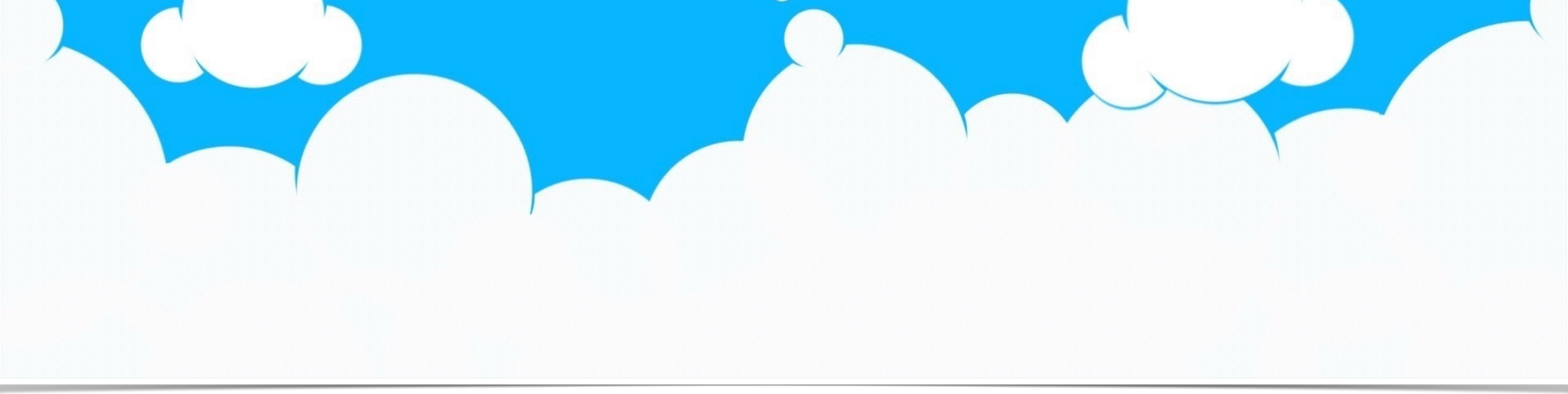

openstack
CLOUD SOFTWARE
PRIVATE CLOUD
>Kilo Release
HEAT ORCHESTRATION
>AUTOSCALE
>Ceilometer
>LBaaS



AutoScaling Service on Openstack

Basic Concept
Heat template
Ceilometer
LBaaS

IT BAKERY
Where ideas baked fresh

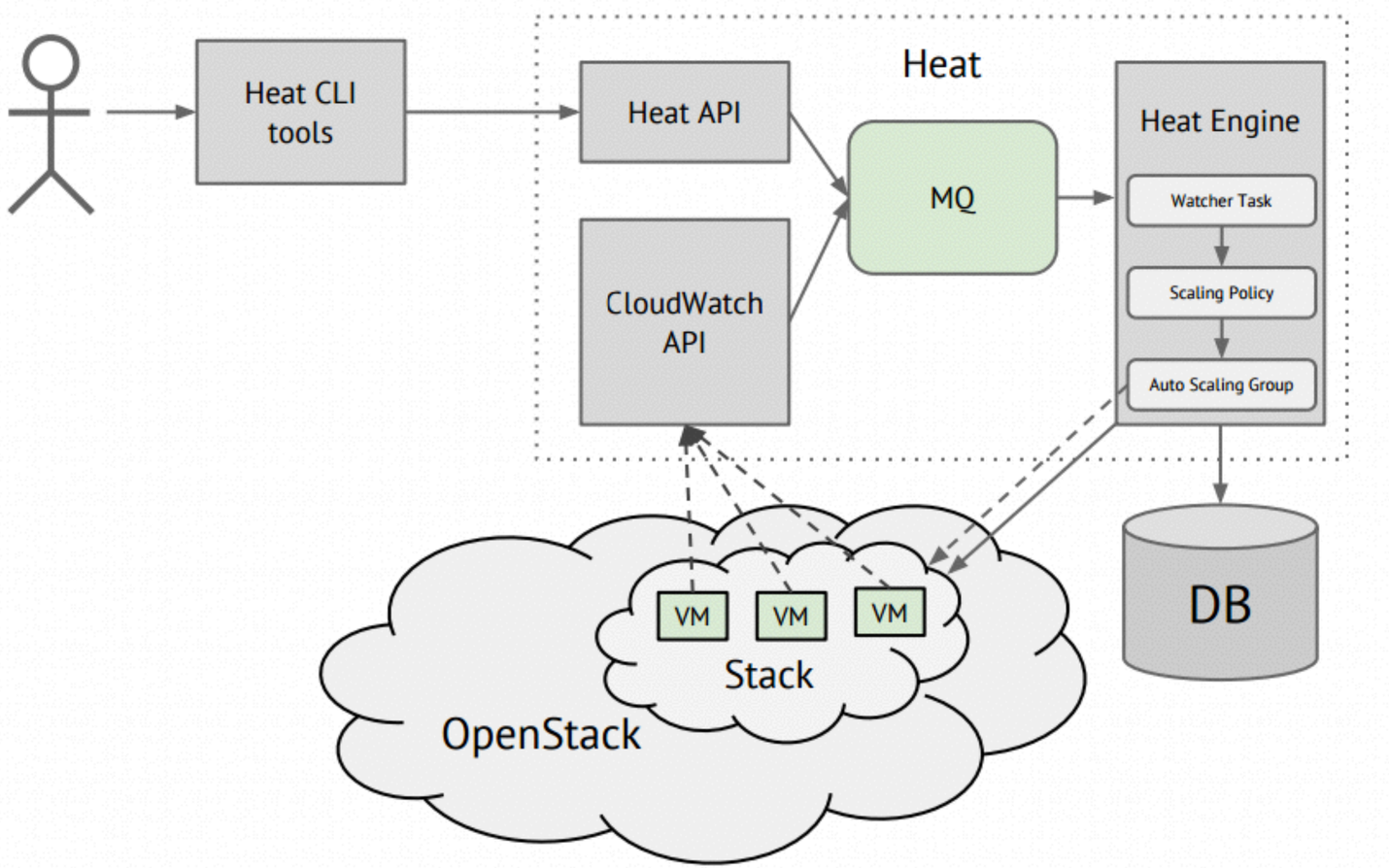
edit packstack answer file

```
43 CONFIG_HEAT_INSTALL=y
1024:CONFIG_HEAT_CLOUDWATCH_INSTALL=y
1028:CONFIG_HEAT_CFN_INSTALL=y

# openstack-status

# export CEILO_SVCS='compute central collector api alarm-
evaluator alarm-notifier'
# for svc in $CEILO_SVCS ; do sudo service openstack-ceilometer-
$svc restart ; done
#for svc in $CEILO_SVCS ; do sudo grep ERROR /var/log/
ceilometer/${svc}.log ; done
```

