาอนุมัติให้รวมโค้ด (Merge) เข้ากับโครงการหลัก

ตัวอย่างเช่น:

นักพัฒนาที่ทำงานในโครงการใดเพ่นซอร์สอาจสร้างฟังก์ชันหรือคอมโพเนนต์ใหม่ และเมื่อเสร็จสิ้นก็ทำการสร้าง pull request ไปยังผู้ดูแลโครงการ เพื่อขอรวมโค้ดนั้นเข้าไปในคลังโค้ดหลัก

แหล่งตัวอย่างโครงการที่เปิดให้ใช้งาน pull request ได้แก่:

- GitHub
- GitLab
- Bitbucket
- DigitalOcean

ไฟล์ .gitignore

ไฟล์ .gitignore เป็นไฟล์พิเศษที่อยู่ภายใน repository ของ Git ใช้เพื่อระบุไฟล์หรือโฟลเดอร์ ที่ไม่ต้องการให้ Git ติดตาม (Track) หรือรวมในการ commit

ตัวอย่างเช่น:

- ไฟล์ README.txt
- รายการงานที่ยังไม่เสร็จ To-Do.txt

ถึงแม้ไฟล์เหล่านี้จะมีประโยชน์ในการพัฒนา แต่ไม่ได้จำเป็นต้องรวมอยู่ในระบบควบคุมเวอร์ชัน .gitignore จะช่วยกรองไม่ให้ไฟล์เหล่านี้ถูก commit โดยไม่ต้องใจ

ไดเรกทอรี .git

ไดเรกทอรี .git/ คือโครงสร้างพื้นฐานที่ Git ใช้สำหรับเก็บข้อมูลและเมต้าดาทาเกี่ยวกับการควบคุมเวอร์ชันทั้งหมดของโปรเจกต์ เช่น:

- ประวัติการ commit
- การตั้งค่า branch
- ข้อมูลการ merge

ไดเรกทอรีนี้จะถูกสร้างโดยอัตโนมัติเมื่อใช้คำสั่ง git init และมักอยู่ในโฟลเดอร์รากของโปรเจกต์

4.8 การตรวจสอบโค้ด (Code Review)

การตรวจสอบโค้ดคือกระบวนการประกันคุณภาพที่ให้ นักพัฒนาคนอื่น (Peer) ตรวจสอบโค้ดก่อนนำไปใช้งานจริง เพื่อ:

- ปรับปรุงคุณภาพโค้ด
- ค้นหาข้อผิดพลาดก่อนนำไปใช้งานในระบบจริง
- เสนอแนวทางปรับปรุงโค้ดให้ดีขึ้น

รูปแบบของ Code Review แบ่งเป็น 2 ประเภทหลัก:

- Static Code Review:** การวิเคราะห์โค้ดโดยไม่ต้องรันโปรแกรม เช่น ตรวจสอบไวยากรณ์ (Syntax), การใช้งานฟังก์ชันที่ไม่ปลอดภัย, หรือบักที่ทราบอยู่แล้ว
- Dynamic Code Review:** การตรวจสอบโดยรันโค้ดจริงเพื่อดูปัญหาเกี่ยวกับประสิทธิภาพการทำงานจริง หรือการเชื่อมต่อระบบภายนอก

แม้ว่าการตรวจสอบแบบ peer review จะเป็นแนวปฏิบัติมาตรฐาน แต่เทคโนโลยี AI ได้เข้ามา มีบทบาทอย่างมากในปัจจุบัน โดย AI สามารถตรวจสอบโค้ดได้อย่างรวดเร็ว มีความแม่นยำ เสมอ และสามารถขยายขอบเขตได้อย่างมีประสิทธิภาพ

ข้อควรระวัง: การพึ่งพา AI เพียงอย่างเดียวอาจทำให้ละเลยมุ่งมองเชิงลึกและบริบททางวิศวกรรมซอฟต์แวร์ ที่มนุษย์สามารถให้ได้ จึงควรใช้ AI ควบคู่กับกระบวนการรีวิวอื่น ๆ

4.9 GitHub Actions

GitHub คือ บริการโฮสต์โครงการที่ใช้ระบบควบคุมเวอร์ชัน Git โดยนักพัฒนาสามารถแบ่งปันโค้ดทั้งภายในและภายนอกองค์กร

หนึ่งในความสามารถสำคัญของ GitHub คือ GitHub Actions ซึ่งช่วยให้สามารถสร้าง “เวิร์กโฟล์ว อัตโนมัติ” ภายใน repository ได้ เช่น:

- การอนุมัติ pull request โดยอัตโนมัติ
- การ merge โค้ดโดยอัตโนมัติเมื่อเงื่อนไขตรงตามกำหนด

- การกำหนดผู้ตรวจสอบ (Reviewer) โดยอัตโนมัติ
- การตรวจสอบหรือข้อมูลลับที่อาจถูกฝังอยู่ในโค้ด
- การตรวจสอบช่องโหว่ในโค้ดอัตโนมัติ

GitHub Actions รองรับระบบปฏิบัติการและภาษาการเขียนโปรแกรมที่หลากหลาย จึงสามารถใช้งานร่วมกับสภาพแวดล้อม DevOps ส่วนใหญ่ได้โดยไม่เกิดปัญหา

หน่วยที่ 5 การเขียนสคริปต์ (Scripting)

การเขียนสคริปต์ (Scripting) เป็นกระบวนการสร้างคำสั่งที่สามารถรันแบบอัตโนมัติในรูปแบบที่กำหนดเวลาไว้ล่วงหน้า (Scheduled) และ ทำซ้ำได้อย่างสม่ำเสมอ (Repeatable) โดยการเขียนสคริปต์ทำให้คำสั่งดำเนินไปในลักษณะที่เหมือนเดิมทุกครั้งที่มีการเรียกใช้งาน

5.1 ประโยชน์ของการเขียนสคริปต์ (Scripting Benefits)

การเขียนสคริปต์ช่วยให้ผู้ดูแลระบบสามารถรวมคำสั่งต่าง ๆ ไว้ในไฟล์เดียว แล้วให้ระบบอ่านและประมวลผลคำสั่งเหล่านั้นตามลำดับที่เขียนไว้ได้โดยอัตโนมัติ เมื่อไฟล์มีความถูกต้อง คำสั่งจะถูกดำเนินการในลักษณะเดียวกันทุกครั้งที่รัน และสามารถตั้งเวลาให้รันโดยอัตโนมัติได้ เช่น การสำรองข้อมูลในช่วงเวลาที่ไม่ใช่เวลาทำการ

ข้อดีของการเขียนสคริปต์ มีดังนี้

- ความสม่ำเสมอ (Consistency):** สคริปต์จะทำงานในลักษณะเดิมทุกครั้ง
- ความรวดเร็ว (Speed):** คำสั่งภายในสคริปต์ทำงานได้เร็วกว่าเมื่อเปรียบเทียบกับการพิมพ์โดยมนุษย์
- ความแม่นยำ (Accuracy):** ลดโอกาสเกิดข้อผิดพลาดที่อาจเกิดจากมนุษย์
- การปรับแต่ง (Customization):** สามารถรวมคำสั่งต่าง ๆ เพื่อสร้างเครื่องมือเฉพาะที่ตรงกับความต้องการ
- การตั้งเวลา (Scheduling):** สามารถกำหนดให้สคริปต์เริ่มทำงานในเวลาที่กำหนด

5.2 ความสำคัญของคำอธิบายโค้ด (Script Comments)

คำอธิบายโค้ด (Comment) เป็นส่วนที่ใช้ประกอบข้อความในโค้ดโปรแกรมเพื่อให้ผู้อ่านเข้าใจเจตนาหรือการทำงานของโค้ดนั้นได้ชัดเจนยิ่งขึ้น โดยระบบจะ ไม่ประมวลผลคำอธิบายเหล่านี้

ใน Bash เครื่องหมาย # ใช้เพื่อบรุ่งข้อความที่ตามหลังจะเป็นความคิดเห็น เช่น:

```
# This script determines how many files are remaining to process in a directory.

num_file=432    # จำนวนไฟล์ที่ประมวลผลแล้ว
total_files=512 # จำนวนไฟล์ทั้งหมดที่ต้องประมวลผล
echo "There are $((total_files - num_file)) files remaining."
```

แนวปฏิบัติที่ดี:

- เขียนคำอธิบายด้านบนสุดของไฟล์เพื่ออธิบายวัตถุประสงค์ของสคริปต์
- ให้คำอธิบายเฉพาะในส่วนของโค้ดที่มีความซับซ้อน
- หลีกเลี่ยงการใส่คำอธิบายที่ไม่จำเป็นหรือซ้ำซ้อน

5.3 ไวยากรณ์ของสคริปต์ (Script Syntax)

ไวยากรณ์ (Syntax) หมายถึง กฎเกณฑ์ของภาษาเขียนโปรแกรมที่ใช้ในการจัดรูปแบบคำสั่งให้ระบบสามารถเข้าใจและทำงานได้ถูกต้อง ใน Bash ไวยากรณ์ของสคริปต์จะคล้ายคลึงกับคำสั่งที่ใช้ใน CLI (Command Line Interface) ของลินุกซ์

นอกจาก Bash แล้ว ผู้ดูแลระบบคลาวด์อาจพบเจอสคริปต์ในภาษาอื่น เช่น JavaScript, Python, หรือ PowerShell ซึ่งแต่ละภาษาจะมีไวยากรณ์เฉพาะของตน

5.4 ชนิดข้อมูล (Data Types)

ชนิดข้อมูลเป็นการระบุลักษณะของข้อมูลที่ตัวแปลงภาษา (interpreter) หรือคอมไพล์เวอร์จะใช้ในการประมวลผลและตีความหมายของข้อมูลนั้น ตัวอย่างชนิดข้อมูลทั่วไปได้แก่:

- จำนวนเต็ม (Integers): เช่น 100, -5
- ตัวอักษรหรือสายอักษร (Characters/Strings): เช่น "hello", 'A'
- ค่าความจริง (Boolean): เช่น true, false

การกำหนดชนิดข้อมูลที่ถูกต้องมีความสำคัญ โดยเฉพาะในการคำนวนหรือเงื่อนไขต่าง ๆ

5.5 ฟังก์ชันในสคริปต์ (Script Functions)

ฟังก์ชัน (Function) คือ กลุ่มคำสั่งที่สามารถนำกลับมาใช้ซ้ำได้ โดยไม่ต้องเขียนคำสั่งเดิมซ้ำหลายครั้ง ฟังก์ชันสามารถถูกเรียกใช้งานจากส่วนอื่นในสคริปต์ หรือแม้กระทั่งจากสคริปต์อื่นได้ ตัวอย่างการประกาศฟังก์ชันในภาษา Bash:

```
function my_func {  
    echo "This is a function."  
    # เพิ่มคำสั่งอื่น ๆ ตามต้องการ  
}
```

5.6 ตัวแปรในสคริปต์ (Script Variables)

ตัวแปร (Variable) คือค่าหรือข้อมูลที่สามารถเปลี่ยนแปลงได้ระหว่างการทำงานของสคริปต์ โดยค่าที่เก็บไว้ในตัวแปรสามารถถูกเรียกใช้งานในส่วนต่าง ๆ ของสคริปต์ได้อย่างสะดวก ตัวแปรช่วยให้สามารถ

ควบคุมกระบวนการทำงานขึ้น การคำนวณทางคณิตศาสตร์ และการเบรียบเที่ยบเงื่อนไขได้อย่างมีประสิทธิภาพ ซึ่งเป็นหัวใจสำคัญของการทำ ระบบอัตโนมัติ (**Automation**)

ตัวแปรส่วนใหญ่มักถูกกำหนดโดยระบบปฏิบัติการในขณะที่ผู้ใช้เข้าสู่ระบบ หรือถูกกำหนดโดยชেลล์ เมื่อเริ่มต้นใช้งาน ตัวแปรจึงเป็นองค์ประกอบหลักของสภาพแวดล้อมในชีลล์ (Shell Environment) หากต้องการเปลี่ยนรายละเอียดของสภาพแวดล้อม ผู้ใช้สามารถทำได้โดยการเปลี่ยนแปลงค่าของตัวแปร

การกำหนดค่าตัวแปร (Variable Assignment) หมายถึงการตั้งค่าเริ่มต้นให้กับตัวแปรหนึ่ง ๆ ในรูปแบบชื่อเท่ากับค่า (name=value) โดยค่าที่กำหนดอาจคงที่ หรือเปลี่ยนแปลงได้ตลอดช่วงเวลา การทำงานของสคริปต์ ทั้งนี้ จุดประสงค์หลักของการใช้ตัวแปรคือเพื่อกีบข้อมูลไว้เรียกใช้ในภายหลัง และเพื่อหลีกเลี่ยงการเขียนค่าซ้ำ ๆ ในโค้ด

ตัวอย่างการกำหนดตัวแปรในภาษา Bash:

```
my_str='Hello, World!'
```

5.7 คำสั่งควบคุมเงื่อนไขในสคริปต์ (Script Conditionals)

แม้ว่าโค้ดสคริปต์บางประเภทจะมีรูปแบบเรียบง่าย แต่พังที่แท้จริงของการเขียนสคริปต์อยู่ที่ความสามารถในการควบคุมลำดับการทำงาน (logic flow) ผ่านการกำหนดเงื่อนไขต่าง ๆ ซึ่งช่วยให้สามารถตัดสินใจและเลือกเส้นทางการประมวลผลได้อย่างชาญฉลาด

ตรรกะของสคริปต์ (Script Logic) คือการออกแบบกระบวนการตัดสินใจเพื่อกำหนดลำดับและเส้นทางของการประมวลผลคำสั่งในโค้ด โดยในภาษา Bash และภาษาโปรแกรมส่วนใหญ่ ผู้พัฒนาสามารถเขียนตรรกะได้หลากหลายรูปแบบเพื่อให้ได้ผลลัพธ์เดียวกัน โดยคำนึงถึงประสิทธิภาพและความอ่านง่ายของโค้ดเป็นหลัก

คำสั่งควบคุม (Control Statement) เป็นโครงสร้างสำคัญที่ใช้ในการกำหนดลำดับการประมวลผลของโค้ด โดยสามารถควบคุมให้คำสั่งทำงานในเส้นทางที่กำหนดตามเงื่อนไขที่กำหนดไว้

คำสั่งเงื่อนไข (Conditional Statement) เป็นคำสั่งควบคุมประเภทหนึ่งที่ใช้ในการตัดสินใจของโปรแกรม หากเงื่อนไขที่กำหนดไว้เป็นจริง (true) โปรแกรมจะทำงานตามคำสั่งในล็อกนั้น แต่หากเป็นเท็จ (false) โปรแกรมจะไม่ทำงานตามคำสั่งนั้น

หน่วยที่ 6 การบูรณาการระบบ (Integration of Systems)

การบูรณาการบริการ (Service Integration) หมายถึงการรวมระบบหลากหลายที่อยู่ต่างสถานที่เข้าด้วยกันผ่านเทคโนโลยีเว็บที่อิงตามโปรโตคอล HTTP (Hypertext Transfer Protocol) ซึ่งเป็นกลไกหลักที่ช่วยให้ระบบสามารถสื่อสารระหว่างกันได้จากระยะไกล โดยไม่จำกัดว่าระบบเหล่านั้นจะอยู่บนแพลตฟอร์มหรือเทคโนโลยีใด

การสื่อสารนี้ขึ้นเคลื่อนด้วยสถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ (Event-Driven Architecture) และชุดของอินเทอร์เฟซการเขียนโปรแกรมประยุกต์ (Application Programming Interfaces: APIs) ที่กำหนดโดยผู้ใช้งาน เช่น API ที่มีบทบาทสำคัญในการเชื่อมโยงระบบ เช่น ลูกข่าย (client) กับเซิร์ฟเวอร์ (server) หรือระหว่างบริการหนึ่งกับอีกบริการหนึ่ง

6.1 สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ (Event-Driven Architectures)

สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ (Event-Driven Architecture) คือ แนวทางการออกแบบที่กำหนดให้มี ตัวกระตุ้น (Trigger) ที่ชัดเจนเพื่อเริ่มต้นกระบวนการสื่อสารระหว่างบริการหรือแอปพลิเคชันที่แยกออกจากกัน (decoupled services/applications) แนวคิดนี้มักถูกนำมาใช้กับระบบแบบไร้เซิร์ฟเวอร์ (Serverless) หรือระบบที่ออกแบบตามแนวทางไมโครเซอร์วิส (Microservices)

ตัวอย่าง: ระบบสั่งซื้อสินค้าออนไลน์อาจถูกออกแบบโดยใช้สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ ซึ่งประกอบด้วยเหตุการณ์ต่อเนื่องกันดังนี้:

1. ผู้ซื้อเพิ่มสินค้าในตะกร้าสินค้าออนไลน์
2. ผู้ซื้อยืนยันคำสั่งซื้อจากรายการในตะกร้า
3. ระบบเรียกเก็บเงินจากบัญชีธนาคารของผู้ซื้อ
4. ระบบจัดส่งดำเนินการจัดส่งสินค้า
5. ระบบจัดส่งแจ้งผู้ซื้อว่าสินค้าได้ถูกจัดส่งแล้ว

เหตุการณ์ เช่น การเพิ่มสินค้าหรือการอัปเดตสถานะการจัดส่ง คือสิ่งที่ “กระตุ้น” ให้กระบวนการต่าง ๆ เหล่านี้เกิดขึ้นโดยอัตโนมัติ

ข้อดีของสถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์:

- สนับสนุนการพัฒนาแบบ Agile ในสภาพแวดล้อมไมโครเซอร์วิสที่แยกจากกัน
- สามารถปรับขยายระบบได้ตามความต้องการ (scalable)
- รักษาความเป็นอิสระของแต่ละไมโครเซอร์วิสได้อย่างมีประสิทธิภาพ

เทคโนโลยีสนับสนุน: การสื่อสารในระบบมักใช้ API (Application Programming Interface) และเทคโนโลยีอื่น ๆ ที่เกี่ยวข้อง เพื่อให้สามารถโต้ตอบระหว่างระบบได้อย่างมีประสิทธิภาพ

6.2 Web Services ในฐานะองค์ประกอบของระบบ

Web Services คือส่วนประกอบสำคัญของสถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ โดยใช้โครงสร้างการสื่อสารผ่านโปรโตคอล HTTP และมาตรฐาน XML เพื่อให้บริการต่าง ๆ บนอินเทอร์เน็ต เช่น การแลกเปลี่ยนข้อมูลและการทำธุรกรรมอีคอมเมิร์ซ

Web Services ใช้ API และโปรโตคอลต่าง ๆ ทำหน้าที่เป็น “เกตเวย์” เชื่อมโยงผู้ใช้งานกับทรัพยากรบนเว็บ

6.3 สถาปัตยกรรม REST และ REST API

REST (Representational State Transfer) เป็นแนวทางการออกแบบสถาปัตยกรรมที่กำหนดแนวทางการสื่อสารระหว่าง Client และ Server โดย REST API ที่ปฏิบัติตามแนวทางนี้จะสามารถเข้าถึงบริการเว็บ (Web Services) ได้อย่างปลอดภัยและน่าเชื่อถือ

ลำดับการทำงานของ REST API:

1. ไคลเอนต์ส่งคำร้องขอ (request) ไปยังเซิร์ฟเวอร์
2. เซิร์ฟเวอร์ตรวจสอบสิทธิ์การเข้าถึง
3. เซิร์ฟเวอร์ประมวลผลคำร้องขอ
4. เซิร์ฟเวอร์ตอบกลับด้วยข้อมูลที่ร้องขอ

ข้อดีของ REST API:

- สามารถขยายระบบได้จ่าย (Scalable)
- มีความยืดหยุ่น (Flexible)
- ไม่ขึ้นกับภาษาโปรแกรม (Language agnostic)
- ไม่ขึ้นกับแพลตฟอร์ม (Platform agnostic)

REST API บางครั้งเรียกว่า **RESTful API** ซึ่งหมายถึง API ที่พัฒนาโดยยึดตามหลักการของ REST

6.4 Simple Object Access Protocol (SOAP)

SOAP เป็นโปรโตคอลที่ใช้ XML สำหรับการเข้าถึง Web Services โดยมีจุดเด่นด้านความสามารถในการทำงานข้ามแพลตฟอร์ม เช่น Windows, macOS และ Linux โดย SOAP นิยมใช้ในองค์กรที่ต้องการความปลอดภัยและโครงสร้างที่แน่นอน โดยสามารถส่งข้อมูลเชิงโครงสร้างผ่าน HTTP ได้อย่างมีประสิทธิภาพ

6.5 Remote Procedure Call (RPC)

RPC เป็นแนวคิดที่มีมาอย่างยาวนานในระบบเครือข่าย โดยอนุญาตให้ระบบหนึ่งสามารถเรียกใช้โค้ดหรือฟังก์ชันที่อยู่ในพื้นที่ความจำของระบบอื่นได้ ในทางปฏิบัติ RPC มักใช้งานในลักษณะ client-server เพื่อให้ client เรียกใช้บริการหรือข้อมูลจาก server พร้อมประมวลผลเบื้องหลังที่จำเป็น

6.6 WebSockets API

WebSockets API ใช้สำหรับการสื่อสารแบบสองทาง (two-way communication) ระหว่างเว็บเบราว์เซอร์ของผู้ใช้และเซิร์ฟเวอร์ ซึ่งหมายความว่าการใช้งานแบบเรียลไทม์ เช่น อีคอมเมิร์ซ เกม หรือระบบแจ้งเตือนต่าง ๆ โดย WebSockets มีความสามารถในการตอบสนองตามเหตุการณ์ (event-driven) ได้เป็นอย่างดี

6.7 GraphQL

GraphQL เป็นภาษาสำหรับการสืบค้น API แบบโอเพ่นซอร์ส ที่ช่วยให้ระบบไอคลอนต์สามารถระบุข้อมูลที่ต้องการจาก API ได้อย่างเฉพาะเจาะจง ซึ่งต่างจาก REST ที่อาจส่งข้อมูลเกินความจำเป็น GraphQL ลดปริมาณข้อมูลที่ส่งกลับ ช่วยเพิ่มประสิทธิภาพในการสื่อสารและโหลดระบบ

GraphQL ยังสามารถนำมาใช้สร้าง schema ที่กำหนดรูปแบบข้อมูลและโครงสร้างภายในระบบ ที่ขับเคลื่อนด้วยเหตุการณ์ได้อีกด้วย

สรุปการทำงานอัตโนมัติในระบบคลาวด์ (Automating Cloud Resources)

แนวทางปฏิบัติแบบ DevOps ถือเป็นองค์ประกอบสำคัญในการบริหารจัดการระบบคลาวด์ เนื่องจากการใช้โค้ดเพื่อควบคุมการติดตั้งระบบและการกำหนดค่าต่าง ๆ ช่วยเพิ่ม ประสิทธิภาพ, ความสามารถในการขยายตัว, ความรวดเร็ว, และ ความถูกต้องแม่นยำ ในการดำเนินงาน

ปัจจุบันมีเครื่องมือจำนวนมากที่ช่วยให้นักพัฒนา DevOps และผู้ดูแลระบบคลาวด์สามารถควบคุมกระบวนการต่าง ๆ ได้อย่างมีประสิทธิภาพ เช่น:

- เครื่องมือสำหรับ ควบคุมเวอร์ชันของโค้ด และสร้าง ที่เก็บโค้ด (repository) ที่ปลอดภัย
- เครื่องมือสำหรับ การติดตั้งและการกำหนดค่าแบบอัตโนมัติ เช่น Ansible
- เทคโนโลยีการจัดการแอปพลิเคชันในรูปแบบ container เช่น Docker และ Kubernetes
- แนวทางและเครื่องมือที่เกี่ยวข้องกับกระบวนการ CI/CD (Continuous Integration / Continuous Deployment) เพื่อให้งานพัฒนาและปรับใช้เป็นไปอย่างต่อเนื่องและเชื่อมโยงกัน
- เครื่องมือสำหรับ ติดตามและตรวจสอบระบบ ซึ่งช่วยให้สามารถประเมินสุขภาพของสภาพแวดล้อมระบบคลาวด์ได้แบบเรียลไทม์

บทที่ 9

การบริหารจัดการด้านความมั่นคงปลอดภัยในระบบคลาวด์ (Implementing Security Management)

บทนำของบทเรียน

การรักษาความมั่นคงปลอดภัยของระบบคลาวด์เริ่มต้นจาก การบริหารจัดการอัตลักษณ์ (Identity Management) ซึ่งประกอบด้วยกระบวนการ พิสูจน์ตัวตน (Authentication) และ การกำหนดสิทธิ์ การเข้าถึง (Authorization) เพื่อควบคุมการใช้งานทรัพยากรในระบบคลาวด์อย่างปลอดภัย

องค์ประกอบสำคัญอีกประการหนึ่งคือ การควบคุมความมั่นคงปลอดภัย (Security Controls) เช่น การบริหารจัดการเครื่องเสมือน (VM) ในระบบคลาวด์ ซึ่งครอบคลุมถึง

- การตั้งค่ามาตรฐานเบื้องต้น (Baseline Configuration)
- การอัปเดตแพตช์ความปลอดภัย
- การรักษาความปลอดภัยของแอปพลิเคชัน
- การเพิ่มความแข็งแกร่งของระบบ (Hardening)

นอกจากนี้ การประเมินช่องโหว่และการลดความเสี่ยงถือเป็นสิ่งจำเป็น เนื่องจากช่องโหว่ (Vulnerabilities) เป็นจุดอ่อนที่ผู้ไม่หวังดีอาจใช้เป็นช่องทางในการโจมตี ส่งผลกระทบต่อประสิทธิภาพ ความมั่นคงปลอดภัย และความพร้อมใช้งานของทรัพยากรในระบบคลาวด์

การเฝ้าระวังเหตุการณ์ (Event Monitoring) ยังเป็นเครื่องมือที่ช่วยให้ผู้ดูแลระบบสามารถตรวจสอบ และระบุพฤติกรรมผิดปกติที่อาจนำไปสู่เหตุการณ์ด้านความมั่นคงปลอดภัย โดยผู้ให้บริการระบบคลาวด์มักมีเครื่องมือเหล่านี้ให้ใช้งานเพิ่มเติมจากที่มีในระบบปฏิบัติการหรืออุปกรณ์โดยตรง

วัตถุประสงค์ของบทเรียน

ในบทเรียนนี้ ผู้เรียนจะสามารถ:

- ประยุกต์ใช้เทคนิคการพิสูจน์ตัวตน การกำหนดสิทธิ์ และการตรวจสอบบัญชีผู้ใช้ (Authentication, Authorization, Accounting)
- เข้าใจการควบคุมด้านความมั่นคงปลอดภัย รวมถึงการจัดการการรับ-ส่งข้อมูลบนเครือข่าย
- ประเมินและตรวจสอบช่องโหว่ในสถานการณ์การติดตั้งระบบคลาวด์
- ตรวจจับพฤติกรรมที่น่าสงสัยเพื่อป้องกันเหตุการณ์ด้านความมั่นคงปลอดภัย

หน่วยที่ 1 แนวคิดเรื่องการบริหารจัดการอัตลักษณ์และการเข้าถึง (Identity and Access Management: IAM)

การบริหารความมั่นคงปลอดภัยเริ่มต้นจาก การระบุผู้ใช้งานที่ต้องการเข้าถึงข้อมูล ซึ่งเป็นฐานรากของการป้องกันข้อมูลในสภาพแวดล้อมระบบคลาวด์ องค์กรส่วนใหญ่จะใช้วิธีสมมติฐานระหว่าง:

- การควบคุมการเข้าถึงตามบทบาทหน้าที่ (Role-Based Access Control: RBAC) ซึ่งเป็นการกำหนดสิทธิ์ตามตำแหน่งหน้าที่ของผู้ใช้งานในองค์กร
- การควบคุมการเข้าถึงโดยอิสระ (Discretionary Access Control: DAC) ซึ่งเจ้าของทรัพยากรสามารถกำหนดสิทธิ์ให้ผู้อื่นเข้าถึงได้

1.1 การบริหารจัดการอัตลักษณ์และการเข้าถึง (Identity and Access Management: IAM)

การเข้าถึงทรัพยากรระบบตามอัตลักษณ์ผู้ใช้

การเข้าถึงทรัพยากรในระบบคอมพิวเตอร์มีพื้นฐานอยู่บนแนวคิดของ “อัตลักษณ์” (Identity) ของผู้ใช้งาน แม้ในเครื่องคอมพิวเตอร์เดียว การรันโปรแกรมหรือการจัดการไฟล์ต่าง ๆ ก็ต้องผ่านการตรวจสอบอัตลักษณ์ ในสภาพแวดล้อมเครือข่ายขนาดใหญ่ ความสำคัญของการตรวจสอบอัตลักษณ์ยิ่งทวีความสำคัญมากขึ้น

ระบบการบริหารจัดการอัตลักษณ์และการเข้าถึง (IAM) มีเป้าหมายเพื่อผสานความมั่นคงปลอดภัยเข้ากับความสะดวกในการบริหารจัดการ โดยมีวัตถุประสงค์ให้ผู้ใช้แต่ละรายสามารถใช้อัตลักษณ์เพียงหนึ่งเดียวในการเข้าถึงทรัพยากรทุกประเภทที่ได้รับอนุญาต ในขณะเดียวกัน ผู้ดูแลระบบสามารถติดตาม ตรวจสอบ และบันทึกพฤติกรรมของอัตลักษณ์ดังกล่าวได้ เช่น เวลาเข้าใช้งานเครือข่าย เวลาเข้าถึงข้อมูล และการใช้งานทรัพยากรต่าง ๆ

แม้เทคโนโลยี IAM จะถือว่ามีความสมบูรณ์ก่อนที่ระบบคลาวด์จะกลายเป็นเทคโนโลยีหลักในปัจจุบัน แต่ระบบ IAM สมัยใหม่สามารถผสานการทำงานร่วมกับระบบบีนยันต์ตัวตนภายในองค์กรที่มีอยู่เดิมได้ เช่น Microsoft Active Directory

องค์ประกอบหลักของ IAM

IAM มุ่งเน้นการดำเนินงาน 3 ประการ ได้แก่

- การพิสูจน์ตัวตน (Authentication): การที่ผู้ใช้แสดงหลักฐานเพื่อยืนยันอัตลักษณ์ของตน
- การกำหนดสิทธิ์ (Authorization): ระบบจะตรวจสอบว่าผู้ใช้นี้มีสิทธิ์เข้าถึงหรือกระทำการใดกับทรัพยากรที่กำหนดไว้
- การตรวจสอบและบันทึก (Auditing หรือ Accounting): มีการจัดเก็บบันทึกของการกระทำที่ผู้ใช้กระทำในระบบ เพื่อใช้ในการตรวจสอบย้อนหลัง

1.2 นโยบายของ IAM (IAM Policies)

การบริหารจัดการ IAM เริ่มต้นจากการกำหนดนโยบายเพื่อควบคุมการเข้าถึงข้อมูล ซึ่งนโยบายเหล่านี้สามารถดำเนินการได้ทั้งในรูปแบบทางเทคนิค เช่น การจัดการบัญชีผู้ใช้งานจากศูนย์กลาง และในรูปแบบที่ไม่ใช่ทางเทคนิค เช่น การควบคุมความปลอดภัยทางภาษาพาร์เซอร์

ผู้ใช้สามารถถูกจัดกลุ่มเพื่อความสะดวกในการบริหาร หรือถูกกำหนด “บทบาท” (Roles) ซึ่งสะท้อนตำแหน่งหน้าที่ในโครงสร้างองค์กร บางกรณียังมีการตรวจสอบอัตลักษณ์ของเครื่องคอมพิวเตอร์ด้วย

ตัวอย่างนโยบาย IAM ประกอบด้วย:

- ข้อกำหนดเกี่ยวกับการจัดการผู้ใช้และกลุ่มผู้ใช้
- การจัดการวงจรชีวิตของบัญชีผู้ใช้ เช่น การสร้าง ปรับเปลี่ยน ปิดใช้งาน หรือลบบัญชี
- ข้อกำหนดในการยืนยันตัวตนตามช่วงเวลา ตำแหน่งที่ตั้ง หรืออุปกรณ์ที่องค์กรอนุญาต
- ข้อกำหนดการยืนยันตัวตน เช่น การตั้งรหัสผ่าน การใช้ระบบยืนยันตัวตนแบบหลายปัจจัย (MFA) หรือสมาร์ตการ์ด

นโยบาย IAM มักถูกรวบอยู่ภายใต้นโยบายด้านความมั่นคงปลอดภัยขององค์กรที่มีขอบเขตครอบคลุมมากกว่า

1.3 API และ SDK

Application Programming Interface (API)

API คือ ช่องทางในการสื่อสารระหว่างบริการหรือแอปพลิเคชันผ่าน HTTP ช่วยให้สามารถดึงข้อมูลหรือ�行ทำการกับทรัพยากรได้จากภายนอกระบบ

ตัวอย่างของ API ที่ใช้กับคลาวด์ ได้แก่:

- REST API (สำหรับระบบทั่วไป)
- IoT API (สำหรับอุปกรณ์อัจฉริยะ)
- AI/ML API (สำหรับงานด้านปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง)

ควรมีการจัดการสิทธิ์การเข้าถึง API อย่างระมัดระวัง เช่น การควบคุม API Key และการเข้ารหัส

Software Development Kits (SDK)

SDK เป็นชุดเครื่องมือที่ช่วยให้นักพัฒนาสร้างแอปพลิเคชันได้สะดวกยิ่งขึ้น โดยมีทั้งไลบรารี โค้ดตัวอย่าง และเอกสารอธิบายการใช้งาน

ตัวอย่าง SDK ที่ใช้ในระบบคลาวด์:

- SDK สำหรับภาษา Java และ Python
- SDK จากผู้ให้บริการคลาวด์ เช่น AWS SDK สำหรับ Python (Boto3), JavaScript, Ruby ฯลฯ

ควรมีการรักษาความปลอดภัยในการเชื่อมต่อ SDK กับทรัพยากรคลาวด์ด้วยการจัดการสิทธิ์และการตรวจสอบอย่างรัดกุม

1.4 การเข้าถึงทรัพยากรคลาวด์อย่างมั่นคงปลอดภัย (Secure Access to Cloud Resources)

ผู้ดูแลระบบมักใช้มั่นคงต่อ กับ เครื่องเสมือนในระบบคลาวด์ (Cloud VM) โดยใช้ **Secure Shell (SSH)** และ **Remote Desktop Protocol (RDP)** ซึ่งเป็นวิธีการที่มั่นคงปลอดภัยเพื่อปรับปรุงจัดการบริการและแอปพลิเคชันที่ใช้งานอยู่ในเครื่องปลายทาง

- SSH มักใช้สำหรับการเชื่อมต่อ กับ ระบบปฏิบัติการ Linux
- RDP มักใช้สำหรับ Windows อย่างไรก็ตาม ปัจจุบันมีเครื่องมือจำนวนมากที่สามารถใช้ได้ข้ามระบบปฏิบัติการ

Secure Shell (SSH)

SSH คือ เครื่องมือสำหรับการจัดการระบบ Linux และอุปกรณ์เครือข่ายอื่น ๆ ผ่านการเชื่อมต่อที่ผ่านการพิสูจน์ตัวตนและเข้ารหัสข้อมูลอย่างปลอดภัย นอกจากนี้ SSH ยังสามารถรักษาความปลอดภัยให้กับโปรโตคอล TCP อื่น ๆ ที่ไม่มีการเข้ารหัสหรือกลไกการยืนยันตัวตนในตัวเอง

ด้วยการเติบโตอย่างรวดเร็วของระบบคลาวด์ SSH จึงกลายเป็นเครื่องมือสำคัญที่ช่วยให้ผู้ดูแลระบบสามารถเข้าถึง VM ได้อย่างปลอดภัย

ไฟล์กำหนดค่า SSH ในระบบ Linux:

1. /etc/ssh/sshd_config: ใช้กำหนดค่าฝั่ง klient (client) บนเครื่องผู้ดูแลระบบ มักไม่จำเป็นต้องปรับแต่ง
2. /etc/ssh/sshd_config: ใช้กำหนดค่าฝั่งเซิร์ฟเวอร์ (server) บนเครื่อง VM มีตัวเลือกด้านความปลอดภัยมากมาย

หมายเลขพอร์ตเริ่มต้นของ SSH คือ 22/tcp

ตัวอย่างการตั้งค่า SSH Server ที่พบบ่อย:

- ปิดการยืนยันตัวตนของ root ผ่าน SSH
- แสดงข้อความเตือนเกี่ยวกับการใช้งานที่เหมาะสม
- เปิดใช้งานการยืนยันตัวตนด้วยคีย์ (key-based authentication) เพื่อความปลอดภัยและความสะดวกผู้ให้บริการคลาวด์มักแนะนำให้ใช้ การยืนยันตัวตนแบบใช้คีย์ (SSH key-based authentication) แทนการใช้รหัสผ่าน เพื่อเพิ่มความปลอดภัย

Remote Desktop Protocol (RDP)

ระบบปฏิบัติการ Microsoft Windows ใช้ RDP ในการเข้าถึงเครื่องเสมือน (VM) ที่อยู่บนคลาวด์ โดย RDP ให้ประสบการณ์การใช้งานแบบ GUI (Graphical User Interface) ซึ่งสะดวกต่อผู้ใช้ที่คุ้นเคยกับการใช้เมาส์และหน้าต่าง

ขั้นตอนการเชื่อมต่อ RDP กับเครื่อง VM บนคลาวด์:

1. เข้าถึง VM ผ่านพอร์ตแล็บ
2. ยืนยันว่าเครื่องมี Public IP
3. ตรวจสอบว่า VM เปิดพอร์ต 3389/tcp สำหรับ IP เครื่องของผู้ใช้

จากนั้น:

1. เข้าสู่หน้า VM และเลือก “Connect”
2. เลือกการเชื่อมต่อแบบ RDP
3. ดาวน์โหลดไฟล์ .rdp และเปิดเพื่อเขื่อมต่อ

หมายเหตุ: Windows มี RDP client มาในตัว ส่วน Linux และ macOS ต้องติดตั้งแยกต่างหาก

1.5 รูปแบบการอนุญาต (Authorization Models)

การอนุญาต (Authorization) หมายถึงกระบวนการใช้ตัวตนของผู้ใช้งานเพื่อขอเข้าถึงทรัพยากรต่าง ๆ ในระบบ โดยการควบคุมการเข้าถึง (Access Control) ถือเป็นองค์ประกอบสำคัญในการบริหารจัดการความมั่นคงปลอดภัยของระบบสารสนเทศ

การควบคุมการเข้าถึงทางกายภาพและเชิงตรรกะ

- การควบคุมการเข้าถึงทางกายภาพ (Physical Access Controls) ได้แก่ การรักษาความปลอดภัยทางกายภาพ เช่น ประตูล็อก, ห้องแยก, ป้ายสำหรับแขกผู้มาเยือน ฯลฯ เพื่อป้องกันไม่ให้บุคคลที่ไม่ได้รับอนุญาตเข้าถึงอุปกรณ์หรือระบบได้
- การควบคุมการเข้าถึงเชิงตรรกะ (Logical Access Controls) เป็นการบริหารจัดการผ่านซอฟต์แวร์ เช่น การรับรองตัวตนด้วยชื่อผู้ใช้และรหัสผ่าน, การใช้กลุ่มผู้ใช้งาน, การตั้งสิทธิ์ในระบบ Windows/Linux, การตั้งค่า firewall, security groups หรือ web application firewall ในระบบคลาวด์ ประเภทของการควบคุมการเข้าถึง

1. Mandatory Access Control (MAC)

เป็นแบบจำกัดโดยอิงตามป้ายหรือระดับความลับของทรัพยากร (เช่น ไฟล์) ซึ่งถูกกำหนดโดยผู้ดูแลระบบ ผู้ใช้จะถูกกำหนดระดับชั้น และจะสามารถเข้าถึงทรัพยากรได้เฉพาะในระดับเดียวกันเท่านั้น ผู้ใช้ไม่สามารถเปลี่ยนแปลงระดับความลับเหล่านี้ได้ (ตัวอย่างเช่น SELinux ในระบบ Linux)

2. Discretionary Access Control (DAC)

เป็นแบบที่ให้เจ้าของข้อมูลสามารถกำหนดสิทธิ์ให้กับผู้อื่นได้ตามความสมัครใจ ใช้กันอย่างแพร่หลายในระบบปฏิบัติการ เช่น NTFS ของ Windows หรือสิทธิ์แบบ chmod ของ Linux โดยใช้ ACL (Access Control List)

3. Group-based Access Control

เป็นการจัดกลุ่มผู้ใช้เพื่อให้การจัดการสิทธิ์ง่ายขึ้น เช่น การกำหนดสิทธิ์ให้กับกลุ่ม “ฝ่ายขาย” แทนที่จะกำหนดทีละบุคคล ซึ่งช่วยลดความผิดพลาดและประหยัดเวลาในการจัดการ

4. Role-based Access Control (RBAC)

เป็นการกำหนดสิทธิ์ตามบทบาทหน้าที่ เช่น ผู้ใช้ที่มีบทบาท “บัญชี” จะสามารถเข้าถึงเฉพาะข้อมูลที่เกี่ยวข้องกับงานบัญชีเท่านั้น RBAC ถูกใช้ในระบบคลาวด์ เช่น Azure โดยมีองค์ประกอบ คือ resource groups, resources, roles และ identities

5. Rule-based Access Control

ใช้กฎทางธุรกิจเพิ่มเติมร่วมกับ RBAC หรือ DAC เช่น ให้สิทธิ์เฉพาะช่วงเวลา หรือจากเครือข่ายภายในเท่านั้น

1.6 รูปแบบการพิสูจน์ตัวตน (Authentication Models)

การพิสูจน์ตัวตนของผู้ใช้งาน

การพิสูจน์ตัวตน (Authentication) เป็นกระบวนการที่ระบบใช้เพื่อยืนยันตัวตนของผู้ใช้งานก่อนให้สิทธิ์เข้าถึงทรัพยากร

Local Authentication

ในระบบขนาดเล็ก เช่น คอมพิวเตอร์ส่วนบุคคล ผู้ใช้สามารถเข้าสู่ระบบโดยใช้ชื่อและรหัสผ่านที่จัดเก็บไว้ในเครื่องเดียวกัน ซึ่งเป็นวิธีที่ง่ายแต่ไม่เหมาะสมกับองค์กรขนาดใหญ่

Directory-based Authentication

ระบบจะจัดเก็บข้อมูลบัญชีผู้ใช้ในเซิร์ฟเวอร์ศูนย์กลาง เช่น Microsoft Active Directory (AD) หรือ Entra ID (สำหรับระบบคลาวด์) โดยสามารถเชื่อมโยงกับระบบภายนอกองค์กรผ่านการ sync หรือ federation ได้

Federated Authentication

เป็นระบบที่ผู้ให้บริการแอปพลิเคชัน (Relying Party) ไม่ได้ทำการยืนยันตัวตนผู้ใช้โดยตรง แต่เชื่อถือผลการพิสูจน์ตัวตนจาก Identity Provider (IdP) เช่น Google หรือ Microsoft ผู้ใช้สามารถเข้าสู่ระบบต่าง ๆ ด้วยบัญชีเดียว (Single Sign-On)

SAML (Security Assertion Markup Language)

เป็นโปรโตคอลที่ใช้ในการแลกเปลี่ยนข้อมูลการพิสูจน์ตัวตนระหว่าง IdP และ Service Provider (เช่น Microsoft 365 หรือ Salesforce)

OpenID Connect

คล้ายกับ SAML แต่พัฒนาอยู่บนพื้นฐานของ OAuth 2.0 มักใช้ในระบบเว็บสมัยใหม่ เช่น การเข้าสู่ระบบด้วย Google หรือ Facebook

Token-based Authentication

ระบบจะออก token เพื่อใช้ยืนยันตัวตนของผู้ใช้หรือลูกข่ายในแต่ละ session โดยไม่ต้องกรอกรหัสผ่านซ้ำ ตัวอย่างการใช้งาน เช่น OAuth 2.0

Multifactor Authentication (MFA)

เป็นกระบวนการพิสูจน์ตัวตนด้วยปัจจัยหลายอย่าง ได้แก่

- สิ่งที่คุณรู้ (เช่น รหัสผ่าน)
- สิ่งที่คุณมี (เช่น smart card, token)
- สิ่งที่คุณเป็น (เช่น ลายนิ้วมือ)

การใช้มากกว่าหนึ่งปัจจัยช่วยเพิ่มความปลอดภัยให้กับระบบ ตัวอย่างเช่น การเข้าสู่ระบบ AWS Management Console ด้วยรหัสผ่านและรหัส OTP

Single Sign-On (SSO)

SSO คือการที่ผู้ใช้สามารถเข้าสู่ระบบต่างๆ ได้โดยไม่ต้องเข้าสู่ระบบใหม่ในแต่ละบริการ ซึ่งช่วยลดภาระในการจดจำรหัสผ่านหลายชุด และลดโอกาสในการเขียนรหัสผ่านไว้ในที่ไม่ปลอดภัย

หน่วยที่ 2 การควบคุมความปลอดภัยในระบบคลาวด์ (Security Controls in the Cloud)

การควบคุมการเข้าถึงเครือข่ายในระบบคลาวด์ หมายถึง กลไกด้านความปลอดภัยและนโยบายที่ใช้ในการบริหารจัดการการเข้าถึงทรัพยากรในระบบคลาวด์ โดยเป็นแนวทางการรักษาความมั่นคงปลอดภัย เชิงองค์รวม ซึ่งมักผสานเทคโนโลยีความปลอดภัยหลายรูปแบบเข้าด้วยกัน

2.1 การควบคุมความปลอดภัยปลายทาง (Endpoint Protection Controls)

ระบบตรวจจับและตอบสนองปลายทาง (Endpoint Detection and Response – EDR)

ระบบ EDR มีบทบาทสำคัญในการช่วยผู้ดูแลระบบเฝ้าระวังเหตุการณ์ในเครือข่าย เพื่อตรวจสอบช่องโหว่หรือการกำหนดค่าที่ไม่ปลอดภัย โดย EDR จะทำงานอย่างต่อเนื่องในการตรวจจับการเปลี่ยนแปลงของระบบซึ่งอาจบ่งชี้ถึงการคุกคามหรือการโจมตี พร้อมทั้งรวบรวมข้อมูลเข้าสู่ศูนย์กลางเพื่อเพิ่มความแม่นยำของการวิเคราะห์

EDR ต้องติดตั้งซอฟต์แวร์เจนต์ลงในอุปกรณ์ของผู้ใช้งาน เช่น คอมพิวเตอร์ตั้งโต๊ะหรือแล็ปท็อป ซึ่งถือเป็นจุดปลายทาง (Endpoint) ที่เชื่อมต่อกับทรัพยากรของระบบคลาวด์ เช่น VPCs หรือบริการคลาวด์อื่น ๆ ความปลอดภัยของอุปกรณ์เหล่านี้จึงมีความสำคัญอย่างยิ่ง

ระบบตรวจจับและป้องกันการบุกรุกแบบไฮสต์ (Host-Based IDS/IPS)

- HIDS (Host-Based Intrusion Detection System):

