# กระบวนติดตั้งและใช้งาน Openstack เบื้องต้น

# How to install and use Openstack on basic

นัฐวัฒน์ ธงชัยธนาวุฒิ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

ปริญญานิพนธ์ปีการศึกษา 2559

ภาควิชาวิศวกรรมคอมพิวเตอร์

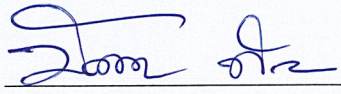
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง กระบวนติดตั้งและใช้งาน Openstack เบื้องต้น

How to install and use Openstack on basic

ผู้จัดทำ

1. นายนัฐวัฒน์ ธงชัยธนาวุฒิ รหัสนักศึกษา 56010664



อาจารย์ที่ปรึกษา

(อาจารย์ บัณฑิต พัสยา)

# กระบวนติดตั้งและใช้งาน Openstack เบื้องต้น

นายนัฐวัฒน์ ธงชัยธนาวุฒิ 56010664
อาจารย์บัณฑิต พัสยา อาจารย์ที่ปรึกษา
ปีการศึกษา 2559

## บทคัดย่อ

ปัจจุบัน Openstack เป็นที่นิยมและใช้งานอย่างแพร่หลายในหลายๆบริษัทโดยที่ OpenStack คือ Cloud Operating System แบบ Open Source ทำหน้าที่บริหารจัดการทรัพยากรด้าน Compute, Storage, Networking ของ Data Center ให้มีประสิทธิภาพในการทำงานสูงที่สุด แบ่งการทำงานออกเป็น Module ต่างๆ ซึ่งมีหน้าที่การทำงานที่แตกต่างกัน เพื่อความง่ายในการบริหารจัดการทรัพยากรในแต่ละด้าน และคงไว้ซึ่งประสิทธิภาพสูงสุดในการทำงานของระบบ

# How to install and use Openstack on basic

Mr.Nattawat Thongchaitanawut 56010664

Mr.Bundit Pasaya Advisor

Academic Year 2016

## ABSTRACT

Openstackis so population in Cloud computing on IAAS, that mean however you see openstackso lot in the social of cloud, you will see the Openstackdo or duty. Openstackis use many computer get to connection each other to use service of Cloud infra

This Project I will show you why we choose Openstack, how to install and how to use Openstack

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ดี อันเนื่องมาจากได้รับการแนะนำ ข้อเสนอแนะเป็นอย่างดีจากอาจารย์ที่ปรึกษาโครงงาน อาจารย์ บัณฑิต พัสยา ซึ่งทางคณะผู้จัดทำรู้สึกซาบซึ้งเป็นอย่างมากในความกรุณาของอาจารย์ และขอขอบพระคุณอาจารย์เป็นอย่างสูงที่คอยให้การสนับสนุนในการทำปริญญานิพนธ์นี้เสมอมา

ขอบคุณอาจารย์ทุกท่านและเพื่อนทุกคนในภาควิชาวิศวกรรมคอมพิวเตอร์ที่คอยให้คำปรึกษาและข้อเสนอแนะในกระบวนการทำโครงงานและช่วยเหลือในการทำทดลองต่าง ๆ ของโครงงาน

ขอบพระคุณบิดามารดา และบุคคลภายในครอบครัวที่อยู่เบื้องหลังในความสำเร็จ ที่ได้ให้ความช่วยเหลือ สนับสนุนให้กำลังใจตลอดมา

คณะผู้จัดทำขอขอบพระคุณเป็นอย่างสูง และหวังอย่างยิ่งว่าปริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์ต่อทุกท่าน

นัฐวัฒน์ ธงชัยธนาวุฒิ

# สารบัญ

หน้า

# สารบัญ

หน้า

# สารบัญ

หน้า

# สารบัญรูป

รูป หน้า

# สารบัญรูป

รูป หน้า

# สารบัญรูป

รูป หน้า

# บทที่ 1

# บทนำ

## 1.1 ความสำคัญและที่มาของโครงงาน

OpenStack คือ Cloud Operating System แบบ Open Source ทำหน้าที่บริหารจัดการทรัพยากรด้าน Compute, Storage, Networking ของ Data Center ให้มีประสิทธิภาพในการทำงานสูงที่สุด แบ่งการทำงานออกเป็น Module ต่างๆ ซึ่งมีหน้าที่การทำงานที่แตกต่างกัน เพื่อความง่ายในการบริหารจัดการทรัพยากรในแต่ละด้าน และคงไว้ซึ่งประสิทธิภาพสูงสุดในการทำงานของระบบ

จากที่ได้ฝึกงานที่บริษัท INET และได้ศึกษาเกี่ยวกับ Openstack จึงเกิดสนใจทางด้านนี้และศึกษาเพิ่มเติมเพื่อสามารถนำไปใช้งานได้ภายในอนาคต

## 1.2 วัตถุประสงค์ของโครงงาน

สามารถติดตั้ง Openstack และสามารถใช้งานได้จริง

## 1.3 ขอบเขตของโครงงาน

ศึกษาและติดตั้ง Openstack ลงบน Ubuntu 16.04 เท่านั้น

## 1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) เข้าใจในการติดตั้ง openstack และสามารถนำไปใช้งานได้

2) สามารถติดตั้ง Openstack ได้ในทุกระบบปฏิบัติการ (OS) เช่น Ubuntu , CentOS , OpenSUSE

## 1.5 ข้อจำกัดของระบบ

1) ในรายงานข้างต้นนี่ไม่ได้พบเจอปัญหาต่างๆในการติดตั้งหากพบเจอปัญหาต่างๆในการติดตั้งต้องหาศึกษาหาข้อมูลเพิ่มเติมเอง

2) ไม่สามารถนำไปใช้งานในระบบ Virtual Machine (VM) ได้เพราะไม่สามารถนำไปใช้งานได้

# บทที่ 2

# ทฤษฎีที่เกี่ยวข้อง

## 2.1 เทคโนโลยี Cloud

Cloud computing ก็คือ การประมวลผลบนกลุ่มเมฆหรือจะกล่าวอีกอย่างก็คือ บริการที่ให้เราเลือกใช้ทรัพยากรคอมพิวเตอร์ได้ตามความต้องการ โดยสามารถปรับเพิ่ม ลด ปริมาณการใช้งานได้ง่ายและรวดเร็วผ่าน web interface

ซึ่งรูปแบบบริการ Cloud Computing ก็จะแบ่งได้ 3 รูปแบบ ดังนี้

- Infrastructure as a Service (IaaS) – ให้บริการในส่วน Physical หรือทรัพยากรคอมพิวเตอร์ เช่น CPU, Memory, Operating System, Storage เป็นต้น ตัวอย่าง Amazon
- Platform as a Service (PaaS) – ให้บริการในส่วนทรัพยากรและเครื่องมือสำหรับนักพัฒนา เช่น Google Apps Engine, IBM Mashup Hub, Microsoft Azure
- Software as a Service (SaaS) – ให้บริการในส่วนข้อมูลหรือ Application เช่น Google Apps, Picasa, Youtube, Facebook