Heat Template

http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

Heat Architecture



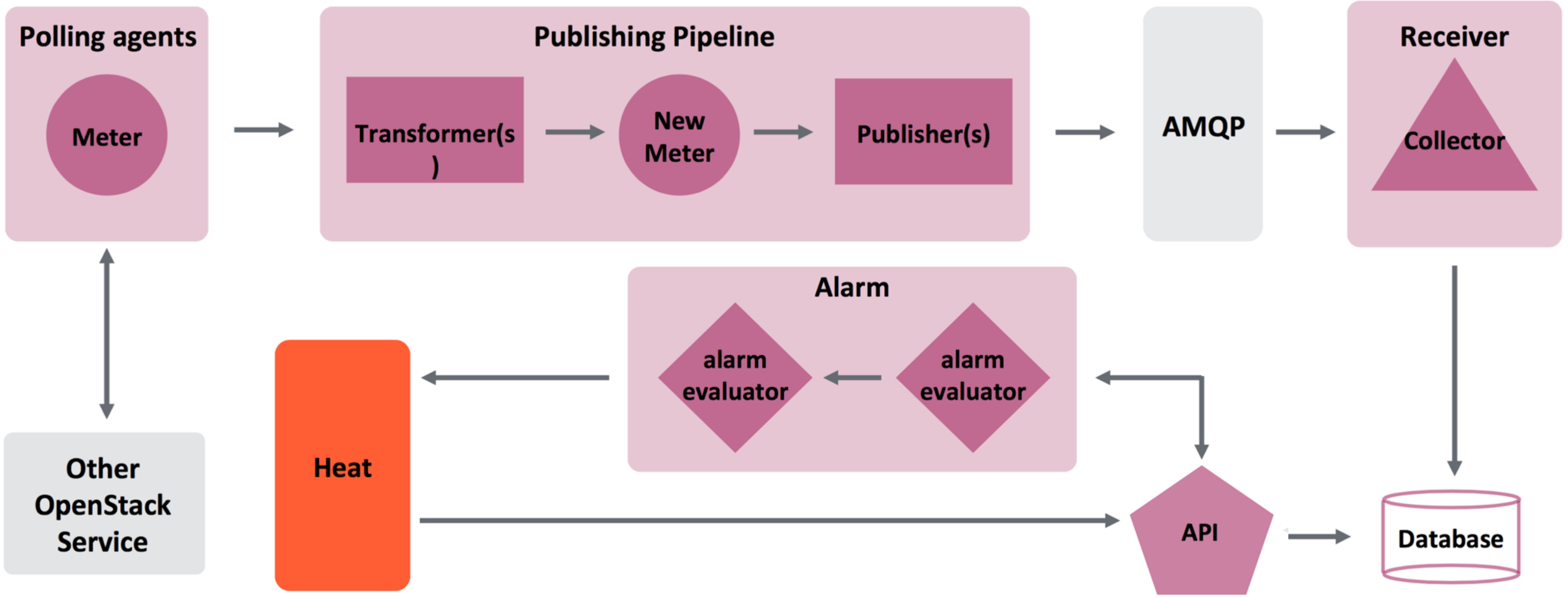
Heat Component

- Heat-api
- Heat-api-cfn
- Heat-engine
- Heat-cli

Ceilometer Overview

http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

Ceilometer Overview



command line

```
# ceilometer meter-list |cut -d'|' -f 2,3 |sort -n |uniq

+---------------------------------------+------------+-------+
|          NAME                         |   Type     |  Unit |
+---------------------------------------+------------+-------+
 cpu                                    | cumulative | ns
 cpu_util                               | gauge      | %


# vi /etc/ceilometer/pipeline.yaml
```

การทดสอบ 1

พื้นฐานการสร้าง instance ด้วย HOT แบบ single template

ต้องการทดสอบการสร้าง instance จำนวน 1 instance ด้วย Heat template เข้าใจการกำหนดค่าของตัวแปรที่จำเป็นในการสร้าง instance ได้แก่ image, network, subnet และ security group

Demo

การทดสอบ 2

สร้าง Database Server ด้วย การrun script ภายใน Heat Template

ต้องการทดสอบการ run script ภายใน instace ที่สร้างขึ้นด้วยโปรแกรม cloud-init โดยจะทำงานตาม script ที่อยู่ค่าของ template ที่อยู่ภายในค่าของ user_data: และparams: การทำงานจะเป็นทำงานเหมือนกับการติดตั้งปรกติทั่วไปเหมือนการติดตั้งแบบ manualและ script ที่นำมาทดสอบการติดตั้งคือการการติดตั้ง ฐานข้อมูล