เป็นระบบที่ตรวจสอบและแจ้งเตือนเมื่อพบพฤติกรรมผิดปกติหรือไม่ตรงกับรูปแบบที่กำหนดไว้ ซึ่งอาจบ่งชี้ถึงความพยายามในการบุกรุกหรือการละเมิดความปลอดภัย

- HIPS (Host-Based Intrusion Prevention System):

มีความสามารถที่เหนือกว่า HIDS โดยสามารถตอบสนองอย่างอัตโนมัติ เช่น การปรับแต่งไฟร์wall ของไฮสต์ หรือการปิดการเข้าถึงบริการบางประเภท เพื่อป้องกันการบุกรุกด้วยไม่ต้องรอการตัดสินใจจากผู้ดูแลระบบ

แม้จะเป็นระบบคลาวด์ แต่ผู้ดูแลระบบยังคงควรติดตั้งซอฟต์แวร์ประเภท Antivirus หรือ HIDS/HIPS เพื่อเสริมความมั่นคงปลอดภัย

ไฟร์วอลล์แบบซอฟต์แวร์ (Software Firewall)

ไฟร์วอลล์มีสองประเภทหลัก:

- **ไฟร์วอลล์ระดับเครือข่าย (Network Firewall):**
ใช้กรองทราฟิกระหว่างเครือข่าย เช่น ระหว่างอินเทอร์เน็ตกับระบบภายในองค์กร
- **ไฟร์วอลล์ระดับโฮสต์ (Host-Based Firewall):**
ใช้ควบคุมการเข้าถึงในระดับเครื่องเซิร์ฟเวอร์ เช่น ปิดพอร์ตที่ไม่จำเป็น เปิดเฉพาะพอร์ต 22 (SSH), 80 (HTTP), และ 443 (HTTPS) บนเว็บเซิร์ฟเวอร์ Linux
ควรตั้งค่าความปลอดภัยให้เป็น “ปฏิเสธโดยค่าเริ่มต้น” และเปิดเฉพาะบริการที่ได้รับอนุญาตอย่างชัดเจน

2.2 การตรวจสอบบันทึกเหตุการณ์ (Log and Event Monitoring)

การตรวจสอบทรัพยากรและบันทึกเหตุการณ์ในระบบเป็นวิธีสำคัญในการป้องกันการโจมตีและเพิ่มความพร้อมใช้งานของระบบ

- ระบบปฏิบัติการ Windows ใช้ Event Viewer
- ระบบ Linux ใช้ rsyslog จัดเก็บที่ /var/log
- ผู้ให้บริการระบบคลาวด์มีเครื่องมือที่สามารถจัดการ log ผ่านคอนโซลของผู้ดูแลระบบ หรือภายใน VM ของ IaaS

2.3 การตรวจสอบความถูกต้องของไฟล์ (File Integrity Monitoring)

การเปลี่ยนแปลงไฟล์อาจเกิดขึ้นได้จากความผิดพลาดของผู้ใช้ การคัดลอกข้อมูล หรือจากการโจมตีโดยมัลแวร์ การตรวจสอบความถูกต้องของไฟล์จะช่วยระบุการเปลี่ยนแปลงที่ไม่พึงประสงค์ และในบางกรณีสามารถป้องกันหรือย้อนคืนข้อมูลก่อนที่จะถูกเปลี่ยนแปลงได้

2.4 การป้องกันการสูญหายของข้อมูล (Data Loss Prevention – DLP)

DLP เป็นเทคโนโลยีที่ตรวจจับและป้องกันไม่ให้ข้อมูลสำคัญหลุดออกจากระบบโดยไม่ได้รับอนุญาตซึ่งอาจเกิดจากความตั้งใจหรือความผิดพลาดของผู้ใช้

DLP ครอบคลุมการควบคุมข้อมูลใน 3 สถานะ ได้แก่:

- ขณะใช้งาน (Data in Use)
- ขณะส่งผ่าน (Data in Transit)
- ขณะจัดเก็บ (Data at Rest)

เป้าหมายหลักของ DLP ได้แก่:

1. ระบุและทำความเข้าใจการใช้งานของข้อมูลที่เป็นความลับ
2. ใช้เทคโนโลยีเพื่อปกป้องข้อมูลโดยอัตโนมัติ

3. ตรวจสอบ, ตรวจจับ และตอบสนองต่อความพยายามในการนำข้อมูลออกทั้ง AWS, Azure และ GCP ต่างมีโซลูชัน DLP ที่พร้อมใช้งานใน Marketplace หรือผ่านเครื่องมือภายในระบบ

2.5 ระบบตรวจจับและป้องกันการบุกรุกเครือข่าย (IDS/IPS)

Intrusion Detection System (IDS):

ระบบแบบพาสซีฟ (passive) ตรวจจับพฤติกรรมที่ตรงกับรูปแบบการโจมตีที่รู้จัก และแจ้งเตือนผู้ดูแลระบบเท่านั้น โดยไม่สามารถปิดกั้นการจราจรเครือข่ายได้โดยตรง

Intrusion Prevention System (IPS):

ระบบแบบแอคทีฟ (active) ที่สามารถกรองและปิดกั้นทรัพฟิกเครือข่ายที่สงสัยว่าเป็นอันตรายได้ทันที พร้อมทั้งแจ้งเตือนและบันทึกเหตุการณ์

ความแตกต่างหลัก:

- IDS ตรวจจับและแจ้งเตือน → ไม่สามารถปิดกั้น
- IPS ตรวจจับ, แจ้งเตือน และปิดกั้นการโจมตีได้

ตัวอย่างผลลัพธ์ที่อาจเกิดขึ้น:

- **True Positive:** ตรวจพบภัยคุกคามที่มีอยู่จริง
- **False Positive:** แจ้งเตือนภัยคุกคามที่ไม่เกิดขึ้นจริง
- **False Negative:** ไม่สามารถตรวจจับภัยคุกคามที่เกิดขึ้นจริงได้

2.6 การควบคุมด้วยไฟร์วอลล์ (Firewall Controls)

ไฟร์วอลล์เป็นกลไกพื้นฐานที่สำคัญที่สุดในการรักษาความปลอดภัยเครือข่าย โดยทำหน้าที่ควบคุมการให้ของทรัพฟิกระหว่างเครือข่ายหรือเซ็กเมนต์ของเครือข่าย ผ่านการใช้กฎการควบคุมการเข้าถึง (Access Control Rules) ที่กำหนดโดยผู้ดูแลระบบ

- **ไฟร์วอลล์แบบไม่มีสถานะ (Stateless Firewall)** ใช้รายการควบคุมการเข้าถึง (ACLs) เพื่อจับคู่แพ็กเก็ตข้อมูลรายตัวกับกฎโดยไม่พิจารณาบริบทของการเชื่อมต่อทั้งหมด
- สามารถไว้ว่าที่จุดขอบเขตของเครือข่าย, ระหว่างเครื่องแม่ป้ำกับอุปกรณ์รอบข้าง, หรือระหว่างเซ็กเมนต์ภายในเครือข่าย เช่น การแยกการสื่อสารระหว่างฝ่ายบัญชีกับฝ่ายทรัพยากรบุคคล
- บริบทของคลาวด์: ใช้ควบคุมการเข้าถึงระหว่าง Virtual Private Cloud (VPC), ทรัพยากรคลาวด์แต่ละตัว, หรือจากภายนอกสู่คลาวด์

Network Access Control List (ACL)

ACL เป็นกฎที่ใช้กำหนดว่าจะอนุญาตหรือปฏิเสธการเข้าถึงของทรัพฟิกตามเกณฑ์ เช่น:

- ที่อยู่ IP ต้นทาง/ปลายทาง

- ประเภทของโปรโตคอล (เช่น TCP, UDP)
- พอร์ตปลายทาง

คุณลักษณะของ ACL:

- ไม่มีสถานะ: พิจารณาแต่ละแพ็กเก็ตอย่างอิสระ
- สามารถบันทึกการลงทะเบียนได้เพื่อใช้ในการตรวจสอบย้อนหลัง

2.7 Web Application Firewall (WAF)

WAF เป็นองค์ประกอบด้านความปลอดภัยที่สำคัญสำหรับปกป้องแอปพลิเคชันเว็บจากการโจมตี เช่น:

- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- การโจมตีแบบ DDoS

หน้าที่หลักของ WAF คือทำหน้าที่เป็นเกตเวย์ (gateway) สำหรับ HTTP traffic โดยตรวจสอบ และจับคู่กับรายการควบคุมการเข้าถึง (ACLs)

ประเภทของ ACL ที่ใช้กับ WAF:

- Allow List – อนุญาตเฉพาะการเข้าถึงต่อที่กำหนดไว้เท่านั้น
- Deny List – ปฏิเสธเฉพาะรายการที่ระบุไว้เท่านั้น อนุญาตที่เหลือ

เกณฑ์ที่สามารถนำมาใช้ใน ACL ของ WAF ได้แก่:

- ที่อยู่ IP ต้นทาง
- ภูมิภาคหรือประเทศ
- ขนาดของทรานส์มิสชัน
- คำเฉพาะใน payload
- โค้ดที่ต้องสงสัยว่าเป็นอันตราย

Virtual Patching: คือการแก้ไขจุดบกพร่องของโค้ดโดยใช้ WAF แทนที่จะปรับแต่งโค้ดของแอปพลิเคชันโดยตรง เน茫ในกรณีที่ยังไม่มีทรัพยากรในการแก้ไขโค้ดโดยตรง

ข้อควรตรวจสอบเมื่อเกิดปัญหาเกี่ยวกับ WAF:

- WAF ติดตั้งอยู่หน้าตัวเซิร์ฟเวอร์แอปพลิเคชัน
- กำหนดกฎ (Rules) อย่างถูกต้อง
- หากใช้ virtual patch ต้องแน่ใจว่า patch ถูกนำไปใช้กับ WAF ทุกตัว และใช้กับแอปพลิเคชัน เป้าหมายอย่างถูกต้อง

หน่วยที่ 3 แนวคิดเกี่ยวกับการจัดการช่องโหว่ (Vulnerability Management Concepts)

ช่องโหว่หมายถึงจุดอ่อนที่อาจถูกผู้ไม่หวังดี (malicious actors) ใช้เป็นช่องทางในการโจมตีระบบหรือแอปพลิเคชัน โดยช่องโหว่อาจปรากฏใน:

- ฟีเจอร์ของระบบปฏิบัติการ (Operating System)
- บริการที่เปิดใช้งาน (Services)
- แอปพลิเคชัน (Applications)
- กระบวนการที่ทำงานโดยอัตโนมัติ (Automated Processes)

3.1 การจัดการช่องโหว่ (Vulnerability Management)

โมเดลความรับผิดชอบร่วม (Shared Responsibility Model)

ในระบบคลาวด์แบบ Infrastructure as a Service (IaaS) ผู้ดูแลระบบขององค์กรมีหน้าที่รับผิดชอบในการรักษาความปลอดภัยของทรัพยากรที่ตนจัดสรร ไม่ว่าจะอยู่ในรูปแบบคลาวด์สาธารณะ (Public), ส่วนตัว (Private), ชุมชน (Community) หรือไฮบริด (Hybrid)

ระบบปฏิบัติการเสมือน (VM Instances)

เครื่อง VM ที่ถูกใช้งานในคลาวด์มีลักษณะไม่ต่างจากเครื่องจริง เช่น RHEL หรือ Windows Server ดังนั้น การบริหารค่ากำหนด (Configuration Management) และการตั้งค่าความปลอดภัยพื้นฐาน (Security Baseline) จึงจำเป็นอย่างยิ่ง เพื่อให้เป็นไปตามเป้าหมายของการป้องกันการสูญเสียของข้อมูล (DLP) และการให้บริการอย่างต่อเนื่อง (Availability)

ขั้นตอนของการจัดการช่องโหว่ (Vulnerability Management Steps)

1. Scan: ทำการสแกนระบบเพื่อหาช่องโหว่
2. Identify: ระบุช่องโหว่ที่พบ
3. Assess: ประเมินความเสี่ยงและความรุนแรง
4. Remediate: ดำเนินการแก้ไขหรือบรรเทาความเสี่ยง

ประเภทของการสแกนช่องโหว่

- Credentialed Scan: ใช้บัญชีผู้ใช้งานจริง เช่น root/admin เพื่อเข้าถึงข้อมูลเชิงลึก
- Non-Credentialed Scan: จำลองการโจมตีจากภายนอกแบบไม่ใช้บัญชี
- Network-Based Scan: ตรวจสอบอุปกรณ์เครือข่ายและบริการที่เปิดอยู่
- Agent-Based Scan: ติดตั้งตัวแทน (Agent) บนเครื่องเพื่อตรวจสอบภายใน
- Service Availability Scan: ตรวจหาบริการที่ไม่pingpong ประสงค์หรือกำหนดผิดพลาด เช่น ด้วย Nmap

3.2 เครื่องมือจากผู้ให้บริการคลาวด์ (CSP Tools)

- GCP: Container Analysis – วิเคราะห์ช่องโหว่ในคอนเทนเนอร์แบบอัตโนมัติ
- AWS: Amazon Inspector – ตรวจสอบช่องโหว่ใน EC2 และการเข้าถึงเครือข่าย
- Azure: Azure Defender – ครอบคลุม VM, PaaS และ Kubernetes พร้อมระบบแจ้งเตือนภัยเครือข่าย

3.3 เครื่องมือภายนอก (Third-Party Tools)

- Burp Suite – วิเคราะห์ช่องโหว่เว็บแอป (รองรับ CI/CD)
- Nessus – ตรวจจับมัลแวร์, การละเมิดนโยบาย, ช่องโหว่ในระบบ
- Nmap / Zenmap – ตรวจจับพอร์ตและบริการที่เปิดอยู่ รวมถึงการใช้สคริปต์สำหรับทดสอบเจาะระบบ

3.4 การประเมินช่องโหว่ (Assessment)

ช่องโหว่ที่ถูกค้นพบต้องถูกวิเคราะห์โดยอิงตามมาตรฐาน เช่น:

- CVE (Common Vulnerabilities and Exposures) – ฐานข้อมูลช่องโหว่ที่ยอมรับในระดับสากล
- CVSS (Common Vulnerability Scoring System) – ระบบให้คะแนนความรุนแรงของช่องโหว่ การใช้ Risk Registry ช่วยในการจัดลำดับความสำคัญและประเมินต้นทุนในการแก้ไข

หน่วยที่ 4 การเฝ้าระวังกิจกรรมที่น่าสงสัยเพื่อระบุการโจมตีทั่วไป (Monitoring Suspicious Activities to Identify Common Attacks)

แม้ว่าเทคโนโลยีคลาวด์จะมีความยืดหยุ่นและทันสมัย แต่ระบบคลาวด์ยังคงเผชิญกับภัยคุกคามแบบเดียวกับระบบในองค์กร (on-premises systems) ไม่ว่าจะเป็นการโจมตีแบบ brute-force, privilege escalation, หรือ port scanning

4.1 การใช้ประโยชน์จากช่องโหว่ (Vulnerability Exploitation)

การโจมตีในระบบคลาวด์สามารถเกิดขึ้นจากช่องโหว่หลายประเภท ไม่ว่าจะเป็นเครื่องเสมือน (Virtual Machine: VM) ที่ไม่ได้รับการดูแลอย่างเหมาะสม แอปพลิเคชันที่ล้าสมัย หรือระบบปฏิบัติการที่ไม่ได้รับการติดตั้งแพตช์ (Patch) ล่าสุด ล้วนเป็นช่องทางที่ผู้มุ่งหวังดีอาจใช้ในการเข้าถึงและโจมตีระบบคลาวด์ได้ประเภทของช่องโหว่

ช่องโหว่ในระบบ หมายถึง จุดอ่อนที่เกิดขึ้นในระบบปฏิบัติการ แอปพลิเคชัน บริการ หรือการตั้งค่าระบบที่ไม่เหมาะสม ซึ่งอาจเกิดจากข้อผิดพลาดในโค้ดหลัก (Core Code) หรือการกำหนดค่าที่ผิดพลาดโดยทั่วไป ช่องโหว่ที่พบบ่อยได้แก่:

- **ข้อผิดพลาดของมนุษย์ (Human Error):** อาจเกิดจากนักพัฒนาระบบที่เขียนโปรแกรมผิดพลาด หรือผู้ดูแลระบบที่ตั้งค่าระบบผิด
 - วิธีการลดความเสี่ยงจากข้อผิดพลาดของมนุษย์คือการฝึกอบรมเชิงความปลอดภัยอย่างสม่ำเสมอ รวมถึงการใช้เครื่องมืออัตโนมัติในการจัดการและตั้งค่าระบบอย่างมีมาตรฐาน
- **ซอฟต์แวร์ที่ล้าสมัย (Outdated Software):** เมื่อระบบปฏิบัติการหรือแอปพลิเคชันหมดอายุการสนับสนุนจากผู้ผลิต จะไม่มีการอุปเดตซึ่งเพื่อแก้ไขข้อบกพร่องด้านความปลอดภัย ส่งผลให้ระบบมีช่องโหว่ที่ยังไม่ได้รับการแก้ไข
 - การแก้ไขสามารถทำได้โดยการใช้ซอฟต์แวร์เวอร์ชันล่าสุด และตรวจสอบซอฟต์แวร์ภายในระบบอย่างสม่ำเสมอผ่านเครื่องมือการจัดการโครงสร้างพื้นฐานและการตรวจสอบเวอร์ชันอัตโนมัติ

4.2 วิศวกรรมสังคมและมัลแวร์ (Social Engineering and Malware)

วิศวกรรมสังคม (Social Engineering)

วิศวกรรมสังคมคือกระบวนการหลอกลวงโดยอาศัยความไว้วางใจของผู้ใช้ เพื่อให้ผู้ใช้เปิดเผยข้อมูลสำคัญ ตัวอย่างเช่น การปลอมตัวเป็นเจ้าหน้าที่ฝ่ายสนับสนุนทางเทคนิค ส่งข้อความแจ้งว่าต้องใช้รหัสผ่านเพื่ออัปเดตระบบ ซึ่งอาจทำให้ผู้ใช้หลงเชื่อและเปิดเผยข้อมูลโดยไม่รู้ตัว

ฟิชชิ่ง (Phishing)

ฟิชชิ่งคือหนึ่งในเทคนิคของวิศวกรรมสังคมที่แพร่หลาย โดยการส่งอีเมลที่ดูเหมือนว่ามาจากแหล่งที่เชื่อถือได้ (เช่น ธนาคาร หรือระบบคลาวด์ขององค์กร) พร้อมลิงก์ให้กรอกข้อมูลส่วนบุคคล เช่น ชื่อผู้ใช้และรหัสผ่าน เว็บไซต์ปลอมเหล่านี้มักมีหน้าตาคล้ายเว็บไซต์จริงมาก ทำให้ผู้ใช้ไม่ทันระวังและตกเป็นเหยื่อได้ง่าย

4.3 มัลแวร์ (Malware)

มัลแวร์คือซอฟต์แวร์ที่มีเจตนาร้ายต่อระบบ ตัวอย่างได้แก่:

- **ไวรัส (Virus) และ เวิร์ม (Worm):** ทำลายไฟล์หรือแพร่กระจายตัวเองไปยังระบบอื่น
- **สปายแวร์ (Spyware) และ คีย์ล็อกเกอร์ (Keylogger):** แอบเก็บข้อมูลผู้ใช้ เช่น ข้อมูลการพิมพ์ หรือรหัสผ่าน
- **แรนซัมแวร์ (Ransomware):** เข้ารหัสไฟล์สำคัญในระบบและเรียกค่าไถ่เพื่อปลดล็อกไฟล์นั้น แรนซัมแวร์ในปัจจุบันใช้เทคนิคการเข้ารหัสแบบสมมาตร (Asymmetric Encryption) โดยจะส่ง Public Key มาอยู่ในระบบที่ติดมัลแวร์ และเก็บ Private Key ไว้เพื่อเรียกค่าไถ่ หากเหยื่อยินยอมจ่ายเงิน อาจได้รับรหัสอุดหนัต แต่ไม่มีการรับประกันว่าจะสามารถกู้คืนข้อมูลได้จริง

ผลกระทบของแรนซัมแวร์อาจรวมถึง:

- การหยุดชะงักของธุรกิจ
- สูญเสียความเชื่อมั่นจากลูกค้า
- ผลกระทบทางกฎหมาย
- การปิดกิจการ โดยเฉพาะกับธุรกิจขนาดเล็กถึงกลาง

4.4 การโจมตีแบบ DDoS, การขุดเงินดิจิทัลโดยไม่ได้รับอนุญาต และทรัพยากร้อมบี้ การโจมตีแบบ Distributed Denial of Service (DDoS)

การโจมตีแบบปฏิเสธการให้บริการแบบกระจาย (DDoS) เป็นการใช้ทรัพยากรคอมพิวเตอร์จากหลายระบบที่ติดมัลแวร์ในการโจมตีระบบเป้าหมายพร้อมกัน โดยมีวัตถุประสงค์เพื่อทำให้ระบบเป้าหมายไม่สามารถให้บริการตามปกติได้ เนื่องจากทรัพยากรถูกใช้จนหมดไปในการตอบสนองต่อคำขอจากระบบที่โจมตี

ตัวอย่าง: อาจมีเครื่องลูกข่ายหลายพันเครื่องส่งคำร้องขอเชื่อมต่อปلومไปยังเซิร์ฟเวอร์เว็บไซต์ ส่งผลให้ระบบไม่สามารถตอบสนองผู้ใช้งานจริงได้

แนวทางป้องกัน:

- ผู้ให้บริการระบบคลาวด์มักมีบริการป้องกัน DDoS ที่สามารถช่วยบรรเทาภัยคุกคามนี้ (โดยมีค่าใช้จ่ายเพิ่มเติม)
- การตรวจสอบ Log การเข้าใช้งานและการใช้ทรัพยากรสามารถช่วยระบุการโจมตี DDoS และประเมินประสิทธิภาพของระบบป้องกัน

4.5 การโจมตีแบบ Cryptojacking

Cryptojacking คือการบุกรุกระบบเพื่อแอบใช้ทรัพยากรของเครื่อง เช่น CPU และหน่วยความจำในการขุดเหรียญคริปโต (cryptocurrency) โดยไม่ได้รับอนุญาต

ผลกระทบ:

- ประสิทธิภาพระบบลดลงอย่างเห็นได้ชัด
- อาจส่งผลถึงขั้นระบบหยุดทำงาน หากมีการใช้ทรัพยากรจนเกินขีดจำกัด

แนวทางป้องกันและตรวจสอบ:

- ตรวจสอบไฟล์บันทึกระบบ (Log files)
- เฝ้าติดตามการใช้ทรัพยากร (Performance monitoring) เพื่อจับตาระบบที่มีพฤติกรรมใช้งานผิดปกติ

4.6 ทรัพยากรชอมบ์ (Zombie Resources and Instances)

ทรัพยากรชอมบ์หมายถึงทรัพยากรในระบบคลาวด์ เช่น VM, คอนเทนเนอร์, Storage buckets หรือบริการอื่น ๆ ที่ถูกละเลยและไม่มีการจัดการอีกต่อไป

ความเสี่ยง:

- ขาดการแพตช์และอัปเดตระบบความปลอดภัย
- ไม่อยู่ภายใต้การควบคุมของระบบจัดการการตั้งค่า
- อาจกลายเป็นช่องทาง (Backdoor) สำหรับผู้ไม่หวังดีในการเจาะเข้าสู่ระบบคลาวด์

แนวทางป้องกัน:

- ตรวจสอบใบแจ้งหนี้จากผู้ให้บริการคลาวด์เทียบกับทรัพยากรที่ใช้งานจริง
- บังคับใช้ระบบการตั้งชื่อและแท็กทรัพยากร (Tagging) อย่างเข้มงวด เพื่อรับ��เจ้าของหรือผู้ดูแล
- เบริยบเทียบข้อมูลจากระบบจัดการการตั้งค่ากับทรัพยากรที่ใช้งานจริงเพื่อหาสิ่งที่ไม่อยู่ภายใต้การควบคุม

4.7 เมตาดาทา (Metadata)

เมตาดาทา (Metadata) คือ “ข้อมูลเกี่ยวกับข้อมูล” ซึ่งหมายถึงข้อมูลเพิ่มเติมที่เกี่ยวข้องกับไฟล์ระบบปฏิบัติการ บริการ หรือองค์ประกอบอื่น ๆ ในระบบ เมตาดาทาอาจดูเหมือนไม่สำคัญ แต่สามารถเปิดเผยรายละเอียดที่สำคัญต่อผู้ไม่หวังดี

ตัวอย่างข้อมูลในเมตาดาทา:

- ชื่อผู้เขียนเอกสาร
- เวอร์ชันของซอฟต์แวร์ที่ใช้
- ระบบปฏิบัติการที่ใช้สร้างไฟล์
- วันที่สร้างหรือปรับปรุงเอกสารล่าสุด

ความเสี่ยง:

ผู้โจมตีอาจใช้เมตาดาทาเพื่อวิเคราะห์จุดอ่อน เช่น เวอร์ชันของระบบหรือแอปพลิเคชันที่ยังไม่ได้รับการอัปเดต และใช้ข้อมูลนั้นเพื่อเจาะระบบ

แนวทางป้องกัน:

- กำจัดเมตาดาทาก่อนเผยแพร่ไฟล์หรือเอกสาร
- ใช้เครื่องมือที่สามารถล้างหรือเข้ารหัสเมตาดาทาได้
- ฝึกอบรมผู้ใช้งานให้รู้เท่าทันความเสี่ยงจากเมตาดาทา

4.8 การตอบสนองต่อเหตุการณ์ด้านความมั่นคงปลอดภัย (Responding to Security Incidents)

องค์กรควรมีนโยบายหรือแผนตอบสนองต่อเหตุการณ์ (Incident Response Plan) ที่ชัดเจนเป็นลายลักษณ์อักษร เพื่อใช้ในการรับมือกับการโจมตีหรือเหตุการณ์ด้านความมั่นคงปลอดภัยในระบบคลาวด์ โดยควรมีแผนแยกย่อยตามประเภทของภัยคุกคามแต่ละรูปแบบ

แนวทางตอบสนองต่อเหตุการณ์ตัวอย่าง

1. การรั่วไหลหรือขโมยข้อมูล (Data Exfiltration/Breach):

- ตรวจสอบว่าข้อมูลใดถูกนำออกไป
- แจ้งเดือนผู้ที่ได้รับผลกระทบ
- แจ้งฝ่ายกฎหมายและฝ่ายทรัพยากรบุคคล หากเกี่ยวข้อง
- ดำเนินการควบคุมและกำจัดภัยคุกคาม
- เริ่มกระบวนการกู้คืนระบบ

2. เว็บไซต์ถูกเปลี่ยนเนื้อหา (Website Defacement):

- แจ้งทีมพัฒนาเว็บไซต์
- แจ้งผู้มีส่วนเกี่ยวข้อง
- แจ้งฝ่ายกฎหมายและ HR หากจำเป็น
- กู้คืนเว็บไซต์จากสำเนาข้อมูลที่เชื่อถือได้

3. การโจมตีแบบ DDoS (Distributed Denial of Service):

- แจ้งทีมเครือข่าย
- ปรับการกรอง trafic ในเราเตอร์
- แจ้งผู้ให้บริการระบบคลาวด์ (CSP)
- ตรวจสอบบันทึกการเข้าใช้งาน (Logs)

4. การติดไวรัส (Virus Infection):

- แจ้งผู้ใช้งานและผู้ดูแลระบบถึงวิธีการแพร่กระจายของไวรัส
- อัปเดตฐานข้อมูลไวรัสในระบบป้องกัน
- ล้างข้อมูลและติดตั้งระบบใหม่ (Reimage) ในเครื่องที่ได้รับผลกระทบ เนื่องจากการลบมัลแวร์อาจไม่สมบูรณ์และใช้เวลามาก

4.9 การติดตามเหตุการณ์ (Event Monitoring)

การติดตามเหตุการณ์ (Event Monitoring) คือกระบวนการเฝ้าระวังการเปลี่ยนแปลงในระบบคลาวด์ เพื่อประเมินความสอดคล้องกับการปรับใช้ที่วางแผนไว้และมาตรฐานความมั่นคงปลอดภัยที่กำหนด ตัวอย่างของเหตุการณ์ที่ควรติดตาม:

- การเพิ่มหรือลดภาระงาน (Workload)

- การเปลี่ยนแปลงการตั้งค่าที่ไม่สอดคล้องกับนโยบาย (Configuration Drift)
- กิจกรรมของบัญชีผู้ใช้
- การเปลี่ยนแปลงในการใช้งานทรัพยากร
- ความผิดปกติในเครือข่าย

เครื่องมือแนะนำ:

- **Azure Monitor:** ตรวจสอบและวิเคราะห์ข้อมูลจากแอปพลิเคชัน อุปกรณ์โครงสร้างพื้นฐาน และแพลตฟอร์มคลาวด์
- **AWS CloudWatch:** ตรวจสอบการเปลี่ยนแปลงของระบบใน AWS และตอบสนองต่อเหตุการณ์ได้แบบเรียลไทม์

4.10 การตรวจสอบการเบี่ยงเบนจากค่ามาตรฐาน (Baseline Deviation)

Security Baseline คือชุดค่ามาตรฐานขั้นต่ำที่ระบบคลาวด์ทุกระบบจะต้องปฏิบัติตาม เพื่อให้มั่นใจในความมั่นคงปลอดภัยของระบบทั้งหมด หากระบบใดเบี่ยงเบนจากค่าเหล่านี้จะต้องได้รับการตรวจสอบและปรับกลับเข้าสู่สภาพที่เหมาะสม

ตัวอย่างค่ามาตรฐานพื้นฐาน (Baseline Standards):

- เวอร์ชันและระดับแพตช์ของระบบปฏิบัติการ
- เวอร์ชันของแอปพลิเคชันที่ได้รับอนุญาตให้ใช้งาน
- การตั้งค่าความมั่นคงปลอดภัยของฐานข้อมูล
- การกำหนดค่าการเข้ารหัสข้อมูล (ทั้งระหว่างการส่งและที่จัดเก็บ)
- การตั้งค่าเครือข่าย เช่น IP address, พортที่เปิด/ปิด, โปรโตคอลที่รองรับ

การจัดการกับความเบี่ยงเบน:

- เครื่องมือจัดการการตั้งค่าแบบอัตโนมัติ เช่น Ansible สามารถช่วยจัดการค่าคอนฟิกจำนวนมากให้ตรงตาม baseline ได้
- ตรวจสอบระบบที่ติดตั้ง OS เก่าหรือไม่ได้รับการสนับสนุน เช่น Windows Server 2012 ที่ควรนำออกหรืออัปเกรด
- ทบทวน baseline เป็นระยะ เพื่อให้สอดคล้องกับมาตรฐานด้านความปลอดภัยในปัจจุบัน

เครื่องมือที่ใช้ในการจัดการ baseline:

- Azure Policy และ Azure Resource Monitor: ใช้ตรวจสอบความสอดคล้องของทรัพยากรกับนโยบายความมั่นคงปลอดภัยขององค์กรใน Microsoft Azure

สรุปการดำเนินการบริหารจัดการความมั่นคงปลอดภัย (Implementing Security Management)

การบริหารจัดการความมั่นคงปลอดภัยของระบบสารสนเทศอาศัยแนวคิดหลักที่เรียกว่า “3A” ได้แก่:

1. การพิสูจน์ตัวตน (Authentication) – เป็นกระบวนการที่ระบบใช้ในการระบุและยืนยันตัวบุคคลที่พยายามเข้าถึงทรัพยากร
2. การกำหนดสิทธิ์ (Authorization) – ใช้เพื่อระบุว่าผู้ใช้นั้นมีสิทธิ์ในการเข้าถึงทรัพยากรใดบ้าง
3. การบัญชีและติดตามกิจกรรม (Accounting) – ทำหน้าที่บันทึกกิจกรรมที่เกิดขึ้นเพื่อการตรวจสอบย้อนหลัง

การบริหารจัดการสิทธิ์เข้าถึงของผู้ใช้งานต้องพิจารณาจากแหล่งที่มาของการเชื่อมต่อ ปลายทาง โปรโตคอลที่ใช้ และค่าการตั้งค่าของระบบเพื่อกำหนดว่าจะอนุญาตหรือปฏิเสธการเข้าถึงนั้น

นอกจากนี้ การจัดการความมั่นคงปลอดภัยยังครอบคลุมถึง:

- การสแกนหาช่องโหว่ (Vulnerability Scanning) และ
- การลดความเสี่ยง (Mitigation)

เพื่อป้องกันไม่ให้ผู้ไม่ประสงค์ดีใช้ประโยชน์จากจุดอ่อนในระบบ เช่น

- การโจมตีแบบ DDoS
- การติดมัลแวร์และแรนซัมแวร์
- การหลอกลวงทางสังคม (Social Engineering)

การจัดตั้งกระบวนการ บริหารจัดการช่องโหว่ (Vulnerability Management)

การ ตรวจสอบเหตุการณ์ (Event Monitoring) และ

การ คงไว้ซึ่งค่าคอนฟิกพื้นฐาน (Baseline Configurations)

ถือเป็นองค์ประกอบสำคัญที่ช่วยลดความเสี่ยงจากช่องโหว่เหล่านี้และรักษาความมั่นคงปลอดภัยของระบบในภาพรวม

บทที่ 10

ความเข้าใจด้านการปฏิบัติตามข้อกำหนดด้านความมั่นคงปลอดภัย

และการแก้ไขปัญหา

(Comprehending Security Compliance and Troubleshooting)

บทนำของบทเรียน (Module Introduction)

องค์กรต่าง ๆ รวมถึงหน่วยงานทั่วโลกต่างๆ ได้กำหนดมาตรฐานด้านความมั่นคงปลอดภัยไว้ในรูปของกรอบการดำเนินงาน (Security Frameworks) เพื่อใช้ในการปกป้องข้อมูล บริการ และความพร้อมใช้งานของระบบ ซึ่งมาตรฐานเหล่านี้มีทั้งรูปแบบที่สมัครใจและแบบที่ต้องปฏิบัติตามโดยข้อบังคับทางกฎหมาย

ผู้ดูแลระบบคลาวด์ (Cloud Administrators) ต้องมีความสามารถในการดำเนินการภายใต้ข้อกำหนดที่หลากหลายซึ่งแตกต่างกันตามภาระ มีภาระ ประเภท และอุตสาหกรรม

นอกจากนี้ บุคลากรด้านคลาวด์ทุกคนมีบทบาทด้านความมั่นคงปลอดภัยที่หลีกเลี่ยงไม่ได้ ตั้งแต่การดำเนินการพื้นฐาน เช่น

- การเสริมความมั่นคงปลอดภัยของระบบ (Hardening)
- การลดสิทธิ์ในการเข้าถึงของผู้ใช้งาน (Least Privilege)

ไปจนถึงแนวปฏิบัติที่ซับซ้อนยิ่งขึ้นซึ่งเกี่ยวข้องกับการแก้ไขปัญหา (Troubleshooting) ในสถานการณ์ด้านความมั่นคงปลอดภัย

วัตถุประสงค์ของบทเรียน (Module Objectives)

เมื่อเรียนรู้จบเนื้อหาในบทเรียนนี้ ผู้เรียนจะสามารถ:

- ทำความเข้าใจข้อกำหนดด้าน การปฏิบัติตามมาตรฐานความมั่นคงปลอดภัยและข้อบังคับ (Security Compliance and Regulation)
- รับรู้และ นำแนวปฏิบัติที่ดีที่สุดด้านความมั่นคงปลอดภัย (Security Best Practices) ไปใช้ได้อย่างเหมาะสม
- มีทักษะในการ วิเคราะห์และแก้ไขปัญหาด้านความมั่นคงปลอดภัย (Troubleshoot Security Issues) ได้อย่างมีประสิทธิภาพ

หน่วยที่ 1 ประเด็นด้านการปฏิบัติตามข้อกำหนดและข้อบังคับ (Aspects of Compliance and Regulation)

ข้อกฎหมายและข้อบังคับ (Laws and Regulations) มีอิทธิพลอย่างสำคัญต่อเทคนิคและวิธีการบริหารจัดการระบบคลาวด์ โดยเฉพาะอย่างยิ่งในด้านของ:

- การคุ้มครองข้อมูลส่วนบุคคล
- การควบคุมการจัดเก็บและถ่ายโอนข้อมูล
- การปฏิบัติตามมาตรฐานอุตสาหกรรม

1.1 ผลกระทบของกฎหมายและข้อบังคับ (The Impact of Laws and Regulations)

แม้องค์กรทุกรูปแบบจะมีความกังวลเกี่ยวกับความมั่นคงปลอดภัยของระบบคลาวด์ แต่บางอุตสาหกรรม เช่น สาธารณสุข หน่วยงานภาครัฐ การเงิน และกลุ่มที่เกี่ยวข้องกับเด็กหรือเยาวชน มักถูกควบคุมโดยกฎหมายและข้อบังคับในระดับที่เข้มงวดกว่าปกติ

ประเด็นทางกฎหมายที่องค์กรควรพิจารณาเมื่อมีการจัดเก็บข้อมูลบนระบบคลาวด์ ได้แก่:

- การสูญหายหรือการรั่วไหลของข้อมูลธุรกิจ
- การรั่วไหลของข้อมูลส่วนบุคคล
- การเปิดเผยข้อมูลด้านบุคลากรหรือทรัพยากรมนุษย์
- ความรับผิดทางกฎหมายจากการละเมิดหรือข้อมูลรั่วไหล
- ทรัพย์สินทางปัญญา
- ความเป็นส่วนตัวตามกฎหมาย เช่น HIPAA
- ภาระทางภาษี
- ข้อจำกัดในการโอนข้อมูลข้ามพรมแดน

การลดความเสี่ยงทางกฎหมายจากการใช้ระบบคลาวด์ควรดำเนินการผ่าน:

- การกำหนดขั้นตอนรักษาความมั่นคงปลอดภัยอย่างชัดเจน
- การตรวจสอบสัญญาและข้อตกลงกับผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP)
- ความเข้าใจในกฎหมายท้องถิ่น ระดับประเทศ และระดับสากลที่ควบคุมข้อมูลที่องค์กรเก็บไว้

1.2 ประเด็นสำคัญเกี่ยวกับการคุ้มครองข้อมูล (Data Privacy and Protection)

การคุ้มครองข้อมูลและความเป็นส่วนตัว (Data Privacy) เป็นประเด็นสำคัญที่ส่งผลต่อการออกแบบสถาปัตยกรรมระบบคลาวด์ โดยเฉพาะเมื่อมีการเคลื่อนย้ายข้อมูลระหว่างพื้นที่ที่มีการกำกับดูแลต่างกัน เช่น ระดับท้องถิ่น ประเทศ หรือภูมิภาค

ประเภทของข้อมูลที่เกี่ยวข้อง ได้แก่:

- ข้อมูลส่วนบุคคล
- ข้อมูลทางการเงิน
- เวชระเบียน
- ทรัพย์สินทางปัญญา

หัวข้อที่เกี่ยวข้องกับ Data Privacy:

- เหตุผลทางภาษี
- ความคุ้มครองทางกฎหมาย
- ความรับผิดทางกฎหมาย
- ความเป็นส่วนตัวของบุคคล
- ข้อบังคับเฉพาะอุตสาหกรรม

1.3 การจัดประเภทข้อมูล (Data Classification)

- เป็นการระบุความสำคัญและผลกระทบของข้อมูลในกรณีที่เกิดการรั่วไหล
- ตัวอย่างการจัดประเภทข้อมูล:
 - Confidential (ลับมาก)
 - Internal Use (ใช้ภายในองค์กรเท่านั้น)
 - Public (เปิดเผยได้)

เครื่องมือช่วยในการจัดประเภทข้อมูล:

- Amazon Macie – ใช้ Machine Learning เพื่อระบุและจัดประเภทข้อมูล
- Azure Information Protection – ให้ระบบจัดชั้นความลับ เช่น Confidential, Highly Confidential การจัดประเภทข้อมูลเป็นความรับผิดชอบของลูกค้า ไม่ใช่ผู้ให้บริการคลาวด์โดยตรง

1.4 นโยบายการเก็บรักษาข้อมูล (Data Retention Policies)

- ระยะเวลาที่ข้อมูลต้องถูกเก็บไว้เพื่อให้สอดคล้องกับข้อกำหนดทางกฎหมาย สัญญา หรือข้อบังคับ
- อาจกำหนดทั้งระยะเวลาขั้นต่ำและระยะเวลาสูงสุด

กรณีการเก็บรักษาข้อมูล:

- Litigation Hold (การระงับข้อมูลเพื่อการดำเนินคดี): ห้ามลบหรือดัดแปลงข้อมูล
- Contractual Hold (ข้อกำหนดจากสัญญา): พันธสัญญาระหว่างองค์กรที่ต้องเก็บข้อมูลตามข้อตกลง
- Regulatory Hold (ข้อกำหนดจากกฎหมายเบี่ยง): สอดคล้องกับกฎหมาย เช่น กฎหมายภาษีหรือเวชระเบียน

1.5 มาตรฐานอุตสาหกรรมสำหรับการปฏิบัติตามข้อกำหนดและข้อบังคับ (Industry Standards for Compliance and Regulation)

อุตสาหกรรมแต่ละประเภทมีมาตรฐานเฉพาะที่ควบคุมการปฏิบัติตามข้อกำหนดด้านข้อมูล มาตรฐานเหล่านี้อาจเป็นกฎหมาย ข้อบังคับ หรือแนวทางปฏิบัติที่มีความสำคัญในระบบคลาวด์ โดยมาตรฐานทั่วไปที่เกี่ยวข้องกับหลายกรณี ได้แก่:

1. ระบบควบคุมองค์กรและระบบงาน (SOC2 – Systems and Organization Controls 2)

SOC2 เป็นมาตรฐานที่ช่วยให้มั่นใจได้ว่าผู้ให้บริการระบบคลาวด์หรือระบบภายนอกที่องค์กรใช้บริการ มีการจัดการความเป็นส่วนตัวและความปลอดภัยของข้อมูลตามระดับที่ยอมรับได้ ซึ่งมีความสำคัญอย่างยิ่งเนื่องจากข้อมูลเหล่านี้ยังคงเป็นความรับผิดชอบทั้งในด้านกฎหมายและจริยธรรมขององค์กรต้นทาง SOC2 พิจารณาภายใต้หลักการ 5 ประการ ได้แก่:

- ความเป็นส่วนตัว (Privacy)
- ความมั่นคงปลอดภัย (Security)
- ความพร้อมใช้งาน (Availability)
- ความถูกต้องของกระบวนการ (Processing Integrity)
- ความลับของข้อมูล (Confidentiality)

SOC2 เป็นมาตรฐานที่มีความยืดหยุ่น สามารถปรับให้สอดคล้องกับกระบวนการทางธุรกิจ และมักได้รับการตรวจสอบโดยผู้ตรวจสอบบัญชีภายนอกที่ได้รับการรับรอง

SOC3 คือ รายงานสาธารณะที่สรุปเนื้อหาสำคัญจาก SOC2 และใช้เพื่อแสดงจุดเด่นด้านความปลอดภัยของผู้ให้บริการระบบคลาวด์ เช่น AWS, Microsoft Azure และ Google Cloud Platform

2. มาตรฐานความมั่นคงปลอดภัยของข้อมูลการชำระเงิน (PCI DSS – Payment Card Industry Data Security Standards)

PCI DSS เป็นมาตรฐานที่กำหนดแนวทางการรักษาความปลอดภัยของธุกรรมบัตรเครดิตและบัตรเดบิต โดยออกแบบมาเพื่อปกป้องทั้งผู้บริโภคและผู้ค้า จากการโจรมหรือฉ้อโกง แนวทางที่กำหนดไว้ใน PCI DSS ได้แก่:

- การเข้ารหัสข้อมูล (Data Encryption)
- การใช้ซอฟต์แวร์ป้องกันไวรัส (Antivirus)
- การใช้ไฟร์วอลล์และระบบตรวจสอบเครือข่าย (Firewalls and Monitoring)
- การควบคุมการเข้าถึงข้อมูลบัตร (Access Control)

เว็บไซต์อีคอมเมิร์ซที่ดำเนินการบนระบบคลาวด์ควรปฏิบัติตามมาตรฐานนี้เพื่อสร้างความน่าเชื่อถือและความไว้วางใจจากลูกค้า

3. มาตรฐาน ISO 27000 ด้านการจัดการความมั่นคงปลอดภัยสารสนเทศ (ISO 27000 Series – Information Security Management)

มาตรฐานชุด ISO 27000 เป็นแนวทางระดับสากลสำหรับการจัดการระบบความมั่นคงปลอดภัยสารสนเทศ (Information Security Management System: ISMS) ซึ่งช่วยให้องค์กรสามารถระบุความเสี่ยง ปรับปรุงมาตรการควบคุม และบริหารจัดการความปลอดภัยของข้อมูลอย่างเป็นระบบ โดยเฉพาะอย่างยิ่งในบริบทของการให้บริการระบบคลาวด์

หนึ่งในมาตรฐานหลัก คือ ISO/IEC 27001 ซึ่งระบุข้อกำหนดสำหรับการออกแบบ ดำเนินงาน ตรวจสอบ และปรับปรุง ISMS อย่างต่อเนื่อง เพื่อให้มั่นใจใน หลักการ CIA Triad ได้แก่:

- Confidentiality (ความลับของข้อมูล) – การป้องกันไม่ให้ข้อมูลถูกเข้าถึงโดยผู้ที่ไม่ได้รับอนุญาต
- Integrity (ความถูกต้องครบถ้วน) – การรักษาความถูกต้องและความสมบูรณ์ของข้อมูล
- Availability (ความพร้อมใช้งาน) – การทำให้มั่นใจว่าข้อมูลสามารถเข้าถึงได้เมื่อจำเป็น

การขอการรับรองมาตรฐาน ISO/IEC 27001 เป็นกระบวนการที่ต้องอาศัยการวางแผนที่ซับซ้อน ประกอบด้วยการกำหนดขอบเขตของระบบ ตรวจสอบสถานะปัจจุบันของการบริหารจัดการสารสนเทศ ปรับปรุงกระบวนการตามข้อกำหนดของมาตรฐาน รวมถึงการฝึกอบรมและจัดทำเอกสาร เพื่อเตรียมความพร้อมสำหรับการตรวจประเมินโดยหน่วยรับรองอิสระ

4. แนวปฏิบัติจาก Cloud Security Alliance (CSA)

Cloud Security Alliance (CSA) เป็นองค์กรไม่แสวงหาผลกำไรที่มีบทบาทสำคัญในการส่งเสริมความมั่นคงปลอดภัยของระบบคลาวด์ โดยมุ่งเน้นการสร้างมาตรฐาน ทรัพยากรทางวิชาการ การฝึกอบรม และการรับรอง เพื่อยกระดับความรู้และการปฏิบัติด้านความมั่นคงปลอดภัยของข้อมูลในสภาพแวดล้อมคลาวด์