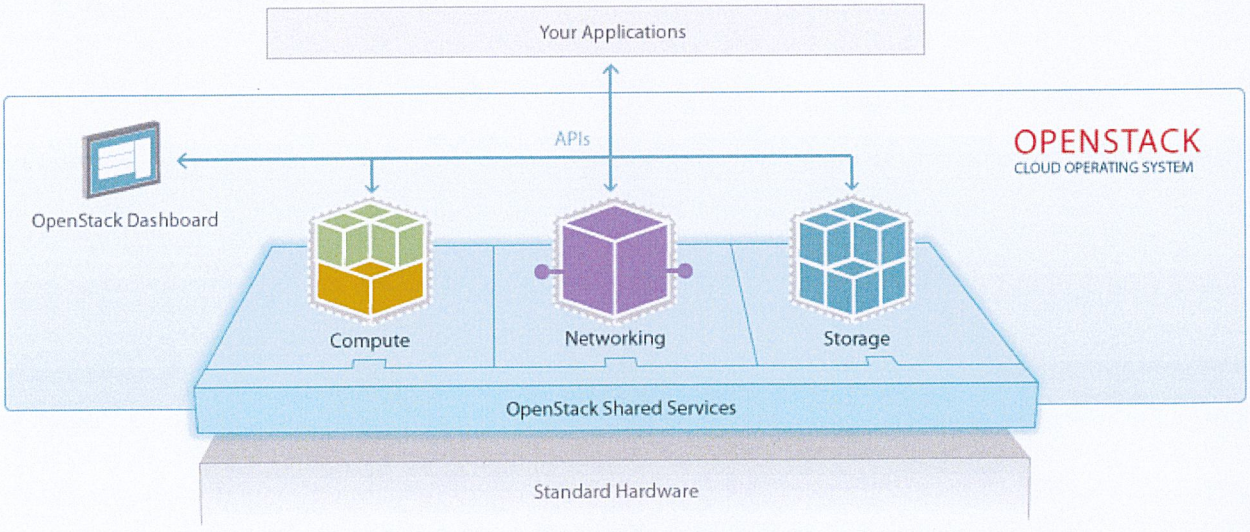
## 2.2 Openstack

OpenStack คือ Cloud Operating System แบบ Open Source ทำหน้าที่บริหารจัดการทรัพยากรด้าน Compute , Storage , Networking ของ Data Center ให้มีประสิทธิภาพในการทำงานสูงที่สุด แบ่งการทำงานออกเป็น Module ต่างๆ ซึ่งมีหน้าที่การทำงานที่แตกต่างกัน เพื่อความง่ายในการบริหารจัดการทรัพยากรในแต่ละด้าน และคงไว้ซึ่งประสิทธิภาพสูงสุดในการทำงานของระบบ

การใช้งานของโปรแกรมหลักๆจะเป็นส่วนของการสร้างและบริหารจัดการระบบ Virtual ต่างๆไม่ว่าจะเป็นการสร้างและใช้งาน Virtual Machine , Virtual Network และ Virtual Storage สำหรับรองรับการใช้งานของระบบ Cloud และให้บริการแบบ multi-tenant ซึ่งเป็นการให้บริการแก่ผู้ใช้หลายคนด้วยซอฟท์แวร์เพียงชุดเดียว

คุณสมบัติเด่นของ OpenStack ดังต่อไปนี้

- OpenStack Compute (code-name Nova)
- OpenStack Networking (code-name Neutron)
- OpenStack Object Storage (code-name Swift)
- OpenStack Block Storage (code-name Cinder)
- OpenStack Identity (code-name Keystone)
- OpenStack Image Service (code-name Glance)
- OpenStack Dashboard (code-name Horizon)
- OpenStack Telemetry (code-name Ceilometer)
- OpenStack Orchestration (code-name Heat)
- OpenStack Database (code-name Trove)
- OpenStack Data Processing (code-name Sahara)
- OpenStack Bare-Metal Provisioning (code-name Ironic)

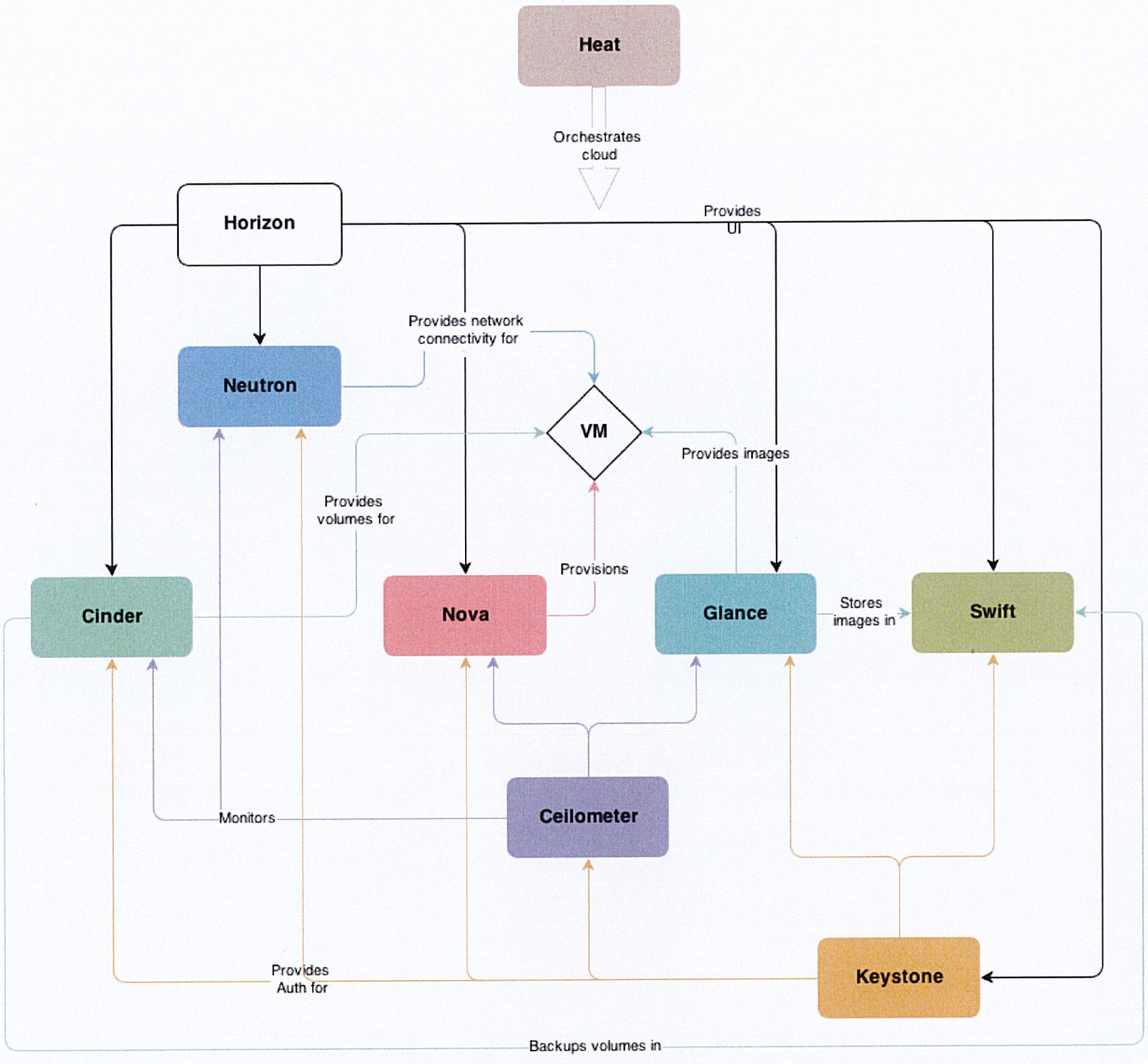



**รูปที่ 2.1 โครงสร้างของ Openstack**

## 2.3 องค์ประกอบของ Openstack

การทำงานของ Openstack เกิดขึ้นจากองค์ประกอบหลายๆ ส่วน ที่เรียกว่า Openstack component โดยในรุ่น Havana ประกอบด้วย Component ดังนี้