Demo

การทดสอบ 3

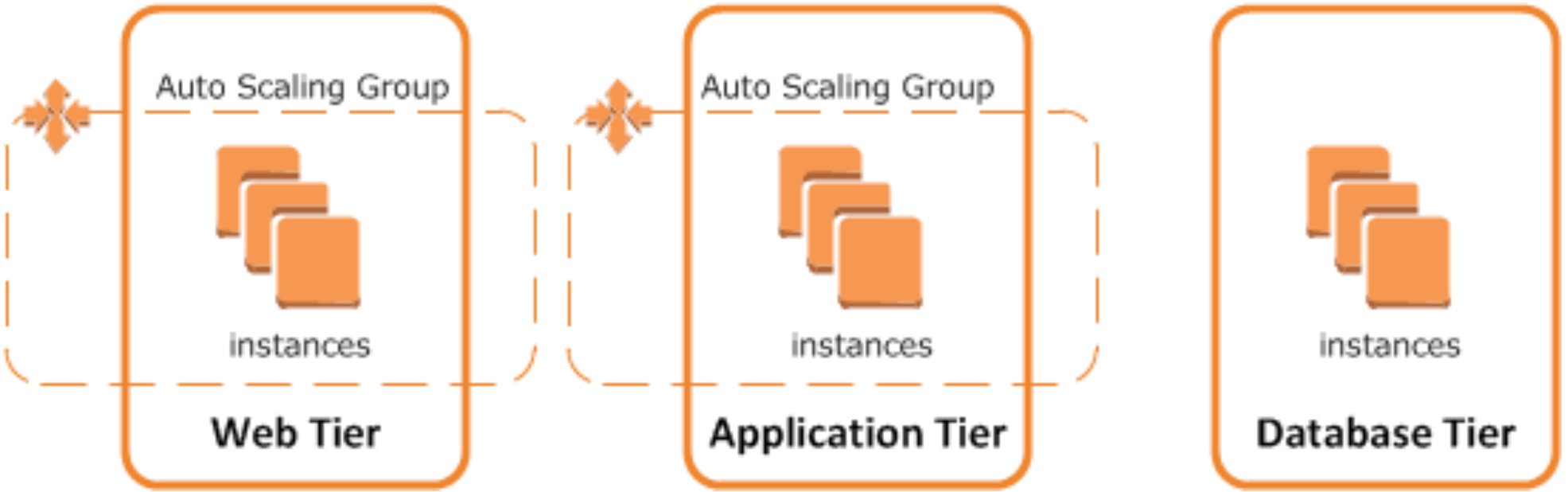
สร้าง Instance Autoscale multinode (แบบไม่มี load balance)

ต้องการทดสอบการสร้าง instance ด้วย Heat template โดยมีการใช้งานฟังก์ชั้นAutoscale (OS::Heat::AutoScalingGroup) ในการทดสอบนี้ผู้ใช้งานจะเข้าใจการทำงานร่วมกันระหว่าง Heat สำหรับการทำการสร้าง image โดยการสร้างแต่ละ image จะเป็นผลการการทำงานร่วมกันกับ Ceilometer เพื่อสร้าง Alert

ในการทดสอบแนะนำให้ใช้ OS::Heat::CloudConfig เพื่อ runโปรแกรมสำหรับ สร้าง load cpu ให้แก่ instance คือ stress เพื่อให้เกิด alert ส่งไปยัง heat

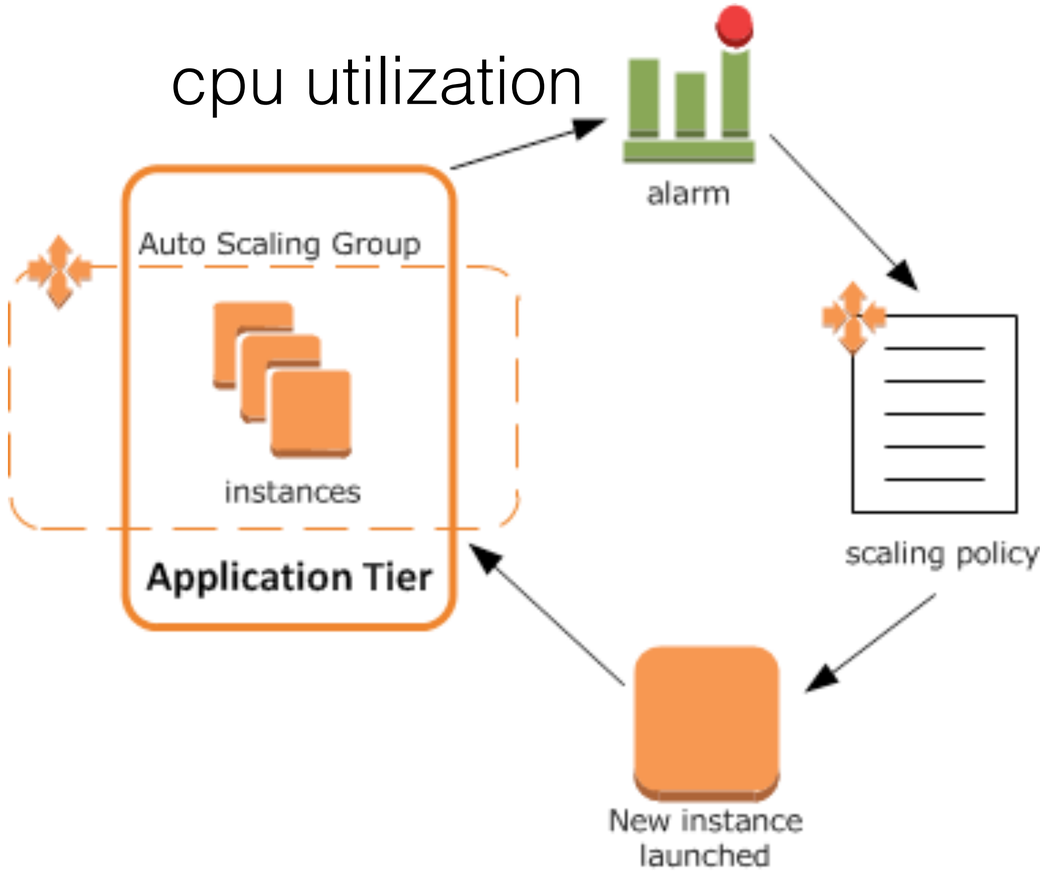
สำหรับการ monitor จะใช้ OS::Heat::ScalingPolicy เพื่อสร้าง scale up และ scale down