CSA ประกอบด้วยผู้เชี่ยวชาญจากภาครัฐ เอกชน อุตสาหกรรมเทคโนโลยี และสถาบันการศึกษา ซึ่งทำงานร่วมกันเพื่อพัฒนาเครื่องมือและแนวทางที่เป็นประโยชน์ต่อองค์กรต่าง ๆ ที่ใช้บริการคลาวด์ ทั้งในฐานะผู้ให้บริการ (Cloud Provider) และผู้ใช้บริการ (Cloud Consumer) หนึ่งในกรอบแนวทางที่สำคัญของ CSA คือ:

Cloud Controls Matrix (CCM)

CCM เป็นเฟรมเวิร์กที่จัดกลุ่มการควบคุมด้านความมั่นคงปลอดภัยตามหมวดหมู่ต่าง ๆ เช่น:

- การบริหารจัดการความเสี่ยง (Risk Management)
- การควบคุมการเข้าถึง (Access Control)
- ความมั่นคงปลอดภัยด้านข้อมูล (Data Security and Information Lifecycle Management)
- การเข้ารหัสและการจัดการกุญแจ (Encryption and Key Management)
- การกำกับดูแลและการปฏิบัติตามกฎหมาย (Governance and Compliance)

CCM ถูกออกแบบให้สามารถเชื่อมโยงกับมาตรฐานอื่น ๆ เช่น ISO/IEC 27001, PCI DSS, NIST, และ HIPAA เพื่อให้องค์กรสามารถประเมินความเสี่ยงและตรวจสอบการปฏิบัติตามข้อกำหนดได้อย่างครอบคลุม ในสภาพแวดล้อมคลาวด์

หน่วยที่ 2 แนวปฏิบัติที่ดีที่สุดด้านความมั่นคงปลอดภัย (Security Best Practices)

บทเรียนนี้กล่าวถึงแนวปฏิบัติที่สำคัญด้านความมั่นคงปลอดภัยในระบบคลาวด์ ซึ่งครอบคลุมหัวข้อสำคัญ เช่น การควบคุมการเข้าถึง การเข้ารหัสข้อมูล และการติดตั้งแพตช์ระบบ แนวทางเหล่านี้ควรถูกบูรณาการเข้ากับนโยบายความมั่นคงปลอดภัยโดยรวมขององค์กร เพื่อให้สามารถลดความเสี่ยงจากภัยคุกคาม และรักษาความมั่นคงของระบบสารสนเทศได้อย่างมีประสิทธิภาพ

2.1 หลักการมอบสิทธิ์น้อยที่สุด (Principle of Least Privilege)

แนวคิดในการให้สิทธิ์การเข้าถึงทรัพยากรของผู้ใช้งานในระบบนั้น มีพื้นฐานจากหลักการสำคัญหลายประการ โดยหนึ่งในหลักการที่สำคัญที่สุด คือ “หลักการมอบสิทธิ์น้อยที่สุด” (Principle of Least Privilege: PoLP) ซึ่งระบุว่า ผู้ใช้งานควรได้รับสิทธิ์การเข้าถึงเท่าที่จำเป็นต่อการปฏิบัติงานเท่านั้น หากมีความจำเป็นเพิ่มเติมจึงค่อยพิจารณาให้สิทธิ์เพิ่มเติมโดยผ่านกระบวนการอนุมัติ

ตัวอย่าง:

พนักงานคนหนึ่งปฏิบัติงานวันจันทร์ถึงศุกร์ เวลา 8.00 น. ถึง 17.00 น. โดยมีหน้าที่อ่านรายงานของฝ่ายขายที่เก็บอยู่ในไฟล์เดอร์ชื่อ SalesDepartment แต่ไม่จำเป็นต้องแก้ไขหรือเขียนข้อมูลลงในไฟล์เดอร์ดังกล่าว นอกจากนี้ พนักงานยังต้องพิมพ์เอกสารจากเครื่องพิมพ์ ColorPrinter7 แต่ไม่ต้องบริหารจัดการเครื่องพิมพ์แต่อย่างใด การประยุกต์ใช้หลักการ PoLP กับกรณีดังกล่าวสามารถดำเนินการได้ดังนี้:

- กำหนดให้บัญชีผู้ใช้งานสามารถเข้าสู่ระบบได้เฉพาะในช่วงเวลาทำงานที่กำหนด หากมีความพยายามเข้าสู่ระบบนอกเวลาดังกล่าว ระบบจะปฏิเสธการยืนยันตัวตน
- กำหนดสิทธิ์ “อ่านอย่างเดียว” (Read-only) ให้กับไฟล์เดอร์ SalesDepartment โดยไม่ให้สิทธิ์ในการเขียน (Write) เนื่องจากไม่ได้เกี่ยวข้องกับหน้าที่
- กำหนดสิทธิ์ “พิมพ์เอกสาร” (Print) ให้กับ ColorPrinter7 โดยไม่อนุญาตให้จัดการการตั้งค่าของเครื่องพิมพ์

หลักการ PoLP นี้ครุภูณฑ์ไปใช้กับระบบรักษาความมั่นคงปลอดภัยขององค์กรโดยรวม เพราะการจำกัดสิทธิ์ช่วยลดความเสี่ยงจากการเข้าถึงโดยมิชอบ และเป็นแนวทางพื้นฐานในสภาพแวดล้อมแบบ Zero Trust ซึ่งจะกล่าวต่อไป

2.2 สถาปัตยกรรมความมั่นคงปลอดภัยแบบ Zero Trust (Zero Trust Cloud Security)

แนวคิด Zero Trust ถือว่าทุกระบบที่อุปกรณ์—ไม่ว่าจะเป็นระบบภายในหรือภายนอก—จะต้อง “ไม่ถูกไว้วางใจโดยอัตโนมัติ” และทำการร้องขอเข้าถึงต้องผ่านการยืนยันและกำหนดสิทธิ์อย่างชัดเจนก่อนเสมอ

องค์ประกอบสำคัญของ Zero Trust ได้แก่:

- การควบคุมการเข้าถึงแบบละเอียด (Granular Access Controls): ใช้หลักการ microsegmentation และนโยบายการอนุญาตเฉพาะ เพื่อบังคับใช้หลักการ PoLP ทั้งในระบบคลาวด์และระบบในองค์กร
- การตรวจสอบตัวตนอย่างต่อเนื่อง (Continuous Authentication Verification): ต้องมีการพิสูจน์ตัวตนสำหรับแต่ละคำร้อง ไม่พึ่งพียงแค่การเข้าสู่ระบบครั้งแรก
- ไม่มีความไว้วางใจโดยปริยาย (Zero Implicit Trust): ไม่ว่าจะเป็นอุปกรณ์หรือระบบใด ๆ ไม่ว่าจะเป็นลูกข่ายหรือเซิร์ฟเวอร์
- ระบบรักษาความมั่นคงปลอดภัยแบบพลวัต (Dynamic Security): ตอบสนองต่อความผิดปกติหรือพฤติกรรมเปลกปลอมแบบเรียลไทม์

ประโยชน์ของ Zero Trust:

- เพิ่มความสามารถในการควบคุมการเข้าถึง
- ลดความเสี่ยงจากการโจมตี
- เพิ่มประสิทธิภาพในการปฏิบัติตามกฎระเบียบด้านความมั่นคงปลอดภัย (Compliance)

แนวทางปฏิบัติของ Zero Trust:

- กำหนดนโยบายด้าน IAM และการจัดการบัญชีอย่างเข้มงวด
- ใช้ micro-segmentation แยกทรัพยากรตามหน้าที่การใช้งาน
- ตรวจสอบและติดตามการเข้าถึงอย่างต่อเนื่อง พร้อมตอบสนองโดยอัตโนมัติ
- เข้ารหัสข้อมูลทั้งขณะส่งผ่าน (in transit) และขณะจัดเก็บ (at rest)

2.3 แนวทางปฏิบัติด้านเกณฑ์มาตรฐาน (Benchmark Practices)

เกณฑ์มาตรฐานถือเป็นแนวทางสำคัญที่ใช้ในการประเมินความเสี่ยง กำหนดค่าความมั่นคงปลอดภัย ขั้นต่ำ และแสดงให้เห็นถึงผลตอบแทนจากการลงทุน (Return on Investment: ROI) เกณฑ์เหล่านี้ช่วยขับเคลื่อนผู้ดูแลระบบให้กำหนดเวอร์ชันและการตั้งค่าที่ได้รับการรับรองเมื่อมีการติดตั้งระบบคลาวด์ใหม่ โดยสามารถใช้เป็นรากฐานในการวางแผนด้านความมั่นคงปลอดภัยในระดับสูงขึ้นได้ในอนาคต

แนวทาง เช่น Security Technical Implementation Guides (STIGs) ได้รับการพัฒนาโดย Defense Information Systems Agency (DISA) สำหรับกระทรวงกลาโหมสหรัฐอเมริกา (Department of Defense: DoD)

นอกจากนี้ยังมีแนวทางอื่นที่สำคัญ ดังนี้:

- คู่มือการกำหนดค่าความมั่นคงปลอดภัยของ NSA (NSA Security Configuration Guides)
- Security Requirements Guide (SRG) ของ DISA
- มาตรฐานของ NIST เช่น NIST SP 800-53 และ NIST SP 800-171

มาตรฐานหลักอีกประการหนึ่งคือ Center for Internet Security (CIS) Benchmarks for Cloud Security ซึ่งเป็นแนวทางปฏิบัติที่ได้รับความเชื่อถืออย่างแพร่หลายในการรักษาความมั่นคงปลอดภัยของระบบคลาวด์สาธารณะ

CIS Benchmarks

มาตรฐานของ CIS สำหรับความมั่นคงปลอดภัยระบบคลาวด์กำหนดแนวทางปฏิบัติที่ดีที่สุด (Best Practices) และการตั้งค่าที่เหมาะสมในการใช้บริการคลาวด์สาธารณะ มาตรฐานเหล่านี้ไม่มีข้อผูกพันกับผู้ให้บริการรายใดรายหนึ่ง (Vendor-neutral) และสามารถประยุกต์ใช้ได้กับสภาพแวดล้อมที่หลากหลาย ทั้งในระบบปฏิบัติการ ซอฟต์แวร์ และระบบเชิร์ฟเวอร์

หัวข้อหลักที่ครอบคลุม ได้แก่:

- การพิสูจน์ตัวตนและการบังคับใช้หลักการสิทธิ์น้อยที่สุด (Least Privilege)
- การจัดเก็บและเข้ารหัสข้อมูล
- การควบคุมการเข้าถึงเครือข่ายและการบันทึกข้อมูล (Logging)
- กระบวนการเสริมความแข็งแกร่งระบบ (Hardening)
- การบริหารจัดการภาระงาน (Workload Management)

CIS แบ่งคำแนะนำออกเป็น 2 ระดับ ได้แก่:

- ระดับที่ 1 (Level 1): การตั้งค่าพื้นฐานที่สามารถนำไปใช้ได้ทันที
- ระดับที่ 2 (Level 2): การตั้งค่าเพิ่มเติมที่ซับซ้อนมากขึ้นและอาจส่งผลต่อประสิทธิภาพของระบบ

ประโยชน์ของ CIS Benchmarks

- ลดพื้นที่เสี่ยง (Attack Surface)
- ลดช่องโหว่ระบบ (Vulnerabilities)
- กำหนดค่าพื้นฐานด้านความมั่นคงปลอดภัยอย่างเป็นรูปธรรม
- ช่วยให้สามารถตรวจสอบและเฝ้าระวังได้อย่างแม่นยำ

แนวทางเฉพาะผู้ให้บริการ (Vendor-Specific Benchmarks)

ผู้ให้บริการคลาวด์บางรายกำหนดแนวทางปฏิบัติเฉพาะของตนเอง เช่น:

- AWS Well-Architected Framework: Security Pillar
- Azure Security Benchmark
- Google Cloud Security Foundations Guide
- Oracle Cloud Security Best Practices

2.4 การเสริมความแข็งแกร่งระบบ (Hardening)

การเสริมความแข็งแกร่งของระบบ (Hardening) ไม่ใช่เป็นการติดตั้งระบบจริงหรือเปลี่ยน ทั้งในองค์กรหรือระบบคลาวด์ ถือเป็นภารกิจที่สำคัญยิ่ง โดยหลักการพื้นฐานคือ “ลบสิ่งที่ไม่จำเป็น และใช้วอร์ชันที่ทันสมัยที่สุดของสิ่งที่จำเป็น”

แนวทางในการ Hardening ระบบปฏิบัติการ (OS) ได้แก่:

- รักษาค่าการตั้งค่าความมั่นคงปลอดภัยพื้นฐานที่ลดบริการและฟีเจอร์ที่ไม่จำเป็น
- ใช้เครื่องมือบริหารจัดการการตั้งค่าแบบอัตโนมัติ
- เปลี่ยนชื่อบัญชีผู้ใช้ค่าเริ่มต้น (เช่น Administrator, Root)
- ปรับค่าตั้งต้นของระบบ
- ติดตั้งแพตช์ความปลอดภัยอย่างสม่ำเสมอ
- ควบคุมการเข้าถึงระยะไกล (เช่น SSH, RDP)
- ตรวจสอบบันทึก (Logs) อย่างต่อเนื่อง

- ติดตั้งและอัปเดตโปรแกรมป้องกันไวรัส/มัลแวร์
- สำรวจข้อมูลอย่างสม่ำเสมอ

การตั้งค่าคอนฟิกพื้นฐาน (Configuration Baselines)

Baseline หมายถึง เวอร์ชันและการตั้งค่ามาตรฐานของระบบปฏิบัติการที่องค์กรกำหนด เช่น Windows Server 2022 หรือ Ubuntu Server 22 LTS ซึ่งจะช่วยให้สามารถบริหารจัดการ แก้ไขปัญหา และเสริมความปลอดภัยได้อย่างมีประสิทธิภาพ

ตัวอย่าง: องค์กรอาจสร้างแม่แบบ (Template) ของ VM ที่ติดตั้ง Ubuntu 22 พร้อม SSH และ Apache โดยไม่มี GUI ให้ผู้พัฒนาใช้งานได้ทันทีโดยมั่นใจว่าปลอดภัยตามนโยบายขององค์กร

2.5 การเข้ารหัสข้อมูลขณะส่งผ่าน (Encrypting Data in Transit)

ในระบบเครือข่ายที่ใช้โปรโตคอล TCP/IP ส่วนใหญ่ การรับส่งข้อมูลจะไม่ได้รับการเข้ารหัสโดยค่าเริ่มต้น ส่งผลให้ข้อมูลในระหว่างการส่งผ่าน (In Transit) เสี่ยงต่อการถูกดักฟัง ตัวอย่างเช่น โปรโตคอลพื้นฐานต่าง ๆ อย่าง SMTP (สำหรับอีเมล), HTTP (สำหรับเว็บไซต์), และ FTP (สำหรับการโอนไฟล์) ไม่ได้ออกแบบมาเพื่อให้มีความลับหรือความถูกต้องของข้อมูล

ในยุคแรกของระบบเครือข่าย ข้อมูลส่วนใหญ่ไม่มีความสำคัญมากพอที่จะต้องได้รับการปกป้องอย่างไรก็ตาม ในปัจจุบัน ระบบเครือข่ายได้กล้ายเป็นช่องทางสำคัญที่มีการส่งข้อมูลส่วนบุคคลและข้อมูลลับจำนวนมาก การดักจับข้อมูลในระบบเครือข่ายสามารถทำได้โดยใช้เครื่องมือวิเคราะห์แพ็กเก็ต เช่น Wireshark หรือ tcpdump

Hypertext Transfer Protocol Secure (HTTPS)

โปรโตคอล HTTP ใช้สำหรับส่งข้อมูลระหว่างเว็บเซิร์ฟเวอร์และไคลเอนต์ แต่ไม่มีการเข้ารหัสข้อมูล จึงเสี่ยงต่อการถูกแฮกเกอร์ดักฟังหรือแทรกแซงข้อมูลในรูปแบบของการโจมตีแบบ Man-in-the-Middle

เพื่อเพิ่มความปลอดภัย จึงได้มีการพัฒนาโปรโตคอล Secure Sockets Layer (SSL) ในช่วงปี 1990 เพื่อเพิ่มการเข้ารหัสให้กับ HTTP อย่างไรก็ตาม SSL ปัจจุบันได้ถูกยกเลิกการใช้งานและแทนที่ด้วย Transport Layer Security (TLS)

เมื่อ HTTP ถูกใช้งานร่วมกับ TLS จึงเรียกว่า HTTPS โดย HTTPS จะใช้ใบรับรองดิจิทัล (Digital Certificates) เพื่อยืนยันตัวตนของเซิร์ฟเวอร์ และเริ่มกระบวนการเข้ารหัสข้อมูล

Transport Layer Security (TLS)

TLS ใช้การเข้ารหัสแบบสมมติ โดยใช้ทั้งการเข้ารหัสแบบสมมาตร (Symmetric) และอสมมาตร (Asymmetric) กระบวนการเริ่มจากไคลเอนต์ร้องขอการเชื่อมต่อกับเว็บเซิร์ฟเวอร์ เซิร์ฟเวอร์จะตอบกลับด้วยกุญแจสาธารณะ (Public Key) ซึ่ง放อยู่ในใบรับรองดิจิทัล ไคลเอนต์จะตรวจสอบความถูกต้องและร่วมกับเซิร์ฟเวอร์ในการสร้าง “คีย์เซชัน” ซึ่งใช้เข้ารหัสข้อมูลระหว่างการส่งข้อมูลของกระบวนการนี้ มี 2 ประการหลัก:

1. ยืนยันตัวตนของเว็บไซต์ฟรีผ่านการตรวจสอบกุญแจสาธารณะ
2. การถ่ายโอนข้อมูลลูกเข้ารหัสด้วยคีย์เซสชัน ทำให้ผู้ที่ดักข้อมูลไม่สามารถอ่านเนื้อหาได้โดยง่าย
HTTPS และ TLS จึงเป็นองค์ประกอบสำคัญของบริการระบบคลาวด์ เพราะใช้ในการเชื่อมต่อระหว่างผู้ใช้และบริการต่าง ๆ อย่างแพร่หลาย รวมถึงการจัดการระบบคลาวด์เองก็ใช้ผ่าน HTTPS เป็นหลัก
IPsec (Internet Protocol Security)

ระบบปฏิบัติการสามารถตั้งค่าให้ใช้ IPsec เพื่อเข้ารหัสข้อมูลก่อนที่ข้อมูลจะออกจากเครือข่ายทางและจะไม่ถูกถอดรหัสจนกว่าจะถึงเครื่องปลายทาง ซึ่งช่วยป้องข้อมูลตลอดเส้นทางการส่งผ่านเครือข่าย

2.6 การเข้ารหัสข้อมูลที่เก็บอยู่ (Encrypting Data at Rest)

ข้อมูลที่จัดเก็บอยู่ในระบบมีความเสี่ยงต่อการถูกขโมยหรือถูกเปลี่ยนแปลงโดยไม่ได้รับอนุญาต การเข้ารหัสข้อมูลในระดับไดรฟ์หรือไฟล์สามารถช่วยลดความเสี่ยงดังกล่าวได้ โดยเฉพาะอย่างยิ่งหากอุปกรณ์จัดเก็บข้อมูลถูกโจรมารุณ

การเข้ารหัสในระบบคลาวด์ (Cloud Encryption)

ผู้ให้บริการคลาวด์มีทางเลือกสำหรับการเข้ารหัสข้อมูลผ่านเซิร์ฟเวอร์ (Server-Side Encryption: SSE) ตัวอย่างเช่น บริการ Amazon S3 และ EBS ของ AWS รองรับการเข้ารหัสข้อมูลทั้งในระดับดิสก์และระดับไฟล์

การเข้ารหัสระดับไดรฟ์และไฟล์ (Drive and File Encryption)

ระบบปฏิบัติการสามารถเข้ารหัสข้อมูลที่เก็บไว้ได้ทั้งในระดับดิสก์และระดับไฟล์ ซึ่งเป็นพังก์ชันที่ทำงานร่วมกับระบบแฟ้มข้อมูล (File System)

ตัวอย่างการเข้ารหัสระดับไดรฟ์:

- Linux Unified Key Setup (LUKS)
- BitLocker ของ Microsoft

การเข้ารหัสดิสก์จะดำเนินการระหว่างขั้นตอนการปิดเครื่องและเปิดเครื่องใหม่ โดยข้อมูลจะถูกเข้ารหัสมีระบบปิดและถอดรหัสเมื่อเริ่มระบบ ซึ่งช่วยป้องกันข้อมูลกรณีไดรฟ์ถูกขโมย

ตัวอย่างการเข้ารหัสระดับไฟล์:

- gzip (Linux/Unix)
- NTFS Encrypting File System (EFS) บน Windows

การเข้ารหัสไฟล์หมายความว่าห้ามผู้ใช้งานหลายคนภายใต้ระบบเดียวกัน โดยจะสามารถจำกัดสิทธิ์การเข้าถึงไฟล์ให้เฉพาะผู้ใช้ที่ได้รับอนุญาตเท่านั้น

การตรวจสอบความถูกต้องของไฟล์ (File Integrity Monitoring: FIM)

FIM เป็นกระบวนการตรวจสอบว่ามีการเปลี่ยนแปลงไฟล์ที่สำคัญโดยไม่ได้รับอนุญาตหรือไม่ โดยระบบปฏิบัติการ เช่น Windows ใช้ FIM เพื่อตรวจสอบไฟล์ระบบที่สำคัญ ทั้งนี้ยังสามารถใช้งานกับแอปพลิเคชันและไฟล์ข้อมูลอื่น ๆ ได้เช่นกัน

มีโซลูชันของผู้ให้บริการภายนอกที่รองรับการตรวจสอบความถูกต้องของไฟล์ในระบบคลาวด์ โดยสามารถใช้ร่วมกับบริการของผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP)

2.7 ความมั่นคงปลอดภัยของคอนเทนเนอร์ (Container Security)

คอนเทนเนอร์เป็นองค์ประกอบที่พับได้ทั่วไปในโซลูชันระบบคลาวด์ในปัจจุบัน โดยช่วยสนับสนุนสถานการณ์การประมวลผลได้หลากหลายรูปแบบ อย่างไรก็ตาม การรักษาความมั่นคงปลอดภัยของคอนเทนเนอร์ ถือเป็นสิ่งสำคัญยิ่ง แม้ว่าจะมีแนวทางบางประการที่คล้ายคลึงกับระบบอื่น ๆ แต่คอนเทนเนอร์ยังมีลักษณะเฉพาะที่ต้องใช้แนวทางเฉพาะด้านเพิ่มเติม โดยเฉพาะอย่างยิ่งในเรื่องสิทธิ์การเข้าถึงทรัพยากรของคอนเทนเนอร์ แบบมีสิทธิพิเศษ (privileged) และแบบไม่มีสิทธิพิเศษ (unprivileged)

การบริหารจัดการคอนเทนเนอร์ตลอดช่วง 生命周期 (container lifecycle) เป็นหัวใจสำคัญของความมั่นคงปลอดภัย ซึ่งควรประกอบด้วยแนวปฏิบัติดังต่อไปนี้:

- หลีกเลี่ยงการเรียกใช้งานคอนเทนเนอร์ด้วยสิทธิ์ root
- ใช้การแบ่งเครือข่าย (network segmentation) เพื่อแยกและควบคุมการเข้ามายื่นต่อ
- กำหนดการสื่อสารระหว่างคอนเทนเนอร์ให้ปลอดภัย
- ใช้นโยบายความปลอดภัยเพื่อกำกั้นกิจกรรมของคอนเทนเนอร์
- ดึงภาพคอนเทนเนอร์ (container image) จากแหล่งที่เชื่อถือได้ และตรวจสอบความถูกต้องผ่านลายเซ็นดิจิทัล
- สแกนและทดสอบช่องโหว่ในภาพคอนเทนเนอร์
- จัดการเวอร์ชันของภาพคอนเทนเนอร์และดำเนินการอัปเดตอย่างสม่ำเสมอ
- จัดการข้อมูลลับ (secrets) ด้วยเครื่องมือ เช่น Kubernetes Secrets
- ตรวจสอบและบันทึกกิจกรรมของคอนเทนเนอร์อย่างต่อเนื่อง

คอนเทนเนอร์แบบมีสิทธิพิเศษ (Privileged Container)

คอนเทนเนอร์ประเภทนี้จะถูกสร้างและดำเนินการโดยบัญชีผู้ใช้ root (บัญชีผู้ดูแลระบบในระบบ Linux) และกระบวนการภายในคอนเทนเนอร์จะทำงานด้วยสิทธิ์ root เช่นกัน หากโค้ดที่เป็นอันตรายสามารถหลุดออกจากการคอนเทนเนอร์ได้ ก็จะมีสิทธิ์ในการควบคุมระบบโฮสต์ทั้งหมด ซึ่งถือเป็นความเสี่ยงด้านความมั่นคงปลอดภัยอย่างยิ่ง และควรหลีกเลี่ยงการใช้งานคอนเทนเนอร์ลักษณะนี้ เว้นแต่มีความจำเป็นอย่างแท้จริง

คอนเทนเนอร์แบบไม่มีสิทธิพิเศษ (Unprivileged Container)

คอนเทนเนอร์แบบไม่มีสิทธิพิเศษจะไม่รันด้วยสิทธิ์ root โดยบัญชีผู้ใช้งานที่ทำงานอยู่ในคอนเทนเนอร์ จะถูกแมปกับบัญชีผู้ใช้ธรรมดามาบนเครื่องโฮสต์ การออกแบบ workload สำหรับคอนเทนเนอร์ในยุคปัจจุบัน จึงนิยมใช้แบบไม่มีสิทธิพิเศษเพื่อเพิ่มความปลอดภัย

คอนเทนเนอร์ลักษณะนี้สามารถแยกตัวเองจากระบบโฮสต์และคอนเทนเนอร์อื่น ๆ ได้ดียิ่งขึ้น ลดพื้นผิวที่เป็นเป้าหมายของการโจมตี (attack surface) และเอื้อให้สามารถบังคับใช้นโยบายความมั่นคง ปลอดภัยได้อย่างเข้มงวด

ใน Azure Container Apps คอนเทนเนอร์แบบ privileged จะถูกปิดใช้งานโดยค่าเริ่มต้น และ หากต้องการเปิดใช้งานจะต้องสร้าง Docker image ด้วยตนเอง เช่นเดียวกับ AWS ที่มีการควบคุมลักษณะนี้

คอนเทนเนอร์แบบไม่มีสิทธิพิเศษนี้ยังเป็นตัวอย่างที่ชัดเจนของหลักการ “สิทธิ์ต่ำสุดเท่าที่จำเป็น” (Principle of Least Privilege)

การกำหนดสิทธิ์เข้าถึงไฟล์

ปัญหาด้านสิทธิ์ในการเข้าถึงไฟล์ก็เกี่ยวข้องกับการตั้งค่า privileged/unprivileged เช่นกัน โดยคอนเทนเนอร์จะเข้าถึงไฟล์ตามสิทธิ์ของบัญชีผู้ใช้ที่รันอยู่ การกำหนดค่าอย่างรอบคอบของ UID (User ID) ระหว่าง คอนเทนเนอร์และระบบจัดเก็บข้อมูลเป็นสิ่งจำเป็นเพื่อป้องกันการเข้าถึงไฟล์โดยไม่ได้รับอนุญาต

2.8 ความมั่นคงปลอดภัยของการจัดเก็บไฟล์และอ็อบเจกต์ (File and Object Storage Security)

การจัดเก็บไฟล์ (File Storage)

เป็นรูปแบบที่ผู้ใช้งานส่วนใหญ่มักนึกถึง โดยอาศัยโครงสร้างแบบลำดับชั้น (hierarchical) ที่มีไดเรกทอรีและไฟล์อยู่ภายใต้ ข้อมูลอาจถูกแยกออกเป็นหลายชั้นและจัดเก็บในตำแหน่งต่าง ๆ ของไดรฟ์ การจัดเก็บแบบอ็อบเจกต์ (Object Storage)

ต่างจากแบบไฟล์โดยสิ้นเชิง เพราะใช้โครงสร้างแบบแบน (flat structure) ซึ่งสามารถขยายขนาดได้ ง่ายและมีประสิทธิภาพสูงกว่า จึงมักถูกใช้ในบริการคลาวด์ เช่น AWS S3

แม้โครงสร้างของทั้งสองแบบจะแตกต่างกัน แต่ก็สามารถใช้แนวปฏิบัติด้านความมั่นคงปลอดภัยร่วมกันได้ดังนี้:

- ใช้การเข้ารหัสข้อมูลขณะจัดเก็บ (Data-at-Rest Encryption)
- จัดการกุญแจเข้ารหัสอย่างปลอดภัยและมีประสิทธิภาพ
- ตรวจสอบความถูกต้องของไฟล์หรืออ็อบเจกต์ (Integrity Verification)
- กำหนดสิทธิ์การเข้าถึงโดยยึดตามหลัก “สิทธิ์ต่ำสุดเท่าที่จำเป็น”
- ใช้ระบบ IAM เพื่อควบคุมสิทธิ์ของผู้ใช้
- เปิดใช้งานระบบเวอร์ชัน (Versioning) เพื่อให้สามารถย้อนกลับข้อมูลได้
- ทำการจำลองข้อมูลข้ามภูมิภาค (Cross-Region Replication) เพื่อเพิ่มความพร้อมใช้งานและความทนทานของข้อมูล

หน่วยที่ 3 การแก้ไขปัญหาด้านความมั่นคงปลอดภัย (Troubleshooting Security Issues)

การแก้ไขปัญหาด้านความมั่นคงปลอดภัยมีความสำคัญอย่างยิ่งต่อการรักษาความปลอดภัยของระบบสารสนเทศ โดยมุ่งเน้นไปที่การตรวจจับความผิดปกติหรือความเบี่ยงเบนจากค่ามาตรฐานที่กำหนดไว้ เช่น การตั้งค่าความมั่นคงปลอดภัยเริ่มต้น (Security Baseline Configuration)

3.1 ปัญหาการตรวจสอบตัวตน (Authentication Issues)

ปัญหาด้านความมั่นคงปลอดภัยในระบบสารสนเทศมักเกี่ยวข้องกับการตรวจสอบตัวตนหรือการกำหนดสิทธิ์การเข้าถึง ซึ่งอาจเกิดจากการกำหนดค่าผิดพลาดหรือการใช้ชื่อယายที่ไม่เหมาะสม การตรวจสอบตัวตน (Authentication) คือกระบวนการที่ผู้ใช้พิสูจน์ตัวตนต่อเซิร์ฟเวอร์ เครื่องเสมือน บริการ หรือแอปพลิเคชัน

3.2 ปัญหาการกำหนดสิทธิ์ (Authorization Issues)

การกำหนดสิทธิ์ (Authorization) คือการอนุญาตให้ผู้ใช้เข้าถึงทรัพยากรหลังจากผ่านการตรวจสอบตัวตนเรียบร้อยแล้ว ปัญหาที่พบบ่อยได้แก่ การกำหนดสิทธิ์ผิดพลาด สิทธิ์ที่มากเกินความจำเป็น หรือ การเข้าถึงทรัพยากรโดยไม่ได้รับอนุญาต ซึ่งอาจนำไปสู่การเลื่อนสิทธิ์ (Privilege Escalation) และความเสี่ยงด้านความปลอดภัย

3.3 ปัญหาการกำหนดสิทธิ์ (Authorization Issues)

การกำหนดสิทธิ์ (Authorization) คือการอนุญาตให้ผู้ใช้เข้าถึงทรัพยากรหลังจากผ่านการตรวจสอบตัวตนเรียบร้อยแล้ว ปัญหาที่พบบ่อยได้แก่ การกำหนดสิทธิ์ผิดพลาด สิทธิ์ที่มากเกินความจำเป็น หรือ การเข้าถึงทรัพยากรโดยไม่ได้รับอนุญาต ซึ่งอาจนำไปสู่การเลื่อนสิทธิ์ (Privilege Escalation) และความเสี่ยงด้านความปลอดภัย

3.4 กลุ่มความปลอดภัย (Security Groups) และรายการควบคุมการเข้าถึง (Access Control Lists: ACLs)

ในการปรับใช้ระบบคลาวด์ อินสแตนซ์ของเครื่องเสมือน (Virtual Machine – VM) จะต้องกำหนดให้อยู่ภายใต้กลุ่มความปลอดภัย (Security Groups) เพื่อควบคุมการเข้าถึงเครือข่าย กลุ่มความปลอดภัยนี้ทำหน้าที่เสมือนไฟร์วอลล์เสมือน (Virtual Firewall) หากมีการกำหนดค่าผิดพลาด อาจส่งผลให้ไม่สามารถเชื่อมต่อกับอินสแตนซ์บางเครื่องได้

กรณีปัญหา: ไม่สามารถเชื่อมต่อกับอินสแตนซ์บางรายการได้

ให้ตรวจสอบว่าอินสแตนซ์นั้นได้รับการกำหนดกลุ่มความปลอดภัยที่เหมาะสมหรือไม่

รายการควบคุมการเข้าถึง (Access Control Lists – ACLs) เป็นกฎที่ระบุชนิดของทรัพย์สินที่อนุญาตให้เข้าถึงสมาชิกของกลุ่มความปลอดภัย โดยปกติแล้ว ทรัพย์สินเข้าทั้งหมดจะถูกปฏิเสธโดยค่าเริ่มต้น จนกว่าผู้ดูแลระบบคลาวด์จะอนุญาตการเชื่อมต่อเฉพาะอย่างเป็นทางการ อินสแตนซ์หนึ่ง

เครื่องอาจถูกเข้ามายังกับกลุ่มความปลอดภัยมากกว่าหนึ่งกลุ่ม และกฎทั้งหมดในแต่ละกลุ่มจะมีผลกับอินสแตนซ์นั้น ผู้ดูแลระบบจึงจำเป็นต้องตรวจสอบการสืบทอดกฎให้ครบถ้วน

กรณีปัญหา: ไม่สามารถเข้ามายังต่อไปนี้

ให้ตรวจสอบว่าได้มีการระบุกฎเข้ามายังหนึ่งในอินสแตนซ์ที่ต้องตรวจสอบ

ในกรณีที่อินสแตนซ์ได้รับการกำหนดกลุ่มความปลอดภัยหลายกลุ่มพร้อมกัน อาจมีอย่างน้อยหนึ่งกฎที่ปิดกั้นการเข้าถึงอยู่ ผู้ดูแลระบบควรตรวจสอบเงื่อนไขของกฎทั้งหมด

3.5 การควบคุมการเข้าถึงเครือข่าย (Network Access Control: NAC)

การควบคุมการเข้าถึงเครือข่าย (NAC) เป็นกลไกที่ใช้ตรวจสอบสุขภาพและการกำหนดค่าความปลอดภัยของอุปกรณ์ที่พยายามเข้าร่วมเครือข่าย การวิเคราะห์ปัญหาที่เกี่ยวข้องกับ NAC จำเป็นต้องตรวจสอบว่าไหนดีเครือข่าย (Network Nodes) ที่เกี่ยวข้องได้รับการเข้ามายังกับระบบ NAC อย่างถูกต้อง พร้อมทั้งตรวจสอบค่าการกำหนดค่าของระบบ

กรณีปัญหา: NAC ไม่ตรวจสอบอุปกรณ์ทั้งหมด

ให้พิจารณาประเด็นต่อไปนี้:

- อุปกรณ์ใดที่ได้รับการตรวจสอบหรือถูกยกเว้นจากการตรวจสอบ
- จุดเข้าเครือข่ายใดที่อยู่ภายใต้การควบคุมของ NAC (เช่น การเข้ามายังผ่าน VPN, การเข้าถึงจากระยะไกล, หรือจากภายในองค์กร)

กรณีปัญหา: NAC ไม่สามารถตรวจสอบอุปกรณ์ที่ไม่มีความปลอดภัย

ให้ตรวจสอบค่าการจัดการการกำหนดค่าของระบบ NAC เพื่อยืนยันว่าได้กำหนดเงื่อนไขการตรวจสอบอย่างถูกต้อง

3.6 ประเด็นด้านช่องโหว่ของซอฟต์แวร์ (Software Vulnerability Issues)

ซอฟต์แวร์จำนวนมากมีความซับซ้อนอย่างยิ่ง โดยอาจมีโค้ดนับพันบรรทัด ซึ่งทำให้การตรวจสอบหาข้อผิดพลาดหรือช่องโหว่ด้านความปลอดภัยเป็นเรื่องยาก การลดความเสี่ยงจากช่องโหว่ของซอฟต์แวร์สามารถทำได้ด้วยการใช้ซอฟต์แวร์เวอร์ชันล่าสุดและดำเนินการอัปเดตแพตช์ตามคำแนะนำของผู้จัดจำหน่ายอย่างเคร่งครัด

การใช้ซอฟต์แวร์เวอร์ชันล่าสุดถือเป็นองค์ประกอบสำคัญของการรักษาความมั่นคงปลอดภัยของระบบ (System Hardening) โดยเฉพาะอย่างยิ่งในกรณีที่ทีมพัฒนาขององค์กรดูแลระบบหรือแอปพลิเคชันบุคคลากรที่พัฒนาเอง

หากปัญหาคือ มีการตรวจสอบช่องโหว่ด้านความปลอดภัยของซอฟต์แวร์จากการตรวจสอบ:

ให้พิจารณา:

- ซอฟต์แวร์ที่ใช้อยู่เป็นเวอร์ชันล่าสุดหรือไม่
- เวอร์ชันของซอฟต์แวร์ตรงกับมาตรฐานการกำหนดค่า (Baseline) หรือไม่
- ได้มีการติดตั้งแพตช์ล่าสุดแล้วหรือไม่

หากปัญหาคือ ผู้จ้างหน่ายซอฟต์แวร์ไม่สามารถแก้ไขช่องโหว่ได้:

ควรพิจารณายกเลิกการใช้งานซอฟต์แวร์ดังกล่าวและเปลี่ยนไปใช้โซลูชันอื่นแทน

หากปัญหาคือ พบร่องโหว่ในซอฟต์แวร์ที่พัฒนาโดยทีมภายนอกองค์กร:

ให้ส่งคำร้องไปยังทีมพัฒนาเพื่อดำเนินการแก้ไข

3.7 ซอฟต์แวร์ที่ไม่ได้รับอนุญาต (Unauthorized Software)

องค์กรส่วนใหญ่มีการกำหนดรายการซอฟต์แวร์ที่ได้รับอนุญาตไว้สำหรับการใช้งานภายใน การติดตั้ง การกำหนดค่า และการอบรมใช้งานมักจะเน้นที่ซอฟต์แวร์ที่ได้รับอนุญาตเหล่านี้เท่านั้น ซอฟต์แวร์ที่ไม่ได้รับอนุญาตไม่ควรถูกติดตั้งบนเครื่องลูกข่าย เชิร์ฟเวอร์ภายนอก หรือแม้แต่ VM บนระบบคลาวด์ เพราะซอฟต์แวร์ดังกล่าวอาจไม่ได้อยู่ภายใต้ระบบควบคุมการกำหนดค่า (Configuration Management) หรือกลไกการอัปเดตที่เหมาะสม จึงอาจมีช่องโหว่ที่ยังไม่ได้รับการแก้ไข นอกจากนี้ยังอาจนำไปสู่การละเมิดลิขสิทธิ์ได้

หลายองค์กรจัดทำคลังซอฟต์แวร์ (Software Repository) สำหรับการติดตั้งซอฟต์แวร์ที่ได้รับอนุญาต โดยแนะนำให้ดำเนินการติดตั้งผ่านคลังนี้เท่านั้น และควรใช้ระบบอัตโนมัติในการติดตั้งให้มากที่สุด เพื่อหลีกเลี่ยงการติดตั้งซอฟต์แวร์ที่ไม่ได้รับอนุญาต:

- เพื่อให้สามารถสนับสนุนและฝึกอบรมได้อย่างเหมาะสม
- เพื่อควบคุมการอัปเดตแพตช์และเวอร์ชัน
- เพื่อให้มั่นใจว่าเป็นซอฟต์แวร์ที่ผ่านการตรวจสอบแล้ว
- เพื่อให้มั่นใจว่ามาจากผู้จ้างหน่ายที่เชื่อถือได้
- เพื่อบริหารจัดการลิขสิทธิ์ซอฟต์แวร์

หากปัญหาคือ เครื่องมือการตรวจสอบระบบหรือลงบัญชีซอฟต์แวร์พบว่ามีซอฟต์แวร์ที่ไม่ได้รับอนุญาต:

- ตรวจสอบว่าเป็นซอฟต์แวร์ที่ไม่ได้รับอนุญาตจริงหรือไม่
- พิจารณาว่าควรอนุญาตซอฟต์แวร์นั้นหรือไม่ หากมีเหตุผลทางธุรกิจที่จำเป็น

หากปัญหาคือ ตรวจพบซอฟต์แวร์ที่ไม่ได้รับอนุญาตบนระบบ:

- ตรวจสอบว่าผู้ใช้หรือผู้ดูแลระบบมีสิทธิ์ติดตั้งซอฟต์แวร์หรือไม่
- ตรวจสอบว่าเครื่องมือที่ใช้ติดตั้งซอฟต์แวร์มีการติดตั้งซอฟต์แวร์ที่เลิกใช้ไปแล้วหรือไม่

3.8 ประเด็นด้านความปลอดภัยของข้อมูล (Data Security Issues)

ปัญหาด้านความปลอดภัยของข้อมูลมีขอบเขตกว้างขวาง แต่มีบางประเด็นเฉพาะที่ควรตรวจสอบ เมื่อต้องสงสัยว่ามีการละเมิดข้อมูล

ข้อมูลที่ไม่ถูกเข้ารหัส (Unencrypted Data)

ในหลายกรณี ข้อมูลควรได้รับการเข้ารหัสทั้งขณะส่งผ่านเครือข่าย (In Transit) และขณะจัดเก็บ (At Rest) วัตถุประสงค์ของการเข้ารหัสข้อมูลมี 3 ประการ:

- ความถูกต้อง (Integrity): เพื่อให้แน่ใจว่าข้อมูลไม่ได้ถูกเปลี่ยนแปลงโดยไม่ได้ตั้งใจ

- ความลับ (Confidentiality): เพื่อให้เฉพาะผู้ที่ได้รับอนุญาตเท่านั้นที่สามารถเข้าถึงข้อมูลได้
- ความถูกต้องของแหล่งที่มา (Authenticity): เพื่อยืนยันว่าแหล่งที่มาของข้อมูลเป็นของแท้

หากปัญหาคือ มีการเปิดเผยข้อมูลทรัพย์สินทางปัญญา:

ให้ตรวจสอบว่า:

- มีการใช้ระบบป้องกันการสูญหายของข้อมูล (DLP) ครอบคลุมพื้นที่จัดเก็บข้อมูลทั้งหมดหรือไม่
- การรับส่งข้อมูลผ่านเครือข่ายมีการเข้ารหัสหรือไม่
- ข้อมูลที่จัดเก็บอยู่ได้รับการเข้ารหัสหรือไม่

หากปัญหาคือ มีการเปิดเผยข้อมูลส่วนบุคคลของลูกค้าหรือพนักงาน:

ให้พิจารณา:

- มีการเข้ารหัสการรับส่งข้อมูลเครือข่ายหรือไม่
- มีการเข้ารหัสข้อมูลที่จัดเก็บอยู่หรือไม่
- ระบบโครงสร้างพื้นฐานเปิดให้บุคคลภายนอกเข้าถึงได้โดยไม่ได้รับอนุญาตหรือไม่
- สิทธิ์และการควบคุมการเข้าถึงถูกกำหนดไว้อย่างถูกต้องหรือไม่

การรั่วไหลของข้อมูล (Data Breach)

การรั่วไหลของข้อมูลหมายถึงการเข้าถึงข้อมูลขององค์กรโดยไม่ได้รับอนุญาต ซึ่งอาจเกิดจากบุคคลภายนอกหรือผู้ไม่ประสงค์ดีภายนอกที่เจาะระบบเข้ามา

หากปัญหาคือ การลักลอบนำข้อมูลออกจากระบบ (Data Exfiltration):

ให้พิจารณา:

- มีการตั้งค่าระดับการจัดประเภทข้อมูล (Classification) ที่เหมาะสมหรือไม่
- สิทธิ์หรือการควบคุมการเข้าถึงถูกกำหนดไว้ผิดพลาดหรือไม่
- การรับส่งข้อมูลภายนอกเครือข่ายหรือระหว่างระบบภายนอกกับคลาวด์ไม่มีการเข้ารหัสหรือไม่
- ข้อมูลที่เก็บในระบบคลาวด์ไม่มีการเข้ารหัสหรือไม่

การจัดประเภทข้อมูลผิดพลาด (Data Misclassification)

การจัดประเภทข้อมูลผิดพลาดอาจนำไปสู่การควบคุมการเข้าถึงที่ไม่เหมาะสม เช่น ข้อมูลลับถูกระบุว่าเป็นข้อมูลสาธารณะ ซึ่งทำให้ระบบอัตโนมัติอนุญาตให้เผยแพร่ต่อสาธารณะโดยไม่ได้ตั้งใจ

หากปัญหาคือ มีการจัดประเภทข้อมูลผิดพลาด:

ให้พิจารณา:

- ระบบอัตโนมัติสำหรับการจัดประเภทข้อมูลถูกกำหนดค่าไว้อย่างถูกต้องหรือไม่
- ระบบอัตโนมัติดังกล่าวได้ถูกนำมาใช้กับพื้นที่จัดเก็บข้อมูลที่เหมาะสมหรือไม่
- การจัดประเภทข้อมูลด้วยตนเองมีความแม่นยำเพียงพอหรือไม่
- บทบาทและสิทธิ์ของผู้ดูแลข้อมูลจัดอยู่ในระดับที่เหมาะสมหรือไม่

สรุปความเข้าใจด้านการปฏิบัติตามข้อกำหนดด้านความมั่นคงปลอดภัยและการแก้ไขปัญหา (Summary Comprehending Security Compliance and Troubleshooting)

โดยสรุป องค์กรจำเป็นต้องมีความเข้าใจอย่างชัดเจนว่าตนเองมีข้อมูลประเภทใดอยู่บ้าง ซึ่งสามารถจัดการได้ง่ายขึ้นผ่านกระบวนการจัดประเภทข้อมูล (Data Classification) เมื่อสามารถระบุข้อมูลได้แล้ว ขั้นตอนถัดไปคือการทราบแหล่งที่จัดเก็บข้อมูลเหล่านั้น เนื่องจากสถานที่จัดเก็บจะเป็นตัวกำหนดว่าหน่วยงานใดมีอำนาจกำกับดูแลข้อมูลดังกล่าว ไม่ว่าจะเป็นหน่วยงานด้านกฎหมายหรือหน่วยงานที่เกี่ยวข้องกับข้อบังคับเฉพาะ

องค์กรยังต้องประเมินความสอดคล้องของผู้ให้บริการระบบคลาวด์ (Cloud Service Providers – CSPs) ว่าเป็นไปตามข้อกำหนดทางกฎหมายและข้อบังคับหรือไม่ นอกจากนี้ กระบวนการทางธุรกิจภายในขององค์กรเองก็ต้องดำเนินการให้สอดคล้องกับข้อกำหนดเหล่านี้ด้วย โดยสามารถประเมินความสอดคล้องผ่านมาตรฐานที่ได้รับการยอมรับในระดับสากลและแนวทางปฏิบัติที่ดี (Best Practices)