- Nova Module
- Neutron Module
- Swift Module
- Cinder Module
- Keystone Module
- Glance Module
- Horizon Module
- Ceilometer Module
- Heat Module




**รูปที่ 2.2 องค์ประกอบของ Openstack**

### 2.3.1 Nova Module

เป็นพื้นฐานของการทำงานในส่วน Compute ของ OpenStack แบ่งการทำงานออกเป็น 2 ส่วน คือ ส่วนการทำงาน API และ Compute Worker ซึ่งทั้ง 2 ส่วนนี้ ทำหน้าที่ควบคุมการประมวลผลการทำงานต่างๆ ของระบบ

### 2.3.2 Neutron Module

เป็นพื้นฐานของการทำงานในส่วน Network ของ OpenStack ผสานการทำงานของ Physical Network ต่างๆ เพิ่มประสิทธิภาพการทำงานด้วยคุณสมบัติ Network Management ด้วยเทคนิคการทำงานของ Virtual Interface ในรูปแบบต่างๆ ไม่ว่าจะเป็น Router , Switch , Network Interface Card หรือ Firewall

### 2.3.3 Swift Module

เป็นพื้นฐานของการทำงานในส่วน Storage ในรูปแบบ Object Storage ของ OpenStack ที่มาพร้อมคุณสมบัติ Scalability, Redundancy และ Durability โดยพัฒนาโครงสร้างการจัดเก็บข้อมูลต่างๆ ตามประเภทของข้อมูล เช่น Images, Photo Storage, Email Storage, Backups และ Archives พร้อมทั้งการบันทึกข้อมูลต่างๆ แบบ Multiple Disks

### 2.3.4 Cinder Module

เป็นพื้นฐานของการทำงานในส่วน Storage ในรูปแบบ Block Storage ของ OpenStack ซึ่งเป็นการทำงานร่วมกับ Nova Module โดยลักษณะการทำงานของ Cinder Module สามารถแบ่งออกได้ 2 รูปแบบ คือ 1. แบบ Ephemeral เป็นลักษณะการเก็บข้อมูลชั่วคราว ซึ่งข้อมูลเหล่านั้นจะหายไปทันที เมื่อมีการหยุดการทำงานของ Instance , 2. แบบ Persistant เป็นลักษณะการเก็บข้อมูลแบบถาวร ซึ่งข้อมูลเหล่านั้นจะไม่สูญหายไป แม้ว่า Instance จะกำลังทำงานอยู่ หรือยุติการทำงานไปแล้ว

### 2.3.5 Keystone Module

เป็นระบบพื้นฐานที่ใช้ในการจัดการสิทธิ์การใช้งานแต่ละ Module ของ OpenStack โดยจะดำเนินการตรวจสอบสิทธิ์การใช้งาน Module ของผู้ใช้ เพื่อระบุตัวตน และความสามารถในเรียกใช้งานแต่ละ Module ของผู้ใช้ ช่วยเพิ่มประสิทธิภาพด้านเสถียรภาพและความปลอดภัยของ OpenStack

### 2.3.6 Glance Module

จัดเป็นระบบ Stored Images ของ OpenStack ทำหน้าที่จัดเก็บ Server Image ต่างๆ ที่จัดเตรียมไว้เพื่อรองรับความต้องการใช้งาน Server Image ของระบบ ไม่ว่าจะเป็นการค้นหา บันทึก หรือ เรียกใช้ อีกทั้งความสามารถในการ Copy หรือ Snapshot Disk และ Server Image รวมทั้งการจัดเก็บข้อมูลแบบ Object Storage อีกด้วย

### 2.3.7 Horizon Module

จัดเป็นระบบ Dashboard ที่ OpenStack พัฒนาขึ้น เพื่อให้ผู้ดูแลระบบ และผู้ใช้ สามารถบริหารจัดการทรัพยากรด้าน Compute , Storage , Networking ได้อย่างมีประสิทธิภาพ

### 2.3.8 Ceilometer Module

จัดเป็นระบบที่ OpenStack ได้ทำการออกแบบมา เพื่อเก็บรวบรวมข้อมูลการใช้งานในด้านต่างๆ และชี้วัดประสิทธิภาพการทำงานของทรัพยากรระบบ เพื่อประเมินความต้องการใช้ทรัพยากรระบบจากการใช้งานของผู้ใช้ นำมาซึ่งการพัฒนาปรับปรุง เพิ่มประสิทธิภาพและคุณภาพในการทำงานของระบบ

### 2.3.9 Heat Module

จัดเป็น Template-Driven Engine ที่อนุญาตให้ผู้พัฒนาโปรแกรมสามารถ ตั้งค่า/ปรับแต่ง Template โครงสร้างของทรัพยากรแต่ละด้าน (Compute , Storage และ Networking) เพื่อรองรับการใช้งาน Cloud Application ของผู้ใช้ ซึ่งรวมไปถึงคุณสมบัติ Autoscaling เมื่อนำ Heat Module และ Ceilometer Module มาทำงานร่วมกัน อีกทั้งยังสามารถตั้งค่า Scaling Group ของทรัพยากรของแต่ละ Template

## 2.4 ทำไมถึงควรลองติดตั้ง Openstack

### 2.4.1 เรียนรู้และลองผิดลองถูกกับการสร้าง Private Cloud

ขั้นตอนหนึ่งที่กินเวลามากๆ ของการเริ่มต้นนำระบบ Cloud ในระดับ Infrastructure as a Service (IaaS) หรือ Platform as a Service (PaaS) มาใช้นั้นก็คือการลองผิดลองถูก ไม่ว่าจะเป็นการคัดเลือกเทคโนโลยี, การปรับแต่งการตั้งค่า, การปรับแต่ง Workflow หรือ Process ให้เหมาะสมกับธุรกิจ, การหา Plugin เชื่อมต่อที่ตอบโจทย์ความต้องการขององค์กร และการแก้ปัญหาเฉพาะหน้าต่างๆ

การเริ่มต้นลองใช้ OpenStack ให้เร็วขึ้นนั้นจะหมายถึงการที่องค์กรและผู้ดูแลระบบจะได้ลองผิดลองถูกก่อนกับเทคโนโลยีที่ใหม่และมีสโคปใหญ่อย่าง Private Cloud เพื่อค้นหาว่าความต้องการในเชิงธุรกิจที่ OpenStack จะสามารถมาช่วยตอบโจทย์ให้กับองค์กรนั้นได้คืออะไร ในขณะที่ผู้ดูแลระบบเองก็จะได้สัมผัสและเรียนรู้กับเทคโนโลยีใหม่ๆ เพื่อปรับตัวไปในเวลาเดียวกัน

การเรียนรู้ตรงนี้ถือเป็นส่วนที่สำคัญมาก เพราะถ้าหากองค์กรเริ่มมีความต้องการระบบ Private Cloud หรือ IT Infrastructure ที่มีความยืดหยุ่นภายในองค์กรสำหรับใช้งานแล้ว ความรู้ในส่วนนี้จะเป็นประโยชน์อย่าง

มากในการออกแบบและกำหนดสโคปของงานทั้งในเชิงธุรกิจและเชิงเทคนิค และทำให้การปรับปรุง IT Infrastructure ครั้งใหญ่ขององค์กรลุล่วงไปได้ด้วยดี แน่นอนว่าด้วยข้อดีของ OpenStack ที่เป็น Open Source นี้ก็ช่วยให้องค์กรสามารถประหยัดค่าใช้จ่ายไปได้มากทีเดียวในระหว่างลองผิดลองถูกครั้งนี้