Auto Scaling



http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/Cooldown.html

Auto Scaling cooldown

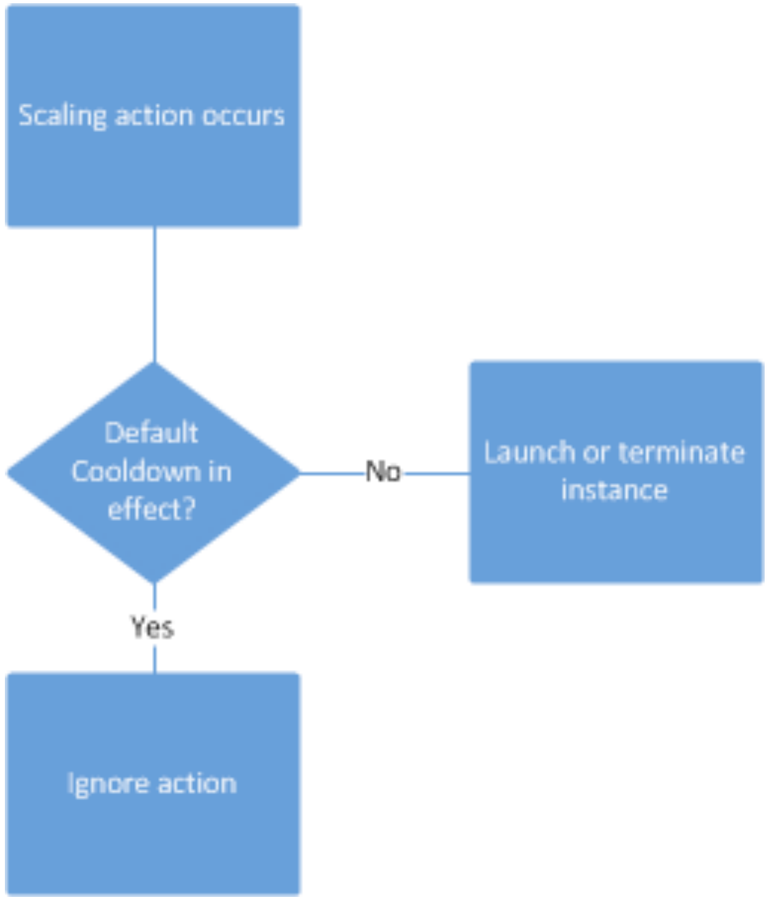



```
scale_up_policy:
  type: OS::Heat::ScalingPolicy
  properties:
    adjustment_type: change_in_capacity
    auto_scaling_group_id: {get_resource: asg}
    cooldown: 60
    scaling_adjustment: 1
scale_down_policy:
  type: OS::Heat::ScalingPolicy
  properties:
    adjustment_type: change_in_capacity
    auto_scaling_group_id: {get_resource: asg}
    cooldown: 60
    scaling_adjustment: '-1'
```

http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/Cooldown.html

CoolDown มีความสำคัญ

auto scaling cooldown คือ ช่วงเวลาหนึ่ง ที่ป้องกันไม่ ให้ launch หรือ terminate ในช่วงเวลาที่ระบุไว้ในค่าของ cooldown เมื่อหมดช่วงเวลาที่ระบบ ใน cooldown แล้วมีการสร้าง instance ใหม่



Demo

การทดสอบ4

สร้าง instance Autoscale multinode (แบบมี load balance)