กลยุทธ์การบริหารจัดการข้อมูลที่แข็งแกร่งจะช่วยสร้างความเชื่อมั่นและความพร้อมใช้งาน (Availability) ระหว่างลูกค้า ผู้จำหน่าย และบุคคลภายนอกที่เกี่ยวข้อง อย่างไรก็ตาม การดำเนินการให้ได้ตามกลยุทธ์ดังกล่าวอาจมีความซับซ้อนยิ่งขึ้นในกรณีที่องค์กรดำเนินธุรกิจในระดับสากล ซึ่งเป็นเรื่องที่พบบ่อยในระบบคลาวด์

การนำแนวทางปฏิบัติด้านความมั่นคงปลอดภัยที่เหมาะสมมาใช้ร่วมกับกระบวนการแก้ไขปัญหาที่มีประสิทธิภาพ จะช่วยให้องค์กรรักษาความมั่นคงปลอดภัย ปฏิบัติตามข้อกำหนด และลดความเสี่ยงจากการถูกคุกคาม ควรมีการศึกษาและทำความเข้าใจข้อกำหนดทางกฎหมายและข้อกำหนดดูแลที่เกี่ยวข้องกับแต่ละประเทศ และแต่ละอุตสาหกรรมที่องค์กรเข้าไปดำเนินธุรกิจ

แนวทางด้านความมั่นคงปลอดภัยหลายประการ เช่น หลักการกำหนดสิทธิ์ตามความจำเป็นเท่านั้น (Principle of Least Privilege) และขั้นตอนการเพิ่มความแข็งแกร่งของระบบ (System Hardening) มักถูกนำมาใช้ทั้งในระบบภายในองค์กร (On-Premises) และในระบบคลาวด์ การติดตามและตรวจสอบระบบอย่างมีประสิทธิภาพ (Monitoring) ถือเป็นปัจจัยสำคัญในการตรวจจับและจัดการเหตุการณ์ด้านความมั่นคงปลอดภัยได้อย่างทันท่วงที

บทที่ 11

การดำเนินการด้านประสิทธิภาพและการตรวจสอบระบบ (Implementing Performance and Monitoring)

บทนำของบทเรียน

การเพิ่มประสิทธิภาพและการปรับขนาดทรัพยากรระบบคลาวด์ (Cloud Resources) อย่างเหมาะสมช่วยให้มั่นใจได้ว่าระบบจะสามารถให้บริการได้อย่างมีประสิทธิผล คุ้มค่า และตอบสนองต่อความต้องการของผู้ใช้ในระดับราคาที่เหมาะสม โดยเป้าหมายหลักคือการให้บริการคลาวด์ที่มีคุณภาพสูงสุดในต้นทุนที่สมเหตุสมผล ซึ่งสามารถบรรลุได้ผ่านการตรวจวัด (Observability) ที่ครอบคลุมทั้งในด้านตัวชี้วัด (Metrics), บันทึกเหตุการณ์ (Logs) และข้อมูลด้านประสิทธิภาพในทุกช่วงของวงจรชีวิตระบบคลาวด์

วัตถุประสงค์ของบทเรียน

ในบทเรียนนี้ ผู้เรียนจะสามารถ:

- ศึกษาเครื่องมือและกลยุทธ์สำหรับการเพิ่มประสิทธิภาพของบริการระบบคลาวด์ประเภทต่าง ๆ
- ทบทวนเทคนิคการปรับขนาดระบบที่จำเป็น
- ศึกษาตัวเลือกและการกำหนดค่าการตรวจวัดระบบ (Observability)
- ทำความเข้าใจวงจรชีวิตของทรัพยากรระบบคลาวด์ ตั้งแต่ขั้นตอนการพัฒนาและการนำไปใช้งานไปจนถึงการดูแลรักษาและการเลิกใช้งานระบบ

หน่วยที่ 1 การเพิ่มประสิทธิภาพการประมวลผลงานโดยใช้ทรัพยากรระบบคลาวด์ (Optimize Workloads Using Cloud Resources)

การเพิ่มประสิทธิภาพของการประมวลผลงาน (workload optimization) ในสภาพแวดล้อมของระบบคลาวด์สามารถจำแนกได้เป็นสองประเด็นสำคัญ ได้แก่ 1) ความพร้อมใช้งานของทรัพยากร (resource availability) และ 2) ประเด็นที่เกี่ยวข้องกับแอปพลิเคชัน (application-related issues) การวิเคราะห์และจัดการปัญหาทั้งสองประเภทนี้เป็นสิ่งสำคัญสำหรับการรักษา紀錄ดับประสิทธิภาพที่เหมาะสมของบริการคลาวด์

1.1 การใช้งานทรัพยากรการประมวลผล (Implement Compute Resources)

การจัดการทรัพยากรการประมวลผล (Compute Resources) ครอบคลุมถึงการใช้หน่วยประมวลผลกลาง (CPU) หน่วยความจำ (Memory) ตลอดจนความสามารถด้านการจัดเก็บข้อมูลและเครือข่าย ซึ่งทรัพยากรเหล่านี้ถูกจัดสรรให้แก่เครื่องเสมือน (Virtual Machines: VM), คอนเทนเนอร์ (Containers) และระบบแบบไร้เซิร์ฟเวอร์ (Serverless Workloads)

การกำหนดสเปกที่เหมาะสมสำหรับทรัพยากรการประมวลผลในระบบคลาวด์มักพิจารณาจากประเภทของอินสแตนซ์ (Instance Types) ที่ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers: CSPs) กำหนดไว้ล่วงหน้า โดยแต่ละประเภทจะถูกปรับให้เหมาะสมกับการประมวลผล การจัดเก็บ หรือการเชื่อมต่อเครือข่าย เพื่อให้ได้สมดุลระหว่างประสิทธิภาพและต้นทุน ผู้ดูแลระบบจึงเลือกใช้ประเภทของอินสแตนซ์ที่สอดคล้องกับลักษณะของงาน เช่น ฐานข้อมูล หรือระบบพัฒนาโปรแกรม

การเพิ่มประสิทธิภาพของคอนเทนเนอร์จำเป็นต้องเลือกใช้อเอนจินที่เหมาะสม เช่น Docker ควบคู่กับความสามารถในการอัตโนมัติและควบคุมผ่านระบบ Orchestration เช่น Kubernetes เพื่อให้ระบบมีความพร้อมใช้งาน (Availability) และสามารถปรับขยาย (Scalability) ได้ตามต้องการ ทั้งนี้ การจัดการการสื่อสารในเครือข่ายของคอนเทนเนอร์ก็เป็นองค์ประกอบสำคัญที่ต้องวางแผนอย่างรอบคอบโดยผู้ให้บริการคลาวด์หลายแห่งมีเครื่องมือในการติดตามและควบคุมจราจรข้อมูลในระดับคอนเทนเนอร์

สำหรับระบบประมวลผลแบบไร้เซิร์ฟเวอร์ (Serverless Computing) การจัดสรรทรัพยากรจะเป็นแบบไดนามิกตามการเรียกใช้งานโค้ด ซึ่งแนวทางการเพิ่มประสิทธิภาพจะรวมถึงการกำหนดขนาดทรัพยากรที่เหมาะสม (Rightsizing), การใช้การปรับขนาดอัตโนมัติ (Auto-scaling) และการเพิ่มประสิทธิภาพของแอปพลิเคชันหรือโค้ด ทั้งนี้ควรมีการตรวจสอบอย่างสม่ำเสมอเพื่อให้แน่ใจว่ามีการใช้ทรัพยากรอย่างคุ้มค่า

บทเรียนนี้ จะนำเสนอรายละเอียดเพิ่มเติมเกี่ยวกับการใช้เทคโนโลยีเวอร์ชวลไลเซชัน (Virtualization), คอนเทนเนอร์ (Containerization) และโมเดลการทำงานแบบไร้เซิร์ฟเวอร์ (Serverless Execution Model) เพื่อเสริมความเข้าใจในการเพิ่มประสิทธิภาพของระบบคลาวด์

1.2 ทรัพยากรการประมวลผลสำหรับเครื่องเสมือน (Compute Resources for Virtual Machines)

การพิจารณา CPU (vCPU Considerations)

ประเภทของอินสแตนซ์ในระบบคลาวด์มักถูกกำหนดค่ามาไว้ล่วงหน้า โดยเฉพาะในส่วนของหน่วยประมวลผลเสมือน (vCPU) ซึ่งคำนวณจากจำนวนคอร์ (Core) และจำนวนเฟรดต่อคอร์ (Thread/Core):

$$vCPU = \text{จำนวนคอร์} \times \text{จำนวนเฟรดต่อคอร์}$$

ตัวอย่างเช่น อินสแตนซ์ที่มี 4 คอร์ และแต่ละคอร์รองรับ 2 เฟรด จะได้เป็น 8 vCPU

การเลือกใช้ vCPU ควรพิจารณาทั้งในแง่ของประสิทธิภาพและต้นทุน หากไม่จำเป็นต้องใช้พลังประมวลผลสูง ควรลดสเปกเพื่อประหยัดค่าใช้จ่าย นอกจากนี้ การเก็บข้อมูลประสิทธิภาพของระบบยังช่วยให้สามารถปรับปรุงค่าการประมวลผลให้เหมาะสมได้มากขึ้นในอนาคต

1.3 การพิจารณา GPU (GPU Considerations)

สำหรับงานบางประเภท อาจสามารถถ่ายโอนภาระงานจาก CPU ไปยังหน่วยประมวลผลกราฟิก (GPU) ได้ ซึ่งช่วยให้ CPU มีทรัพยากรเพียงพอสำหรับจัดการกับงานสำคัญอื่น ๆ ตัวอย่างของงานที่เหมาะสมกับ GPU ได้แก่:

- ปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง (Machine Learning)
- การประมวลผลประสิทธิภาพสูง (High-Performance Computing)
- งานด้านกราฟิกที่ซับซ้อน
- การวิเคราะห์ข้อมูลขนาดใหญ่

ในระบบ AWS การใช้งาน GPU จะมีคุณสมบัติ Autoboost สำหรับปรับความเร็วสัญญาณนาฬิกาอัตโนมัติตามภาระงาน แต่สามารถปิดฟีเจอร์นี้เพื่อปรับความเร็วให้สูงสุดแบบกำหนดเองได้

1.4 การพิจารณาหน่วยความจำ (Memory Considerations)

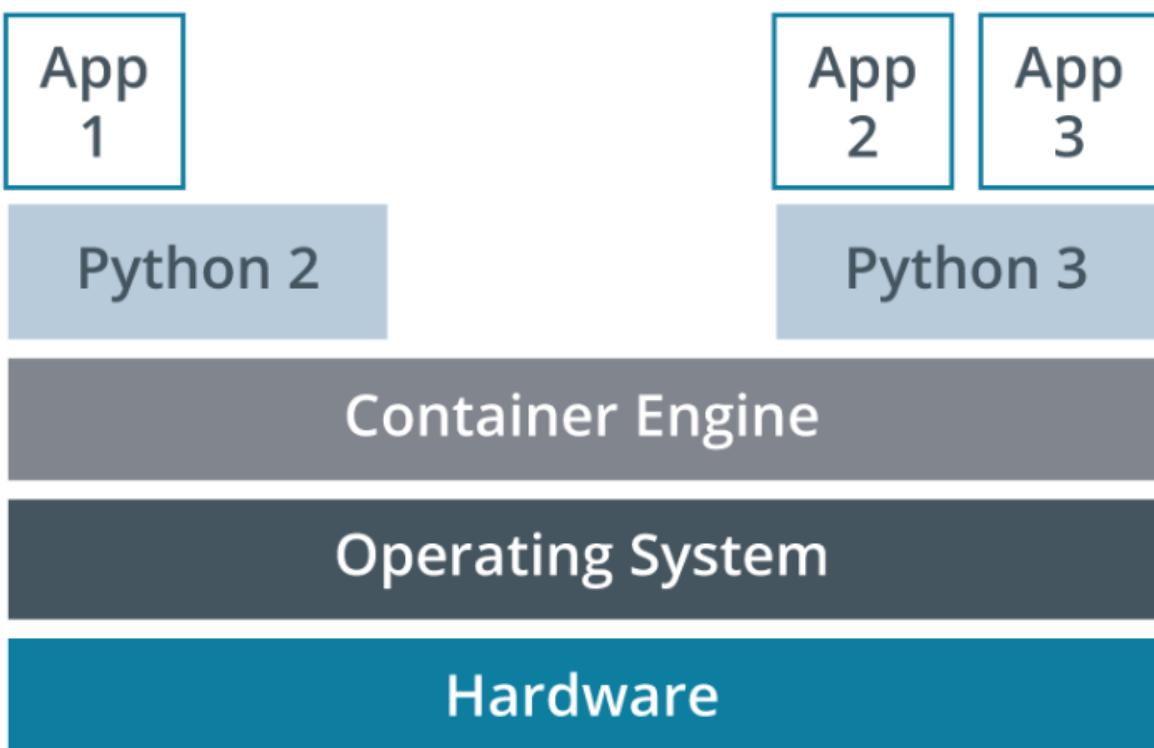
อินสแตนซ์ที่เพิ่มประสิทธิภาพด้านหน่วยความจำ (*Memory-Optimized Instances*) เหมาะสำหรับงานที่ต้องใช้ข้อมูลจำนวนมากในหน่วยความจำ เช่น:

- ระบบฐานข้อมูล (Databases)
- การวิเคราะห์ข้อมูลขนาดใหญ่ (Big Data Analytics)
- ระบบแคลชที่ทำงานในหน่วยความจำ เช่น เว็บแอปพลิเคชัน

การเลือกหน่วยความจำที่เหมาะสมช่วยลดค่าใช้จ่ายได้ เช่นเดียวกับการปรับแต่ง vCPU และยังช่วยให้แอปพลิเคชันทำงานได้อย่างมีประสิทธิภาพสูงสุดตามความต้องการของงาน

1.5 ทรัพยากรการประมวลผลสำหรับคอนเทนเนอร์ (Compute Resources for Containers)

ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers: CSPs) หลายแห่งได้พัฒนาอินสแตนซ์ที่ออกแบบมาโดยเฉพาะสำหรับการใช้งานคอนเทนเนอร์ ซึ่งอินสแตนซ์เหล่านี้สามารถจัดสรรทรัพยากรหน่วยประมวลผลกลาง (CPU) และหน่วยความจำ (Memory) ให้เหมาะสมกับลักษณะของงานได้ ทั้งยังสามารถ “จัดสรรเกินความต้องการ” (Over-allocate) ได้ในบางกรณี เพื่อรับภาระงานที่ผันผวน



App1 requires a Python 2 container, and Apps 2 and 3 require a Python 3 container.

ภาพที่ 44 Compute Resources for Containers

อีกหนึ่งกลยุทธ์ที่ว่าไปในการเพิ่มประสิทธิภาพ คือ การดึงภาพคอนเทนเนอร์ (*Container Images*) มาจัดเตรียมไว้ล่วงหน้าบนไอดีสต์ (*Pre-pull*) เพื่อให้พร้อมสำหรับการนำไปใช้งานได้ทันที ซึ่งจะช่วยลดระยะเวลาในการเริ่มต้นระบบ (*Startup Time*) ของแต่ละคอนเทนเนอร์ ทั้งนี้ อินสแตนซ์สำหรับคอนเทนเนอร์ยังสามารถใช้ระบบปรับขนาดอัตโนมัติ (*Auto-scaling*) เพื่อรับความต้องการที่เปลี่ยนแปลงได้อย่างยืดหยุ่น ประเภทของการทำงานที่เหมาะสมกับอินสแตนซ์ที่เพิ่มประสิทธิภาพสำหรับคอนเทนเนอร์ ได้แก่

- ระบบไมโครเซอร์วิส (*Microservices*)
- แอปพลิเคชันที่ใช้โครงสร้างแบบบริการ (*Service-based Applications*)
- ระบบจัดการเนื้อหา เช่น Drupal และ WordPress
- ระบบฐานข้อมูลเอกสาร เช่น CouchDB

ตัวอย่างสถานการณ์

หากนักพัฒนารายหนึ่งกำลังพัฒนาโปรเจกต์ภาษา Python อยู่ 3 โปรเจกต์ โดยโปรเจกต์หนึ่งใช้ Python เวอร์ชัน 2 ส่วนอีกสองโปรเจกต์ใช้ Python เวอร์ชัน 3 ซึ่งทั้งสองโปรเจกต์หลังมีการพึ่งพาไลบรารีที่ไม่สามารถใช้งานร่วมกันได้ (*Incompatible Dependencies*) การจัดการห้องแม่ดูนี้ในสภาพแวดล้อมเดียวกันบนเครื่องทำงานเครื่องเดียวอาจสร้างความยุ่งยากอย่างมาก

ทางแก้ปัญหาที่เหมาะสมคือ การแยกแต่ละโปรเจกต์ไปทำงานในคอนเทนเนอร์ของตนเอง ซึ่งจะช่วยแยกสภาพแวดล้อมให้เป็นอิสระจากกัน โดยไม่ส่งผลกระทบซึ่งกันและกัน อีกทั้งยังสามารถปรับแต่งแต่ละคอนเทนเนอร์ให้เหมาะสมกับความต้องการเฉพาะของโปรเจกต์นั้น ๆ ได้ เช่น คอนเทนเนอร์ที่รัน Python 2 อาจต้องใช้ทรัพยากรการประมวลผลมากกว่าคอนเทนเนอร์อื่น

แนวทางนี้ไม่เพียงช่วยแก้ปัญหาด้านความเข้ากันได้ของซอฟต์แวร์เท่านั้น แต่ยังส่งเสริมความยืดหยุ่น การขยายตัว และการจัดการทรัพยากรได้อย่างมีประสิทธิภาพอีกด้วย

1.6 ทรัพยากรการประมวลผลสำหรับแอปพลิเคชันแบบไร้เซิร์ฟเวอร์

การประมวลผลแบบไร้เซิร์ฟเวอร์ (Serverless) มีลักษณะแตกต่างจากคอนเทนเนอร์ โดยแอปพลิเคชันแบบไร้เซิร์ฟเวอร์สามารถทำงานได้เฉพาะในสภาพแวดล้อมระบบคลาวด์เท่านั้น ในขณะที่คอนเทนเนอร์สามารถทำงานได้ทั้งในระบบคลาวด์และในระบบในองค์กร (On-premises) แอปพลิเคชันแบบไร้เซิร์ฟเวอร์มักถูกออกแบบให้ทำงานในลักษณะไม่มีสถานะ (Stateless) และโดยทั่วไปจะมีความสามารถในการพกพาได้อยกว่าคอนเทนเนอร์

แนวคิดของระบบไร้เซิร์ฟเวอร์อาจเรียกได้ว่าเป็น “ฟังก์ชันในรูปแบบบริการ” (Function as a Service – FaaS) โดยมีตัวอย่างเช่น Amazon Web Services (AWS) Lambda ซึ่งเป็นแพลตฟอร์มที่รองรับการพัฒนาในลักษณะดังกล่าว

เนื่องจากฟังก์ชันในระบบไร้เซิร์ฟเวอร์เป็นแบบขับเคลื่อนด้วยเหตุการณ์ (Event-driven) โครงสร้างพื้นฐานที่รองรับจึงอาจแตกต่างจากระบบคอนเทนเนอร์ ในบางกรณี อาจเลือกใช้ Spot Instance เพื่อประหยัดต้นทุนในงานประจำหนึ่ง อีกทั้งกระบวนการพัฒนาโปรแกรมสำหรับสภาพแวดล้อมไร้เซิร์ฟเวอร์ยังแตกต่างออกไป โดยสามารถใช้ทรัพยากรที่ไม่ใช่การประมวลผล เช่น AWS StepFunctions เพื่อควบคุมการทำงานแบบลำดับขั้น

ฟังก์ชันแบบไร้เซิร์ฟเวอร์ยังสามารถใช้กลไก “Warm Start” ซึ่งเป็นการเตรียมการล่วงหน้าสำหรับฟังก์ชันที่เรียกใช้งานบ่อย ช่วยลดระยะเวลาการตอบสนอง

เครื่องมือสำหรับการตรวจสอบและติดตามการทำงานของระบบทั้งแบบคอนเทนเนอร์และไร้เซิร์ฟเวอร์ ได้แก่:

- AWS CloudTrail
- AWS CloudWatch
- AWS X-Ray
- Azure Monitor
- Azure Application Insights
- Google Cloud Monitoring
- Google Cloud Logging

1.7 ภาระงานด้านการจัดเก็บข้อมูล (Storage Workloads)

การเพิ่มประสิทธิภาพของการจัดเก็บข้อมูลในระบบคลาวด์ เริ่มต้นจากการเลือกชนิดของอินสแตนซ์ที่เหมาะสมกับลักษณะงาน ผู้ให้บริการระบบคลาวด์ เช่น Google Cloud Platform (GCP) มีอินสแตนซ์ที่ออกแบบมาสำหรับการจัดเก็บข้อมูลโดยเฉพาะ โดยพิจารณาจากลักษณะต่าง ๆ เช่น ประเภทของอุปกรณ์ (SSD หรือ HDD), จำนวนรายการอ่าน/เขียนต่อวินาที (IOPS) และค่าใช้จ่าย

ภาระงานที่เหมาะสมกับอินสแตนซ์ที่เพิ่มประสิทธิภาพด้านการจัดเก็บ ได้แก่:

- ฐานข้อมูล
- การวิเคราะห์ข้อมูล (Analytics)
- ระบบไฟล์แบบกระจายหรือแบบเครือข่าย (Distributed/Network File Systems)

ชั้นการจัดเก็บ (Storage Tiers)

ชั้นการจัดเก็บเป็นแนวทางหนึ่งที่ช่วยให้ผู้ดูแลสามารถจัดสรรพื้นที่จัดเก็บได้อย่างคุ้มค่า โดยแต่ละชั้นมีความเร็วในการเข้าถึงและค่าใช้จ่ายที่แตกต่างกัน เช่นใน AWS S3 ประกอบด้วย:

- Standard:** สำหรับการเข้าถึงข้อมูลบ่อยครั้ง
- Intelligent Tiering:** สำหรับข้อมูลที่มีรูปแบบการเข้าถึงไม่แน่นอน
- Standard Infrequent Access:** สำหรับข้อมูลสำรองที่ไม่ค่อยถูกเข้าถึง
- Glacier:** สำหรับข้อมูลที่เก็บระยะยาวแต่ไม่ค่อยมีการเรียกใช้งาน

Adaptive Optimization เป็นฟีเจอร์อัตโนมัติที่ช่วยสลับชั้นระหว่างชั้นต่าง ๆ ตามลักษณะการใช้งาน เช่น AWS S3 Intelligent Tiering

ค่าการอ่าน/เขียนต่อวินาที (IOPS)

IOPS เป็นหน่วยวัดความเร็วในการเข้าถึงข้อมูล โดยระบุจำนวนรายการอ่านหรือเขียนต่อวินาที เหมาะสมสำหรับแอปพลิเคชันที่มีการดำเนินการกับข้อมูลขนาดเล็กจำนวนมาก เช่นฐานข้อมูล เทคโนโลยีที่รองรับ IOPS สูง ได้แก่:

- AWS Provisioned IOPS SSD
- Azure Ultra Disks
- GCP SSD Persistent Disks

อัตราการถ่ายโอนข้อมูล (Throughput)

Throughput วัดปริมาณข้อมูลที่สามารถส่งผ่านได้ต่อวินาที มีหน่วยเป็น MB/s หรือ GB/s เหมาะกับงานที่ต้องการถ่ายโอนข้อมูลจำนวนมากอย่างรวดเร็ว เช่น วิดีโอสตรีมมิ่ง

ความจุในการจัดเก็บ (Storage Capacity)

การปรับขนาดความจุในการจัดเก็บให้เหมาะสมกับการใช้งานช่วยลดต้นทุน ตัวอย่างเทคนิคที่นิยมใช้ได้แก่:

- **Thin Provisioning:** กำหนดขนาดเริ่มต้นและให้ระบบขยายตามการใช้งานจริง
- **Thick Provisioning:** จัดสรรพื้นที่ทั้งหมดล่วงหน้า
- **Cloud Bursting:** ใช้คลาวด์สาธารณะเสริมเมื่อพื้นที่ส่วนตัวไม่เพียงพอ

การจัดข้อมูลซ้ำซ้อน (Data Deduplication)

Deduplication เป็นกระบวนการลบข้อมูลที่ซ้ำกันและใช้การอ้างอิงแทน หมายความว่า สำหรับข้อมูลเดียวกันที่มีอยู่ในหลายที่ แทนที่จะเก็บทุกที่ แค่เก็บที่หนึ่งแล้วอ้างอิงไปยังที่อื่น

- ระบบสำรองข้อมูล
- ภาพระบบปฏิบัติการ
- เซิร์ฟเวอร์ไฟล์ทั่วไป

การบีบอัดข้อมูล (Data Compression)

การบีบอัดใช้ในการลดขนาดไฟล์เพื่อประหยัดพื้นที่จัดเก็บ แต่ไม่สามารถใช้กับไฟล์ทุกประเภทได้ การตรวจสอบไฟล์ที่เหมาะสมจะช่วยวางแผนบีบอัดได้มีประสิทธิภาพมากขึ้น

1.8 บริการที่มีการจัดการ (Managed Services)

บริการที่มีการจัดการบนระบบคลาวด์ (Cloud-managed services) เป็นแนวทางที่องค์กรสามารถถ่ายโอนภาระด้านการบริหารจัดการระบบคลาวด์ไปยังผู้ให้บริการบุคคลที่สาม ซึ่งผู้ให้บริการเหล่านี้จะรับผิดชอบในหลายด้านของการดำเนินงานระบบคลาวด์ เช่น:

- การออกแบบสถาปัตยกรรมระบบ
- การดำเนินงานระบบประจำวัน
- การให้การสนับสนุนทางเทคนิค
- การให้บริการไฮสติ๊ก

ผู้ให้บริการบุคคลที่สามสามารถเข้ามาช่วยดำเนินการในส่วนของการย้ายระบบจากโครงสร้างพื้นฐานเดิมไปยังระบบคลาวด์ (on-premises to cloud migration), การปรับแต่งให้เหมาะสม (optimization), การตั้งค่าความมั่นคงปลอดภัย (security configuration) และการบำรุงรักษาในแต่ละวัน

แนวทางนี้มีข้อได้เปรียบทลายประการ เช่น:

- การใช้ความเชี่ยวชาญของผู้ให้บริการ แทนการพัฒนาและรักษาความเชี่ยวชาญไว้ภายในองค์กร
- การควบคุมต้นทุนให้มีประสิทธิภาพ
- การลดความซับซ้อนในการบริหารจัดการระบบ

โดยเฉพาะอย่างยิ่ง ผู้ให้บริการที่มีการจัดการสามารถช่วยองค์กรปรับปรุงประสิทธิภาพของการใช้งานระบบเสมือนจริง (Virtual Machine), คอนเทนเนอร์ (Container), และแอปพลิเคชันแบบไร้เซิร์ฟเวอร์ (Serverless) ซึ่งอาจช่วยลดค่าใช้จ่ายและเพิ่มประสิทธิภาพของการกำหนดค่าระบบได้

ทีมงานของผู้ให้บริการมักประกอบด้วยผู้เชี่ยวชาญด้านสถาปัตยกรรมระบบจัดเก็บข้อมูล และเครือข่าย ซึ่งจะให้คำแนะนำและบริหารจัดการทรัพยากรเหล่านี้อย่างเหมาะสม ผู้ให้บริการเหล่านี้ มีบุคลากรผู้เชี่ยวชาญเฉพาะด้านที่องค์กรภายนอกอาจไม่สามารถจัดจ้างได้อย่างถาวร การใช้บริการผู้ให้บริการที่มีการจัดการจึงทำให้องค์กรสามารถเข้าถึงความรู้ความเชี่ยวชาญนี้ได้โดยตรง

นอกจากนี้ ผู้ให้บริการระบบคลาวด์รายใหญ่บางรายยังมีบริการจัดการที่ให้ลูกค้าสามารถทำงานร่วมกับทีมงานของผู้ให้บริการนั้นโดยตรง และในกรณีที่องค์กรมีการใช้งานระบบคลาวด์หลายราย (Multi-cloud deployments) ก็อาจจำเป็นต้องใช้บริการจากผู้ให้บริการที่มีการจัดการจากภายนอก (Third-party managed service providers)

หน่วยที่ 2 การกำหนดแนวทางการปรับขนาดที่เหมาะสม (Configure Appropriate Scaling Approaches)

การปรับขนาดระบบ (Scaling) เป็นกระบวนการสำคัญที่ช่วยให้บริการบนระบบคลาวด์สามารถรองรับภาระงานที่เปลี่ยนแปลงได้อย่างมีประสิทธิภาพ โดยเฉพาะการปรับขนาดแบบอัตโนมัติ (Auto-scaling) ซึ่งระบบจะเพิ่มหรือลดทรัพยากรการประมวลผลให้เหมาะสมกับโหลดงานที่เกิดขึ้นจริงในช่วงเวลาหนึ่งอย่างรวดเร็วและต่อเนื่องโดยไม่ต้องอาศัยการควบคุมจากผู้ดูแลระบบตลอดเวลา

2.1 แนวทางการปรับขนาดระบบ (Scaling Approaches)

มีแนวทางที่หลากหลายในการปรับขนาดทรัพยากรบนระบบคลาวด์ ซึ่งโดยทั่วไปองค์กรส่วนใหญ่ จะใช้วิธีการปรับขนาดแบบอัตโนมัติ (Auto-scaling) โดยอิงตามตัวชี้วัดหรือเหตุการณ์ที่กำหนดไว้ ขณะที่การปรับขนาดแบบแมนนวลยังคงสามารถทำได้ แต่จะมีความยุ่งยากและใช้เวลามากกว่า การปรับขนาดที่แม่นยำถือเป็นสิ่งสำคัญเพื่อคงไว้ซึ่งความคุ้มค่าในการใช้ทรัพยากร

การปรับขนาดแบบแมนนวลและแบบตามกำหนดเวลา

การปรับขนาดแบบแมนนวลอาจมีประโยชน์ในสถานการณ์ทดสอบระบบหรือในช่วงเบต้า ผู้ดูแลระบบสามารถเพิ่มหรือลดจำนวน VM หรือทรัพยากรอื่นๆ ได้ตามต้องการ แต่กระบวนการนี้มักซ้ำและอาจไม่อิงจากข้อมูลที่เป็นระบบ

อีกแนวทางคือ การปรับขนาดตามกำหนดเวลา (Scheduled Scaling) ซึ่งมีประโยชน์เมื่อผู้ดูแลระบบสามารถคาดการณ์ช่วงเวลาที่มีการใช้งานสูงได้อย่างแม่นยำ เช่น ช่วงโปรโมชันออนไลน์ วันหยุดนักขัตฤกษ์ หรือช่วงเปิดลงทะเบียนงานต่างๆ

การปรับขนาดแบบกระตุ้นตามเหตุการณ์ (Triggered Scaling)

การปรับขนาดแบบกระตุ้นตามเหตุการณ์เป็นองค์ประกอบหลักของการปรับขนาดแบบอัตโนมัติ โดยระบบจะปรับทรัพยากรตามเงื่อนไขที่กำหนดไว้ล่วงหน้า ซึ่งตัวอย่างของตัวชี้วัดที่ใช้ได้แก่:

- แนวโน้มการใช้งาน (Trending): รูปแบบการใช้งานที่ตรวจพบในช่วงเวลาหนึ่ง

- **โหลดงาน (Load):** ตัวชี้วัดด้านประสิทธิภาพ เช่น การใช้ CPU หรือเวลาในการตอบสนอง
- **เหตุการณ์ (Event):** ข้อมูลสถานะที่ได้จากการmonitoring หรือระบบล็อกผู้ดูแลระบบควรตรวจสอบการตั้งค่าการปรับขนาดอัตโนมัติอย่างใกล้ชิดเพื่อหลีกเลี่ยงค่าใช้จ่ายที่ไม่จำเป็น เช่น การเปิดใช้ Spot Instances โดยไม่มีเหตุผลรองรับ ซึ่งอาจส่งผลต่อค่าใช้จ่ายอย่างมีนัยสำคัญ บริการอย่าง AWS CloudWatch สามารถช่วยตรวจสอบและแจ้งเตือนเหตุการณ์เหล่านี้ได้

2.2 รูปแบบการกำหนดค่าการปรับขนาด (Configuration Types for Scaling)

ผู้ดูแลระบบสามารถเลือกปรับขนาดระบบในแนวโน้ม (Horizontal Scaling) หรือแนวตั้ง (Vertical Scaling) ได้ ขึ้นอยู่กับลักษณะของแอปพลิเคชัน งบประมาณที่มี และความต้องการด้านประสิทธิภาพ การปรับขนาดในแนวโน้ม (Horizontal Scaling)

หรือที่เรียกว่า Scaling Out เป็นการเพิ่มจำนวนเซิร์ฟเวอร์ (ทั้งจริงและเสมือน) เพื่อรับภาระงาน เช่น เว็บไซต์ที่ปกติใช้เซิร์ฟเวอร์ 3 เครื่อง เมื่อมีผู้ใช้งานเพิ่มขึ้น ระบบอาจเพิ่มเป็น 5 เครื่องเพื่อรับโหลดที่เพิ่มขึ้น

ข้อดีของการปรับขนาดแนวโน้ม:

- เพิ่มความพร้อมใช้งาน (High Availability)
- เพิ่มความสามารถในการทนต่อความล้มเหลว (Fault Tolerance)
- รองรับปริมาณงานที่พุ่งสูงได้

การปรับขนาดแนวโน้มหมายความว่าระบบจะเพิ่มลดจำนวนเครื่องตามความต้องการ

การปรับขนาดแนวตั้ง (Vertical Scaling)

หรือที่เรียกว่า Scaling Up เป็นการเพิ่มความสามารถในการประมวลผลของเครื่องเซิร์ฟเวอร์ที่มีอยู่ เช่น เพิ่มหน่วยความจำจาก 8 GB เป็น 16 GB เพื่อรับข้อมูลที่มีความต้องการสูงขึ้น

ข้อดีของการปรับขนาดแนวตั้ง:

- หมายความว่าการเพิ่มประสิทธิภาพที่สามารถคาดการณ์ได้
- ใช้ได้ในระบบภายในองค์กร
- ค่าใช้จ่ายในการอัปเกรดมักน้อยกว่าการซื้อเครื่องใหม่
- กระบวนการตั้งค่าง่ายกว่าการเพิ่มเครื่องใหม่

แต่การปรับขนาดแนวตั้งมีข้อจำกัดด้านฮาร์ดแวร์ เช่น จำนวน CPU และ RAM ที่สามารถติดตั้งได้ สูงสุด และในบางกรณีอาจต้องมีการหยุดระบบเพื่อทำการอัปเกรด

2.3 การใช้คลาวด์เบิร์สต์ (Cloud Bursting)

Cloud Bursting คือการใช้ทรัพยากรของระบบคลาวด์สาธารณะชั่วคราว เมื่อระบบคลาวด์ภายใน (Private Cloud) มีการใช้งานสูงเกินขีดจำกัด โดยภาระงานจะถูกย้ายไปยังระบบคลาวด์สาธารณะเพื่อรักษา ระดับการให้บริการ แม้จะมีค่าใช้จ่ายเพิ่มเติม แต่ก็ช่วยหลีกเลี่ยงการหยุดชะงักของระบบได้

2.4 การแก้ไขปัญหาการปรับขนาดอัตโนมัติ (Troubleshoot Auto-scaling Issues)

การมอนิเตอร์อย่างมีประสิทธิภาพจะช่วยให้สามารถตรวจสอบปัญหาเกี่ยวกับการใช้ทรัพยากรทั้งที่ใช้งานมากเกินไปหรือน้อยเกินไป และช่วยให้สามารถระบุบริการหรือแอปพลิเคชันที่ควรได้รับการปรับขนาดได้อย่างถูกต้อง

หากพบว่า ระบบไม่ได้ปรับขนาดตามที่ต้องการ ให้พิจารณาดังนี้:

- ได้กำหนดค่า AutoScaling ไว้อย่างถูกต้องหรือไม่
- แอปพลิเคชันหรือบริการนั้นสามารถได้รับประโยชน์จากการปรับขนาดแวนวนหรือไม่
- แอปพลิเคชันหรือบริการนั้นเหมาะสมกับการปรับขนาดตั้งหรือไม่

ยกตัวอย่างเช่น ระบบอาจไม่ปรับขนาด เพราะไม่มีการตั้งค่าทริกเกอร์ที่เหมาะสมไว้ หรือระดับสมาชิกบริการที่ใช้อยู่อาจไม่รองรับฟีเจอร์ auto scaling

หน่วยที่ 3 การกำหนดค่าทรัพยากรเพื่อให้เกิดการสังเกตการณ์ (Configure Resources to Achieve Observability)

การสังเกตการณ์ (Observability) เป็นกระบวนการที่ช่วยให้นักพัฒนา ผู้ดูแลระบบ รวมถึงผู้ใช้งานสามารถเรียนรู้และเข้าใจการทำงานของบริการและแอปพลิเคชันบนระบบคลาวด์ได้อย่างลึกซึ้ง โดยใช้เครื่องมือต่าง ๆ เช่น การมอนิเตอร์ (Monitoring) การบันทึกเหตุการณ์ (Logging) และการติดตามการทำงาน (Tracing) เพื่อร่วมข้อมูลมาใช้ในการวิเคราะห์ ทั้งในรูปแบบแม่นวลดและแบบอัตโนมัติ

การวิเคราะห์ข้อมูลที่ได้จากการสังเกตการณ์มีบทบาทสำคัญในการ เพิ่มประสิทธิภาพของการใช้ทรัพยากร และช่วยให้สามารถ ปรับปรุงกระบวนการทำงาน ได้ดียิ่งขึ้น ตัวอย่างที่เห็นได้ชัดของการปรับปรุงกระบวนการดังกล่าว คือ การนำแนวทาง DevOps และระบบ Continuous Integration/Continuous Deployment (CI/CD) มาใช้ในงานด้านไอที

3.1 การสังเกตการณ์บริการคลาวด์ (Cloud Service Observability)

การบริหารจัดการแบบรวมศูนย์ (Centralized Administration) ถือเป็นข้อได้เปรียบที่สำคัญของบริการคลาวด์ การออกแบบสถาปัตยกรรมระบบคลาวด์ที่เหมาะสมช่วยให้ผู้มีอำนาจในการตัดสินใจสามารถเข้าถึงและควบคุมองค์ประกอบต่าง ๆ ของระบบได้อย่างทั่วถึง การสังเกตการณ์ในลักษณะนี้ครอบคลุมถึงการบริหารจัดการทรัพยากร การตรวจสอบการใช้บริการ การเชื่อมต่อระบบเครือข่าย และสุขภาพโดยรวมของสภาพแวดล้อมระบบคลาวด์ โดยข้อมูลที่เกี่ยวกับสมรรถนะและสถานะของระบบจะช่วยให้ผู้ดูแลสามารถประเมินความพร้อมในการให้บริการและวางแผนด้านกำลังการผลิตได้อย่างแม่นยำ

3.2 การติดแท็ก (Tagging)

การติดแท็กคือการกำหนด **ป้ายกำกับ (Label)** ให้กับทรัพยากรต่าง ๆ เพื่อใช้ในกระบวนการกำหนดค่าและควบคุมค่าใช้จ่าย ตัวอย่างเช่น บนระบบ Amazon Web Services (AWS) ผู้ดูแลระบบสามารถใช้ Tag ในการระบุว่าทรัพยากร EC2 ทั้งหมด 12 เครื่องนั้นอยู่ภายใต้หน่วยพัฒนาซอฟต์แวร์ เมื่อมีการใช้งานระบบบันทึกค่าใช้จ่ายและเครื่องมือวิเคราะห์ เช่น AWS Billing and Cost Management Console ข้อมูลการใช้งานเหล่านี้จะถูกแยกออกตาม Tag อย่างชัดเจน

3.3 ค่าใช้จ่าย (Costs)

ข้อมูลจาก Tag สามารถนำมาใช้เพื่อ **จัดสรรค่าใช้จ่ายกลับไปยังหน่วยงานต้นทาง** หรือที่เรียกว่า Chargeback เช่น แผนกไอทีอาจเรียกเก็บค่าใช้บริการจากแผนกพัฒนาเพื่อให้ครอบคลุมต้นทุนการใช้งานคลาวด์ และยังสามารถแสดงผลแบบ Showback รายปี เพื่อให้หน่วยงานทราบถึงปริมาณการใช้ทรัพยากรที่แท้จริง

3.4 การใช้งานแบบยืดหยุ่น (Elasticity Usage)

การตรวจสอบความสามารถในการขยายตัวของระบบคลาวด์ (Elasticity) ช่วยให้ผู้บริหารทรัพยากรสามารถวิเคราะห์แนวโน้มการใช้งานทั้งในรายปี รายฤดูกาล หรือพัฒนาระบบเปลี่ยนแปลงตามความต้องการ เพื่อให้สามารถจัดสรรงบประมาณการดำเนินงาน (OpEx) ได้อย่างมีประสิทธิภาพ

3.5 การเชื่อมต่อระบบเครือข่าย (Connectivity)

การรายงานข้อมูลเกี่ยวกับการเชื่อมต่อระบบเครือข่ายจะช่วยให้ทราบถึงตำแหน่งทางภูมิศาสตร์ของผู้ใช้งาน รวมถึงรูปแบบของการรับส่งข้อมูลที่อาจเปลี่ยนแปลงไปตามเวลา ตัวอย่างของการเชื่อมต่อ ได้แก่:

- การเชื่อมต่อแบบตรงจากศูนย์ข้อมูลขององค์กร (Hybrid Cloud)
- การเชื่อมต่อผ่าน VPN จากสำนักงานสาขา
- การเชื่อมต่อ VPN จากผู้ใช้งานระยะไกลหรือทำงานจากบ้าน
- ตำแหน่งผู้ใช้บริการผ่านเว็บแอปพลิเคชัน

ข้อมูลสถานที่เหล่านี้สามารถนำไปใช้กำหนด ภูมิภาคที่เหมาะสม สำหรับจัดเก็บข้อมูล และยังสามารถนำไปออกแบบระบบ Content Delivery Network (CDN) เพื่อเพิ่มความเร็วในการเข้าถึงบริการ

3.6 การใช้งานโดยรวม (Overall Utilization)

ข้อมูลจากรายงานและแดชบอร์ดช่วยให้มองเห็นภาพรวมของสถานะการให้บริการ ประสิทธิภาพของเครื่องและระบบเครือข่าย การวางแผนกำลังการผลิต และการจัดการเหตุการณ์ โดยเฉพาะในระบบที่มีการใช้มัลติคลาวด์ (Multi-cloud)

ตัวอย่างเครื่องมือที่นิยมใช้:

- AWS CloudWatch – สำหรับการรวมข้อมูลและสร้างรายงาน
- Azure Monitor – ใช้กับระบบคลาวด์ของ Microsoft
- Google Cloud Monitoring – สำหรับระบบคลาวด์ของ Google

3.7 การบันทึกเหตุการณ์ (Logging Collection)

ระบบปฏิบัติการ Linux และ Windows มีบริการบันทึกเหตุการณ์เป็นของต้นเอง ได้แก่:

- rsyslog – ใช้ใน Linux สำหรับจัดเก็บ log ที่ตำแหน่ง /var/log
- Event Viewer – ใช้ใน Windows สำหรับดู log ที่เกี่ยวกับระบบ แอปพลิเคชัน และผู้ใช้ ทั้งสองระบบสามารถส่ง log ไปยังระบบคลาวด์หรือเซิร์ฟเวอร์ศูนย์กลางเพื่อการจัดเก็บและวิเคราะห์เพิ่มเติมได้

3.8 การรวมล็อก (Log Aggregation)

ผู้ดูแลระบบคลาวด์สามารถกำหนดค่าเซิร์ฟเวอร์ทั้งในระบบภายในองค์กร (on-premises) และบนระบบคลาวด์ ทั้งแบบพิสิคัลและเวอร์ชัล ที่ใช้ระบบปฏิบัติการ Linux ให้ส่งข้อมูลล็อกไปยังศูนย์กลางเพื่อการตรวจสอบและการจัดเก็บที่มีประสิทธิภาพมากยิ่งขึ้น การกำหนดค่านี้สามารถทำได้ผ่านไฟล์ /etc/rsyslog.conf

ตัวอย่างเช่น หากต้องการส่งล็อกของ cron จากเครื่องต้นทางไปยังเซิร์ฟเวอร์กลางที่อยู่ไอพี 10.1.0.10 โดยใช้พอร์ต UDP 514 จะต้องทำการแก้ไขไฟล์กำหนดค่าบนเครื่องต้นทางก่อน และวิ่งแก้ไขไฟล์เดียวบนเซิร์ฟเวอร์ปลายทางเพื่อให้สามารถรับข้อมูลได้ผ่านพอร์ตดังกล่าว

ในผู้ดูแลระบบปฏิบัติการ Windows ผู้ดูแลระบบสามารถจัดการไฟล์ล็อกผ่าน Event Viewer ซึ่งแบ่งออกเป็นล็อกหลักสามประเภท ได้แก่ System, Security และ Application อย่างไรก็ตาม ยังมีล็อกประเภทอื่น ๆ อีกจำนวนมาก Event Viewer ยังสามารถกำหนดค่าให้ส่งต่อข้อมูลไปยังเซิร์ฟเวอร์รวมล็อกศูนย์กลางได้ ทั้งในสภาพแวดล้อมขององค์กรและระบบคลาวด์

การใช้งานตัวแทนสำหรับการล็อกบนระบบคลาวด์ (cloud logging agent) ช่วยให้สามารถส่งข้อมูลล็อกไปยังคอนโซลวิเคราะห์บนคลาวด์ เพื่อวิเคราะห์และแจ้งเตือนได้แบบอัตโนมัติ

ตัวอย่างเช่น ในการใช้งาน Azure cloud เซิร์ฟเวอร์ทุกเครื่องจะรวบรวมข้อมูลล็อกผ่าน monitoring agent ที่ติดตั้งในเครื่อง และส่งข้อมูลเข้าสู่ระบบวิเคราะห์ศูนย์กลาง ซึ่งในกรณีของ Azure คือแพลตฟอร์ม Azure Monitor

3.9 การวิเคราะห์ไฟล์ล็อก (Log File Analysis)

การวิเคราะห์ล็อกคือกระบวนการทำความเข้าใจและตอบสนองต่อเหตุการณ์ที่ถูกบันทึกไว้ในไฟล์ล็อกโดยแต่ละรายการจะระบุระดับความรุนแรง (severity) รายละเอียดของระบบที่ได้รับผลกระทบ ผู้ใช้ที่เกี่ยวข้อง (ถ้ามี) รวมถึงสาเหตุของเหตุการณ์นั้น ๆ

The image shows two side-by-side tables comparing log severity levels. The left table is from the Windows Event Viewer, showing levels: Information (0), Error (1), Information (2), Information (3), Information (4), Information (5), and Warning (6). The right table is from the rsyslog configuration file, showing levels: KERN_EMERG (0), KERN_ALERT (1), KERN_CRIT (2), KERN_ERR (3), KERN_WARNING (4), KERN_NOTICE (5), KERN_INFO (6), and KERN_DEBUG (7). Both tables map levels 0-5 to the same names (Information, Error, etc.) and levels 6-7 to the same names (Warning, Info, Debug).