**2.4.2 สร้าง Environment สำหรับให้องค์กรได้ลองผิดลองถูกในการสร้างเทคโนโลยีใหม่ๆ ทั้งในแง่ของเทคโนโลยีและกระบวนการ**

ไม่เพียงแต่การลองผิดลองถูกในเชิง Infrastructure เท่านั้น หนึ่งในสาเหตุหลักที่เหล่าองค์กรเริ่มนำ OpenStack ไปใช้งานนั้นก็เพื่อสร้าง IT Infrastructure สำหรับทดลองเทคโนโลยีใหม่ๆ หรือพัฒนาเทคโนโลยีของตัวเองขึ้นมาทดสอบได้อย่างง่ายดาย ไม่ว่าจะเป็นการทดลองใช้งานระบบ Big Data Analytics อย่าง Apache Hadoop, Cloudera หรืออื่นๆ, การทดสอบพัฒนา Mobile Application หรือ Web Application ใหม่ขึ้นมาเพื่อให้บริการลูกค้า, การทดสอบเทคโนโลยี Database แนวคิดใหม่ๆ หรือแม้แต่การทดสอบเทคโนโลยี Open Source หรือ Commercial ใหม่ๆ ให้ได้อย่างคล่องตัวนั่นเอง

การลองผิดลองถูกถือเป็นขั้นตอนที่สำคัญมากสำหรับทุกๆ การพัฒนาเทคโนโลยีใหม่ๆ และการพัฒนาเทคโนโลยีใหม่ๆ เองก็เป็นส่วนสำคัญของการก้าวไปสู่การทำ Digital Transformation ดังนั้นการสร้างสนามสำหรับลองผิดลองถูกโดยเฉพาะนี้จึงเป็นสิ่งสำคัญอันดับต้นๆ ของหลายๆ องค์กรไปแล้วในเวลานี้

และแน่นอนว่านอกจากการลองผิดลองถูกในเชิงเทคโนโลยีแล้ว การลองผิดลองถูกในการวางกระบวนการ DevOps เพื่อให้เหล่าผู้ดูแลระบบและเหล่านักพัฒนาสามารถสร้างสรรค์เทคโนโลยีใหม่ๆ ขึ้นมาได้ภายใต้กระบวนการที่เป็นระบบก็ถือเป็นอีกปัจจัยที่สำคัญในระยะยาว เพราะการออกแบบกระบวนการการพัฒนา Software ไปจนถึงการ Deploy ระบบที่เป็นมาตรฐานนั้น จะเป็นตัวเร่งให้การพัฒนาเทคโนโลยีต่างๆ มีความรวดเร็วสูงยิ่งขึ้นนั่นเอง ซึ่ง OpenStack เองนั้นก็รองรับการทำงานร่วมกับเครื่องมือต่างๆ ในการทำ DevOps ได้เป็นอย่างดี

**2.4.3 เริ่มสร้างวัฒนธรรมองค์กรแบบ Self-Service เปลี่ยนแนวคิดในการทำงานของผู้ดูแลระบบ**

วัฒนธรรมองค์กรแบบ Self-Service นี้ถือเป็นอีกจุดเปลี่ยนของระบบ IT ภายในองค์กรเลยก็ว่าได้ และเป็นอีกวัฒนธรรมใหม่ที่องค์กรเองก็ต้องเรียนรู้จากการลองผิดลองถูกด้วยเช่นกัน เพราะต่อไปนั้นแนวโน้มของระบบ IT ต่างๆ ทั้งในระดับของ Data, Application และ Infrastructure นั้นก็มีแนวโน้มที่จะกลายเป็น Self Service มากขึ้นเรื่อยๆ และผู้ใช้งานทั่วๆ ไปที่ไม่ใช่คนสาย IT เองก็ต้องเข้ามาใช้บริการเหล่านี้ด้วยเช่นกัน การ

เริ่มต้นลองผิดลองถูกด้วยระบบ Self-Service สำหรับให้บริการ IT Infrastructure แก่คนสาย IT ด้วยกันเองก่อน โดยใช้ OpenStack จึงถือเป็นทางเลือกที่น่าสนใจ

สิ่งที่องค์กรต้องเรียนรู้ในการสร้างวัฒนธรรม Self-Service ที่ประสบความสำเร็จนี้ ก็คือกระบวนการในการสร้าง Workflow สำหรับการออกแบบบริการ Self-Service แต่ละบริการให้ประสบความสำเร็จในเชิงการใช้งาน ไม่ว่าจะเป็นการเก็บความต้องการ, การออกแบบระบบและบริการ, การดูแลรักษาและติดตาม Resource ที่มีการใช้งาน, การวางแผนว่าควรจะปรับบริการ Self-Service เดิมที่มีอยู่อย่างไร และการ Maintenance หรือ Upgrade แต่ละบริการที่จะต้องมีการสื่อสารและวางแผนกันล่วงหน้าทั้งหมด

Self-Service นั้นถือเป็นเทคโนโลยีที่จะมาเพิ่มความเร็วในการสร้างเทคโนโลยีใหม่ๆ ให้มีความง่ายดายมากยิ่งขึ้น และจำเป็นเป็นอย่างยิ่งสำหรับองค์กรที่ต้องการขับเคลื่อนด้วยเทคโนโลยีและข้อมูลในอนาคต แต่ในขณะเดียวกันการสร้างบริการ Self-Service ที่ดีให้ได้นั้นก็ไม่ใช่เรื่องง่ายเท่าไหร่ และเป็นสิ่งที่องค์กรควรต้องเริ่มเรียนรู้ด้วยการลองผิดลองถูกกับคน IT กันเองก่อนจะต้องเปิดบริการ Self-Service สำหรับผู้ใช้งานทั่วไปในอนาคต

### 2.4.4 เปิดทางเลือกใหม่ๆ สำหรับ IT Infrastructure สำหรับองค์กรในอนาคต

หนึ่งในสาเหตุที่ OpenStack นั้นกลายเป็นที่นิยมก็เป็นเพราะความที่ OpenStack นั้นเป็น Open Source Software ที่ไม่มีค่าใช้จ่ายในการจัดซื้อแต่อย่างใด และรองรับ IT Infrastructure ได้หลากหลาย รวมถึงมีเทคโนโลยีที่น่าสนใจให้พร้อมใช้ในตัว การเริ่มต้นเรียนรู้ Open Stack นั้นเรียกได้ว่าเป็นการเปิดโอกาสและทางเลือกใหม่ๆ สำหรับระบบ IT Infrastructure ที่จะมาช่วยประหยัดค่าใช้จ่ายให้กับองค์กรได้ในระยาวเลยก็ไม่ผิดนัก

จากเดิมที่ธุรกิจองค์กรมักจะคุ้นชินกับการซื้อ Server มาลง Hypervisor สร้างเป็นระบบ Virtualization และซื้อ Storage มาเชื่อมต่อเข้าไปในระบบ Virtualization กันเป็นหลักนั้น ก็จะกลายเป็นแค่ทางเลือกหนึ่งในอนาคตหลังการมาของ OpenStack เพราะ OpenStack นั้นมีครบถ้วนทั้งระบบ Compute, Storage, Network และ Management ในตัว ทำให้องค์กรสามารถเพิ่มทรัพยากรใดๆ ก็ได้จากการลงทุนเพียงแค่ Server ซึ่งถือเป็น Component ที่มีราคาถูกที่สุดใน Data Center แล้วในเวลานี้ ซึ่งข้อดีตรงนี้ก็จะช่วยทำให้องค์กรวางแผนการลงทุนทรัพยากรต่างๆ ได้อย่างง่ายดายไปด้วยอีกทาง