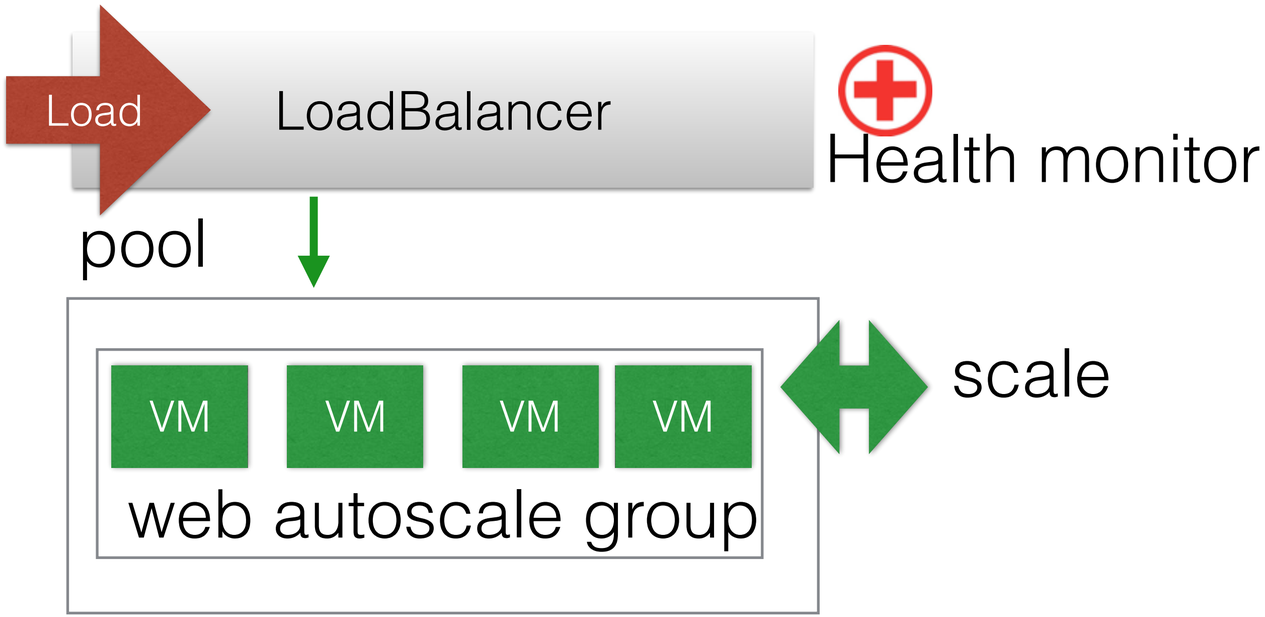
ต้องการทดสอบ autoscale โดยทำงานร่วมกับ Load Balancer As-a-Service (LBAAS) ซึ่ง backend technology ที่ใช้นั้นคือ ha-proxy ในแต่ละinstace ที่สร้างขึ้นจะเป็นสมาชิก VIP ของ pool ใน loadbalance และได้ทำการติดตั้ง webserver ร่วมกับ php เพื่อใช้ทดสอบด้วย phpให้แสดงผลค่าของ hostname ออกมาที่หน้าจอเท่านั้น

Load Balancer

สร้าง instance Autoscale multinode (แบบมี load balance)

ต้องการทดสอบ autoscale โดยทำงานร่วมกับ Load Balancer As-a-Service (LBAAS) ซึ่ง backend technology ที่ใช้นั้นคือ ha-proxy ในแต่ละinstace ที่สร้างขึ้นจะเป็นสมาชิก VIP ของ pool ใน loadbalance และได้ทำการติดตั้ง webserver ร่วมกับ php เพื่อใช้ทดสอบด้วย phpให้แสดงผลค่าของ hostname ออกมาที่หน้าจอเท่านั้น

การทำงานร่วมกัน ระหว่าง Load Balance และ Autoscale Group



- สร้าง Load balancer (pool)
- AutoscaleGroup อยู่ภายใน pool
- สร้าง VM ให้อยู่ใน AutoscaleGroup
- Policy Scale up/down
- CPU Alarm