			Kernel constant	Level value	Meaning
Information	3/24/2021 7:47:52 AM		KERN_EMERG	0	System is unusable
Error	3/24/2021 7:47:53 AM		KERN_ALERT	1	Action must be taken immediately
Information	3/24/2021 7:47:53 AM		KERN_CRIT	2	Critical conditions
Information	3/24/2021 7:47:53 AM		KERN_ERR	3	Error conditions
Information	3/24/2021 7:47:53 AM		KERN_WARNING	4	Warning conditions
Information	3/24/2021 7:47:52 AM		KERN_NOTICE	5	Normal but significant condition
Warning	3/24/2021 7:47:52 AM		KERN_INFO	6	Informational messages
			KERN_DEBUG	7	Debug-level messages

ภาพที่ 45 Log File Analysis

ทั้งระบบ rsyslog ของ Linux และ Event Viewer ของ Windows มีการจำแนกระดับความรุนแรงของเหตุการณ์ เช่น:

- EMERG (0): ระบบไม่สามารถใช้งานได้
- ALERT (1): จำเป็นต้องดำเนินการทันที
- CRIT (2): grave/wikฤต
- ERR (3): grave/pิดพลาด
- WARNING (4): grave/แจ้งเตือน
- NOTICE (5): เหตุการณ์สำคัญแต่ไม่ใช่ปัญหา
- INFO (6): ข้อมูลทั่วไป
- DEBUG (7): ข้อมูลระดับการดีบัก

การวิเคราะห์ล็อกสามารถใช้ร่วมกับระบบอัตโนมัติเพื่อกระทำการอย่างเมื่อพบรเหตุการณ์ที่กำหนดไว้ล่วงหน้า เช่น เมื่อมีการแจ้งเตือนว่าเนื้อที่จัดเก็บข้อมูลใกล้เต็ม ระบบอัตโนมัติสามารถเรียกใช้สคริปต์เพื่อบีบอัดหรือนำข้อมูลบางส่วนออก

ผู้ให้บริการระบบคลาวด์ เช่น Google Cloud Platform (GCP) Cloud Logging รองรับการวิเคราะห์แบบเรียลไทม์ การแจ้งเตือนอัตโนมัติ รวมถึงการเก็บทราบและการค้นหาข้อมูลมาตรฐาน

3.10 การจัดเก็บไฟล์ล็อก (Log File Retention)

การเก็บรักษาไฟล์ล็อกช่วยให้สามารถตรวจสอบย้อนหลังได้ว่าเกิดอะไรขึ้นบนระบบเชิร์ฟเวอร์โดยการรวบรวมข้อมูลล็อกจากหลายเครื่องหรืออุปกรณ์เครือข่ายช่วยให้เข้าใจถึงความสัมพันธ์ของเหตุการณ์ต่าง ๆ เนื้อหาในล็อกมักประกอบด้วย:

- ใครเป็นผู้กระทำ
- เหตุการณ์เกิดขึ้นเมื่อใด
- เหตุการณ์ใดที่เกิดขึ้น

- เหตุการณ์นั้นมีผลอย่างไรกับระบบ

การเก็บรักษาล็อกสามารถใช้เป็นหลักฐานในการปฏิบัติตามข้อกำหนดทางกฎหมาย (compliance), การยืนยันข้อตกลงระดับบริการ (SLA), การบริหารจัดการความเสี่ยง และการวิเคราะห์ปัญหาของระบบ ประเภทของล็อกที่สำคัญ ได้แก่:

- ล็อกการยืนยันตัวตน
- ล็อกการเข้าถึงไฟล์
- ล็อกระบบ
- ล็อกแอปพลิเคชัน

แม้ว่าข้อมูลในล็อกมักจะถูกใช้งานภายในเวลาอันใกล้หลังเกิดเหตุการณ์ เช่น การอัปเดตระบบที่ล้มเหลว แต่การเก็บรักษาข้อมูลในระยะยาวช่วยให้สามารถวิเคราะห์แนวโน้มการใช้งาน วางแผนด้านขีดความสามารถ และประเมินการปรับขนาดของระบบในอนาคตได้

3.11 โซลูชันการตรวจจับและตอบสนองบนระบบคลาวด์ (Cloud Detection and Response Solutions)

เครื่องมือออนไลน์ (Monitoring utilities) ทำหน้าที่แจ้งเตือนผู้ดูแลระบบเมื่อมีตัวชี้วัด (metrics) หรือเงื่อนไขบางประการถูกตรวจสอบ โดยมักใช้ในบริบทของการตรวจจับ วิเคราะห์ และตอบสนอง ต่อภัยคุกคามด้านความมั่นคงปลอดภัยหรือเหตุขัดข้องของบริการอย่างรวดเร็ว ระบบแดชบอร์ดแสดงข้อมูลภาพรวมในลักษณะภาพ (visual indicators) และยังสามารถส่งการแจ้งเตือนไปยังช่องทางสื่อสารต่าง ๆ เช่น บริการข้อความ (messaging services)

3.12 การแจ้งเตือนเหตุการณ์ (Incident Alerts)

การแจ้งเตือนเหล่านี้ บริหารจัดการผ่านระบบ Cloud Detection and Response (CDR) ซึ่งทำหน้าที่รวบรวมและวิเคราะห์ข้อมูลจากแหล่งต่าง ๆ เช่น ทรัพฟิกเครือข่าย การปรับขนาดระบบ (scaling) บันทึกล็อก (logs) และกิจกรรมอื่น ๆ เพื่อให้ผู้ดูแลระบบมีมุมมองแบบองค์รวมของทรัพยากรบนคลาวด์

ระบบ CDR ช่วยในการระบุภัยคุกคามที่อาจเกิดขึ้น เมื่อเกิดการแจ้งเตือน ระบบจะเชื่อมโยงข้อมูลที่เกี่ยวข้องทั้งหมดไว้กับการแจ้งเตือนนั้น ทำให้ผู้ดูแลระบบหรือเครื่องมืออัตโนมัติสามารถจัดลำดับเหตุการณ์ (triage) ได้อย่างมีประสิทธิภาพ

Triaging คือกระบวนการจัดลำดับความสำคัญของการตอบสนองตามระดับความรุนแรงของเหตุการณ์ ช่วยให้เจ้าหน้าที่ IT ที่ต้องจัดการกับเหตุการณ์หลายรายการพร้อมกันสามารถเลือกจัดการสิ่งที่สำคัญที่สุดก่อน อีกทั้งยังสามารถมอบหมายงานไปยังบุคลากรแต่ละคนได้อย่างชัดเจน ลดความซ้ำซ้อน หรือการละเลยเหตุการณ์บางรายการ

3.13 การตอบสนองต่อเหตุการณ์ (Incident Responses)

เมื่อมีการแจ้งเตือนเกิดขึ้น จะต้องมีการเลือกแนวทางตอบสนองที่เหมาะสม หากไม่มีการตั้งค่าให้ดำเนินการโดยอัตโนมัติ ผู้ดูแลระบบอาจเลือกที่จะปิด (ignore) การแจ้งเตือนหรือดำเนินการตามที่จำเป็น โดยรวมมีการบันทึกเหตุผลในการปิด หรือรายละเอียดของการดำเนินการเพื่อนำไปใช้ในการวิเคราะห์แนวโน้ม และพัฒนาระบบของเหตุการณ์ในอนาคต

การตอบสนองอาจอยู่ในรูปแบบอัตโนมัติหรือดำเนินการด้วยตนเอง เช่น การบล็อกทรัพพิจกรรมเครือข่าย หรือแยกทรัพยากร่างรายการออก

ตัวอย่างการตอบสนองจาก Microsoft สำหรับ Azure ได้แก่:

- บัญชีผู้ใช้งานบุกรุก: ปิดการใช้งานบัญชีนั้นเพื่อตรวจสอบเพิ่มเติม
- มีการสร้างผู้ดูแลระบบใหม่: ตรวจสอบตัวตนของผู้ใช้งานที่เพิ่มเข้ามา
- กิจกรรมต้องสงสัย: ดำเนินการตรวจสอบเพิ่มเติมโดยละเอียด

ผู้ดูแลระบบสามารถสร้างช่องทางการสื่อสาร (communication channels) และกำหนดเกณฑ์การแจ้งเตือน (alert thresholds) ให้ตรงกับความรับผิดชอบของทีมสนับสนุนที่เกี่ยวข้อง

ในตัวอย่างของ Google Cloud Platform (GCP) ผู้ดูแลระบบอาจตั้งค่าให้เปิดเหตุการณ์แจ้งเตือน เมื่อค่าหน่วงเวลา (latency) ของ HTTP เกิน 2 วินาทีติดต่อกันนานกว่า 5 นาที และระบบจะปิดเหตุการณ์โดยอัตโนมัติเมื่อค่าหน่วงเวลาลับมาต่ำกว่าค่าที่กำหนด

ใน Azure ระบบการจัดกลุ่มการแจ้งเตือนเรียกว่า Action Groups

ส่วน AWS ใช้บริการ Simple Notification Service (SNS) โดยผู้ดูแลระบบสามารถสมัครับข้อมูลจากหัวข้อ (topic) ต่าง ๆ ที่กำหนดไว้

3.14 การเปิดใช้งานและปิดใช้งานการแจ้งเตือน (Enable and Disable Alerts)

ระบบมอนิเตอร์สามารถแจ้งเตือนผู้ดูแลระบบผ่านทางอีเมลหรือข้อความ SMS โดยควรระบุความเร่งด่วน (urgency) และความรุนแรง (severity) ของเหตุการณ์เพื่อช่วยในการจัดลำดับความสำคัญ ใน AWS การแจ้งเตือนจะอยู่ในสถานะได้สถานะหนึ่งจาก 3 แบบ ได้แก่:

- OK (สถานะปกติ)
- Alarm (มีปัญหา)
- Insufficient Data (ข้อมูลไม่เพียงพอ)

AWS ยังรองรับการแจ้งเตือนแบบ:

- Single-metric alerts: อิงจากตัวชี้วัดเดียว
- Composite alerts: ใช้หลายตัวชี้วัดร่วมกัน ซึ่งมีความแม่นยำสูงกว่า เพื่อความสามารถของเหตุการณ์เล็ก ๆ ที่อาจไม่เกี่ยวข้องออกໄປได้

อย่างไรก็ตาม ควรหลีกเลี่ยงการแจ้งเตือนที่มากเกินไป ซึ่งอาจทำให้เจ้าหน้าที่สับสนหรือพลาดเหตุการณ์สำคัญ ควรตั้งค่าให้ระบบอยู่ในโหมดบำรุงรักษา (maintenance mode) ในช่วงที่มีการทดสอบหรือปรับปรุงระบบใหม่

โหมดบำรุงรักษา yang หมายความว่า สำหรับสถานการณ์ที่มีการเปลี่ยนแปลงค่าการตั้งค่าด้วยตนเอง เช่น การเปลี่ยนอุปกรณ์ฮาร์ดแวร์ หรือการปรับโครงสร้างระบบ เพื่อหลีกเลี่ยงการเกิดแจ้งเตือนที่ไม่จำเป็น

หน่วยที่ 4 การจัดการของทรัพยากรบนคลาวด์ (Managing the Life Cycle of Cloud Resources)

ทรัพยากรบนระบบคลาวด์ มีชีวิต (Lifecycle) ที่คล้ายคลึงกับข้อมูลและบริการที่อยู่ในระบบภายในองค์กร (On-premises) โดยจะมีชีวิตตั้งแต่ตัวเริ่มต้นจนถึงสิ้นสุด แบ่งออกเป็นระบบ 4 ระยะ ได้แก่ การพัฒนา (Development) การปรับใช้ (Deployment) การบำรุงรักษา (Maintenance) และการลิขิต (Decommissioning)

ในบรรดาภาระทั้งหมด ภาระการบำรุงรักษา ถือเป็นช่วงที่ใช้ระยะเวลาที่สุด โดยประกอบไปด้วย การอัปเดตระบบในระดับต่าง ๆ ทั้งการอัปเดตหลัก (Major Updates) การอัปเดตร่อง (Minor Updates) รวมถึงการติดตั้งแพทช์ (Patches) เพื่อแก้ไขช่องโหว่ด้านความปลอดภัยหรือปรับปรุงประสิทธิภาพการทำงาน

4.1 แผนที่เส้นทางของชีวิตของทรัพยากรคลาวด์ (The Lifecycle Roadmap)

การดำเนินงานในระบบคลาวด์นั้นไม่ใช่เพียงแค่การจัดการในชีวิตประจำวันของเครื่อง VM แต่เป็นการจัดการข้อมูล หรือการปรับใช้บริการเท่านั้น แต่ยังรวมถึงการเข้าใจวิธีการจัดการของทรัพยากรบนคลาวด์ แอปพลิเคชันและบริการด้วย

“แผนที่เส้นทางของชีวิต (Lifecycle Roadmap)” คือการติดตามระยะต่าง ๆ ของระบบ ซึ่งสามารถใช้ได้ทั้งกับระบบในองค์กร (On-premises) และระบบคลาวด์ โดยมักครอบคลุม 5 ระยะหลัก ดังนี้:

1. ระยะพัฒนา (Development Phase)

ในระยะนี้ แอปพลิเคชันหรือบริการที่ได้รับอนุมัติให้นำมาพัฒนาจะถูกบันทึกไว้ในแผนที่วางแผนชีวิต เพื่อใช้ในการติดตามความคืบหน้า โดยอาจเป็นเวอร์ชันทดสอบ (Beta) ที่ยังไม่เปิดใช้งานจริง

2. ระยะปรับใช้ (Deployment Phase)

เป็นช่วงที่แอปพลิเคชันหรือบริการถูกนำเข้าสู่การใช้งานในระบบจริง โดยมักมีการตั้งค่าเวลาทดสอบ (Staging) และปรับใช้ด้วยเครื่องมืออัตโนมัติ เช่น Ansible และอาจมีการระบุว่าเป็นเวอร์ชัน “Long-Term Support (LTS)” หรือเวอร์ชันเสถียร

3. ระยะบำรุงรักษา (Maintenance Phase)

เป็นระยะที่ยาวนานที่สุด โดยระบบจะถูกใช้งานจริงและมีการดูแลรักษา เช่น การติดตั้งแพทช์ การอัปเดตซอฟต์แวร์ และการย้ายบริการไปยังระบบที่มีประสิทธิภาพสูงขึ้น เช่น การย้ายจากระบบภายในไปยังระบบคลาวด์ หรือจากเครื่องจริงไปยัง VM (P2V)

4. ระยะเลิกใช้งาน (Deprecation Phase)

เป็นการประกาศว่าจะยุติการให้บริการหรือสนับสนุนซอฟต์แวร์นั้น โดยยังมีการสนับสนุนอยู่ในช่วงเวลาหนึ่ง ก่อนถึง “End of Support” ตัวอย่างเช่น Ubuntu 24.04 LTS มีการสนับสนุนถึงปี 2029 และจะหมดอายุ การใช้งานในปี 2036

5. ระยะยกเลิกการให้บริการ (Decommissioning Phase)

หลังจากประกาศเลิกใช้งานแล้ว ขั้นตอนนี้จะเป็นการปิดระบบ เช่น ปิด VM ลบข้อมูล หรือบันทึกขั้นตอน การปิดใช้งานอย่างรอบคอบ รวมถึงตรวจสอบว่าไม่มี “Zombie Workloads” ที่ยังคงอยู่ในระบบ

4.2 การอัปเดตระบบ: อัปเดตหลัก อัปเดตร่อง และแพตช์ (Major and Minor Updates / Patching)

ระบบเวอร์ชันแบบ Semantic (Semantic Versioning):

รูปแบบเวอร์ชันที่ใช้ทั่วไปคือ major.minor.patch

ตัวอย่าง: Ubuntu 22.04

- 22 = เวอร์ชันหลัก
- 04 = เวอร์ชันย่อย
- xx = ระดับแพตช์

การอัปเดตหลัก (Major Update):

มีการเปลี่ยนแปลงใหญ่ อาจไม่สามารถใช้ร่วมกับระบบเดิมได้ (Backward Incompatible) เช่น เปลี่ยนจากเวอร์ชัน 3.x ไปยัง 4.x

การอัปเดตร่อง (Minor Update):

ปรับปรุงเล็กน้อย เช่น แก้บักหรือเพิ่มฟีเจอร์ย่อย โดยไม่กระทบต่อการใช้งานเดิม เช่น จากเวอร์ชัน 3.1 ไป 3.2

การติดตั้งแพตช์ (Patching):

ใช้สำหรับการปรับปรุงความปลอดภัย แก้ไขข้อบกพร่อง และเพิ่มประสิทธิภาพ

- OS, แอปพลิเคชัน, เพิร์มแวร์, ไ/drเวอร์ ล้วนต้องมีการติดตั้งแพตช์
- ในกรณีของบริการ PaaS และ SaaS ผู้ให้บริการคลาวด์ (CSP) มากเป็นผู้รับผิดชอบ
- การตรวจสอบการแพตช์สามารถทำได้ผ่านรายงาน, log files, และฟีดแบคจากผู้ใช้งาน

4.3 การบริหารจัดการการแพตช์ (Patch Management)

เหตุผลในการติดตั้งแพตช์:

- แก้ไขช่องโหว่ด้านความปลอดภัย
- แก้บักของระบบหรือแอป
- เพิ่มหรือปรับปรุงฟีเจอร์

การตรวจสอบความสำเร็จในการแพตช์:

- ใช้เครื่องมือตรวจสอบ เช่น Azure Monitor, Nessus
- ตรวจสอบ log file, รายงานการแพตช์จากระบบจัดการ เช่น Chef, Ansible

แนวทางทั่วไปขององค์กร:

- ใช้หลักการ N-1 คือ เว้นจากแพตช์ล่าสุด 1 เวอร์ชัน เพื่อรอให้ผู้ใช้รายอื่นตรวจสอบก่อน
- สำรวจข้อมูลก่อนติดตั้งแพตช์เสมอ เพื่อรับทราบถ้ามีปัญหาที่ต้อง rollback

รายการส่วนประกอบที่ควรถูกแพตช์:

- Hypervisor
- VM และ Virtual Appliances
- ระบบเครือข่าย (ผ่าน IaC)
- แอปพลิเคชัน
- Storage (รวมถึง Firmware)
- ระบบปฏิบัติการ (OS)

สรุปการดำเนินการด้านประสิทธิภาพและการตรวจสอบระบบ (Summary Implementing Performance and Monitoring)

การตรวจสอบและปรับแต่งทรัพยากรในระบบคลาวด์เป็นสิ่งสำคัญอย่างยิ่งในการรักษาประสิทธิภาพการทำงานให้อยู่ในระดับที่เหมาะสม พร้อมทั้งควบคุมต้นทุนอย่างมีประสิทธิภาพ

หนึ่งในองค์ประกอบที่สำคัญที่สุดคือ “ความสามารถในการสังเกต (Observability)” ซึ่งหมายถึงความสามารถในการดูข้อมูลที่เกี่ยวข้องกับประสิทธิภาพของบริการ โดยข้อมูลเหล่านี้ช่วยให้องค์กรสามารถดำเนินการปรับปรุงการใช้ทรัพยากรให้มีประสิทธิภาพสูงขึ้น และยกระดับประสบการณ์ของผู้ใช้ การเพิ่มประสิทธิภาพของทรัพยากร (Resource Optimization) อาจรวมถึง:

- การจ้างผู้ให้บริการภายนอก (Managed Service Provider: MSP) เพื่อเข้าถึงเครื่องมือและความเชี่ยวชาญเฉพาะด้าน
- การประเมินกระบวนการจัดการแบบอัตโนมัติ (Orchestration Workflows) ว่าใช้ได้อย่างมีประสิทธิภาพ หรือไม่ และเป็นกระบวนการที่ถูกต้องเหมาะสม

บทที่ 12

การจัดการการกู้คืนจากภัยพิบัติและความต่อเนื่องทางธุรกิจ (Managing Disaster Recovery and Business Continuity)

บทนำของบทเรียน

แนวคิดของการกู้คืนจากภัยพิบัติ (Disaster Recovery) ครอบคลุมถึงการสำรองและการกู้คืนข้อมูลอย่างมีประสิทธิภาพ เพื่อให้สามารถเข้าถึงข้อมูลได้เมื่อต้องการใช้งาน นอกเหนือไปนี้ยังรวมถึงการบริหารจัดการความพร้อมใช้งานของทรัพยากรและบริการผ่านวิธีการต่าง ๆ เช่น การมีระบบสำรอง (Redundancy), การใช้เขตความพร้อมใช้งาน (Availability Zones), การใช้งานระบบแบบมัลติคลาวด์และไฮบริดคลาวด์ รวมถึงการตรวจสอบระบบอย่างมีประสิทธิภาพ

ผู้ดูแลระบบคลาวด์จำเป็นต้องเข้าใจคำศัพท์เฉพาะทางที่เกี่ยวข้องกับการกู้คืนจากภัยพิบัติ รวมถึงหลักเกณฑ์ด้านกฎระเบียบที่เกี่ยวข้อง เพื่อให้สามารถจัดการให้ข้อมูลเข้าถึงได้ตามความจำเป็น และได้รับการปกป้องอย่างเหมาะสมในขณะจัดเก็บ

วัตถุประสงค์ของบทเรียน

ในบทเรียนนี้ ผู้เรียนจะได้:

- เข้าใจวิธีการสำรองและการกู้คืนข้อมูลในระบบคลาวด์
- ทบทวนแนวคิดเรื่องความพร้อมใช้งานของบริการ (Service Availability)

หน่วยที่ 1 วิธีการสำรองและการกู้คืนข้อมูล (Backup and Recovery Methods)

นโยบายการสำรองและการกู้คืนข้อมูลถือเป็นองค์ประกอบสำคัญของความพร้อมใช้งาน (Availability) และความมั่นคงปลอดภัยของระบบ (Security) การเลือกประเภทและเป้าหมายของการสำรองข้อมูล (Backup Type & Target) จะเปลี่ยนแปลงไปตามสภาพแวดล้อมของระบบคลาวด์ ในขณะที่ตัวเลือกในการกู้คืนข้อมูล (Restore Options) และตำแหน่งที่ใช้ในการกู้คืน (Recovery Locations) ก็แตกต่างจากระบบในสถานที่ (On-premises) เช่นกัน

1.1 ประเภทของการสำรองข้อมูล (Backup Types)

การสำรองข้อมูลอย่างสม่ำเสมอเป็นสิ่งจำเป็นในระบบคอมพิวเตอร์ เนื่องจากผู้ใช้อาจลบหรือเปลี่ยนแปลงข้อมูลโดยไม่ตั้งใจ อุปกรณ์จัดเก็บข้อมูลอาจเสียหาย หรือเกิดเหตุการณ์ร้ายแรง เช่น ไฟไหม้หรือน้ำท่วม ซึ่งอาจส่งผลกระทบต่ออาคารสถานที่โดยตรง อีกทั้งบางภาคอุตสาหกรรมยังมีข้อกำหนดด้านกฎหมายที่บังคับให้ต้องเก็บข้อมูลเป็นระยะเวลาที่กำหนด

การบริหารจัดการการสำรองข้อมูลต้องพิจารณาสมดุลระหว่างเวลาในการสำรองข้อมูลและเวลาในการกู้คืนข้อมูล โดยมีรูปแบบและกลยุทธ์หลากหลายในการสำรองข้อมูลให้เหมาะสมกับลักษณะการใช้งาน และลักษณะของระบบ เช่น ระบบเซิร์ฟเวอร์แบบตั้งเดิม หรือระบบเครื่องเสมือน (VM)

ในระบบคลาวด์ การสำรองข้อมูลมีข้อพิจารณาเพิ่มเติม โดยเฉพาะประเด็นการลดความเสี่ยง จากการเสียหายของฮาร์ดแวร์ซึ่งเป็นความรับผิดชอบของผู้ให้บริการคลาวด์ (Cloud Service Provider – CSP) อย่างไรก็ตาม ความผิดพลาดของผู้ใช้ เช่น การลบหรือเปลี่ยนแปลงไฟล์โดยไม่ตั้งใจ ยังคงเป็นเหตุผลสำคัญที่ต้องมีการสำรองข้อมูลในระบบคลาวด์

ความแตกต่างระหว่าง STaaS และ BUaaS

- **Storage as a Service (STaaS):** เป็นบริการจัดเก็บข้อมูลทั่วไป ซึ่งสามารถใช้เก็บไฟล์สำรองข้อมูลได้แต่ไม่เหมาะสมสำหรับงานสำรองข้อมูลขนาดใหญ่ เนื่องจากมีค่าใช้จ่ายสูง
- **Backup as a Service (BUaaS):** เป็นบริการสำรองข้อมูลโดยเฉพาะ ช่วยให้ธุรกิจขนาดเล็กและกลางสามารถจัดเก็บข้อมูลสำรองไว้ในระบบคลาวด์ได้อย่างปลอดภัยและมีต้นทุนต่ำ เหมาะสำหรับการใช้งานในระยะยาว

ประเภทของการสำรองข้อมูล

- **Full Backup:** การสำรองข้อมูลทั้งระบบ ใช้เวลานานในการสำรอง แต่สามารถกู้คืนได้รวดเร็ว เนื่องจากใช้ชุดข้อมูลเพียงชุดเดียว
- **Synthetic Full Backup:** การรวมข้อมูลจากการสำรองแบบเต็มครั้งล่าสุด และการสำรองแบบเพิ่มขึ้น (Incremental) เข้าด้วยกันเป็นชุดข้อมูลเดียว
- **Incremental Backup:** การสำรองเฉพาะข้อมูลที่เปลี่ยนแปลงไปจากครั้งก่อน พร้อมรีเซ็ตค่าบิตกำกับข้อมูล (Archive Bit) การสำรองใช้เวลาอ่อนโยน แต่การกู้คืนต้องใช้ชุดข้อมูลหลายชุด
- **Differential Backup:** การสำรองข้อมูลที่เปลี่ยนแปลงตั้งแต่การสำรองแบบเต็ม โดยไม่รีเซ็ต Archive Bit จึงทำให้การสำรองใช้เวลานานขึ้นในแต่ละวัน แต่สามารถกู้คืนได้รวดเร็วกว่าแบบ Incremental
- **Snapshot:** การสำรองสถานะของระบบในช่วงเวลาใดเวลาหนึ่ง เหมาะสำหรับ VM หรือบริการที่ต้องการเก็บการตั้งค่าระบบ เช่น Active Directory หรือ Exchange สามารถนำไปใช้ในการกู้คืนหรือโคลนระบบได้รวดเร็ว
- **Archive:** การเก็บข้อมูลระยะยาวที่ไม่ถูกใช้งานบ่อย โดยเน้นความประหยัดพื้นที่มากกว่าความเร็ว เช่น บริการ AWS Glacier, Azure Archive หรือ GCP Coldline ซึ่งเหมาะสมสำหรับการจัดเก็บข้อมูลตามข้อกำหนดของหน่วยงานกำกับดูแล

1.2 วัตถุประสงค์ของการสำรองข้อมูล (Backup Objects)

งานสำรองข้อมูลสามารถออกแบบให้เหมาะสมกับวัตถุประสงค์ที่หลากหลาย เช่น ปกป้องข้อมูลของผู้ใช้งาน การบันทึกค่าการตั้งค่าระบบ หรือการย้ายบริการไปยังอุปกรณ์ใหม่ โดยมีรูปแบบการสำรองข้อมูลที่พับบอยดังนี้:

- **System-State Backup:** สำรองข้อมูลเฉพาะค่าการตั้งค่าระบบปฏิบัติการ เช่น รีจิสทรีและไฟล์ระบบ แนะนำสำหรับการกู้คืนระบบที่เกิดความผิดพลาดอย่างรวดเร็ว
- **Application-Level Backup:** สำรองไฟล์โปรแกรมและค่าการตั้งค่าเฉพาะของแอปพลิเคชัน แนะนำสำหรับการอัปเกรดหรือย้ายแอปไปยังระบบใหม่
- **Filesystem Backup:** สำรองข้อมูลผู้ใช้ทั่วไป เช่น เอกสารหรือสเปรดชีต ซึ่งมีการเปลี่ยนแปลงบ่อย จึงมักมีการสำรองแบบรายวัน
- **Database Dump:** เป็นการสำรองข้อมูลฐานข้อมูลโดยแปลงข้อมูลเป็นคำสั่ง SQL เพื่อให้สามารถกู้คืนหรือย้ายฐานข้อมูลได้ โดยไม่จำเป็นต้องปิดระบบบริการ
- **Configuration File Backup:** สำรองไฟล์ที่ใช้กับระบบอัตโนมัติหรือระบบจัดการการตั้งค่า เช่น Ansible, Chef, Puppet หรือ PowerShell DSC เพื่อป้องกันการสูญหายของสคริปต์หรือบทเรียนที่ปรับแต่งไว้

1.3 สถานที่จัดเก็บข้อมูลสำรอง (Backup Locations)

การเลือกสถานที่จัดเก็บข้อมูลสำรองถือเป็นการตัดสินใจที่สำคัญของผู้ดูแลระบบ โดยทั่วไปการจัดเก็บข้อมูลสำรองไว้นอกสถานที่ (offsite) มักถือว่ามีความปลอดภัยมากกว่า เนื่องจากสามารถป้องกันความเสียหายจากเหตุการณ์ที่ส่งผลต่อสถานที่หลัก เช่น ไฟไหม้หรืออุทกภัย อย่างไรก็ตาม การเก็บข้อมูลไว้ในสือบันทึก เช่น เทปแม่เหล็กหรือฮาร์ดไดร์ฟที่อยู่นอกสถานที่อาจทำให้ยากต่อการเรียกคืนข้อมูลอย่างเร่งด่วน เช่น กรณีที่ผู้ใช้ลบไฟล์โดยไม่ได้ตั้งใจ

ตัวอย่างสถานที่จัดเก็บข้อมูลสำรองนอกสถานที่ ได้แก่:

- **สำนักงานสาขา:** ข้อมูลสำรองจากสถานที่หนึ่งขององค์กรอาจถูกจัดเก็บไว้ที่อีกสาขาหนึ่งขององค์กร
- **ศูนย์จัดเก็บข้อมูลของบุคคลที่สาม:** มีบริษัทเฉพาะทางที่ให้บริการจัดเก็บสือบันทึกข้อมูลอย่างปลอดภัย
- **คลาวด์สตอร์เจ:** ข้อมูลถูกจัดเก็บไว้ในศูนย์ข้อมูลของผู้ให้บริการคลาวด์ ซึ่งมักมีระบบสำรองและการจำลองข้อมูลไปยังภูมิภาคอื่นเพื่อเพิ่มความปลอดภัย

ข้อดีของการใช้คลาวด์คือ องค์กรไม่ต้องดูแลและรักษาความปลอดภัยของข้อมูลสำรองภายใต้สถานที่อีกต่อไป อย่างไรก็ตาม การกู้คืนข้อมูลต้องพึงพาการเชื่อมต่ออินเทอร์เน็ต จึงยังถือว่าเป็นรูปแบบของการจัดเก็บข้อมูลแบบ offsite เช่นกัน

กฎ 3-2-1 สำหรับการสำรองข้อมูล (3-2-1 Backup Rule)

กฎ 3-2-1 เป็นแนวปฏิบัติที่แนะนำให้มีรูปแบบการจัดเก็บข้อมูลสำรองดังนี้:

- 3 สำเนาของข้อมูล
- 2 รูปแบบของสื่อจัดเก็บที่แตกต่างกัน (เช่น เทป และฮาร์ดดิสก์)
- 1 สำเนาที่จัดเก็บไว้นอกสถานที่

หลักการนี้ช่วยลดความเสี่ยงจากความเสียหายของสื่อบันทึกและช่วยให้สามารถกู้คืนข้อมูลได้อย่างรวดเร็ว โดยเฉพาะการมีสำเนาอยู่ในสถานที่เดียวกัน ซึ่งเข้าถึงได้รวดเร็วกว่าสำเนาที่อยู่นอกสถานที่

ในกรณีที่ใช้ระบบคลาวด์ องค์กรอาจเลือกปรับเปลี่ยนแนวทางดังกล่าว โดยเก็บหนึ่งสำเนาไว้ภายในองค์กร และอีกสองสำเนาไว้ในคลาวด์ โดยจัดเก็บไว้ในภูมิภาคที่ต่างกัน เพื่อเพิ่มความทนทานต่อภัยพิบัติ

1.4 การจัดการการสำรองข้อมูล (Backup Management)

การเปรียบเทียบระหว่างระบบสำรองข้อมูลแบบภายใน (เทป/ดิสก์) กับระบบคลาวด์สามารถพิจารณาได้จากแนวทางการบริหารจัดการดังนี้:

- ระบบเทปและดิสก์ต้องมีค่าใช้จ่ายเริ่มต้นสูงในด้านอุปกรณ์และสื่อจัดเก็บ
- ระบบคลาวด์อาจมีความเร็วในการกู้คืนข้อมูลกับคุณภาพของการเชื่อมต่ออินเทอร์เน็ต
- ระบบคลาวด์มีแนวโน้มรองรับอนาคตได้ดีกว่าระบบเทป
- ระบบคลาวด์สามารถเข้าถึงได้ทั่วโลก และไม่ต้องกังวลเรื่องการจัดเก็บนอกสถานที่
- ระบบเทปและดิสก์ต้องมีการควบคุมความปลอดภัยโดยตรงจากองค์กร
- ระบบเทปและดิสก์ให้สิทธิ์การควบคุมข้อมูลแก่เจ้าของระบบได้อย่างเต็มที่

1.5 กำหนดเวลาการสำรองข้อมูล (Backup Schedules)

การสำรองข้อมูลจะดำเนินการตามกำหนดเวลาที่แน่นอน อย่างไรก็ตาม กำหนดเวลาหนึ่งจะแตกต่างกันไปตามประเภทของข้อมูล เช่น ข้อมูลผู้ใช้มักมีการสำรองทุก 24 ชั่วโมง ในขณะที่ไฟล์การตั้งค่าของเซิร์ฟเวอร์ซึ่งมีเปลี่ยนแปลงบ่อย อาจไม่จำเป็นต้องสำรองข้อมูลบ่อยเท่ากัน

ระบบสำรองข้อมูลทั้งระดับเบื้องต้น เช่น Windows Server Backup ใน Windows Server 2022 และระบบระดับองค์กรที่จัดการทั้งการสำรองข้อมูลในองค์กรและบนคลาวด์ ต่างก็มีคุณสมบัติในการกำหนดเวลาการสำรองข้อมูลเป็นพื้นฐาน

การตั้งเวลาในการสำรองข้อมูลเป็นรูปแบบหนึ่งของการทำงานแบบอัตโนมัติ ซึ่งควรนำมาใช้ให้เกิดประโยชน์สูงสุด โดยระบบสำรองข้อมูลระดับองค์กรสามารถกำหนดเวลาการสำรองได้อย่างยืดหยุ่นตามแนวทางดังนี้:

- **ข้อมูลผู้ใช้:** ควรสำรองบ่อยพอที่จะสามารถกู้คืนเวอร์ชันของไฟล์ได้
- **ไฟล์การตั้งค่า:** ควรสำรองทุกครั้งที่มีการแก้ไขหรือเปลี่ยนแปลงการตั้งค่า

- การตั้งค่าระบบปฏิบัติการ (System-State): ควร 설정 เมื่อมีการอัปเกรดเวอร์ชันหรือเปลี่ยนแปลงการตั้งค่าครั้งใหญ่
- การตั้งค่าฐานข้อมูล: ไม่จำเป็นต้องสำรองบ่อย เนื่องจากโดยทั่วไปจะคงที่
- ข้อมูลฐานข้อมูล: อาจต้องสำรองหลายครั้งต่อวัน โดยเฉพาะฐานข้อมูลที่มีการใช้งานหนาแน่น

1.6 การเก็บรักษาข้อมูล (Data Retention)

นโยบายการเก็บรักษาข้อมูลจะกำหนดระยะเวลาที่ข้อมูลจะถูกเก็บรักษาไว้ ไม่ว่าจะเป็นข้อมูลในไฟล์ฐานข้อมูล หรือที่อื่น ๆ โดยเฉพาะข้อมูลส่วนบุคคล (Personally Identifiable Information - PII) หรือข้อมูลที่มีความอ่อนไหว

แนวปฏิบัติที่ดีในการจัดการนโยบายการเก็บรักษาข้อมูลสำรองบนคลาวด์ ได้แก่:

- ปฏิบัติตามข้อกำหนดของอุตสาหกรรมและกฎหมายที่เกี่ยวข้อง
- ให้ความสำคัญกับหลักเกณฑ์ด้านอธิปไตยข้อมูล (Data Sovereignty)
- จัดประเภทของข้อมูลเพื่อช่วยในการควบคุมตามนโยบาย
- ทำให้แน่ใจว่าข้อมูลและลักษณะข้อมูลสำรองเป็นไปโดยอัตโนมัติ เพื่อควบคุมต้นทุนการจัดเก็บ

ผู้ให้บริการคลาวด์ เช่น AWS มีระบบที่สามารถนัดกำหนดการจัดการนโยบายนี้เข้ากับเครื่องมือจัดการอายุข้อมูล เช่น AWS Data Lifecycle Manager ซึ่งช่วยให้สามารถควบคุมรอบเวลาและระยะเวลาเก็บรักษาข้อมูลได้อย่างเป็นระบบ

1.7 การจำลองข้อมูล (Data Replication)

การจำลองข้อมูลหมายถึงการคัดลอกข้อมูลไปยังเซิร์ฟเวอร์หรือสถานที่อื่น เพื่อรักษาความต่อเนื่องทางธุรกิจ รองรับการขยายระบบ และให้ข้อมูลอยู่ใกล้กับผู้ใช้งาน

ตัวอย่างของการจำลองข้อมูล:

- การจำลองข้อมูลระหว่างเซิร์ฟเวอร์เสมือน (VMs) ภายในไซต์เดียวกันเพื่อรับความพร้อมใช้งาน
- การจำลองข้อมูลจากเซิร์ฟเวอร์หนึ่งไปยังอีกเซิร์ฟเวอร์หนึ่งในไซต์เดียวกันหรือไซต์อื่น เพื่อให้มีข้อมูลสำรองที่อัปเดตอยู่เสมอ

ประเภทของการจำลอง:

- แบบ Asynchronous: เครื่องถูกข่ายจะได้รับการยืนยันว่าเขียนข้อมูลสำเร็จทันทีที่ข้อมูลถูกบันทึกไว้ในเซิร์ฟเวอร์หลัก และข้อมูลจะถูกจำลองไปยังเซิร์ฟเวอร์สำรองภายหลัง ซึ่งเสี่ยงต่อการสูญหายหากเซิร์ฟเวอร์หลักล้มเหลวก่อนการจำลองเสร็จ
- แบบ Synchronous: จะยืนยันผลสำเร็จให้เครื่องถูกข่ายก์ต่อเมื่อข้อมูลถูกบันทึกลงในทั้งสองเซิร์ฟเวอร์แล้ว ซึ่งมีความปลอดภัยมากกว่า

การจำลองข้อมูลควรเป็นส่วนหนึ่งของแผนฟื้นฟูจากภัยพิบัติขององค์กร

1.8 การทดสอบการกู้คืนข้อมูล (Recovery Testing)

การทดสอบระบบสำรองและกู้คืนข้อมูลควรทำอย่างสม่ำเสมอ เพื่อให้มั่นใจว่าข้อมูลถูกสำรองไว้อย่างถูกต้องและสามารถกู้คืนได้อย่างมีประสิทธิภาพ

รูปแบบการทดสอบ ได้แก่:

- การกู้คืนไฟล์รายบุคคล (เช่น จากการลบโดยไม่ตั้งใจ)
- การกู้คืนระบบ (เช่น ในกรณีเซิร์ฟเวอร์ล้มเหลว)
- การกู้คืนจากภัยพิบัติ (เช่น ศูนย์ข้อมูลไม่สามารถใช้งานได้)

หัวใจของการทดสอบคือการยืนยันว่าได้สำรองข้อมูลที่ถูกต้องไว้ โดยเฉพาะเมื่อติดตั้งเซิร์ฟเวอร์หรือบริการใหม่ ต้องกำหนดให้มีการสำรองข้อมูลเหล่านั้นทันทีที่เริ่มใช้งานจริง

ความสามารถในการกู้คืน (Recoverability) ขึ้นอยู่กับ:

- การสำรองที่สมบูรณ์ เข้าถึงได้ และไม่เสียหาย
- ความสามารถในการกู้คืนได้รวดเร็วและทันเวลาตามแผน
- ความสามารถในการกู้คืนแบบละเอียด (granular) หรือแบบครอบคลุมทั้งระบบ
- การปฏิบัติตามข้อกำหนดด้านกฎหมายและอุตสาหกรรม

ความสมบูรณ์ของข้อมูลสำรอง (Data Integrity) สามารถตรวจสอบได้โดยใช้การแฮช (Hashing) เพื่อสร้างค่าตรวจสอบ (Checksum) หากค่าตรวจสอบก่อนและหลังการสำรองตรงกัน แสดงว่าข้อมูลไม่มีการเปลี่ยนแปลงหรือเสียหาย

หน่วยที่ 2 แนวคิดเกี่ยวกับความพร้อมให้บริการของระบบ (Service Availability Concepts)

ความพร้อมให้บริการของระบบคลาวด์ (Cloud Service Availability) หมายถึง ระยะเวลาที่ผู้ใช้สามารถเข้าถึงและใช้งานบริการบนคลาวด์ได้อย่างมีประสิทธิภาพ ซึ่งมักถูกประเมินเป็นเปอร์เซ็นต์ เช่น “99.99%” (หรือที่เรียกว่า four nines) ซึ่งหมายถึงบริการจะหยุดชะงักได้ไม่เกินประมาณ 52.6 นาทีต่อปี

ความพร้อมให้บริการนี้มักเป็นข้อกำหนดที่ตกลงไว้ล่วงหน้าระหว่างผู้ให้บริการและผู้ใช้ ผ่านเอกสารข้อตกลงระดับการให้บริการ (Service Level Agreement: SLA) โดยผู้ให้บริการจะต้องรับประกันว่าระบบมีความสามารถในการให้บริการตามที่ระบุไว้ และสามารถฟื้นตัวจากความผิดพลาดได้อย่างมีประสิทธิภาพ

2.1 ความพร้อมใช้งานของทรัพยากร (Resource Availability)

ความพร้อมใช้งาน (Availability) และ ความซ้ำซ้อน (Redundancy) เป็นลักษณะสำคัญของระบบคลาวด์ที่ช่วยให้สามารถเข้าถึงบริการและข้อมูลจากทั่วโลกได้อย่างต่อเนื่อง ผู้ให้บริการคลาวด์มักแบ่งพื้นที่ให้บริการออกเป็น “ภูมิภาค” (Regions) และให้บริการในแต่ละภูมิภาคเหล่านั้นอย่างเป็นระบบ ตัวอย่างเช่น Microsoft Azure

ແປ່ງກຸມີກາຄອກເປັນ ອາເມຣິກາ ຍູໂຣປ ຕະວັນອອກລາງ ແອຟຣິກາ ແລະເອເຊີຍແບຊີຟຒກ ໂດຍໃນແຕ່ລະກຸມີກາຄຈະມີໂສນ
ຄວາມພ້ອມໃໝ່ງານ (Availability Zones) ຜຶ່ງປະກອບດ້ວຍສູນຍົ່ວ່ມລາຍແທ່ງ

ໂສນຄວາມພ້ອມໃໝ່ງານ (Availability Zones) ຄືອກລຸ່ມຂອງສູນຍົ່ວ່ມລາຍໃນກຸມີກາຄທີ່ມີການເຂື່ອມຕ່ອ
ຝ່ານຮະບບເຄື່ອງຂ່າຍຄວາມເຮົວສູງ ໂສນເຫັນນີ້ມີຮະບບໄຟຟ້າແລະເຄື່ອງຂ່າຍທີ່ແຍກເປັນອີສະຈາກກັນ ແຕ່ຍັງຄອນສາມາດ
ເຂື່ອມຕ່ອຮ່ວ່າງກັນດ້ວຍລິງກົງເຄື່ອງຂ່າຍທີ່ມີປະສິທິພາພູສູງ ແລະມີການຈັດວາງຕໍາແໜ່ງໃນພື້ນທີ່ທີ່ທ່າງກັນພອສມຄວາ
ເພື່ອປົ້ນກັນຄວາມເສີ່ງຈາກກັບຕິທາງຮຽມຈາຕີຫຼືຮູ້ອີກັນມູນໆ

ການໃໝ່ Availability Zones ປ່າຍໃຫ້ສາມາດຈັດວາງທຮພາກໄວ້ໄກລັກຝູ້ໃໝ່ມາກັ້ນ ເພື່ອເພີ່ມຄວາມທນທານ
ຂອງຮະບບແລະຮັບປະກັນວ່າບໍລິການແລະຂ້ອມລະສາມາດເຂົ້າສົ່ງໄດ້ຕົລອດເວລາ

2.2 ການໃໝ່ງານໜາຍຮະບບຄລາວດ້ວມກັນ (Multi-Cloud Tenancy)

ການໃໝ່ຮະບບຄລາວດ້ແບບຜສນ (Hybrid Cloud) ແລະຮະບບຄລາວດ້ຈາກໜາຍຜູ້ໃໝ່ບໍລິການ (Multi-Cloud)
ໜ່າຍໃຫ້ອົງຄຣສາມາດໃຫ້ບໍລິການທີ່ໜາກໜາຍແລະມີຄວາມໜ້າໜັນ (Redundant) ເພີ່ມຄວາມພ້ອມໃໝ່ງານແລະ
ຄວາມຢືດໜູ່ນີ້ໄດ້ມາກັ່ງຢືນ

- **Hybrid Cloud** ຄືອ ການຮັບຮັບຄລາວດ້ສ່ວນຕ້າ (Private Cloud) ແລະບໍລິການຄລາວດ້ສາຮາຣະນະ
(Public Cloud)
- **Multi-Cloud** ຄືອ ການໃໝ່ທຮພາກຈາກຜູ້ໃໝ່ບໍລິການຄລາວດ້ສາຮາຣະໜາຍຮາຍ ເຊັ່ນ AWS, Microsoft
Azure, ແລະ Google Cloud Platform (GCP)

ຕ້ວຍ່າງໜີ່ງຂອງ Hybrid Cloud ຄືອ **Cloud Bursting** ຜຶ່ງເປັນກະບວນການທີ່ຍ້າຍໂລດຈານໄປຢັງ
ຄລາວດ້ສາຮາຣະເມື່ອທຮພາກຂອງຮະບບກາຍໃນໄມ່ເພີ່ມພວ ເຊັ່ນໃນໜັງເວລາທີ່ມີການສັ່ງເຊື້ອຈຳນວນມາກໃນເວັບໄຊຕົວ
ໂຄມມີເມື່ອງ