ในทางกลับกัน OpenStack นั้นสามารถใช้ในการควบคุมและบริการจัดการ Hypervisor ต่างๆ ได้หลากหลายค่าย รวมถึงสร้างระบบ Self Service ให้แก่ IT Infrastructure นั้นๆ ได้ด้วย การนำ OpenStack มาใช้จึงไม่ต้องละทิ้งเทคโนโลยีเดิมที่มีการใช้งานอยู่แต่อย่างใด ในขณะที่ยังเปิดทางเลือกใหม่ในการเลือกใช้เทคโนโลยีได้หลากหลายยิ่งขึ้นกว่าเดิมในอนาคตไปพร้อมๆ กัน

ปัจจุบันผู้ผลิตรายใหญ่แทบทุกรายต่างก็ให้การสนับสนุน OpenStack กันทั้งนั้น ดังนั้นจึงไม่ต้องกังวลกันเลยในเรื่องของความเข้ากันได้กับระบบเดิม หรืออนาคตของ OpenStack เองก็ตาม

**2.4.5 รับการมาของเทคโนโลยี Network Functions Virtualiztion (NFV) และ Software-Defined Networking (SDN)**

นับวันเทคโนโลยีฝั่งระบบเครือข่ายเองก็ถูกเปลี่ยนมาให้กลายเป็น Virtualization มากขึ้นกันเรื่อยๆ และ OpenStack หรือ KVM ที่เป็น Open Source Hypervisor เองก็เป็นหนึ่งใน IT Infrastructure ที่เหล่าผู้ผลิตระบบเครือข่ายให้การรองรับกันอย่างเป็นมาตรฐาน การใช้ OpenStack นั้นนอกจากจะเปิดให้เหล่าองค์กรได้มีทางเลือกใหม่ๆ สำหรับเทคโนโลยีฝั่ง Server และ Storage แล้ว จึงยังช่วยเปิดโอกาสให้กับเทคโนโลยีฝั่ง Network เป็นอย่างมากอีกด้วย

การมาของ NFV และ SDN นี้จะช่วยให้องค์กรสามารถปรับแต่งและบริหารจัดการระบบเครือข่ายเดิมที่มีอยู่ได้อย่างยืดหยุ่นมากยิ่งขึ้น ซึ่งถึงแม้เทคโนโลยีเหล่านี้จะยังถือว่าค่อนข้างใหม่ แต่ก็จะเป็นเทคโนโลยีที่เข้ามามีบทบาทเป็นอย่างมากในกลุ่มธุรกิจขนาดใหญ่ โดยเฉพาะอย่างยิ่งกลุ่มธุรกิจโทรคมนาคม และธุรกิจองค์กรที่มีสาขาเป็นจำนวนมาก

# บทที่ 3

# การวิเคราะห์และออกแบบระบบ

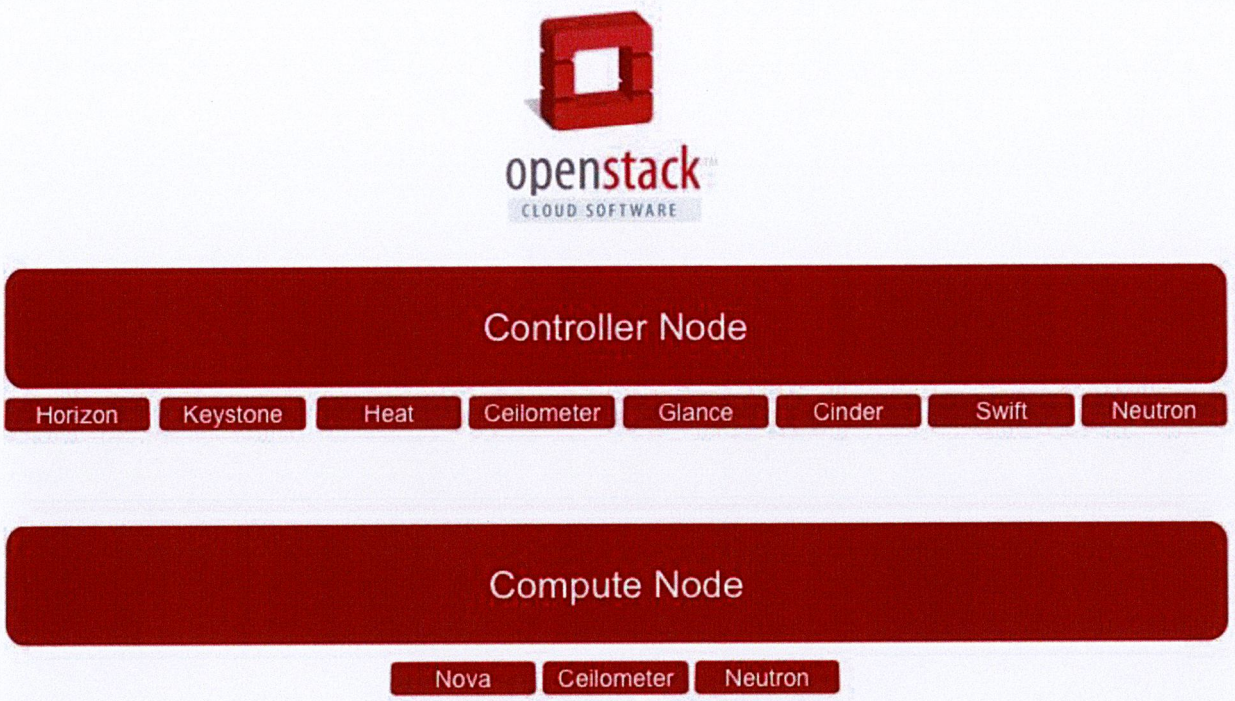
Openstack เป็นการจำลอง Server และการสร้าง VM ทำหน้าที่บริหารจัดการทรัพยากรด้าน Compute, Storage, Networking ของ Data Center ให้มีประสิทธิภาพในการทำงานสูงที่สุดดังนั้นจำเป็นต้องใช้ hardware ที่มีประสิทธิภาพในการทำงานสูง