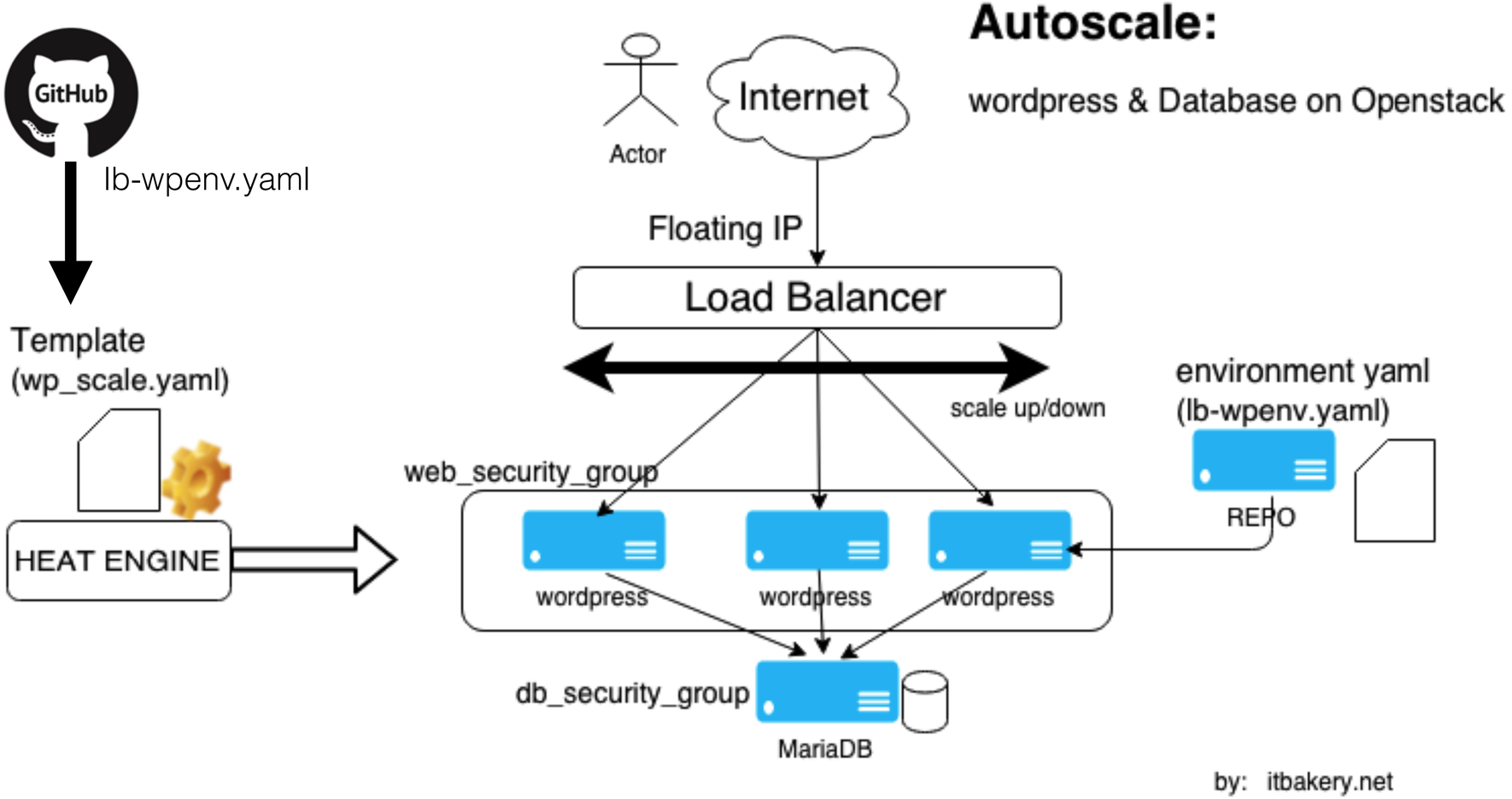
Demo

การทดสอบ5

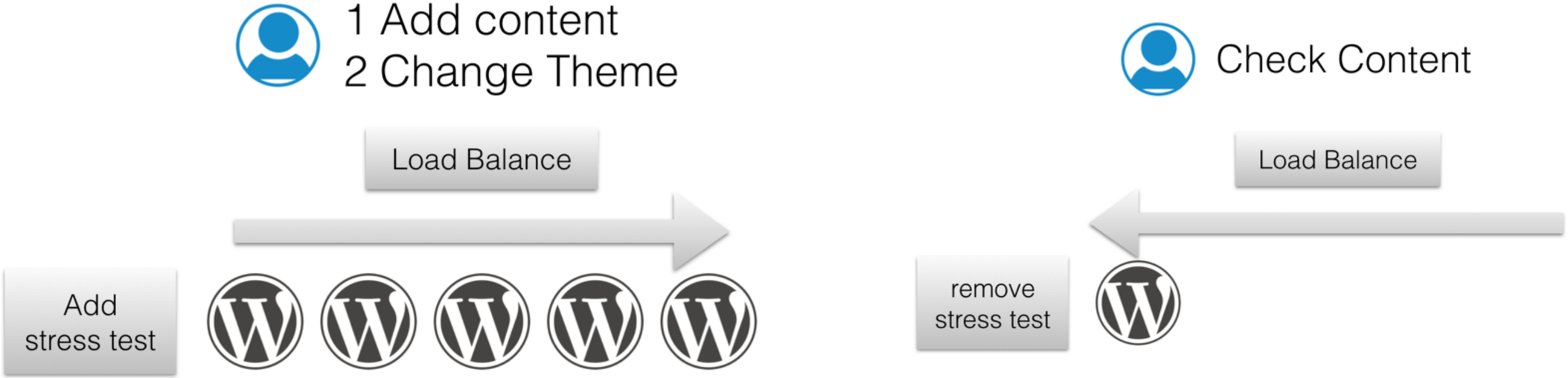
การสร้าง Auto scale สำหรับ Wordpress เพื่อให้สามารถใช้งานในส่วนของ Cloud Application

ต้องการทดสอบการสร้าง wordpress ให้สามารถใช้งาน autoscale โดยการทดสอบจะทำการแยก ส่วนของ Web Application Tier ออกจากส่วนของDatabase Tier ออกจากกัน โดย instance ของ Wordpress จะอยู่ในส่วนของสมาชิกของ Loadbalance เช่นเดียวกับการทดสอบ 4