Multi-Cloud ຍັງໜ່າຍໜີກາລື່ອງກາງຜູ້ໃໝ່ບໍລິການໄດ້ (Vendor Lock-in) ແລະເພີ່ມຄວາມຕ່ອນເນື່ອງ
ໃນການໃຫ້ບໍລິການ ທາກຜູ້ໃໝ່ບໍລິການໄດ້ຫຼຸດທໍາງນາມ ເວັບໄຊທີ່ກະຈາຍອູ່ໜາຍຜູ້ໃໝ່ບໍລິການຈະຍັງສາມາດໃຫ້ບໍລິການຕ່ອດໄໝ

ອ່າຍ່າງໄຮກ໌ຕາມ ການວາງຮະບບ Multi-Cloud ຕ້ອງມີການວາງແນນ ຕຽບສອບ ແລະບໍລິການຈັດກາຮອຍ່າງຮອບຄອບ
ໂດຍເພັະໃນເຮືອກການຈຳລອງຂ້ອມລຸ ການທຳນາຍໜ້າ ແລະຮະບບອັດໂນມັຕິໃນກັງກົດຈາກຄວາມເສີ່ງຫາຍ

2.3 ແນວດີດເຮືອກກາງກົດຈາກຮະບບແລະຄວາມຕ່ອນເນື່ອງທາງຮຽກຈົງ (Disaster Recovery and Business Continuity) ຂ້ອຕກລົງຮະດັບການໃຫ້ບໍລິການ (Service Level Agreements: SLA)

SLA ຄືອຂ້ອຕກລົງຮະຫວ່າງຜູ້ໃໝ່ບໍລິການກັບຜູ້ໃໝ່ບໍລິການທີ່ຮະບຸຮະດັບການໃຫ້ບໍລິການ ເຊັ່ນ ຄວາມພ້ອມໃໝ່ງານ
ຂອງເຊີ່ງົງໄວ້ຮັບສົ່ງ ທາກຜູ້ໃໝ່ບໍລິການໄມ່ສາມາດໃຫ້ບໍລິການຕາມ SLA ໄດ້ ຈາກຕ້ອງໜີ່ຫຼຸດທໍາງນາມ

ຕ້ວ້ັ້ວດສຳຄັນໃນ SLA ໄດ້ແກ່:

- **Recovery Time Objective (RTO):** ຮະຍະເວລາສູງສຸດທີ່ຮະບບສາມາດຫຼຸດທໍາງນາມໄດ້ໂດຍໄມ່ກະທບຕ່ອງຮຽກຈົງ
ອ່າຍ່າງຮູນແຮງ

- Recovery Point Objective (RPO): ปริมาณข้อมูลสูงสุดที่สามารถสูญเสียได้โดยไม่ส่งผลกระทบต่อธุรกิจ เช่น การสำรองข้อมูลทุก 1 ชั่วโมง
- Mean Time to Recovery (MTTR): เวลาที่เฉลี่ยในการฟื้นฟูระบบให้กลับมาใช้งานได้ตามปกติ
- Mean Time Between Failures (MTBF): เวลาที่เฉลี่ยระหว่างเหตุการณ์ความล้มเหลวของระบบ

2.4 ศูนย์คืนจากภัยพิบัติ (Disaster Recovery Sites)

เป็นสถานที่สำรองที่ใช้ในการดำเนินงานต่อเมื่อไซต์หลักไม่สามารถใช้งานได้ แบ่งออกเป็น 3 ประเภท หลัก:

ประเภท	ค่าใช้จ่าย	ความเร็วในการฟื้นฟู	ลักษณะ
Hot Site	สูง	ฟื้นฟูทันที	มีระบบ ฮาร์ดแวร์ ซอฟต์แวร์ และข้อมูลพร้อมใช้งาน
Warm Site	ปานกลาง	ฟื้นฟูได้ภายในเวลาไม่นาน	มีบางส่วนของอุปกรณ์และต้องติดตั้งเพิ่มเติม
Cold Site	ต่ำ	ฟื้นฟูช้า	มีเพียงพื้นที่เปล่า ต้องนำอุปกรณ์และข้อมูลมาติดตั้งเอง

ตาราง 8 ศูนย์คืนจากภัยพิบัติ

ศูนย์คืนผ่านระบบคลาวด์ (Cloud Sites)

ระบบคลาวด์กลายเป็นทางเลือกที่ได้รับความนิยมในการเป็นศูนย์คืน เพราะมีต้นทุนต่ำกว่า และสามารถขยายตัวได้ง่าย เช่น การจำลองเครื่อง VM ขึ้นระบบคลาวด์ล่วงหน้า

ข้อควรคำนึง:

- เลือกสถานที่ที่ห่างไกลทางภูมิศาสตร์จากไซต์หลัก เพื่อลดความเสี่ยงจากภัยพิบัติ
- พิจารณาภูมิภาคและข้อบังคับของประเทศต่าง ๆ กรณีเก็บข้อมูลข้ามเขตแดน

สรุปการจัดการการคืนจากภัยพิบัติและการดำเนินธุรกิจอย่างต่อเนื่อง (Managing Disaster Recovery and Business Continuity)

แนวทางที่สำคัญในการรักษาความพร้อมของบริการ แอปพลิเคชัน และข้อมูลสำหรับผู้ใช้งาน และพนักงาน ได้แก่ การวางระบบเพื่อเหลือเฟือ (Redundancy), การใช้งานโฉนดความพร้อมให้บริการ (Availability Zones) และการเลือกใช้โครงสร้างแบบมัลติคลาวด์ (Multi-Cloud)

การสำรองข้อมูลที่ถูกต้องและทันเวลาเป็นปัจจัยสำคัญในการปกป้องข้อมูลขององค์กร นอกจากนี้ การบริหารจัดการกระบวนการสำรองและคืนข้อมูลอย่างมีประสิทธิภาพก็มีความสำคัญต่อการรักษาความสอดคล้องตามข้อกำหนดด้านกฎหมายหรือข้อบังคับ (Compliance)

องค์กรจำนวนมากเลือกใช้สถานที่สำหรับภัยคุกคามระบบสำรองตามระดับความพร้อม 'ได้แก่'

- **Hot Site:** ศูนย์สำรองที่สามารถใช้งานได้ทันที
- **Warm Site:** ศูนย์สำรองที่มีการเตรียมระบบไว้บางส่วน ต้องใช้เวลาสักระยะในการใช้งาน
- **Cold Site:** ศูนย์สำรองที่ยังไม่มีระบบหรือข้อมูล ต้องติดตั้งและตั้งค่าทั้งหมดก่อนใช้งาน

บทที่ 13

การประยุกต์ใช้คลาวด์คอมพิวติ้งภายใต้กรอบกฎหมายและข้อบังคับของไทย (Applying Cloud Computing within the Thai Legal and Regulatory Framework)

บทนำของบทเรียน (Module Introduction)

บทเรียนนี้มีวัตถุประสงค์เพื่อเชื่อมโยงความรู้ทางเทคนิคด้านคลาวด์คอมพิวติ้งที่ได้ศึกษาในบทก่อนหน้า เข้ากับบริบททางกฎหมายและข้อบังคับเฉพาะของประเทศไทย การนำบริการคลาวด์มาใช้ในองค์กรไทย ไม่ได้เป็นเพียงการตัดสินใจทางเทคโนโลยี แต่ยังเป็นประเด็นด้านการกำกับดูแล ความเสี่ยง และการปฏิบัติตามกฎหมาย (Governance, Risk, and Compliance: GRC) ที่มีความสำคัญอย่างยิ่ง

การละเอียดข้อกำหนดทางกฎหมายอาจจำนำไปสู่ทลงโทษที่รุนแรง ทั้งทางแพ่ง อาญา และปกครอง ดังนั้น ผู้เชี่ยวชาญด้านคลาวด์จึงจำเป็นต้องมีความเข้าใจในกฎหมายดิจิทัลหลักของไทย เพื่อออกแบบ ปรับใช้ และบริหารจัดการสถาปัตยกรรมคลาวด์ได้อย่างมั่นคงปลอดภัยและสอดคล้องกับกฎหมาย

บทเรียนนี้จะสำรวจกฎหมายสำคัญ 3 ฉบับ ได้แก่ พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล, พระราชบัญญัติการรักษาความมั่นคงปลอดภัยไซเบอร์ และพระราชบัญญัติว่าด้วยการกระทำการทำความผิดเกี่ยวกับคอมพิวเตอร์ จากนั้นจะวิเคราะห์แนวทางการนำไปปฏิบัติจริงในสภาพแวดล้อมคลาวด์ พร้อมกรณีศึกษาจากองค์กรชั้นนำ และสรุปเป็นแนวทางปฏิบัติอย่างเป็นขั้นตอนสำหรับผู้ดูแลระบบคลาวด์

หน่วยที่ 1 ภาพรวมกฎหมายดิจิทัลของไทยที่เกี่ยวข้องกับระบบคลาวด์ (Overview of Thai Digital Laws Relevant to Cloud Computing)

1.1 พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562 (PDPA)

พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562 หรือ PDPA (Personal Data Protection Act) ถูกตราขึ้นเพื่อป้องกันการล่วงละเมิดสิทธิความเป็นส่วนตัวของข้อมูลส่วนบุคคล ซึ่งที่ความสำคัญขึ้นในยุคดิจิทัลที่การเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูลสามารถทำได้ง่ายและส่งผลกระทบในวงกว้าง กฎหมายฉบับนี้ได้รับอิทธิพลอย่างสูงจากระเบียบการคุ้มครองข้อมูลทั่วไปของสหภาพยุโรป หรือ GDPR (General Data Protection Regulation) และมีวัตถุประสงค์เพื่อสร้างมาตรฐานการคุ้มครองข้อมูลส่วนบุคคลของไทยให้เทียบเท่าสากล

นิยามศัพท์สำคัญ

- **ข้อมูลส่วนบุคคล (Personal Data):** หมายถึง ข้อมูลเกี่ยวกับบุคคลซึ่งทำให้สามารถระบุตัวบุคคลนั้นได้ไม่ว่าทางตรงหรือทางอ้อม แต่ไม่รวมถึงข้อมูลของผู้ถึงแก่กรรมโดยเฉพาะ ตัวอย่างเช่น ชื่อ-สกุล ที่อยู่ เลขประจำตัวประชาชน หมายเลขโทรศัพท์ อีเมล ข้อมูลสุขภาพ ข้อมูลชีวภาพ เป็นต้น
- **ผู้ควบคุมข้อมูลส่วนบุคคล (Data Controller):** หมายถึง บุคคลหรือนิติบุคคลซึ่งมีอำนาจหน้าที่ตัดสินใจเกี่ยวกับการเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูลส่วนบุคคล ในบริบทของคลาวด์คอมพิวติ้ง โดยทั่วไปแล้ว องค์กรผู้ใช้บริการคลาวด์ (Cloud Customer) จะมีสถานะเป็นผู้ควบคุมข้อมูลส่วนบุคคล
- **ผู้ประมวลผลข้อมูลส่วนบุคคล (Data Processor):** หมายถึง บุคคลหรือนิติบุคคลซึ่งดำเนินการเกี่ยวกับการเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูลส่วนบุคคลตามคำสั่งหรือในนามของผู้ควบคุมข้อมูลส่วนบุคคล ในบริบทของคลาวด์คอมพิวติ้ง โดยทั่วไปแล้ว ผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) จะมีสถานะเป็นผู้ประมวลผลข้อมูลส่วนบุคคล

ฐานกฎหมายในการประมวลผล (Lawful Basis for Processing): การเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูลส่วนบุคคลจะกระทำได้ก็ต่อเมื่อมีฐานทางกฎหมายรองรับ โดยฐานที่สำคัญที่สุดคือ ความยินยอม (Consent) ที่ขัดแจ้งจากเจ้าของข้อมูลส่วนบุคคล ซึ่งต้องเป็นการขอความยินยอมที่เข้าใจง่ายและไม่หลอกลวง นอกจากนี้ยังมีฐานกฎหมายอื่น ๆ ที่อนุญาตให้ประมวลผลข้อมูลได้โดยไม่ต้องขอความยินยอม เช่น เป็นการจำเป็นเพื่อปฏิบัติตามสัญญา, เป็นการจำเป็นเพื่อประโยชน์สาธารณะ, หรือเพื่อประโยชน์อันชอบด้วยกฎหมายของผู้ควบคุมข้อมูล เป็นต้น

สิทธิของเจ้าของข้อมูล (Data Subject Rights): กฎหมาย PDPA ได้กำหนดสิทธิของเจ้าของข้อมูลส่วนบุคคลไว้หลายประการ ซึ่งองค์กรต้องมีกระบวนการรองรับการใช้สิทธิเหล่านี้ สิทธิที่สำคัญประกอบด้วย สิทธิในการขอเข้าถึงและรับทราบข้อมูล, สิทธิในการคัดค้านการเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูล, สิทธิขอให้ลบหรือทำลายข้อมูล (Right to be Forgotten), และสิทธิในการเพิกถอนความยินยอม

บทลงโทษ: PDPA กำหนดบทลงโทษที่รุนแรงสำหรับผู้ที่ไม่ปฏิบัติตามกฎหมาย แบ่งออกเป็น 3 ประเภทหลัก ได้แก่:

- **ความรับผิดทางแพ่ง:** ต้องชดใช้ค่าสินไหมทดแทนตามความเสียหายที่เกิดขึ้นจริง และศาลอาจสั่งให้จ่ายค่าสินไหมทดแทนเพื่อการลงโทษเพิ่มเติมสูงสุดไม่เกิน 2 เท่าของค่าเสียหายจริง
- **โทษทางอาญา:** สำหรับการกระทำการใดๆ ก็ตามที่กฎหมายกำหนดเป็นอาชญากรรม อาจมีโทษจำคุกสูงสุดไม่เกิน 1 ปี หรือปรับไม่เกิน 1 ล้านบาท หรือทั้งจำทั้งปรับ

- **โทษทางปกครอง:** คณะกรรมการคุ้มครองข้อมูลส่วนบุคคลมีอำนาจสั่งปรับสูงสุดไม่เกิน 5 ล้านบาท ขึ้นอยู่กับความรุนแรงของการกระทำความผิด

1.2 พระราชบัญญัติการรักษาความมั่นคงปลอดภัยไซเบอร์ พ.ศ. 2562 (Cybersecurity Act)

หลักการและเหตุผล: พ.ร.บ. การรักษาความมั่นคงปลอดภัยไซเบอร์ มีวัตถุประสงค์เพื่อจัดตั้งกลไกในการป้องกัน รับมือ และลดความเสี่ยงจากภัยคุกคามทางไซเบอร์ ซึ่งอาจส่งผลกระทบต่อความมั่นคงของรัฐ ความปลอดภัยสาธารณะ และความสงบเรียบร้อยของประชาชน โดยมุ่งเน้นการปกป้องโครงสร้างพื้นฐานสำคัญทางสารสนเทศของประเทศไทยเป็นพิเศษ

โครงสร้างพื้นฐานสำคัญทางสารสนเทศ (Critical Information Infrastructure - CII): กฎหมายได้ให้อำนาจคณะกรรมการการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ (กมช.) ในการประกาศกำหนดให้หน่วยงานที่มีภารกิจหรือให้บริการใน 8 ด้านหลัก เป็นหน่วยงาน CII ซึ่งรวมถึงด้านความมั่นคงของรัฐ, บริการภาครัฐที่สำคัญ, การเงินการธนาคาร, เทคโนโลยีสารสนเทศและโทรคมนาคม, การขนส่งและโลจิสติกส์, พลังงานและสาธารณูปโภค, สาธารณสุข และด้านอื่น ๆ องค์กรที่ถูกจัดเป็น CII จะมีหน้าที่และความรับผิดชอบตามกฎหมายนี้อย่างเคร่งครัด

หน้าที่ของหน่วยงาน CII:

- **จัดทำมาตรฐานและประเมินความเสี่ยง:** ต้องจัดทำประมวลแนวทางปฏิบัติและกรอบมาตรฐานด้านความปลอดภัยไซเบอร์ของตนเอง และต้องจัดให้มีการประเมินความเสี่ยงอย่างน้อยปีละหนึ่งครั้ง
- **จัดทำแผนรับมือเหตุการณ์:** ต้องมีแผนการรับมือภัยคุกคามทางไซเบอร์ (Incident Response Plan) ที่ชัดเจนและสามารถนำไปปฏิบัติได้
- **รายงานเหตุคุกคาม:** เมื่อเกิดเหตุภัยคุกคามทางไซเบอร์ที่มีนัยสำคัญ จะต้องรายงานต่อสำนักงานคณะกรรมการการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ (สกมช. หรือ NCSA) และหน่วยงานกำกับดูแลที่เกี่ยวข้องโดยเร็ว

ระดับของภัยคุกคาม: กฎหมายได้จำแนกวัยคุกคามทางไซเบอร์ออกเป็น 3 ระดับ คือ ภัยคุกคามระดับไม่ร้ายแรง, ระดับร้ายแรง, และระดับวิกฤต ซึ่งแต่ละระดับจะมีกลไกการรับมือและให้อำนาจแก่เจ้าหน้าที่รัฐในการเข้าดำเนินการแตกต่างกันไป

1.3 พระราชบัญญัติว่าด้วยการกระทำความผิดเกี่ยวกับคอมพิวเตอร์ พ.ศ. 2560 (ฉบับแก้ไข) (Computer Crime Act)

ภาพรวม: พ.ร.บ. คอมพิวเตอร์ฯ เป็นกฎหมายที่กำหนดฐานความผิดและบทลงโทษทางอาญาโดยตรง สำหรับการกระทำที่เป็นการใช้ระบบคอมพิวเตอร์หรือเครือข่ายอินเทอร์เน็ตในทางที่ผิดกฎหมาย

ความผิดที่เกี่ยวข้องกับระบบคลาวด์:

- การเข้าถึงโดยมิชอบ (มาตรา 5, 7): การเจาะเข้าสู่ระบบคลาวด์ คอนโซลผู้ดูแลระบบ หรือเข้าถึงข้อมูลใน Object Storage โดยไม่ได้รับอนุญาต ถือเป็นความผิด
- การตั้งรับข้อมูล (มาตรา 8): การใช้เครื่องมือตักจับข้อมูล (Packet Sniffing) เพื่อดูข้อมูลที่ส่งผ่านระหว่างผู้ใช้งานและบริการบนคลาวด์โดยมิชอบ
- การรบกวนหรือทำลายข้อมูล/ระบบ (มาตรา 9, 10): การโจมตีด้วย Ransomware ที่เข้ารหัสข้อมูลบน Cloud Storage หรือการโจมตีแบบ DDoS เพื่อให้เว็บไซต์หรือแอปพลิเคชันที่iosต์บนคลาวด์ไม่สามารถให้บริการได้
- การส่งสแปม (มาตรา 11): การใช้ทรัพยากรคลาวด์ เช่น VM หรือบริการอีเมล เพื่อส่งอีเมลขยะหรือข้อความที่สร้างความเดือดร้อนรำคาญจำนวนมาก
- การนำเข้าข้อมูลเท็จ/ลามก (มาตรา 14): การใช้บริการคลาวด์เพื่อiosต์เว็บไซต์ที่มีเนื้อหาหลอกลวง ข้อมูลอันเป็นเท็จที่สร้างความเสียหายต่อประชาชน หรือเนื้อหาลามกอนาจาร

ความรับผิดของผู้ให้บริการ (มาตรา 15): มาตรานี้มีความสำคัญอย่างยิ่งต่อผู้ให้บริการคลาวด์และองค์กรที่ให้บริการบนคลาวด์ โดยกำหนดว่าหากผู้ให้บริการ "ให้ความร่วมมือ ยินยอม หรือรู้เห็นเป็นใจ" ให้มีการกระทำความผิดตามมาตรา 14 เกิดขึ้นในระบบของตน ผู้ให้บริการนั้นจะต้องรับโทษเช่นเดียวกับผู้กระทำความผิด

การทำงานที่เชื่อมโยงกันของกฎหมายทั้งสามฉบับนี้สร้างกรอบการกำกับดูแลที่ครอบคลุมสำหรับการดำเนินงานบนระบบคลาวด์ในประเทศไทย เหตุการณ์ด้านความปลอดภัยเพียงครั้งเดียวสามารถก่อให้เกิดความรับผิดภายนอกกฎหมายทั้งสามฉบับพร้อมกันได้ ตัวอย่างเช่น หากแฮกเกอร์เจาะระบบคลาวด์ของธนาคาร (ซึ่งเป็น CII) และพยายามข้อมูลบัตรเครดิตของลูกค้าไป จะเกิดผลกระทบทางกฎหมายดังนี้:

1. ภายใต้ พ.ร.บ. คอมพิวเตอร์ฯ การกระทำดังกล่าวถือเป็นการ "เข้าถึงระบบและข้อมูลโดยมิชอบ" (มาตรา 5, 7) และ "การทำให้ข้อมูลเสียหาย" (มาตรา 9) ซึ่งเป็นความผิดทางอาญาโดยตรง
2. ภายใต้ พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคลฯ (PDPA) ข้อมูลบัตรเครดิตคือ "ข้อมูลส่วนบุคคล" ธนาคารในฐานะ "ผู้ควบคุมข้อมูล" มีหน้าที่ "จัดให้มีมาตรการรักษาความมั่นคงปลอดภัยที่เหมาะสม" การที่ข้อมูลรั่วไหลแสดงถึงความล้มเหลวในการปฏิบัติตามหน้าที่นี้ ซึ่งนำไปสู่โทษปรับทางปกครองและการชดใช้ค่าเสียหายทางแพ่ง
3. ภายใต้ พ.ร.บ. ความมั่นคงปลอดภัยไซเบอร์ฯ เนื่องจากธนาคารเป็น CII การถูกเจาะระบบจึงถือเป็น "ภัยคุกคามทางไซเบอร์" ที่ต้องรายงานต่อ สมช. ตามกฎหมาย หากธนาคารล้มเหลวในการประเมินความเสี่ยงหรือไม่มีแผนรับมือที่เพียงพอ ก็อาจมีความผิดเพิ่มเติมภายใต้ พ.ร.บ. นี้ได้

ดังนั้น การบริหารจัดการความเสี่ยงบนคลาวด์จึงต้องพิจารณากฎหมายเหล่านี้แบบองค์รวม ไม่สามารถแยกจากกันได้

หน่วยที่ 2 การปฏิบัติตาม PDPA ในสภาพแวดล้อมคลาวด์ (PDPA Compliance in a Cloud Environment)

2.1 การกำหนดบทบาทและความรับผิดชอบ: ผู้ใช้บริการคลาวด์และผู้ให้บริการ

การทำความเข้าใจบทบาทและความรับผิดชอบตามกฎหมาย PDPA เป็นขั้นตอนแรกที่สำคัญที่สุดในการใช้งานคลาวด์อย่างถูกต้อง

กฎหมาย PDPA กำหนดบทบาทที่ชัดเจนซึ่งสามารถนำมาจับคู่กับผู้เกี่ยวข้องในระบบคลาวด์ได้โดยตรง องค์กรผู้ใช้บริการคลาวด์ (Cloud Customer) มีสถานะเป็น ผู้ควบคุมข้อมูลส่วนบุคคล (Data Controller) เนื่องจากเป็นผู้ "ตัดสินใจ" ว่าจะเก็บข้อมูลส่วนบุคคลอะไร, เพื่อวัตถุประสงค์ใด, และจะนำไปประมวลผลอย่างไร ในทางกลับกัน ผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP) เช่น AWS, Azure, หรือ GCP มีสถานะเป็น

ผู้ประมวลผลข้อมูลส่วนบุคคล (Data Processor) เนื่องจากเป็นผู้ดำเนินการให้ร่วมกับข้อมูลตาม "คำสั่ง" ของลูกค้าเท่านั้น

สิ่งสำคัญที่ต้องทราบก็คือ ผู้ควบคุมข้อมูล (ลูกค้า) เป็นผู้ที่ต้องรับผิดชอบโดยตรงต่อเจ้าของข้อมูล และต้องกฎหมายในการปฏิบัติตาม PDPA ทั้งหมด องค์กรไม่สามารถโอนความรับผิดชอบทางกฎหมายนี้ไปยังผู้ประมวลผล (CSP) ได้ แม้ว่าจะใช้บริการที่มีการจัดการอย่างเต็มรูปแบบก็ตาม

โมเดลความรับผิดชอบร่วม (Shared Responsibility Model) ในมุมมองของ PDPA: แนวคิดเรื่องโมเดลความรับผิดชอบร่วมที่ได้ศึกษาในบทที่ 1 สามารถนำมากย้ายความในบริบทของกฎหมาย PDPA ได้ดังนี้

- IaaS (Infrastructure as a Service):** ลูกค้า (Controller) มีความรับผิดชอบในขอบเขตกว้างที่สุด ตั้งแต่การรักษาความปลอดภัยของระบบปฏิบัติการ (OS), การติดตั้งแอปพลิเคชัน, การกำหนดค่าเครือข่ายและความปลอดภัย (Security Groups, IAM), ไปจนถึงการจัดการข้อมูลเองทั้งหมด ในขณะที่ CSP (Processor) จะรับผิดชอบความปลอดภัยของโครงสร้างพื้นฐานทางกายภาพ เช่น ศูนย์ข้อมูล และระบบเครือข่ายหลัก
- PaaS (Platform as a Service):** ความรับผิดชอบของ CSP (Processor) เพิ่มขึ้น โดยจะครอบคลุมถึงการดูแลรักษาระบบปฏิบัติการและ Middleware แต่ลูกค้า (Controller) ยังคงต้องรับผิดชอบในการพัฒนาแอปพลิเคชันให้ปลอดภัย, การจัดการข้อมูล, และการกำหนดสิทธิ์การเข้าถึงของผู้ใช้

- **SaaS (Software as a Service):** CSP (Processor) รับผิดชอบโครงสร้างพื้นฐานเกือบทั้งหมดของสเต็กเทคโนโลยี อย่างไรก็ตาม ลูกค้า (Controller) ยังคงมีความรับผิดชอบที่สำคัญอย่างยิ่งในการจัดการข้อมูลที่นำเข้าสู่ระบบ, การกำหนดค่าสิทธิ์การเข้าถึงของผู้ใช้งานภายในแอปพลิเคชัน, และการปฏิบัติตามกฎหมายในการเก็บรวบรวมข้อมูลนั้น ๆ

2.2 การเลือกผู้ให้บริการคลาวด์ (CSP) และความสำคัญของข้อตกลงการประมวลผลข้อมูล (DPA)

ก่อนที่จะตัดสินใจเลือกใช้บริการจาก CSP รายใด องค์กรในฐานะผู้ควบคุมข้อมูลมีหน้าที่ต้องตรวจสอบสถานะเพื่อให้แน่ใจว่า CSP รายนั้นมีมาตรการรักษาความมั่นคงปลอดภัยที่เหมาะสมตามที่ PDPA กำหนด ซึ่งรวมถึงการตรวจสอบว่า CSP มีบริรองมาตรฐานสากลที่นำไปใช้หรือไม่ เช่น ISO/IEC 27001, SOC 2, หรือมาตรฐานอื่น ๆ ที่เกี่ยวข้อง

DPA คือสัญญาทางกฎหมายที่ต้องจัดทำขึ้นระหว่างผู้ควบคุมข้อมูล (ลูกค้า) และผู้ประมวลผลข้อมูล (CSP) เพื่อกำหนดขอบเขตและเงื่อนไขในการประมวลผลข้อมูลส่วนบุคคล ซึ่งเป็นข้อกำหนดที่จำเป็นตามกฎหมาย

สาระสำคัญที่ควรมีใน DPA

- **ขอบเขตการประมวลผล:** ระบุชัดเจนว่า CSP จะประมวลผลข้อมูลส่วนบุคคลตามคำสั่งที่เป็นลายลักษณ์อักษรของลูกค้าเท่านั้น
- **มาตรการความปลอดภัย:** กำหนดให้ CSP ต้องมีมาตรการรักษาความปลอดภัยทั้งทางเทคนิคและทางองค์กรที่เหมาะสมเพื่อป้องกันข้อมูล
- **การแจ้งเหตุละเมิด:** CSP มีหน้าที่ต้องแจ้งให้ลูกค้าทราบโดยไม่ชักช้าเมื่อเกิดเหตุการณ์ละเมิดข้อมูลส่วนบุคคลขึ้น
- **ผู้ประมวลผลช่วง (Sub-processors):** กำหนดเงื่อนไขและข้อจำกัดในการที่ CSP จะใช้บริการจากผู้ประมวลผลรายอื่นต่อ (เช่น ผู้ให้บริการเครือข่ายหรือศูนย์ข้อมูลย่อย) และต้องได้รับอนุญาตจากลูกค้าก่อน
- **การให้ความร่วมมือ:** CSP ต้องให้ความร่วมมือกับลูกค้าในการปฏิบัติตามคำขอใช้สิทธิ์ของเจ้าของข้อมูล และให้ความร่วมมือในการตรวจสอบ (Audit) ตามที่ตกลงกัน

2.3 ความท้าทายด้าน Data Sovereignty และการโอนข้อมูลข้ามพรมแดน

หลักการตามมาตรา 28 แห่ง PDPA กฎหมายกำหนดไว้อย่างชัดเจนว่า การส่งหรือโอนข้อมูลส่วนบุคคลไปยังต่างประเทศจะสามารถทำได้ก็ต่อเมื่อประเทศปลายทางหรือองค์กรระหว่างประเทศที่รับข้อมูลนั้น มี "มาตรฐานการคุ้มครองข้อมูลส่วนบุคคลที่เพียงพอ"

ผลกระทบต่อการใช้คลาวด์ การใช้บริการคลาวด์จากผู้ให้บริการระดับโลก (Hyperscalers) เช่น AWS, Azure, หรือ GCP ซึ่งมีศูนย์ข้อมูล (Data Center หรือ Region) ตั้งอยู่ในประเทศไทย จะถือเป็นการ "โอนข้อมูลส่วนบุคคลไปยังต่างประเทศ" โดยทันทีที่ข้อมูลถูกอัปโหลดขึ้นสู่ระบบ ดังนั้น องค์กรจึงต้องพิจารณาประเด็นนี้อย่างรอบคอบ

แนวทางปฏิบัติเพื่อการโอนข้อมูลอย่างถูกกฎหมาย:

- **การเลือก Region:** แนวทางที่ปลดภัยและตรงไปตรงมาที่สุดคือการเลือกใช้ Region ของผู้ให้บริการคลาวด์ที่ตั้งอยู่ในประเทศไทย (หากมีให้บริการ) หรือเลือก Region ที่ตั้งอยู่ในประเทศไทย คณะกรรมการคุ้มครองข้อมูลส่วนบุคคลได้ประกาศรับรองแล้วว่ามีมาตรฐานการคุ้มครองข้อมูลที่เพียงพอ
- **การอาศัยข้อยกเว้น:** ในกรณีที่ประเทศไทยยังไม่ได้รับการรับรองว่ามีมาตรฐานที่เพียงพอ องค์กรยังสามารถโอนข้อมูลได้หากเข้าข้อยกเว้นตามกฎหมาย ซึ่งได้แก่:
 - ได้รับ ความยินยอมโดยชัดแจ้ง จากเจ้าของข้อมูล หลังจากที่ได้แจ้งให้ทราบถึงความเสี่ยงของประเทศไทยที่ไม่มีมาตรฐานเพียงพอแล้ว
 - เป็นการจำเป็นเพื่อ ปฏิบัติตามสัญญา ที่เจ้าของข้อมูลเป็นคู่สัญญา
 - เป็นการกระทำตามสัญญาระหว่างผู้ควบคุมข้อมูลกับบุคคลอื่น เพื่อประโยชน์ของเจ้าของข้อมูล
 - เพื่อป้องกันหรือระงับอันตรายต่อชีวิต ร่างกาย หรือสุขภาพ
 - เป็นการจำเป็นเพื่อการดำเนินการกิจเพื่อ ประโยชน์สาธารณะที่สำคัญ
- **มาตรการอื่น ๆ:** องค์กรสามารถใช้ข้อบังคับเกี่ยวกับการคุ้มครองข้อมูลส่วนบุคคล (Binding Corporate Rules - BCRs) สำหรับการโอนข้อมูลภายใต้เครือบริษัท หรือจัดให้มีมาตรการคุ้มครองที่เหมาะสมอื่น ๆ ตามที่กฎหมายลูกกำหนด

2.4 แนวทางเทคนิคเพื่อการปฏิบัติตาม PDPA บนคลาวด์

นอกเหนือจากข้อกำหนดทางกฎหมายและสัญญาแล้ว การปฏิบัติตาม PDPA ยังต้องอาศัยการนำเทคโนโลยีและมาตรการควบคุมทางเทคนิคมาปรับใช้อย่างเหมาะสมบนแพลตฟอร์มคลาวด์

- **การค้นหาและจัดประเภทข้อมูล (Data Discovery and Classification):** ขั้นตอนพื้นฐานที่สุดคือ องค์กรต้องทราบว่าข้อมูลส่วนบุคคลถูกจัดเก็บไว้ที่ใดในระบบคลาวด์ และข้อมูลนั้นมีความอ่อนไหวระดับใด บริการคลาวด์มักมีเครื่องมือที่ช่วยในกระบวนการนี้ เช่น Amazon Macie หรือ Azure Information Protection ซึ่งสามารถสแกนหาและจัดประเภทข้อมูลที่ละเอียดอ่อนได้โดยอัตโนมัติ

- การบริหารจัดการตัวตนและการเข้าถึง (Identity and Access Management - IAM): ใช้บริการ IAM ของผู้ให้บริการคลาวด์เพื่อบังคับใช้ หลักการให้สิทธิ์น้อยที่สุด (Principle of Least Privilege) และ การควบคุมการเข้าถึงตามบทบาท (Role-Based Access Control - RBAC) เพื่อให้แน่ใจว่าพนักงานหรือระบบจะสามารถเข้าถึงข้อมูลส่วนบุคคลได้เท่าที่จำเป็นต่อการปฏิบัติงานเท่านั้น
- การเข้ารหัสข้อมูล (Encryption): ข้อมูลส่วนบุคคลจะต้องถูกเข้ารหัสเพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต ทั้งในขณะที่ข้อมูลถูกจัดเก็บ (Data at Rest) บน Cloud Storage และในขณะที่ข้อมูลกำลังถูกส่งผ่านเครือข่าย (Data in Transit)
- การป้องกันข้อมูลรั่วไหล (Data Loss Prevention - DLP): ใช้เครื่องมือ DLP ที่มีในระบบคลาวด์ เพื่อสร้างนโยบายในการตรวจสอบและป้องกันการส่งข้อมูลส่วนบุคคลที่ละเมียดละอ่อนออกจากสภาพแวดล้อมคลาวด์ที่ควบคุมไว้โดยไม่ได้รับอนุญาต
- การบันทึกและตรวจสอบ (Logging and Monitoring): เปิดใช้งานบริการบันทึกกิจกรรมอย่างละเอียด เช่น AWS CloudTrail หรือ Azure Monitor เพื่อติดตามการดำเนินการทั้งหมดที่เกิดขึ้นกับข้อมูลส่วนบุคคล Log เหล่านี้มีความสำคัญอย่างยิ่งสำหรับการตรวจสอบ (Audit) และการสืบสวนเมื่อเกิดเหตุการณ์ละเมิดข้อมูล

หน้าที่ความรับผิดชอบตาม PDPA	IaaS (Infrastructure as a Service)	PaaS (Platform as a Service)	SaaS (Software as a Service)
การขอความยินยอมจากเจ้าของข้อมูล	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)
การจัดการคำขอใช้สิทธิ์ของเจ้าของข้อมูล	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)
ความปลอดภัยของข้อมูลและการกำหนดสิทธิ์ในแอปพลิเคชัน	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)
ความปลอดภัยของระบบปฏิบัติการ (OS) และการจัดการแพตช์	ผู้ใช้บริการ (Controller)	ผู้ให้บริการ (Processor)	ผู้ให้บริการ (Processor)

หน้าที่ความรับผิดชอบตาม PDPA	IaaS (Infrastructure as a Service)	PaaS (Platform as a Service)	SaaS (Software as a Service)
การกำหนดค่าความปลอดภัย เครือข่าย (เช่น Security Groups)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ให้บริการ (Processor)
การเข้ารหัสข้อมูลขณะส่งผ่าน (In-Transit)	ร่วมกัน (Shared)	ร่วมกัน (Shared)	ผู้ให้บริการ (Processor)
การเข้ารหัสข้อมูลขณะจัดเก็บ (At-Rest)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ให้บริการ (Processor)
ความปลอดภัยของโครงสร้างพื้นฐานกายภาพ	ผู้ให้บริการ (Processor)	ผู้ให้บริการ (Processor)	ผู้ให้บริการ (Processor)
การแจ้งเหตุล้มเมิดข้อมูลส่วนบุคคล (ต่อเจ้าของข้อมูล)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)	ผู้ใช้บริการ (Controller)
การแจ้งเหตุล้มเมิดข้อมูลส่วนบุคคล (ต่อ Controller)	ผู้ให้บริการ (Processor)	ผู้ให้บริการ (Processor)	ผู้ให้บริการ (Processor)

ตารางที่ 12.1: ตารางเปรียบเทียบความรับผิดชอบตาม PDPA ระหว่างผู้ใช้บริการ (Data Controller) และผู้ให้บริการคลาวด์ (Data Processor) ในรูปแบบ IaaS, PaaS, และ SaaS

หน่วยที่ 3 การดำเนินการตาม พ.ร.บ. ความมั่นคงปลอดภัยไซเบอร์ (Implementing the Cybersecurity Act)

3.1 การประเมินความเสี่ยงทางไซเบอร์และการจัดทำแผนรับมือเหตุการณ์สำหรับบริการคลาวด์

แม้ว่าองค์กรที่เข้าข่ายเป็น CII จะใช้บริการจากผู้ให้บริการคลาวด์ภายนอก แต่ความรับผิดชอบตาม พ.ร.บ. ไซเบอร์ฯ ยังคงอยู่กับองค์กรนั้น ๆ โดยองค์กรมีหน้าที่ต้องประเมินความเสี่ยงทางไซเบอร์และจัดทำแผนรับมือเหตุการณ์ที่ครอบคลุมบริการคลาวด์ที่ใช้งานอยู่

การประเมินความเสี่ยงต้องพิจารณาแบบองค์รวม โดยครอบคลุมทั้งส่วนที่อยู่ในความรับผิดชอบของผู้ให้บริการคลาวด์ (เช่น ความปลอดภัยทางกายภาพของศูนย์ข้อมูล, ความมั่นคงของระบบ Hypervisor) และส่วนที่องค์กรต้องรับผิดชอบเองอย่างเต็มที่ (เช่น การกำหนดค่า Security Groups และ IAM Policies ที่หละหลวม, ช่องโหว่ในแอปพลิเคชันที่พัฒนาขึ้นเอง, หรือการจัดการข้อมูลประจำตัวที่ไม่รัดกุม)

แผนรับมือเหตุการณ์ต้องระบุขั้นตอนการดำเนินการที่ชัดเจนเมื่อเกิดภัยคุกคามทางไซเบอร์บนสภาพแวดล้อมคลาวด์ ซึ่งควรประกอบด้วย

- การตรวจจับและวิเคราะห์ (Detection & Analysis): วิธีการรับรู้ถึงเหตุการณ์ผ่านระบบ Monitoring และ Log
- การจำกัดความเสียหาย (Containment): ขั้นตอนทางเทคนิคในการจำกัดผลกระทบ เช่น การแยกส่วนระบบที่ถูกโจมตี (Isolating compromised instances) หรือการปรับแก้กฎ Firewall ทันที
- การกำจัดภัยคุกคาม (Eradication): การกำจัดต้นตอของภัยคุกคาม เช่น การลบมัลแวร์ หรือการปิดบัญชีผู้ใช้ที่ถูกขโมย
- การกู้คืนระบบ (Recovery): กระบวนการกู้คืนบริการให้กลับสู่ภาวะปกติ เช่น การกู้คืนระบบจากข้อมูลสำรอง (Backup/Snapshot)
- กิจกรรมหลังเกิดเหตุ (Post-Incident Activity): การวิเคราะห์สาเหตุที่แท้จริง (Root Cause Analysis) และการปรับปรุงมาตรการป้องกัน
- การรายงาน: กระบวนการและผู้รับผิดชอบในการแจ้งเหตุไปยัง สมช. (NCSA) และหน่วยงานกำกับดูแลอื่น ๆ ตามที่กฎหมายกำหนด

3.2 การปฏิบัติตามมาตรฐานความปลอดภัยไซเบอร์บนระบบคลาวด์ของ สมช. (NCSA)

การเกิดขึ้นของมาตรฐานความปลอดภัยไซเบอร์บนระบบคลาวด์โดยเฉพาะจาก สมช. สะท้อนให้เห็นถึงการยกระดับการกำกับดูแลของไทย จากการออกกฎหมายที่เป็นหลักการกว้างๆ ไปสู่การกำหนดมาตรฐานทางเทคนิคที่เฉพาะเจาะจงและสามารถตรวจสอบได้ (Auditable) พ.ร.บ. ไซเบอร์ฯ ปี 2562 ได้กำหนด "หน้าที่" ให้หน่วยงาน CII ต้องประเมินความเสี่ยงและมีมาตรการป้องกัน แต่ไม่ได้ระบุ "วิธีการ" ทางเทคนิคอย่างละเอียด ซึ่งอาจสร้างความคลุมเครือให้กับองค์กร การประกาศมาตรฐานใหม่ของ สมช. จึงเข้ามาปิดช่องว่างนี้โดยตรง โดยให้กรอบการทำงาน (Framework) และรายการตรวจสอบ (Checklist) ที่ชัดเจน ซึ่งอ้างอิงจากมาตรฐานสากล สำหรับผู้เชี่ยวชาญด้านคลาวด์ นี้หมายความว่าการปฏิบัติตามกฎหมายไม่ใช่เรื่องของการตีความอีกต่อไป แต่เป็นเรื่องของการนำ "ข้อกำหนดทางเทคนิค" ที่ระบุไว้ไปปฏิบัติจริง และเตรียมพร้อมรับการตรวจสอบจากหน่วยงานรัฐ

สมช. ได้ประกาศ "ร่างมาตรฐานความปลอดภัยไซเบอร์บนระบบคลาวด์" ซึ่งจะเป็นข้อบังคับสำหรับหน่วยงานของรัฐ หน่วยงานควบคุมหรือกำกับดูแล และหน่วยงาน CII ที่ใช้บริการคลาวด์

โครงสร้างมาตรฐาน: มาตรฐานนี้แบ่งออกเป็น 2 ส่วนหลัก

- การกำกับดูแลด้านความมั่นคงปลอดภัยระบบคลาวด์ (Cloud Security Governance): ครอบคลุมเรื่องนโยบาย, การบริหารจัดการความเสี่ยง, การปฏิบัติตามข้อกำหนด, และการจัดการความสัมพันธ์กับผู้ให้บริการภายนอก
- การปฏิบัติและการรักษาความมั่นคงปลอดภัยโครงสร้างพื้นฐานระบบคลาวด์ (Cloud Infrastructure and Operations): ครอบคลุมการควบคุมด้านเทคนิคที่สำคัญ เช่น การจัดการทรัพย์สิน (Asset Management), การควบคุมการเข้าถึง (Access Control), การเข้ารหัส (Cryptography), การรักษาความปลอดภัยในการปฏิบัติการ (Operations Security), และการจัดการเหตุการณ์ทางสารสนเทศ

มาตรฐานกำหนดให้องค์กรต้องประเมินระดับผลกระทบของข้อมูลและระบบสารสนเทศของตน (ต่ำ, กลาง, สูง) และปฏิบัติตามข้อกำหนดขึ้นต่ำตามระดับผลกระทบนั้นๆ ประเด็นสำคัญคือ **ข้อมูลส่วนบุคคล** จะต้องถูกจัดระดับผลกระทบด้านความลับ (Confidentiality) ไม่ต่ำกว่าระดับกลางเป็นอย่างน้อย ซึ่งหมายความว่าองค์กรที่ประมวลผลข้อมูลส่วนบุคคลบนคลาวด์จะต้องปฏิบัติตามมาตรการควบคุมในระดับกลางเป็นอย่างต่ำ

องค์กรที่อยู่ในขอบเขตของมาตรฐานนี้ จะต้องจัดส่งรายงานสรุปผลการดำเนินการตามมาตรฐานให้ สมกช. ทราบภายใน 30 วันหลังจากดำเนินการเสร็จสิ้น

หน่วยที่ 4 กรณีศึกษาและการดำเนินการอย่างเป็นขั้นตอน (Case Studies and Step-by-Step Implementation)

4.1 กรณีศึกษา: การย้ายระบบสู่คลาวด์ของสถาบันการเงินในประเทศไทย

สถาบันการเงินเป็นหนึ่งในกลุ่มอุตสาหกรรมที่เป็น CII ซึ่งอยู่ภายใต้การกำกับดูแลที่เข้มงวดที่สุด และกำลังเผชิญกับความท้าทายในการปรับเปลี่ยนองค์กรสู่ดิจิทัล (Digital Transformation) อย่างหลีกเลี่ยงไม่ได้ กรณีศึกษาของธนาคารชั้นนำในประเทศไทยให้บทเรียนที่สำคัญเกี่ยวกับการนำคลาวด์มาใช้ภายใต้ข้อจำกัดเหล่านี้

ตัวอย่างที่ 1: ธนาคารไทยพาณิชย์ (SCB) กับการย้ายสู่ Microsoft Azure

- แรงผลักดัน (Drivers):** ความต้องการปลดล็อกศักยภาพของข้อมูลที่มีอยู่มหาศาล เพื่อสร้างประสบการณ์ที่ดีขึ้นให้แก่ลูกค้า และเพิ่มประสิทธิภาพการดำเนินงานภายใน
- ความท้าทาย (Challenges):** ระบบคลังข้อมูล (Data Warehouse) เดิมที่ติดตั้งภายในองค์กร (On-premises) มีเทคโนโลยีที่ล้าสมัย, มีค่าใช้จ่ายในการดูแลรักษาสูง, ขาดความยืดหยุ่นในการขยายตัว และไม่สามารถรองรับข้อมูลที่ไม่มีโครงสร้าง (Unstructured Data) ซึ่งจำเป็นต่อการวิเคราะห์ขั้นสูงได้

- แนวทางการแก้ปัญหา (Solutions):** SCB ตัดสินใจย้ายคลังข้อมูล (Data Lake) ไปยังแพลตฟอร์ม Microsoft Azure ซึ่งทำให้ธนาคารสามารถใช้เครื่องมือวิเคราะห์ข้อมูลขั้นสูง, ปัญญาประดิษฐ์ (AI), และการเรียนรู้ของเครื่อง (Machine Learning) เพื่อวิเคราะห์ข้อมูลธุกรรมกว่า 12 ล้านจุด และตัวแปรอีกกว่า 200 รายการ เพื่อเพิ่มประสิทธิภาพการจัดการเงินสดในตู้ ATM และใช้ข้อมูลเชิงลึกในการทำการตลาดแบบเฉพาะบุคคล (Personalized Marketing)
- ผลลัพธ์:** การย้ายระบบสู่คลาวด์ช่วยลดต้นทุนการจัดเก็บข้อมูล, ลดกระบวนการที่ต้องทำด้วยมือลงถึง 40%, และเพิ่มการตอบสนองต่อแคมเปญการตลาดได้ถึง 10%
- ประเด็นสำคัญด้าน GRC:** เพื่อให้การใช้ข้อมูลเป็นไปอย่างปลอดภัยและมีประสิทธิภาพ SCB ได้จัดตั้ง สำนักงานธรรมาภิบาลข้อมูล (Data Governance Office) และ ศูนย์ความเป็นเลิศด้านการวิเคราะห์ (Analytics Center of Excellence) ขึ้นมาโดยเฉพาะ เพื่อสร้างกรอบและแนวปฏิบัติที่ชัดเจนในการกำกับดูแลข้อมูล