## 3.1 ระบบที่ใช้

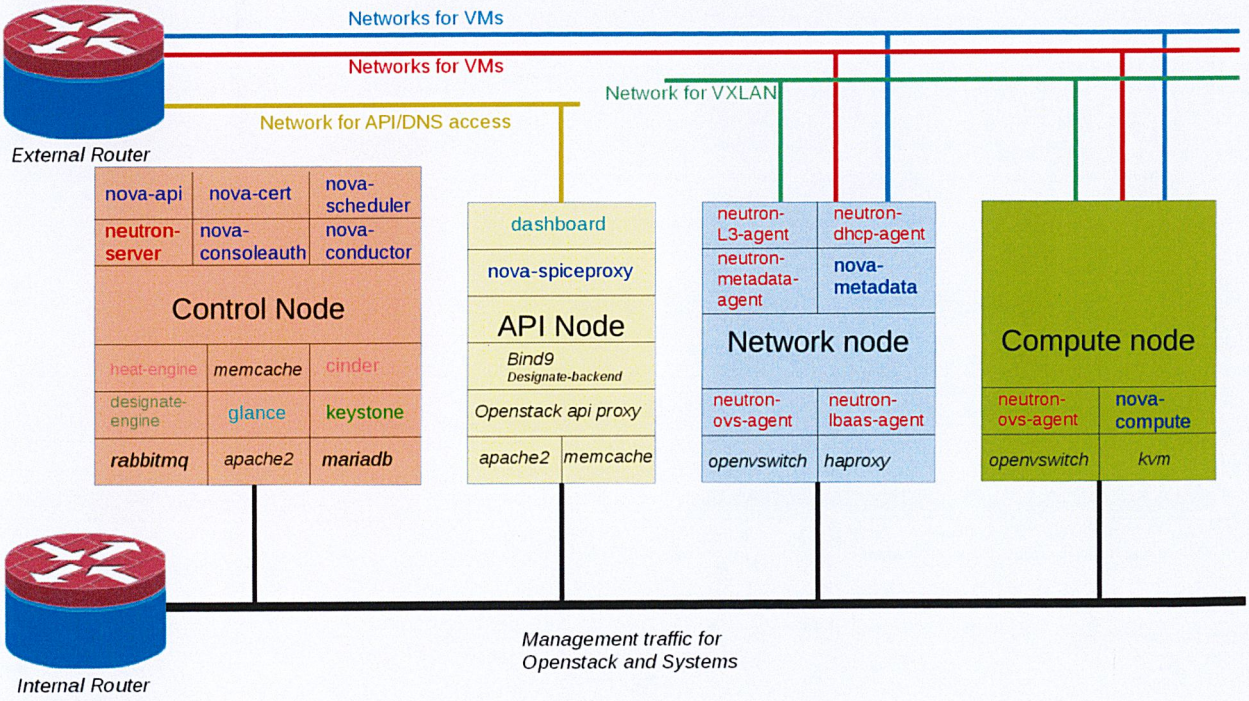
1) ระบบปฏิบัติการ : Ubuntu 16.04
2) Core : 8core vCPU
3) Ram : 16GB vRam
4) Harddisk : 160GB vDisk

## 3.2 การออกแบบของระบบ

จำเป็นต้องมี Controller Node และ Compute Node จะมี Node อื่นๆเป็นส่วนช่วยในการทำงาน โดย Controller Node จะเป็นตัวควบคุม Node อื่นๆในการทำงาน




**รูป 3.1** โครงสร้างพื้นฐานของ Openstack




**รูป 3.2** โครงสร้างภายในของ Openstack

# บทที่ 4

# การทดลองและผลการทดลอง

## 4.1 สภาพแวดล้อมของระบบ

ในการติดตั้ง Openstack แบบ All-in-One node ต้องจัด spec คอมดังนี้

1) ระบบปฏิบัติการ : Ubuntu 16.04
2) Core : 8core vCPU
3) Ram : 16GB vRam
4) Harddisk : 160GB vDisk

## 4.2 ขั้นตอนการติดตั้ง Network Time Protocol (NTP)

### 4.2.1 Controller Node

**4.2.1.1** ใช้คำสั่ง sudo su เพื่อเปิดใช้งาน root user

**4.2.1.2** ใช้คำสั่ง apt –get update และ apt –get –f install เพื่อ update ให้อยู่ใน version ล่าสุด

**4.2.1.3** ติดตั้ง packages โดยใช้คำสั่ง apt install chrony

**4.2.1.4** แก้ไข NTP SERVER โดยใช้คำสั่ง /etc/chrony/chrony.conf จากนั้นให้แก้ NTP SERVER เป็น th.pool.ntp.org เพื่อเปิดการใช้งาน NTP SERVER ของไทย และเปิดการเชื่อมต่อกับ node โดยการ allow 10.0.0.0/24

**4.2.1.5** เมื่อติดตั้งเสร็จเรียบร้อยให้ restart ระบบด้วยคำสั่ง service chrony restart

### 4.2.2 Other nodes

**4.2.2.1** ใช้คำสั่ง sudo su เพื่อเปิดใช้งาน root user

**4.2.2.2** ใช้คำสั่ง apt –get update และ apt –get –f install เพื่อ update ให้อยู่ใน version ล่าสุด

**4.2.2.3** ติดตั้ง packages โดยใช้คำสั่ง apt install chrony

**4.2.2.4** แก้ไข NTP SERVER โดยใช้คำสั่ง /etc/chrony/chrony.conf จากนั้นให้แก้ NTP SERVER เป็น server controller iburst เพื่อให้ใช้ Server เดียวกันกับ controller

**4.2.2.5** เมื่อติดตั้งเสร็จเรียบร้อยให้ restart ระบบด้วยคำสั่ง service chrony restart

หากเราติดตั้งเรียบร้อยเราจะสามารถเช็คการเชื่อมต่อของ node ต่างๆได้ด้วยคำสั่ง chronyc sources โดยจะเป็นตามดังรูปข้างต้น

```
# chronyc sources
210 Number of sources = 2
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- 192.0.2.11                2   7    12   137  -2814us[-3000us] +/-   43ms
^* 192.0.2.12                2   6   177    46    +17us[  -23us] +/-   68ms
```

**รูป 4.1** เช็คการเชื่อมต่อ node จาก Controller Node

```
# chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* controller                3   9   377   421    +15us[  -87us] +/-   15ms
```

**รูป 4.2 เช็คการเชื่อมต่อ node จาก Other Node**

## 4.3 ขั้นตอนการติดตั้ง Openstack Package

**4.3.1** ใช้งานคำสั่ง apt install software-properties-common

**4.3.2** ใช้งานคำสั่ง add-apt-repository cloud-archive:newton

**4.3.3** ให้ update ระบบต่างๆด้วยคำสั่ง apt update && apt dist-upgrade

**4.3.4** ติดตั้ง client ของ openstack ด้วยคำสั่ง apt install python-openstackclient

**4.3.5** restart ระบบด้วยการปิดและเปิดเครื่องใหม่

## 4.4 ขั้นตอนการติดตั้ง SQL database

**4.4.1** ติดตั้ง packages SQL database ด้วยคำสั่ง apt install mariadb-server python-pymysql

**4.4.2** สร้างและแก้ไขไฟล์ /etc/mysql/mariadb.conf.d/99-openstack.cnf ให้เป็น mysqld ดังรูป

```
[mysqld]
bind-address = 10.0.0.11
default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

**รูปที่ 4.3** การ setting ค่า database

**4.4.3** เมื่อแก้ไขเรียบร้อยแล้วให้ restart ระบบด้วยคำสั่ง service mysql restart

**4.4.4** จากนั้นให้ติดตั้ง mysql ด้วยคำสั่ง mysql_secure_installation

## 4.5 ขั้นตอนการติดตั้ง Message queue

**4.5.1** ติดตั้ง package ด้วยคำสั่ง apt install rabbitmq-server

**4.5.2** สร้าง user ขึ้นมาด้วยคำสั่ง rabbitmqctl add_user openstack RABBIT_PASS

**4.5.3** set permission ด้วยคำสั่ง rabbitmqctl set_permissions openstack ".*" ".*" ".*"

## 4.6 ขั้นตอนการติดตั้ง Memcached

**4.6.1** ติดตั้ง package ด้วยคำสั่ง apt install memcached python-memcache

**4.6.2** แก้ไข ip address ที่เชื่อมต่อให้เป็น ip address เดียวกันกับ controller node ด้วยคำสั่ง /etc/memcached.conf

**4.6.3** เมื่อแก้ไขเสร็จเรียบร้อยแล้วให้ restart ระบบด้วยคำสั่ง service memcached restart

## 4.7 การติดตั้งและตั้งค่าระบบเบื้องต้น

**4.7.1** สร้าง sql ขึ้นมาด้วย root user ด้วยคำสั่ง mysql -u root –p

**4.7.2** สร้าง keystone ขึ้นใน database ด้วยคำสั่ง mysql> CREATE DATABASE keystone;

**4.7.3** ติดตั้ง keystone package ด้วยคำสั่ง apt install keystone

**4.7.4** แก้ไข keystone ในส่วนต่างๆดังรูปด้วยคำสั่ง /etc/keystone/keystone.conf