โครงสร้างของ Heat Template



Wordpress Autoscale Test



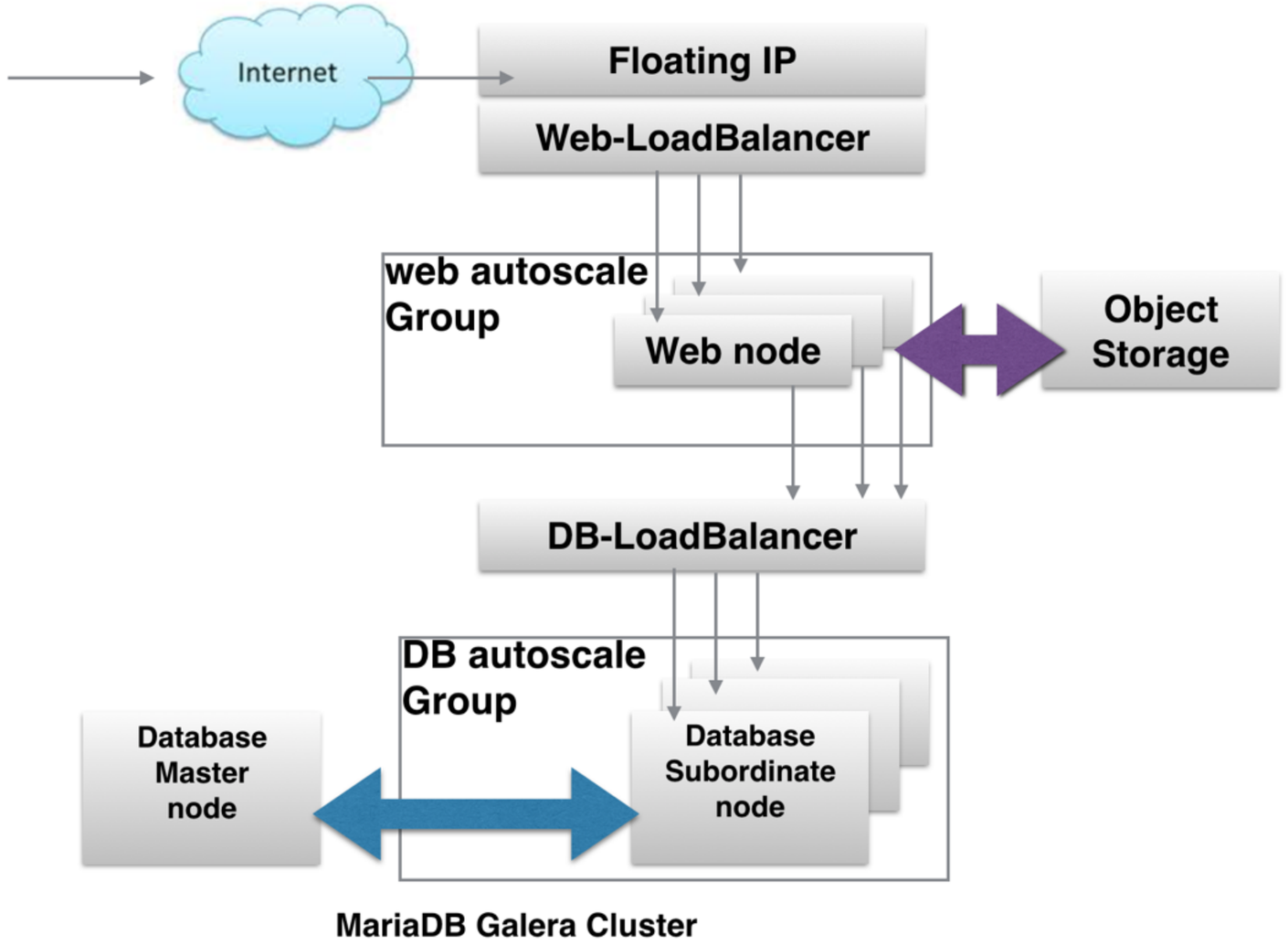
การทดสอบ6

การสร้าง Auto scale สำหรับ Wordpress ร่วมกับ Mysql Galera Cluster

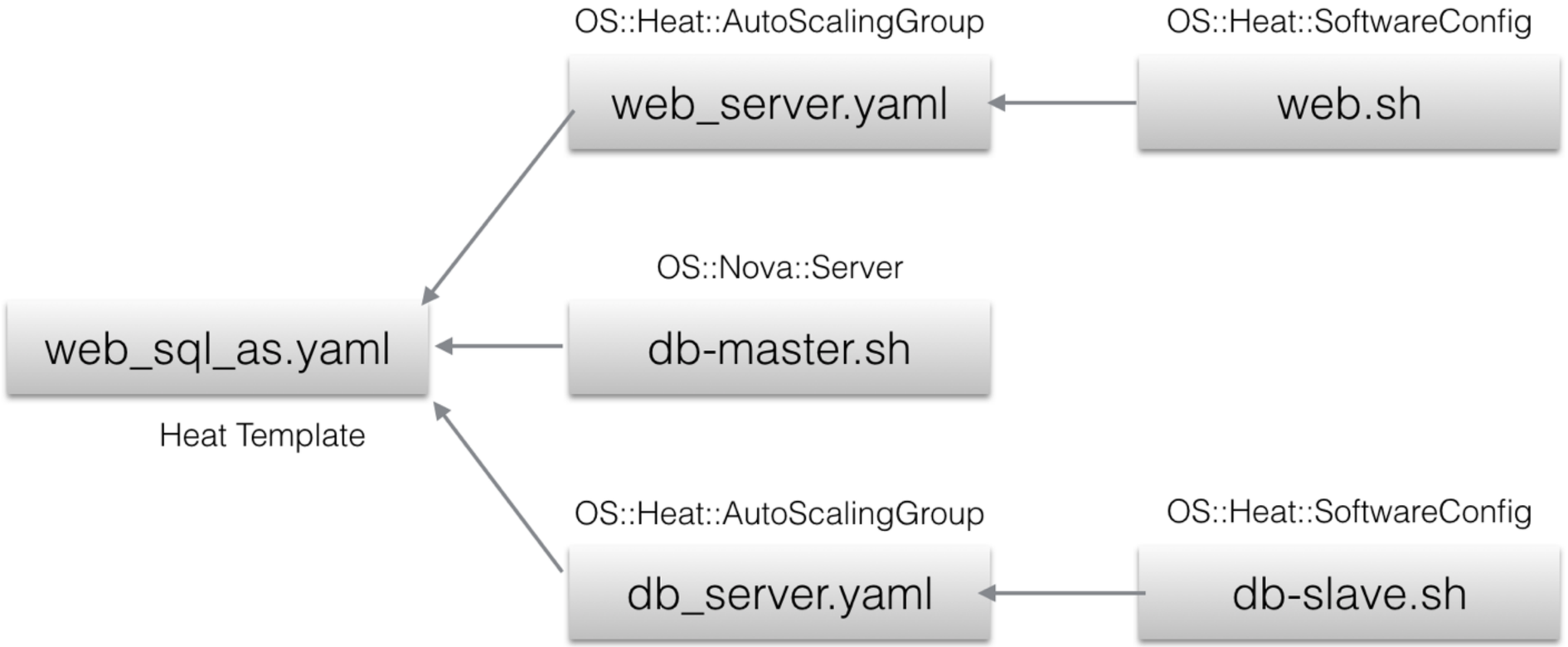
ต้องการทดสอบการสร้าง wordpress โดยแบ่งออกเป็นส่วนของ WebAutoscale และส่วนของ Mysql Galera Cluster (DB Autoscale Group) และมีการ sync มายัง swift storage เพื่อเก็บ staticfile ระหว่างการ scale

```
$ git clone git@github.com:thaiopen/privatecloud.git
$ cd privatecloud/workshop6
$ ls
db-master.sh   db_server.yaml   web.sh    web_sql_as.yaml  db-slave.sh                  example-deploy-
sequence.yaml web_server.yaml
```