ตัวอย่างที่ 2: ธนาคารกรุงศรีอยุธยา (Krungsri) กับการสร้าง Hybrid Cloud บน AWS

- แรงผลักดัน (Drivers):** มีเป้าหมายที่จะเป็นหนึ่งในสถาบันการเงินที่มีนวัตกรรมสูงสุด ผ่านการขยายบริการธนาคารออนไลน์และโมบายแบงก์กิ้ง และการนำ AI มากระดับการบริการลูกค้า
- ความท้าทาย (Challenges):** เพชญ์ความท้าทายด้านการปฏิบัติการ (Operations) และการนำแนวทาง DevOps มาปรับใช้บน Public Cloud ให้เกิดประสิทธิภาพสูงสุด
- แนวทางการแก้ปัญหา (Solutions):** ธนาคารกรุงศรีฯ ร่วมมือกับพันธมิตร (Kyndryl) ในการสร้างสถาปัตยกรรมแบบ Hybrid Cloud บน Amazon Web Services (AWS) โดยย้าย Workload ที่มีความสำคัญเชิงกลยุทธ์ขึ้นสู่คลาวด์ และจัดตั้ง ศูนย์ความเป็นเลิศด้านคลาวด์ (Cloud Center of Excellence - CCoE) เพื่อสร้างมาตรฐานและแนวปฏิบัติที่ดีที่สุดในการบริหารจัดการระบบคลาวด์
- ผลลัพธ์:** ทำให้เกิดระบบอัตโนมัติในการจัดสรรทรัพยากร (Automated Provisioning), สามารถบริหารจัดการต้นทุนและสินทรัพย์จากส่วนกลางได้อย่างมีประสิทธิภาพ, และสามารถขยายขอบเขตการรักษาความปลอดภัยไปยังแอปพลิเคชันที่ทำงานอยู่บนคลาวด์ได้

4.2 แนวทางการดำเนินการอย่างเป็นขั้นตอน: Checklist สำหรับการย้าย Workload สู่คลาวด์อย่างสอดคล้องกับกฎหมายไทย

กระบวนการย้ายระบบสู่คลาวด์ให้ประสบความสำเร็จและสอดคล้องกับกฎหมายไทยนั้นต้องอาศัยการวางแผนอย่างเป็นระบบและรอบคอบในทุกขั้นตอน

ขั้นตอนที่ 1: การประเมินและวางแผน (Assessment and Planning)

- **ระบุข้อมูล:** ทำการสำรวจข้อมูล (Data Discovery) เพื่อระบุว่าข้อมูลใดเป็น "ข้อมูลส่วนบุคคล" ภายใต้คำนิยามของ PDPA และทำการจัดประเภทความอ่อนไหวของข้อมูล (Data Classification)
- **ประเมินสถานะ CII:** ตรวจสอบว่าองค์กรของท่านเข้าข่ายเป็น "หน่วยงานโครงสร้างพื้นฐานสำคัญทางสารสนเทศ" ภายใต้ พ.ร.บ. ไซเบอร์ฯ หรือไม่
- **ประเมินความเสี่ยง:** ดำเนินการประเมินความเสี่ยงทางไซเบอร์ตามแนวทางของ พ.ร.บ. ไซเบอร์ฯ และทำการประเมินผลกระทบด้านการคุ้มครองข้อมูลส่วนบุคคล (Data Protection Impact Assessment - DPIA) ตามหลักการของ PDPA
- **กำหนดนโยบาย:** จัดทำหรือทบทวนนโยบายความเป็นส่วนตัว (Privacy Policy) และนโยบายความมั่นคงปลอดภัยไซเบอร์ (Cybersecurity Policy) ให้ครอบคลุมถึงการใช้งานบริการคลาวด์

ขั้นตอนที่ 2: การเลือกผู้ให้บริการและจัดทำข้อตกลง (Provider Selection and Agreements)

- **ตรวจสอบ CSP:** ประเมินมาตรการรักษาความปลอดภัยและตรวจสอบให้รับรองมาตรฐานสากลของผู้ให้บริการคลาวด์ (CSP)
- **พิจารณาที่ตั้งข้อมูล:** ตัดสินใจเลือก Region ของศูนย์ข้อมูล โดยคำนึงถึงภาระการโอนข้อมูลข้ามประเทศของ PDPA เป็นสำคัญ
- **จัดทำ DPA:** เจรจาและลงนามในข้อตกลงการประมวลผลข้อมูล (Data Processing Agreement - DPA) ที่มีเนื้อหาสอดคล้องกับข้อกำหนดของ PDPA อย่างครบถ้วน

ขั้นตอนที่ 3: การออกแบบและกำหนดค่าทางเทคนิค (Technical Design and Configuration)

- **ออกแบบสถาปัตยกรรม:** ออกแบบเครือข่ายส่วนตัวเสมือน (VPC), Subnet, และ Security Groups โดยใช้หลักการแบ่งส่วนเครือข่ายอย่างละเอียด (Microsegmentation) และ Zero Trust
- **กำหนดค่า IAM:** สร้าง Roles และ Policies สำหรับผู้ใช้และบริการ โดยยึดตามหลักการให้สิทธิ์น้อยที่สุดเท่าที่จำเป็น (Principle of Least Privilege)
- **กำหนดค่าการเข้ารหัส:** เปิดใช้งานการเข้ารหัสข้อมูลทั้งในขณะจัดเก็บ (at rest) และในขณะส่งผ่าน (in transit) สำหรับข้อมูลที่ละเอียดอ่อนทั้งหมด
- **กำหนดค่า Logging & Monitoring:** เปิดใช้งานบริการบันทึกและตรวจสอบกิจกรรมทั้งหมดบนแพลตฟอร์มคลาวด์ เพื่อให้สามารถตรวจสอบย้อนหลังได้

ขั้นตอนที่ 4: การย้ายระบบและการตรวจสอบ (Migration and Validation)

- **ย้ายข้อมูลอย่างปลอดภัย:** ใช้วิธีการย้ายข้อมูลที่มีการเข้ารหัสและปลอดภัยตลอดกระบวนการ
- **ทดสอบเจาะระบบ (Penetration Testing):** ดำเนินการทดสอบเจาะระบบบนสภาพแวดล้อมคลาวด์ใหม่ เพื่อค้นหาและแก้ไขช่องโหว่ก่อนการใช้งานจริง
- **ตรวจสอบการกำหนดค่า:** ใช้เครื่องมือตรวจสอบเพื่อยืนยันว่าการกำหนดค่าทางเทคนิคทั้งหมดสอดคล้องกับ Security Baseline และนโยบายที่ได้วางไว้

ขั้นตอนที่ 5: การดำเนินงานและการตรวจสอบอย่างต่อเนื่อง (Operation and Continuous Monitoring)

- **บริหารจัดการแพตช์:** จัดทำกระบวนการอัปเดตและติดตั้งแพตช์ความปลอดภัยสำหรับเครื่องเสมือน (VM) และคอนเทนเนอร์อย่างสม่ำเสมอ
- **ตรวจสอบและตอบสนอง:** เฝ้าระวัง Log และ Alert จากระบบ Monitoring อย่างต่อเนื่อง และดำเนินการตามแผนรับมือเหตุการณ์ (IRP) เมื่อตรวจพบความผิดปกติ
- **ทบทวนสิทธิ์การเข้าถึง:** จัดให้มีการตรวจสอบและทบทวนสิทธิ์การเข้าถึงของผู้ใช้และบริการเป็นประจำ
- **ฝึกอบรมพนักงาน:** จัดการฝึกอบรมเพื่อสร้างความตระหนักรู้ด้านความปลอดภัยไซเบอร์และข้อกำหนดของ PDPA ให้กับพนักงานอย่างต่อเนื่อง

สรุป (Summary)

การนำระบบคลาวด์มาใช้ในองค์กรไทยให้ประสบความสำเร็จและยั่งยืนนั้น ไม่สามารถมองแยกส่วนระหว่างมิติทางเทคโนโลยีและมิติทางกฎหมายได้อีกต่อไป ผู้เขียนชี้ว่าด้านคลาวด์ในปัจจุบันจำเป็นต้องมีความสามารถในการบูรณาการความรู้ทั้งสองด้านเข้าด้วยกัน เพื่อสร้างสถาปัตยกรรมที่ทั้งทรงประสิทธิภาพและสอดคล้องกับข้อบังคับ

สิ่งสำคัญที่ต้องตระหนักอยู่เสมอคือ แม้จะเลือกใช้บริการจากผู้ให้บริการคลาวด์ชั้นนำของโลกที่มีมาตรการความปลอดภัยระดับสูง แต่ความรับผิดชอบสูงสุดในการปฏิบัติตามกฎหมายไทย โดยเฉพาะอย่างยิ่ง พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคล ยังคงอยู่กับองค์กรผู้ใช้บริการในฐานะ "ผู้ควบคุมข้อมูลส่วนบุคคล"

นอกจากนี้ ครอบกฎหมายและมาตรฐานของประเทศไทยมีการพัฒนาและเปลี่ยนแปลงอย่างต่อเนื่อง ดังที่เห็นได้จากการออกประกาศมาตรฐานความปลอดภัยไซเบอร์บนระบบคลาวด์โดย สกมช. ดังนั้น การเรียนรู้และปรับตัวอย่างสม่ำเสมอจึงเป็นทักษะที่จำเป็นอย่างยิ่งสำหรับบุคลากรด้านคลาวด์ทั้งในปัจจุบันและอนาคต เพื่อให้สามารถนำพาองค์กรใช้ประโยชน์จากเทคโนโลยีคุณภาพดีได้อย่างเต็มศักยภาพบนฐานรากที่มั่นคง ปลอดภัย และมีประสิทธิภาพ

บทที่ 14

หลักการเป็นวิทยากรที่ดี

(Principles of Effective Instruction)

วัตถุประสงค์การเรียนรู้

การถ่ายทอดองค์ความรู้และทักษะอย่างมีประสิทธิภาพ ถือเป็นหัวใจสำคัญของการพัฒนาบุคลากรในทุกองค์กร วิทยากรไม่ได้เป็นเพียงผู้บรรยาย แต่มีบทบาทเป็นผู้อำนวยการเรียนรู้ ซึ่งต้องบูรณาการทั้งศาสตร์และศิลป์ในการสร้างสรรค์ประสบการณ์ที่ส่งเสริมให้ผู้เรียนสามารถนำความรู้ไปประยุกต์ใช้ได้จริง บทบาทดังกล่าวจึงต้องการการเตรียมความพร้อมอย่างเป็นระบบ ความเข้าใจในหลักการเรียนรู้ และการพัฒนาตนเองอย่างต่อเนื่อง

เนื้อหาในบทเรียนนี้จะกล่าวถึงหลักการสำคัญของการเป็นวิทยากรที่มีประสิทธิภาพ ตั้งแต่คุณลักษณะพื้นฐานและจรรยาบรรณที่วิทยากรพึงมี กระบวนการเตรียมความพร้อมก่อนการบรรยายอย่างละเอียด เทคนิคการนำเสนอและการสร้างปฏิสัมพันธ์เพื่อส่งเสริมบรรยากาศการเรียนรู้เชิงบวก ไปจนถึงแนวทางการวัดผลและประเมินผลที่สอดคล้องกับวัตถุประสงค์ของหลักสูตร เพื่อให้มั่นใจว่าการฝึกอบรมนั้นบรรลุเป้าหมายอย่างแท้จริง

เมื่อสิ้นสุดการเรียนรู้ในบทเรียนนี้ ผู้เรียนจะสามารถ

- อธิบายคุณลักษณะที่สำคัญและจรรยาบรรณของวิทยากรที่มีประสิทธิภาพได้
- วางแผนและเตรียมความพร้อมก่อนการบรรยายได้อย่างเป็นระบบ ตั้งแต่การวิเคราะห์ผู้เรียนไปจนถึงการจัดเตรียมสื่อการสอน
- ประยุกต์ใช้เทคนิคการนำเสนอและการสื่อสารเพื่อสร้างปฏิสัมพันธ์และส่งเสริมบรรยากาศการเรียนรู้เชิงบวก
- ออกแบบและจัดทำการวัดผลและประเมินผลการเรียนรู้ที่สอดคล้องกับวัตถุประสงค์ของหลักสูตรได้
- ระบุแนวทางการพัฒนาตนเองอย่างต่อเนื่องเพื่อความเป็นเลิศในวิชาชีพวิทยากร

หน่วยที่ 1 คุณลักษณะและจรรยาบรรณของวิทยากร (Qualities and Ethics of an Instructor)

การเป็นวิทยากรที่มีประสิทธิภาพนั้นไม่ได้อ้าศัยเพียงความสามารถในการพูด แต่ต้องตั้งอยู่บนฐานราก คุณลักษณะที่จำเป็นและจรรยาบรรณวิชาชีพที่มั่นคง คุณลักษณะเหล่านี้เป็นเครื่องมือที่ช่วยสร้างความน่าเชื่อถือ ในขณะที่จรรยาบรรณเป็นกรอบการปฏิบัติที่กำกับการกระทำทั้งหมดให้เป็นไปเพื่อประโยชน์สูงสุดของผู้เรียน

1.1 คุณลักษณะสำคัญของวิทยากร (Core Qualities of an Instructor)

คุณลักษณะสำคัญของวิทยากรเปรียบเสมือนเสาหลักที่ค้ำจุนความสำเร็จในการถ่ายทอดความรู้ ซึ่งประกอบด้วยองค์ประกอบหลัก 3 ประการ ดังนี้

- 1) **ความรู้จริง (Subject Matter Expertise)** พื้นฐานที่สำคัญที่สุดของวิทยากรคือความเชี่ยวชาญในเนื้อหาที่จะสอน วิทยากรต้องมีความรู้ที่ลึกซึ้งและกว้างขวาง สามารถอธิบายรายละเอียดที่ซับซ้อน ชี้แจงเหตุผลเบื้องหลัง และบอกเล่าถึงที่มาหรือสมมติฐานขององค์ความรู้นั้นได้อย่างชัดเจน ความรู้จริงนี้ไม่ได้จำกัดอยู่แค่ในตำรา แต่ยังรวมถึงความสามารถในการประยุกต์ใช้ให้เห็นเป็นรูปธรรม ซึ่งมักมาจากการทดลอง 试验 ที่มีความน่าเชื่อถือและความมั่นใจในการถ่ายทอด ทำให้วิทยากรสามารถตอบคำถามและรับมือกับสถานการณ์ที่ไม่คาดคิดได้อย่างมั่นคง
- 2) **ทักษะการถ่ายทอด (Communication and Delivery Skills)** การมีความรู้เพียงอย่างเดียวไม่เพียงพอ หากไม่สามารถสื่อสารให้ผู้อื่นเข้าใจได้ วิทยากรที่ดีจึงต้องมีทักษะในการถ่ายทอดความรู้ที่ซับซ้อนให้กลایเป็นเรื่องง่าย ซึ่งครอบคลุมถึงการสรุปประเด็นสำคัญให้ชัดเจน การเลือกใช้ภาษาที่เหมาะสมกับระดับของผู้เรียน และการมีเทคนิคการนำเสนอที่หลากหลายเพื่อสร้างความเข้าใจและความน่าสนใจ ทักษะนี้ยังรวมถึงความสามารถในการใช้สื่อการสอนอย่างมีประสิทธิภาพเพื่อสนับสนุนการเรียนรู้
- 3) **บุคลิกภาพที่ส่งเสริมการเรียนรู้ (Personality that Fosters Learning)** คุณลักษณะส่วนบุคคลมีผลอย่างมากต่อบรรยากาศในห้องเรียน บุคลิกภาพที่ดี เช่น ความมั่นใจในตนเอง การมีมนุษยสัมพันธ์ที่ดี ความร่าเริงยิ้มแย้มแจ่มใส จะช่วยลดช่องว่างและสร้างความเป็นกันเองกับผู้เรียน นอกจากนี้ การเป็นคนซ่างสังเกตจะช่วยให้วิทยากรสามารถประเมินปฏิกรรมของผู้เรียน และปรับเปลี่ยนวิธีการสอนให้เหมาะสมได้ ขณะที่ความคิดสร้างสรรค์และการมโนขันจะทำให้การเรียนรู้ไม่น่าเบื่อและน่าติดตาม

1.2 จรรยาบรรณวิชาชีพ (Professional Ethics)

จรรยาบรรณเป็นการครอบคลุมที่สำคัญซึ่งขึ้นมาให้กับวิทยากรปฏิบัติหน้าที่อย่างมีความรับผิดชอบและมีเกียรติ โดยมีหลักการสำคัญดังนี้

- 1) **การมุ่งประโยชน์ของผู้เรียนเป็นที่ตั้ง (Learner-Centric Focus)** หัวใจของจรรยาบรรณวิทยากรคือการคำนึงถึงประโยชน์ที่ผู้เรียนจะได้รับเป็นอันดับแรก ทุกองค์ประกอบของ การฝึกอบรม ตั้งแต่การออกแบบเนื้อหา การเลือกกิจกรรม ไปจนถึงวิธีการถ่ายทอด ล้วนต้องมี เป้าหมายเพื่อส่งเสริมการเรียนรู้และพัฒนาศักยภาพของผู้เรียนอย่างแท้จริง
- 2) **ความรับผิดชอบและความซื่อสัตย์ (Responsibility and Integrity)** วิทยากรต้องมี ความรับผิดชอบต่อความรู้ที่ตนถ่ายทอด โดยต้องมั่นใจว่าเป็นข้อมูลที่ถูกต้องและ เป็นจริง นอก จากนี้ ต้องมีความซื่อสัตย์ในวิชาชีพ ไม่ฉ้อฉลโอกาสจากการเป็นวิทยากร เพื่อแสวงหาผลประโยชน์ส่วนตัวในทางที่ไม่เหมาะสม และหลีกเลี่ยงการกล่าวโวใจตีบคุคลหรือ องค์กรอื่น
- 3) **การเป็นแบบอย่างที่ดี (Being a Good Role Model)** วิทยากรที่ดีต้องเป็นแบบอย่างใน การประพฤติปฏิบัติตน โดยพัฒนาระบบส่วนตัวควรสอนด้วยกับเนื้อหาที่สอน เพื่อสร้าง ความน่าเชื่อถือและเป็นแรงบันดาลใจให้แก่ผู้เรียน

คุณลักษณะและจรรยาบรรณของวิทยากรนั้นมีความสัมพันธ์กันอย่างแยกไม่ออก จรรยาบรรณไม่ได้ เป็นเพียงข้อบังคับ แต่เป็นแรงผลักดันภายในที่ขับเคลื่อนให้วิทยากรต้องพัฒนาคุณลักษณะที่จำเป็น ตัวอย่างเช่น การยึดมั่นในจรรยาบรรณที่ว่า "ต้องมุ่งประโยชน์ของผู้เรียนเป็นที่ตั้ง" จะกระตุนให้วิทยากรต้อง พัฒนา "ทักษะการถ่ายทอด" อยู่เสมอ เพื่อค้นหาวิธีการที่ดีที่สุดที่จะทำให้ผู้เรียนเกิดความเข้าใจอย่างแท้จริง แทนที่จะมุ่งเน้นเพียงการบรรยายเนื้อหาให้จบไป ในท่านองเดียว กัน คุณลักษณะ "ความรู้จริง" ก็เป็นรากฐานที่ สำคัญของจรรยาบรรณข้อที่ว่า "ต้องมั่นใจว่ามีความรู้จริงในเรื่องที่จะสอน" ดังนั้น จรรยาบรรณจึงเป็นทั้ง จุดเริ่มต้นและเป้าหมายที่หล่อหลอมให้บุคคลหนึ่งสามารถเป็นวิทยากรที่มีประสิทธิภาพได้อย่างสมบูรณ์

หน่วยที่ 2 การเตรียมความพร้อมก่อนการบรรยาย (Pre-Instruction Preparation)

ความสำเร็จของการบรรยายไม่ได้เกิดขึ้นเฉพาะหน้า แต่เป็นผลมาจากการวางแผนและเตรียมการอย่างเป็นระบบและรอบคอบ การเตรียมความพร้อมที่ดีเปรียบเสมือนการวางแผนรากฐานที่แข็งแกร่ง ซึ่งช่วยลดความประหม่า สร้างความมั่นใจ และทำให้กระบวนการถ่ายทอดความรู้เป็นไปอย่างราบรื่นและบรรลุวัตถุประสงค์ที่ตั้งไว้

2.1 การวิเคราะห์ผู้เรียน (Learner Analysis)

ขั้นตอนแรกที่สำคัญที่สุดก่อนการออกแบบเนื้อหา คือการทำความเข้าใจผู้เรียน การวิเคราะห์ผู้เรียน ช่วยให้วิทยากรสามารถออกแบบหลักสูตรที่ตอบสนองต่อความต้องการและความพร้อมของผู้เรียนได้อย่างแท้จริง ซึ่งควรพิจารณาในมิติต่างๆ ดังนี้

- 1) **ความรู้และประสบการณ์เดิม** ประเมินว่าผู้เรียนมีความรู้พื้นฐานในหัวข้อที่จะบรรยายมากน้อยเพียงใด เพื่อหลีกเลี่ยงการสอนเนื้อหาที่ซ้ำซ้อนหรือง่ายเกินไป หรือในทางกลับกัน คือการสอนเนื้อหาที่ยากเกินกว่าจะทำความเข้าใจได้ การทำแบบทดสอบก่อนเรียน (Pre-test) หรือการสำรวจเบื้องต้นสามารถให้ข้อมูลในส่วนนี้ได้
- 2) **ลักษณะทั่วไป** พิจารณาข้อมูลประชากรศาสตร์ เช่น อายุ ระดับการศึกษา และบทบาทหน้าที่การงานของผู้เรียน ซึ่งเป็นปัจจัยที่ส่งผลต่อรูปแบบการยกตัวอย่าง การเลือกใช้ภาษา และการออกแบบกิจกรรมให้เหมาะสม
- 3) **ความคาดหวังและเป้าหมาย** ทำความเข้าใจว่าผู้เรียนคาดหวังว่าจะได้รับอะไรจากการอบรมครั้งนี้ และต้องการนำความรู้ไปใช้ประโยชน์ในด้านใด เพื่อให้สามารถเชื่อมโยงเนื้อหา กับเป้าหมายของผู้เรียนได้โดยตรง
- 4) **รูปแบบการเรียนรู้** ตระหนักรู้ว่าผู้เรียนแต่ละคนมีรูปแบบการเรียนรู้ที่แตกต่างกัน บางคนเรียนรู้ได้ดีจากการฟัง บางคนเรียนรู้จากการมองเห็น หรือบางคนต้องได้ลงมือปฏิบัติ การทราบข้อมูลเหล่านี้จะช่วยให้วิทยากรสามารถออกแบบกิจกรรมที่หลากหลายเพื่อตอบสนองต่อผู้เรียนทุกกลุ่มได้

2.2 การกำหนดวัตถุประสงค์การเรียนรู้เชิงพฤติกรรม (Defining Behavioral Learning Objectives)

วัตถุประสงค์การเรียนรู้ที่ชัดเจนเป็นหัวใจสำคัญที่กำหนดทิศทางของหลักสูตรทั้งหมด โดยทำหน้าที่เป็นแนวทางในการเลือกเนื้อหา กิจกรรม และวิธีการประเมินผล วัตถุประสงค์ที่ดีควรเขียนในรูปแบบ "เชิงพฤติกรรม" ซึ่งหมายถึงการระบุถึงสิ่งที่ผู้เรียนจะสามารถ "ทำได้" หลังสิ้นสุดการเรียนรู้ โดยมีองค์ประกอบสำคัญ 3 ส่วน ดังนี้

- 1) **พฤติกรรมที่คาดหวัง (Behavior)** ระบุการกระทำที่สามารถสังเกตและวัดผลได้ โดยใช้คำกริยาที่ชัดเจน เช่น อธิบาย เปรียบเทียบ วิเคราะห์ สาสัต ออกแบบ สร้าง และประเมินค่า
- 2) **เงื่อนไข (Condition)** ระบุสถานการณ์หรือข้อกำหนดที่พฤติกรรมนั้นจะเกิดขึ้น เช่น "เมื่อ กำหนดกรณีศึกษาให้" "โดยใช้ข้อมูลจากตารางที่ 1" หรือ "โดยไม่ต้องเปิดหนังสือ"
- 3) **เกณฑ์ (Criteria)** ระบุมาตรฐานหรือระดับความสำเร็จขั้นต่ำที่ยอมรับได้ เช่น "ได้อย่างน้อย 3 ประการ" "ถูกต้องไม่น้อยกว่าร้อยละ 80" หรือ "ภายในเวลา 15 นาที"
 - ตัวอย่างวัตถุประสงค์ที่ไม่ดี "เพื่อให้ผู้เรียนเข้าใจหลักการปฐมพยาบาลเบื้องต้น" (คำว่า "เข้าใจ" ไม่สามารถสังเกตหรือวัดผลได้โดยตรง)
 - ตัวอย่างวัตถุประสงค์ที่ดี "เมื่อสิ้นสุดการเรียนรู้ (เงื่อนไข) ผู้เรียนสามารถ สาสัต ขั้นตอนการทำ CPR (พฤติกรรม) ได้อย่างถูกต้องตามลำดับ ครบถ้วน (เกณฑ์)"

2.3 การออกแบบโครงสร้างเนื้อหาและกิจกรรม (Structuring Content and Activities)

เมื่อมีวัตถุประสงค์ที่ชัดเจนแล้ว ขั้นตอนต่อไปคือการออกแบบโครงสร้างเนื้อหาและกิจกรรมให้สอดคล้องกัน

- 1) **หลักการจัดลำดับเนื้อหา** เนื้อหาควรถูกเรียบเรียงอย่างมีตรรกะและต่อเนื่อง เพื่อให้ผู้เรียนสามารถสร้างความเข้าใจได้อย่างเป็นลำดับขั้น เช่น การจัดลำดับจากง่ายไปสู่ยาก จากภาพรวมไปสู่รายละเอียด หรือตามลำดับเหตุการณ์
- 2) **โครงสร้างการบรรยาย** โดยทั่วไปควรแบ่งการนำเสนอออกเป็น 3 ส่วนหลัก ได้แก่
 - **ขั้นนำ** เพื่อสร้างความสัมพันธ์ กระตุ้นความสนใจ และแจ้งวัตถุประสงค์
 - **ขั้นนำเสนอเนื้อหา** เพื่อถ่ายทอดองค์ความรู้หลักพร้อมยกตัวอย่างประกอบอย่างเป็นระบบ
 - **ขั้นสรุป** เพื่อทบทวนประเด็นสำคัญ เปิดโอกาสให้ซักถาม และเขื่อมโยงสู่หัวข้อถัดไป
- 3) **การเลือกกิจกรรม** ควรเลือกกิจกรรมที่สอดคล้องกับเนื้อหาและส่งเสริมให้ผู้เรียนบรรลุวัตถุประสงค์ เช่น การอภิปรายกลุ่มเพื่อแลกเปลี่ยนความคิดเห็น การใช้กรณีศึกษาเพื่อฝึกการวิเคราะห์ หรือการแสดงบทบาทสมมติเพื่อฝึกทักษะการปฏิบัติ

2.4 การเตรียมสื่อการสอนและสภาพแวดล้อม (Preparing Materials and Environment)

สื่อการสอนและสภาพแวดล้อมเป็นปัจจัยสนับสนุนที่สำคัญต่อการเรียนรู้

- 1) **การเลือกสื่อ** สื่อการสอนควรมีความสัมพันธ์กับเนื้อหา เหมาะสมกับวัยและระดับความรู้ของผู้เรียน และมีคุณภาพดี สามารถใช้สื่อได้หลากหลายประเภท เช่น สไลด์นำเสนอ เอกสารประกอบการบรรยาย วิดีโอทัศน์ หรือของจริง/ของจำลอง เพื่อสร้างความน่าสนใจและช่วยให้เข้าใจเนื้อหาได้ง่ายขึ้น
- 2) **การเตรียมสื่อ ก่อนการบรรยาย** ควรตรวจสอบสภาพของสื่อและอุปกรณ์ที่เกี่ยวข้องทั้งหมดให้อยู่ในสภาพพร้อมใช้งาน ทดลองใช้งานล่วงหน้าเพื่อป้องกันปัญหาทางเทคนิค และจัดเตรียมสื่อตามลำดับการใช้งานเพื่อความสะดวก
- 3) **การจัดสภาพแวดล้อม** สถานที่จัดอบรมควรมีความสะอาด เป็นระเบียบ มีแสงสว่างเพียงพอ และปราศจากเสียงรบกวน เพื่อสร้างบรรยากาศที่เอื้อต่อการเรียนรู้

2.5 การซักซ้อมและการบริหารเวลา (Rehearsal and Time Management)

การซักซ้อมเป็นขั้นตอนสุดท้ายที่ช่วยเติมความพร้อม การซ้อมบรรยายจะช่วยให้วิทยากรมีความคล่องแคล่วในการนำเสนอ สร้างความมั่นใจ และสามารถบริหารจัดการเวลาได้อย่างมีประสิทธิภาพ ควรมีการวางแผนจัดสรรเวลาสำหรับแต่ละหัวข้อและกิจกรรมอย่างชัดเจน และเตรียมพร้อมที่จะปรับเปลี่ยนหรือยืดหยุ่นได้ตามสถานการณ์หน้างาน

กระบวนการเตรียมความพร้อมนี้ไม่ใช่ขั้นตอนที่แยกจากกัน แต่เป็นวงจรการออกแบบที่เชื่อมโยงกันอย่างเป็นระบบ ผลลัพธ์จากการวิเคราะห์ผู้เรียนจะถูกนำมาใช้กำหนดวัตถุประสงค์การเรียนรู้ที่เหมาะสม จากนั้นวัตถุประสงค์ที่ชัดเจนจะเป็นแนวทางในการคัดเลือกและจัดลำดับเนื้อหาและกิจกรรม และท้ายที่สุดเนื้อหาและกิจกรรมที่เลือกจะเป็นตัวกำหนดว่าต้องใช้สื่อการสอนประเภทใดจึงจะถ่ายทอดได้อย่างมีประสิทธิภาพสูงสุด การละเอียดขั้นตอนได้ขั้นตอนหนึ่ง เช่น การออกแบบเนื้อหาโดยไม่ได้เคราะห์ผู้เรียนก่อนอาจส่งผลให้หลักสูตรขาดประสิทธิผล เนื่องจากเนื้อหาอาจไม่ตรงกับความต้องการหรือระดับความรู้ของผู้เรียนอย่างแท้จริง

หน่วยที่ 3 เทคนิคการนำเสนอและการสร้างปฏิสัมพันธ์ (Presentation and Interaction Techniques)

เมื่อการเตรียมความพร้อมเสร็จสมบูรณ์แล้ว ความสำเร็จในห้องเรียนจะขึ้นอยู่กับความสามารถของวิทยากรในการนำเสนอเนื้อหาและสร้างปฏิสัมพันธ์กับผู้เรียน การนำเสนอที่มีประสิทธิภาพไม่ใช่เพียงการถ่ายทอดข้อมูล แต่คือการสร้างประสบการณ์การเรียนรู้ที่ดึงดูด น่าสนใจ และกระตุ้นให้ผู้เรียนมีส่วนร่วมอย่างแข็งขัน

3.1 ทักษะการสื่อสารที่มีประสิทธิภาพ (Effective Communication Skills)

ทักษะการสื่อสารเป็นเครื่องมือที่สำคัญที่สุดของวิทยากร ซึ่งประกอบด้วย 2 ส่วนหลัก ได้แก่

- 1) **วัจนาภาษา (Verbal Communication)** หมายถึงสิ่งที่พูดและวิธีการพูด ซึ่งรวมถึงการใช้น้ำเสียงที่มีพลัง ดังฟังชัด และน่าฟัง มีการเน้นเสียงสูง-ต่ำและจังหวะจะโอนเพื่อสร้างความน่าสนใจ การเลือกใช้ถ้อยคำความมีความสุภาพ ชัดเจน เข้าใจง่าย และเหมาะสมกับระดับของผู้เรียน
- 2) **อวจนาภาษา (Non-verbal Communication)** ภาษากาย เป็นองค์ประกอบที่มีอิทธิพลอย่างสูงต่อการสื่อสาร วิทยากรควรสนับสนุนผู้เรียนอย่างทั่วถึงเพื่อสร้างความเชื่อมโยง การแสดงออกทางสีหน้าที่เป็นมิตรและยิ้มแย้มจะช่วยสร้างบรรยากาศที่ดี และการใช้ท่าทางประกอบการพูดอย่างเป็นธรรมชาติและเหมาะสมจะช่วยเสริมสร้างความเข้าใจและความน่าเชื่อถือ

3.2 การสร้างบรรยากาศการเรียนรู้เชิงบวก (Creating a Positive Learning Atmosphere)

บรรยากาศในห้องเรียนส่งผลโดยตรงต่อการเรียนรู้ของผู้เรียน บรรยากาศที่ดี ปลอดภัย และเป็นกันเองจะช่วยให้ผู้เรียนกล้าแสดงความคิดเห็น กล้าซักถาม และเปิดใจรับความรู้ใหม่ๆ

- 1) **การสร้างความคุ้นเคย** เริ่มต้นการอบรมด้วยกิจกรรมละลายพฤติกรรม (Ice Breaking) เพื่อให้ผู้เรียนได้ทำความรู้จักกันและลดความเกร็ง การทักทายผู้เรียนแต่ละคนอย่างเป็นกันเอง หรือการชวนคุยในเรื่องทั่วไปก่อนเริ่มนิءอหัวใจช่วยสร้างความสัมพันธ์ที่ดี
- 2) **การแสดงความใส่ใจ** การพยายามจดจำชื่อผู้เรียนและเรียกชื่อเมื่อมีปฏิสัมพันธ์จะทำให้ผู้เรียนรู้สึกว่าตนเองมีความสำคัญ การตั้งใจรับฟังความคิดเห็นของผู้เรียนอย่างให้เกียรติและไม่ตัดสินเป็นสิ่งสำคัญอย่างยิ่งในการสร้างความไว้วางใจ
- 3) **การมีอารมณ์ขัน** การสอดแทรกอารมณ์ขันอย่างเหมาะสมและถูกกาลเทศสามารถช่วยลดความตึงเครียดและทำให้บรรยากาศการเรียนรู้ผ่อนคลายและสนุกสนานยิ่งขึ้น

3.3 กลยุทธ์การเรียนรู้เชิงรุก (Active Learning Strategies)

การเรียนรู้ที่มีประสิทธิภาพเกิดขึ้นเมื่อผู้เรียนได้มีส่วนร่วมอย่างแข็งขัน (Active Learning) แทนที่จะเป็นเพียงผู้รับฟัง วิทยากรควรเปลี่ยนบทบาทจากผู้บรรยายเป็นผู้อำนวยความสนใจในการเรียนรู้ โดยใช้กลยุทธ์ต่างๆ ดังนี้

- 1) **เทคนิคการใช้คำถ้า** คำถ้าเป็นเครื่องมือที่ทรงพลังในการกระตุนให้ผู้เรียนคิดตาม วิเคราะห์ และสังเคราะห์ข้อมูล ควรใช้คำถ้าที่หลากหลาย ตั้งแต่คำถ้าเพื่อทบทวนความรู้ความจำไปจนถึงคำถ้าที่ต้องการการวิเคราะห์ เปรียบเทียบ และประยุกต์ใช้ สิ่งสำคัญคือการให้เวลาผู้เรียนได้คิดหาคำตอบ และสร้างบรรยายกาศที่ปลอดภัยที่ผู้เรียนกล้าที่จะตอบแม้ไม่มั่นใจ ตัวอย่างเช่น แทนที่จะถามว่า "ทุกคนเข้าใจเรื่องนี้ใช่ไหมครับ" ซึ่งมักจะไม่มีใครตอบ วิทยากรอาจเปลี่ยนเป็น "จากหลักการที่เราเพิ่งเรียนไป ใครจะยกตัวอย่างได้บ้างว่าจะนำไปรับใช้กับงานของตนเองได้อย่างไร" คำถามลักษณะนี้จะกระตุนให้เกิดการคิดและสะท้อนความเข้าใจที่แท้จริง
- 2) **กิจกรรมกลุ่ม** การจัดกิจกรรมให้ผู้เรียนได้ทำงานร่วมกัน เช่น การอภิปรายกลุ่มย่อย การระดมสมอง หรือการทำกรณีศึกษา จะช่วยส่งเสริมการเรียนรู้จากเพื่อน (Peer Learning) และพัฒนาทักษะการทำงานร่วมกัน

3.4 การใช้สื่อประกอบการบรรยายอย่างมีประสิทธิผล (Effective Use of Instructional Media)

สื่อการสอนเป็นเครื่องมือช่วยที่สำคัญ แต่ต้องใช้อย่างถูกวิธี

- 1) **วัตถุประสงค์ของการใช้สื่อ** สื่อควรถูกใช้เพื่อช่วยดึงดูดความสนใจ ทำให้เนื้อหาที่ซับซ้อนกลายเป็นรูปธรรมและเข้าใจง่ายขึ้น หรือเพื่อสรุปประเด็นสำคัญ
- 2) **แนวทางปฏิบัติ** ไม่ควรบรรจุเนื้อหาลงในสไลด์มากเกินไปจนผู้เรียนต้องอ่านตามตลอดเวลา วิทยากรควรเป็นจุดศูนย์กลางของการนำเสนอ ไม่ใช้สื่อที่ใช้ สื่อควรทำหน้าที่เสริมการบรรยาย ไม่ใช่มาแทนที่การบรรยาย และควรเลือกใช้ในปริมาณที่เหมาะสม ไม่มากจนเกินความจำเป็น

การสร้างปฏิสัมพันธ์นั้นไม่ได้เป็นเพียง "กิจกรรม" ที่ลูกแทรกเข้ามาเป็นครั้งคราว แต่เป็นผลลัพธ์โดยตรงของ "บรรยาย" ที่วิทยากรสร้างขึ้น บรรยายศาสตร์ที่อบอุ่น ปลอดภัย และไว้วางใจซึ้งกันและกัน ถือเป็นเงื่อนไขที่จำเป็นอย่างยิ่งที่จะทำให้กลยุทธ์การเรียนรู้เชิงรุก เช่น การถาม-ตอบ หรือการอภิปราย สามารถเกิดขึ้นได้อย่างมีประสิทธิภาพสูงสุด ผู้เรียนจะไม่กล้าเสียงที่จะถามคำถามที่อาจดูไม่ถูกต้อง หรือแบ่งปันความคิดเห็นที่แตกต่าง หากพวกรู้สึกว่าอาจถูกตัดสินหรือตำหนิ ดังนั้น ทักษะการสื่อสารระหว่างบุคคลของวิทยากรจึงไม่ใช่เพียง "ทักษะเสริม" แต่เป็นเครื่องมือพื้นฐานที่สำคัญที่สุดในการเปิดประชุมสู่การมีส่วนร่วมและการเรียนรู้ในระดับลึกของผู้เรียน

หน่วยที่ 4 การวัดผลและประเมินผลการเรียนรู้ (Measuring and Evaluating Learning)

การวัดผลและประเมินผลเป็นกระบวนการที่ขาดไม่ได้ในการฝึกอบรม เพื่อให้ทราบว่าผู้เรียนได้บรรลุวัตถุประสงค์การเรียนรู้ที่ตั้งไว้หรือไม่ และเพื่อรวบรวมข้อมูลสำหรับนำไปปรับปรุงหลักสูตรให้มีประสิทธิภาพยิ่งขึ้นในอนาคต

4.1 หลักการและวัตถุประสงค์ของการประเมินผล (Principles and Purposes of Evaluation)

การประเมินผลไม่ได้ทำขึ้นเพื่อตัดสินผู้เรียนเพียงอย่างเดียว แต่เมื่อวัตถุประสงค์ที่หลากหลาย :

- 1) เพื่อประเมินผู้เรียน เพื่อตรวจสอบว่าผู้เรียนมีความรู้ ความเข้าใจ และทักษะเพิ่มขึ้นตามวัตถุประสงค์ที่กำหนดไว้หรือไม่
- 2) เพื่อวินิจฉัย เพื่อค้นหาจุดแข็งและจุดอ่อนของผู้เรียน หรือประเด็นที่เนื้อหา�ังมีความเข้าใจคลาดเคลื่อน เพื่อให้วิทยากรสามารถให้ความช่วยเหลือหรือปรับปรุงการสอนได้
- 3) เพื่อประเมินหลักสูตร เพื่อประเมินประสิทธิภาพของเนื้อหา กิจกรรม และวิธีการสอน สำหรับนำไปพัฒนาและปรับปรุงการจัดอบรมในครั้งต่อไป

หลักการสำคัญของการวัดผลคือ ต้องมีความสอดคล้องกับวัตถุประสงค์การเรียนรู้ที่กำหนดไว้ (Alignment) ควรเลือกใช้เครื่องมือวัดผลที่หลากหลายและมีคุณภาพ และดำเนินการประเมินอย่างยุติธรรมและโปร่งใส

4.2 ประเภทของแบบทดสอบ (Types of Assessments)

เครื่องมือที่ใช้ในการวัดผลการเรียนรู้สามารถแบ่งได้เป็น 2 ประเภทหลัก:

- 1) ข้อสอบปรนัย (Objective Tests) เป็นแบบทดสอบที่มีคำตอบที่ถูกต้องชัดเจนแน่นอน ทำให้การตรวจให้คะแนนเป็นไปอย่างตรงไปตรงมาและเป็นมาตรฐาน ประเภทที่นิยมใช้ ได้แก่ แบบถูก-ผิด แบบเติมคำ แบบจับคู่ และโดยเฉพาะอย่างยิ่งแบบเลือกตอบ (Multiple-choice) ข้อสอบปรนัยเหมาะสมสำหรับการวัดความรู้ความจำและความเข้าใจในเนื้อหาที่กว้างขวาง
- 2) ข้อสอบอัตนัย (Subjective Tests) เป็นแบบทดสอบที่เปิดโอกาสให้ผู้เรียนสร้างสรรค์คำตอบของตนเอง เช่น การเขียนบรรยายหรือตอบคำถามปลายเปิด ข้อสอบประเภทนี้เหมาะสมสำหรับการวัดความสามารถในการคิดวิเคราะห์ สังเคราะห์ และประเมินค่า ซึ่งเป็นทักษะการคิดขั้นสูงที่ไม่สามารถวัดได้ด้วยข้อสอบปรนัย

4.3 แนวทางการสร้างข้อสอบที่ดี (Guidelines for Creating Good Tests)

การสร้างแบบทดสอบที่มีคุณภาพต้องอาศัยการวางแผนอย่างเป็นระบบ

- 1) การวางแผน (Test Blueprint) ก่อนสร้างข้อสอบ ควรจัดทำตารางวิเคราะห์หลักสูตร เพื่อกำหนดขอบเขตเนื้อหาที่ต้องการวัดให้มีความสมดุลและครอบคลุมตามวัตถุประสงค์
- 2) หลักการเขียนข้อสอบ
 1. ปรนัย คำถ้าและตัวเลือกต้องมีความซัดเจน ไม่กำกวມ ความมีความน่าสนใจและสมเหตุสมผลสำหรับผู้ที่ยังไม่มีความเข้าใจในเรื่องนั้นอย่างแท้จริง
 2. อัตนัย คำสั่งต้องระบุขอบเขตและประเด็นของคำตอบที่ต้องการอย่างชัดเจน เพื่อให้ผู้เรียนเข้าใจตรงกันว่าจะต้องตอบอะไรบ้าง นอกจากนี้ ควรมีการกำหนดเกณฑ์การให้คะแนนที่ชัดเจนไว้ล่วงหน้าเพื่อลดคอดีในการตรวจ
 - 3) คุณภาพของแบบทดสอบ แบบทดสอบที่ดีต้องมี ความเที่ยงตรง (Validity) คือสามารถวัดในสิ่งที่ต้องการวัดได้จริง และ ความน่าเชื่อถือ (Reliability) คือให้ผลการวัดที่คงเส้นคงวาเมื่อวัดซ้ำ

คุณลักษณะ (Characteristic)	แบบทดสอบปรนัย (Objective Test)	แบบทดสอบอัตนัย (Subjective Test)
การวัดผล	เหมาะสมกับการวัดความรู้ ความจำ และความเข้าใจ	เหมาะสมกับการวัดการคิดวิเคราะห์ สังเคราะห์ และประเมินค่า
ความครอบคลุมเนื้อหา	สามารถถามได้ครอบคลุมเนื้อหาในวงกว้าง	ครอบคลุมเนื้อหาได้จำกัดในเวลาที่เท่ากัน
การตรวจให้คะแนน	ง่าย รวดเร็ว และมีความเป็นกลางสูง	ใช้เวลามาก ซับซ้อน และอาจมีอคติของผู้ตรวจเข้ามาเกี่ยวข้อง
การสร้างข้อสอบ	สร้างได้ยาก ต้องใช้เวลาในการสร้างตัวลงที่ดี	สร้างคำถ้าได้ง่ายและรวดเร็วกว่า
โอกาสในการเดา	ผู้เรียนมีโอกาสเดาคำตอบที่ถูกต้องได้	เดาคำตอบได้ยากมาก

4.4 การประเมินผลหลังการฝึกอบรมและการให้ข้อเสนอแนะ (Post-Training Evaluation and Feedback)

การประเมินผลไม่ควรสิ้นสุดแค่ในห้องเรียน แต่ควรติดตามผลในระยะยาวเพื่อคุ้มครองทบทวน ของการฝึกอบรม โดยทั่วไปนิยมประเมินใน 4 ระดับ ดังนี้

- 1) **ปฏิกิริยา (Reaction)** ประเมินความพึงพอใจของผู้เรียนที่มีต่อหลักสูตร วิทยากร และสิ่งอำนวยความสะดวก

- 2) **การเรียนรู้ (Learning)** วัดระดับความรู้และทักษะที่ผู้เรียนได้รับเพิ่มขึ้น โดยอาจใช้การทดสอบก่อนและหลังการอบรม (Pre-test and Post-test) เพื่อเปรียบเทียบพัฒนาการ
- 3) **พฤติกรรม (Behavior)** ติดตามผลในระยะยาวว่าผู้เรียนได้นำความรู้และทักษะที่เรียนไปประยุกต์ใช้ในการปฏิบัติงานจริงหรือไม่ ซึ่งอาจประเมินผ่านการสังเกตการณ์โดยหัวหน้างานหรือการประเมินผลการปฏิบัติงาน
- 4) **ผลลัพธ์ (Results)** ประเมินผลกระทบของการฝึกอบรมที่มีต่อเป้าหมายขององค์กร เช่น ผลผลิตที่เพิ่มขึ้น ข้อผิดพลาดที่ลดลง หรือความพึงพอใจของลูกค้าที่สูงขึ้น