```
[database]
...
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

**รูปที่ 4.4** การ config ค่า keystone ในส่วนของ database

```
[token]

...

provider = fernet
```

**รูปที่ 4.5** การ config ค่า keystone ในส่วนของ token

**4.7.5** ตั้งค่าที่อยู่ของ keystone ด้วยคำสั่ง su -s /bin/sh -c "keystone-manage db_sync" keystone

**4.7.6** ติดตั้ง Fernet key และ Bootstrap ด้วยคำสั่งตามรูปข้างล่าง

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone

# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

# keystone-manage bootstrap --bootstrap-password ADMIN_PASS \

  --bootstrap-admin-url http://controller:35357/v3/ \

  --bootstrap-internal-url http://controller:35357/v3/ \

  --bootstrap-public-url http://controller:5000/v3/ \

  --bootstrap-region-id RegionOne
```

**รูปที่ 4.6** ติดตั้ง Fernet key และ Bootstrap keystone

## 4.8 การสร้าง domain, projects, users, and roles

### 4.8.1 ขั้นตอนการสร้าง service project

```
$ openstack project create --domain default \
 --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 24ac7f19cd944f4cba1d77469b2a73ed |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | default                          |
+-------------+----------------------------------+
```

รูป4.7 ขั้นตอนการสร้าง service project

**4.8.2** ขั้นตอนการสร้าง demo project

```
$ openstack project create --domain default \
  --description "Demo Project" demo
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 231ad6e7ebba47d6a1e57e1cc07ae446 |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | default                          |
+-------------+----------------------------------+
```

**รูป 4.8** ขั้นตอนการสร้าง demo project

**4.8.3** การสร้าง demo user

```
$ openstack user create --domain default \
  --password-prompt demo
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | aeda23aa78f44e859900e22c24817832 |
| name                | demo                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

**รูป 4.9** การสร้าง demo user

**4.8.4** การสร้าง user role

```
$ openstack role create user

+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 997ce8d05fc143ac97d83fdfb5998552 |
| name      | user                             |
+-----------+----------------------------------+
```

**รูป 4.10** การสร้าง user role

**4.8.5** การแอด user role เข้าไปใน demo project

```
$ openstack role add --project demo --user demo user
```

**รูป 4.11** การแอด user role เข้าไปใน demo project

## 4.9 การ Verify operation

**4.9.1** ใช้คำสั่ง unset OS_AUTH_URL OS_PASSWORD

**4.9.2** ใช้ admin user ส่งคำขอการเข้าถึง token

```
$ openstack --os-auth-url http://controller:35357/v3 \
  --os-project-domain-name Default --os-user-domain-name Default \
  --os-project-name admin --os-username admin token issue


Password:
+------------+----------------------------------------------------------------+
| Field      | Value                                                          |
+------------+----------------------------------------------------------------+
| expires    | 2016-02-12T20:14:07.056119Z                                    |
| id         | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmjEO643d-e_j-XXq9AmIegIbA7UHGPv |
|            | atnN21qtOMjCFWX7BReJEQnVOAj3nclRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
|            | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dclHVnz-OMVLiyM9ws       |
| project_id | 343d245e850143a096806dfaefa9afdc                               |
| user_id    | ac33776331494012961c0d92d79dc16                                |
+------------+----------------------------------------------------------------+
```

**รูป 4.12** admin user ส่งคำขอการเข้าถึง token

**4.9.3** ใช้ demo user ส่งคำขอการเข้าถึง token

```
$ openstack --os-auth-url http://controller:5000/v3 \
 --os-project-domain-name Default --os-user-domain-name Default \
 --os-project-name demo --os-username demo token issue


Password:
+------------+-----------------------------------------------------------+
| Field      | Value                                                     |
+------------+-----------------------------------------------------------+
| expires    | 2016-02-12T20:15:39.014479Z                               |
| id         | gAAAAABWvi9bsh7vkiby5BpCCnc-JkbGhm9wH3fabS_cY7uabOubesi-Me6IGWW |
|            | yQqNegDDZ5jw7grI26vvgy1J5nCVwZ_zFRqPiz_qhbq29mgbQLglbkq6FQvzBRQ |
|            | JcOzq3uwhzNxszJWmzGC7rJE_H0A_a3UFhqv8M4zMRYSbS2YF0MyFmp_U      |
| project_id | ed0b60bf607743088218b0a533d5943f                          |
| user_id    | 58126687cbcc4888bfa9ab73a2256f27                          |
+------------+-----------------------------------------------------------+
```

**รูป 4.13** demo user ส่งคำขอการเข้าถึง token

## 4.10 การสร้าง OpenStack client environment scripts

### 4.10.1 แก้ไข admin-openrc

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

**รูป 4.14** แก้ไข admin-openrc

### 4.10.2 แก้ไข demo-openrc

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

**รูป 4.15** แก้ไข demo-openrc

**4.10.3** โหลด admin-openrc และ ร้องคำ token

```
$ . admin-openrc

$ openstack token issue

+------------+----------------------------------------------------------+

| Field      | Value                                                    |

+------------+----------------------------------------------------------+

| expires    | 2016-02-12T20:44:35.659723Z                              |

| id         | gAAAAABWvjYj-Zjfg8WXFaQnUd1DMYTBVrKw4h3fIagi5NoEmh21U72SrRv2trl |

|            | JWFYhLi2_uPR31Igf6A8mH2Rw9kv_bxNo1jbLNPLGzW_u5FC7InFqx0yYtTwa1e |

|            | eq2b0f6-18KZyQhs7F3teAta143kJEWuNEYET-y7u29y0be1_64KYkM7E      |

| project_id | 343d245e850143a096806dfaefa9afdc                         |

| user_id    | ac3377633149401296f6c0d92d79dc16                         |

+------------+----------------------------------------------------------+
```

**รูป 4.16** โหลด admin-openrc และ ร้องคำ token

## 4.11 การติดตั้ง Image service

**4.11.1** ใช้ root user สร้าง database โดยใช้คำสั่ง mysql -u root –p

**4.11.2** สร้าง glance และตั้งค่า glance

```
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

**รูปที่ 4.17** สร้าง glance และตั้งค่า glance

**4.11.3** สร้าง glance user

```
$ openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
| Field               | Value                            |
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 3f4e777c4062483ab8d9edd7dff829df |
| name                | glance                           |
| password_expires_at | None                             |
```

**รูปที่ 4.18** สร้าง glance user

**4.11.4** สร้าง glance service enity

```
$ openstack service create --name glance \
  --description "OpenStack Image" image
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Image                  |
| enabled     | True                             |
| id          | 8c2c7f1b9b5049ea9e63757b5533e6d2 |
| name        | glance                           |
| type        | image                            |
+-------------+----------------------------------+
```

**รูปที่ 4.19** สร้าง glance service enity

**4.11.5** ติดตั้ง glance package ด้วยคำสั่ง apt install glance

**4.11.6** แก้ไขค่าใน /etc/glance/glance-api.conf ในส่วนของ database และ keystone ดังนี้

```
[database]

...

connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

auth_uri = http://controller:5000

auth_url = http://controller:35357

memcached_servers = controller:11211

auth_type = password

project_domain_name = Default

user_domain_name = Default

project_name = service

username = glance

password = GLANCE_PASS

[paste_deploy]

flavor = keystone
```

**รูปที่ 4.20** แก้ไขค่าใน /etc/glance/glance-api.conf

**4.11.7** แก้ไขค่าใน /etc/glance/glance-registry.conf ในส่วนของ database และ keystone ดังนี้

```
[database]

...

connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]

...

auth_uri = http://controller:5000

auth_url = http://controller:35357

memcached_servers = controller:11211

auth_type = password

project_domain_name = Default

user_domain_name = Default

project_name = service

username = glance

password = GLANCE_PASS


[paste_deploy]

flavor = keystone
```

**รูปที่ 4.21** แก้ไขค่าใน /etc/glance/glance-registry.conf

**4.11.8** restart ระบบ image service ด้วยคำสั่ง

```
# service glance-registry restart

# service glance-api restart
```

**รูปที่ 4.22** restart ระบบ image service

## 4.12 การติดตั้ง Compute Service ใน Controller Node

**4.12.1** ใช้ root user เชื่อมต่อกับ database และสร้าง nova database

```
$ mysql -u root –p
mysql> CREATE DATABASE nova_api;
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
 IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
 IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

**รูปที่ 4.23** root user เชื่อมต่อกับ database และสร้าง nova database

**4.12.2** สร้าง nova user และ ให้ admin เชื่อมต่อ nova user

```
$ openstack user create --domain default \
  --password-prompt nova
User Password:
Repeat User Password:
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 8a7dbf5279404537b1c7b86c033620fe |
| name                | nova                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
$ openstack role add --project service --user nova admin
```

**รูปที่ 4.24** สร้าง nova user และ ให้ admin เชื่อมต่อ nova user

**4.12.3** ติดตั้ง package ด้วยคำสั่ง apt install nova-api nova-conductor nova-consoleauth \

nova-novncproxy nova-scheduler

**4.12.4** แก้ไขค่าใน /etc/nova/nova.conf ให้ได้ผลดังนี้

```
[api_database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api
[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
```

**รูปที่ 4.25** แก้ไขค่าใน /etc/nova/nova.conf

**4.12.5** แก้ไขค่าใน RabbitMQ ให้ได้ผลดังนี้

```
[DEFAULT]
auth_strategy = keystone
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS
```

**รูปที่ 4.26** แก้ไขค่าใน RabbitMQ

**4.12.6** ใน compute database ให้ใช้คำสั่ง su -s /bin/sh -c "nova-manage api_db sync" nova และ su -s /bin/sh -c "nova-manage db sync" nova

**4.12.7** restart ระบบทั้งหมดด้วยคำสั่ง

```
# service nova-api restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

**รูปที่ 4.27** restart ระบบทั้งหมดของ compute service ใน controller

## 4.13 การติดตั้ง Compute Service ใน Compute Node

**4.13.1** ติดตั้ง package ด้วยคำสั่ง apt install nova-compute

**4.13.2** แก้ไขค่าใน /etc/nova/nova.conf ให้เป็นไปดังนี้

```
[DEFAULT]
...
auth_strategy = keystone

[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
```

```
auth_type = password

project_domain_name = Default

user_domain_name = Default

project_name = service

username = nova

password = NOVA_PASS
```

**รูปที่ 4.28** แก้ไขค่าใน /etc/nova/nova.conf

**4.13.3** ใน [DEFAULT] เพิ่มคำสั่งต่างๆลงไปดังนี้

```
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS

use_neutron = True

firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

**4.13.4** กำหนดให้ compute node เชื่อมต่อกับ hardware ต่างๆใน VM ด้วยคำสั่ง

egrep -c '(vmx|svm)' /proc/cpuinfo

4.13.5 restart ระบบของ compute service ด้วยคำสั่ง service nova-compute restart

## 4.14 การติดตั้ง Networking Service ใน Controller Node

**4.14.1** ใช้ root user ในการเชื่อมต่อ database เข้ากับ client ดังนี้ mysql -u root –p

**4.14.2** สร้าง neutron database

```
mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

**รูปที่ 4.29** สร้าง neutron database

**4.14.3** สร้าง neutron user

```
$ openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 319f34694728440eb8ffcb27b6dd8b8a |
| name                | neutron                          |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

**รูปที่ 4.30** สร้าง neutron user

**4.14.4** เพิ่ม admin เข้าไปใน neutron user ด้วยคำสั่ง openstack role add --project service --user neutron admin

**4.14.5** สร้าง neutron service entity

```
$ openstack service create --name neutron \
  --description "OpenStack Networking" network
| Field       | Value                            |
| description | OpenStack Networking             |
| enabled     | True                             |
| id          | f71529314dab4a4d8eca427e701d209e |
| name        | neutron                          |
| type        | network                          |
```

**รูปที่ 4.31** สร้าง neutron service entity

**4.14.6** restart ระบบทั้งหมดด้วยคำสั่ง

```
# service nova-api restart
# service neutron-server restart
# service neutron-linuxbridge-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
# service neutron-l3-agent restart
```

**รูปที่ 4.32** restart ระบบทั้งหมดของ networking service ใน controller node

## 4.15 การติดตั้ง Networking Service ใน Compute Node

**4.15.1** ติดตั้ง package โดยใช้คำสั่ง apt install neutron-linuxbridge-agent

**4.15.2** แก้ไขค่าใน /etc/neutron/neutron.conf ให้เป็นดังนี้

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

**รูปที่ 4.33** แก้ไขค่าใน /etc/neutron/neutron.conf

**4.15.3** restart ระบบทั้งหมดของ Networking Service ใน Compute Node

```
# service nova-compute restart
# service neutron-linuxbridge-agent restart
```

**รูปที่ 4.34** restart ระบบทั้งหมดของ Networking Service ใน Compute Node

**4.16 การติดตั้ง Dashboard**

**4.16.1** ติดตั้ง package ด้วยคำสั่ง apt install openstack-dashboard

**4.16.2** แก้ไขค่าใน /etc/openstack-dashboard/local_settings.py ให้เป็นดังนี้

```
OPENSTACK_HOST = "controller"
ALLOWED_HOSTS = ['*', ]
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
   'default': {
       'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
 'LOCATION': 'controller:11211',    }        }
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
   "identity": 3,
   "image": 2,
   "volume": 2,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_NEUTRON_NETWORK = {
   ...
```

```
    'enable_router': False,
    'enable_quotas': False,
    'enable_ipv6': False,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
    'enable_fip_topology_check': False,
}

TIME_ZONE = "TIME_ZONE"
```

**รูปที่ 4.35** แก้ไขค่าใน Dashboard

**4.16.3** restart ระบบด้วยคำสั่ง service apache2 reload

# บทที่ 5

# บทสรุปและแนวทางแก้ไข

## 5.1 บทสรุป

ในการติดตั้ง Openstack ใน Ubuntu นั้นจะต้องมีความรู้เบื้องต้นในการใช้งาน Ubuntu พอสมควร หากทำตามการทดลองทุกขั้นตอนโดยไม่มีปัญหาเกิดขึ้นจะสามารถใช้งาน Openstack

## 5.2 ปัญหาที่เกิดขึ้นและแนวทางแก้ปัญหา

ตรวจสอบ hardware ว่ารองรับสามารถขึ้น node ในแต่ละ node ได้หรือไม่หากรองรับไม่ได้ให้เปลี่ยน hareware ใหม่