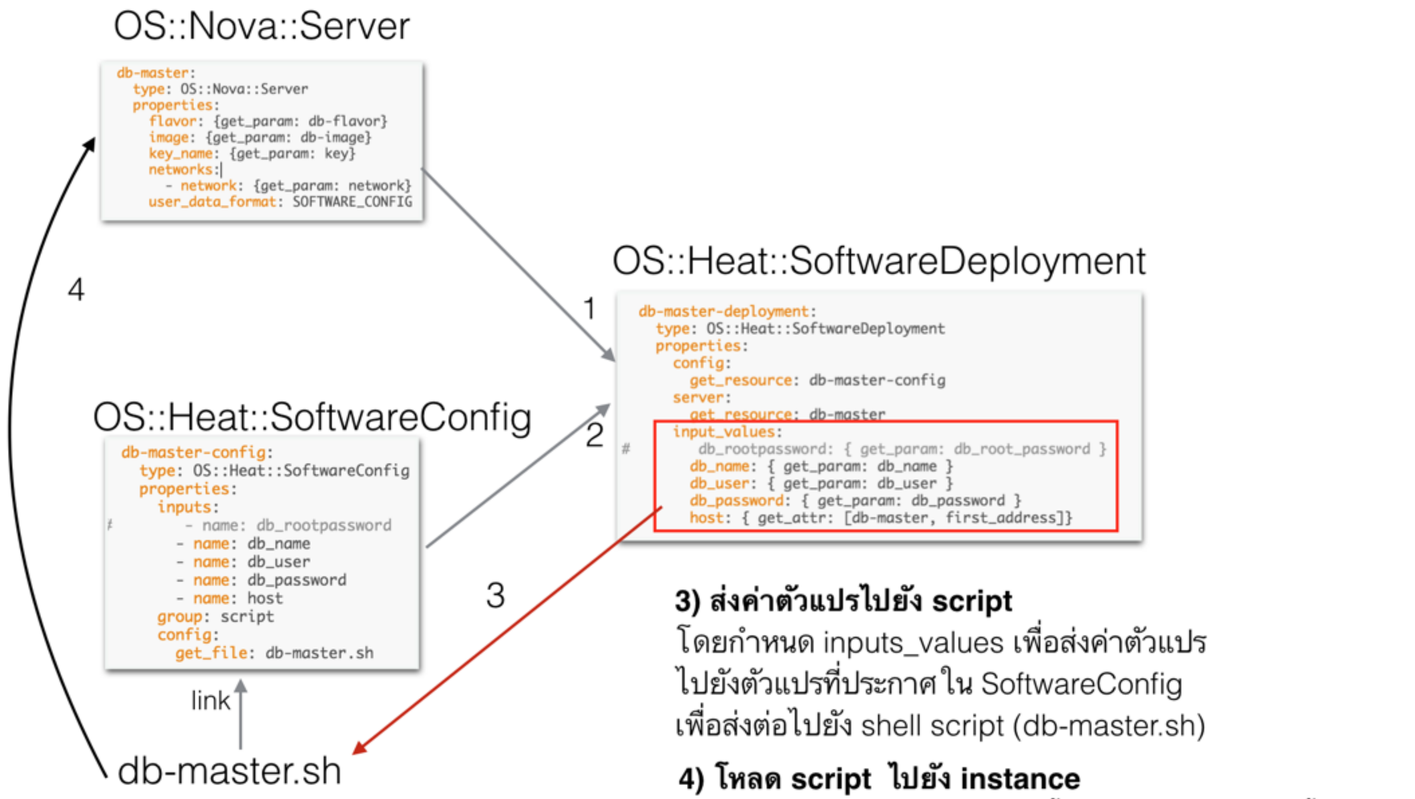
โครงสร้างของ Heat Template



File ที่ใช้


OS::Heat::AutoScalingGroup
web_server.yaml
OS::Heat::SoftwareConfig
web.sh
OS::Nova::Server
web_sql_as.yaml
db-master.sh
Heat Template
OS::Heat::AutoScalingGroup
db_server.yaml
OS::Heat::SoftwareConfig
db-slave.sh

OS::Heat::SoftwareDeployment



3) ส่งค่าตัวแปรไปยัง script
โดยกำหนด inputs_values เพื่อส่งค่าตัวแปรไปยังตัวแปรที่ประกาศใน SoftwareConfig เพื่อส่งต่อไปยัง shell script (db-master.sh)

4) โหลด script ไปยัง instance
จะโหลด db-master.sh ไปติดตั้ง ด้วย Heat API ดังนั้น จะต้องมีการเตรียม cloud image ให้รองรับ heat (ไม่สามารถใช้แบบ generic ได้)

Heat image

```
#!/bin/bash
yum install qemu-img
yum install python-pip git
pip install git+git://git.openstack.org/openstack/dib-utils.git
git clone https://git.openstack.org/openstack/diskimage-builder
git clone https://git.openstack.org/openstack/dib-utils.git

export ELEMENTS_PATH=tripleo-image-elements/elements:heat-templates/hot/software-config/elements
export BASE_ELEMENTS="centos7 selinux-permissive"
export AGENT_ELEMENTS="os-collect-config os-refresh-config os-apply-config"
export DEPLOYMENT_BASE_ELEMENTS="heat-config heat-config-script"

export IMAGE_NAME=software-deployment-image
diskimage-builder/bin/disk-image-create vm $BASE_ELEMENTS $AGENT_ELEMENTS
$DEPLOYMENT_BASE_ELEMENTS $DEPLOYMENT_TOOL -o $IMAGE_NAME.qcow2

glance image-create --name CentOS-7-x86_64-heat --disk-format qcow2 --container-format bare --
file software-deployment-image.qcow2
```

Demo

Thank you

sawangpong@itbakery.net