นอกจากนี้ การให้ข้อเสนอแนะ (Feedback) ที่ดีแก่ผู้เรียนถือเป็นส่วนหนึ่งของการประเมินเพื่อการพัฒนา โดยข้อเสนอแนะควรมีความเฉพาะเจาะจง มุ่งเน้นที่พฤติกรรมที่สามารถปรับปรุงได้ และเสนอแนะแนวทางในการพัฒนาอย่างสร้างสรรค์

หน่วยที่ 5 การพัฒนาตนเองอย่างต่อเนื่อง (Continuous Professional Development)

การเป็นวิทยากรที่มีประสิทธิภาพไม่ใช่จุดหมายปลายทาง แต่เป็นการเดินทางที่ต้องมีการเรียนรู้และพัฒนาอย่างไม่หยุดนิ่ง โดยที่เปลี่ยนแปลงอย่างรวดเร็วทั้งในด้านองค์ความรู้และเทคโนโลยีการสอน ทำให้วิทยากรต้องพร้อมที่จะปรับตัวและพัฒนาตนเองอยู่เสมอเพื่อรักษาความเป็นเลิศในวิชาชีพ

5.1 ความสำคัญของการเป็นผู้เรียนรู้ตลอดชีวิต (The Importance of Lifelong Learning)

แนวคิดการเรียนรู้ตลอดชีวิต (Lifelong Learning) เป็นหัวใจสำคัญของวิทยากรมืออาชีพ เนื่องจาก

- 1) **ความรู้มีการเปลี่ยนแปลง** เนื้อหาวิชาในหลายสาขาวิชามีการพัฒนาและคนพบสิ่งใหม่อยู่เสมอ วิทยากรจึงต้องไฟหามาตรฐานอย่างสม่ำเสมอเพื่อให้มีความทันสมัย ถูกต้อง และน่าเชื่อถือ
- 2) **เทคนิคการสอนมีการพัฒนา** มีการคิดค้นเทคนิคและเทคโนโลยีใหม่ๆ สำหรับการฝึกอบรมอย่างต่อเนื่อง การเรียนรู้สิ่งเหล่านี้จะช่วยให้วิทยากรสามารถสร้างสรรค์ประสบการณ์การเรียนรู้ที่น่าสนใจและมีประสิทธิภาพมากขึ้น
- 3) **การปรับตัวต่อการเปลี่ยนแปลง** วิทยากรต้องพร้อมที่จะลงทะเบียนรู้หรือวิธีการเดิมที่อาจล้าสมัย และเรียนรู้สิ่งใหม่ๆ หรือเรียนรู้สิ่งเดิมในมุมมองใหม่ เพื่อให้สามารถตอบสนองต่อความต้องการของผู้เรียนและองค์กรที่เปลี่ยนแปลงไปได้

5.2 การรับและใช้ประโยชน์จากข้อเสนอแนะ (Receiving and Utilizing Feedback)

ข้อเสนอแนะคือข้อมูลสำคัญสำหรับการพัฒนาตนเอง วิทยากรควรแสวงหาข้อเสนอแนะจากหลากหลายแหล่งเพื่อให้ได้มุมมองที่รอบด้าน

- 1) **จากผู้เรียน** ข้อมูลจากแบบประเมินหลังการอบรมเป็นแหล่งข้อมูลโดยตรงที่สะท้อนมุมมองของผู้เรียนต่อการสอน ซึ่งสามารถให้ข้อมูลเชิงลึกเกี่ยวกับความชัดเจนของเนื้อหา ประสิทธิผลของกิจกรรม และบรรยากาศโดยรวม
- 2) **จากเพื่อนร่วมงานหรือผู้เชี่ยวชาญ** การเชิญวิทยากรท่านอื่นหรือผู้เชี่ยวชาญเข้ามาสังเกตการณ์สอน และให้ข้อเสนอแนะ จะช่วยให้เห็นในสิ่งที่ตนเองอาจมองข้ามไป และได้รับมุมมองจากผู้มีประสบการณ์
- 3) **จากตนเอง** การบทหวานการสอนของตนเอง เช่น การบันทึกวิดีโอแล้วนำกลับมาวิเคราะห์ จะช่วยให้เห็นจุดแข็งและจุดที่ควรปรับปรุงได้อย่างชัดเจน ทั้งในด้านการใช้ภาษา ท่าทาง และการบริหารจัดการเวลา

ทัศนคติในการรับข้อเสนอแนะเป็นสิ่งสำคัญอย่างยิ่ง ต้องเปิดใจรับฟังโดยปราศจากอคติ มองว่าเป็นข้อมูลเพื่อการพัฒนา ไม่ใช่คำวิจารณ์ส่วนบุคคล และนำข้อมูลที่ได้มาไตร่ตรองเพื่อวางแผนการปรับปรุงแก้ไขต่อไป

5.3 การวางแผนพัฒนาตนเอง (Personal Development Planning)

การพัฒนาตนเองอย่างมีทิศทางต้องอาศัยการวางแผนที่เป็นระบบ ซึ่งประกอบด้วยขั้นตอนดังนี้

- 1) **การสำรวจและประเมินตนเอง** วิเคราะห์จุดแข็งและจุดที่ต้องพัฒนาของตนเองอย่างสมำเสมอ ทั้งในด้านความรู้ในเนื้อหา ทักษะการถ่ายทอด และบุคลิกภาพ เพื่อให้ทราบว่าควรจะมุ่งเน้นการพัฒนาในด้านใดเป็นพิเศษ
- 2) **การกำหนดเป้าหมาย** ตั้งเป้าหมายในการพัฒนาที่ชัดเจนและวัดผลได้ เช่น "ภายใน 3 เดือน ข้างหน้า จะพัฒนาทักษะการใช้คำamoto เปิดเพื่อกระตุ้นการอภิปรายให้มากขึ้น" หรือ "จะเข้าร่วมอบรมหลักสูตรการออกแบบสื่อการสอนภายในปีนี้"
- 3) **การเลือกวิธีการพัฒนา** เลือกวิธีการที่เหมาะสมกับเป้าหมาย เช่น การเข้ารับการอบรมในหลักสูตรเฉพาะทาง การอ่านหนังสือและบทความที่เกี่ยวข้อง การเข้าร่วมชุมชนแห่งการเรียนรู้ เพื่อแลกเปลี่ยนประสบการณ์กับวิทยากรท่านอื่น หรือการขอคำปรึกษาจากผู้มีประสบการณ์

การพัฒนาตนเองอย่างต่อเนื่องคือกระบวนการที่ช่วยปิดวงจรของการเป็นวิทยากรที่สมบูรณ์แบบ เชื่อมโยงกลับไปยังคุณสมบัติพื้นฐานที่กล่าวถึงในหน่วยแรกโดยตรง การเป็นผู้เรียนรู้ตลอดชีวิตคือกลไกที่หล่อเลี้ยง "ความรู้จริง" ให้ทันสมัยอยู่เสมอ ในขณะที่การแสวงหาและใช้ประโยชน์จากข้อเสนอแนะคือเครื่องมือสำคัญที่ช่วยขับเคลื่อน "ทักษะการถ่ายทอด" ให้เฉียบคมยิ่งขึ้น ดังนั้น ความเป็นเลิศของวิทยากรจึงไม่ใช่

คุณสมบัติที่หยุดนิ่ง แต่เป็นผลลัพธ์ของการพัฒนาอย่างไม่หยุดยั้ง วิทยากรที่มีประสิทธิภาพไม่ใช่แค่คนที่ "เก่ง" แต่คือคนที่ไม่เคยหยุดที่จะ "เก่งขึ้น"

สรุปสาระสำคัญของบทเรียน

บทเรียนนี้ได้นำเสนอภาพรวมที่ครอบคลุมเกี่ยวกับหลักการของการเป็นวิทยากรที่มีประสิทธิภาพ โดยเริ่มต้นจากการวางแผนที่สำคัญที่สุดคือ คุณลักษณะและจรรยาบรรณของวิทยากร ซึ่งเน้นย้ำว่าความสำเร็จไม่ได้ขึ้นอยู่กับความสามารถในการพูดเพียงอย่างเดียว แต่ต้องประกอบด้วยความรู้จริงในเนื้อหา ทักษะการถ่ายทอดที่ดี และบุคลิกภาพที่ส่งเสริมการเรียนรู้ ทั้งหมดนี้ต้องดำเนินอยู่ภายใต้กรอบจรรยาบรรณที่มุ่งประโยชน์ของผู้เรียนเป็นสำคัญ

จากนั้น ได้ลงรายละเอียดในกระบวนการ การเตรียมความพร้อมก่อนการบรรยาย ซึ่งเป็นขั้นตอนที่บ่งชี้ถึงความเป็นมืออาชีพ ตั้งแต่การวิเคราะห์ผู้เรียนอย่างละเอียดเพื่อออกแบบหลักสูตรให้ตรงจุด การกำหนดวัตถุประสงค์เชิงพฤติกรรมที่ชัดเจนและวัดผลได้ ไปจนถึงการออกแบบโครงสร้างเนื้อหา การเตรียมสื่อ และการซักซ้อมบริหารเวลา ซึ่งทั้งหมดนี้เป็นกระบวนการที่เป็นระบบและเชื่อมโยงกัน

เมื่อเข้าสู่ขั้นตอนการบรรยายจริง บทเรียนได้ชี้ให้เห็นถึงความสำคัญของ เทคนิคการนำเสนอและการสร้างปฏิสัมพันธ์ ซึ่งเป็นศิลปะในการสร้างประสบการณ์การเรียนรู้เชิงบวก โดยอาศัยทักษะการสื่อสารทั้งวิจารณ์และอวاجนภาษา การสร้างบรรยากาศที่ปลอดภัยและเป็นกันเอง และการใช้กลยุทธ์การเรียนรู้เชิงรุก เพื่อกระตุนให้ผู้เรียนมีส่วนร่วมอย่างแข็งขัน

เพื่อให้กระบวนการฝึกอบรมสมบูรณ์ จึงจำเป็นต้องมี การวัดผลและประเมินผลการเรียนรู้ ที่เป็นระบบและสอดคล้องกับวัตถุประสงค์ เพื่อตรวจสอบประสิทธิผลของการสอนและเป็นข้อมูลในการพัฒนาหลักสูตรต่อไปในอนาคต ท้ายที่สุด บทเรียนได้เน้นย้ำว่าการเป็นวิทยากรคือการเดินทางที่ไม่มีวันสิ้นสุดผ่านหัวข้อ การพัฒนาตนเองอย่างต่อเนื่อง ซึ่งเรียกร้องให้วิทยากรเป็นผู้เรียนรู้ตลอดชีวิต เปิดรับข้อเสนอแนะ และวางแผนพัฒนาตนเองอย่างสม่ำเสมอ เพื่อรักษาความเป็นเลิศและปรับตัวให้ทันต่อโลกที่เปลี่ยนแปลงอยู่เสมอ

คำศัพท์ (Glossary)

ก

- การกำกับดูแลข้อมูลสารสนเทศ (Information Governance) หมายถึงการวางแผนนโยบายและแนวปฏิบัติต่าง ๆ ภายในองค์กรในการจัดการและปกป้องข้อมูลอิเล็กทรอนิกส์อย่างเหมาะสม
- การเก็บรักษาข้อมูล (Data Retention) นโยบายการเก็บรักษาข้อมูลจะกำหนดระยะเวลาที่ข้อมูลจะถูกเก็บรักษาไว้ ไม่ว่าจะเป็นข้อมูลในไฟล์ ฐานข้อมูล หรือที่อื่น ๆ
- การกำหนดค่าผิดพลาดด้านความมั่นคงปลอดภัย (Security Misconfiguration) รวมถึงการใช้ค่าตั้งต้นที่ไม่ปลอดภัย การไม่อัปเดตซอฟต์แวร์ หรือการเปิดใช้ฟีเจอร์โดยไม่จำเป็น ซึ่งอาจเปิดช่องให้ผู้โจมตีเข้าถึงระบบ
- การเก็บรวบรวมข้อมูล (Collection) เป็นกระบวนการรวบรวมข้อมูลอิเล็กทรอนิกส์ที่ได้รับการส่งเสริมจากแหล่งต่าง ๆ ทั้งที่อยู่บนระบบออนไลน์ อุปกรณ์ทางกายภาพ หรืออุปกรณ์ส่วนกลาง
- การกำหนดสิทธิ์ (Authorization) คือการอนุญาตให้ผู้ใช้เข้าถึงทรัพยากรหลังจากผ่านการตรวจสอบตัวตนเรียบร้อยแล้ว
- การเขียนสคริปต์ (Scripting) เป็นกระบวนการสร้างคำสั่งที่สามารถรันแบบอัตโนมัติในรูปแบบที่กำหนดเวลาไว้ล่วงหน้า (Scheduled) และทำซ้ำได้อย่างสม่ำเสมอ (Repeatable)
- การเข้ารหัสข้อมูล (Encryption) ข้อมูลส่วนบุคคลจะต้องถูกเข้ารหัสเพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต ทั้งในขณะที่ข้อมูลถูกจัดเก็บ (Data at Rest) และในขณะที่ข้อมูลกำลังถูกส่งผ่านเครือข่าย (Data in Transit)
- การเข้าถึงผ่านเครือข่ายที่หลากหลาย (Broad Network Access) ผู้ใช้งานสามารถเข้าถึงบริการคลาวด์ผ่านเครือข่ายอินเทอร์เน็ตได้จากอุปกรณ์ต่าง ๆ ที่รองรับ โดยไม่จำกัดสถานที่และเวลา
- การควบคุมเวอร์ชันของซอฟต์แวร์ (Source Control) การจัดการโค๊ดที่จำเป็นต้องมีระบบติดตามและระบุว่าโค๊ดเวอร์ชันใดคือเวอร์ชันที่เป็นปัจจุบันหรือ “เวอร์ชันที่เชื่อถือได้” (Authoritative Version)
- การจัดเก็บข้อมูลแบบแบ่งระดับ (Tiered Storage) เป็นแนวทางที่ผู้ดูแลระบบใช้เพื่อปรับสมดุลระหว่างต้นทุนและประสิทธิภาพ โดยพิจารณาจากความต้องการเข้าถึงข้อมูลและระยะเวลาในการจัดเก็บข้อมูล
- การจัดประเภทข้อมูล (Data Classification) เป็นการระบุความสำคัญและผลกระทบของข้อมูลในกรณีที่เกิดการรั่วไหล

- **การแจ้งเตือนเหตุการณ์ (Incident Alerts)** บริหารจัดการผ่านระบบ Cloud Detection and Response (CDR) ซึ่งทำหน้าที่รวบรวมและวิเคราะห์ข้อมูลจากแหล่งต่าง ๆ เพื่อให้ผู้ดูแลระบบมีมุมมองแบบองค์รวมของทรัพยากรบนคลาวด์
- **การทดสอบการกู้คืนข้อมูล (Recovery Testing)** การทดสอบระบบสำรองและกู้คืนข้อมูลการทำอย่างสม่ำเสมอ เพื่อให้มั่นใจว่าข้อมูลถูกสำรองไว้อย่างถูกต้องและสามารถกู้คืนได้อย่างมีประสิทธิภาพ
- **การบูรณาการขั้นสูง (Hyperconverged)** คือ การรวมทรัพยากรคอมพิวเตอร์ พื้นที่จัดเก็บข้อมูล และเครือข่าย เข้าด้วยกันเป็นองค์ประกอบเดียว โดยมีเป้าหมายเพื่อลดความซับซ้อนในการจัดการและเพิ่มความสามารถในการขยายระบบ
- **การแบ่งเครือข่ายและชั้นเน็ต (Network Segmentation and Subnetting)** การแบ่งเครือข่าย (Segmentation) ช่วยเพิ่มความปลอดภัยของข้อมูลในเครือข่ายโดยแยกการรับส่งข้อมูลให้อยู่ภายในส่วนต่าง ๆ ของเครือข่ายแน่นอน
- **การปรับใช้ทรัพยากรระบบคลาวด์ (Deploy Cloud Resources)** การจัดเตรียมทรัพยากรระบบคลาวด์สามารถดำเนินการได้ตามความต้องการ (On-Demand Provisioning) และยังสามารถปรับขนาดของทรัพยากรให้เพิ่มขึ้นหรือลดลงตามความต้องการได้ในกระบวนการที่เรียกว่า การปรับขนาดอัตโนมัติ (Auto-Scaling)
- **การปรับขนาดอัตโนมัติ (Auto-Scaling)** เป็นการใช้ประโยชน์จากการปรับใช้งานแบบอัตโนมัติและการจำลองเสมือน (Virtualization) เพื่อให้ทรัพยากรตรงกับความต้องการใช้งานในแต่ละช่วงเวลา
- **การปรับขนาดในแนวอน (Horizontal Scaling)** หรือที่เรียกว่า Scaling Out เป็นการเพิ่มจำนวนเซิร์ฟเวอร์ (ทั้งจริงและเสมือน) เพื่อรับรองรับภาระงาน
- **การปรับขนาดในแนวตั้ง (Vertical Scaling)** หรือที่เรียกว่า Scaling Up เป็นการเพิ่มความสามารถในการประมวลผลของเครื่องเซิร์ฟเวอร์ที่มีอยู่ เช่น เพิ่มน้ำหนักความจำ
- **การผ่อนรวมอย่างต่อเนื่องและการปรับใช้แบบต่อเนื่อง (Continuous Integration / Continuous Deployment Pipelines)** การทำงานอัตโนมัติและการจัดการระบบแบบออร์คีสเตรต (orchestration) สำหรับระบบคลาวด์ทั้งแบบส่วนตัว (Private Cloud) และสาธารณะ (Public Cloud) มีบทบาทสำคัญในการสนับสนุนการปรับใช้อินสแตนซ์ คอนเทนเนอร์ และแอปพลิเคชันให้สามารถดำเนินการได้อย่างรวดเร็วและสม่ำเสมอ
- **การพิสูจน์ตัวตน (Authentication)** เป็นกระบวนการที่ระบบใช้เพื่อยืนยันตัวตนของผู้ใช้งานก่อนให้สิทธิ์เข้าถึงทรัพยากร
- **การพิสูจน์ตัวตนแบบใช้กุญแจใน Secure Shell (SSH Key-Based Authentication)** วิธีมาตรฐานในการพิสูจน์ตัวตนต่อเซิร์ฟเวอร์ SSH ในปัจจุบันมักใช้ การพิสูจน์ตัวตนด้วยกุญแจ (Key-Based Authentication) แทนการใช้รหัสผ่านแบบดั้งเดิม

- การย้ายจากคลาวด์หนึ่งไปยังอีกคลาวด์หนึ่ง (Cloud-to-Cloud Migration) เกิดขึ้นเมื่องค์กรต้องการรวมบริการคลาวด์ที่เคยใช้จากหลายผู้ให้บริการให้เหลือเพียงรายเดียว หรือเมื่องค์กรต้องการย้ายออกจากผู้ให้บริการเดิม
- การย้ายจากคลาวด์กลับไปยังระบบภายในองค์กร (Cloud-to-On-Premises) ในบางกรณี องค์กรอาจตัดสินใจย้ายข้อมูลและบริการกลับจากคลาวด์มายังระบบภายในองค์กร
- การย้ายจากระบบภายในองค์กรชั้นคลาวด์ (On-Premises-to-Cloud Migration) เป็นรูปแบบที่พบบ่อยที่สุดในการย้ายระบบ โดยเป็นการโอนย้ายทรัพยากรจากระบบที่ติดตั้งภายในองค์กรไปยังทรัพยากรที่จัดสรรจากผู้ให้บริการคลาวด์ (Cloud Service Provider: CSP)
- การย้ายฐานข้อมูล (Migrate Databases) การย้ายฐานข้อมูลถือเป็นความท้าทายที่แตกต่างจากการย้ายระบบเสมือน (Virtualized Systems) หรือข้อมูลดิบทั่วไป เนื่องจากฐานข้อมูลมักมีความเชื่อมโยงกับการให้บริการที่ต่อเนื่องและมีความซับซ้อนสูง
- การรวมทรัพยากรเพื่อใช้งานร่วมกัน (Resource Pooling) ผู้ให้บริการระบบคลาวด์จะจัดสรรทรัพยากรต่าง ๆ เช่น พลังประมวลผล พื้นที่จัดเก็บข้อมูล และระบบเครือข่าย ให้กับผู้ใช้งานหลายรายผ่านการรวมทรัพยากรไว้ในระบบคลาสสิก (Multi-tenant Model)
- การวัดผลและคิดค่าบริการตามการใช้งานจริง (Measured Service) ทรัพยากรที่ใช้งานจะถูกตรวจสอบและวัดผลอย่างต่อเนื่อง ทำให้สามารถควบคุม ตรวจสอบ และเรียกเก็บค่าบริการตามปริมาณการใช้งานจริงได้อย่างแม่นยำ
- การเรียนรู้ของเครื่อง (Machine Learning: ML) คือ สาขาย่อยของ AI ที่มุ่งเน้นให้ระบบสามารถเรียนรู้จากข้อมูลและประสบการณ์ เพื่อนำไปใช้ในการทำงานอย่างผลลัพธ์โดยไม่ต้องตั้งโปรแกรมอย่างชัดเจนให้ผลลัพธ์ใดๆ
- การเรียนรู้เชิงลึก (Deep Learning: DL) คือสาขาย่อยที่ลึกกว่าของ ML ซึ่งเลียนแบบกระบวนการเรียนรู้ของสมองมนุษย์ โดยสามารถวิเคราะห์ข้อมูลที่ไม่มีโครงสร้างได้อย่างแม่นยำมากขึ้น
- การออกแบบแบบคลาวด์เนทีฟ (Cloud-native Design) หมายถึง การพัฒนาแอปพลิเคชันโดยอาศัยเทคโนโลยีของระบบคลาวด์เป็นหลัก เพื่อเพิ่มความยืดหยุ่นและประสิทธิภาพในการใช้งาน
- เกตเวย์ทранซิตในระบบคลาวด์ (Transit Gateway in the Cloud) ทำหน้าที่เป็นศูนย์กลางในการจัดการและควบคุมการรับส่งข้อมูลและบริการระหว่างพื้นที่เครือข่ายต่าง ๆ

ค

- คลาวด์ (Cloud) หรือ “การประมวลผลแบบคลาวด์” (Cloud Computing) มีคุณลักษณะที่สำคัญ 5 ประการตามนิยามของ NIST ได้แก่ การให้บริการด้วยตนเองตามต้องการ การเข้าถึงได้ผ่าน

เครื่องข่ายที่หลากหลาย การจัดสรรทรัพยากรแบบรวมศูนย์ ความยืดหยุ่นและการปรับขนาดทรัพยากรอย่างรวดเร็ว และการวัดผลและคิดค่าบริการตามการใช้งาน

- **คลาวด์ชุมชน (Community Cloud)** เป็นรูปแบบของระบบคลาวด์ที่ถูกออกแบบมาเพื่อให้บริการเฉพาะกลุ่มขององค์กรที่มีลักษณะการดำเนินธุรกิจหรือข้อกำหนดด้านความปลอดภัยที่คล้ายคลึงกัน
- **คลาวด์แบบผสม (Hybrid Cloud)** คือการผสมผสานรูปแบบการให้บริการคลาวด์ทั้งแบบสาธารณะ (Public Cloud) ส่วนตัว (Private Cloud) และ/หรือคลาวด์ชุมชน (Community Cloud) เข้าด้วยกัน
- **คลาวด์สาธารณะ (Public Cloud)** เป็นบริการคลาวด์ที่องค์กรต่าง ๆ ซึ่งไม่เกี่ยวข้องกันสามารถเข้ามาใช้งานร่วมกันได้
- **คลาวด์ส่วนตัว (Private Cloud)** เป็นรูปแบบการให้บริการคลาวด์ที่จำกัดการใช้งานเฉพาะองค์กรที่เป็นเจ้าของระบบเท่านั้น
- **คลาวด์ส่วนตัวเสมือน (Virtual Private Cloud: VPC)** คือ บริการที่ผู้ให้บริการคลาวด์ไว้ในพื้นที่เฉพาะที่แยกออกจากทรัพยากรที่องค์กรอื่นใช้งานร่วมกัน
- **คอนเทนเนอร์ (Containers)** เป็นรูปแบบหนึ่งของการจำลองเสมือนระบบปฏิบัติการ (Virtualized Operating System) เป็นชุดซอฟต์แวร์แบบสมบูรณ์และสามารถพกพาได้ ซึ่งรวมถึงโค้ดของแอปพลิเคชัน รันไทม์ ไลบรารี การตั้งค่าต่าง ๆ และองค์ประกอบอื่น ๆ ที่จำเป็นต่อการทำงานของซอฟต์แวร์ทั้งหมดไว้ในแพ็คเกจเดียว
- **ความยืดหยุ่นและการปรับขนาดอย่างรวดเร็ว (Rapid Elasticity)** ระบบคลาวด์สามารถขยายหรือลดขนาดของทรัพยากรได้อย่างรวดเร็วและทันต่อความต้องการ
- **โครงสร้างพื้นฐานเป็นบริการ (Infrastructure as a Service: IaaS)** คือรูปแบบการให้บริการที่ช่วยให้องค์กรสามารถถ่ายโอนภาระด้านการดูแล硬件ตัวจริงไปยังผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) ได้อย่างสมบูรณ์ โดยผู้ใช้งานจะได้รับทรัพยากรพื้นฐานในรูปแบบของเครื่องเสมือน (Virtual Machines) ที่โฮสต์อยู่บนโครงสร้างพื้นฐานของผู้ให้บริการ

จ

- **เจสัน (JSON - JavaScript Object Notation)** เป็นรูปแบบข้อมูล (text-based) สำหรับการส่งและจัดเก็บข้อมูล มีความเข้าใจง่ายและสามารถอ่านได้โดยตรง

ข

- **แชทบอท (Chatbot)** เป็นการประยุกต์ใช้ Generative AI ที่สามารถสร้างเพลง เขียนเรื่องราว หรือ สร้างภาพได้หากมีข้อมูลเพียงพอ

ข

- **ซอฟต์แวร์เป็นบริการ (Software as a Service: SaaS)** ผู้ใช้สามารถเข้าถึงและใช้งานซอฟต์แวร์ ได้โดยตรงผ่านเครือข่าย โดยไม่ต้องติดตั้งหรือดูแลระบบเอง

ด

- **ดีฟ เลิร์นนิ่ง (Deep Learning)** คือสาขาย่อยที่ลึกกว่าของ ML ซึ่งเลียนแบบกระบวนการเรียนรู้ของ สมองมนุษย์ โดยสามารถวิเคราะห์ข้อมูลที่ไม่มีโครงสร้างได้อย่างแม่นยำมากขึ้น และใช้เวลาและ ข้อมูลในการฝึกมากกว่า แต่ผลลัพธ์มีความแม่นยำสูงขึ้น

ต

- **ตัวกระจายโหลด (Load Balancers)** มีหน้าที่จัดการปริมาณการรับส่งข้อมูลที่เข้าสู่อินเทอร์เน็ตของ เครื่องเสมือน (VM) และบริการอื่น ๆ

ท

- **เทคโนโลยีเวอร์ชวลайเซชัน (Virtualization Technology)** เป็นเทคโนโลยีที่ช่วยแยก ระบบปฏิบัติการ บริการ และแอปพลิเคชัน ออกจากขอจำกัดของฮาร์ดแวร์จริง

ธ

- **ธรรมาภิบาลข้อมูล (Data Governance)** เป็นการสร้างกรอบและแนวปฏิบัติที่ชัดเจนในการกำกับ ดูแลข้อมูล

บ

- **บริการแบบสมัครสมาชิก (Subscription Services)** เป็นรูปแบบการเรียกเก็บค่าบริการ แบบต่อเนื่องตามรอบระยะเวลาที่กำหนด เช่น รายเดือนหรือรายปี

ป

- **ปัญญาประดิษฐ์ (Artificial Intelligence: AI)** คือศาสตร์ที่มุ่งเน้นการจำลองความสามารถของมนุษย์ในการใช้สติปัญญา โดยการประมวลผลข้อมูลทั้งที่มีโครงสร้าง กึ่งมีโครงสร้าง และไม่มีโครงสร้าง เพื่อนำมาใช้ในการแก้ไขปัญหาที่ซับซ้อน
- **ปัญญาประดิษฐ์เชิงสร้างสรรค์ (Generative AI)** สร้างเนื้อหาต่างๆ เช่น ข้อความ โค้ดคอมพิวเตอร์ หรือภาพจากคำสั่ง (prompt) โดยมักใช้การเรียนรู้เชิงลึก (Deep Learning: DL)

ผ

- **ผู้ให้บริการคลาวด์แบบมีการจัดการ (Managed Service Providers: MSPs)** เป็นบุคคลหรือนิติบุคคลอิสระที่ไม่เข้าลงกับผู้ให้บริการคลาวด์รายใหญ่ (Cloud Service Providers: CSPs) ซึ่งองค์กรสามารถว่าจ้างให้ช่วยดำเนินงานด้านต่าง ๆ เช่น การออกแบบระบบคลาวด์ การโยกย้ายระบบ (Migration) การติดตั้งระบบ (Deployment) ตลอดจนการบริหารจัดการและดูแลระบบคลาวด์อย่างต่อเนื่อง
- **ผู้ให้บริการระบบคลาวด์ (Cloud Service Providers: CSPs)** คือองค์กรที่ให้บริการทรัพยากรคอมพิวเตอร์ผ่านระบบคลาวด์ในรูปแบบการสมัครใช้งาน (Subscription-Based)

ผ

- **ผู้เชิร์ฟเวอร์ (Server-side)** การเข้ารหัสข้อมูลผู้เชิร์ฟเวอร์ (Server-Side Encryption: SSE) เป็นทางเลือกสำหรับการเข้ารหัสข้อมูลในระบบคลาวด์

พ

- **แพลตฟอร์มเป็นบริการ (Platform as a Service: PaaS)** ผู้ให้บริการจะจัดเตรียมโครงสร้างพื้นฐานและสภาพแวดล้อมสำหรับการพัฒนาแอปพลิเคชัน เช่น เครื่องมือพัฒนา ระบบฐานข้อมูล และ Web Server ในขณะที่ผู้ใช้งานมีหน้าที่จัดการโค้ดและเนื้อหาภายในแพลตฟอร์ม

พ

- **ฟังก์ชันเป็นบริการ (Function as a Service: FaaS)** เป็นรูปแบบที่นักพัฒนาสามารถเขียนและรันโค้ดได้โดยไม่ต้องดูแลระบบโครงสร้างพื้นฐานใด ๆ เป็นอย่างหลัง
- **ไฟร์wall (Firewall)** ทำหน้าที่ควบคุมและการกรองการเข้าถึงเครือข่าย โดยอาศัยกฎเกณฑ์ที่กำหนดไว้ (Rule-based Access) เพื่อป้องกันการเข้าถึงที่ไม่ได้รับอนุญาตไปยังทรัพยากรในระบบคลาวด์

- ไฟร์วอลล์สำหรับเว็บแอปพลิเคชัน (Web Application Firewalls – WAF) ทำงานที่ระดับเลเยอร์ 7 (Application Layer) เพื่อป้องกันช่องโหว่ที่เกิดขึ้นในระดับแอปพลิเคชัน

ก

- ภาพรวมกฎหมายดิจิทัลของไทยที่เกี่ยวข้องกับระบบคลาวด์ (Overview of Thai Digital Laws Relevant to Cloud Computing) การนำบริการคลาวด์มาใช้ในองค์กรไทยไม่ได้เป็นเพียงการตัดสินใจทางเทคโนโลยี แต่ยังเป็นประเด็นด้านการกำกับดูแล ความเสี่ยง และการปฏิบัติตามกฎหมาย (Governance Risk and Compliance: GRC) ที่มีความสำคัญอย่างยิ่ง

ม

- มัลติคลาวด์ (Multi-Cloud) คือการที่องค์กรเลือกใช้งานบริการคลาวด์จากผู้ให้บริการมากกว่าหนึ่งรายพร้อมกัน
- ไมโครเซ็กเมนเทชัน (Microsegmentation) ทำการแบ่งในระดับเวิร์กโหลด (Workload-level) แยกแอปพลิเคชันและโครงสร้างพื้นฐานที่เกี่ยวข้องออกจากกันแบบละเอียดมากขึ้น
- ไมโครเซอร์วิส (Microservices) คือองค์ประกอบย่อยอิสระที่ทำหน้าที่เฉพาะเจาะจงแต่ละด้าน โดยมีลักษณะเด่นคือ มีขนาดเล็ก แยกจากกันโดยสมบูรณ์ ทำหน้าที่เดียว เป็นระบบปิดในตัวเอง และสามารถขยายขนาดได้ง่าย
- แม่แบบ (Templates) คือการกำหนดค่าล่วงหน้าสำหรับเครื่องเสมือนที่มีการออกแบบมาอย่างเหมาะสม ทั้งในด้านของหน่วยประมวลผล พื้นที่จัดเก็บ ระบบปฏิบัติการ ซอฟต์แวร์ที่จำเป็น และการตั้งค่าด้านความปลอดภัย

ย

- yaml (YAML - Yet Another Markup Language) เป็นภาษาที่ใช้ใน Playbook ของ Ansible และแหล่งกำหนดค่าต่าง ๆ เพราะมีโครงสร้างที่อ่านง่ายและเข้าใจง่าย

ร

- ระเบียบวิธีการแก้ไขปัญหา (Troubleshooting Methodology) เป็นกระบวนการแก้ไขปัญหาในระบบคลาวด์ที่สามารถดำเนินการตามขั้นตอนพื้นฐาน

ว

- เวอร์ชัลไลเซชัน (Virtualization) เป็นเทคโนโลยีที่ช่วยแยกระบบปฏิบัติการ บริการ และแอปพลิเคชัน ออกจากข้อจำกัดของฮาร์ดแวร์จริง

ส

- สถาปัตยกรรมความมั่นคงปลอดภัยแบบ Zero Trust (Zero Trust Cloud Security) เป็นแนวคิดที่ถือว่าทุกระบบหรืออุปกรณ์จะต้อง “ไม่ถูกไว้วางใจโดยอัตโนมัติ” และทุกรายการร้องขอเข้าถึงต้องผ่านการยืนยันและกำหนดสิทธิ์อย่างชัดเจนก่อนเสมอ
- สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ (Event-Driven Architectures) คือ แนวทางการออกแบบที่กำหนดให้มี ตัวกระตุ้น (Trigger) ที่ชัดเจนเพื่อเริ่มต้นกระบวนการสื่อสารระหว่างบริการ หรือแอปพลิเคชันที่แยกออกจากกัน
- สภาพแวดล้อมแบบไร้เซิร์ฟเวอร์ (Serverless Environments) คือแนวทางการพัฒนาแอปพลิเคชันที่ยกเลิกความจำเป็นในการจัดการฮาร์ดแวร์ ระบบปฏิบัติการ และการบำรุงรักษาบริการของนักพัฒนาโดยตรง

อ

- อินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) หมายถึง การผสมผสานระหว่างอุปกรณ์อัจฉริยะและการเชื่อมต่อเครือข่ายที่ทำให้อุปกรณ์เหล่านั้นสามารถตรวจสอบและวิเคราะห์ข้อมูลได้อย่างมีประสิทธิภาพ
- เอปีไอ (API - Application Programming Interface) คือ ช่องทางในการสื่อสารระหว่างบริการ หรือแอปพลิเคชันผ่าน HTTP ช่วยให้สามารถติดข้อมูลหรือกระทำการกับทรัพยากรได้จากภายนอกระบบ
- ออร์คेसเตอเรชัน (Orchestration) คือ การรวมกระบวนการที่เป็น Automation หลายชั้นตอนเข้าไว้ด้วยกันเป็นลำดับขั้นตอนเดียว ซึ่งสามารถทำงานต่อเนื่องได้โดยอัตโนมัติ

ย

- ไฮเปอร์ไวเซอร์ (Hypervisor) เป็นซอฟต์แวร์ที่ทำหน้าที่จัดการทรัพยากรฮาร์ดแวร์ให้ VM ใช้งาน และควบคุมการเข้าถึงฮาร์ดแวร์ของแต่ละ VM

A

- **Ansible** เป็นเครื่องมือจัดการการกำหนดค่าระบบและการจัดลำดับขั้นตอนแบบอัตโนมัติ (Orchestration) ซึ่งมีจุดเด่นที่สามารถทำงานได้โดยไม่ต้องติดตั้ง Agent ลงในเครื่องเป้าหมาย (Agentless)
- **Application Programming Interface (API)** คือ ช่องทางในการสื่อสารระหว่างบริการหรือแอปพลิเคชันผ่าน HTTP ช่วยให้สามารถตีดึงข้อมูลหรือกระทำการกับทรัพยากรได้จากภายนอกระบบ

B

- **Border Gateway Protocol (BGP)** เป็นโปรโตคอลที่กำหนดการกำหนดค่าเราเตอร์แบบไดนามิก เพื่อแลกเปลี่ยนข้อมูลเกี่ยวกับเส้นทางที่สามารถรองรับได้

C

- **Cloud Bursting** คือการใช้ทรัพยากรของระบบคลาวด์สาธารณะชั่วคราว เมื่อระบบคลาวด์ภายใน (Private Cloud) มีการใช้งานสูงเกินขีดจำกัด
- **Content Delivery Networks (CDNs)** เป็นโครงสร้างเครือข่ายแบบกระจาย (Distributed Network) ที่มีวัตถุประสงค์เพื่อเพิ่มความพร้อมใช้งาน (Availability) และประสิทธิภาพ (Performance) โดยใช้การแคช (Cache) เนื้อหาหรือบริการให้อยู่ใกล้กับผู้ใช้งานมากที่สุด

D

- **DevOps** เป็นแนวทางปฏิบัติที่ผสมรวมการพัฒนาซอฟต์แวร์ (Development) และการดำเนินงานด้านไอที (Operations) เข้าด้วยกัน
- **Docker** เป็นเอนจินหลักที่ใช้ในการจัดการคอนเทนเนอร์ โดยมีความสามารถในการสร้าง รัน และจัดการวงจรชีวิตของคอนเทนเนอร์ได้อย่างมีประสิทธิภาพ
- **Dynamic Host Configuration Protocol (DHCP)** อุปกรณ์ทุกเครื่องในเครือข่ายต้องมีการตั้งค่าที่อยู่ IP ซึ่งสามารถตั้งค่าได้แบบแม่นวนล็อโดยผู้ดูแลระบบ หรือแบบอัตโนมัติโดยบริการ DHCP

E

- **Edge Computing** คือแนวทางที่ให้การประมวลผล (หรือเตรียมข้อมูลก่อนประมวลผล) เกิดขึ้นใกล้กับอุปกรณ์ IoT แทนที่จะส่งข้อมูลไปยังคลาวด์โดยตรง

- **ELK Stack (Elasticsearch Logstash และ Kibana)** เป็นระบบที่ประกอบด้วย 3 ส่วนหลัก ซึ่งทำงานร่วมกันเพื่อรับรวม วิเคราะห์ และแสดงผลข้อมูลจากไฟล์บันทึกเหตุการณ์ (Log Files) ของระบบ

F

- **Function as a Service (FaaS)** เป็นรูปแบบที่นักพัฒนาสามารถเขียนและรันโค้ดได้โดยไม่ต้องดูและระบบโครงสร้างพื้นฐานใด ๆ เป็นหลัง

G

- **Git** เป็นบริการควบคุมเวอร์ชัน (Version Control Service) ที่ช่วยติดตามการเปลี่ยนแปลงของไฟล์ และจัดการเวอร์ชัน
- **GitHub** คือ บริการไฮสต์โครงสร้างที่ใช้ระบบควบคุมเวอร์ชัน Git โดยนักพัฒนาสามารถแบ่งปันโค้ด ทั้งภายในและภายนอกองค์กร

H

- **Hyper-V** เป็น Type 1 Hypervisor ของ Microsoft ที่ทำงานโดยตรงบนฮาร์ดแวร์ของเซิร์ฟเวอร์ โดยไม่ต้องมีระบบปฏิบัติการกลาง

I

- **Infrastructure as Code (IaC)** เป็นแนวทางที่ผู้ดูแลระบบกำหนดค่าที่ต้องการในรูปแบบโค้ด แล้วนำโค้ดนั้นไปใช้กับอุปกรณ์โครงสร้างพื้นฐานต่าง ๆ
- **Infrastructure as a Service (IaaS)** คือรูปแบบการให้บริการที่ช่วยให้องค์กรสามารถถ่ายโอนภาระด้านการดูแลฮาร์ดแวร์ไปยังผู้ให้บริการระบบคลาวด์ (Cloud Service Provider: CSP) ได้อย่างสมบูรณ์ โดยผู้ใช้งานจะได้รับทรัพยากรพื้นฐานในรูปแบบของเครื่องเสมือน (Virtual Machines) ที่โฮสต์อยู่บนโครงสร้างพื้นฐานของผู้ให้บริการ

K

- **Kubernetes** เป็นเครื่องมือโอเพนซอร์สยอดนิยมที่ใช้สำหรับการจัดลำดับขั้นตอน (orchestration) ของคอนเทนเนอร์

M

- **Multitenancy** เป็นพื้นฐานของการให้บริการคลาวด์แบบสาธารณณะ โดยหมายถึงการที่ผู้ใช้งานหลายคนราย (เรียกว่า ผู้เช่าเช่นเดียวกัน หรือ Tenants) ใช้ทรัพยากรร่วมกัน

N

- **Network Address Translation (NAT)** เป็นเทคนิคที่ใช้เพื่อเพิ่มประสิทธิภาพ การตรวจสอบ และความปลอดภัยของเครือข่าย โดยหลักการคือ การแปลงที่อยู่ IP ภายในให้เข้ามายังที่อยู่ IP สาธารณะ
- **Network Time Protocol (NTP)** ถูกใช้เพื่อให้เกิดการซิงโครไนซ์เวลาระหว่างอุปกรณ์ต่าง ๆ บน เครือข่าย เช่น เครื่องลูกข่าย (Client) และเครื่องแม่ข่าย (Server)

O

- **Observability** เป็นกระบวนการที่ช่วยให้นักพัฒนา ผู้ดูแลระบบ รวมถึงผู้ใช้งาน สามารถเรียนรู้และเข้าใจการทำงานของบริการและแอปพลิเคชันบนระบบคลาวด์ได้อย่างลึกซึ้ง
- **OpenID Connect** คล้ายกับ SAML แต่พัฒนาอยู่บนพื้นฐานของ OAuth 2.0 นักใช้ในระบบเว็บ สมัยใหม่ เช่น การเข้าสู่ระบบด้วย Google หรือ Facebook

P

- **Platform as a Service (PaaS)** ผู้ให้บริการจะจัดเตรียมโครงสร้างพื้นฐานและสภาพแวดล้อม สำหรับการพัฒนาแอปพลิเคชัน เช่น เครื่องมือพัฒนา ระบบฐานข้อมูล และ Web Server ในขณะที่ผู้ใช้งานมีหน้าที่จัดการโค้ดและเนื้อหาภายในแพลตฟอร์ม

R

- **Remote Desktop Protocol (RDP)** ระบบปฏิบัติการ Microsoft Windows ใช้ RDP ในการ เข้าถึงเครื่องเสมือน (VM) ที่อยู่บนคลาวด์
- **Rightsizing** คือการปรับความสามารถของเครื่องเสมือน (VM) ให้เหมาะสมกับปริมาณงาน โดยมี เป้าหมายเพื่อไม่จำสูตรทรัพยากรมากเกินไป (ซึ่งสิ้นเปลือง) หรือ น้อยเกินไป (ซึ่งกระทบต่อ ประสิทธิภาพ)

S

- **Security Assertion Markup Language (SAML)** เป็นโปรโตคอลที่ใช้ในการแลกเปลี่ยนข้อมูลการพิสูจน์ตัวตนระหว่าง IdP และ Service Provider
- **Secure Shell (SSH)** คือ เครื่องมือสำหรับการจัดการระบบ Linux และอุปกรณ์เครือข่ายอื่น ๆ ผ่านการเข้ามายังตัวตนและเข้ารหัสข้อมูลอย่างปลอดภัย
- **Software as a Service (SaaS)** ผู้ใช้สามารถเข้าถึงและใช้งานซอฟต์แวร์ได้โดยตรงผ่านเครือข่าย โดยไม่ต้องติดตั้งหรือดูแลระบบเอง
- **Software-defined Networking (SDN)** แยกการจัดการออกเป็น 2 ส่วน: Data Plane (ควบคุมการส่งข้อมูล) และ Control Plane (จัดการคำสั่งควบคุมและนโยบายผ่าน software จากส่วนกลาง)
- **Software-defined Storage (SDS)** เป็นองค์ประกอบสำคัญของระบบจัดเก็บข้อมูลระดับองค์กรโดยแยกระบบจัดการออกจากฮาร์ดแวร์ที่ใช้จัดเก็บข้อมูล

T

- **Transport Layer Security (TLS)** ใช้การเข้ารหัสแบบสมมาตร โดยใช้ทั้งการเข้ารหัสแบบสมมาตร (Symmetric) และอสมมาตร (Asymmetric)

V

- **Virtual LAN (VLAN)** ช่วยจัดข้อมูลของเครือข่ายทางกายภาพ โดยเพิ่มพอร์ตของสวิตช์ใน Layer 2 เช้าไปยัง VLAN ที่กำหนด และใช้กูซอฟต์แวร์เพื่อแยกการรับส่งข้อมูลเฉพาะพอร์ตที่อยู่ใน VLAN เดียวกัน
- **Virtual Private Cloud (VPC)** คือ บริการที่ผู้ให้บริการคลาวด์สาธารณะ (Cloud Service Provider – CSP) จัดสรรทรัพยากรคลาวด์ไว้ในพื้นที่เฉพาะที่แยกออกจากทรัพยากรที่องค์กรอื่นใช้งานร่วมกัน
- **Virtual Private Network (VPN)** เป็นโซลูชันที่พับได้ทั่วไปในระบบเครือข่ายปัจจุบัน โดยมีสถาปัตยกรรมหลักอยู่สองรูปแบบ ได้แก่ VPN แบบไซต์ต่อไซต์ (Site-to-Site VPN) และ VPN แบบจุดต่อจุด (Point-to-Site VPN)

X

- **Anything as a Service (XaaS)** เป็นคำรวมที่ใช้อธิบายบริการระบบคลาวด์รูปแบบต่าง ๆ ที่ขยายออกไปนอกเหนือจากรูปแบบหลัก เช่น ฐานข้อมูลเป็นบริการ (Database as a Service: DBaaS)

เดสก์ท็อปเป็นบริการ (Desktop as a Service: DaaS) และคอนเทนเนอร์เป็นบริการ (Containers as a Service: CaaS)

Y

- **YAML (Yet Another Markup Language)** เป็นภาษาที่ใช้ใน Playbook ของ Ansible และ
แหล่งกำเนิดค่าต่าง ๆ เพราะมีโครงสร้างที่อ่านง่ายและเข้าใจง่าย